CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Software-Engineered Human-AI Annotation Systems for Safety-Critical Perception Pipelines

Master's Thesis in Computer science and engineering

Qiqi Lin & Sezin Kokum Mutludogan

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's Thesis 2026

Software-Engineered Human-AI Annotation Systems for Safety-Critical Perception Pipelines

Qiqi Lin & Sezin Kokum Mutludogan

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Software-Engineered Human-AI Annotation Systems for Safety-Critical Perception Pipelines
Qiqi Lin & Sezin Kokum Mutludogan

Supervisor: Malsha Ashani Mahawatta Dona, Department of Computer Science and Engineering
Industry Supervisors: Ricardo Silva, Owen Liu, Volvo Autonomous Solutions
Examiner: Gregory Gay, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Qiqi Lin & Sezin Kokum Mutludogan
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

AI-assisted annotation systems are increasingly used to support perception data generation for autonomous and ML-enabled systems. While modern AI models can improve annotation efficiency and detection coverage, integrating multiple AI components into human-in-the-loop workflows also introduces architectural trade-offs related to functional correctness, performance efficiency, maintainability, human cognitive workload, and usability. Existing research often focuses primarily on model-level accuracy, while the broader impact of workflow orchestration and human-AI interaction remains less explored.

This thesis investigates how different architectural design patterns influence the behavior and quality attributes of AI-assisted annotation systems. Building upon architectural design patterns for ML-enabled systems, six human-AI annotation variants were designed and evaluated, including sequential, parallel, single-model, and VLM-assisted workflows. A modular annotation platform integrating YOLO11n, OWL-ViT, VLM-based prompt generation, and human validation was developed following a design science research approach.

The evaluation combined controlled annotation experiments and a complementary user study. Functional correctness, maintainability, performance efficiency, annotation consistency, human intervention effort, and cognitive workload were analyzed across architectural variants using both quantitative metrics and qualitative feedback from participants.

The results demonstrated that no single architectural variant optimized all evaluated quality attributes simultaneously. The Parallel variant improved detection diversity and recall but increased cognitive workload and interaction complexity. The Sequential variant reduced cognitive effort and improved usability through simplified validation-oriented interaction, although it introduced stronger dependency propagation and orchestration complexity. The findings further showed that increasing architectural complexity or adding additional AI components does not automatically improve overall effectiveness.

Overall, the thesis highlights that evaluating ML-enabled human-AI systems requires considering not only predictive performance, but also maintainability, interaction design, human oversight, and long-term sustainability. The findings provide practical architectural recommendations for future AI-assisted perception annotation systems.

Keywords: Human-AI Collaboration, AI-Assisted Annotation, ML-Enabled Systems, Architectural Design Patterns, Human-in-the-Loop (HITL), Autonomous Systems, Maintainability, Cognitive Workload, Quality Attributes

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Malsha Ashani Mahawatta Dona, for her continuous guidance, valuable feedback, and support throughout this thesis project. Her insightful comments, constructive discussions, and academic guidance significantly contributed to the development and improvement of this work.

We would also like to sincerely thank our supervisors, Ricardo Silva and Owen Liu, at Volvo Autonomous Solutions for their continuous support, valuable guidance, and constructive collaboration throughout the thesis project. Their practical insights, technical expertise, and willingness to support us during every stage of the study significantly contributed to both the research process and the overall experience of conducting this work in an industrial environment. In addition, we are grateful to Volvo Autonomous Solutions for providing the industrial context, datasets, and practical insights that supported this research.

We also would like to thank our examiner, Gregory Gay at the Department of Computer Science and Engineering, for the valuable comments and recommendations provided during the thesis process.

We further thank all participants who contributed to the user study and shared their valuable experiences and feedback during the evaluation process.

Qiqi:
I would like to express my deepest gratitude to my girlfriend, Yuting Chen, for her endless encouragement, understanding, and patience. She consistently helped me see things clearly when I was lost, and understood who I am even better than I do. I also want to express my sincere thanks to my thesis partner, Sezin, for her dedicated efforts and contribution throughout this work. Lastly, I would like to thank myself for persevering through the difficult moments and for staying focused on the goal.

Sezin:
I would like to express my sincere gratitude to my beloved husband, Korhan Mutludogan, for his continuous encouragement, patience, and support throughout this thesis journey. I would also like to thank my mother, Nur Kokum, my father, Goksel Kokum, and all my family members for their unwavering support, understanding, and belief in me throughout my studies. I would also like to thank my thesis partner, Qiqi, for the effort and commitment invested in this work.

Qiqi Lin & Sezin Kokum Mutludogan, Gothenburg, June 2026

Contents

List of Figures

List of Tables

1

Introduction

Machine learning has become an essential component of modern software systems across a wide range of industries, such as healthcare, automotive, and industrial automation [1, 2, 3, 4]. These systems increasingly rely on data-driven approaches to perform tasks such as object detection, classification, and decision-making. In many of these applications, machine learning models' performance depends mostly on the presence of large-scale datasets that accurately represent real-world scenarios [5]. As a result, the process of preparing and structuring data is a critical step in developing AI-enabled systems, and labeled data has a critical role since it provides the necessary information for models to learn about the relevant characteristics and relationships between inputs and outputs.

Data annotation is the process of assigning labels to raw data, such as images, and describing the objects or features present within them [6]. Labeling raw data enables machine learning models to be efficiently trained by creating the required annotated data. Depending on the level of automation, data annotation may be divided into three approaches. One approach is manual annotation, where human annotators label the raw data directly [6]. Another approach is fully automated annotation, which relies on machine learning models to generate labels without human intervention [7]. Finally, semi-automatic annotation that combines machine-generated predictions with human correction and validation [3, 8]. Between these approaches, semi-automatic human-AI annotation systems are commonly used in modern machine learning pipelines, as they integrate automated processing and human input within the annotation workflow.

In addition to these data-centric challenges, the design of ML-enabled systems introduces important software engineering challenges. One of the most important architectural design decisions is how multiple AI models are controlled and integrated with human input; such decisions can influence the quality attributes of the complete system [9].

Prior research indicates that design patterns play a key role in managing complexity, modularity, and scalability for ML-enabled systems [10, 11]. Moreover, design patterns also introduces trade-offs between qualities such as accuracy, maintainability, and performance efficiency [12, 9]. Understanding these trade-offs is important for both researchers and practitioners, especially in systems where both automated

components and human interaction must be coordinated effectively.

In this context, this thesis is conducted in collaboration with Volvo Autonomous Solutions (VAS), in relation to the perception system of autonomous vehicles for use in mining/quarry operations. In safety-critical domains, the quality of annotated data directly affects how well and reliably downstream perception models work. Overall, our study examines how architectural design patterns and quality attribute trade-offs can be applied and evaluated in ML-enabled systems through the lens of human-AI collaborative annotation workflows.

In this thesis, we use the term architectural design pattern to refer to different ways of organizing and orchestrating multiple AI models and human interaction within an annotation workflow. In particular, each pattern corresponds to a specific configuration of model execution (e.g., sequential or parallel) and informed by established AI architectural design patterns [12, 9], specifically Parallel Independent Models and Sequentially Dependent Models.

1.1 Problem Description

As the demand for data annotation continues to grow, purely manual approaches are becoming too expensive, while fully automated AI-based annotation often falls short in terms of accuracy [13]. This has made Human-in-the-loop (HITL) [14] systems a common compromise.

Current research on semi-automated annotation often focuses on how AI and human efforts can be combined to balance accuracy and efficiency [15]. However, most of this work is limited to single-model scenarios. Questions about multi-model settings, such as how the order of model invocation or the choice between parallel and sequential execution affects system quality, remain largely under-explored from a software architecture perspective.

Although new architectural design patterns for AI-based systems have emerged [12, 9], they rarely consider how human involvement fits into the system. Decisions such as how humans interact with model outputs can directly impact annotation quality, system responsiveness, and maintenance costs, yet these aspects are not extensively addressed in existing patterns. Recent work has begun to examine how HITL integration affects architectural quality attributes [16], highlighting that human factors such as workload and timing are often overlooked. However, as the authors acknowledge, such studies are not situated in specific real-world application contexts.

Together, prior work lacks a unified understanding of how multi-model architecture design and HITL integration interact within real-world annotation system architectures. This leaves developers to rely on intuition or past experience when designing HITL annotation systems, making it difficult to reason about trade-offs or anticipate how design decisions influence the degree of human involvement required.

To address this gap, an empirical study is needed because the impact of architectural design patterns on system-level quality attributes and human intervention cannot be fully understood without observing their behavior in a controlled experimental setting. This enables us to investigate how different patterns affect human involvement and the key quality attributes, and to analyze the trade-offs among them.

1.2 Purpose of the Study

The purpose of this study is to bridge a critical gap in the current literature: while research on semi-automated annotation systems has largely overlooked architectural design patterns for multi-model scenarios [15], and recent work on AI system patterns provides guidance for integrating multiple AI models [12, 9], there is a lack of studies that systematically evaluate these patterns in human-in-the-loop annotation workflows with respect to key quality attributes and human involvement.

Our goal is to improve system-level quality for semi-automated annotation workflows at the architectural design level, focusing on how system components are structured and interact rather than optimization of individual algorithms. Through an empirical comparison of six architectural variants (will be explained in Section 4.3), we investigate how those patterns influence critical system quality attributes, specifically functional correctness, maintainability, and performance efficiency, and how these design choices affect the amount and timing of human labor in the loop.

By revealing the complex trade-offs between automation and human intervention, this study provides actionable insights and design guidance to help developers build more reliable, scalable, and cost-effective human-AI annotation infrastructures. Moreover, the results of the study are expected to benefit Volvo Autonomous Solutions by reducing the effort for annotation, improving consistency, and offering a structured and maintainable method for integrating AI-assisted annotation into development processes. The findings may also provide guidance for other organizations developing human-AI collaborative annotation systems and ML-enabled perception workflows that face similar challenges related to automation, maintainability, and human oversight.

1.2.1 Research Questions

This study investigates how different architectural design patterns affect the behavior and effectiveness of human-AI annotation systems implemented in perception pipelines. Human annotators and AI models collaborate to produce labeled datasets, which are later used to train perception models. While AI models help automate such systems and reduce manual human workload, the strategy of organizing these components within the system architecture can influence several system quality attributes, including system accuracy, maintainability, and the level of human intervention.

Although semi-annotation systems are widely used, there is limited understanding

of how architectural design decisions affect these qualities in practice. Exploring the differences between architectural designs is important to implement systems that balance automation and human oversight wisely. For this purpose, our research examines different architectural design patterns for integrating human validation and AI models in an annotation system. Therefore, the study is structured around three research questions.

RQ1: How do different architectural design patterns of human-AI annotation systems affect key system quality attributes including functional correctness, maintainability, and performance efficiency?

Designing effective annotation systems requires considering multiple aspects of the architecture. We quantify the baseline performance of each pattern across key quality attributes. Functional correctness is evaluated to ensure the correctness of AI-generated annotations. Maintainability captures how easily AI components can be updated or replaced. Performance efficiency considers system throughput and resource usage. By analyzing these dimensions together, we establish a baseline that shows how structural choices affect the core performance of annotation systems. These performance profiles provide the empirical foundation for examining trade-offs and understanding human-AI collaboration.

RQ2: What trade-offs exist between these quality attributes across different architectural design patterns?

Improving one quality attribute may negatively or positively affect another. This question aims to investigate trade-offs that arise between quality attributes when different architectural design patterns are implemented. By analyzing the relations between these quality attributes, this study will explore how architectural choices affect the balance of automation efficiency, annotation reliability, and maintenance cost. Identifying these trade-offs can inform developers about architectural decisions during the design process of annotation systems that contain multiple AI components and human validation.

RQ3: How do different architectural design patterns affect the level of human intervention required during annotation workflows?

Although AI models help automate annotation processes, human validation is essential in annotation workflows, especially in safety-critical perception systems. This question focuses on understanding how different architectural design patterns affect human intervention in data annotation tasks. By examining various human intervention metrics for each variant, we aim to determine which architectural choices reduce manual effort while still ensuring high-quality results.

1.3 Significance of the Study

This study has the potential to refine and extend existing knowledge on how architectural design patterns influence AI-enabled systems that integrate machine learning

models and human interaction. In addition, the research addresses the challenges of organizing heterogeneous AI components, such as supervised detection models (e.g., YOLO-based [17]), vision-language models (VLMs) (e.g., CLIP [18]) used for zero-shot classification, as modular elements within a coherent software architecture, particularly in human-in-the-loop workflows. This contributes to knowledge on how to build maintainable and efficient AI-integrated software systems within the SE4AI domain.

In terms of methodological contribution, the study demonstrates how design science research can be applied to the development of a prototype system with controlled experimental evaluation of architectural design patterns in systems that integrate AI components. To conduct development and evaluation focused design science research, multiple architectural configurations are implemented and evaluated under consistent conditions, enabling a systematic comparison of their impact on quality attributes such as accuracy, maintainability, performance efficiency, and human intervention effort.

Furthermore, the study contributes evidence on how these qualities are influenced not only individually but also in relation to each other by systematically comparing multiple architectural design patterns under controlled conditions. Understanding how AI-enabled systems behave at the architectural level is particularly strengthened by an investigation of trade-offs between quality attributes, such as the balance between automation and human effort, which contributes to the broader discourse in software engineering on designing systems that integrate AI components in a structured and measurable way.

Moreover, the results of the study provide insights for practitioners when designing AI-enabled systems, particularly in the perception domain and similar data-driven pipelines, regarding how architectural decisions impact system functional correctness, performance, human intervention, and maintainability. Applying these insights may support the development of more efficient and maintainable systems and data processing pipelines in safety-critical environments where data annotation quality is necessary.

Overall, the study provides empirical insights into how architectural design patterns affect system-level properties in AI-integrated systems for researchers and practitioners. By concentrating on measurable system quality attributes and the inherent trade-offs, the study improves the existing comprehension of human-AI collaboration and extends ongoing research in Software Engineering.

1.4 Thesis Outline

Our thesis is structured into the following chapters.

- Chapter 1 - Introduction: This chapter introduces the research problem, outlines the study's purpose, and presents the research questions, contributions and significance of the study.

- Chapter 2 - Background: The background chapter provides an overview of human-in-the-loop annotation systems, discusses the challenges of multi-model AI integration, and introduces the quality attributes used for evaluation.

- Chapter 3 - Related Work: In this chapter, we review existing literature on annotation systems, AI-based perception approaches in the automotive domain, and software architecture design patterns for ML-enabled systems, highlighting the research gap addressed in our study.

- Chapter 4 - Methods: This chapter describes the research methodology based on design science, presents the developed human-AI annotation system artefact, and explains the architectural design patterns, implementation details, and experimental setup and data collection.

- Chapter 5 - Results: This chapter presents the results of the empirical evaluation, including analysis of system quality attributes, their trade-offs, and human intervention across different architectural variants.

- Chapter 6 - Discussion: In this chapter, we discuss the findings in relation to the research questions, analyze trade-offs between quality attributes, and reflect on the implications for system design.

- Chapter 7 - Conclusion: In the last chapter, we conclude the study by summarizing the contributions, discussing limitations, and outlining directions for future work.

2

Background

This chapter establishes the conceptual and technical foundation for the study. It begins by examining the limitations of existing annotation approaches and the motivation for HITL systems, then identifies the gap in understanding how architectural design patterns influence such systems. Finally, it introduces the quality challenges inherent to multi-model HITL annotation and defines the evaluation framework used throughout this work.

2.1 Annotation Strategies and the Need for Human Involvement

Annotation strategies can generally be grouped into three categories: manual annotation, fully automatic annotation, and semi-automatic annotation [19]. Manual annotation relies entirely on human labeling, which can achieve high accuracy but is costly, time-consuming, and subjective [15, 19]. Fully automatic annotation uses AI models without human intervention; while efficient, it often lacks sufficient reliability, particularly for edge cases and ambiguous inputs [19]. These limitations have motivated the development of semi-automatic annotation systems, which combine automated pre-labeling with selective human oversight to balance efficiency and output quality [20, 17].

A key mechanism enabling semi-automatic annotation is human-in-the-loop (HITL), in our context often called human-in-the-loop machine learning (HITL-ML) [21], which emphasizes collaborative interactions between humans and algorithms to enhance both model accuracy and human efficiency [22].

In practice, human involvement in HITL annotation workflows typically takes one or more of the following forms [20, 15]:

- Validation: Reviewing and confirming AI-generated labels that meet a confidence threshold.
- Correction and Refinement: Adjusting labels, boundaries, or attributes where model output is uncertain or incorrect.

- Quality Control: Sampling completed annotations to ensure consistency and accuracy across the dataset.

This integration introduces complexity beyond model performance alone; factors such as human workload, response timing, and the risk of human error become intrinsic to system behavior [16].

These challenges are particularly prominent in safety-critical fields because labeling errors can directly affect model training and ultimately compromise system safety. In such contexts, HITL is not merely an efficiency strategy but a necessary mechanism for reducing systematic errors and maintaining reliability.

2.2 The Need to Compare Architecture Design Patterns for Multi-Model HITL Annotation

Current annotation systems employ different AI paradigms, primarily supervised models (e.g., YOLO-based [17]) and zero-shot/open-vocabulary detection models (e.g., CLIP [18], OWL-ViT [23]). Supervised models provide high accuracy for predefined known classes but lack flexibility in dynamic environments due to their limited vocabulary. However, zero-shot models can adapt to unseen categories [24, 25] but sacrifice consistency and accuracy. Integrating these paradigms as complementary components into the HITL system could balance reliability and flexibility.

However, existing systems often combine models in an ad-hoc manner, resulting in a lack of systematic understanding of how architectural design patterns influence system-level quality attributes and the required human effort. This challenge motivates a structured comparison of architecture design patterns for multi-model HITL annotation systems.

2.3 Quality Challenges and Evaluation Framework

2.3.1 Quality Challenges in Multi-Model HITL Annotation

Introducing human intervention into automated annotation workflows adds complexity beyond model performance. HITL systems introduce additional dimensions: human workload, timing [16], and the possibility of human error or inconsistency [26, 27].

Moreover, adding multiple AI models increases this complexity further. Models may produce conflicting outputs or have different execution times. Orchestrating them, whether sequential, parallel, or hybrid, introduces coupling and potential failure. For example, if a model is replaced or retrained, and the architecture is tightly coupled, the impact on the overall workflow may be difficult to predict. Similarly, coordinating multiple models also complicates latency, resource usage, and error diagnosis.

Thus, evaluating multi-model HITL annotation systems solely on annotation accuracy is insufficient. The architectural decisions regarding how to orchestrate multiple models and how humans interact with them directly affect system quality and the effectiveness of human-AI collaboration.

2.3.2 The Shift in Quality Models

Traditional software quality models like ISO/IEC 25010 [28] assume deterministic systems. AI-based systems, however, introduce uncertainty due to their reliance on data and probabilistic outputs. To address this, ISO/IEC 25059 [29] extends the standard with sub-characteristics specifically for AI systems. In this study, we draw on ISO/IEC 25059 to inform our selection and definition of quality attributes.

2.3.3 Selected Quality Attributes

Based on the challenges above, this study focuses on three core quality attributes. Together, they capture the essential trade-off between output quality, long-term adaptability, and operational sustainability.

Functional Correctness It refers to the degree to which a system provides correct results with the required precision [29]. As ISO/IEC 25059 notes, machine learning models do not guarantee correctness across all circumstances, making careful measurement essential. In HITL annotation, this directly reflects the final annotation accuracy.

Maintainability We focus on two sub-characteristics from ISO/IEC 25059: modularity (the degree to which a system is composed of discrete components, minimizing the impact of changes) and analysability (how effectively the impact of changes can be assessed or failures diagnosed). In a HITL context, maintainability is crucial because models may need to be updated or replaced frequently, the logic controlling human intervention (e.g., when to trigger a review, how to present results) must remain adaptable, and failures must be quickly traceable to the model, the human workflow, or their integration.

Performance Efficiency Performance relative to resource usage under stated conditions [29], including time behavior and resource utilization. In HITL annotation systems, this directly affects user experience and operational cost. If system-level inference takes too long, human annotators will be idle. Furthermore, assessing resource utilization helps determine whether running multi-models increase additional operation costs.

By evaluating them together, we aim to understand how architectural choices affect the balance between system effectiveness, operational cost, and sustainability.

3

Related Work

In this chapter, we review existing research about annotation systems, AI-based perception approaches, and software architectural design patterns for multi-model HITL annotation. This chapter also discusses previous research on quality attributes and tradeoffs of AI systems by highlighting limitations and the research gap addressed in this study.

3.1 Human-in-the-Loop Annotation Approaches

Existing research on semi-automated annotation has explored various strategies to optimize annotation efficiency with accuracy. Bonet et al. [20] propose a framework that leverages human feedback to iteratively retrain the auto-annotator, achieving a trade-off between annotation time reduction and accuracy. Similarly, Gamalii et al. [15] demonstrate that incorporating human feedback into system training, combined with thoughtful system design, can significantly reduce human intervention compared to fully manual annotation. Another recent study [30] has also examined the use of AI agents and automated pipelines for annotation tasks to assist annotation, where they often emphasize model capabilities rather than architectural design patterns and system-level evaluation.

Most of these studies assume a single fixed architecture (e.g., a single model with a human review loop). The architectural flexibility of systems that integrate multiple AI models, and how such flexibility affects system behavior and human effort, remains largely unexplored. This gap motivates our investigation into the architectural design pattern of multi-model HITL annotation systems.

3.2 Architectural Design Patterns for Multi-Model HITL Annotation

As AI components become increasingly integrated into software systems, researchers have begun to identify and formalize architectural design patterns specific to AI-based systems. Heiland et al. [12] provide an overview of design patterns for AI-based systems, including both novel patterns and adaptations of traditional ones. Among these, they introduce two patterns relevant to multi-model orchestration:

the *Microservice Vertical Pattern* (sequential execution with dependencies) and the *Microservice Horizontal Pattern* (parallel execution).

Eriksson et al. [9] adopt these concepts, renaming them as *Sequentially Dependent Models* and *Parallel Independent Models*. Importantly, they also introduce *Validation Patterns* where human reviewers inspect AI outputs before passing them to downstream components, one of the few examples that explicitly incorporate human involvement into the architectural pattern.

However, the application of these patterns in HITL annotation systems remains largely underexplored. While the patterns themselves can accommodate human intervention, there appears to be limited work systematically comparing how different patterns (sequential, parallel, or hybrid) affect system behavior in a multi-model HITL annotation context, especially when combining supervised and zero-shot models.

This thesis builds upon AI-system design patterns proposed by Heiland et al. [12] and Eriksson and Olausson [9]. In this context, design patterns describe recurring architectural coordination structures for organizing AI components, information flow, and human validation stages within ML-enabled systems.

3.3 Quality Attribute Evaluation in Multi-Model HITL Annotation

As architectural design patterns for AI-based systems have emerged, researchers have also begun to focus on how to evaluate the quality attributes of such systems. However, a noticeable gap exists between generic architectural studies and the specific needs of annotation systems.

Limitations of Generic Architectural Studies: Heiland et al. [12] provide a comprehensive catalog of design patterns for AI-based systems, but do not systematically analyze how different patterns affect system quality attributes. Building on this foundation, Eriksson et al. [9] conduct a systematic study of trade-offs among multiple quality attributes (e.g., performance efficiency, maintainability) across different patterns. Their work represents an important step toward understanding how architectural choices influence software quality in AI-enabled systems.

However, this body of research has two key limitations that are relevant to our study.

First, existing studies are not situated in domain-specific annotation contexts. The performance and reliability of AI-based systems are heavily dependent on data characteristics [31]. In particular, annotation systems for specialized domains (e.g., mining and quarry environments) face unique data characteristics and quality requirements that are not addressed by generic pattern analyses.

Second, existing architectural studies do not fully account for human-in-the-loop

collaboration. While Eriksson et al. [9] touch upon human involvement through their Validation Patterns, where human reviewers inspect AI outputs before passing them to downstream components, their evaluation framework focuses solely on traditional software quality attributes and does not incorporate human intervention metrics such as correction effort. Complementing this view, Indykov et al. [16] examine how HITL is integrated into ML-enabled systems and compare quality attributes across architectural design pattern alternatives. Notably, they find that human factors such as workload and timing are frequently overlooked. However, as the authors acknowledge, their findings are not grounded in specific real-world application contexts. As noted in our first point, this lack of empirical evidence highlights the need for studies situated in concrete, specialized domains.

