

# Multi-Dimensional Readiness Signals for Structured and LLM-Assisted Requirements Inspection

by

Sali Alsafadi

A thesis submitted to the School of  
Graduate and Postdoctoral Studies in  
partial fulfillment of the requirements for  
the degree of

Master of Applied Science in Software Engineering

Faculty of Engineering and Applied Science

University of Ontario Institute of Technology

(Ontario Tech University)

Oshawa, Ontario, Canada

June 2026

Copyright © Sali Alsafadi, 2026

## THESIS EXAMINATION INFORMATION

Submitted by Sali Alsafadi

Master of Applied Science in Software Engineering

Thesis title: Multi-Dimensional Readiness Signals for Structured and LLM-Assisted  
Requirements Inspection

An oral defense of this thesis took place on 19 May 2026 in front of the following  
examining committee:

### Examining Committee

---

|                              |  |
|------------------------------|--|
| Chair of Examining Committee | Dr. Akramul Azim                           |
| Research Supervisor          | Dr. Sanaa Alwidian                         |
| Research Co-Supervisor       | Dr. Khalid Elgazzar                        |
| Examining Committee Member   | Dr. Masoud Makrehchi                       |
| Thesis Examiner              | Dr. Nasim Beigi, Assistant Professor, FEAS |

---

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

# Abstract

Effective software quality assurance requires that development teams inspect large collections of requirements before implementation begins, yet no existing tool helps them decide where to focus their limited inspection effort. The inspection risk of any given requirement is not a property of that requirement in isolation. Rather, it is determined by other factors, including how each requirement relates to others in the requirements specification document, how structurally central it is, how clearly it is written, and how likely it is to change. Such factors are invisible when requirements are examined individually. Moreover, the prevailing practice of collapsing these dimensions into a single weighted score through compensatory aggregation destroys the diagnostic value of each dimension independently.

This thesis proposes and empirically evaluates a two-layer inspection support system. The first layer, Prioritised Requirements Inspection through Signal Metrics (PRISM), computes three collection-level readiness metrics (structural centrality, linguistic specificity, and volatility exposure), and combines them through Pareto dominance-based structuring. Rather than collapsing these orthogonal metrics into a single scalar score, PRISM preserves each dimension independently, yielding a

multi-dimensional inspection ordering. Evaluated on 262 requirements, the metrics are statistically independent, and the ordering aligns significantly with independent human inspection judgment (Spearman  $\rho = -0.478$ ,  $p = 0.0037$ ), outperforming all single-metric baselines.

The second layer, PRISM-Copilot, instantiates the Signal-Conditioned Inspection Prompting (SCIP) architectural pattern, which leverages the validated PRISM metrics to condition an LLM-based inspection assistant by directing each requirement toward the type of scrutiny most relevant to its specific risk profile. Rather than subjecting all requirements to undifferentiated general-purpose inspection prompts, SCIP maps each requirement’s signal profile to a targeted inspection policy, focusing the LLM’s reasoning on the concern dimensions where collection-level evidence indicates the greatest risk. A proof-of-concept study on 50 requirements demonstrates that this targeted signal conditioning causes the LLM to raise change-sensitivity concerns where volatility metrics support them and suppress them where they do not, providing empirical evidence that collection-level metrics can meaningfully steer LLM inspection reasoning.

The two layers constitute a principled, end-to-end inspection support architecture that is context-aware at the collection level rather than the requirement level, establishing a practical foundation for both systematic inspection prioritization and signal-conditioned AI-assisted inspection reasoning in large-scale software engineering practice.

**Keywords:** requirements inspection; collection-level analysis; Pareto dominance; multi-dimensional risk signals; signal-conditioned prompting; large language models; requirements quality

# Declaration of Authorship

- I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.
- I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize Ontario Tech University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Signed: Sali Alsafadi

---

Date: 25 May 2026

---

# Statement of Contributions

This thesis represents original research conducted by the author. The work presented in Chapters 3, 4, 5, and 6 is the product of independent investigation carried out under academic supervision at Ontario Tech University.

Two manuscripts have been prepared from this thesis and submitted for peer review. Both are currently under review at the time of submission of this thesis.

The first manuscript, titled *PRISM: Pareto-Based Structuring for Readiness-Aware Requirements Inspection*, presents the PRISM framework described in Chapter 3 and the evaluation results reported in Section 5.2 of Chapter 5. The second manuscript, titled *Signal-Conditioned Prompting for LLM-Assisted Requirements Inspection*, presents the SCIP architectural pattern and PRISM-Copilot instantiation described in Chapter 4 and the evaluation results reported in Section 5.3 of Chapter 5.

The supervisors are listed as co-authors on both manuscripts in recognition of supervisory guidance, critical feedback on research design, and editorial contributions throughout the preparation of the manuscripts. All research ideas, framework design, implementation, empirical evaluation, analysis, and written content are the original work of the author. The author conducted all experiments, collected and analyzed all data, developed all code, and wrote all drafts of both manuscripts independently.

No portion of this thesis has been submitted for any other degree or qualification at this or any other institution.

# Acknowledgements

To my husband, you were my ground through all of it. Thank you for never once making me feel like this was too much to ask. I could not have done this without you.

To my father, you are the first reason I am standing here. Everything I have reached, I reached because of you.

I am deeply grateful to my supervisor, Dr. Sanaa Alwidian, for her guidance, her patience, and for always pushing me to think deeper. Thank you also to Dr. Khalid for your support throughout this journey.

And to my family, thank you for being my home, no matter how far the work took me.

# Contents

|                                       |             |
|---------------------------------------|-------------|
| <b>Thesis Examination Information</b> | <b>ii</b>   |
| <b>Abstract</b>                       | <b>iii</b>  |
| <b>Declaration of Authorship</b>      | <b>v</b>    |
| <b>Statement of Contributions</b>     | <b>vi</b>   |
| <b>Acknowledgements</b>               | <b>viii</b> |
| <b>List of Figures</b>                | <b>xiv</b>  |
| <b>List of Tables</b>                 | <b>xvi</b>  |
| <b>List of Abbreviations</b>          | <b>xvii</b> |
| <b>1 Introduction</b>                 | <b>1</b>    |
| 1.1 Introduction . . . . .            | 1           |
| 1.2 Motivation . . . . .              | 4           |
| 1.3 Problem Statement . . . . .       | 6           |
| 1.4 Research Objectives . . . . .     | 7           |
| 1.5 Research Questions . . . . .      | 8           |

|          |   |           |
|----------|---|-----------|
| 1.6      | Contributions . . . . .   | 9         |
| 1.7      | Thesis Organization . . . . .   | 11        |
| <b>2</b> | <b>Background and Related Work</b>  | <b>13</b> |
| 2.1      | Requirements Inspection: Foundations and Challenges . . . . .                 | 13        |
| 2.2      | Multi-Dimensional Quality Signals in Requirements . . . . .                   | 16        |
| 2.2.1    | Structural Centrality . . . . .   | 16        |
| 2.2.2    | Linguistic Precision (Specificity) . . . . .                                  | 17        |
| 2.2.3    | Volatility Exposure . . . . .   | 19        |
| 2.2.4    | Why the Three Dimensions Have Been Studied Independently                      | 20        |
| 2.2.5    | The Failure of Compensatory Aggregation . . . . .                             | 21        |
| 2.3      | Multi-Criteria Decision Support and Pareto Methods . . . . .                  | 22        |
| 2.3.1    | Pareto Dominance and Pareto Optimality . . . . .                              | 22        |
| 2.3.2    | Pareto Methods in Search-Based Software Engineering . . . . .                 | 23        |
| 2.3.3    | The Gap: Pareto Methods and Inspection Risk . . . . .                         | 24        |
| 2.3.4    | LLM-Based Requirements Prioritization . . . . .                               | 25        |
| 2.3.5    | The Collection-Blind Limitation . . . . .                                     | 26        |
| 2.4      | Summary and Positioning . . . . .   | 27        |
| 2.4.1    | What the Literature Has Established . . . . .                                 | 27        |
| 2.4.2    | The Unified Gap . . . . .   | 29        |
| 2.4.3    | Positioning the Thesis Contributions . . . . .                                | 29        |
| <b>3</b> | <b>PRISM: A Multi-Dimensional Requirements Inspection Readiness Framework</b> | <b>31</b> |
| 3.1      | Overview . . . . .  | 31        |

|          |   |           |
|----------|---|-----------|
| 3.2      | The PRISM Framework . . . . .                               | 33        |
| 3.2.1    | Stage 1: Data Cleaning and Validation . . . . .             | 34        |
| 3.2.2    | Stage 2: Semantic Representation . . . . .                  | 35        |
| 3.2.3    | Stage 3: Canonical Redundancy Resolution . . . . .          | 37        |
| 3.2.4    | Stage 4: Readiness Metric Computation . . . . .             | 40        |
| 3.2.5    | Stage 5: Multi-Objective Inspection Structuring . . . . .   | 49        |
| 3.3      | Experimental Design . . . . .                               | 51        |
| 3.3.1    | Dataset and Domain Restriction . . . . .                    | 51        |
| 3.3.2    | Evaluation Protocol . . . . .                               | 52        |
| 3.3.3    | Single-Metric Baselines . . . . .                           | 55        |
| <b>4</b> | <b>PRISM-Copilot: Signal-Conditioned Inspection</b>         | <b>57</b> |
| 4.1      | Overview . . . . .  | 57        |
| 4.2      | The SCIP Pattern . . . . .                                  | 58        |
| 4.2.1    | Motivation . . . . .  | 58        |
| 4.2.2    | The Three Components . . . . .                              | 60        |
| 4.2.3    | Policy Mapping Rules . . . . .                              | 61        |
| 4.2.4    | Multiple Policy Composition . . . . .                       | 64        |
| 4.3      | PRISM-Copilot Architecture . . . . .                        | 64        |
| 4.3.1    | Collection Ingestion and Signal Attachment . . . . .        | 66        |
| 4.3.2    | Inspection Policy Mapping and Prompt Construction . . . . . | 66        |
| 4.3.3    | LLM Reasoning and Observation Parsing . . . . .             | 68        |
| 4.4      | Experimental Design . . . . .                               | 69        |
| 4.4.1    | Dataset and Sampling . . . . .                              | 69        |
| 4.4.2    | Prompting Conditions . . . . .                              | 70        |

|          |  |           |
|----------|--|-----------|
| 4.4.3    | Reviewer Reference Standard . . . . .                    | 71        |
| 4.4.4    | Evaluation Measures . . . . .                            | 72        |
| <b>5</b> | <b>Evaluation and Results</b>                            | <b>73</b> |
| 5.1      | Overview . . . . .                                       | 73        |
| 5.2      | PRISM Evaluation . . . . .                               | 73        |
| 5.2.1    | Multi-Objective Structuring Outcome . . . . .            | 73        |
| 5.2.2    | Construct Validity . . . . .                             | 75        |
| 5.2.3    | Signal Independence . . . . .                            | 76        |
| 5.2.4    | Structural Stability . . . . .                           | 77        |
| 5.2.5    | Human Behavioral Alignment and Baseline Comparison . . . | 78        |
| 5.2.6    | Discussion . . . . .                                     | 80        |
| 5.3      | PRISM-Copilot Evaluation . . . . .                       | 84        |
| 5.3.1    | Inter-Rater Agreement . . . . .                          | 84        |
| 5.3.2    | RQ1: Recall-Based Alignment . . . . .                    | 84        |
| 5.3.3    | RQ2: Observation Profile Distribution . . . . .          | 85        |
| 5.3.4    | Discussion . . . . .                                     | 88        |
| 5.4      | Cross-System Discussion . . . . .                        | 91        |
| <b>6</b> | <b>General Conclusions</b>                               | <b>93</b> |
| 6.1      | Overview . . . . .                                       | 93        |
| 6.2      | Answers to the Research Questions . . . . .              | 95        |
| 6.2.1    | RQ1: Collection-Level Metric Validity . . . . .          | 95        |
| 6.2.2    | RQ2: Metric-Conditioned Inspection Reasoning . . . . .   | 96        |
| 6.2.3    | The Unified Thesis Contribution . . . . .                | 97        |

|     |                              |            |
|-----|------------------------------|------------|
| 6.3 | Contributions . . . . .      | 98         |
| 6.4 | Limitations . . . . .        | 100        |
| 6.5 | Future Work . . . . .        | 102        |
|     | <b>Bibliography</b>          | <b>104</b> |
|     | <b>Bibliography</b>          | <b>105</b> |
|     | <b>A Replication Package</b> | <b>111</b> |
|     | <b>B Anchor Statements</b>   | <b>112</b> |