Limitations of Annotation System Research: Current research on annotation systems has evaluated performance primarily through data quality metrics. For fully automated annotation systems, evaluation typically focuses on precision, recall, and mAP [13]. For HITL annotation systems, studies such as Bonet et al. [20] and Gamalii et al. [15] extend evaluation to include human effort (e.g., annotation time reduction), but they rarely analyze software quality attributes trade-offs.

Addressing these gaps, our study presents a systematic comparison of architectural design patterns in a domain-specific HITL annotation setting. We evaluate functional correctness, maintainability, performance efficiency, and human intervention together, enabling a structured analysis of trade-offs that reflects the unique characteristics of human-AI collaborative annotation patterns.

3.4 Quality Attribute Trade-offs in AI/ML Enabled Systems

Several studies have investigated trade-offs between quality attributes in software systems, particularly in embedded and industrial contexts [32]. For example, Sas & Avgeriou [33] perform an exploratory case study to illustrate that run-time qualities (e.g., performance and reliability) tend to be prioritised over design-time qualities (e.g., maintainability) in practice; as a result, run-time qualities are usually given priority and design-time qualities (such as maintainability) are often compromised.

In the specific context of ML-enabled systems, systematic investigations of quality trade-offs have also been conducted. Through a systematic literature review of 206 primary sources, Indykov et al.[16] conducted a systematic literature review identifying common quality attributes and architectural tactics for ML-enabled systems, along with their potential quality trade-offs, providing a structured foundation for understanding how architectural decisions affect system quality.

The work by Oliveira et al. [34] provides evidence that there is a trade-off between performance and maintainability in the context of code (improving the performance of software may have the effect of increasing the complexity of the code).

Other studies demonstrate that a large number of implicit trade-offs exist in software development that are not systematically analyzed; typically developers make decisions based on their experience or a light decision-making process, rather than using formal methods [33].

In AI/ML-enabled automotive systems in safety-critical domains, these trade-offs are more critical. Perception systems must achieve high levels of functional correctness since failures may directly affect the safety of the system. Studies indicate that accurate perception is a key requirement for safe decision-making in autonomous vehicles. Moreover, detection or classification errors can directly lead to unsafe behavior [35, 36]. Therefore, metrics such as accuracy, recall, and precision are considered more critical than other attributes such as performance efficiency or development costs [37, 38].

In addition, these trade-offs manifest in multiple ways in data annotation for perception pipelines. For example, if we increase automation, we can improve throughput but reduce annotation accuracy. On the other hand, if we introduce human validation, we can improve correctness at the cost of increased time and effort. Likewise, combining multiple models (e.g., supervised and zero-shot models) can improve detection coverage, while it introduces some challenges in terms of latency, maintainability, and coordination of the system [39].

Although previous research has recognized that trade-offs exist, they usually analyze trade-offs from an overall architectural perspective or examine an isolated attribute of a system. While existing work has systematically catalogued trade-offs across ML-enabled systems through literature [16] and industrial case studies [40], neither addresses how specific architectural design patterns, combined with varying degrees of human intervention, affect trade-offs within multi-model annotation workflows. There is less research regarding how design decisions regarding architectural design patterns, and defining when they might involve human intervention, affect the trade-offs in practice.

This gap motivates our investigation to expand on previous research by examining how various architectural design patterns can impact multiple quality attributes at the same time and to examine the resulting trade-offs in a controlled, domain-specific setting in the automotive domain.

3.5 Maintainability in AI/ML-Enabled Systems

While maintainability and modularity have been extensively studied in traditional software systems, their role in ML-enabled systems introduces some unique challenges with respect to maintainability and modularity. For instance, most of the recent work on maintainability in ML-enabled systems has shown that maintainability problems arise from tight coupling between the various parts of an ML system (e.g., data acquisition pipelines, model components, system-level logic), making it difficult to maintain and update parts of an ML system in isolation [39].

Due to the additional complexity related to the data, model performance, and training pipelines associated with ML systems, these factors create new challenges when designing systems that are both flexible and maintainable [41]. Habibullah et al. indicate that their empirical findings show training pipelines and ML interfacing components tend to exhibit higher-than-average coupling, while data acquisition and ML interfacing components often show lower cohesion, further complicating modular system design in ML-enabled architectures [42].

Multiple metrics have been proposed in the literature for evaluating modularity, including complexity, coupling, and cohesion metrics at the class, package, and system levels [43]. While these metrics primarily quantify the attributes of (code-level) structure and do provide some level of quantification, they do not in themselves capture the influence of higher-level architectural decisions related to how the extension of an ML-enabled software system's functionality is handled.

Recent studies on the relationship between architectural styles and modularity and evolution in ML-enabled architectures provide an avenue for understanding better how architectural style impacts modularity. In particular, comparing architectural styles has shown that greater separation of concerns generally results in poorer coupling or increases in complexity, thereby exposing the inherent architectural trade-offs associated with the design of ML systems [44].

Overall, there is limited empirical understanding of how architectural design pattern decisions affect maintainability and modularity in systems that integrate multiple AI models and human-in-the-loop processes. Therefore, our study aims to address the gap by investigating architectural design and component independence in human-AI annotation workflows.

3.6 AI-Based Perception Approaches in Automotive Systems

Supervised object detection models are commonly used because they provide fast and accurate detection of predefined object categories [45, 35]. In addition, these models are particularly effective in safety-critical perception tasks, where real-time performance and localization accuracy are necessary.

More recently, VLMs have been introduced into automotive systems to enable higher-level semantic understanding of scenes. VLMs can generate contextual descriptions, reason about complex environments, and support downstream perception tasks [46, 47]. Furthermore, in the autonomous driving context, VLMs have been used to improve scene understanding and also achieve robustness in complex scenarios, where VLMs assist perception modules through natural language reasoning [48]. They incorporate semantic reasoning capabilities that can complement existing perception pipelines.

Despite the growing adoption of these approaches in automotive perception, there

is limited research on how to effectively combine these AI-based approaches within human-in-the-loop annotation systems from a software architectural design pattern perspective, considering system-level quality attributes and human intervention effort.

3.7 Research Gap

The reviewed literature highlights several important gaps related to human-AI collaborative annotation systems and ML-enabled architectures.

Existing research on human-in-the-loop annotation systems primarily focuses on improving annotation efficiency and model performance, while largely assuming fixed single-model architectures. There is limited investigation into how different multi-model architectural design patterns influence workflow behavior, annotation quality, and human intervention effort within annotation systems.

Research on architectural design patterns for AI/ML-enabled systems has introduced sequential, parallel, and validation-oriented coordination structures. However, these studies mainly discuss architectural concepts and generic quality trade-offs without systematically evaluating their behavior in domain-specific HITL annotation workflows that combine supervised perception, zero-shot models, semantic reasoning, and human validation.

Previous work on quality attributes, maintainability, and trade-offs in ML-enabled systems further demonstrates that architectural decisions influence performance efficiency, maintainability, coupling, and complexity. Nevertheless, there is limited empirical understanding of how architectural design patterns simultaneously affect functional correctness, maintainability, cognitive workload, human oversight, and workflow coordination within practical multi-model annotation pipelines.

In addition, although AI-based perception approaches such as supervised object detection and vision-language models are increasingly used in automotive systems, there is limited research examining how these approaches can be effectively integrated and orchestrated within human-AI annotation architectures from a system-level design perspective.

To address these gaps, this thesis presents a systematic empirical investigation of architectural design patterns in domain-specific human-AI collaborative annotation workflows by combining quantitative system-level evaluation, maintainability analysis, and qualitative user-centered assessment.

4

Methods

In this chapter, we presented our research methodology, design science approach with our research plan. Moreover, the system architecture, architectural design patterns, and artefact implementation were further presented in this chapter. We also discussed the experimental setup, data collection, and evaluation strategy used in the study. Finally, we explained our user study design and its scope.

4.1 Design Science Research

Our study employed a design science research approach, based on the methodological framework proposed by Johannesson and Perjons [49]. As shown in Figure 4.1, the framework includes five main activities that range from problem investigation, definition of requirements, design and development of the artefact, demonstration, and evaluation of the artefact.

According to the framework, design science research can focus on different subsets of these activities [49]. Our study was characterized as a development and evaluation focused design science research project as it investigates architectural design patterns and the development of human-AI annotation systems. In this context, our artefact is a prototype annotation system that integrates AI models and human validation and interaction components within a modular architecture.

In this study, the human AI-annotation system was implemented and evaluated with a dataset collected in the mining and quarry domain, including objects such as vehicles, machinery, and infrastructure elements. Different architectural design patterns were used to generate and refine annotations, allowing the system to be exercised under realistic conditions and providing an initial indication of how these architectural design patterns function in the autonomous systems domain.

Our thesis focused on the design and execution of alternative architectural design patterns, building on an existing understanding of semi-automated annotation systems. These configurations represented different ways of integrating supervised object detection models, zero-shot models, VLMs, and human validation within the annotation workflow.

The developed human-AI annotation system served as a research artefact and our

study combined artefact development with empirical evaluation and aimed to generate design recommendations about how architectural decisions affect the behavior of human-AI annotation systems. We did controlled experiments to compare the influence of different architectural design patterns on system quality attributes. The findings may provide practical guidance for practitioners designing ML-enabled annotation systems under different quality and operational requirements.

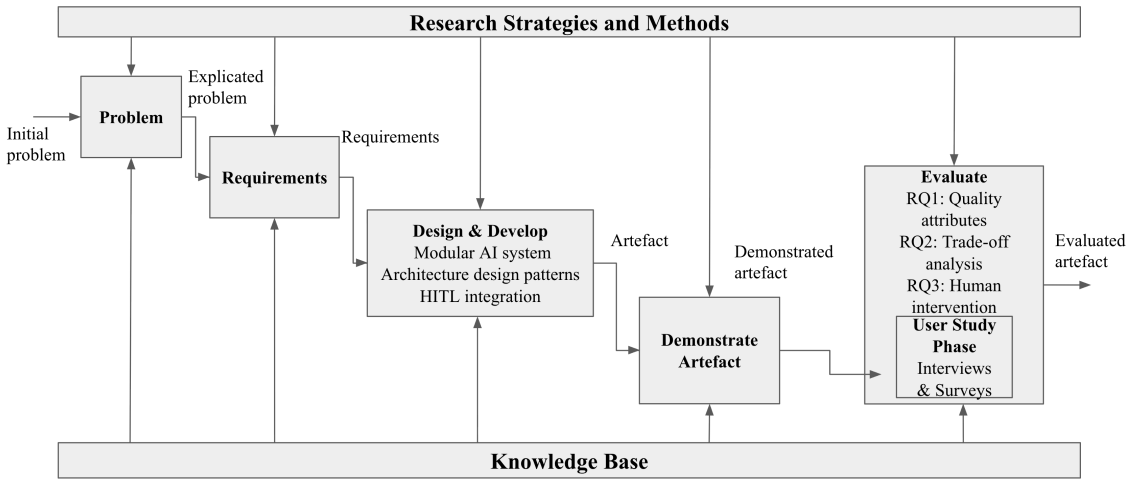


Figure 4.1: Overview of the research methodology following a design science approach

To address the research questions, we conducted the study in five main steps.

- **Definition of architectural variants (Section 4.3).** We designed six human-AI annotation architectural variants based on established AI architectural design patterns, particularly sequential and parallel approaches [12, 9], and extend them with a novel VLM-assisted variant introduced in this study. The architectural variants were derived from identified system requirements and architectural coordination strategies discussed in ML-enabled systems literature.

  This step enabled controlled comparison of how different architectural design patterns influence system quality attributes and architectural trade-offs in human-AI annotation workflows (RQ1, RQ2).

- **Development of a modular human-AI annotation system (Sections 4.2 and 4.4).** We implemented a modular annotation platform integrating YOLO11n, OWL-ViT, VLM-based prompt generation, and human-in-the-loop validation. Using a shared infrastructure ensured that observed differences originated from architectural design decisions rather than implementation inconsistencies. This enabled systematic evaluation of workflow behavior across all research questions (RQ1, RQ2, RQ3).

- **Demonstration through controlled annotation experiments (Sections 4.5 and 4.6).** We executed all architectural variants on the same dataset and

annotation tasks under controlled experimental conditions. During execution, we collected annotation time, interaction logs, bounding-box modifications, and system latency. These measurements enabled evaluation of functional correctness, performance efficiency, and human intervention effort (RQ1, RQ2, RQ3).

- **User study and qualitative evaluation (Section 4.8).** We conducted a user study including annotation sessions, interviews, and surveys with participants interacting with different architectural variants. This step enabled investigation of cognitive workload, usability, automation bias, and perceived collaboration quality between humans and AI systems (RQ2, RQ3). The user study and inter-annotator agreement analysis enabled investigation of how architectural variants influenced cognitive workload, human oversight, and annotation consistency (RQ3).

- **Comparative analysis of quality attributes and trade-offs (Sections 5 and 6).** We analyzed the collected quantitative and qualitative data to compare architectural variants across functional correctness, maintainability, performance efficiency, annotation consistency, and human intervention effort. This analysis enabled the identification of architectural trade-offs between automation, maintainability, performance, and user experience (RQ1, RQ2, RQ3).

4.2 System Architecture Overview

In this study, we developed a modular platform to compare different human-AI annotation architectural design patterns for a perception-based image dataset collected from VAS's mining and quarrying environment.

The system terminology is defined as follows: a job represents a complete workflow execution, while a task corresponds to the processing of a single image within a job. The ontology defines the project's label categories and annotation rules. A workflow is the high-level execution plan that defines how raw inputs are transformed into final annotations by organizing stages, decision points, and optional human review into one coherent process. A stage is a discrete workflow phase (e.g., YOLO detection, OWL-ViT zero-shot refinement, and HITL validation), and the workflow is the ordered composition of these stages into a complete annotation process.

As shown in Figure 4.2, our system uses a layered architecture that consists of an user interaction layer for annotators, with an API gateway for workflow/API control, an orchestration layer for executing stages and architectural design patterns, and a visual perception layer for integrating AI models (a human annotator validates annotations with HITL review), and a data management and analytics layer for storage and evaluation. The system's software architecture separates the web-based annotation client from the backend services.

Furthermore, we implement various architectural design patterns, including sequen-

tial, parallel, and VLM-assisted as configurable pipelines under a common interface. This approach enables controlled comparison of quality attributes.

On the frontend side, an annotation client is developed in React and capable of several actions, such as managing batch image uploads, visualization of model results, manual bounding-box editing, per-image review states, and transitions between workflow stages. The backend leverages FastAPI route groups to handle job creation and completion, execution of workflow stages, asynchronous processing, as well as task-level review and export operations.

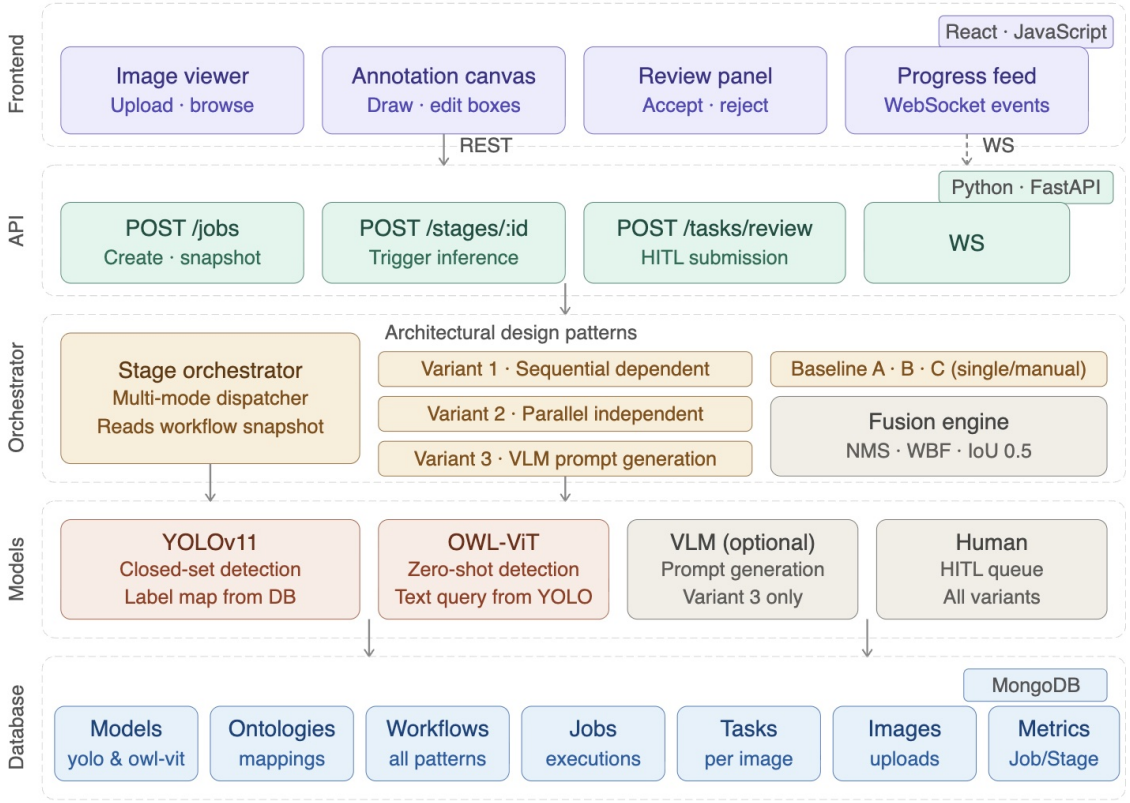


Figure 4.2: High-level diagram of system

In our system, the frontend is the part where the annotator actually interacts with the user interface that lets users run architectural design patterns, inspect predictions, and provide corrections, and communicates with the backend with regular REST calls for actions such as loading data or starting jobs, and WebSocket updates for live progress. On the backend, FastAPI serves as the entry point. It routes requests to workflow, stage, job, and annotation endpoints, while the orchestration layer executes stages asynchronously. After that, the visual perception/inference layer chooses AI model runners, fuses and postprocesses outputs, and packages the results into task-ready annotations. Finally, a data management repository layer persists structured metadata in MongoDB and stores files such as images, and has a clean separation between user interaction, execution logic, AI processing, and data storage.

4.2.1 Annotation Interface

We designed the annotation interface of the human-AI annotation system to make the human reviewing task more explicit, traceable, and efficient. Uploaded images are stored with per-image annotation state, enabling each box to be accepted, rejected, edited, hidden, or deleted independently. This interaction model supports fine-grained measurement of reviewer behavior, which is necessary for evaluating intervention effort under RQ3.

Moreover, the interface supports class-level and bounding-box-level visibility control, confidence-threshold filtering, and manual bounding-box creation. All these features are integrated with workflow state so that user actions are constrained when AI inference is running (UI lock), reducing race conditions and accidental edits during asynchronous stage execution. The review submission is prepared and then sent to the backend review endpoints.

Finally, the system supports realistic HITL behavior across image sequences. For example, the user interface includes optional sequential carry-forward logic that allows accepted boxes to be kept in the next image and revalidated by the annotator. This functionality helps reduce repetitive manual effort while maintaining human control over final acceptance.

A screenshot of the user interface is shown in Figure 4.3, to represent the sequentially dependent models pattern. The system shows AI model-detected bounding boxes, their object classes, and the confidence percentage to the HITL stage.

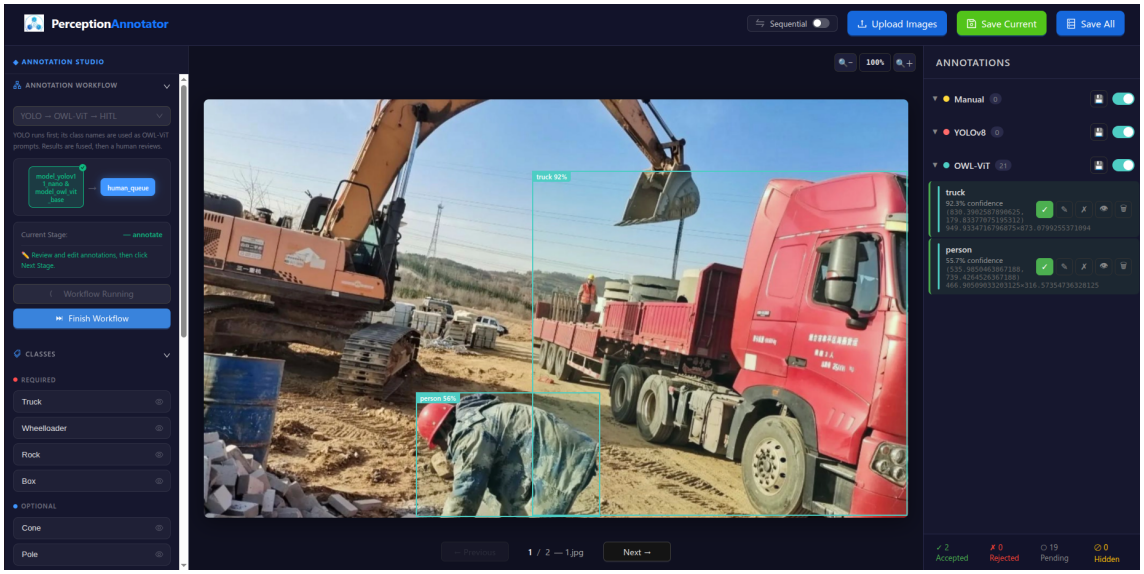


Figure 4.3: Annotation user interface

4.2.2 Workflow Orchestrator

As the foundation for executing requests received from the API layer, the workflow orchestrator allows for the asynchronous execution of stage-driven requests while

providing the coordination of sequential and parallel stage patterns, maintaining state transitions, and generating progress events to the frontend.

A key architectural principle supporting the configurability and data-driven nature of the workflow logic was the ability of the orchestration layer to support both automated execution paths and human-in-the-loop checkpoints.

Additionally, fallback behavior can be enabled when the confidence or validation conditions are not met, with rule and policy checks in place to ensure that quality and safety constraints are met. The centralization of orchestration makes tracking, reproducing, and improving workflow behavior over time much easier.

4.2.3 Visual Perception Module

The Visual Perception module generates initial annotations from input images by using a variety of AI-based detection models. It is built within the context of the overall architecture as an interchangeable set of components that provide perception capability under various assumptions and constraints.

This module combines multiple detection models to support the evaluation of alternative architectural design patterns. We use YOLO11n [50], a supervised model for object detection which generates bounding boxes and class labels for predefined object categories. This model provides quick and dependable detections in a fixed label space and constitutes a baseline perception component of the workflow.

The second detection model employed by the Visual Perception Module is an open vocabulary model OWL-ViT [23] (google/owlvit-base-patch32). The model performed zero-shot detection using textual prompts corresponding to the target object categories (e.g., “car”, “truck”, and “person”). OWL-ViT is capable of detecting object categories that the system had not previously been exposed to. Thus, the OWL-ViT facilitates greater flexibility in dynamically changing environments as compared with supervised models. However, this flexibility introduces variability in detection quality, particularly in terms of localization precision.

Also, we consider incorporating a semantic reasoning component through the integration of a VLM into this module. The VLM produces textual descriptions of the visual content in a scene and then these textual descriptions are used as prompts to the zero-shot detection model. This allows for a more meaningful understanding of the visual content and enables more contextually aware detection.

Selected models are viewed as modular building blocks that can be combined into different configurations. Our design approach allows us to investigate how various combinations of perception components impact overall system quality attributes, rather than investigating the performance of each individual model.

4.2.4 Data Management and Logging

In our system, operational entities like workflows, jobs, tasks, annotations, and their associated metadata are stored within the repository abstraction in MongoDB. On the other hand, large file types, such as images and exports, are stored in the file system to avoid large binary objects from overloading the database. The backend enforces consistent read and write flows for task state, annotation updates, and export generation that support traceability and reduce coupling between modules.

4.3 Proposed Architectural Design Patterns

Our proposed architectural variants are informed by established AI architectural design patterns [12, 9], specifically Parallel Independent Models and Sequentially Dependent Models. Variants 1 and 2 directly instantiate these two patterns respectively, while Variant 3 proposes a novel hybrid that introduces a VLM for prompt generation, aiming to provide richer semantic context for OWL-ViT while retaining YOLO as an independent detector for fusion. While these patterns are widely used for AI-based systems, their efficacy in human-AI collaborative environments, where both predictive accuracy and human labor cost are critical, remains underexplored.