# List of Figures

|     |  |    |
|-----|--|----|
| 3.1 | The SCIP inspection architecture. Requirements first undergo metric computation to derive the stored profile $(S(i), V(i), C(i))$ . The profile is then used for Pareto-structured inspection ordering and deterministic policy selection, which conditions the prompt submitted to GPT-4o for structured requirements inspection. . . . . | 32 |
| 3.2 | The PRISM pipeline: five deterministic stages from raw requirement collection to Pareto-stratified inspection ordering. . . . .  | 34 |
| 4.1 | The SCIP policy mapping layer. Each requirement enters with its pre-computed signal vector $(C(i), S(i), V(i))$ and is routed to one or more inspection policies by deterministic tertile rules. . . . .   | 65 |
| 5.1 | Pareto front distribution across the 262-requirement evaluation collection. Front 1 contains the 27 requirements for which no other requirement is simultaneously more structurally critical, more linguistically imprecise, and more volatility-exposed. . . . .  | 74 |

|     |   |    |
|-----|---|----|
| 5.2 | Human inspection-worthiness endorsement rate for Front 1 versus Front 11. Three independent raters assessed 35 requirements blind to all PRISM metric values and front assignments. Front 1 received majority endorsement from 73% of requirements while Front 11 received 0% (Fisher’s exact $p < 0.05$ ). . . . .   | 80 |
| 5.3 | Proportion of requirements flagged per concern category by prompting condition ( $n = 50$ ). The key result is the Change Sensitivity column: metric-conditioned prompting (Condition B) reduces the overall flag rate from 68% to 60% while concentrating flags on high-volatility requirements, a pattern absent under generic heuristic prompting (Condition C). . . . . | 86 |

# List of Tables

|     |   |    |
|-----|---|----|
| 3.1 | Summary of validation analyses, threats addressed, methods, and metrics. . . . .  | 53 |
| 4.1 | Metric-to-policy mapping in PRISM-Copilot. Activation thresholds are collection tertiles computed over the 262-requirement experiment collection. . . . .   | 62 |
| 4.2 | Per-front sampling allocation across three priority strata. . . . .   | 70 |
| 5.1 | Pareto front distribution for the 262-requirement evaluation collection. Front 1 contains the highest inspection priority requirements. . . . .   | 74 |
| 5.2 | Baseline comparison results for $k = 15$ . Precision is the proportion of selected requirements receiving majority human endorsement. High- $V$ coverage is the proportion of high-volatility requirements in the audited subset that are selected. . . . . | 79 |
| 5.3 | Recall-based alignment by prompting condition ( $n = 31$ requirements with non-empty reference sets). . . . .   | 84 |
| 5.4 | Proportion of requirements flagged per category by condition ( $n = 50$ ). MMC = Missing Measurable Criteria, DR = Dependency Risk, CS = Change Sensitivity. . . . .  | 85 |

# List of Abbreviations

|       |  |
|-------|--|
| C     | Structural Criticality metric                              |
| LLM   | Large Language Model                                       |
| NLP   | Natural Language Processing                                |
| PRISM | Prioritised Requirements Inspection through Signal Metrics |
| RE    | Requirements Engineering                                   |
| S     | Linguistic Specificity metric                              |
| SCIP  | Signal-Conditioned Inspection Prompting                    |
| V     | Volatility Exposure metric                                 |

# Chapter 1

## Introduction

### 1.1 Introduction

Building reliable software is an endeavor that begins long before the first line of code is written. Every software system is preceded by a requirements specification, a structured collection of written statements that describe what the system must do, how it must behave, and what constraints it must satisfy. Reviewing this document carefully before development begins, examining each statement for ambiguity, missing detail, and risk, is a practice known as *requirements inspection*. Decades of empirical evidence establish requirements inspection as one of the most impactful quality assurance activities available to software development teams [17]. Fagan's seminal inspection studies demonstrated that structured requirements review can eliminate up to 82 percent of all defects before the first line of code is written [17], and subsequent cost analyses have estimated that a defect caught at the requirements stage costs between 10 and 200 times less to remediate than the same defect discovered after deployment [8]. For large projects involving hundreds or thousands of requirements, it is not just valuable; it is an operational necessity.

Not all requirements, however, carry the same inspection risk. ISO/IEC/IEEE 29148:2018 standard specifies that a well-formed requirement must be unambiguous, verifiable, and stable [24]. Of these properties, three are particularly consequential for determining where inspection effort should be concentrated. According to the ISO/IEC/IEEE 29148:2018 standard, requirement must be *precise*: written clearly enough to be tested and verified, free of vague language that developers and testers would interpret differently. For example, A requirement stating that *"the system shall respond quickly"* provides no measurable criterion against which any implementation can be verified, leaving conformance entirely to subjective judgment. In addition, a requirement must be *stable*: unlikely to change before or during development, since requirements carrying provisional commitments, external dependencies, or conditional language predict future rework regardless of how clearly they are currently written. For instance, a requirement conditioned on a third-party API that has not yet been finalized is structurally volatile irrespective of its current linguistic precision. That is, when the dependency changes, the requirement will change with it. Last but not least, a requirement must be *impactful*: some requirements are structurally central to the collection, referenced by many others, such that a defect in one propagates outward and silently corrupts every requirement that depends on it. As an example, a requirement defining a core authentication mechanism upon which dozens of downstream requirements depend will, if mis-specified, corrupt the behavior of every one of those dependents, regardless of how carefully those downstream requirements are subsequently inspected.

These three properties are not merely independent quality criteria; they are orthogonal dimensions of inspection risk, and prior research has confirmed that each

independently predicts the presence of inspection-relevant defects [3-8]. In any real requirements collection, these three properties vary considerably across requirements. Some requirements are imprecise, some are volatile, some are structurally critical, and some requirements exhibit all three risk profiles simultaneously. Yet inspection teams currently have no principled, automatic, or reproducible way to identify which requirements score poorly on which property, nor to combine all three into a coherent inspection ordering that preserves the independent diagnostic value of each dimension. The predominant alternative (which is to collapse multiple quality dimensions into a single weighted aggregate score) is particularly harmful: a requirement that is highly central but precisely worded receives the same composite score as one that is moderately deficient on both dimensions, despite the two requirements demanding entirely different inspection responses. When teams lack structured guidance of this kind, they apply roughly the same level of scrutiny to every requirement regardless of its actual risk profile, and the requirements that matter most receive no more attention than those that matter least.

This thesis proposes and empirically evaluates a two-layer inspection support system built to close this gap. The first layer, PRISM (Pareto-Based Requirements Inspection Structuring and Monitoring) automatically computes three inspection readiness metrics directly from the requirement text, one for each quality property identified above, and combines them, through Pareto dominance-based multi-objective structuring, into a principled multi-dimensional inspection ordering that deliberately avoids compensatory aggregation, preserving the independent character of each metric.

The second layer, PRISM-Copilot, instantiates the Signal-Conditioned Inspection

Prompting (SCIP) architectural pattern. PRISM-Copilot uses those validated metrics to condition Large Language Model (LLM)-assisted inspection reasoning, directing each requirement toward the type of scrutiny most relevant to its specific risk profile, rather than subjecting all requirements to an undifferentiated, general-purpose inspection prompt. The result is an end-to-end inspection support architecture that is context-aware at the collection level, treating the full requirements collection as an integrated analytical unit rather than as a set of isolated artifacts.

## 1.2 Motivation

The empirical case for requirements inspection is well established. Fagan’s foundational studies showed that structured peer review of requirements artifacts could remove a substantial proportion of defects before a single line of code was written [17], and later work confirmed that defects identified at this stage are far cheaper to address than those surfacing after deployment [8]. This thesis does not aim to revisit the established value of requirements inspection. Rather, it aims to address the more pressing and operationally consequential question of how inspection teams should systematically allocate their limited attention across a large requirements collection.

Industrial software projects routinely contain hundreds or thousands of requirements, and inspection teams are small relative to the scale of the work. Teams cannot give every requirement the same depth of scrutiny and must therefore make choices about where to focus. Without systematic guidance, those choices are driven by familiarity, recency, or surface features of the text rather than by any principled assessment of which requirements carry the greatest *quality risk* across the three

properties that matter, namely the precision, stability, and structural impact of requirements.

The consequences of this absence of guidance are concrete and systematic. For instance, Requirements that are referenced by many others are considered to be more structurally critical, in the sense that a defect in one requirement could propagate silently outward to other dependent requirements. Despite this challenge, we can see that such structurally critical requirements receive no more formal attention than peripheral requirements that affect nothing else. Requirements written in vague, untestable language are not systematically distinguished from precisely specified ones. Requirements carrying linguistic markers of impending change are not flagged for extra scrutiny before they destabilize the collection. Research has independently confirmed that each of these three dimensions, structural criticality, linguistic specificity, and volatility exposure, predicts inspection-relevant problems [14, 38, 20, 33, 29, 35], yet no framework has brought all three together to give inspection teams actionable, pre-inspection guidance across all dimensions simultaneously.

Two recent developments make it possible to address this gap in a way that was not previously feasible. *Transformer-based language models* now make it possible to represent the semantic content of natural language requirements as high-dimensional vectors, from which structural relationships across an entire specification can be computed automatically without manual dependency annotation [31]. Separately, studies have shown that large language models can detect ambiguity, flag missing measurable criteria, and surface dependency risks from the requirement text with accuracy comparable to earlier *rule-based approaches* [18, 5, 28]. Taken together, these developments open a path not previously available in the requirements

inspection literature toward an inspection support system that characterizes each requirement’s risk profile at the collection level and uses that characterization to direct LLM inspection reasoning toward the concerns that are most relevant for each requirement individually.

### 1.3 Problem Statement

Requirements inspection is undermined by two compounding problems that existing approaches have not resolved together. Each is distinct, but they are logically connected. The first must be solved before the second becomes tractable.

The first problem is the *absence of a unified multi-dimensional framework for characterizing inspection risk across a requirements collection*. Research has independently established that structural criticality, linguistic specificity, and volatility exposure each predict inspection-relevant defects [14, 38, 20, 33, 29, 35]. Existing tools, however, either address one dimension in isolation or, when combining multiple dimensions, collapse them into a single weighted score that allows one dimension to compensate for another [21]. A requirement that scores high on structural criticality but low on linguistic specificity receives the same aggregate score as one that scores moderately on both, even though the two carry entirely different inspection profiles and demand entirely different inspection responses. This compensatory aggregation discards precisely the information that makes each dimension valuable. Inspection teams are therefore left without a principled basis for knowing which requirements carry the highest combined risk, and without a systematic way to identify in which dimension each requirement’s risk is most elevated.

The second problem concerns the context-blindness of existing LLM-based inspection tools. Every current LLM-based approach processes each requirement as a standalone piece of text, with no knowledge of where that requirement sits within the full collection. Whether a requirement is referenced by fifty others or by none, whether its wording is unusually imprecise relative to its peers, whether its language carries the signatures of impending change relative to the distribution of the full collection, none of this is visible to the model. A systematic review of LLM-based requirements engineering research confirmed that collection-level context is absent from all existing inspection approaches [21]. As a result, even capable LLM inspection tools [18, 5, 28] apply the same prompt structure to every requirement regardless of its risk profile, flagging change sensitivity on stable requirements and dependency risk on peripheral ones with the same force as on requirements, where those concerns are genuinely elevated.

Taken together, these two problems leave inspection teams without a tool that can tell them both where to focus their effort and how to direct automated inspection reasoning once that focus is established. Prioritization remains subjective and non-reproducible. LLM inspection reasoning remains undirected. Neither problem can be fully addressed in isolation.

## 1.4 Research Objectives

The problems identified in Section 1.3 point to four concrete objectives that this thesis aims to achieve.

1. To automatically quantify the precision (i.e., specificity), stability, and structural impact (i.e., criticality) of requirements from the requirement text alone, without manual annotation or stakeholder input.
2. To produce a principled inspection ordering that preserves all three quality dimensions simultaneously, rather than aggregating them into a single compensatory score.
3. To translate validated quality metrics into per-requirement LLM inspection guidance that targets each requirement’s specific risk profile.
4. To evaluate whether metric-conditioned guidance shifts LLM behavior toward practitioner-identified concerns, relative to generic inspection prompting.

Objectives 1 and 2 establish the analytical framework. Objectives 3 and 4 apply it to LLM-assisted inspection. The four objectives collectively define the scope of the two-layer system developed in this thesis.

## 1.5 Research Questions

The two problems identified in Section 1.3 motivate two research questions that structure the thesis.

**RQ1:** Can analysis of a full collection of natural language requirements produce independent, valid, and stable multi-dimensional readiness metrics that align with human inspection, judgment, and support principled inspection prioritization?

**RQ2:** Can readiness metrics derived from a full collection of requirements be used to condition LLM inspection reasoning in a way that concentrates inspection

attention on the concerns most relevant to each requirement’s specific risk profile, rather than applying uniform inspection logic across the full collection?

The two questions are logically ordered. RQ2 is only meaningful if RQ1 is answered affirmatively: a metric-conditioned inspection system is only as trustworthy as the metrics it is conditioned on. RQ1 is addressed in Chapter 3. RQ2 is addressed in Chapter 4. Chapter 6 synthesizes both answers into a unified account of what collection-aware inspection support can achieve at the current state of the art.