In human-in-the-loop annotation workflows, additional factors must be considered:

- whether model dependencies improve or propagate errors
- whether automated fusion reduces or increases human correction effort
- whether OWL-ViT generates more accurate results when VLMs provide more accurate prompts

To investigate these questions, we design three architectural variants and three baselines.

All variants share the same ontology, and model implementations to ensure controlled comparison.

4.3.1 Baseline Architectures

To contextualize the performance of multi-model workflows, three baseline architectures are included in the comparison.

These baselines represent commonly used annotation strategies and provide references for evaluating the benefits and trade-offs of more complex human-AI collaboration structures.

Baseline A: Single-Model Closed-Set Detection (Figure 4.4)

YOLO performs object detection using a predefined label space, producing an initial detection set r_1. The annotator reviews and corrects bounding boxes to produce the

final annotation r_2. This baseline represents conventional AI-assisted annotation design patterns using a single supervised detection model.

RQ relevance This architecture provides a reference for evaluating whether the proposed architectures that combine multiple AI models improve accuracy or reduce human correction effort (RQ1, RQ3).

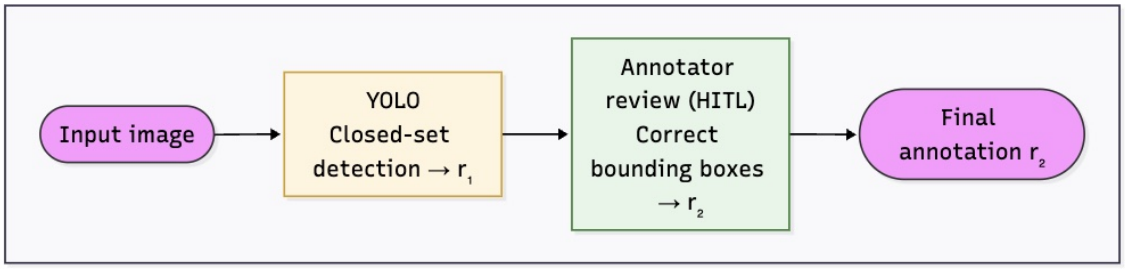


Figure 4.4: Baseline A: YOLO → HITL

Baseline B: Single-Model Open-Vocabulary Detection (Figure 4.5)

OWL-ViT generates bounding boxes conditioned on predefined textual prompts derived from the ontology. The annotator reviews and corrects detections to produce final annotations. This baseline evaluates the effectiveness of open-vocabulary detection without structured model cooperation.

RQ relevance This architecture isolates the effect of language-based detection on annotation quality and effort (RQ1, RQ3).

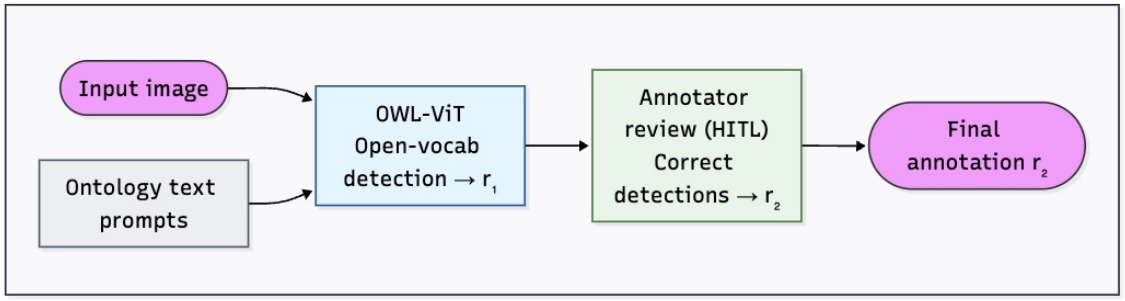


Figure 4.5: Baseline B: OWL-ViT → HITL

Baseline C: Manual Annotation (Figure 4.6)

Annotators manually create bounding boxes and labels without AI-generated suggestions. This baseline represents traditional manual annotation workflows and provides a lower-bound reference for human effort and annotation time.

RQ relevance This architecture enables evaluation of how AI assistance influences annotation speed and cognitive workload (RQ3).

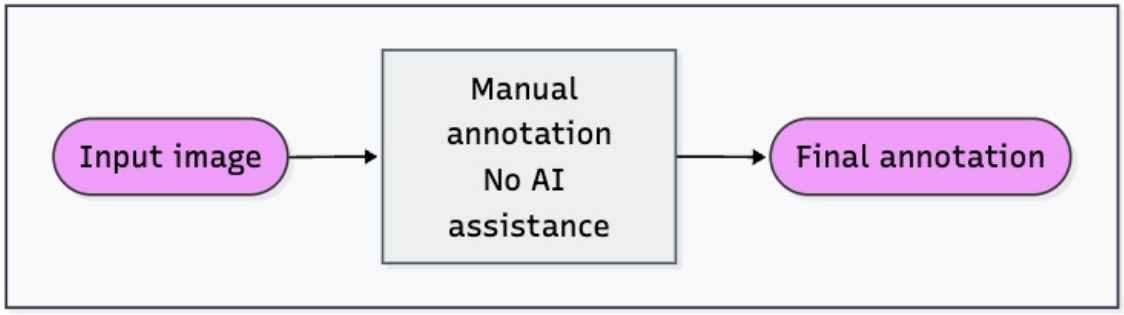


Figure 4.6: Baseline C: Manual Labeling

4.3.2 Multi-model Architectural Variants

Variant 1: Sequentially Dependent Models Pattern (Figure 4.7)

The Sequentially Dependent Models Pattern organizes AI components in staged execution pipelines where downstream models consume outputs generated by previous stages. This pattern enables iterative refinement and information propagation across workflow stages, but may also introduce stronger dependency propagation and error accumulation between components. In this thesis, the pattern is realized through a workflow where YOLO detections are forwarded to OWL-ViT before human validation.

Workflow In this architecture, YOLO is first applied to the input image to generate an initial set of bounding box detections r_1. Detected class labels from YOLO are mapped to ontology concepts and converted into textual prompts that can be used as input queries for OWL-ViT. With images and input queries, OWL-ViT then performs open-vocabulary detection conditioned on these prompts, producing a refined detection set r_2. The outputs r_1 and r_2 are combined using the fusion mechanism described in Section 4.4.3, resulting in a unified detection set r_3. The fused result is presented to the annotator for review, producing the final annotation r_4.

Design Rationale This architectural variant explores data-driven dependency. We aim to test whether using a supervised learning model (YOLO) to constrain a zero-shot model (OWL-ViT) can improve precision while reducing the overall search space, although at the risk of error propagation from the first stage.

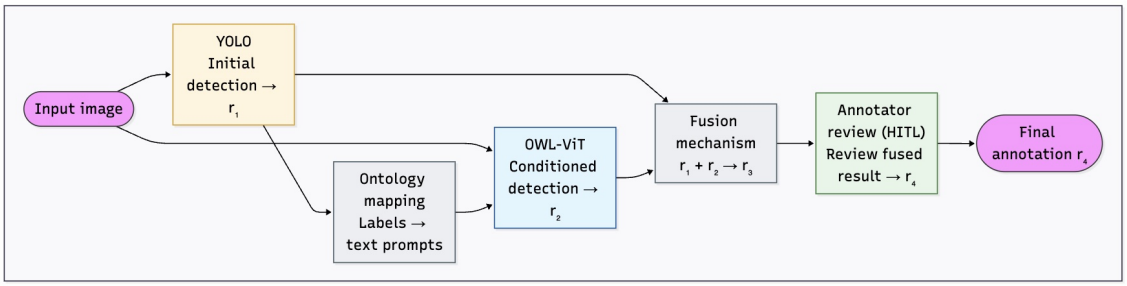


Figure 4.7: Variant 1: Sequentially Dependent Models Pattern

Variant 2: Parallel Independent Models Pattern (Figure 4.8)

The Parallel Independent Models Pattern executes multiple AI models independently on the same input data before combining or validating their outputs at a later stage. This pattern may improve detection diversity and reduce direct inter-model dependencies, but can also increase synchronization complexity and human decision burden during review. In this thesis, the pattern is realized by executing YOLO and OWL-ViT simultaneously before presenting both outputs during the HITL review stage.

Workflow In this architecture, YOLO and OWL-ViT process the input image independently and in parallel. The resulting detection sets r_1 and r_2 are presented directly to the annotator without automatic fusion. Bounding boxes produced by both models are visualized simultaneously in the annotation interface. When conflicts occur, the annotator selects the preferred bounding box or rejects incorrect predictions. The reviewed result r_3 becomes the final annotation output.

Design Rationale This design focuses on performance efficiency (via concurrency) and human agency. We want to investigate whether presenting raw, multi-source information reduces the automation bias and whether the lack of automated fusion increases the cognitive load on humans.

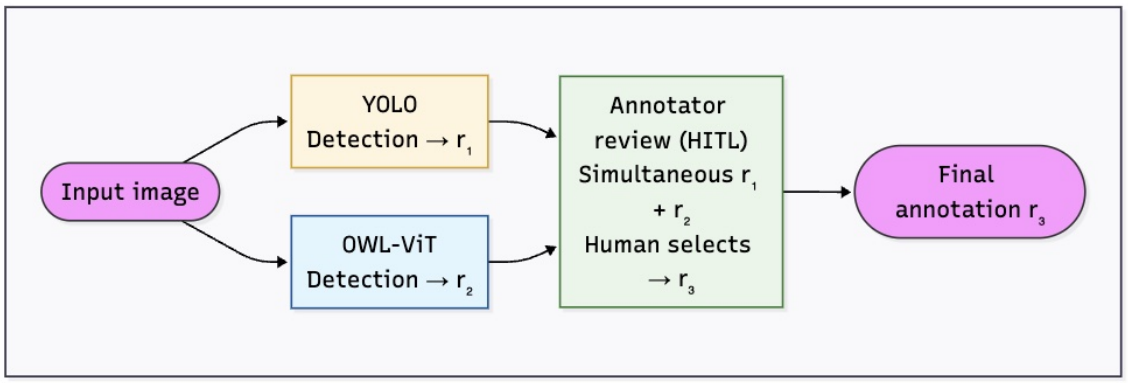


Figure 4.8: Variant 2: Parallel Independent Models Pattern

Variant 3: VLM-assisted Prompt Generation (Figure 4.9)

Workflow In this architecture, YOLO produces initial detections r_1. Simultaneously, a vision-language model (VLM) generates a textual description of the image, such as "The image includes an excavator and a dump truck, both construction-related equipment". The generated description d_1 is converted into text prompts for OWL-ViT, producing detection results r_2. The resulting detection sets r_1 and r_2 are presented without automatic fusion. The annotator may then select, reject, or manually add bounding boxes, and the reviewed result becomes the final annotation r_3.

Design Rationale Variant 3 represents a novel hybrid architecture that synthesizes elements from both the Sequentially Dependent Models Pattern and the Parallel Independent Models Pattern. This architectural variant targets system robustness. By using a VLM for prompt generation instead of YOLO, we aim to mitigate the dependency failures observed in Variant 1. We hypothesize that the VLM's superior semantic understanding provides more accurate prompts for OWL-ViT, potentially leading to higher recall and lower human correction rates.

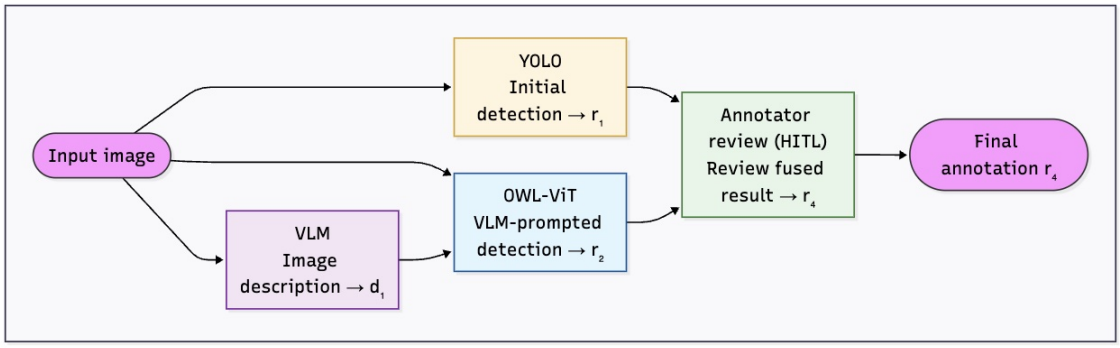


Figure 4.9: Variant 3: VLM-assisted Prompt Generation

4.4 Implementation

This section describes the key system components implemented to enable controlled empirical comparison of different architectural design patterns. The implementation follows three methodological principles:

1. All patterns share the same underlying infrastructure so that differences in system quality attributes can be attributed to the architectural pattern design itself, rather than to inconsistencies in implementation.

2. Key technical components are implemented in a configurable way, making it possible to systematically vary the architectural structure while keeping other factors constant.

3. Implementation details are presented selectively, focusing only on aspects that

directly influence the evaluation of system quality attributes, including maintainability, performance efficiency, and functional correctness.

4.4.1 Maintainability-Oriented Design Decisions

This subsection describes the implementation mechanisms and architectural decisions introduced to support maintainability and modularity within the proposed annotation system.

Model Registry Mechanism We introduce a model registry abstraction to decouple model-specific configuration from workflow logic. Each model is represented by a configuration record that contains its identifier, weight path, version, and default settings. With this design, models can be updated or replaced without modifying the orchestration logic, making it easier to explore alternative patterns with minimal development effort. This supports a systematic comparison of maintainability across different architectural design patterns.

Ontology Configuration To enhance system maintainability, we designed a configurable ontology management mechanism in which the label taxonomy and its mappings to models are stored in a database rather than hardcoded in the system. Table 4.1 presents the concrete taxonomy used in this study, defining 8 object categories across two priority levels. The aliases column provides natural-language synonyms that serve directly as zero-shot text prompts for OWL-ViT. Table 4.2 shows how YOLO's raw output labels are mapped to class IDs, decoupling model internals from the shared taxonomy. This design allows new annotation categories to be added or modified without redeploying models or changing code, thereby reducing the cost of system evolution. By centrally storing category names, aliases, and model label mappings, the system is able to:

- Support flexible adaptation of zero-shot models: The `aliases` field provides synonyms for each category, enabling OWL-ViT to dynamically recognize categories based on textual prompts. Since OWL-ViT returns whichever prompt string best matches the detected object (e.g., "machiner" instead of "wheel_loader"), a post-processing normalization step maps all alias variants back to their class ID before results are stored.

- Decouple models from labels: The `model_mappings` field links YOLO's raw output labels to unified ontology class IDs. Even if YOLO weights are updated or training labels change, only the mapping table needs to be modified.

- Facilitate version management: The `version` field supports semantic versioning, allowing historical label schema versions to be tracked as the ontology evolves, ensuring compatibility with previously annotated data.

This design directly contributes to the maintainability considerations in RQ1. In practice, extending the system with new categories or replacing models only requires updating configuration data rather than modifying the codebase. Compared to di-

rect code refactoring, this approach is expected to reduce the effort and risk involved in system maintenance.

Table 4.1: Traffic Objects Ontology v1.0.0

Class ID	Display Name	Category	Aliases
`truck`	Truck	Required	lorry, pickup
`wheel_loader`	Wheel Loader	Required	machinery, excavator
`rock`	Rock	Required	—
`box`	Box	Required	—
`cone`	Cone	Optional	—
`pole`	Pole	Optional	—
`car`	Car	Optional	automobile, vehicle, sedan
`person`	Person	Optional	pedestrian, human

Table 4.2: YOLO Label Mapping to Ontology Class IDs (`model_yolov11_nano`)

YOLO Raw Label	Class ID
`truck`	`truck`
`wheel_loader`	`wheel_loader`
`wheel loader`	`wheel_loader`
`rock`	`rock`
`box`	`box`
`cone`	`cone`
`pole`	`pole`
`car`	`car`
`person`	`person`

4.4.2 Performance Efficiency-Oriented Design Decisions

This subsection presents the architectural and implementation decisions introduced to improve workflow execution efficiency and operational performance.

Workload-Oriented Database Selection Human-AI annotation workflows involve a combination of high-volume batch insertion of AI-generated detections and frequent localized updates triggered by human corrections. Because each image may contain multiple bounding boxes subject to repeated modification, efficient support for partial document updates is essential. A workload-oriented comparison was conducted between PostgreSQL and MongoDB. While PostgreSQL provides strong ACID guarantees and structured relational querying capabilities, partial updates to JSONB fields require rewriting entire rows. Under high-frequency HITL corrections, this may result in increased disk I/O overhead and reduced performance efficiency. MongoDB's document-oriented representation allows each image and its associated bounding boxes to be stored as a single document. Partial updates to nested bounding boxes can be applied using operators such as `$set`, `$push`, and `$pull`, reducing

write amplification. Given that the dominant workload characteristic involves frequent localized updates rather than complex relational joins, MongoDB is selected to improve update efficiency and reduce latency during iterative annotation refinement.

Real-Time Communication Strategy The system uses WebSocket-based communication to push model execution status and intermediate detection results to the frontend. Compared to polling-based approaches, WebSocket communication reduces redundant client requests and decreases server load, particularly when multiple models are executed sequentially. This design reduces latency between model execution and user feedback, improving interaction efficiency in HITL workflows.

4.4.3 Functional Correctness-Oriented Design Decisions

This subsection describes the implementation mechanisms and architectural decisions introduced to support functional correctness within the proposed AI-assisted annotation system.

Complementary Model Integration Several detection approaches are combined:

YOLO11n [50] is used as a closed-set detector optimized for fast and precise detection of predefined domain-specific object categories.

OWL-ViT [23] (google/owlvit-base-patch32) is used as an open-vocabulary detector capable of identifying unseen or rare objects described using natural language prompts.

VLM is included to support prompt generation and contextual understanding, rather than direct bounding box prediction.

Together, these components allow detection of both common and rare object categories and enable analysis of how architectural structure influences detection correctness and human correction effort (RQ1, RQ3).

Label Space Alignment Different models produce heterogeneous label formats. To ensure consistent semantic interpretation, a unified ontology mapping strategy is implemented: YOLO integer class ID $\rightarrow$ label string $\rightarrow$ ontology concept

This mapping allows detection results from different models to be compared and fused consistently across workflows.

Soft Deletion Strategy Soft deletion is applied at the image level to preserve annotation history. This design allows removed bounding boxes to be recorded as hard negative samples for potential future model fine-tuning, which may improve detection accuracy.

Fusion Logic In sequential workflows, detection results from YOLOv11 and OWL-ViT are merged into a unified annotation set. We consider two candidate fusion

strategies.

Non-Maximum Suppression (NMS) [51] suppresses overlapping bounding boxes by retaining only the one with the highest confidence, which is computationally simple but may discard potentially complementary detections. Weighted Box Fusion (WBF) [52], in contrast, merges overlapping boxes through confidence-weighted averaging, preserving more information but requiring confidence scores to be comparable across models.

A key challenge in fusion-based workflows arises from the incompatibility of confidence scores produced by different detection models. YOLO outputs supervised class probabilities, whereas OWL-ViT produces similarity-based scores between image regions and textual embeddings. Because these confidence values are not directly comparable, a shared operational threshold was applied as a practical filtering mechanism during annotation experiments. The threshold was therefore used for workflow consistency rather than as a statistically calibrated comparison between model outputs.

Further discussion of calibration limitations is provided in Section 6.7.3.

4.5 Experimental Setup

The experimental setup is designed to enable a controlled and systematic comparison of different architectural design patterns in human-AI annotation systems. The goal is to evaluate how six architectural variants affect the quality attributes and human intervention defined in RQ1–RQ3.

4.5.1 Research Variables

Each architectural variant is executed independently on the same dataset. The workflows differ in how model outputs are generated, combined, and presented to the user, including sequential execution and parallel processing.

Table 4.3: Variables and their descriptions

Variable Type	**Description**
Independent	Six architectural variants (Baseline A, B, C; Variant 1, 2, 3) - detailed in Section 4.3
Dependent	Functional correctness (precision, recall, F1-score, mAP50), maintainability (modularity, structural complexity metrics), performance efficiency (latency, throughput), human intervention (correction rate, time on task, acceptance rate, AI coverage, inter-annotator agreement metrics) - defined in Section 4.7

4.5.2 Dataset Preparation and Fine-Tuning

In the initial phase of the study, open datasets that include construction and road scene images were used [1]. The dataset contained relevant object categories such as construction vehicles, traffic elements, and roadwork indicators. Example images are shown in Figure 4.10. A subset of 50 images was manually selected for controlled experiments to match the ontology needed for VAS and scene composition. This subset was used across all architectural design pattern alternatives to conduct a comparable and equal evaluation.

In later stages of the study for the final experimental evaluation, domain-specific datasets that include real-world perception data collected in mining and quarrying environments under varying conditions, such as different lighting and weather scenarios provided by Volvo Autonomous Solutions, were used. Sample images collected from fisheye cameras mounted on top of autonomous trucks can be seen in Figure 4.11. Due to confidentiality restrictions, only representative examples were presented.

To make the experiment process consistent, all images were preprocessed to a uniform format and resolution before being used in the experiments. This avoided variability in model performance caused by differences in input representation.

Figure 4.10: Example images from the open dataset.

[1] https://github.com/miniexcav/Construction-Machines-Images-Dataset

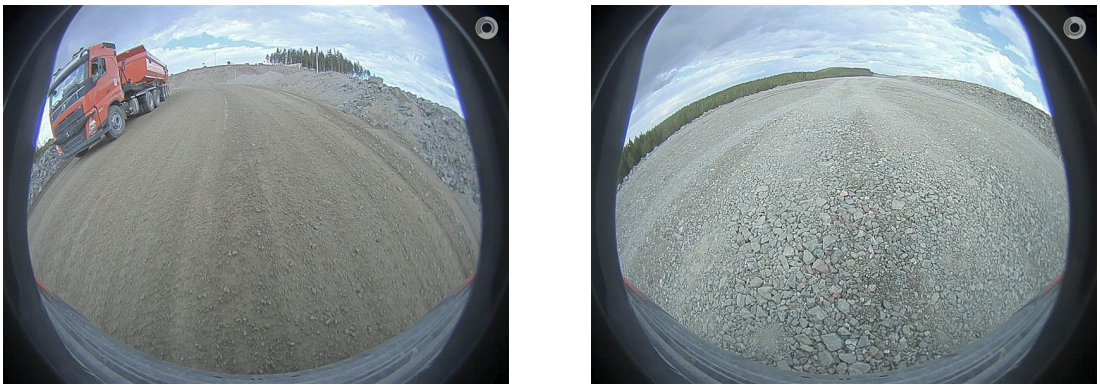


Figure 4.11: Example images from Volvo Autonomous Solutions.

Fine-tuning was necessary because the pretrained YOLO model was not originally optimized for the mining-domain ontology and environmental conditions used in this study. Domain-specific adaptation was therefore required to improve detection performance for industrial perception scenarios.

To support domain adaptation, a fine-tuning dataset was constructed from real-world mining and quarrying images provided by Volvo Autonomous Solutions. The images were collected from fisheye cameras mounted on autonomous haul trucks operating under varying environmental conditions and manually annotated by the authors.

Approximately 600 images were initially selected for dataset preparation. Images that did not contain relevant object instances were excluded from the supervised fine-tuning subset and retained as unused samples for this stage of the experiments.

The remaining annotated dataset was divided into training, validation, and test subsets using an 80/10/10 split strategy [53]. Stratified sampling was applied to preserve class distribution consistency across the subsets and reduce the risk of severe class imbalance between splits. The independent validation and test subsets further supported fair comparison across architectural variants while reducing the risk of overfitting during model development.

- Training set (80%): Used to fine-tune the YOLO model on domain-specific classes.
- Validation set (10%): Used for monitoring model generalization performance during fine-tuning and for preliminary evaluation of detection quality on unseen mining-domain images. The validation subset supported iterative observation of training behavior and helped reduce the risk of overfitting during domain adaptation.
- Test set (10%): Reserved for final evaluation of the fine-tuned YOLO model on previously unseen mining-domain images.

After dataset splitting, the training, validation, and test subsets consisted of 414, 52, and 52 images, respectively. The annotated images include trucks, wheel loaders, rocks, boxes, cones, persons, poles, cars, and piles of rocks.

Due to practical annotation and evaluation constraints, a controlled subset of 150 images was selected for the final comparative experiments across architectural variants. Using a fixed evaluation subset ensured consistent comparison conditions across all architectural variants while keeping the human-in-the-loop evaluation process manageable.

4.5.3 Participants and Annotation Procedure

The controlled annotation experiments used for quantitative architectural variants evaluation and inter-annotator agreement analysis involved three annotators: the two thesis authors and one industrial supervisor from Volvo Autonomous Solutions.

The annotators evaluated the architectural variants using the same ontology definitions, annotation interface, and image subset. Prior to the experiments, annotation guidelines based on the company's annotation procedures were discussed jointly to reduce interpretation inconsistencies. These discussions covered object taxonomy, object boundary definitions, bounding box placement conventions, treatment of partially occluded objects, and criteria for determining whether objects should be annotated in safety-relevant scenarios.

Due to the time-intensive nature of detailed annotation review, each annotator evaluated a subset of the architectural variants rather than all variants. Annotator 1 evaluated the Sequential and OWL-ViT + HITL variants, Annotator 2 evaluated the Manual and VLM-assisted variants, and Annotator 3 evaluated the Parallel and YOLO + HITL variants.

During the annotation tasks, annotators reviewed AI-generated detections, corrected bounding boxes, rejected incorrect predictions, and manually added missing annotations when necessary.

In addition to the controlled annotation experiments, a broader qualitative user study involving 12 participants from Volvo Autonomous Solutions was conducted to investigate usability, cognitive workload, trust in AI-generated annotations, and perceived architectural trade-offs. The number of participants was determined by the availability of relevant personnel within the industrial setting and the exploratory nature of the study. Unlike the controlled experiments, the user study focused primarily on qualitative feedback and did not require all participants to evaluate every architectural variant. Before the experiments, all participants received standardized training on the annotation tool and task requirements for the trial experiments.

Each participant interacted with the annotation system by performing annotation tasks on a fixed set of images across the architectural variants. During the tasks, participants reviewed AI-generated detections, corrected bounding boxes, rejected

incorrect predictions, and manually added missing annotations when necessary.

For each annotation task, the system recorded the following:

- AI-generated predictions
- Participant corrections and final annotations
- Interaction logs (e.g., bounding box adjustments, annotation time)
- System logs (e.g., model inference time, API calls, execution times)

The study design has been reviewed to ensure it does not involve excessive participant burden. Each user study session lasted approximately 60 minutes, including training, annotation tasks, interviews, and survey completion. All data was anonymized prior to analysis. And the collected data is used to compute evaluation metrics and supports the analysis of how architectural decisions influence both system performance and human effort.

4.5.4 Experimental Control

To ensure that observed differences are attributable to architectural design patterns rather than confounding factors, the following elements were held constant across all variants:

- Ontology: Same class definitions, aliases, and model mappings (as described in Section 4.4.1).
- Datasets: Identical images are used, and all workflow stages are applied to each image.
- AI models: The same YOLO11n [50] and OWL-ViT [23] (google/owlvit-base-patch32) models and pretrained weights are used.
- Annotation interface: The same frontend interface is used, differing only in the presentation of AI suggestions (e.g., fused vs. separate).
- Hardware environment: All experiments are run on identical hardware to ensure valid performance comparisons.

4.5.5 Execution Environment

All workflow experiments and annotation tasks were executed using the same operational environment to ensure fair comparison across architectural variants.

The backend services, AI model execution, and workflow orchestration components were executed on the same workstation throughout the experiments. The system was deployed locally and accessed through the web-based annotation interface using a standard browser environment.

All annotation tasks were performed using the same annotation interface, ontology definitions, and workflow configurations. During the experiments, identical pretrained model weights and inference settings were used across all architectural variants. To reduce external variability, no concurrent large-scale computational workloads were executed during the experiments.

Table 4.4: Execution environment specifications

Component	Specification
GPU	NVIDIA A30
GPU Memory	24 GB VRAM
CUDA Version	CUDA 12.2
Deployment	On-premise workstation
Frontend	Browser-based annotation interface
Backend	Local workflow orchestration and AI inference

4.5.6 Parameter Calibration

As discussed in Section 4.4.3, heterogeneous models such as YOLO and OWL-ViT produce confidence scores with different statistical characteristics and distributions. Ideally, model-specific calibration or score normalization procedures would be applied before comparing or combining model outputs.

However, robust confidence calibration requires a sufficiently large and representative validation dataset independent from both training and evaluation data. Due to company-specific dataset access constraints and the limited availability of additional annotated mining-domain images within the thesis time frame, a separate large-scale calibration dataset could not be established.

Instead, an operational confidence threshold of 40% was selected empirically through pilot experiments and applied consistently across all variants during annotation evaluation. Lower thresholds produced excessive low-confidence detections that increased interaction complexity and cognitive workload for annotators, whereas higher thresholds suppressed potentially useful detections, particularly for OWL-ViT-based workflows.

The selected threshold was therefore used as a practical filtering criterion for human-in-the-loop evaluation rather than as a statistically calibrated equivalence between heterogeneous model confidence scores. The same operational threshold was applied consistently across all variants to maintain comparable experimental conditions.

Further investigation into model-specific calibration or normalization strategies is left for future work.

4.6 Data Collection

In line with the design science research approach, data collection in this study is designed to support both the development and evaluation of the proposed artefact. The collected data enabled systematic comparison of different architectural design patterns in human-AI annotation systems and supports both quantitative and qualitative analysis.

The data collection process consisted of three main components:

1. Experimental annotation data collected from controlled experiments,

2. System-generated data obtained during the experiments for the execution of architectural design patterns, and

3. User interaction data gathered during human-in-the-loop (HITL) evaluation.

In addition to system-generated and interaction data, complementary qualitative and quantitative data were collected through the user study described in Section 4.8. Following the annotation tasks, participants completed semi-structured interviews and optional survey questions to provide feedback on usability, human effort, trust in AI assistance, and trade-offs between automation and human control. Section 4.8 User Study Design provides a detailed description of the study procedure, interview protocol, and survey instrument.

4.6.1 Annotation Experiment Outputs

For each architectural variant, annotation experiments were conducted using a controlled subset of 150 images. The resulting prediction outputs, including bounding boxes, class labels, confidence scores, and workflow metadata, were stored in structured JSON format for subsequent analysis.

The generated outputs were compared against manually verified ground-truth annotations using a custom evaluation script. Precision, recall, F1-score, and mAP metrics were computed for each architectural variant to enable quantitative comparison of detection quality and annotation performance across experiments.

To evaluate inter-annotator agreement, each annotator independently completed annotation experiments on an additional subset of 50 images. The resulting annotations were subsequently compared to analyze annotation consistency across different architectural variants.

4.6.2 Model-Generated Data

AI components in our system had different output types and were essential for evaluating the architectural design behavior. Supervised detection was performed by using YOLO, and the output of supervised learning contained bounding boxes, class labels, and confidence scores that represent closed-set predictions based on pre-

defined object ontologies. On the other hand, OWL-VIT enabled open-vocabulary detection by using textual prompts derived from the ontology or generated dynamically.

Moreover, several vision language models were initially considered for integration into the system, including GPT-5.2 [54], LLaVA:latest (8dd30f6b0cb1) [55], Qwen3-vl:latest (901cae732162) [56], MiniCPM-V:latest (c92bfad01205) [57] and Gemma3:4b (a2af6cc3eb7f) [58], because they perform well in generating meaningful descriptions while allowing users to develop applications using API-based access to those models.

The locally hosted VLMs (LLaVA, Qwen3-vl, MiniCPM-V, and Gemma3) were accessed through Ollama using the default inference configuration, including a temperature setting of 0.8. A temperature value of 0.8 was retained because it corresponds to the default Ollama configuration and produced sufficiently detailed object descriptions during preliminary experiments without requiring additional model-specific tuning. Other generation parameters were kept at their default values unless otherwise specified. GPT-5.2 was accessed through the OpenAI API using the default generation settings available at the time of experimentation. Because the objective was not to perform a controlled benchmark comparison between VLMs, each model was initially evaluated using its recommended or default generation settings. All options were evaluated based on integration feasibility, response quality, and system compatibility.

We designed our prompt to extract construction-related objects and contextual information from the scene in a consistent and machine-readable format.

An example prompt corresponding to the left image in Figure 4.10 is shown below:

You are analyzing a road scene for construction-related objects. Determine whether the image contains any construction-related objects or roadwork indicators. Examples include: traffic cones, construction barriers, roadwork signs, excavators, bulldozers, cranes, construction trucks, temporary fencing, safety drums, reflective barricades, workers in high-visibility construction gear, scaffolding, or any other road construction equipment. Answer in exactly this format: Construction detected: Yes or No. Objects: <comma-separated list of detected construction-related objects, or None>. Explanation: <one short sentence>.

The VLMs produced structured responses, including detected objects and a brief explanation. Different captions that were generated by different VLM models for the same image are shown below:

GPT-5.2: Construction detected: Yes. Objects: excavators, traffic cone, temporary fencing, caution tape, construction worker in high-visibility gear. Explanation: The scene shows active excavation with heavy machinery, safety markings, and a worker in a construction area.

LLaVA:latest (8dd30f6b0cb1): Construction detected: Yes. Objects: Construction truck, crane, excavator, roadwork signs, traffic cones, construction barrels. Explanation: There is ongoing construction activity in the image, with a large excavator and a crane visible near the area of repair on the road. The presence of heavy equipment, roadwork signs, and safety markings indicate that there is construction work being carried out in this location.

MiniCPM-V:latest (c92bfad01205): Construction detected: Yes. Objects: Excavator, bulldozer, traffic cones, safety barrier tape, high-visibility clothing; workers in hard hats and reflective gear are also present but not listed as 'construction equipment'. The scene is clearly indicative of ongoing roadwork or construction activities based on the presence of heavy machinery, barriers to control access, and personnel dressed for work in such environments.

Qwen3-vl:latest (901cae732162): Construction detected: Yes Objects: traffic cone, excavators, temporary fencing, workers in high-visibility construction gear Explanation: The image includes construction machinery, safety cones, temporary fencing, and workers wearing high-visibility gear.

Gemma3:4b (a2af6cc3eb7f): Construction detected: Yes Objects: excavator, concrete barriers, safety drums, orange cones, worker, None Explanation: The image shows an excavator, concrete barriers, and safety drums, indicating ongoing roadwork.

The purpose of incorporating a VLM was not to compare model performance, but to investigate how the inclusion of a semantic reasoning component influences the behavior of the annotation workflow at the architectural level. VLMs were incorporated to enhance zero-shot capabilities and natural language descriptions of the scene or relevant objects, then transformed into prompts for OWL-ViT which enables an additional layer of semantic reasoning into the detection pipeline.

After preliminary evaluations, we employed MiniCPM-V as the final VLM because it generated the most consistent structured outputs for downstream prompt generation. The VLM took a road-scene image and outputs a structured description of construction-related objects, which is then used to generate text queries for OWL-

ViT.

The final prompt was designed to produce machine-parsable output without irrelevant text:

> **Analyze this road scene image and identify construction-related objects or roadwork indicators. Prioritize detecting the following object categories: truck, person, cone, car, pole, box, rock, wheel loader, pile of rocks. Include other relevant construction objects if clearly visible. Output ONLY a comma-separated list of unique object names, with no extra text.**

A typical response from MiniCPM-V is: *person, cone, car, wheel loader*.

All model outputs were stored in a structured format, including detection outputs such as bounding box coordinates, predicted classes, confidence values, and source model identifiers, as well as semantic descriptions and generated object prompts produced by the VLM components. These outputs were later used to compute evaluation metrics such as precision, recall, and F1-score, and to analyze how different architectural design pattern alternatives influenced system behavior and human correction effort.

4.6.3 System Logs and Interaction Data

Our human-AI annotation system saves and collects model and stage-level information, as well as user interaction data during HITL workflows. All these logs provide quantitative data, which is used for evaluating human intervention efforts. Detailed information is described in the Section 4.7 Evaluation Metrics.

4.7 Evaluation Metrics

To comprehensively assess the proposed human-AI annotation system, we define a multi-dimensional evaluation framework. The metrics are grouped into four categories aligned with our research questions: functional correctness to measure prediction quality, maintainability to assess software architecture robustness, performance efficiency to evaluate operational cost, and human intervention metrics to quantify annotator effort.

4.7.1 Functional Correctness Metrics

For functional correctness, we focus on correctness, defined as the degree to which the product produces accurate results with the required level of precision, ensuring the system behaves as intended.

To assess the performance of the proposed system and the different architectural design patterns, standard object detection metrics such as precision, recall, and F1-

score have been used as key performance metrics.. These metrics are included in the perception systems since they reflect various aspects of prediction quality.

- **Precision** Precision indicates the fraction of actual positive cases compared to positive predictions [59].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

- **Recall** The fraction of correctly retrieved positive instances over the total number of all relevant positive instances gives the recall value. Also known as true positive rate (TPR) and indicates a system's ability to detect relevant instances in a given dataset [60]. If AI models have high recall, this proves that they are well designed to detect the true instances [61].

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

- **F1-score** F1-score tends to be used to refer to a metric that balances recall and precision and is a harmonic mean of these two metrics [59]. It is particularly useful for imbalanced datasets because it provides the same importance to these two metrics and eliminate the risk of over optimization of one metric to another [62].

$$\text{F1-Score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.3}$$

- **mAP@0.5** Mean Average Precision at an Intersection over Union (IoU) threshold of 0.5, commonly referred to as mAP50, is a standard evaluation metric for object detection tasks. It measures the average precision across all classes while considering a detection as correct if the predicted bounding box overlaps with the ground truth by at least 50% [63].

  Unlike precision, recall, and F1-score, which are computed at a fixed confidence threshold, mAP50 evaluates model performance across varying confidence thresholds using the precision–recall curve. This provides a more comprehensive assessment of both classification and localization performance.

$$mAP@0.5 = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{4.4}$$

- **mAP@[0.50:0.95]** In addition to mAP@0.5 evaluation, the evaluation also includes mAP@[0.50:0.95], which evaluates detection performance across multiple IoU thresholds ranging from 0.5 to 0.95 with increments of 0.05 [63]. This metric provides a stricter and more comprehensive evaluation of localization quality because predicted bounding boxes must increasingly overlap with the ground-truth annotations to be considered correct at higher IoU thresholds.

Unlike mAP@0.5, which primarily evaluates coarse detection capability, mAP50–95 places stronger emphasis on precise object localization and is commonly used in modern object detection benchmarks.

All these metrics have been taken into account in this thesis because they contribute to the evaluation of detection quality and facilitate systematic comparison of different architectural configurations of the human-AI annotation system.

Because the study evaluates object detection workflows rather than simple classification tasks, accuracy was not used as a primary evaluation metric. In object detection scenarios, true negatives are difficult to define meaningfully due to the large background search space and class imbalance. Therefore, precision, recall, F1-score, and mAP-based metrics were considered more appropriate for evaluating detection quality and localization performance.

4.7.2 Maintainability Metrics

Maintainability refers to how easily a software system can be maintained, updated, or improved over time. To evaluate the maintainability of the semi-annotation system, our analysis focuses on modularity and structural complexity.

Modularity refers to the degree to which the system is decomposed into independent and interchangeable components. We evaluate modularity using coupling.

- **Coupling**: It refers to the degree of interdependence between system components.

  According to Habibullah [41], traditional systems primarily consider structured code for maintainability. ML systems contain not only code, but also scripts, data, and models that strongly affect system behavior. They introduce a modified form of the Coupling Between Object Classes (CBO) metric, CBO^{ML}:

$$\text{CBO}^{ML}(C) = \sum_{i=1}^{c} I(C, C_i) + \sum_{j=1}^{d} I(C, D_j) \tag{4.5}$$

  where C is the code file under assessment, c is the total number of code files, d is the total number of data or model files, and $I(C, X)$ equals 1 if C is coupled to X and 0 otherwise.

  In our system, the different execution variants (`human`, `sequential`, `parallel`, `single_model`, `vlm_guided_parallel`) are implemented as separate functions within a single file rather than as separate files. We therefore adapt CBO^{ML} to operate at the function level.

  Additionally, because our system is a runtime orchestration layer with no static data file access, we make two further modifications: (1) we replace

the binary indicator $I(C, C_i)$ with a weighted call count to capture how intensively each function depends on external modules; and (2) we substitute the data file coupling term with coupling to external data-access services (i.e., `TasksRepository` and `ImagesRepository`), which serve the equivalent role of mediating access to persistent data. The adapted metric, $\text{CBO}^{\text{ML}}{}_f$, is defined as:

$$\text{CBO}_f^{ML}(B) = \frac{1}{|F_B|} \sum_{f \in F_B} \left(\sum_{i=1}^{c} w(f, M_i) + \sum_{j=1}^{s} I(f, S_j) \right) \tag{4.6}$$

where B denotes the execution branch under assessment (e.g., `sequential`), F_B is the set of functions belonging to that branch, c is the number of distinct external code modules, s is the number of external data-access services, $w(f, M_i)$ is the total number of calls from function f to module M_i (weighted), and $I(f, S_j)$ equals 1 if function f invokes service S_j and 0 otherwise. The result is averaged over all functions in the branch to produce a per-branch score.

Structural Complexity Metrics: CBO^{ML} metric primarily evaluates the strength of dependencies, but provides limited insight into how dependencies are structurally organized within an architectural variant. To complement coupling-based analysis, we introduce a graph-based Architectural Coordination Complexity Score (ACCS).

Each architectural variant was modeled as a directed graph, where workflow stages were represented as nodes and data or control dependencies were represented as directed edges. The graph representation enabled the extraction of structural characteristics related to coordination and change propagation, including dependency propagation paths and node centrality measures.

ACCS aims to capture the likelihood that modifications in one workflow component propagate through the system. Higher ACCS values indicate greater structural coordination complexity and potentially lower maintainability due to increased change impact. Unlike CBO, ACCS evaluates the organization of those dependencies. Therefore, the two metrics provide complementary perspectives on maintainability.

Graph-based structural properties, including nodes, edges, and workflow depth, are standard graph-based structural properties [64]. They were used to characterize the size and structural complexity of the architectural variants.

For each architectural variant, the following metrics were computed:

- **Number of nodes (N):** The number of nodes represents the total number of workflow stages, including AI models and human-review stages. Larger workflows generally introduce additional integration and maintenance overhead.

- **Number of edges (E):** The number of edges represents the number of directed dependencies between workflow stages. Higher edge counts indicate stronger coupling between workflow components.

- **Workflow depth (longest dependency path):** Workflow depth was defined as the longest directed dependency path within the workflow graph. Greater depth increases the likelihood that modifications in upstream stages propagate through multiple downstream stages.

  For example, the workflow YOLO → OWL-ViT → HITL has a depth of 2 because information propagates through two consecutive dependencies.

- **Betweenness centrality:** Betweenness centrality estimates how strongly a workflow stage acts as an intermediate dependency between other stages. Stages with high betweenness centrality behave as bottlenecks because many workflow paths propagate through them.

$$B_{norm}(v) = \frac{1}{(N-1)(N-2)} \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{4.7}$$

  where σ_{st} denotes the number of shortest paths between nodes s and t, and $\sigma_{st}(v)$ represents the number of those paths that pass through node v. N is the total number of nodes in the graph and is used to normalize the betweenness centrality [65].

  The average betweenness term was inspired by graph-theoretic betweenness centrality analysis, where nodes with high betweenness frequently act as intermediary coordination points between other nodes [65].

  In the context of workflow orchestration, higher average betweenness values indicate that intermediate workflow stages act as stronger coordination bottlenecks between processing components.

$$\text{AvgBetweenness} = \frac{1}{|V|} \sum_{v \in V} B_{norm}(v) \tag{4.8}$$

  where V is the node set and $|V| = N$ is the number of nodes.

  Betweenness Centrality Interpretation

  As an illustrative example, the Sequentially Dependent Models Pattern forms a linear dependency structure:

$$\text{YOLO} \rightarrow \text{OWL-ViT} \rightarrow \text{HITL}$$

In this architectural variant, the OWL-ViT stage acts as an intermediate dependency bridge between the upstream YOLO stage and the downstream human validation stage. Consequently, OWL-ViT obtains non-zero betweenness centrality because shortest dependency paths propagate through this intermediate stage.

The resulting average betweenness value becomes:

$$\frac{0 + 0.5 + 0}{3} = 0.167$$

where OWL-ViT contributes the only non-zero betweenness value in the workflow graph.

In contrast, the VLM-assisted architectural variant distributes dependencies across multiple stages and branches rather than concentrating propagation through a single intermediate stage. Although the variant contains more components, dependency propagation becomes structurally more distributed, resulting in a lower average betweenness value compared to the fully sequential pipeline.

These examples illustrate that the metric captures structural dependency concentration rather than simply counting architectural variant components.

- **Average propagation:** The Average Propagation metric was inspired by graph reachability [64] and dependency propagation concepts commonly used in Design Structure Matrix (DSM) literature [66]. Graph reachability considers the set of vertices that can be reached through directed paths from a given vertex [64], while DSM-based dependency analysis examines how dependencies and changes may propagate through interconnected system components [66].

  Building on these concepts, the specific formulation used in this thesis computes the average number of downstream components reachable from each workflow component and serves as an indicator of dependency propagation potential within the graph representation of an architectural variant.

  Average propagation was computed as the average number of downstream nodes reachable from each node, representing how far changes may propagate throughout the workflow. Higher values indicate increased change propagation and integration complexity.

$$\text{AvgPropagation} = \frac{1}{N} \sum_{v \in V} |Descendants(v)| \tag{4.9}$$

  In sequential architectural variant, changes in upstream AI stages may propagate across multiple downstream stages, producing higher propagation values compared to parallel architectural variant.

Because the average propagation metric may exceed the [0,1] interval depending on workflow graph size and dependency structure, propagation values were normalized prior to aggregation in ACCS to ensure comparability between metric dimensions.

The normalized propagation value is computed as:

$$norm(avg\_propagation) = \frac{AvgPropagation}{N-1}$$

where N represents the number of workflow stages (nodes) in the orchestration graph. This normalization scales propagation relative to the maximum theoretical propagation extent within each architectural variant.

- **Automation ratio:** To characterize the degree of automation in each workflow, we introduce the Automation Ratio, defined as the proportion of AI-driven workflow stages relative to the total number of workflow stages. Higher values indicate stronger reliance on automated model-based processing. Automation ratio was defined as:

$$\text{AutomationRatio} = \frac{ModelNodes}{N} \tag{4.10}$$

  where:

  - $ModelNodes$ represents the number of AI model stages in the workflow,
  - N represents the total number of workflow stages.

- **Fan-in:** Fan-in represents the number of incoming dependencies to a workflow stage [67].

$$\text{FanIn(v)} = |\{u \mid (u, v) \in E\}| \tag{4.11}$$

  where:

  - v represents a workflow stage,
  - E represents the set of directed workflow dependencies.

  Higher fan-in indicates that a workflow stage depends on multiple upstream stages, increasing integration complexity and coordination effort.

- **Fan-out:** Fan-out represents the number of outgoing dependencies from a workflow stage [67].

$$\text{FanOut(v)} = |\{w \mid (v, w) \in E\}| \tag{4.12}$$

  where:

  - v represents a workflow stage,

– E represents the set of directed workflow dependencies.

Higher fan-out indicates that changes in a workflow stage may propagate to multiple downstream stages, increasing maintenance and regression-testing effort.

These metrics were selected because they capture workflow properties associated with maintainability, including coupling, dependency propagation, structural complexity, and reliance on external AI components.

- **Architectural Coordination Complexity Score (ACCS)** The metric is intended as a heuristic comparative indicator rather than an objective measure of maintainability or software quality.

  Existing maintainability metrics primarily capture structural properties such as coupling and dependency relationships. Therefore, ACCS was introduced as a complementary composite metric that summarizes workflow-level coordination characteristics and supports comparison between architectural variants.