## 1.6 Contributions

This thesis makes four contributions to the requirements inspection literature.

1. **A unified multi-dimensional framework for inspection risk characterization.** The Pareto-Based Readiness Inspection Structuring Mechanism (PRISM) is the first framework to operationalize structural criticality, linguistic specificity, and volatility exposure as independent, reproducible metrics derived from transformer embedding geometry and lexical pattern matching on requirement text alone, and to combine them through non-compensatory Pareto dominance that preserves their independence in the inspection ordering. The contribution is not the individual metric concepts, since each of which has prior research behind it. Rather, the contribution is in the joint operationalization of the metrics as independent, collection-level computations and the empirical evidence that the resulting ordering is valid, stable, and aligned with human inspection judgment.

2. **An empirical demonstration that inspection risk is genuinely multi-dimensional.** The three metrics are shown to be empirically orthogonal in the evaluation collection, with near-zero pairwise correlations and variance inflation factors negligibly above 1.0. This finding establishes that structural criticality, linguistic specificity, and volatility exposure are not projections of a shared underlying construct, they are genuinely distinct dimensions of inspection risk. This empirical result is the justification for the non-compensatory combination, combining the three metrics into a scalar score would destroy information that their independence makes it possible to preserve.
3. **The SCIP architectural pattern for metric-conditioned LLM inspection.** The Signal-Conditioned Inspection Prompting (SCIP) architectural pattern introduces a policy translation layer that maps collection-derived readiness metrics to per-requirement LLM inspection priorities before any prompt is issued, converting metric values into inspection-relevant language without exposing raw numeric scores to the model. The pattern is instantiated in PRISM-Copilot, a working six-stage inspection pipeline. Because the policy layer is decoupled from any specific metric computation method, the pattern is reusable: any collection analysis approach that produces normalized per-requirement scores on the three metric dimensions can drive the same policy mapping.
4. **A proof-of-concept demonstration of metric-conditioned inspection targeting.** A proof-of-concept evaluation on 50 requirements provides directional evidence suggesting that metric-conditioned prompting concentrates change-sensitivity inspection flags on requirements where collection metrics

indicate elevated volatility exposure, a pattern absent under generic heuristic prompting despite equivalent aggregate recall. This provides initial proof-of-concept evidence, on this evaluation collection, that collection-level metric conditioning changes the qualitative character of LLM inspection output, not only how much it finds, but where it directs its attention.

## 1.7 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 presents the background and related work. It reviews four research threads relevant to the thesis: multi-dimensional quality metrics in requirements, multi-criteria decision support and Pareto methods, LLM-based requirements inspection, and requirements inspection foundations and challenges. Each thread is reviewed in terms of its contribution to the intellectual foundation of the thesis and its limitations relative to the two problems identified in Section 1.3.

Chapter 3 presents PRISM. It describes the five-stage pipeline in full technical detail, covering collection cleaning and validation, semantic representation, canonical redundancy resolution, readiness metric computation, and Pareto-based decision support. It also describes the experimental design used to evaluate the framework, including the dataset, sampling strategy, validation protocol, and baseline comparisons.

Chapter 4 presents SCIP and its instantiation in PRISM-Copilot. It defines the SCIP architectural pattern, the policy mapping rules, and the PRISM-Copilot six-stage pipeline. It also describes the experimental design used to evaluate the

system, including the prompting conditions, the reviewer reference standard, and the evaluation measures.

Chapter 5 presents the evaluation and results for both systems. It reports the PRISM validation results across construct validity, metric independence, structural stability, and human behavioral alignment, followed by the PRISM-Copilot results across inter-rater agreement, recall-based alignment, and observation profile distribution. It concludes with a cross-system discussion that synthesizes what the two evaluations together establish about the two-layer architecture.

Chapter 6 presents the general conclusions. It answers the two thesis-level research questions, states the four contributions of the thesis, acknowledges the limitations that bound those contributions, and outlines four research directions that follow from the work.

## Chapter 2

# Background and Related Work

This chapter reviews the research threads that form the intellectual foundation of this thesis. Four threads are covered: requirements inspection foundations and challenges, multi-dimensional quality metrics in requirements, multi-criteria decision support and Pareto methods, and LLM-based requirements inspection and prompt engineering. Each thread is reviewed in terms of what it has established and where it falls short. Section 2.4 draws all four threads together into a unified gap statement that explains precisely what this thesis addresses that no prior work has addressed.

## 2.1 Requirements Inspection: Foundations and Challenges

Requirements inspection is the systematic examination of software requirements artifacts by human reviewers with the goal of identifying defects before development begins. The practice was formally introduced by Fagan [17], whose landmark study at IBM demonstrated that structured peer review of software artifacts could remove a substantial proportion of defects at a fraction of the cost of finding those defects

through testing or field operation. Fagan's inspection method prescribed a formal process involving distinct roles, author, moderator, reader, and inspector, and a structured sequence of phases: planning, overview, preparation, inspection meeting, rework, and follow-up. The empirical results were striking: Fagan reported defect removal rates of up to 82% through inspection alone, with cost savings that justified the investment many times over. These findings established requirements inspection as a foundational quality assurance practice in software engineering and motivated decades of subsequent research.

Following Fagan's work, researchers explored a variety of modifications and extensions to the basic inspection process. Checklist-based inspection approaches provided reviewers with structured lists of common defect types to look for during preparation, reducing the cognitive load of open-ended review and improving consistency across reviewers [10]. Perspective-based reading techniques assigned different reviewers to different viewpoints, such as the user perspective, the designer perspective, and the tester perspective, on the premise that different perspectives surface different classes of defects [6]. Scenario-based reading extended this idea by providing concrete usage scenarios that reviewers could trace through the requirements to identify missing or inconsistent behaviors [34]. Each of these approaches improved on the basic checklist model by providing more structured guidance for reviewers, and empirical studies confirmed that structured reading techniques outperform ad hoc review on defect detection rates [9].

The scalability of human inspection became a central concern as software systems grew larger and more complex. Studies of industrial inspection programs found that the effectiveness of inspection degrades as specification size increases:

reviewers experience cognitive overload when working through large documents, defect detection rates decline for requirements reviewed late in a session, and the cost per defect found increases as team size and preparation time grow [3]. These findings motivated interest in tool-supported and automated inspection approaches that could help reviewers manage large specifications more efficiently. Early tool support focused on consistency checking, automatically identifying requirements that used the same entity in conflicting ways, and completeness checking, identifying requirements that referenced undefined terms or missing system behaviors [19]. Natural language processing approaches extended this to linguistic quality analysis, automatically detecting requirements that contained ambiguous language, passive voice constructions, or vague quantifiers that predicted reviewer disagreement [16, 20].

The most recent development in requirements inspection research is the application of large language models. Studies have demonstrated that LLMs can identify ambiguity, flag missing measurable criteria, and surface dependency risks from requirement text with accuracy that approaches or matches that of experienced human reviewers [18, 5, 28]. These results have generated significant interest in LLM-assisted inspection as a scalable alternative to fully manual review, particularly for teams that lack dedicated requirements engineering expertise. However, all existing inspection approaches, whether manual, checklist-based, NLP-assisted, or LLM-based, share a fundamental limitation that has not been addressed in the literature: they treat each requirement as an independent unit of analysis, without systematic consideration of how a requirement’s inspection priority is shaped by its position within the full collection. The allocation of inspection effort remains

unprincipled, and reviewers have no systematic basis for deciding which requirements deserve the most careful attention before inspection begins. This allocation problem is the central challenge that motivates the research presented in this thesis.

## **2.2 Multi-Dimensional Quality Signals in Requirements**

Requirements quality research has independently established three dimensions of inspection risk, structural centrality, linguistic precision, and volatility exposure, each of which has been shown to predict inspection-relevant problems. This section reviews each dimension in turn, examining what has been established, what tools exist, and where the limitations lie. It then explains why the three dimensions have historically been studied in isolation, why existing multi-criteria frameworks fail to combine them appropriately, and why their joint operationalization has only recently become computationally feasible.

### **2.2.1 Structural Centrality**

The structural relationships between requirements have long been recognized as a source of inspection risk. Requirements that are referenced by many others occupy structurally central positions in the dependency network of the requirements collection, and a defect in such a requirement propagates outward to all requirements that depend on it, amplifying the cost of late detection. Dahlstedt and Persson showed that inter-requirements dependencies are pervasive in industrial specifications and that unmanaged dependencies are a leading cause of requirements-related

rework [14]. Wang and Wang demonstrated that structurally central requirements are disproportionately involved in integration defects, confirming that structural position within the requirements network predicts inspection-relevant risk [38]. Carlshamre et al. conducted an industrial survey of requirements interdependencies in software product release planning and found that dependency management was among the most challenging and consequential activities in large-scale requirements engineering [11]. Graph-theoretic centrality measures, including degree centrality, betweenness centrality, and eigenvector centrality, have been applied to requirements dependency graphs to identify structurally important requirements. However, these approaches require explicit dependency graphs that must be manually constructed or extracted through keyword matching, limiting their applicability to large natural language specifications where dependencies are implicit rather than explicitly declared. Embedding-based approaches that derive structural relationships automatically from semantic similarity have been proposed as a more scalable alternative [31], but their systematic application to requirements inspection prioritization had not been explored prior to this thesis.

### **2.2.2 Linguistic Precision (Specificity)**

The linguistic quality of requirements has been studied extensively as a predictor of misunderstanding, ambiguity, and rework. Berry, Kamsties, and Krieger provided a comprehensive taxonomy of ambiguity in natural language requirements, distinguishing lexical ambiguity, syntactic ambiguity, semantic ambiguity, and pragmatic ambiguity as distinct quality concerns [7]. Femmer et al. introduced the concept of requirements smells, linguistic patterns in requirement text that signal potential

quality problems, and showed that smell density is associated with increased review effort and defect rates [20]. Shah et al. demonstrated that linguistically imprecise requirements, characterized by vague quantifiers, subjective adjectives, and missing measurable criteria, are significantly more likely to generate disagreement between developers and testers during implementation [33]. Chantree et al. showed that nocuous ambiguities, ambiguities that cause actual disagreement among readers rather than merely theoretical interpretive possibilities, can be automatically identified in requirement text using heuristics derived from word distribution information, demonstrating the feasibility of NLP-based nocuous ambiguity detection [12]. Tool support for linguistic quality assessment advanced from simple keyword matching to sophisticated NLP pipelines: the QuARS tool [16] applies morphological and syntactic analysis to detect linguistic patterns associated with poor quality, while Femmer et al.'s smell detector operates at the token level to identify vague terms, subjective language, and missing constraints. More recent approaches apply machine learning to classify requirements by their ambiguity and imprecision risk. Despite this progress, all existing linguistic quality tools share a fundamental limitation: they assess each requirement against absolute thresholds rather than against the distribution of the full collection. A requirement that appears vague in absolute terms may be entirely normal for its domain, while one that appears adequately specified in isolation may be unusually imprecise relative to its peers. This collection-relative dimension of linguistic quality has not been addressed by prior work.

### 2.2.3 Volatility Exposure

Requirements volatility, the propensity of requirements to change during development, has been consistently identified as a major source of schedule overrun and rework in software projects. Nurmuliani et al. studied requirements changes across multiple industrial projects and found that volatile requirements caused disproportionate rework costs, with a small proportion of requirements accounting for the majority of change-related effort [29]. Thakurta and Roy showed that requirements volatility is predictable from properties of the requirements themselves, specifically from linguistic markers including hedging language, conditional clauses, and temporal qualifiers, suggesting that volatility risk can be estimated before changes actually occur [35]. Zowghi and Nurmuliani conducted an empirical survey of requirements volatility across software projects and found that requirements volatility has a significant impact on both schedule overrun and cost overrun, confirming that volatile requirements are a major source of project risk regardless of when they are identified [42]. Cleland-Huang et al. examined non-functional requirements volatility and found that requirements expressing system qualities such as performance, reliability, and security were disproportionately likely to change as stakeholder understanding evolved during development [13]. Existing volatility prediction approaches typically rely either on historical change data, which requires project history and is unavailable at the start of a new project, or on individual requirement text analysis that assesses each requirement independently without reference to the volatility distribution of the full collection.

### 2.2.4 Why the Three Dimensions Have Been Studied Independently

The three dimensions reviewed above have a shared characteristic: each was studied in isolation from the others. This is not an accidental oversight. It reflects a genuine computational barrier that prevented joint operationalization until recently. Structural centrality requires graph analysis or network-based computation over the full requirements collection. Linguistic precision requires NLP pipelines operating on individual requirement text. Volatility exposure requires temporal signal detection or linguistic pattern matching calibrated against change history. Before the advent of large pre-trained transformer models, these three computational approaches operated on fundamentally different representations, making unified computation difficult, expensive, and domain-dependent. A team wishing to combine all three signals would have needed three separate toolchains, manual integration steps, and domain-specific calibration for each dimension independently.