  To support comparative analysis of workflow orchestration complexity across variants, an Architectural Coordination Complexity Score (ACCS) was defined by combining workflow centralization, dependency propagation, and automation level:

$$Score = \frac{1}{3}\, avg\_betweenness + \frac{1}{3}\, norm(avg\_propagation) + \frac{1}{3}\, automation\_ratio$$

  The average betweenness term estimates the extent to which workflow stages act as coordination bottlenecks within the annotation pipeline. The average propagation term reflects the relative extent to which dependencies and potential changes may propagate across workflow stages, while the automation ratio represents the proportion of automated stages relative to human-controlled workflow steps.

  The weighting coefficients were explicitly selected to keep the metric interpretable across relatively small workflow graphs. The selected values were heuristic rather than empirically learned parameters.

  Higher scores therefore indicate workflows with stronger dependency propagation, greater coordination bottlenecks, and higher levels of orchestration automation, whereas lower scores represent structurally simpler workflows with fewer coordination dependencies.

  Equal weighting was selected to avoid prioritizing any single structural characteristic during exploratory comparison across variants. The selected metrics were not intended as absolute maintainability measures, but rather as comparative indicators for analyzing structural differences between variants.

4.7.3 Performance Efficiency Metrics

Performance Efficiency refers to the ability of a system to handle workloads effectively by balancing resource utilization with responsiveness. In this study, we focus on time behavior, including latency and throughput.

- **Latency**: the time required to process a single stage from input to output.
- **Throughput**: the number of images the system can process per unit time (e.g., items per second). We measure this by dividing the total processing time by the number of images.

4.7.4 Human Intervention Metrics

For RQ3, we focus on how different architectural design patterns affect the level of human intervention. We consider the following usage metrics:

- **Correction Rate** This is the proportion of AI predictions that require manual modification by human annotators, reflecting the correctness of the AI outputs. It is computed as:
$$CR = 1 - \frac{N_{\text{accepted}}}{N_{\text{total}}} \tag{4.13}$$
where N_{accepted} denotes the number of AI predictions accepted without modification, and N_{total} is the total number of annotated bounding boxes.
- **Time on Task** This measures the average time an annotator spends viewing and interacting with each image. We use a front-end timer to record the duration from when an image is displayed to when the annotator clicks the "Next" button.
- **Acceptance Rate** To assess the quality of AI suggestions, we distinguish between two acceptance rate variants:
  - *Pure Acceptance Rate (Pure AR)*: the proportion of AI suggestions accepted by the annotator without any modification:
$$\text{Pure AR} = \frac{N_{\text{accepted}}}{N_{\text{AI suggestions}}} \tag{4.14}$$
  - *Broad Acceptance Rate (Broad AR)*: the proportion of AI suggestions that were either accepted as-is or retained after manual editing:
$$\text{Broad AR} = \frac{N_{\text{accepted}} + N_{\text{edited}}}{N_{\text{AI suggestions}}} \tag{4.15}$$

  Together, these two rates decompose correction behavior: a large gap between Pure AR and Broad AR indicates that AI suggestions are directionally useful but require refinement, whereas a high Pure AR reflects proposals that are directly usable.

- **AI Coverage** This metric captures the proportion of final bounding boxes that originated from AI suggestions (whether accepted or edited), as opposed to boxes drawn entirely from scratch by the annotator:

$$\text{AI Coverage} = \frac{N_{\text{AI suggestions}}}{N_{\text{total}}} \tag{4.16}$$

  A high AI coverage indicates that the model successfully detects most objects, reducing the annotation burden. A low coverage means annotators must draw the majority of boxes manually regardless of the AI assistance offered.

- **Inter-Annotator Agreement Metrics:** To evaluate annotation consistency across different architectural design patterns, an inter-annotator agreement analysis was conducted on a shared subset of 50 images independently by the three annotators.

  Agreement was measured using Cohen's Kappa[68] for pairwise comparisons and Fleiss' Kappa[69] for overall multi-rater agreement. The analysis was performed separately for each architectural design pattern to investigate how its structure influences annotation consistency.

  Kappa-based agreement metrics were selected because they account for agreement occurring by chance. Interpretation of Kappa values follows the guidelines proposed by Landis and Koch [70], shown in Table 4.5.

Table 4.5: Interpretation of Kappa agreement values

Kappa Value	Strength of Agreement
< 0.00	Poor
0.00–0.20	Slight
0.21–0.40	Fair
0.41–0.60	Moderate
0.61–0.80	Substantial
0.81–1.00	Almost Perfect

  Since the focus of the study is architectural design pattern behavior rather than localization precision, agreement was calculated based on object-class presence annotations rather than exact bounding-box overlap.

4.8 User Study Design

We conducted a user study to obtain qualitative feedback on the designed and developed artefact in terms of human effort and perceived trade-offs between different architectural design patterns. In this user study, we evaluated the consequences of practice in real-world usage contexts by domain experts. Moreover, the purpose of the user study was to develop an understanding of how practitioners reason about the trade-offs between human effort and the system's performance, which relates directly to the research questions of this research.

4.8.1 Participants

Participants were recruited from Volvo Autonomous Solutions using convenience sampling. The study involved 12 participants with backgrounds in system verification, software engineering, perception systems, machine learning, and hardware-in-the-loop testing. Participants represented varying levels of annotation familiarity, ranging from novice to expert, enabling the study to capture perspectives from both experienced and inexperienced annotators. Because of practical limitations, the user study is limited in terms of the number of participants. Nevertheless, the participant group included individuals from several engineering roles and varying levels of annotation experience, providing a diverse set of perspectives on the evaluated workflows. In addition, evidence of thematic saturation was observed during data collection. As the study progressed, recurring patterns and feedback became increasingly repetitive, and additional participants contributed relatively few novel insights. Therefore, within the scope of this study, the sample size was considered sufficient for identifying trends, user perceptions, and recurring observations that complement the quantitative evaluation.

Participant Selection: Our target employees were from Virtual Driver Team and consisted of system verification engineers, software engineers and machine learning engineers.

Participants represented diverse professional backgrounds within autonomous systems development, including system verification, perception software engineering, machine learning, simulation, DevOps, and hardware-in-the-loop testing. Most participants had multiple years of industrial experience working with autonomous vehicles, perception systems, or testing infrastructure.

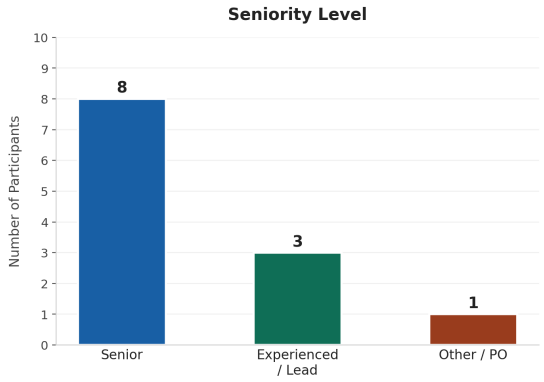
Annotation Familiarity	**Count**
Novice (1–2)	5
Intermediate (3)	6
Expert (4–5)	1

Table 4.7: Participants' Annotation Familiarity Levels

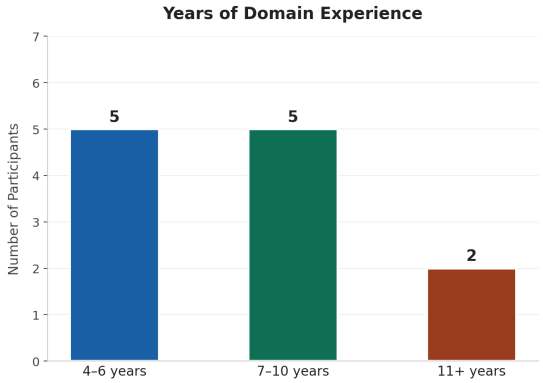
The participant group included both experienced annotators and participants with limited annotation experience, enabling the study to capture perspectives from different levels of familiarity with AI-assisted annotation workflows. Self-reported familiarity with annotation workflows and AI perception systems ranged from novice to expert, with most participants reporting intermediate familiarity.

ID	Experience	Role	Annotation Familiarity
P1	10 years	Expert System Verification Engineer	Intermediate
P2	6 years	Senior System Verification Engineer	Intermediate
P3	4 years	Experienced System Verification Engineer	Intermediate
P4	6 years	Senior System Verification Engineer	Novice
P5	11 years	Senior Perception Software Engineer	Intermediate
P6	10 years	Senior System Verification Engineer	Novice
P7	8 years	Product Owner – Hardware in the Loop	Intermediate
P8	4 years	Senior Software Engineer	Novice
P9	15 years	Lead Machine Learning Engineer	Expert
P10	7 years	Senior System Verification Engineer	Novice
P11	5 years	Senior Software Engineer	Intermediate
P12	7 years	Senior System Verification Engineer	Novice

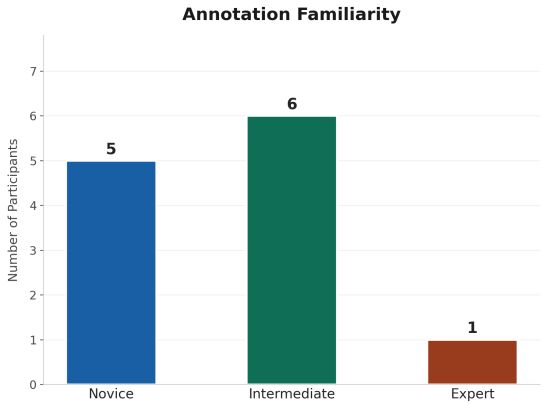
Table 4.6: Professional background and annotation familiarity of study participants



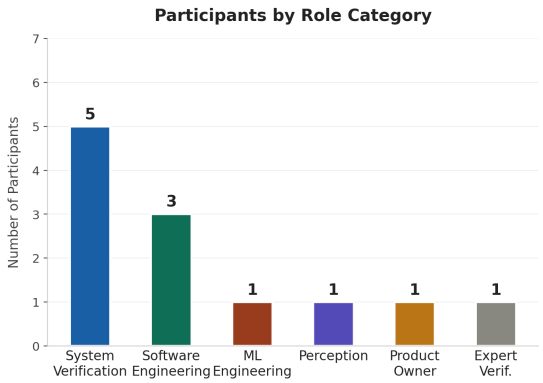
(a) Participant seniority levels



(b) Years of domain experience



(c) Annotation familiarity



(d) Participant role categories

Figure 4.12: Demographic overview of the user study participants

4.8.2 Study Procedure

We designed each user session to last around 60 minutes and conducted the sessions in person. First, we introduced the developed artefact and a brief explanation of the annotation workflow, architectural design patterns, and the role of AI models (supervised detections, zero-shot detections, and vision-language models).

Second, participants then interacted with the system by performing annotation tasks on a subset of images using four architectural variants: manual annotation, sequential, parallel, and VLM-assisted. Due to time limitations, not all architectural variants were included in the user study. Each participant annotated approximately 10 images per workflow while reviewing, accepting, rejecting, and correcting AI-generated bounding boxes.

During the sessions, participants were encouraged to describe their observations, reasoning processes, and difficulties encountered while interacting with the system.

After completing the annotation tasks, semi-structured interviews and short Likert-scale survey questions were conducted to collect qualitative and quantitative feedback related to the research questions focusing on workload, usability, trust in AI-generated annotations, human oversight, and perceived trade-offs between automation and human control.

4.8.3 Data Collection Methods

Besides quantitative data through system logs and interaction metrics collected during annotation experiments, qualitative data was obtained through semi-structured interviews. We asked participants open-ended questions about their experiences with the system, their opinions about different architectural design patterns, and their preferences regarding automation and human involvement. With these interviews, participants provided rationale and further insight into the impact of architectural decisions on usability and workflow efficiency.

Additionally, short survey questions are also asked to participants using Likert-scale ratings to see perceptions of the system, trust in AI predictions, and the overall amount of workload associated with completing annotation tasks. The interview and survey questions are provided in Table 4.8 and Table 4.9.

Prior to the user study, the interview questions, survey instrument, and annotation task procedure were reviewed and piloted with the academic and industrial supervisors to ensure clarity, relevance, and consistency with the study objectives. Feedback from this pilot phase was used to refine the wording of questions and improve the overall study procedure before conducting the full user study.

We held interviews with VAS employees in order to get qualitative feedback on the developed artefact.

The interview and survey instruments are summarized in Table 4.8 and Table 4.9.

Table 4.8: Interview instrument

ID	Question
Demographic and Annotation Experience Questions	
1	Could you describe what you have been working with and your specific role within your current team?
2	What kind of tasks do you usually perform?
3	How many years of experience do you have in your field?
4	How familiar are you with annotation workflows and perception systems? (Scale: 1 = Novice, 5 = Expert)
5	What challenges do you generally encounter during annotation tasks?
Architecture and Workflow Perception Questions RQ2 + RQ3	
6	How easy was it to understand how the annotation system works? (Scale: 1 = Very Confusing, 5 = Very Intuitive)
7	What aspects made the workflow easy or difficult to understand?
8	Which architectural variant felt the most understandable to you?
9	Did you notice significant differences between the architectural variants?
Trade-off and Workflow Evaluation Questions RQ1 + RQ2	
10	When comparing the architectural variants, how would you prioritize accuracy, ease of use, performance speed, maintainability, and required manual effort?
11	Which architectural variant provided the best balance between automation and human control?
12	Did any architectural variants appear faster but less reliable?
Human Intervention and Cognitive Workload Questions RQ3	
13	How did the system affect your annotation workload? (Scale: 1 = Greatly Reduced Workload, 5 = Greatly Increased Workload)
14	Which architectural variant required the most effort and which reduced your workload the most?
15	In which situations did you need to intervene manually the most?
16	How cognitively demanding was each architectural variant? (Scale: 1 = Not Demanding At All, 5 = Extremely Demanding)
Reflection, Usability, and Practicality Questions RQ2 + RQ3	
17	Would you use this annotation system in a real-world workflow? (Scale: 1 = Definitely Not, 5 = Definitely Yes)
18	What recommendations would you provide for improving the system?
19	What concerns would you have regarding production-level deployment of the system?

ID	Question
20	When designing systems that integrate AI models, what aspects do you think are most important to consider from a technical perspective?

Table 4.9: Survey instrument

ID	Question	Response Options
Perceived Workload and Usability Questions		
1	Using the system reduced my annotation effort. (RQ3)	Likert scale (1 = Strongly Disagree, 5 = Strongly Agree)
2	The architectural variants were easy to understand. (RQ2, RQ3)	Likert scale (1 = Strongly Disagree, 5 = Strongly Agree)
3	I felt confident about the AI-generated annotations. (RQ3)	Likert scale (1 = Strongly Disagree, 5 = Strongly Agree)
4	The system was efficient during annotation tasks. (RQ1, RQ2)	Likert scale (1 = Strongly Disagree, 5 = Strongly Agree)

4.8.4 Interview and Survey Analysis

Interview recordings were transcribed and manually reviewed to ensure consistency and correctness. The qualitative data was analyzed using thematic analysis following the approach proposed by Braun and Clarke [71].

Relevant statements from participants were first identified and assigned initial codes representing recurring concepts related to usability, workload, trust in AI predictions, workflow understandability, and human intervention effort. The codes were then iteratively grouped into broader themes describing perceived advantages, limitations, and trade-offs between architectural variants.

Thematic analysis was selected because it enables systematic identification of recurring patterns in participant experiences while remaining flexible for exploratory qualitative research.

Survey responses were analyzed using descriptive statistics. Likert-scale responses were summarized using averages, distributions, and frequency counts to complement the qualitative findings obtained from the interviews.

5

Results

This chapter reports the quantitative and qualitative findings from the user study and the experimental evaluation, including functional correctness, performance efficiency, maintainability, and human intervention, of the six architectural variants.

5.1 Experiment

This section reports the experimental results for functional correctness, performance efficiency, maintainability, and human intervention metrics.

5.1.1 Functional Correctness Results

The functional correctness evaluation results are presented in Table 5.1.

Table 5.1: Functional correctness results across variants

Variant	**Precision**	**Recall**	**F1**	**mAP @0.50**	**mAP @[0.50,0.95]**
Sequential	0.720	0.750	0.735	**0.574**	**0.294**
VLM-assisted	0.706	0.716	0.711	0.524	0.244
Parallel	**0.756**	**0.784**	**0.770**	0.566	0.259
YOLO → HITL	0.741	0.753	0.747	0.527	0.261
OWL-ViT → HITL	0.659	0.781	0.715	0.541	0.280

The Parallel variant achieved the highest overall F1-score (0.770), indicating the strongest balance between precision and recall among the evaluated variants and also achieved the highest recall (0.784), suggesting that combining multiple model outputs improved object coverage.

The Sequential variant achieved the highest mAP@[0.50,0.95] score (0.294), indicating comparatively stronger localization consistency across IoU thresholds while maintaining competitive precision and recall values.

The OWL-ViT-hitl variant achieved the highest recall after the Parallel workflow (0.781) but also produced the lowest precision (0.659), suggesting that the open-vocabulary detector generated more false positive detections.

The YOLO-hitl variant produced relatively balanced performance across all metrics and served as a stable baseline among AI-assisted variants.

The VLM-assisted variant achieved comparable precision and recall values but lower mAP scores compared to the Sequential and Parallel variants.

All variants produced very low mAP@0.95 values which shows that while objects were often detected correctly at lower IoU thresholds, precise bounding-box localization remained difficult across variants.

Overall, the results suggest that the Sequential and Parallel variants improved object coverage and recall compared to single-model baselines. In particular, the Parallel variant achieved the highest recall and F1-score, indicating that combining heterogeneous model outputs may improve detection diversity. However, the Sequential variant achieved stronger localization consistency and was later perceived by participants as easier to validate during the user study which illustrates a trade-off between detection coverage and interaction simplicity.

5.1.1.1 Class-wise Detection Performance

To further analyze class-level detection behavior, a normalized confusion matrix was generated using the test subset of the fine-tuned YOLO11n model, as shown in Figure 5.1.

The normalized confusion matrix indicates that detection performance varied substantially across ontology categories. Classes such as “truck”, “pole”, and “wheel loader” achieved relatively high class-wise detection consistency, whereas categories such as “pile of rocks” and “car” exhibited significantly lower recall and higher background confusion.

The “pile of rocks” category produced particularly high ambiguity, which also aligns with the inter-annotator agreement analysis and participant feedback. Annotators frequently differed in how rock formations were grouped and localized, suggesting that semantic ambiguity within ontology definitions contributed to both human disagreement and model uncertainty.

Furthermore, several object categories showed substantial confusion with the background class, indicating that missed detections remained a major source of error in the mining-domain perception environment.

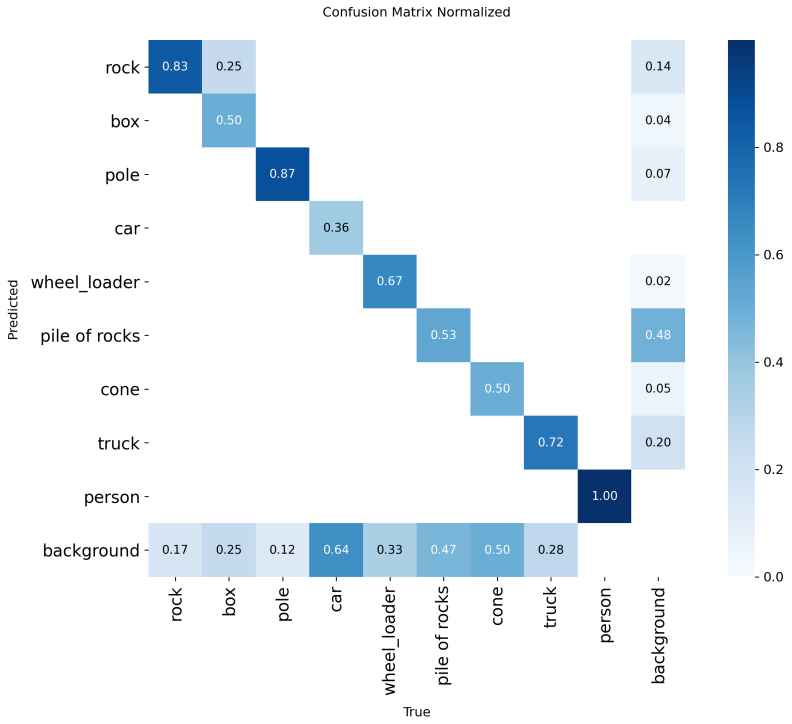


Figure 5.1: Normalized confusion matrix of the fine-tuned YOLO11n detector on the mining-domain test dataset

5.1.2 Performance Efficiency Results

To assess how different human-AI annotation architectural design patterns affect performance efficiency, we measured inference time (total over 150 images), per-image latency (s/img), and throughput (img/s) for each variant. Relative efficiency compared to the Sequential baseline (set to 1.0) is was also computed to support comparative analysis across variants.

The results, summarised in Tables 5.2 and 5.3, reveal substantial differences across patterns.

Table 5.2: Performance Efficiency Comparison Across Variants

Variant	Inference Time (ms) ↓	Latency (s/img) ↓	Throughput (img/s) ↑
Sequential	55,024	0.37	2.73
Parallel	56,077	0.37	2.67
VLM	224,104	1.49	0.67
YOLO + HITL	49,121	0.33	3.05
OWL-ViT + HITL	**8,973**	**0.06**	**16.72**

The single-model variants achieved the highest execution efficiency among the eval-

uated variants. In particular, the results show that the OWL-ViT + HITL approach achieved the highest throughput (16.72 img/s) and lowest latency (0.06 s/img) among all variants, with a 6.12× speedup over the sequential baseline.

Moreover, the YOLO + HITL approach showed modest improvements over the baseline (+12% in throughput and speedup), but was outperformed by OWL-ViT + HITL by a large margin.

In contrast, the sequential and parallel approaches produced nearly identical performance (2.7 img/s, 0.37 s/img). Furthermore, the Parallel variant exhibited slightly higher latency (0.3738 s/img) compared to the Sequential workflow (0.3668 s/img), indicating that parallel execution did not result in measurable performance improvements in the evaluated setup.

The VLM approach showed lower performance efficiency compared to the other variants, with the lowest throughput (0.67 img/s) and highest latency (1.49 s/img).

Table 5.3: Relative Efficiency Compared to Sequential Baseline (Sequential = 1.0)

Variant	Inference Speedup ↑	Latency Reduction ↑	Throughput Gain ↑
Sequential	1.00	1.00	1.00
Parallel	0.98	1.00	0.98
VLM	0.25	0.25	0.25
YOLO + HITL	1.12	1.08	1.12
OWL-ViT + HITL	**6.12**	**6.17**	**6.12**

5.1.3 Maintainability Results

We analyzed structural maintainability using three complementary perspectives: coupling (CBO_f^{ML}), graph-based complexity metrics (depth, betweenness, propagation), and the Architectural Coordination Complexity Score, ACCS.

5.1.3.1 Coupling

To compare the coupling characteristics of different execution branches, we apply our adapted metric CBO_f^{ML} (Equation 4.6) to the orchestrator component of the semi-annotation system. Each execution type is treated as a branch B, whose function set F_B includes both the primary dispatch method and all helper functions it invokes within the same file. The weighted code term $\sum w(f, M_i)$ counts total calls to external code modules, while the service term $\sum I(f, S_j)$ records binary dependency on data-access repositories. The per-branch score is the mean across all functions in F_B.

Table 5.4 reports the results. *Hybrid (VLM-assisted)* achieves the highest score (4.0), which is four times higher than the *Human* baseline (1.0). This is because it simultaneously loads three runners (YOLO, VLM, and OWL-ViT) and broadcasts

ten times to `ws_manager`, resulting in significantly more weighted code invocation than the other branches. *Sequential* scores slightly higher than *Parallel* (2.667 vs. 2.400) because it additionally invokes fusion logic, introducing one extra function into F_B with its own external dependencies. *Single-Model* shares the same score as *Parallel* (2.400), as it is internally routed to the same dispatch method with an identical function set. The *Manual* mode obtains the lowest score, as expected, since it bypasses the ML execution path entirely and depends on only a single data service.