Transformer-based embedding models changed this. Models such as E5-large [31] represent each requirement as a dense vector in a shared semantic space from which structural relationships can be derived through cosine similarity, linguistic signal projections can be computed through anchor-based methods, and volatility patterns can be detected through lexical marker analysis calibrated against the embedding distribution. The computational unification that embeddings make possible is what makes a joint framework computationally feasible for the first time. To our knowledge, the empirical question of whether the three dimensions are genuinely independent in real-world requirements collections, and therefore whether they should be combined non-compensatorily, had not been studied prior to this thesis.

### 2.2.5 The Failure of Compensatory Aggregation

Prior to this thesis, the dominant approach to combining multiple quality dimensions in requirements prioritization was scalar aggregation, in which each dimension receives a weight and a final priority score is computed by summing weighted values. The Analytic Hierarchy Process (AHP), introduced by Saaty and applied to requirements prioritization by Karlsson and Ryan [26], is the most prominent example. In their cost-value approach, AHP pairwise comparisons are used to derive relative value and implementation cost scores for each requirement, which are then aggregated into a single priority ordering. It is explicitly a compensatory method: a requirement that scores poorly on structural centrality can be compensated by a high score on linguistic precision, and the two requirements with opposite profiles may receive the same aggregate priority. Fuzzy AHP variants [1] and weighted scoring approaches address some of AHP's limitations around imprecision and subjectivity in weight elicitation, but retain the same compensatory aggregation structure. The systematic review of requirements prioritization techniques by Achimugu et al. confirmed that compensatory scalar aggregation dominates the requirements prioritization literature, with AHP and its variants accounting for the majority of multi-criteria approaches [1].

The problem with compensatory aggregation for inspection prioritization runs deeper than a technical design choice. It destroys information that matters. A requirement that is structurally central but precisely worded carries a completely different inspection profile from one that is imprecise but stable. Both may score identically under weighted aggregation, yet they demand different inspection responses: the first needs dependency-focused scrutiny, the second needs clarity

and measurability-focused scrutiny. Collapsing the two into a single score makes it impossible to know which type of inspection attention is most needed. If the three dimensions are genuinely orthogonal, as the empirical evidence reviewed above suggests they may be, then preserving each dimension independently is not just preferable. It is the only approach that does not silently discard information the inspection team needs.

## **2.3 Multi-Criteria Decision Support and Pareto Methods**

Multi-criteria decision making (MCDM) is concerned with structuring and solving decision problems that involve multiple, often conflicting criteria. In software engineering, MCDM methods have been widely applied to requirements prioritization, release planning, test case selection, and resource allocation. The fundamental challenge in any MCDM problem is how to combine multiple criteria into a decision that respects the information each criterion carries. As discussed in Section 2.2.5, the dominant approach in requirements engineering has been compensatory aggregation through weighted scoring and AHP. This section reviews the alternative: non-compensatory methods based on Pareto dominance, which preserve the independence of criteria rather than collapsing them into a single score.

### **2.3.1 Pareto Dominance and Pareto Optimality**

The concept of Pareto optimality originates in welfare economics, where it was introduced by Vilfredo Pareto to describe allocations in which no individual can be

made better off without making another worse off [30]. In multi-criteria decision making, a solution  $x$  is said to Pareto-dominate another solution  $y$  if  $x$  is at least as good as  $y$  on all criteria and strictly better on at least one. A solution is Pareto-optimal, or non-dominated, if no other solution dominates it. The set of all non-dominated solutions forms the Pareto front, which represents the set of solutions for which improvement on any one criterion necessarily comes at the cost of at least one other.

The key property of Pareto dominance that distinguishes it from compensatory aggregation is that it makes no assumption about the relative importance of criteria and requires no weights. Two solutions are compared dimension by dimension, and dominance is established only when one solution is unambiguously superior across the full criteria vector. This means that a solution which is strong on one criterion but weak on another is not dominated by a solution that scores moderately on both. The two solutions are incomparable under Pareto dominance, which is the correct outcome when the criteria are genuinely independent and neither should compensate for the other.

### **2.3.2 Pareto Methods in Search-Based Software Engineering**

The application of Pareto-based multi-objective optimization to software engineering problems was systematically reviewed by Harman et al., who showed that a wide range of software engineering problems can be formulated as multi-objective search problems amenable to Pareto-based solution methods [23]. The non-dominated sorting genetic algorithm (NSGA-II), introduced by Deb et al., has been particularly

influential as an efficient algorithm for approximating the Pareto front in high-dimensional objective spaces [15]. NSGA-II sorts solutions into Pareto fronts by iteratively identifying and removing the non-dominated set, producing a ranked partition of the solution space that preserves the full multi-dimensional structure of the objective vector.

In requirements engineering, Pareto-based methods have been applied primarily to the next-release problem, which involves selecting a subset of requirements for implementation in the next software release subject to resource constraints. Bagnall et al. introduced the next-release problem as a formal optimization problem and showed that it is NP-hard in the general case [4]. Subsequent work applied multi-objective evolutionary algorithms, including NSGA-II, to find Pareto-optimal release plans that balance stakeholder value, implementation cost, and technical risk [41]. Greer and Ruhe applied multi-objective optimization to release planning with interdependent requirements, showing that Pareto methods naturally handle requirement dependencies without requiring explicit weight elicitation from stakeholders [22]. These studies established that Pareto-based methods are well-suited to requirements engineering problems where multiple independent objectives must be balanced without a fixed priority ordering.

### **2.3.3 The Gap: Pareto Methods and Inspection Risk**

The application of Pareto methods to software engineering is well-established, and their advantages over compensatory aggregation for problems involving independent objectives are theoretically and empirically documented. What the literature has not provided, however, is the application of Pareto dominance to requirements

inspection risk characterization over independently validated quality signals. Prior Pareto- based requirements engineering work addresses delivery decisions, which requirements to implement and when, rather than inspection decisions, which requirements to review most carefully and in which dimension. The criteria used in next-release planning, stakeholder value, implementation cost, and technical risk, are fundamentally different from inspection risk signals, which characterize each requirement’s quality exposure relative to the full collection rather than its contribution to a release plan. To our knowledge, no prior work has applied Pareto dominance to non-compensatorily combine independently validated quality signals for the purpose of inspection prioritization. This is the gap that PRISM addresses.

### 2.3.4 LLM-Based Requirements Prioritization

A distinct thread of LLM-based requirements engineering research applies LLMs to delivery prioritization and release planning rather than inspection. Sami et al. demonstrated that LLMs can rank requirements backlogs using stakeholder-oriented criteria such as business value, implementation effort, and strategic alignment [32]. Jahi and Sami extended this to sprint planning, showing that LLM-assisted prioritization produces release plans that practitioners rate as comparable to manually produced plans on alignment with business objectives [25]. These studies share SCIP’s interest in using structured information to guide LLM reasoning about requirements, but they address a fundamentally different problem. Delivery prioritization asks which requirements to implement first, based on business value and resource constraints. Inspection prioritization asks which requirements to review most carefully and what type of defect to look for in each one, based on quality risk

metrics derived from the full requirements collection. The information sources, the decision criteria, and the practical consequences of a wrong decision are all different. A requirement that is low priority for delivery may carry high inspection risk if it is structurally critical or linguistically imprecise. Conflating the two problems would direct inspection effort toward requirements that are important to deliver rather than those that are most likely to carry undetected defects.

### 2.3.5 The Collection-Blind Limitation

All existing LLM-based inspection approaches share a structural limitation that has not been addressed in the literature: they analyze each requirement as a standalone piece of text, without any knowledge of where that requirement sits within the full collection. The model cannot know whether the requirement is referenced by many others or by none, whether its wording is unusually imprecise relative to its peers, or whether its language carries the indicators of impending change relative to the distribution of the full collection. Ferrari et al.'s systematic review confirmed that collection-level context is absent from all existing LLM-based inspection approaches and that no framework uses the distribution of the full requirements collection to condition or prioritize individual requirement analysis [21]. This collection-blind limitation means that even the most capable LLM inspection tool applies the same prompt structure to every requirement regardless of its risk profile, flagging change sensitivity on stable requirements and dependency risk on peripheral ones with the same force as on requirements where those concerns are genuinely elevated. To our knowledge, no existing approach uses validated collection-level quality metrics to condition per-requirement LLM inspection reasoning. This is the second gap

that this thesis addresses, through the SCIP architectural pattern introduced in Chapter 4.

## **2.4 Summary and Positioning**

The four threads reviewed in this chapter converge on a consistent picture of the state of the art in requirements inspection support. Each thread has produced valuable results and well-validated tools. Each thread has also reached a boundary that prior work has not crossed. This section draws the four threads together, identifies the unified gap that their boundaries define, and positions the contributions of this thesis within that gap.

### **2.4.1 What the Literature Has Established**

Requirements inspection research, reviewed in Section 2.1, established that structured inspection is one of the highest-return quality assurance investments in software engineering, and that scalability under time and resource constraints is the central practical challenge. The evolution from Fagan’s formal inspection method through checklist-based, perspective-based, and NLP-assisted approaches produced progressively more efficient inspection support, but none of these approaches provides a principled basis for allocating inspection effort across a large requirements collection before inspection begins.

Multi-dimensional quality metric research established that structural criticality, linguistic specificity, and volatility exposure each independently predict inspection-relevant problems. Each dimension has a well-validated research base and operational

tools. What the research has not established is whether the three dimensions are genuinely independent of each other in real-world requirements collections, and therefore whether they should be combined non-compensatorily. The computational barrier that prevented joint operationalization, the need for three separate toolchains operating on fundamentally different representations, was removed by transformer-based embedding models, but the joint framework that embeddings make possible had not been built or validated prior to this thesis.

Multi-criteria decision support research established that Pareto dominance is the theoretically and empirically correct approach for combining independent objectives without compensatory aggregation, and that Pareto methods are well-established in software engineering for delivery decisions. What the research has not established is the application of Pareto dominance to inspection risk characterization over independently validated quality metrics. The existing Pareto-based requirements engineering work addresses which requirements to implement, not which requirements to inspect most carefully and in which dimension.

LLM-based inspection research established that LLMs can assess multiple quality dimensions simultaneously from individual requirement text with accuracy that approaches human reviewer performance, and that prompt content shapes LLM inspection reasoning in measurable ways. What the research has not established is any approach that uses collection-level quality metrics to condition per-requirement LLM inspection reasoning. Every existing LLM-based inspection tool is collection-blind, applying uniform prompt structures regardless of each requirement's risk profile.

### 2.4.2 The Unified Gap

The four threads share two boundaries that define a unified gap. The first boundary is the absence of a non-compensatory multi-dimensional framework that characterizes each requirement’s inspection risk across structural criticality, linguistic specificity, and volatility exposure simultaneously, using independently validated metrics derived automatically from the full requirements collection without manual annotation. The second boundary is the absence of any mechanism that uses collection-level quality metrics to condition LLM inspection reasoning on a per-requirement basis, directing each requirement toward the type of inspection scrutiny most relevant to its specific risk profile.

These two boundaries are not independent. The second boundary cannot be crossed without first crossing the first. A metric-conditioned LLM inspection system is only meaningful if the metrics it is conditioned on are valid, independent, and stable. The validation of the metrics is the prerequisite for the conditioning mechanism. This logical dependency defines the two-layer architecture that this thesis proposes: the first layer must be built and validated before the second layer can be meaningfully designed and evaluated.

### 2.4.3 Positioning the Thesis Contributions

This thesis addresses both boundaries through a two-layer inspection support system. The first layer, PRISM, addresses the first boundary by operationalizing structural criticality, linguistic specificity, and volatility exposure as three independent metrics derived from transformer embedding geometry and lexical pattern matching, and

combining them through non-compensatory Pareto dominance to produce a multi-dimensional inspection ordering. Chapter 3 presents the full PRISM framework and reports the validation evidence for metric independence, construct validity, structural stability, and alignment with human inspection judgment. The second layer, SCIP, addresses the second boundary by introducing a policy translation layer that maps each requirement’s validated metric profile to targeted LLM inspection priorities before any prompt is issued. Chapter 4 presents the SCIP architectural pattern, its instantiation in PRISM-Copilot, and the proof-of-concept evaluation that provides directional evidence of the targeting effect. Chapter 5 reports the consolidated evaluation results for both layers. The two layers jointly establish a principled, end-to-end approach to collection-aware inspection support that addresses the allocation problem at both the prioritization and the reasoning level.

## Chapter 3

# PRISM: A Multi-Dimensional Requirements Inspection Readiness Framework

### 3.1 Overview

This chapter presents PRISM, the Pareto-Based Readiness Inspection Structuring Mechanism, the first layer of the two-layer inspection support system introduced in Chapter 1. The overall SCIP inspection architecture is illustrated in Figure 3.1. PRISM addresses the first research question identified in Section 1.5: whether analysis of a full collection of natural language requirements can produce independent, valid, and stable multi-dimensional readiness metrics that align with human inspection judgment and support principled inspection prioritization.

PRISM operates on a collection of natural language requirements and produces a Pareto-structured inspection ordering that characterizes each requirement's multi-dimensional inspection risk across three independent dimensions. The evaluation

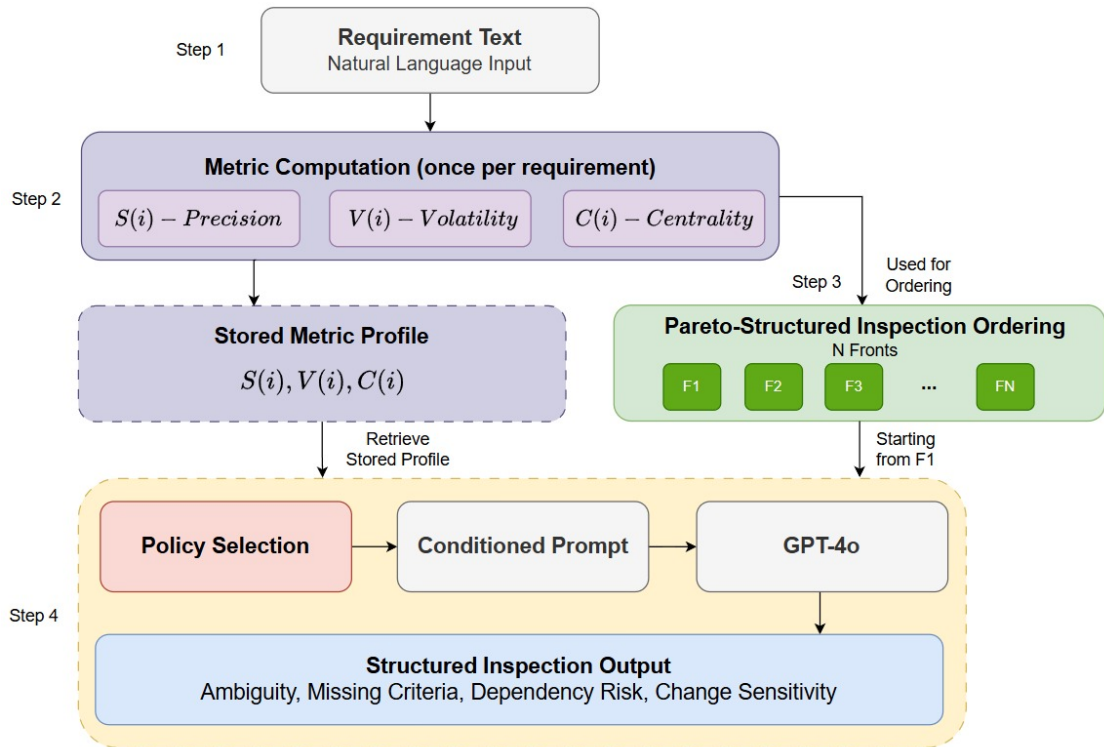


FIGURE 3.1: The SCIP inspection architecture. Requirements first undergo metric computation to derive the stored profile ( $S(i), V(i), C(i)$ ). The profile is then used for Pareto-structured inspection ordering and deterministic policy selection, which conditions the prompt submitted to GPT-4o for structured requirements inspection.

reported in this chapter is based on a subset of 303 requirements drawn from the tera-PROMISE repository [27], restricted to an e-commerce domain to ensure semantic coherence in the embedding-based structural analysis. After semantic redundancy resolution, the final evaluation collection consists of 262 canonical requirements. All signal parameters, anchor sets, and structural choices were fixed prior to validation to prevent post-hoc tuning.

This chapter is organized as follows. Section 3.2 presents the full five-stage PRISM pipeline in technical detail, covering collection cleaning and validation, semantic representation, canonical redundancy resolution, readiness metric computation, and multi-objective inspection structuring. Section 3.3 describes the experimental design used to evaluate the framework, including the dataset, validation protocol, and single-metric baselines. The evaluation results are reported in Chapter 5.

## 3.2 The PRISM Framework

PRISM is implemented as a five-stage deterministic pipeline that transforms a raw collection of natural language requirements into a Pareto-structured inspection ordering. The stages execute in sequence. Stage 1 cleans and validates the collection by normalizing and validating the raw input. Stage 2 maps each requirement into a shared semantic vector space using a pre-trained transformer model. Stage 3 resolves redundancies by identifying and consolidating semantically equivalent requirements. Stage 4 computes the three readiness metrics, Structural Criticality, Linguistic Specificity, and Volatility Exposure, for each requirement in the resolved collection. Stage 5 combines those metrics through non-compensatory Pareto dominance to produce the final inspection ordering. All operations are deterministic under fixed

inputs, meaning that the pipeline produces identical output on every execution given the same collection, ensuring full reproducibility. Figure 3.2 illustrates the end-to-end architecture of the pipeline. The following subsections describe each stage in detail.

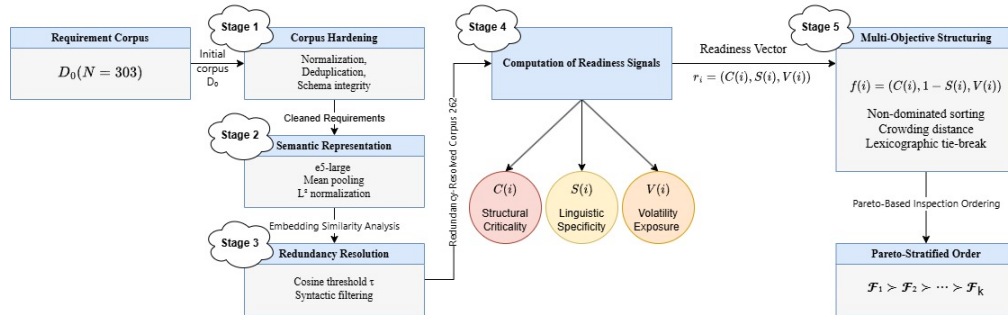


FIGURE 3.2: The PRISM pipeline: five deterministic stages from raw requirement collection to Pareto-stratified inspection ordering.

### 3.2.1 Stage 1: Data Cleaning and Validation

The first stage prepares the raw requirement texts for downstream processing. Four operations are applied in sequence: column types are enforced as strings to preserve requirement identifiers, whitespace is standardized to ensure consistent input to the embedding model in Stage 2, integrity checks verify that every requirement has a unique identifier and a class label, and the collection is sorted deterministically by identifier to guarantee reproducible ordering across executions. The result is a clean, validated collection of 303 requirements that serves as the stable input to all subsequent stages. Implementation details, including the specific normalization rules and sorting procedure, are documented in the replication package [ReplicationPackage].

### 3.2.2 Stage 2: Semantic Representation

Requirements are written in natural language, which means that the structural and linguistic relationships between them are not directly measurable from their text alone. Two requirements can express closely related concepts using entirely different words, and two requirements can share many words while expressing unrelated ideas. To reason about requirements mathematically, to measure how central one requirement is to the collection, how precisely it is worded relative to its neighbors, or how semantically similar it is to prototypical examples of stable or volatile language, PRISM needs a representation of each requirement that captures meaning rather than just surface form. Stage 2 produces this representation by mapping each requirement into a shared semantic vector space, where every requirement becomes a point and the distance between points reflects the semantic similarity between requirements.

PRISM uses the `intfloat/e5-large` pre-trained transformer model to compute these representations [39]. This model was selected for three reasons. First, it was specifically designed and optimized for embedding-based semantic similarity tasks, which is precisely the kind of computation that Stages 3, 4, and 5 rely on. Second, it does not require domain-specific fine-tuning or a query prefix for passage-level embedding, which means it can be applied directly to the requirements collection without introducing any asymmetry or collection-dependent training decisions that would undermine reproducibility. Third, it achieves strong performance on standard semantic similarity benchmarks relative to models of comparable size [39], making it a well-justified choice for a framework that depends on embedding geometry for all of its downstream signal computations. For each requirement  $i$  with cleaned text

$t_i^{\text{clean}}$ , the model applies a semantic embedding function  $\Phi$  that maps the text into a  $d$ -dimensional real-valued vector:

$$\Phi : t_i^{\text{clean}} \mapsto \mathbf{x}_i \in \mathbb{R}^d \quad (3.1)$$

where  $d = 1024$  for the `intfloat/e5-large` model.

The raw output of the transformer model consists of one hidden-state vector per token in the input sequence. To obtain a single vector that represents the entire requirement, PRISM applies mask-aware mean pooling over the final hidden layer. This means averaging the token-level representations across all real tokens in the sequence while excluding the padding tokens that are added to make sequences of different lengths compatible for batch processing. Mean pooling was chosen over the alternative of using the model’s CLS token representation because research has consistently shown that mean pooling produces more stable and semantically rich sentence-level representations for similarity tasks [31]. Let  $\mathbf{z}_i$  denote the pooled representation of requirement  $i$ . The final embedding is obtained by applying L2 normalization, a mathematical scaling operation applied to the numeric vector produced by the transformer model, entirely separate from the text standardization performed in Stage 1:

$$\mathbf{x}_i = \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|_2} \quad (3.2)$$

L2 normalization scales each embedding vector so that its length is exactly 1, which has two practical consequences for the downstream computations. First, it ensures that all requirements are compared on the same scale regardless of differences

in text length or complexity: a short requirement and a long requirement are treated equally because the scaling removes any influence that text length has on the vector magnitude. Second, once all vectors have the same length, measuring the semantic similarity between any two requirements  $i$  and  $j$  reduces to a simple dot product:

$$\text{sim}(i, j) = \mathbf{x}_i^\top \mathbf{x}_j \quad (3.3)$$

which is the similarity measure used throughout Stages 3 and 4.

All inference is performed on CPU with a fixed random seed of 42. GPU-based floating-point arithmetic can produce slightly different numerical results across different hardware configurations due to the non-deterministic nature of parallel processing, which would cause the pipeline to produce different embedding values on different machines. Forcing CPU execution eliminates this source of variation and ensures that the embedding matrix produced by Stage 2 is identical across all compliant execution environments, consistent with the reproducibility guarantee stated for the full pipeline. The output of Stage 2 is a matrix  $\mathbf{X} \in \mathbb{R}^{303 \times 1024}$ , where each row  $\mathbf{x}_i$  is the unit-scaled embedding of requirement  $i$ , and this matrix is the shared geometric foundation from which all three readiness metrics in Stage 4 are derived.

### 3.2.3 Stage 3: Canonical Redundancy Resolution

Before the embedding matrix produced in Stage 2 can be used for signal computation, the collection must be checked for semantic redundancy. Real-world requirement collections frequently contain duplicate or near-duplicate entries. A requirement may have been written independently by two different stakeholders, copied from an earlier

document with minor wording changes, or restated in slightly different language across different sections. If these duplicates remain when Stage 4 computes the readiness metrics, they distort the results: a duplicated requirement accumulates structural influence from both copies, receives double weight in the signal distributions, and occupies two positions in the Pareto ordering when it should occupy one. Stage 3 removes this distortion by identifying and consolidating exact semantic equivalences before any metric computation begins.

The central parameter is a similarity threshold  $\tau$ , which determines how similar two requirements must be in the embedding space before they are considered candidates for consolidation. Rather than selecting  $\tau$  by arbitrary preference, PRISM performs a bounded sweep over  $\tau \in \{0.80, 0.82, 0.84, 0.86, 0.88, 0.90, 0.92\}$  and counts the number of equivalence groups detected at each value. The selected threshold is the smallest value at which the group count stabilizes, meaning that tightening the threshold further does not materially change the groupings. For the evaluation collection, this stabilization occurs at  $\tau = 0.86$ , which yielded 32 equivalence groups. Selecting the smallest stabilizing value is principled because it collapses only those requirements whose similarity is high enough to be structurally stable across nearby threshold values, avoiding over-consolidation of requirements that are merely related rather than equivalent. To prevent floating-point boundary effects, the effective threshold applied during pair comparison is  $\tau_{\text{strict}} = \tau - \varepsilon$  where  $\varepsilon = 0.01$ .

Exceeding the similarity threshold is necessary but not sufficient for consolidation. Three additional gates prevent the collapse of requirements that are semantically similar but operationally distinct.

The first gate restricts comparisons to requirements within the same class label. For example, a functional requirement about password reset and a non-functional requirement about password reset security are semantically close but serve fundamentally different roles. The class gate ensures that only requirements of the same type are ever compared.

The second gate checks syntactic signature compatibility. Two requirements must share the same main action verb, at least one overlapping target object, and the same conditional structure. For example, “the system shall authenticate users if the session has expired” and “the system shall authenticate users if the account is new” are semantically similar but describe different operational conditions. The signature gate detects the difference in their conditional triggers and retains both.