Table 5.4: Comparison of CBO_f^{ML} Across Execution Types

Execution Type	$\lvert F_B \rvert$	$\Sigma w/n$	$\Sigma I/n$	$\mathbf{CBO_f^{ML}}$
Hybrid (VLM-assisted)	5	3.80	0.20	4.000
Sequential	6	2.33	0.33	2.667
Parallel	5	2.20	0.20	2.400
Single-Model	5	2.20	0.20	2.400
Manual	1	0.00	1.00	1.000

5.1.3.2 Complexity

The manual architectural variant exhibited lower structural complexity because it contained no AI dependencies or inter-model coordination stages. Among AI-assisted variants, single-model architectural variants achieved lower structural complexity scores due to their shallow dependency structure and reduced coupling.

Table 5.5: Structural workflow complexity metrics across variants

Variant	**N**	**E**	**Depth**	**Avg. Bet.**	**Avg. Prop.**
Manual	1	0	0	0.000	0.0000
YOLO → HITL	2	1	1	0.000	0.5000
OWL-ViT → HITL	2	1	1	0.000	0.5000
Parallel	3	2	1	0.000	0.6667
Sequential	3	2	2	0.1667	1.0000
VLM assisted	4	3	2	0.0417	1.0000

The Sequential variant produced the highest average betweenness value because intermediate stage OWL-ViT acted as stronger dependency bridges between workflow components. In contrast, the VLM-assisted variant distributed dependencies across multiple stages, reducing the concentration of dependency propagation despite having a larger workflow graph.

Table 5.6: Architectural Coordination Complexity Metrics Across Variants

Variant	Avg Bet.	norm(Avg Prop.)	Automation Ratio E_x	Complexity Score
Manual	0.0000	0.0000	0.0000	0.0000
YOLO → HITL	0.0000	0.5000	0.5000	0.3333
OWL-ViT → HITL	0.0000	0.5000	0.5000	0.3333
Parallel	0.0000	0.3333	0.6667	0.3333
Sequential	0.1667	0.5000	0.6667	0.4445
VLM assisted	0.0417	0.3333	0.7500	0.3750

The architectural coordination complexity analysis revealed substantial structural differences between the evaluated variants as shown in Table 5.6. The Manual variant produced the lowest complexity score because it contained no automated orchestration stages or dependency propagation between components.

The YOLO + HITL and OWL-ViT + HITL variants exhibited relatively low complexity due to their simple linear structures with minimal coordination requirements. Similarly, the Parallel variant maintained moderate complexity despite integrating multiple models because the workflow branches operated independently before converging at the HITL stage.

In contrast, the Sequential variant produced the highest coordination complexity score. This behavior was primarily associated with stronger dependency propagation and increased workflow centralization, since downstream stages directly relied on outputs generated by upstream models.

The VLM-assisted variant also produced relatively high complexity due to the additional semantic prompt generation stage and the increased coordination required between VLM, OWL-ViT, and human validation components.

Overall, the results suggest that increasing workflow automation and integrating additional AI coordination stages may increase orchestration complexity even when improving detection flexibility and semantic capabilities.

In addition to complexity metrics, the maximum fan-in and fan-out analysis further highlights structural differences between variants as shown in Table 5.7. Maximum fan-in and fan-out values were used to capture the most coordination-intensive workflow stages within each orchestration structure. This emphasizes potential integration bottlenecks and dependency concentration points rather than overall graph connectivity.

The Manual and Single-model variants produced the lowest fan-in and fan-out values because they contained minimal branching and limited inter-stage dependency coordination. The Sequential variant maintained the same maximum fan-in and fan-out values as the single-model variants because the architecture primarily in-

troduced staged dependency propagation rather than parallel convergence between workflow stages. The Parallel variant increased fan-in at the HITL stage because multiple model outputs converged into a single review stage. The VLM-assisted variant similarly increased fan-in because both the YOLO and OWL-ViT branches converged at the HITL stage. Although the variant did not substantially increase fan-out values, the additional coordination stages increased orchestration complexity and dependency management requirements. Higher fan-in and fan-out values indicate increased integration complexity and stronger coupling between workflow stages.

Table 5.7: Maximum fan-in and fan-out metrics across variants

Variant	Max Fan-in	Max Fan-out
Manual	0	0
YOLO → HITL	1	1
OWL-ViT → HITL	1	1
Parallel	2	1
Sequential	1	1
VLM assisted	2	1

5.1.4 Human Intervention Results

This section reports quantitative measures of human effort and annotation consistency across the six architectural variants. We evaluate time on task, correction rate, acceptance rates (pure and broad), AI coverage, and inter-annotator agreement to capture both the efficiency and reliability of human-AI collaboration.

5.1.4.1 Time on Task, Correction Rate, Acceptance Rate, and AI coverage

We conducted the experiment by comparing six annotation variants on time, correction rate, acceptance rate, and AI coverage.

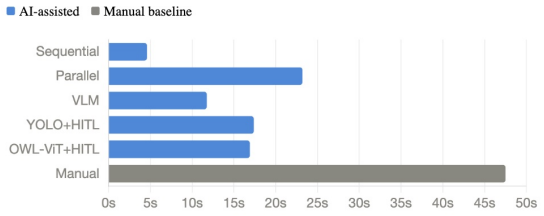
Table 5.8: AI Suggestion Coverage and Acceptance Rates Across Variants

Variant	Total Boxes	AI Suggestions	AI Coverage	Accepted (Pure)	Edited → Kept	Pure AR	Broad AR	CR
Sequential	325	237	72.9%	192	24	81.0%	91.1%	40.9%
Parallel	356	253	71.1%	172	28	68.0%	79.1%	51.7%
VLM	330	256	77.6%	200	11	78.1%	82.4%	39.4%
YOLO+HITL	335	236	70.4%	169	29	71.6%	83.9%	49.6%
OWL-ViT+HITL	346	17	4.9%	15	2	88.2%	100.0%	95.7%
Manual	289	0	—	—	—	—	—	100%

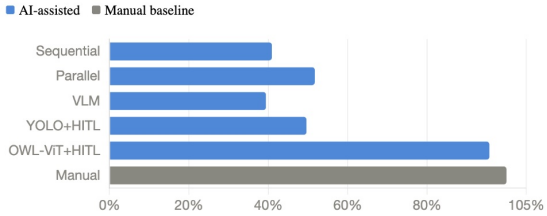
Note: Pure AR = accepted ÷ AI suggestions; Broad AR = (accepted + edited) ÷ AI suggestions; AI coverage = AI suggestions ÷ total boxes; CR $= 1 - \frac{accepted}{total\ boxes}$.

Table 5.8 summarises the human intervention metrics across the six annotation

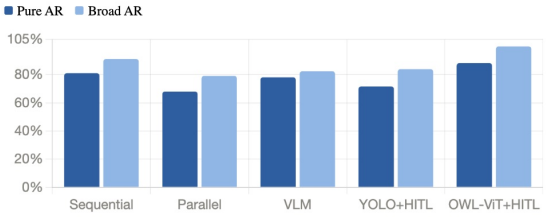
variants. We report Correction Rate (CR), average time on task, and a disaggregated breakdown of how each final bounding box originated (AI accepted as-is, AI-suggested but edited, or human-added from scratch). Because CR alone conflates two distinct failure modes, AI suggestions that were wrong and rejected, versus objects the AI missed entirely. We additionally distinguish between a *pure acceptance rate* (accepted / AI suggestions) and a *broad acceptance rate* ((accepted + edited) / AI suggestions).



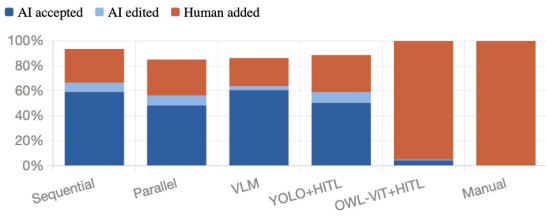
(a) Average time per image



(b) Correction rate



(c) AI suggestion acceptance rates



(d) AI coverage

Figure 5.2: Overview of the human intervention metrics.

Time on task. The sequential variant was the fastest by a substantial margin, with annotators spending an average of 4.6 s per image, less than one-quarter of the manual baseline (47.5 s). VLM assisted variant followed at 11.8 s. The remaining AI-assisted variants clustered between 16.9 and 23.2 s, all still well below manual. OWL-ViT+HITL (16.9 s) was faster than manual despite its near-total reliance on human-drawn boxes, which is addressed further in the Discussion.

Correction rate. VLM achieved the lowest CR (39.4%), meaning that 60.6% of all final boxes were AI predictions accepted without any modification. Sequential variant was close behind (CR = 40.9%). YOLO+HITL (49.6%) and Parallel (51.7%) variants required human correction for approximately half of all boxes. OWL-ViT+HITL variant had a CR of 95.7%, approaching the manual baseline of 100%.

Acceptance rates on AI suggestions. Sequential variant achieved the highest pure AR (81.0%), indicating that 4 in 5 of its proposals required no editing. VLM's pure AR was 78.1% with a narrow gap to its broad AR (82.4%), suggesting that even the edited suggestions needed only minor adjustment. Parallel variant had the lowest pure AR (68.0%) and the largest volume of deleted suggestions (n=53) because we ran two detectors in parallel and got two results. OWL-ViT+HITL appears to achieve a perfect broad AR (100%) but this is just an illusion caused by

the very small suggestion pool (n=17), as described below.

AI coverage. The AI coverage reveals the most operationally informative picture. In the sequential, VLM, YOLO+HITL, and Parallel variants, between 56% and 67% of final boxes were of AI origin (accepted or edited), with the remaining 22–30% added by human annotators to cover AI misses. By contrast, OWL-ViT+HITL produced only 17 AI suggestions across the entire session, meaning 95.1% of its final boxes were drawn manually. The manual baseline, as expected, is 100% human-origin.

5.1.4.2 Inter-Annotator-Agreement

We conducted inter-annotator agreement analysis to evaluate annotation consistency across architectural design patterns and to investigate whether architectural structure influenced annotator behavior.

Table 5.9: Inter-annotator agreement across variants

Variant	A1 vs A2	A1 vs A3	A2 vs A3	Fleiss' Kappa
Manual	0.7912	0.8696	0.8118	0.8247
YOLO → HITL	0.8474	0.8766	0.8485	0.8577
OWL-ViT → HITL	0.7543	0.8472	0.7978	0.8001
Parallel	0.8657	0.8748	0.8396	0.8603
Sequential	0.8718	0.8615	0.8103	0.8480
VLM assisted	0.8865	0.8658	0.8346	**0.8624**

The inter-annotator agreement analysis demonstrated consistently high agreement across all architectural variants. Fleiss' Kappa values ranged between 0.8001 and 0.8624, which indicates strong agreement between annotators for all evaluated workflows.

According to the interpretation guidelines proposed by Landis and Koch [70], Kappa values between 0.61–0.80 indicate substantial agreement, while values above 0.80 indicate almost perfect agreement.

The OWL-ViT-hitl workflow produced the lowest agreement score (0.8001), suggesting that open-vocabulary detections introduced greater ambiguity during annotation review.

In contrast, the VLM-assisted variant achieved the highest overall agreement (0.8624), followed closely by the Parallel AI workflow (0.8603) and the YOLO-assisted workflow (0.8577). The results indicate that variants combining multiple information sources may help annotators make more consistent labeling decisions.

Overall, the agreement values suggest that the ontology design and annotation interface supported reproducible annotations across annotators despite differences in

workflow structure.

Despite the generally high agreement scores, some object categories still produced noticeable disagreement between annotators. In particular, the "pile of rocks" category was interpreted inconsistently across participants, even when annotation guidelines had been discussed beforehand. Participants differed in how they defined object boundaries, grouping behavior, and the extent of partially visible rock formations.

5.2 User Study

The user study assessed participants' perceptions of effort, understandability, confidence, and efficiency, as well as their qualitative feedback on each variant.

5.2.1 Quantitative Results

After completing the annotation tasks, participants rated their agreement with four statements on a 5-point Likert scale (1 = Strongly Disagree, 5 = Strongly Agree). The statements addressed the following aspects:

1. **Effort reduction**: "Using the system reduced my annotation effort."
2. **Understandability**: "The architectural design patterns were easy to understand."
3. **Confidence**: "I feel confident about the AI-generated annotations."
4. **Efficiency**: "The system was efficient."

Eleven participants completed the survey. One participant provided the response "it depends" for Statement 3, which was treated as missing data for that item only. All other responses were complete. Table 5.10 presents descriptive statistics for each survey item, while Figure 5.3 visualizes the results.

Table 5.10: Summary of Participant Ratings

Statement	Mean	Median	Mode	SD	Range
Reduced annotation effort	4.00	4	4	1.15	2–5
Easy to understand	4.27	5	5	0.96	3–5
Confident about AI annotations	3.35	3.25	3	0.69	2–4
System was efficient	3.91	4	4	0.94	2–5

Note: For Statement 3, $N = 10$ due to one missing response.

Participants rated the system most favourably on design pattern comprehensibility ($M = 4.27$, $SD = 0.96$), suggesting that the architectural variants were generally easy to understand.

Perceived effort reduction also received a high mean score ($M = 4.00$, $SD = 1.15$), although the relatively large standard deviation indicates notable disagreement among participants. Two participants rated this item as 2, reflecting cases where automation did not substantially reduce their individual workload.

System efficiency was rated similarly positively ($M = 3.91$, $SD = 0.94$), indicating that the overall workflow was perceived as reasonably performant.

Confidence in AI-generated annotations received the lowest rating ($M = 3.35$, $SD = 0.69$), suggesting that while participants found the system useful, trust in the correctness of automated outputs remained moderate and somewhat cautious across the participant group.

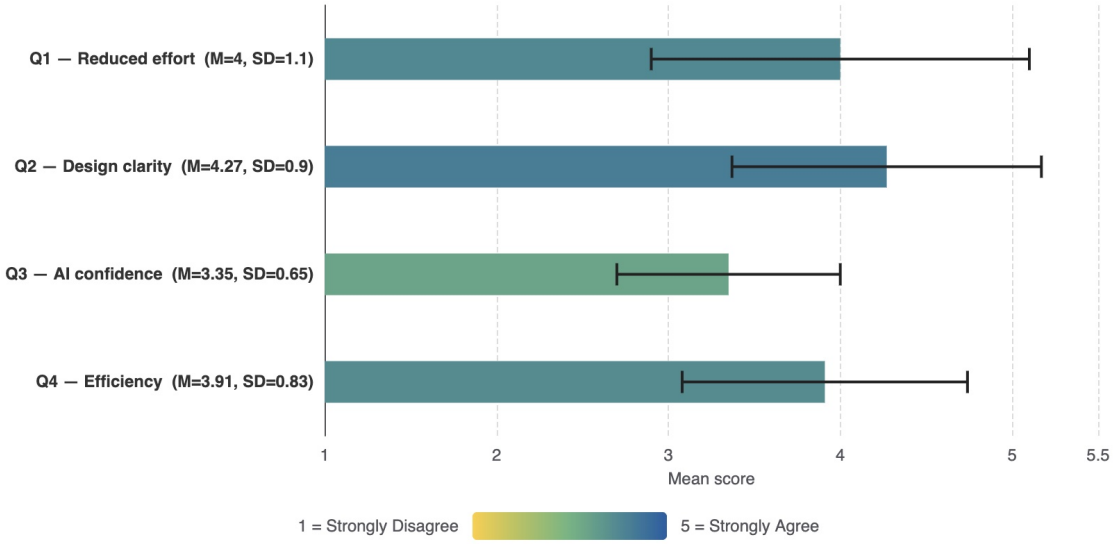


Figure 5.3: Summary of Participant Ratings

5.2.2 Qualitative results

The thematic analysis revealed several recurring themes related to workload, usability, trust in AI-generated annotations, cognitive effort, and maintainability across the evaluated architectural variants.

1. AI-assisted annotation significantly reduced manual workload. All participants agreed that the system substantially reduced annotation workload compared to fully manual labeling. Several participants estimated that the workload reduction was around 80%, despite occasional missing detections or the need for additional corrections.

Participants noted that all automated variants reduced effort to some extent, although manual refinement was still required in some cases.

"It missed a few details that we needed to add, but it reduced the workload by 80%." (P3)

"In all cases, you sometimes needed to add a little bit extra annotation, but they all reduce the workload by quite a bit." (P3)

The findings suggest that AI-assisted annotation can effectively support human annotators by reducing repetitive manual operations while still allowing human oversight for quality assurance.

2. Sequential assistance produced the lowest cognitive and manual workload. Participants consistently reported that the sequential variant required the least manual effort and felt cognitively easier to use compared to manual and parallel variants.

The manual workflow was perceived as the most demanding, while the parallel variant introduced additional effort because users needed to compare and choose between multiple model outputs.

"Manual requires the most effort." (All participants)

"The sequential one was the easiest to work with." (P3)

"Parallel one, having too many options, kind of makes it confusing." (P7) (P11)

"When we had multiple models, how to choose? Am I rejecting it because it's a duplicate?" (P6)

The sequential workflow simplified decision-making by presenting users with a clearer accept/reject process instead of multiple competing alternatives. This indicates that reducing decision complexity can lower cognitive load during annotation tasks.

The findings suggest that architectural simplicity at the interaction level may reduce cognitive burden even when the underlying AI functionality becomes more advanced. Sequential workflows reduced the number of simultaneous decisions presented to the user, allowing participants to focus on validation rather than comparison between competing model outputs.

While participants generally preferred the sequential workflow because of its simplicity and reduced cognitive effort, some also noted that workflows relying heavily on previous model outputs may propagate earlier detection mistakes into later stages.

3. Sequential design achieved a better balance between automation and human control. Participants generally perceived the sequential system as offering the most effective balance between automated assistance and human oversight.

While fully automatic outputs still required occasional correction, the sequential setup allowed users to maintain control without becoming overwhelmed by multiple simultaneous predictions.

4. Parallel outputs can be helpful, but also increased visual complexity and decision burden. For the parallel variant, users observed that one model

sometimes detected missing objects with low confidence, while another produced tighter but incomplete bounding boxes.

“One tool finds it, but with very low confidence, and the other detects it, but the boundary box is narrower.”

This suggests that parallel may improve practical usability by combining detection coverage with manageable user interaction.

Although participants recognized the potential benefits of combining multiple models, many found the parallel variant visually crowded and harder to manage.

The need to compare overlapping outputs increased interaction complexity and slowed the workflow.

“Parallel crowded it.” (P5)

“Instead of having a lot of alternatives...” (P5)

Participants reported that overlapping bounding boxes and multiple candidate predictions made it difficult to quickly identify correct annotations, especially when objects were visually similar.

These findings indicate that presenting too many simultaneous AI suggestions may negatively impact usability despite improving model coverage.

Although sequential workflows were generally perceived as easier to use, participants also acknowledged that parallel workflows could improve object coverage by allowing models to complement each other. This demonstrates an important trade-off between interaction simplicity and detection diversity.

5. Accuracy was considered the highest design priority. When discussing design priorities, participants most frequently ranked accuracy as the most important factor, followed by performance speed and reduction of manual effort.

However, priorities varied depending on usage context. For example, participants stated that fast performance becomes more important in real-time feedback scenarios, whereas maintainability and ease of use become more critical in long-term production systems.

“It depends on context.” (P3)

“If the requirement is more like direct feedback on the model, then I would say performance speed is probably the most.” (P3)

Several participants also emphasized that ease of use only needs to reach a “good enough” threshold, whereas accuracy strongly affects trust and adoption. (P4) (P6) (P11)

Participants consistently emphasized that no single quality attribute should dominate system design decisions universally. Instead, priorities depended heavily on deployment context, operational scale, and workflow requirements. Real-time environments increased the importance of latency and responsiveness, whereas long-term industrial systems emphasized maintainability, scalability, and ease of integration.

6. Participants expressed willingness to use the system in real annotation workflows. Most participants stated that they would consider using the system in actual annotation processes, particularly because of its workload reduction and supportive automation.

Despite usability issues and occasional inaccuracies, participants generally viewed the system as practical and beneficial for real-world tasks.

This suggests that users perceived the system as sufficiently valuable even when automation was imperfect.

7. AI assistance improved efficiency but introduced risks of over-reliance. Several participants stated that while AI-generated annotations significantly reduced manual effort, they also noticed a tendency to trust AI outputs without fully verifying them.

Participants reported that once AI suggestions appeared visually plausible, they became less likely to carefully inspect all detections.

"Eventually you just get so used to it that you just accept everything." (P8)

"People tend to not adjust when the result looks relatively good." (P4)

This indicates that while AI assistance improves efficiency, it may also weaken active human verification behavior over time. In addition, participants suggested that prolonged exposure to plausible AI outputs could reduce active verification behavior, potentially increasing the risk of unnoticed annotation errors over time.

All of these findings highlight the continued importance of human-in-the-loop validation in AI-assisted annotation systems, particularly in workflows where plausible but imperfect predictions may otherwise remain unverified.

8. Participants emphasized maintainability and modularity considerations for production deployment. Participants repeatedly highlighted that production-ready AI annotation systems require maintainable and modular architectures capable of supporting multiple models, domain adaptation, and scalable data pipelines.

Several participants emphasized:

- model specialization for different domains,
- modular model replacement,

- scalable data ingestion pipelines,
- and long-term maintainability.

“The system should be maintainable, so we should have a module for that.” (P9)

“Different regions may require specialized models.” (P9)

“You don’t want to run five different models permanently.” (P9)

These observations support the importance of architectural modularity in ML-enabled systems and align with the study’s focus on maintainability trade-offs in human-AI workflows.

9. Participants highlighted challenges related to annotation ambiguity and ground-truth consistency. Participants also noted that defining correct annotations was not always straightforward, particularly in complex scenes or ambiguous object boundaries. Moreover, several participants mentioned that annotation quality depends heavily on ontology definitions, labeling consistency, image resolution, and shared interpretation of object categories.

Some participants also emphasized that disagreement between annotators may emerge even when AI assistance is available, especially in edge cases involving occlusion, low visibility, or uncertain object boundaries. (P1)

These findings suggest that challenges related to ground-truth definition remain important in AI-assisted annotation workflows and may influence both annotation consistency and evaluation reliability.

6

Discussion

This chapter discusses the findings of the study in relation to the research questions and the broader context of ML-enabled human-AI systems. The discussion integrates quantitative results and qualitative user study findings to examine functional correctness, performance efficiency, maintainability, human oversight, and interaction behavior in AI-assisted annotation systems.

6.1 Impact of Architectural Design Patterns on System Quality Attributes(RQ1)

This section discusses how different architectural design patterns influenced key system quality attributes, including functional correctness, performance efficiency and maintainability based on the quantitative and qualitative findings obtained from the experiments.

6.1.1 Functional Correctness

The results showed that architectural design patterns influence not only usability and maintainability, but also functional correctness characteristics such as detection coverage and localization behavior.

The Parallel variant achieved the highest F1-score and recall values, indicating that the combination of multiple model outputs positively improves object coverage and reduces missed detections. This also supported the qualitative findings from the user study, where participants frequently stated that the Parallel variant allowed models to complement each other by detecting different objects or producing alternative bounding boxes.

Although the Parallel variant improved coverage, participants also reported increased cognitive workload caused by overlapping predictions and additional decision-making complexity. This illustrates an important trade-off between detection performance and human usability.

The Sequential variant achieved the strongest overall localization robustness measured by mAP@[0.50,0.95]. One possible explanation is that the staged dependency

structure reduced conflicting predictions and produced more consistent final annotations during human review.

The OWL-ViT-hitl variant achieved a higher recall value, with a substantially lower precision value. This may stem from the open-vocabulary nature of the model, which increases sensitivity to semantically related objects but may also introduce more false positive detections and ambiguous bounding boxes.

The VLM-assisted variant did not significantly outperform the simpler variants in quantitative detection metrics despite introducing additional semantic context. This also showed that increased architectural complexity and semantic guidance do not necessarily translate directly into improved localization accuracy under the evaluated conditions.

The consistently low mAP@0.95 values across all variants further indicated that accurate object localization remained challenging even when object presence detection was relatively successful. This observation aligned with participant feedback describing difficulties with distant objects, overlapping objects, and precise boundary adjustments.

6.1.2 Performance Efficiency

Although the Parallel variant was expected to improve execution efficiency through concurrent model execution, the results showed nearly identical performance compared to the Sequential variant.

One possible explanation is that synchronization overhead, duplicated processing, and increased result management costs offset the expected benefits of parallelization. This showed that introducing additional concurrent AI components does not necessarily translate into proportional performance gains at the system level.