The third gate checks numeric token consistency. For example, “the system shall respond within 200 milliseconds” and “the system shall respond within 500 milliseconds” would score very high on semantic similarity, yet they specify different performance thresholds. The numeric gate classifies pairs with differing numeric tokens as parametric variants and retains them separately.

When all three gates confirm a group of exact duplicates, one representative is selected deterministically. The selection rule prefers the requirement that contains numeric values, and among requirements with the same numeric status, prefers the longer text. Applying this procedure to the 303-requirement collection identifies 32 equivalence groups encompassing 73 requirements, from which 41 redundant members are consolidated, yielding a canonical collection of 262 requirements. This canonical collection, together with the corresponding subset of the Stage 2 embedding matrix, serves as the sole input to Stage 4.

### 3.2.4 Stage 4: Readiness Metric Computation

Stage 4 computes the three readiness metrics that form the core contribution of the PRISM framework. Each metric captures a different dimension of inspection risk and is computed independently from the canonical collection and its embedding matrix. All three metrics share the same computational architecture. Each combines a lexical component, derived from pattern matching on requirement text, with a semantic component, derived from operations on the embedding vectors produced in Stage 2. Both components are converted to percentile ranks over the collection so that they are on the same scale. The two ranked components are then combined using the geometric mean, which enforces conjunctive coupling: a requirement scores high on a metric only if both components are elevated simultaneously. A high value on one component cannot compensate for a low value on the other. The combined score is then rescaled to  $[0, 1]$  using min-max normalisation. This shared architecture means that all three metrics are directly comparable and can be combined in Stage 5 without any cross-metric weighting.

#### **Structural Criticality**

Structural Criticality measures the extent to which a requirement contributes to the semantic structure of the collection. The concept draws on an established finding in requirements engineering: Wang and Wang [38] showed that requirements with higher graph centrality scores are significantly more likely to be involved in integration defects, establishing that a requirement's position within a specification network matters for quality outcomes. The challenge is that classical graph centrality

measures require an explicit dependency graph, a network of documented links between requirements. In most real-world collections, such links are not available.

PRISM operationalizes this concept as Structural Criticality, a metric that approximates graph centrality from embedding geometry without requiring explicit dependency links. Requirements that are semantically similar are likely to be related: one may reference or depend on the other, or both may describe properties of the same system component. A requirement that is semantically close to many others, and whose removal would reduce the overall coherence of the collection, is likely to be structurally influential even without explicit dependency annotation. This hypothesis is supported by the construct validity results reported in Chapter 5. Three structural primitives capture different geometric aspects of this influence.

The first primitive, centroid proximity  $C_1(i)$ , measures how closely a requirement's embedding is aligned with the global semantic center of the collection. The collection centroid  $\hat{\boldsymbol{\mu}}$  is computed by averaging all requirement embeddings and unit-scaling the result:

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i, \quad \hat{\boldsymbol{\mu}} = \frac{\bar{\mathbf{x}}}{\|\bar{\mathbf{x}}\|_2} \quad (3.4)$$

Centroid proximity is then the dot product between the requirement's embedding and the collection centroid:

$$C_1(i) = \mathbf{x}_i^\top \hat{\boldsymbol{\mu}} \quad (3.5)$$

A requirement with high  $C_1$  expresses concepts that are representative of the collection as a whole, sitting at the semantic center rather than at the periphery.

The second primitive, local density  $C_2(i)$ , measures the average semantic similarity between a requirement and its  $k$  nearest neighbors in the embedding space. A requirement that is tightly clustered with many similar neighbors sits in a dense region where related concerns are concentrated. If such a requirement is misspecified, the error is likely to affect multiple neighboring requirements simultaneously:

$$C_2(i) = \frac{1}{|\mathcal{N}_k(i)|} \sum_{j \in \mathcal{N}_k(i)} \mathbf{x}_i^\top \mathbf{x}_j \quad (3.6)$$

where  $\mathcal{N}_k(i)$  denotes the set of  $k$  nearest neighbors of requirement  $i$  under cosine similarity. The neighborhood size is fixed at  $k = 5$ .

The third primitive, cohesion drop  $C_3(i)$ , quantifies how much the overall semantic coherence of the collection would decrease if requirement  $i$  were removed. Global coherence is defined as the average pairwise similarity across all requirements:

$$\text{coh}(\mathbf{X}) = \frac{2}{N(N-1)} \sum_{p < q} \mathbf{x}_p^\top \mathbf{x}_q \quad (3.7)$$

The cohesion drop is the non-negative difference between global coherence and the coherence of the collection with requirement  $i$  removed:

$$C_3(i) = \max(0, \text{coh}(\mathbf{X}) - \text{coh}(\mathbf{X}_{-i})) \quad (3.8)$$

A requirement with high  $C_3$  acts as a semantic bridge between different regions of the collection. Removing it causes a measurable reduction in how interconnected the collection is. For the collection size used in this study ( $N = 262$ ), exact computation is feasible.

Each primitive is converted to a percentile rank  $r_m(i) = \text{rank}_{\text{pct}}(C_m(i))$  over the collection, placing all three on a common scale in  $[0, 1]$ . The three ranked primitives are then combined using the geometric mean:

$$C^*(i) = (r_1(i) \cdot r_2(i) \cdot r_3(i))^{1/3} \quad (3.9)$$

The geometric mean ensures that structural criticality requires all three properties simultaneously. A requirement that is globally representative but locally isolated scores high on  $C_1$  but low on  $C_2$ , and the geometric mean suppresses its combined score accordingly. Arithmetic mean coupling would allow a high value on one primitive to compensate for low values on the others, producing misleadingly high criticality scores. The final Structural Criticality metric is obtained by min-max scaling  $C^*(i)$  to  $[0, 1]$ :

$$C(i) = \frac{C^*(i) - \min_j C^*(j)}{\max_j C^*(j) - \min_j C^*(j) + \varepsilon} \quad (3.10)$$

where  $\varepsilon$  is machine epsilon used solely for numerical stability. The full set of  $C(i)$  values for the 262-requirement collection is provided in the replication package [2].

### Linguistic Specificity

Linguistic Specificity measures how precisely a requirement is worded and how verifiable it is from its statement alone. The metric is grounded in the well-established finding that requirements written in vague, untestable language are a reliable predictor of misunderstanding and rework during development [20, 33]. Lexical approaches to detecting this problem have demonstrated that surface-level

cues such as vague adjectives and universal quantifiers carry real diagnostic value. However, a purely lexical approach has a known limitation: it measures whether the right words are present, but not whether the requirement as a whole is conceptually precise. A requirement can avoid all flagged vocabulary and still be written in language too abstract to test. Conversely, a requirement can contain a vague adjective in an otherwise highly specified statement without losing its overall precision. The Linguistic Specificity metric addresses this by combining a lexical component that assesses surface precision with a semantic component that assesses conceptual precision, requiring both to be elevated simultaneously.

The lexical component  $L(i)$  counts two types of cues in the requirement text. Verifiable cues signal concrete, testable content: numeric values with measurement units, references to named standards such as ISO or IEEE, and specific protocol or version identifiers such as AES-256 or TLS 1.2. Vague cues signal imprecise, untestable content: qualitative comparatives such as faster or better, soft absolutes such as as possible, and universal quantifiers such as all or every. The lexical ratio is:

$$L(i) = \frac{v_i}{v_i + g_i + 1} \quad (3.11)$$

where  $v_i$  denotes the number of verifiable cue matches and  $g_i$  the number of vague cue matches in requirement  $i$ . The additive constant in the denominator prevents division by zero. A requirement such as “the system shall respond within 200 milliseconds for 95% of requests” scores high on  $L$  because it contains a numeric value with a unit and a percentage threshold. A requirement such as “the system shall be as fast as possible” scores low because it contains a soft absolute and no verifiable cues.

The semantic component  $M(i)$  captures a dimension of precision that lexical matching cannot reach. Each requirement’s embedding is projected against two anchor sets: six specific prototypes representing highly precise, verifiable requirements, and six vague prototypes representing imprecise, untestable requirements. The full anchor statements are provided in Appendix B. Both anchor sets are embedded using the same model and pooling procedure as Stage 2, and their embeddings are averaged and unit-scaled to produce two centroid vectors:

$$\mathbf{c}_{\text{spec}} = \frac{\frac{1}{|A_{\text{spec}}|} \sum_k \mathbf{a}_k^{\text{spec}}}{\left\| \frac{1}{|A_{\text{spec}}|} \sum_k \mathbf{a}_k^{\text{spec}} \right\|_2}, \quad \mathbf{c}_{\text{vag}} = \frac{\frac{1}{|A_{\text{vag}}|} \sum_\ell \mathbf{a}_\ell^{\text{vag}}}{\left\| \frac{1}{|A_{\text{vag}}|} \sum_\ell \mathbf{a}_\ell^{\text{vag}} \right\|_2} \quad (3.12)$$

The semantic precision margin for requirement  $i$  is the difference between its cosine similarity to the specific centroid and its cosine similarity to the vague centroid:

$$M(i) = \mathbf{x}_i^\top \mathbf{c}_{\text{spec}} - \mathbf{x}_i^\top \mathbf{c}_{\text{vag}} \quad (3.13)$$

A positive value indicates that the requirement is more closely aligned with the precise prototype than with the vague prototype in the embedding space. The semantic margin captures requirements that are conceptually imprecise even when their surface vocabulary appears acceptable, because their embeddings cluster near the vague prototype region regardless of the specific words they use.

The anchor sets were designed to be domain-agnostic and were fixed prior to evaluation to prevent post-hoc tuning. A leave-one-out sensitivity analysis confirmed that the ranking structure is stable under single-anchor removal. The full results are reported in Section ??.

Both  $L(i)$  and  $M(i)$  are converted to percentile ranks over the collection and combined using the geometric mean:

$$S^*(i) = \sqrt{\text{rank}_{\text{pct}}(L(i)) \cdot \text{rank}_{\text{pct}}(M(i))} \quad (3.14)$$

The geometric mean enforces the same conjunctive coupling as for Structural Criticality: a requirement scores high on specificity only if both its surface language and its conceptual content are precise. Neither component alone is sufficient. The final Linguistic Specificity metric is obtained by min-max scaling:

$$S(i) = \frac{S^*(i) - \min_j S^*(j)}{\max_j S^*(j) - \min_j S^*(j) + \varepsilon} \quad (3.15)$$

The full set of  $S(i)$  values for the 262-requirement collection is provided in the replication package [2].

### Volatility Exposure

Volatility Exposure measures how exposed a requirement is to change, revision, or coordination risk based on its linguistic content. The signal is grounded in empirical evidence that certain linguistic patterns in requirement text predict a higher likelihood of change during development. Nurmuliani et al. demonstrated that requirements containing hedging, conditional, and externally dependent language are significantly more likely to be revised, and that late-stage revisions carry disproportionate cost [29]. Thakurta and Ahlemann further established that early identification of volatile requirements improves project outcomes [35]. The Volatility Exposure metric converts this knowledge into a collection-level measure computed

automatically from requirement text alone, without project history, stakeholder interviews, or domain expertise. It follows the same two-component architecture as the Specificity signal.

The lexical component  $V_{\text{lex}}(i)$  checks the requirement text against eight frozen pattern families, each targeting a different category of linguistic volatility marker. Four unigram families capture single-word signals: hedging (may, might, possibly), conditionality (if, when, unless), temporality (currently, initially, future), and coordination (external, vendor, pending). Four phrase families capture multi-word signals: coordination phrases (third-party, interface with, depends on), soft commitments (aim to, expected to, where possible), temporal phases (phase 1, phase 2), and conditionality phrases (subject to). Each family is treated as a binary indicator, and the lexical score is the proportion of the eight families that match:

$$V_{\text{lex}}(i) = \frac{\sum_{p \in \mathcal{P}_{\text{uni}}} 1[p(t_i^{\text{clean}})] + \sum_{p \in \mathcal{P}_{\text{phr}}} 1[p(t_i^{\text{clean}})]}{|\mathcal{P}_{\text{uni}}| + |\mathcal{P}_{\text{phr}}|} \quad (3.16)$$

where  $\mathcal{P}_{\text{uni}}$  and  $\mathcal{P}_{\text{phr}}$  denote the sets of unigram and phrase pattern families respectively, and  $1[p(t_i^{\text{clean}})]$  is 1 if requirement  $i$  matches pattern family  $p$  and 0 otherwise.

The semantic component  $V_{\text{sem}}(i)$  captures volatility indicators that the pattern families miss, particularly requirements that express instability through unusual vocabulary or indirect phrasing. Each requirement’s embedding is projected against two anchor sets: fifteen stable prototypes representing settled, unconditional content, and fifteen volatile prototypes representing provisional, conditional, or externally dependent content. The full anchor statements are provided in Appendix B. Both anchor sets are embedded and averaged to produce unit-scaled centroid vectors

$\mathbf{c}_{\text{sta}}$  and  $\mathbf{c}_{\text{vol}}$  using the same procedure as for the Specificity signal. The semantic volatility margin is:

$$V_{\text{sem}}(i) = \mathbf{x}_i^\top \mathbf{c}_{\text{vol}} - \mathbf{x}_i^\top \mathbf{c}_{\text{sta}} \quad (3.17)$$

A positive value indicates that the requirement’s semantic content is more closely aligned with volatile prototypes than with stable ones.

The anchor sets were fixed prior to evaluation. A leave-one-out sensitivity analysis confirmed that the Volatility semantic margin is stable under single-anchor removal, with stronger stability than the Specificity signal due to the larger anchor sets (fifteen prototypes per class compared to six). The full results are reported in Section ??.

Both components are converted to percentile ranks and combined using the geometric mean:

$$V^*(i) = \sqrt{\text{rank}_{\text{pct}}(V_{\text{lex}}(i)) \cdot \text{rank}_{\text{pct}}(V_{\text{sem}}(i))} \quad (3.18)$$

The geometric mean enforces the same conjunctive logic as for Structural Criticality and Linguistic Specificity: both lexical and semantic components must be elevated for a requirement to receive a high volatility score. Lexical markers alone may reflect domain vocabulary rather than genuine instability, and semantic alignment alone without surface confirmation is insufficient evidence of elevated risk. The final Volatility Exposure metric is obtained by min-max scaling:

$$V(i) = \frac{V^*(i) - \min_j V^*(j)}{\max_j V^*(j) - \min_j V^*(j) + \varepsilon} \quad (3.19)$$

The full set of  $V(i)$  values for the 262-requirement collection is provided in the replication package [2].

### 3.2.5 Stage 5: Multi-Objective Inspection Structuring

Because the three metrics produced in Stage 4 are designed to capture distinct dimensions of inspection risk, their combination requires a mechanism that preserves that distinctness rather than collapsing it. Scalar aggregation would treat the metrics as interchangeable contributions to a single score, allowing structural criticality to offset low volatility or vice versa, and producing a ranking that reflects aggregate exposure rather than the specific nature of each requirement's risk profile. A requirement that is highly structurally critical but precisely written and stable would rank equally with a requirement that is moderately critical, moderately imprecise, and moderately volatile if their weighted sums happened to be the same, even though the two requirements have completely different inspection profiles and demand completely different inspection responses. Stage 5 uses Pareto dominance instead, which combines the three signals into an inspection ordering while keeping their contributions structurally separate.

Before Pareto sorting can be applied, the three metric values must be oriented in a consistent direction so that larger values uniformly indicate greater inspection exposure. Structural Criticality  $C(i)$  and Volatility Exposure  $V(i)$  are already correctly oriented: higher values indicate higher inspection risk. Linguistic Specificity  $S(i)$  is oriented in the opposite direction: a higher specificity score means a more precisely written requirement, which corresponds to lower inspection risk. Stage 5

inverts the direction by computing  $1 - S(i)$ , so that higher values indicate greater linguistic imprecision. The objective vector for each requirement is therefore:

$$\mathbf{f}(i) = (C(i), 1 - S(i), V(i)) \quad (3.20)$$

Requirement  $i$  is said to dominate requirement  $j$  if it is no worse than  $j$  on all three objectives and strictly better on at least one:

$$i \succ j \iff \mathbf{f}(i) \geq \mathbf{f}(j) \text{ componentwise and } \mathbf{f}(i) \neq \mathbf{f}(j) \quad (3.21)$$

This non-compensatory property means that no single high objective value can override low values on the other two. A requirement that is highly structurally critical but precisely written and stable does not dominate a requirement that is moderately critical, imprecise, and volatile, because it is not at least as exposed on all three dimensions simultaneously. Dominance requires genuine multi-dimensional superiority.

The collection is partitioned into ordered Pareto fronts  $\{F_1, F_2, \dots, F_K\}$  through deterministic non-dominated sorting. The first front  $F_1$  contains all requirements that are not dominated by any other requirement in the collection. These are the requirements for which no other requirement is simultaneously more structurally critical, more linguistically imprecise, and more volatility-exposed. After removing  $F_1$ , the second front  $F_2$  contains the requirements that are non-dominated within the remaining set. This iterative removal continues until every requirement has been assigned to a front. Requirements in earlier fronts carry higher inspection exposure than those in later fronts, and this ordering is determined entirely by the

dominance relationships among requirements rather than by any weighting of the three metrics against each other. For the 262-requirement evaluation collection, this procedure produces 11 Pareto fronts, with  $|F_1| = 27$  requirements at the highest inspection priority tier. The full front distribution is reported in Chapter 5.

In addition to the primary Pareto-structured inspection ordering, Stage 5 produces three auxiliary queue views sorted by  $C(i)$ ,  $1 - S(i)$ , and  $V(i)$  independently. These views provide dimension-specific entry points for inspection teams that wish to understand the distribution of a particular risk metric across the collection, or that need to investigate why specific requirements appear in particular Pareto fronts. They are intended as diagnostic supplements to the primary ordering rather than standalone inspection plans. The complete inspection plan, front membership assignments, and all three auxiliary queues for the 262-requirement collection are provided in the replication package [2].

## 3.3 Experimental Design

### 3.3.1 Dataset and Domain Restriction

The evaluation is based on the publicly available tera-PROMISE requirements collection [27], which contains 625 labeled natural language requirements collected from multiple software projects across different domains. Because PRISM derives structural relationships from embedding-based semantic similarity, combining requirements from unrelated systems in a single collection would introduce artificial structural interactions. A requirement from a medical device system and a requirement from a financial trading system may share surface vocabulary without sharing

any genuine semantic relationship, and the resulting similarity scores would distort all three readiness metrics. Domain restriction to a semantically coherent subset was therefore necessary to ensure that the structural relationships captured by the embedding geometry reflect genuine collection-level dependencies rather than surface-level vocabulary overlap.

Requirements associated with an e-commerce context were identified using an AI-assisted classification procedure, with manual verification of the resulting subset to confirm domain consistency. The resulting subset contained 303 requirements spanning functional and non-functional categories. After semantic redundancy consolidation using the procedure described in Section 3.2.3 with threshold  $\tau = 0.86$ , the final evaluation collection consisted of  $N = 262$  canonical requirements. All metric parameters, anchor sets, and structural choices were fixed prior to validation to prevent post-hoc tuning. The domain annotation was used exclusively to define the evaluation subset and was not incorporated into any metric computation, normalization, structuring, or validation analysis.

### 3.3.2 Evaluation Protocol

The evaluation examines four validity properties of the PRISM framework. Each property addresses a specific threat to the credibility of the signals and the Pareto structuring mechanism. Table 3.1 summarizes the four analyses, the threat each one addresses, and the metrics used.

**Construct validity** evaluates whether each metric measures the construct it is claimed to measure. For Structural Criticality, Spearman rank correlations are computed between  $C(i)$  and three classical graph centrality measures, eigenvector,

TABLE 3.1: Summary of validation analyses, threats addressed, methods, and metrics.

| Analysis                   | Threat addressed  | Method  | Metric  |
|----------------------------|---|---|---|
| Construct validity         | Metrics may not measure what they claim                     | Correlate each signal with independent indicators from the literature | Spearman $\rho$ and Kendall $\tau$ against established comparators  |
| Metric independence        | Metrics may be redundant projections of the same construct  | Pairwise rank correlations and multicollinearity analysis             | Spearman $\rho$ , Kendall $\tau$ , variance inflation factor  |
| Structural stability       | Pareto ordering may be sensitive to collection perturbation | Bootstrap subsampling and embedding noise perturbation                | Front 1 Jaccard similarity, mean absolute front drift, Spearman $\rho$ between baseline and perturbed $C$ |
| Human behavioral alignment | Ordering may not correspond to human inspection judgements  | Blinded human study with three raters with SE backgrounds             | Spearman $\rho$ between front index and aggregated inspection support, Fisher's exact test                |

closeness, and betweenness centrality, computed over the same similarity structure. Positive correlations confirm directional alignment with established notions of structural influence. For Linguistic Specificity,  $S(i)$  is correlated with independent surface indicators including numeric quantities, measurement units, modal verbs, and vague adjectives. For Volatility Exposure,  $V(i)$  is correlated with independent instability markers including hedging expressions, conditional clauses, and temporal references.

**Signal independence** evaluates whether the three metrics capture genuinely distinct dimensions of inspection risk. Pairwise Spearman correlations and Kendall's  $\tau$  coefficients are computed among  $C$ ,  $S$ , and  $V$ , and variance inflation factors are used to assess multicollinearity. Combining dimensions that are independent of each other preserves information that a scalar ranking would suppress. The independence analysis provides the evidence that this condition is met in the evaluation collection.

**Structural stability** evaluates whether the Pareto ordering is robust to moderate perturbations of the collection and the embedding representation through two stress tests. The first is bootstrap subsampling: across 300 runs, 10% of requirements are randomly removed and the Pareto fronts are recomputed from scratch. The second is embedding perturbation: Gaussian noise with  $\sigma \in \{0.001, 0.005, 0.01\}$  is added to the embedding matrix followed by re-normalisation, and Structural Criticality is recomputed while Specificity and Volatility are held fixed, isolating the sensitivity of the purely embedding-derived signal from the lexical-pattern components that are unaffected by geometric perturbation. High Jaccard similarity and low front drift across runs confirm that the inspection ordering is stable enough to be practically reliable.

**Human behavioral alignment** evaluates whether the Pareto-structured ordering corresponds to independent human judgments of inspection priority. A blinded human study was conducted with three graduate students in software engineering, each with professional industry experience. Raters received requirement text only and were blind to all PRISM metrics and front assignments. Each rater assigned a binary inspection-worthiness label and selected a primary justification category. Labels were aggregated by majority voting and inter-rater agreement was measured using Cohen's  $\kappa$  and Fleiss'  $\kappa$ . The audit subset contained  $N = 35$  requirements, an intentionally limited size to reduce rater fatigue and preserve attention quality across the full set.

### 3.3.3 Single-Metric Baselines

To contextualize the contribution of multi-objective structuring, the Pareto-based inspection ordering is compared against top- $k$  selections derived independently from each individual metric,  $C$  alone,  $1 - S$  alone, and  $V$  alone, as well as random selection, where  $k$  matches the size of Front 1 in the audited subset. Each single-metric baseline represents the best possible outcome achievable by a tool that optimizes exclusively for one quality dimension, which include selecting the  $k$  most structurally central requirements, the  $k$  most linguistically imprecise requirements, or the  $k$  most volatile requirements, respectively. The random baseline establishes a lower bound, representing the expected inspection coverage achievable without any metric-based guidance.

This comparison is not designed to demonstrate that PRISM dominates every single-metric alternative on every metric. Such a result would, in fact, be theoretically

impossible: a top-k selection ranked by centrality will, by construction, achieve higher average centrality than any ordering not ranked exclusively by centrality. The relevant question is not which ordering scores highest on any individual dimension, but which ordering provides the most comprehensive and balanced coverage of inspection risk across all three dimensions simultaneously.

PRISM is actually designed to show whether multi-objective structuring broadens inspection-risk exposure while preserving reasonable precision, specifically whether PRISM surfaces requirements that no single metric would have selected, and whether those requirements are judged worthy of inspection attention by human raters. A requirement that ranks outside the top-k on every individual metric but nonetheless appears in Front 1 of the Pareto ordering represents a requirement whose combined risk profile is elevated across multiple dimensions simultaneously, even if it is not the worst offender on any single one. If human raters consistently judge such requirements as warranting inspection attention, this constitutes evidence that multi-objective structuring captures a qualitatively different and complementary form of inspection risk — one that single-metric approaches, by design, cannot surface.

The comparison therefore evaluates PRISM not as a replacement for single-metric tools but as a structurally distinct alternative that operates at a different level of analytical granularity. Where single-metric tools answer the question which requirements are most problematic on this dimension, PRISM answers the question of *which requirements carry the most complex and multifaceted risk profiles across all dimensions simultaneously*. This is a question that is operationally more relevant to inspection teams who must allocate finite attention across a heterogeneous collection.

## Chapter 4

# PRISM-Copilot: Signal-Conditioned Inspection

### 4.1 Overview

This chapter presents the second layer of the two-layer inspection support system introduced in Chapter 1. While Chapter 3 established that collection-level readiness metrics can be derived, validated, and used to produce a principled multi-dimensional inspection ordering, it did not address how those metrics should be used to direct automated inspection reasoning. Knowing which requirements carry the highest combined inspection risk does not automatically tell an LLM how to inspect each one differently based on its specific risk profile. This chapter addresses that question.