In contrast, the VLM-assisted variant, despite its powerful model capabilities, had the lowest throughput. This indicated that while large visual language models are powerful, their inference costs are too high, severely degrading system-level performance efficiency when used as the primary annotation engine.

The OWL-ViT + HITL and the YOLO + HITL variants outperformed the baseline, which showed that using single-model architectural design patterns is a more effective efficiency strategy than multi-model architectural design patterns.

6.1.3 Maintainability and Structural Complexity

The maintainability analysis showed that workflow structure is strongly influenced by dependency depth, coupling between AI components, and the amount of simultaneous information presented to annotators.

The Sequential variant produced the highest architectural coordination complexity score because modifications in upstream components may propagate through mul-

tiple downstream stages. This increases integration complexity and may require additional debugging, monitoring, and regression testing effort. In contrast, single-model and parallel workflows remained structurally simpler due to their shallower dependency structures and lower dependency propagation.

The fan-in and fan-out analysis further highlighted how architectural dependencies influence maintenance effort. The Parallel and VLM-assisted variants showed increased fan-in at the HITL stage because outputs from multiple AI components converged into a shared human review stage. This increased the coordination requirements for both system integration and human validation.

In addition, the VLM-assisted variant introduced additional orchestration stages and semantic coordination dependencies associated with prompt generation and downstream model interaction. Although these dependencies did not increase the maximum fan-out values, they introduced additional sequential coordination requirements within the workflow architecture.

The coupling analysis further reinforced the maintainability patterns observed in the evaluation of the architectural design pattern structure. Higher coupling scores indicate that a variant depends more heavily on external modules, model coordination, and integration logic, increasing the likelihood that modifications in one component will propagate across the system.

Among all variants, the hybrid (VLM-assisted) branch exhibited the highest coupling complexity due to its multi-model coordination and additional semantic-processing stages. This suggests that maintaining VLM-assisted pipelines may require greater effort in interface synchronization, dependency management, and regression testing when underlying AI components evolve.

In contrast, the Manual branch remained minimally coupled because it operates largely independently from the ML execution stack. This architectural isolation reduces the impact of model-level changes on the human review pipeline and improves independent maintainability.

The comparison between the Sequential and Parallel variants also indicated that maintainability is influenced not only by the number of models involved, but also by how prediction is implemented. Although both variants rely on similar model dependencies, the Sequential variant introduced additional integration complexity through explicit fusion and validation stages. This supported the broader observation that coordination logic itself can become a significant maintenance burden in multi-model annotation systems.

The results also presented that maintainability in AI-assisted annotation systems should not only be evaluated from a software architecture perspective but also from a human-centered interaction perspective. Variants that present multiple simultaneous predictions increased visual clutter and decision complexity during annotation. Participants in the user study frequently described the parallel variant as cognitively more demanding despite its improved object coverage.

Conversely, sequential variant simplified interaction by reducing the number of simultaneous decisions presented to the user. This allowed annotators to focus primarily on validation rather than comparison between competing model outputs, which also suggested that architectural simplicity at the interaction level may improve long-term usability and operational sustainability even when the underlying AI functionality becomes more advanced.

Finally, the VLM-assisted variant demonstrated that introducing additional semantic-processing stages may improve contextual understanding while also increasing external dependency management complexity due to prompt-generation and model-version dependencies. Overall, the findings showed that reduced dependency propagation and simpler interaction structures may improve maintainability in human-AI annotation systems.

6.2 Trade-offs Between Functional Correctness, Maintainability, Performance, and Automation(RQ2)

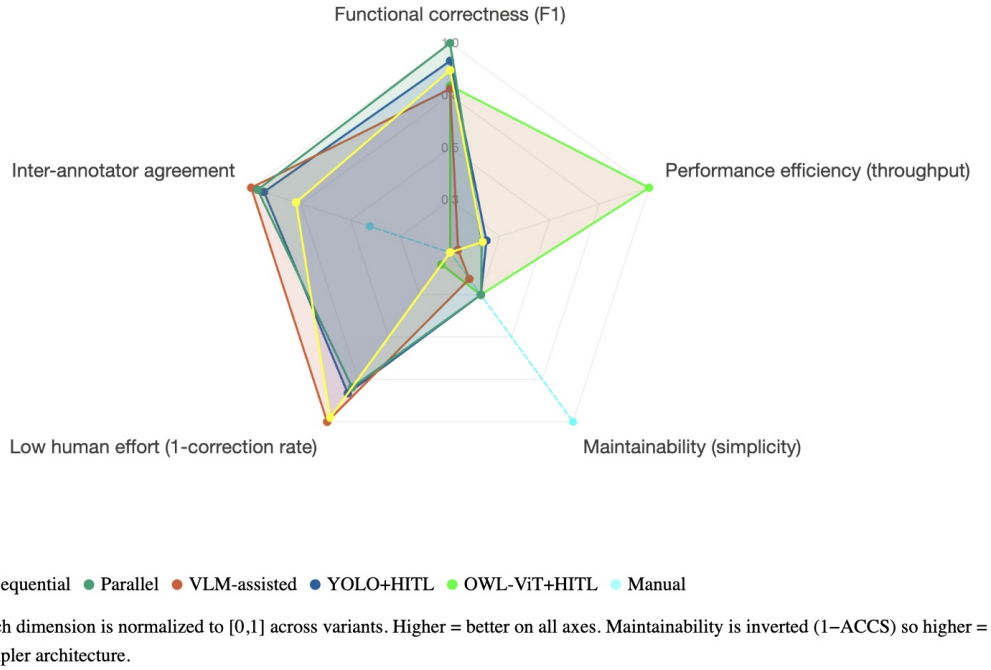


Figure 6.1: Overview of Trade-offs

As illustrated in Figure 6.1, no single architectural design pattern simultaneously optimized all evaluated quality attributes. Instead, each pattern emphasized a distinct operational priority, revealing inherent trade-offs across functional correctness, performance efficiency, maintainability, human intervention, and inter-annotator agreement.

1. **Detection coverage versus interaction complexity.** The parallel variant

achieved the highest F1-score and recall, confirming that combining heterogeneous model outputs improves object coverage. However, this gain came at a cost: overlapping proposals increased cognitive load, correction rates, and per-image review time. Participants frequently described the parallel interface as visually crowded and decision-heavy. This trade-off is particularly relevant in safety-critical perception tasks where missed detections carry high downstream risk, but it implies that deploying parallel designs is only justified when the operational cost of a false negative substantially outweighs the overhead of reviewing redundant proposals.

2. Architectural simplicity versus detection capability. The sequential variant achieved competitive F1, the strongest localization robustness (mAP@[0.50,0.95]), the fastest per-image review time, and the highest pure acceptance rate (81.0%). However, it achieved this at the cost of the highest structural maintenance complexity (ACCS = 0.4445), because its staged dependency structure introduced tighter coupling between upstream and downstream components. Modifications to an early detection stage may propagate through fusion and validation stages, increasing regression testing burden. This illustrated that architectures which reduce interaction complexity for the annotator may simultaneously increase integration complexity for the system developer.

Participants additionally described the sequential variant as easier to understand because it simplified interaction into a more validation-oriented process rather than requiring comparisons between multiple simultaneous predictions.

3. Inference efficiency versus practical annotation utility. The OWL-ViT+HITL variant achieved the highest system-level throughput, a 6.12 × speedup over the sequential baseline. However, its AI coverage was only 4.9%, meaning that nearly all final bounding boxes were drawn manually. The apparent efficiency advantage therefore did not translate into reduced human effort. Conversely, the VLM-assisted variant had the lowest throughput due to high inference cost, yet maintained AI coverage above 77% and the lowest correction rate. These results showed that system-level performance metrics and annotation-level human effort metrics can diverge substantially, and that optimizing inference speed alone does not guarantee operational efficiency in human-in-the-loop workflows.

Taken together, these findings suggest that architectural decisions in AI-assisted annotation systems should be evaluated across multiple complementary dimensions rather than any single metric. The practical value of a workflow depends not only on model performance, but on how effectively model outputs are coordinated, presented, and verified within the human-AI interaction process. Increasing the number of AI components improved detection diversity but also raised interaction complexity, dependency propagation, and cognitive workload, gains and costs that aggregate performance metrics alone cannot capture. These findings further suggested that increasing architectural complexity or adding additional AI components does not automatically improve overall workflow effectiveness.

6.3 Impact on Human Intervention and User Experience(RQ3)

This section discusses how different architectural design patterns influenced human interaction behavior, perceived usability, cognitive workload, and annotation consistency during AI-assisted annotation tasks. The discussion combined qualitative user study findings, interaction observations, and inter-annotator agreement analysis to examine the role of human oversight in different architectural variant structures.

6.3.1 Perceived Workload and Cognitive Effort

A key insight from the user study was that interaction design strongly influences perceived usability and cognitive workload. Although participants generally found the interface intuitive, usability issues such as overlapping bounding boxes, unstable zooming behavior, and the absence of undo functionality negatively affected the annotation experience. These issues became especially problematic in the parallel condition, where multiple model outputs increased visual clutter and decision complexity.

6.3.2 Automation Bias and Human Oversight

At the same time, the user study findings also highlighted an important trust-related challenge in AI-assisted systems. While participants appreciated the high level of automation, several expressed concerns about becoming overly reliant on AI outputs over time. Because annotators sometimes became overly dependent on the AI output and paid less attention to validating background objects or less salient instances. In some cases, participants appeared less willing to perform additional verification work when the AI prediction seemed sufficiently accurate, weakening the "human-in-the-loop" process. This raised the possibility that excessive trust in AI assistance may unintentionally reduce annotation accuracy by encouraging passive confirmation behavior rather than active inspection. Furthermore, the findings reinforced that human annotation itself should not necessarily be treated as a perfect ground truth, as annotators can still miss objects, misclassify categories such as poles or cones, or produce imprecise bounding boxes, particularly when they lack annotation experience. These observations further supported the importance of balancing AI assistance with active human oversight and continuous annotator training, especially in safety-critical domains.

6.3.3 Annotation Consistency Across Architectural Variants

The inter-annotator agreement results showed that architectural variant design may influence annotation consistency in addition to annotation effort.

The relatively lower agreement observed in the OWL-ViT-hitl variant may stem from the open-vocabulary nature of the model, which can produce less predictable detections and introduce greater ambiguity during human review.

In contrast, variants combining supervised detections with additional contextual information generally achieved higher agreement scores. In particular, the VLM-assisted variant produced the highest Fleiss' Kappa value, suggesting that semantically enriched prompts may improve the consistency of human interpretation.

The Parallel and Sequential variants also achieved strong agreement values, indicating that multi-model assistance did not negatively affect annotation consistency despite increasing workflow complexity.

Interestingly, several AI-assisted variants achieved slightly higher agreement than the fully manual annotation. This may indicate that AI-generated suggestions acted as implicit guidance that reduced subjective interpretation differences between annotators.

However, the agreement analysis was based on object-class presence annotations rather than exact bounding-box overlap. Therefore, the reported agreement values primarily reflect semantic consistency and not localization precision.

An additional observation from the user study concerns the role of AI assistance in establishing annotation consistency. AI-generated predictions appeared to provide annotators with an implicit standard regarding object boundaries, object size, and annotation scope. This helped reduce variability between annotators and may lower the amount of training required for inexperienced users. In manual annotation, participants often differed substantially in annotation behavior. Some annotators tended to over-annotate in pursuit of higher precision, while others overlooked less obvious objects or annotated with less detail. These differences showed that human annotation quality is strongly influenced by individual experience, attention to detail, and personal interpretation of annotation guidelines. Consequently, manual annotation typically requires extensive training to achieve consistency across annotators.

The qualitative findings also indicated that annotation quality is influenced not only by architectural design patterns, but also by the inherent ambiguity of defining ground truth in perception datasets. Even with AI assistance, annotators may interpret object boundaries or ontology categories differently in complex scenes. All these observations indicated that certain object classes inherently contain semantic ambiguity that cannot be fully eliminated through workflow design alone.

Moreover, annotation consistency in perception datasets is influenced not only by AI assistance and architectural design patterns, but also by the clarity and subjectivity of ontology definitions.

Overall, improvements in variant automation alone may not fully eliminate annotation inconsistency in industrial perception systems.

6.3.4 Human Intervention Across Architectural Design Patterns

The results show clear differences in how the architectural design patterns distribute human intervention during annotation.

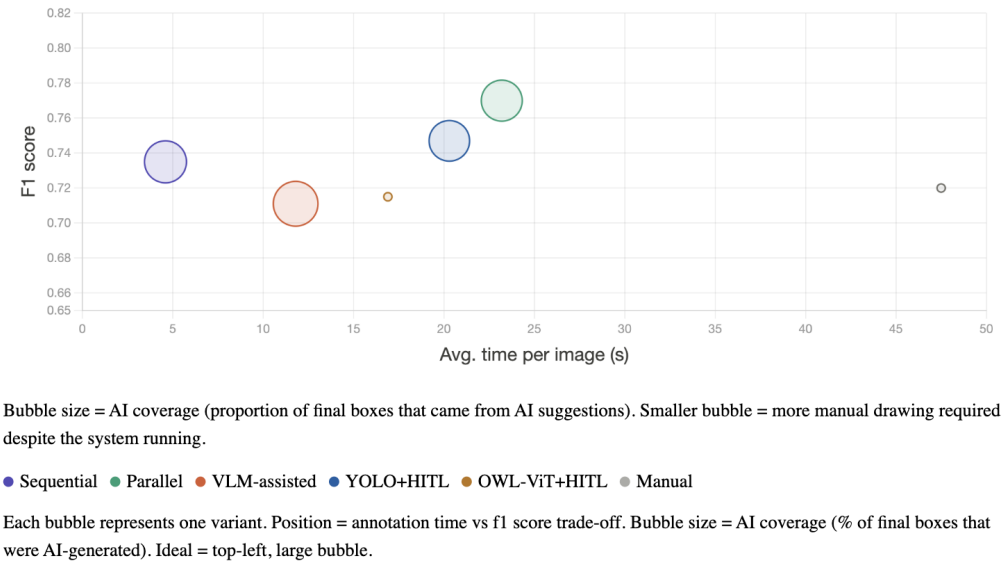


Figure 6.2: Comparison of Annotation Quality and Effort

Figure 6.2 shows the average annotation time against F1 score for each variant, with bubble size encoding AI coverage, the proportion of final bounding boxes that originated from AI suggestions rather than manual drawing. This three-dimensional representation reveals patterns that aggregate metrics alone obscure.

The Sequential workflow occupied the most favorable position: the shortest annotation time, a competitive F1 score, and a large bubble reflecting 72.9% AI coverage. Its structured, staged output allowed annotators to work in a confirmation-oriented mode, reducing the need for repeated visual search and enabling high acceptance rates with minimal correction overhead.

The VLM-assisted workflow clustered nearby in time with similarly high AI coverage and the lowest correction rate overall. The narrow gap between its pure and broad acceptance rates indicates that incorrect predictions typically required only minor spatial adjustments rather than complete replacement, shifting human effort from drawing toward lightweight verification.

The Parallel workflow sat further right on the time axis despite similar AI coverage to Sequential. The additional review time reflects the cost of filtering redundant and overlapping proposals from two concurrent detectors. Although the architecture achieved the highest F1, the increased verification burden reduced much of its practical advantage in annotation throughput. This suggests that parallel designs

are most justified in workflows where the operational cost of a missed detection is extremely high and the marginal cost of reviewing additional proposals is low.

The most striking case was OWL-ViT + HITL. Although it achieved the highest system-level inference throughput, its bubble was nearly absent, AI coverage of only 4.9% meant that annotators manually drew 95% of all final boxes. The system's high throughput did not translate into reduced human effort; annotation behavior closely resembled the fully manual baseline. This dissociation between inference efficiency and annotation utility illustrates that system-level performance metrics can be misleading indicators of human intervention in HITL workflows, and that AI coverage is a more operationally meaningful measure of automation benefit.

6.4 Implications for Human-AI Annotation System Design

The findings of this study suggest that architectural variant design plays a critical role in determining not only technical system behavior, but also human interaction quality in AI-assisted annotation systems.

The study further highlights that ontology design and annotation guidelines play an important role in annotation consistency. Organizations deploying AI-assisted annotation systems should therefore establish clear labeling policies and shared annotation standards to reduce ambiguity and improve inter-annotator agreement.

One important implication is that increasing the number of AI components does not necessarily improve overall usability. Although multi-model variants may improve object coverage and contextual understanding, they can also increase interaction complexity, cognitive workload, and maintenance effort. In particular, variants presenting multiple simultaneous predictions required annotators to compare overlapping outputs, increasing visual clutter and decision ambiguity.

The results further indicate that architectural simplicity at the interaction level may be as important as model capability itself. Sequential workflows generally produced lower perceived cognitive workload and higher annotation consistency because they reduced the number of simultaneous decisions presented to annotators. This suggests that human-AI systems should prioritize interaction clarity and manageable information flow in addition to detection performance.

Another implication concerns the role of AI assistance in standardizing annotation behavior. AI-generated suggestions appeared to function as implicit guidance that reduced subjective interpretation differences between annotators. This may improve annotation consistency and reduce training requirements for inexperienced users. However, the findings also highlight the importance of preserving active human verification, since excessive reliance on AI outputs may weaken critical inspection behavior. The findings suggest that maintaining active human oversight remains important even in highly automated annotation workflows. Systems should therefore

support efficient verification rather than fully removing human review from the annotation process.

The maintainability analysis further suggests that workflow maintainability in AI-assisted systems should be evaluated from both software-engineering and human-centered perspectives. Dependency propagation, external model coupling, and interaction complexity may all influence long-term operational sustainability. As AI workflows become increasingly modular and multi-model oriented, architectural decisions may directly affect debugging effort, regression testing complexity, and scalability. User study participants also emphasized the importance of modular and maintainable architectures capable of supporting model replacement, domain adaptation, and scalable deployment across different operational contexts.

Finally, the study highlights the importance of context-aware system design. Different deployment scenarios may prioritize different quality attributes, including accuracy, performance efficiency, maintainability, or level of automation. Therefore, future human-AI annotation systems should be designed as adaptable architectures capable of balancing automation benefits with human oversight requirements depending on operational needs.

6.5 Synthesis of User Study Findings

The findings indicate that AI-assisted annotation systems can substantially reduce manual workload while still requiring human oversight for quality assurance. Participants consistently perceived the system as useful and practical, particularly because it accelerated repetitive annotation tasks without completely removing human control. This suggests that users are willing to adopt AI-assisted workflows even when automation is imperfect, as long as the system provides meaningful efficiency gains.

Another recurring theme was the importance of context-specific system design. Participants emphasized that priorities such as accuracy, speed, maintainability, and ease of use depend heavily on deployment scenarios. For example, real-time applications require fast feedback, while long-term industrial systems demand robustness, maintainability, and scalable data pipelines. This demonstrates that annotation systems should not only optimize model performance but also consider operational integration and long-term workflow sustainability.

Finally, participants repeatedly emphasized the importance of high-quality training data, domain adaptation, and real-world testing conditions. Concerns regarding weather conditions, sensor limitations, and domain-specific objects suggest that AI-assisted annotation systems must be continuously evaluated and adapted to changing environments. The study therefore reinforces the idea that annotation quality depends not only on model architecture but also on data diversity and human oversight strategies.

6.6 General Architectural Recommendations

The results of this thesis suggest several practical considerations for practitioners designing ML-enabled annotation and perception systems.

The findings also suggest that maintainability in ML-enabled systems extends beyond source-code complexity alone. Maintaining AI-assisted workflows may additionally require model version management, confidence calibration, dataset evolution, prompt engineering, and continuous monitoring of model behavior under changing operational conditions.

Furthermore, workflows integrating VLMs and open-vocabulary models may introduce additional operational challenges because model behavior can become partially non-deterministic across prompts, model versions, or deployment environments. Small changes in prompts or model updates may influence prediction behavior in ways that are difficult to fully reproduce and debug using traditional software engineering practices.

First, increasing the number of AI components may improve detection diversity, but can also introduce higher interaction complexity, dependency propagation, and maintenance effort. Additional model integration should therefore be balanced against usability and operational overhead.

Sequentially dependent workflows may be preferable in scenarios where staged interpretability, validation-oriented interaction, and simpler user review are operational priorities. However, such workflows may also increase dependency propagation because downstream stages rely directly on upstream outputs. In contrast, parallel independent workflows may improve redundancy and detection diversity by allowing heterogeneous AI models to analyze the same input independently. Nevertheless, increasing the number of independently maintained AI components may also increase orchestration complexity, computational resource requirements, and operational maintenance effort due to retraining, synchronization, and model version management requirements. These observations suggest that workflow coordination strategies in ML-enabled systems should be selected based on operational priorities rather than maximizing automation complexity alone.

Second, workflows that simplify user interaction and reduce simultaneous decision complexity may improve human oversight and reduce cognitive workload. In this study, sequential workflows appeared to support more validation-oriented interaction behavior compared to workflows presenting multiple competing predictions simultaneously.

Third, modular architecture and clear separation between workflow orchestration and AI model execution were perceived as important for long-term maintainability and scalability. Participants emphasized the importance of flexible model replacement, scalable data pipelines, and maintainable workflow coordination mechanisms.

Finally, the findings indicate that human oversight remains important even when AI-generated predictions appear visually plausible. Prolonged exposure to convincing AI outputs may reduce active human verification behavior, increasing the risk of unnoticed annotation errors over time.

The combined empirical findings and practitioner observations further suggest several broader architectural considerations for ML-enabled annotation and perception systems.

6.6.1 Design Recommendations for AI/ML-enabled Human-AI Systems

Based on the findings, the following recommendations are proposed for improving AI-assisted annotation systems and supporting real-world deployment.

1. Improve interaction usability

Several usability issues should be addressed to reduce cognitive burden and interaction friction. For example, add undo and rollback functionality for annotation corrections, use clearer color coding and visual differentiation between model outputs, and improve edge-object annotation handling. These improvements may significantly enhance annotation efficiency and reduce user frustration.

2. Enable continuous learning and customization

Participants expressed interest in systems that can evolve based on user feedback and domain-specific data. Future systems should therefore support:

- User-defined classes and categories
- Fine-tuning using corrected annotations
- Online or incremental learning workflows

These capabilities may improve long-term annotation quality and system relevance.

3. Preserve active human validation

To prevent over-reliance on AI predictions, systems should encourage continued human verification:

- Require periodic manual review
- Highlight uncertain or low-confidence detections
- Prevent fully automatic acceptance in critical scenarios
- Design interfaces that maintain user attention and engagement

This is especially important in safety-critical perception systems where annotation errors may propagate into model training.

4. Evaluate models under real-world conditions

Participants repeatedly stressed the importance of testing under diverse operational conditions. Future evaluations should therefore include real-time deployment scenarios, and long-term reliability testing.

Continuous real-world validation is necessary to ensure robustness and trustworthiness in perception systems.

5. Strengthen annotation guidance and human verification workflows.

The study showed that ambiguous object categories and prolonged exposure to plausible AI predictions may reduce annotation consistency and active human verification behavior over time. Practitioners should therefore provide clear ontology definitions, example annotations, and lightweight validation guidance directly within annotation interfaces to support consistent human oversight.

6. Maintain modular separation between workflow orchestration and AI components.

Participants emphasized the importance of flexible model replacement, scalable data pipelines, and maintainable orchestration mechanisms. Modular system architectures may simplify future model integration, experimentation, and long-term maintenance in ML-enabled annotation systems.

7. Continuously monitor annotation consistency and disagreement patterns.

Inter-annotator agreement analysis may help identify ambiguous ontology categories and workflow stages that introduce uncertainty during annotation. Monitoring disagreement rates across object categories can support iterative refinement of annotation guidelines and dataset quality.

6.7 Threats to Validity

This section discusses potential limitations and validity threats associated with the study design, experimental setup, dataset characteristics, and user study methodology. The analysis considers internal, external, construct, and conclusion validity to provide a balanced interpretation of the findings and their generalizability.

6.7.1 Internal Validity

Several aspects of the experimental design limit the internal validity of the participant performance comparison.

First, the quantitative annotation experiments and inter-annotator agreement analysis involved only three annotators, with each annotator evaluating two of the six variants. The limited number of annotators increases the influence of individual differences such as annotation speed, domain familiarity, fatigue tolerance, and tool proficiency. Consequently, part of the observed variation in annotation time and correction behavior may reflect participant-specific characteristics rather than differences between workflow architectures alone.

In addition, two of the three annotators were the thesis authors. Their familiarity with the annotation platform, workflow designs, and study objectives may have influenced annotation and correction decisions compared to independent evaluators. Although all annotators followed the same annotation guidelines and procedures, some degree of researcher participation bias cannot be completely excluded.

Second, all variants were evaluated using the same image set, and workflow order was not counterbalanced across participants. Because participants repeatedly encountered similar scenes during successive annotation rounds, learning effects and workflow familiarization may have influenced performance over time. Participants exposed to later workflow conditions may have required less cognitive effort to interpret the images regardless of the assistance provided by the system. This may partially explain the unusually low annotation time observed for the OWL-ViT+HITL condition relative to the manual baseline.

Third, some architectural variants produced visually different prediction densities and interaction patterns, which may have influenced annotator behavior during review. For example, workflows presenting multiple overlapping predictions may inherently increase interaction complexity compared to confirmation-oriented sequential workflows.

Fourth, the dataset used for fine-tuning and evaluation is limited in absolute size. Although all variants were evaluated on the same dataset and class imbalance reflects a realistic distribution, the small total number of annotated boxes (289–356 per variant) reduces the statistical precision of the quantitative metrics. Differences in correction rates or acceptance rates between variants that are small in magnitude may not be stable under dataset resampling, and the estimates of model performance on very rare categories remain inherently less reliable. However, because all variants share the same data and test distribution, these limitations affect all conditions equally and do not introduce confounding in the within-study comparisons.

6.7.2 External Validity

The user study involved a limited number of participants recruited from a single industrial organization. Therefore, the findings may not generalize to other domains, annotation tasks, or user populations.

However, the purpose of the study was not to derive domain-specific conclusions for mining perception systems alone, but rather to investigate architectural trade-offs

in human-AI collaborative workflows. The observed relationships between workflow structure, cognitive workload, maintainability, and interaction complexity may therefore provide transferable insights for other ML-enabled systems involving human oversight and AI-assisted decision support.

6.7.3 Construct Validity

Construct validity threats exist because concepts such as trust, usability, and cognitive workload are subjective and difficult to measure directly. To mitigate this, the study combines interaction logs, interviews, and Likert-scale survey responses.

In addition, the fine-tuning dataset exhibited class imbalance across ontology categories. Certain object classes, such as "person" and "box", contained substantially fewer annotated instances compared to dominant categories such as "truck" and "pile of rocks". This imbalance may have influenced class-specific detection performance and contributed to variability in recall and localization quality across categories.

Certain object categories, such as "pile of rocks", introduced semantic ambiguity during annotation. Although annotation guidelines were discussed among annotators beforehand, participants still interpreted object boundaries and grouping behavior differently in some cases.

This may have influenced both inter-annotator agreement and evaluation metrics, since the notion of a single objectively correct annotation may not always hold for ambiguous perception scenarios.

Moreover, the proposed Architectural Coordination Complexity Score (ACCS) is an exploratory engineering metric designed specifically for comparative workflow orchestration analysis within this thesis. The weighting scheme, normalization and aggregation strategy are heuristic and were not empirically validated across industrial-scale ML-enabled systems. Therefore, the resulting scores should be interpreted as relative indicators rather than absolute measures of maintainability or software quality.

Furthermore, the maintainability analysis relies on graph-based structural metrics such as workflow depth, dependency propagation, maximum fan-in/fan-out, and betweenness centrality. While these metrics are commonly associated with software complexity and dependency analysis, they do not fully capture operational maintainability factors such as deployment infrastructure, debugging effort, organizational practices, or long-term model lifecycle management.

A further threat to construct validity concerns the interpretation of confidence scores across heterogeneous detection models. Although the same operational confidence threshold (40%) was applied across workflows, YOLO and OWL-ViT produce confidence values using different internal mechanisms. YOLO generates supervised class probabilities, whereas OWL-ViT produces similarity-based scores between image regions and textual embeddings.

As a result, the confidence distributions of the models may not be directly comparable, and the shared threshold may have influenced the number of proposals generated by each model differently. In particular, OWL-ViT generally produced lower-confidence detections, which may have reduced the number of suggestions presented during annotation.

However, because the study focused primarily on architectural variant behavior under consistent operational settings rather than direct model benchmarking, and because all variants incorporated human-in-the-loop validation, the impact of this limitation was partially mitigated. Future work may further investigate model-specific confidence calibration or normalization strategies for heterogeneous multi-model workflows.

6.7.4 Conclusion Validity

The study combines quantitative evaluation metrics with qualitative interview findings. However, the relatively small participant sample size limits the statistical strength of the conclusions and prevents strong generalization of observed differences between architectural variants.

Across the later interviews, participants frequently repeated similar observations regarding workload reduction, usability, automation bias, and architectural trade-offs, suggesting that thematic saturation was approached for several major themes. However, because the participant pool was limited to a single industrial organization and included relatively few annotation experts, complete saturation cannot be claimed and additional themes may emerge in larger or more diverse populations.

In addition, several evaluation metrics, particularly those related to human cognition, usability, and workload, rely on subjective participant perceptions and qualitative interpretation. While thematic analysis was used to identify recurring patterns across interviews, some degree of researcher interpretation remains unavoidable.

Furthermore, some workflow differences were relatively small across certain quantitative metrics, meaning that minor implementation details, dataset composition, or annotation ambiguity may have influenced the observed rankings between variants.

6.8 Future Work

Future work can build on this thesis by conducting a larger-scale user study with more participants and a more balanced experimental design. In particular, future experiments should apply full counterbalancing of variant order and task allocation to reduce learning effects and mitigate potential sequence bias. This would allow a more reliable separation of variant effects from individual differences and temporal learning effects.

Another useful extension would be to explore confidence calibration techniques or score normalization strategies to make outputs from heterogeneous models such as

YOLO and OWL-ViT more comparable. This may improve comparability between heterogeneous model outputs and enable more robust evaluation of multi-model architectural design patterns.

Future research may further investigate adaptive architectural variants, confidence-aware interaction strategies, and domain-specific ontology refinement for large-scale industrial perception systems.

In addition, the proposed ACCS metric could be further validated through empirical studies in real-world deployment settings. Future work could refine the weighting scheme using expert feedback, empirical maintenance data, or data-driven calibration methods, and evaluate the stability of the metric across larger workflow graphs, different orchestration structures, and varying levels of AI component integration.

Future research could also investigate adaptive human-AI workflows that dynamically adjust the level of automation, model assistance, or verification requirements based on annotator behavior, confidence estimates, or task complexity. Such approaches may further improve the balance between annotation efficiency, cognitive workload, and sustained human oversight in industrial perception systems.

Future work may also investigate adaptive human-AI workflows capable of dynamically adjusting interaction complexity based on prediction uncertainty, annotator behavior, or operational context. In addition, further research is needed to better understand reproducibility, explainability, and long-term maintainability challenges introduced by large-scale foundation models and VLM-based perception systems.

The results also suggest that AI-assisted annotation systems may benefit from continuous feedback-loop mechanisms where annotator corrections, disagreement cases, and missed detections are incorporated into future model adaptation and prompt refinement processes.

6.9 Use of Generative AI

Generative AI tools were used as supportive tools during the development and writing process of this thesis. These tools were primarily used for language improvement, readability refinement, grammar correction, and minor implementation troubleshooting during software development and experimental setup.

The core research activities, including the formulation of the research questions, architectural design decisions, dataset preparation, experiment design, model selection, evaluation methodology, result interpretation, and discussion of findings, were conducted by the authors. Generative AI tools were not used to generate experimental results, fabricate data, or replace the authors' own analysis and scientific reasoning.

All AI-generated suggestions and outputs were critically reviewed, modified when necessary, and validated by the authors before inclusion in the thesis. The authors

take full responsibility for the accuracy, originality, and integrity of the submitted work.

7

Conclusion

This thesis investigated how different architectural design patterns influence the functional correctness, maintainability, performance efficiency, and human interaction characteristics of AI-assisted annotation systems.

The results demonstrated that architectural design patterns significantly affect both technical system properties and human cognitive behavior. The Parallel variant improved detection diversity and recall, while the Sequential variant reduced cognitive effort and improved usability through simplified validation-oriented interaction.

The user study further showed that AI-assisted variants reduced annotation effort and helped standardize annotation behavior across annotators. However, participants also reported risks of automation bias and reduced active verification when AI outputs appeared plausible.

The maintainability analysis suggested that dependency propagation, coupling complexity, and interaction structure strongly influence the long-term sustainability of human-AI workflows. Increasing architectural complexity did not necessarily improve practical usability or operational efficiency.

Overall, the findings demonstrate that architectural design patterns significantly influence both technical system qualities and human interaction behavior. No single architectural design pattern consistently outperformed all others across the evaluated quality attributes. Instead, different patterns offered different balances between functional correctness, maintainability, performance efficiency, and human intervention effort. This highlights the importance of selecting architectural structures according to project-specific priorities rather than assuming that increasing automation or architectural complexity will automatically improve overall system effectiveness. The thesis therefore contributes both empirical findings and architectural design insights for the development of future AI-assisted perception annotation systems.

The findings also demonstrate that evaluating AI-assisted systems requires considering not only model performance, but also human cognitive behavior, interaction complexity, and the long-term sustainability of human-AI collaboration workflows.

More broadly, the thesis highlights that designing ML-enabled systems involves chal-

lenges that extend beyond predictive accuracy alone. Architectural decisions influence not only system performance, but also human cognition, operational usability, maintainability, reproducibility, and long-term workflow sustainability. The findings therefore provide practical architectural considerations for practitioners designing future human-AI collaborative perception systems.

Although the study was conducted within the context of perception annotation workflows, the observed trade-offs between automation, coordination complexity, human oversight, and workflow maintainability may also be relevant to broader ML-enabled human-AI systems.

Table 7.1: Summary of research question findings

RQ	**Main Findings**
RQ1	Architectural design patterns influenced functional correctness, maintainability, and performance efficiency. Parallel variant improved recall and object coverage, while sequential variant improved localization consistency and usability.
RQ2	Architectural decisions introduced trade-offs between automation, cognitive workload, maintainability, and interaction complexity. Increasing the number of AI components did not always improve overall workflow effectiveness.
RQ3	AI-assisted variants reduced annotation effort and improved annotation consistency, but also introduced risks of automation bias and reduced active human verification.

Table 7.2: Summary of key strengths across architectural variants

Variant	Functional Correctness	Performance Efficiency	Maintainability	Human Intervention
Manual	–	–	Lowest coupling (CBO=1.0), lowest complexity	Time on task = 47.5 s, highest effort
YOLO+HITL	Balanced F1 (0.747)	3.05 img/s	Mid coupling (CBO=2.4), low complexity (ACCS=0.333)	Time on task = 17.4 s, moderate effort
OWL-ViT+HITL	Highest single-model Recall (0.781)	Highest throughput (16.72 img/s)	Mid coupling (CBO=2.4), low complexity (ACCS=0.333)	Time on task = 16.9 s
Sequential	Highest mAP50-95 (0.294)	2.73 img/s	Mid coupling (CBO=2.667), highest complexity (ACCS=0.445)	Time on task = 4.6 s, lowest annotation time
Parallel	Highest F1 (0.770), Recall (0.784)	2.67 img/s	Mid coupling (CBO=2.4), low complexity (ACCS=0.333)	Time on task = 23.2 s
VLM-assisted	Lowest F1 (0.711), mAP50-95 (0.244)	Lowest throughput (0.67 img/s)	Highest coupling (CBO=4.0), mid complexity (ACCS=0.375)	Time on task = 11.8 s

From a practical perspective, the results suggest that different architectural design patterns may be preferable depending on the objectives of the annotation workflow. For workflows where minimizing cognitive workload and simplifying human validation are primary concerns, sequential architectural variants may be advantageous. For scenarios prioritizing detection coverage and recall, parallel architectural variants may offer benefits despite increased coordination complexity. The VLM-assisted architectural variant demonstrated the potential of semantic reasoning to support perception workflows, while introducing additional orchestration requirements that should be carefully considered during system design.

Overall, the thesis demonstrates that effective ML-enabled annotation systems require balancing automation benefits with maintainability, usability, and sustained human oversight rather than optimizing a single quality attribute in isolation. The results suggest that evaluation of ML-enabled systems should extend beyond model-level performance and incorporate architectural, human-centered, and maintainabil-

ity considerations. As human-AI collaborative systems become increasingly common in safety-critical domains, understanding these interactions at the architectural level will become increasingly important.

Bibliography

[1] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[2] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE, 2019.

[3] Mohamed Zakaria Kamri, Hicham Garrad, and Abdessadek Aaroud. Semi-automatic annotation of mammographic images using ai for enhanced breast cancer detection. In *International Conference on the Global Summit of Digital Health*, pages 341–354. Springer, 2025.

[4] Aman Ratna Kansakar and Ruchika Gupta. A survey on classification of geospatial clustering. In *2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)*, pages 441–446. IEEE, 2023.

[5] Yuji Roh, Geon Heo, and Steven Euijong Whang. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347, 2019.

[6] Bhushan Pande, Kewal Padamwar, Samiran Bhattacharya, Shobhit Roshan, and Mamta Bhamare. A review of image annotation tools for object detection. In *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pages 976–982. IEEE, 2022.

[7] Takuya Kiyokawa, Keita Tomochika, Jun Takamatsu, and Tsukasa Ogasawara. Fully automated annotation with noise-masked visual markers for deep-learning-based object detection. *IEEE Robotics and Automation Letters*, 4(2):1972–1977, 2019.

[8] Beatriz Botella-Gil, Robiert Sepúlveda-Torres, Alba Bonet-Jover, Patricio Martínez-Barco, and Estela Saquete. Semi-automatic dataset annotation applied to automatic violent message detection. *IEEE Access*, 12:19651–19664,

2024.

[9] Erik Eriksson and Joel Olausson. The impact of design patterns on quality attributes in ml-enabled systems-a multivocal study of component models. 2024.

[10] Foutse Khomh and Yann-Gaël Guéhéneuc. Do design patterns impact software quality positively? In *2008 12th European conference on software maintenance and reengineering*, pages 274–278. IEEE, 2008.

[11] Mafizur Rahman, Md Showkat Hossain Chy, and Swapnil Saha. A systematic review on software design patterns in today's perspective. In *2023 IEEE 11th International Conference on Serious Games and Applications for Health (SeGAH)*, pages 1–8. IEEE, 2023.

[12] Lukas Heiland, Marius Hauser, and Justus Bogner. Design patterns for ai-based systems: A multivocal literature review and pattern repository. *arXiv preprint arXiv:2303.13173*, 2023.

[13] Mohamed Abdallah Salem and Ahmed Harb Rabia. Real-time on-the-go annotation framework using yolo for automated dataset generation. *arXiv e-prints*, pages arXiv–2512, 2025.

[14] Daniela Petrelli, Aba-Sah Dadzie, and Vitaveska Lanfranchi. Mediating between ai and highly specialized users. *AI Magazine*, 30(4):95–95, 2009.

[15] Andrii Gamalii, Daniel Górniak, Robert Nowak, Bartłomiej Olber, Krystian Radlak, and Jakub Winter. Semi-automated data annotation in multisensor datasets for autonomous vehicle testing. *arXiv preprint arXiv:2512.24896*, 2025.

[16] Vladislav Indykov, Daniel Strüber, and Rebekka Wohlrab. Architectural tactics to achieve quality attributes of machine-learning-enabled systems: a systematic literature review. *Journal of Systems and Software*, 223:112373, 2025.

[17] Hiroki Tanioka, Tsuyoshi Miura, Kenji Matsuura, and Stephen Karungaru. Development of semi-automatic image annotation using object recognition. In *2023 14th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 546–549. IEEE, 2023.

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[19] Florenc Demrozi, Cristian Turetta, Fadi Al Machot, Graziano Pravadelli, and Philipp H Kindt. A comprehensive review of automated data annotation techniques in human activity recognition. *arXiv preprint arXiv:2307.05988*, 2023.

[20] Alba Bonet-Jover, Robiert Sepúlveda-Torres, Estela Saquete, and Patricio

Martínez-Barco. A semi-automatic annotation methodology that combines summarization and human-in-the-loop to create disinformation detection resources. *Knowledge-Based Systems*, 275:110723, 2023.

[21] Rob Munro. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI.* Manning, 2021.

[22] Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, 56(4):3005–3054, 2023.

[23] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *European conference on computer vision*, pages 728–755. Springer, 2022.

[24] Lukas Picek, Vojtech Cermak, and Marek Hanzl. Zero-shot hazard identification in autonomous driving: A case study on the coool benchmark. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pages 654–663, 2025.

[25] Shashank Shriram, Srinivasa Perisetla, Aryan Keskar, Harsha Krishnaswamy, Tonko Emil Westerhof Bossen, Andreas Møgelmose, and Ross Greer. Towards a multi-agent vision-language system for zero-shot novel hazardous object detection for autonomous driving safety. In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, pages 1511–1518. IEEE, 2025.

[26] Hina Saeeda, Tommy Johansson, Mazen Mohamad, and Eric Knauss. Data annotation quality problems in ai-enabled perception system development. *arXiv preprint arXiv:2511.16410*, 2025.

[27] Jakob Smedegaard Andersen and Walid Maalej. Design patterns for machine learning-based systems with humans in the loop. *IEEE Software*, 41(4):151–159, 2023.

[28] International Organization for Standardization, Software Technical Committee ISO/IEC JTC 1, Information technology. Subcommittee SC 7, and systems engineering. *Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): System and Software Quality Models.* ISO, 2011.

[29] ISO/IEC. Iso/iec 25059: 2023 software engineering–systems and software quality requirements and evaluation (square)–quality model for ai systems.

[30] Md Monjurul Karim, Sangeen Khan, Dong Hoang Van, Xinyue Liu, Chunhui Wang, and Qiang Qu. Transforming data annotation with ai agents: A review of

architectures, reasoning, applications, and impact. *Future Internet*, 17(8):353, 2025.

[31] Yi Peng, Hans-Martin Heyn, and Jennifer Horkoff. Data challenges in ai systems and their solutions: A requirements and ai engineering systematic literature review and comparison. 2025.

[32] Daniel Feitosa, Apostolos Ampatzoglou, Paris Avgeriou, and Elisa Yumi Nakagawa. Investigating quality trade-offs in open source critical embedded systems. In *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, pages 113–122, 2015.

[33] Darius Sas and Paris Avgeriou. Quality attribute trade-offs in the embedded systems industry: an exploratory case study. *Software Quality Journal*, 28(2):505–534, 2020.

[34] Marcio FS Oliveira, Ricardo Miotto Redin, Luigi Carro, Luís da Cunha Lamb, and Flávio Rech Wagner. Software quality metrics and their impact on embedded software. In *2008 5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software*, pages 68–77. IEEE, 2008.

[35] Liang Liang, Haihua Ma, Le Zhao, Xiaopeng Xie, Chengxin Hua, Miao Zhang, and Yonghui Zhang. Vehicle detection algorithms for autonomous driving: A review. *Sensors*, 24(10):3088, 2024.

[36] Andrea Piazzoni, Jim Cherian, Martin Slavik, and Justin Dauwels. Modeling perception errors towards robust decision making in autonomous vehicles. *arXiv preprint arXiv:2001.11695*, 2020.

[37] Ashish Pandharipande, Chih-Hong Cheng, Justin Dauwels, Sevgi Z Gurbuz, Javier Ibanez-Guzman, Guofa Li, Andrea Piazzoni, Pu Wang, and Avik Santra. Sensing and machine learning for automotive perception: A review. *IEEE Sensors Journal*, 23(11):11097–11115, 2023.

[38] Alireza Abbaspour, Tejaskumar Balgonda Patil, B Ravi Kiran, Russel Mohr, and Senthil Yogamani. Dataset safety in autonomous driving: Requirements, risks, and assurance. *arXiv preprint arXiv:2511.08439*, 2025.

[39] Karthik Shivashankar and Antonio Martini. Maintainability challenges in ml: A systematic literature review. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 60–67. IEEE, 2022.

[40] Vladislav Indykov, Rebekka Wohlrab, and Daniel Strüber. Quality trade-offs in ml-enabled systems: a multiple-case study. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, pages 1730–1737, 2025.

[41] Khan Mohammad Habibullah. *Non-Functional Requirements for Machine Learning Systems*. PhD thesis, University of Gothenburg, 2025.

[42] Khan Mohammad Habibullah, D Juan Garcia, Gregory Gay, and Jennifer Horkoff. Maintainability definition, scoping, and measurement for machine learning systems. In *International Conference on the Quality of Information and Communications Technology. Springer*, 2025.

[43] Andi Wahju Rahardjo Emanuel, Retantyo Wardoyo, Jazi Eko Istiyanto, and Khabib Mustofa. Modularity index metrics for java-based open source software projects. *arXiv preprint arXiv:1309.5689*, 2013.

[44] Luan Lazzari and Kleinner Farias. An exploratory study on the effects of event-driven architecture on software modularity. *arXiv preprint arXiv:2110.14699*, 2021.

[45] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends in Computer Graphics and Vision*, 12(1-3):1–308, 2020.

[46] Ziang Guo, Zakhar Yagudin, Artem Lykov, Mikhail Konenkov, and Dzmitry Tsetserukou. Vlm-auto: Vlm-based autonomous driving assistant with human-like behavior and understanding for complex road scenes. *arXiv preprint arXiv:2405.05885*, 2024.

[47] Yue Li, Meng Tian, Dechang Zhu, Jiangtong Zhu, Zhenyu Lin, Zhiwei Xiong, and Xinhai Zhao. Drive-r1: Bridging reasoning and planning in vlms for autonomous driving with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 6708–6716, 2026.

[48] Malsha Ashani Mahawatta Dona, Beatriz Cabrero-Daniel, Yinan Yu, and Christian Berger. Bettercheck: Towards safeguarding vlms for automotive perception systems. *arXiv preprint arXiv:2507.17722*, 2025.

[49] Paul Johannesson and Erik Perjons. Design science. *An introduction to Design Science, Springer*, 10, 2014.

[50] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024.

[51] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th international conference on pattern recognition (ICPR'06)*, volume 3, pages 850–855. Ieee, 2006.

[52] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, 2021.

[53] CS230 Teaching Staff. Splitting into train, dev and test sets. `https://cs230.stanford.edu/blog/split/`, 2025. Accessed: 2026-05-28.

[54] OpenAI. Introducing gpt-5.2. 2025.

[55] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[56] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[57] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

[58] Gemma Team. Gemma 3. 2025.

[59] Khaled Mahmud Sujon, Rohayanti Hassan, Kwonhue Choi, and Md Abdus Samad. Accuracy, precision, recall, f1-score, or mcc? empirical evidence from advanced statistics, ml, and xai for evaluating business predictive models. *Journal of Big Data*, 12(1):268, 2025.

[60] Jingxiu Yao and Martin Shepperd. The impact of using biased performance metrics on software defect prediction research. *Information and Software Technology*, 139:106664, 2021.

[61] Sushant Kumar Pandey, Sivajeet Chand, Jennifer Horkoff, Miroslaw Staron, Miroslaw Ochodek, and Darko Durisic. Design pattern recognition: a study of large language models. *Empirical Software Engineering*, 30(3):69, 2025.

[62] László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. Facing imbalanced data–recommendations for the use of performance metrics. In *2013 Humaine association conference on affective computing and intelligent interaction*, pages 245–251. IEEE, 2013.

[63] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time flying object detection with yolov8. *arXiv preprint arXiv:2305.09972*, 2023.

[64] Mark Newman. Networks: An introduction. *Networks: an Introduction*, 01 2010.

[65] Francesc Comellas and Juan Paz-Sánchez. Reconstruction of networks from their betweenness centrality. In *Workshops on Applications of Evolutionary Computation*, pages 31–37. Springer, 2008.

[66] Steven D Eppinger and Tyson R Browning. *Design structure matrix methods and applications*. MIT press, 2012.

[67] Sallie Henry and Dennis Kafura. Software structure metrics based on information flow. *IEEE transactions on Software Engineering*, (5):510–518, 1981.

[68] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and*

psychological measurement, 20(1):37–46, 1960.

[69] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[70] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[71] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

A

Appendix