Signal-Conditioned Inspection Prompting (SCIP) is an architectural pattern that translates validated readiness metrics into per-requirement LLM inspection policies before any prompt is issued. Rather than sending every requirement to an LLM with the same generic inspection instruction, SCIP uses each requirement's metric profile to determine what type of inspection concern is most relevant for

that specific requirement, and constructs a targeted prompt accordingly. SCIP is instantiated in PRISM-Copilot, a working six-stage produced by Chapter 3 and produces structured per-requirement inspection observations. The evaluation of both SCIP and PRISM-Copilot is reported in Chapter 5.

This chapter is organized as follows. Section 4.2 defines the SCIP architectural pattern, its motivation, its three components, its policy mapping rules, and its handling of multiple simultaneous policy activations. Section 4.3 describes the PRISM-Copilot six-stage pipeline in full technical detail. Section 4.4 presents the experimental design used to evaluate the system, including the dataset, sampling strategy, prompting conditions, reviewer reference standard, and evaluation measures.

## 4.2 The SCIP Pattern

This section presents the SCIP pattern in four parts: the motivation for its policy layer design, its three architectural components, its policy mapping rules, and its handling of multiple simultaneous policy activations.

### 4.2.1 Motivation

The validated metric profiles produced in Chapter 3 give each requirement a numeric characterization of its structural role, linguistic precision, and volatility exposure. The natural first instinct for using these values in an LLM-based inspection system is to insert them directly into the prompt, something like: this requirement has a structural criticality score of 0.91, a specificity score of 0.23, and a volatility

exposure score of 0.78, inspect accordingly. This approach has two failure modes that make it unsuitable as a design choice.

The first failure mode is interpretive. LLMs do not reliably translate numeric collection scores into calibrated inspection behavior without explicit guidance on what those scores mean in inspection terms [37]. A score of 0.91 on structural criticality carries no actionable meaning for a model that has no knowledge of the collection distribution from which that score was derived. The model cannot know whether 0.91 is high, moderate, or typical for this collection, what structural centrality means in terms of defect propagation risk, or which inspection concern category it should prioritise in response. The number is evidence. The model needs an instruction.

The second failure mode is implementation coupling. Inserting raw signal values into prompts exposes implementation details, normalisation ranges, signal computation methodology, and collection-relative thresholds, that are irrelevant to the inspection task and introduce prompt noise. More importantly, it couples the prompting strategy to a specific signal computation method. If the signal computation changes, the prompt must change too. A policy layer that translates signal values into inspection-relevant language decouples the two: the prompting strategy can remain stable even as the underlying signal computation is refined.

SCIP addresses both failure modes by introducing a policy translation layer between the metric values and the prompt. Rather than asking the LLM to interpret numeric scores, the policy layer converts signal magnitudes into inspection instructions expressed in language the LLM can act on directly. Activation thresholds are defined relative to the empirical distribution of the collection, specifically

collection tertiles, rather than as arbitrary absolute cutoffs. This design choice ensures that policy activation reflects each requirement’s position within the actual requirements collection being inspected rather than against a fixed universal standard that may not apply to the collection at hand.

## 4.2.2 The Three Components

SCIP comprises three components that operate in sequence for each requirement.

The first component is signal consumption. Each requirement carries a pre-computed readiness vector  $\mathbf{r}_i = (C(i), S(i), V(i))$  derived from the collection-level analysis described in Chapter 3. All three metric values are normalised to  $[0, 1]$  over the full 262-requirement collection and are consumed by SCIP as fixed inputs without modification. The metric computation is entirely upstream of the prompting pipeline: SCIP does not re-derive or modify the metrics, it uses them as a validated characterization of each requirement’s risk profile.

The second component is inspection policy mapping. Metric values are translated into inspection priorities using deterministic tertile rules defined over the experiment collection. For each signal independently, the 262-requirement collection is divided into three equal-frequency groups. The top tertile of Structural Criticality, the bottom tertile of Linguistic Specificity, and the top tertile of Volatility Exposure each activate a distinct inspection policy block. When multiple rules are satisfied simultaneously, all corresponding policy blocks are included in the prompt sequentially. When no rule is satisfied, a balanced inspection fallback is applied. The full mapping is defined in Section 4.2.3.

The third component is signal-conditioned prompt construction. Each prompt consists of three fixed elements: a four-category inspection task definition that is identical across all requirements, the active policy block derived from the requirement’s metric profile, and the requirement text itself. The LLM receives the same general task definition for every requirement, but the inspection focus embedded in the policy block is specific to each requirement’s collection position. This means the prompting strategy adapts per requirement without requiring any human expert to design requirement-specific instructions in advance.

### 4.2.3 Policy Mapping Rules

The policy mapping rules translate each signal’s activation condition into a targeted inspection instruction. Table 4.1 defines the full mapping. The rationale for each rule is grounded in the construct validity results established in Section 5.2.2.

The dependency impact policy activates when a requirement belongs to the top tertile of Structural Criticality. This rule is grounded in the construct validity result for  $C(i)$  reported in Section 5.2.2, which confirmed that  $C(i)$  is positively correlated with classical graph centrality measures including eigenvector centrality ( $\rho = 0.438$ ) and closeness centrality ( $\rho = 0.570$ ). Requirements in the top tertile of  $C(i)$  are those whose embedding-based structural position most closely resembles the high-centrality requirements that prior research has shown to be disproportionately involved in integration defects [38]. Directing the LLM to focus on dependency specification for these requirements is therefore not an arbitrary choice: it is a response to validated evidence that structural centrality predicts dependency-related inspection risk.

TABLE 4.1: Metric-to-policy mapping in PRISM-Copilot. Activation thresholds are collection tertiles computed over the 262-requirement experiment collection.

| <b>Metric condition</b>      | <b>Policy</b>             | <b>Inspection focus</b>  |
|------------------------------|---------------------------|--|
| $C(i) \geq 67\text{th pct.}$ | Dependency impact         | Check whether the requirement depends on other requirements, interfaces, components, or system behaviors, and whether such dependencies are underspecified, implicit, or likely to create inspection risk. |
| $S(i) \leq 33\text{rd pct.}$ | Clarity and measurability | Check whether the requirement uses vague, underspecified, subjective, or non-testable language, and whether measurable acceptance criteria or observable conditions are missing.                           |
| $V(i) \geq 67\text{th pct.}$ | Change sensitivity        | Check whether the requirement appears sensitive to changing assumptions, policies, context, or evolving expectations, and whether such change risk is visible in the wording.                              |
| None active                  | Balanced inspection       | Review the requirement for possible ambiguity, missing measurable criteria, dependency assumptions, and change sensitivity, and report only issues clearly supported by the wording.                       |

The clarity and measurability policy activates when a requirement belongs to the bottom tertile of Linguistic Specificity. This rule is grounded in the construct validity result for  $S(i)$ , which confirmed that  $S(i)$  is strongly positively correlated with numeric digit presence ( $\rho = 0.606$ ,  $\tau = 0.496$ ), the most reliable surface indicator of testable, verifiable requirements, and negatively correlated with vague adjective presence ( $\rho = -0.086$ ). Requirements in the bottom tertile of  $S(i)$  are those whose combined lexical and semantic precision is lowest relative to the collection distribution. Directing the LLM to focus on vagueness and missing measurable criteria for these requirements responds to validated evidence that low specificity predicts misunderstanding and rework [33].

The change sensitivity policy activates when a requirement belongs to the top tertile of Volatility Exposure. This rule is grounded in the construct validity result for  $V(i)$ , which confirmed that  $V(i)$  is positively correlated with all three independent instability marker categories: conditional clause presence ( $\rho = 0.198$ ), hedging language presence ( $\rho = 0.165$ ), and temporal marker presence ( $\rho = 0.136$ ). Requirements in the top tertile of  $V(i)$  are those whose combined lexical and semantic volatility profile is highest relative to the collection. Directing the LLM to focus on change sensitivity for these requirements responds to validated evidence that volatility exposure predicts revision risk and downstream rework [29, 35].

The balanced inspection fallback applies when no signal-specific rule is activated. This occurs for requirements in the middle tertile of Structural Criticality or Volatility Exposure, and for requirements in the middle or top tertile of Linguistic Specificity. Rather than applying no guidance at all, which the no-guidance baseline already covers, the fallback provides a prompt that applies all four concern categories with

equal weight and without collection-derived prioritization. This ensures that every requirement receives structured inspection guidance while clearly distinguishing signal-conditioned from collection-blind inspection.

#### **4.2.4 Multiple Policy Composition**

A requirement may satisfy more than one activation rule simultaneously. A requirement in the top tertile of both Structural Criticality and Volatility Exposure, for example, activates both the dependency impact and change sensitivity policies. When this occurs, all corresponding policy blocks are included in the prompt sequentially, with no suppression or ranking applied between co-active policies. This design reflects the empirical orthogonality of the three metrics established in Section 5.2.3: because the metrics are independent, co-activation carries genuine information. A requirement that is simultaneously structurally central and highly volatile presents a genuinely compound inspection challenge, and both concerns deserve attention in the prompt.

### **4.3 PRISM-Copilot Architecture**

PRISM-Copilot implements the SCIP pattern as a six-stage deterministic pipeline. Each stage has a single responsibility, produces a validated output that serves as the input to the next stage, and is reproducible under fixed collection inputs. The six stages are collection ingestion, signal attachment, inspection policy mapping, prompt construction, LLM inspection reasoning, and observation parsing.

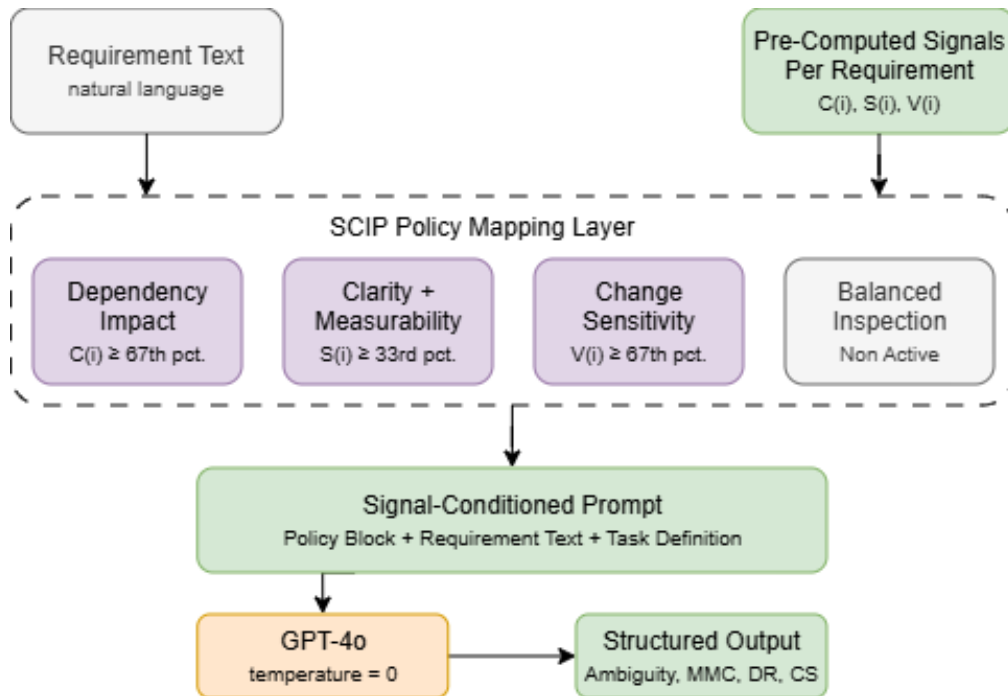


FIGURE 4.1: The SCIP policy mapping layer. Each requirement enters with its pre-computed signal vector ( $C(i), S(i), V(i)$ ) and is routed to one or more inspection policies by deterministic tertile rules.

### 4.3.1 Collection Ingestion and Signal Attachment

The pipeline begins by constructing a master file that pairs each requirement’s text with its pre-computed signal vector. Think of it as a single master spreadsheet that every subsequent stage works from: rather than each stage reaching out to different source files independently, every stage reads from the same validated file. When the requirement text is needed for prompt construction, it comes from the master file. When the signal values are needed for policy activation, they come from the master file. This means that if anything changes upstream, for example if the PRISM metrics are recomputed with a different embedding model, updating a single file is all that is required to propagate the change consistently through the entire pipeline.

The master file is built by joining two sources on a unique requirement identifier. Requirement texts are taken from the canonical 262-requirement collection produced by the PRISM pipeline in Chapter 3. Metric values,  $C_{\text{norm}}$ ,  $S_{\text{norm}}$ , and  $V_{\text{norm}}$ , are taken from the metric scores produced in Stage 4 of Chapter 3. The resulting file contains, for each of the 262 requirements, the requirement text, its class label, and its three normalized signal values. Three integrity checks are applied before any downstream stage executes: every requirement must have a unique identifier, no field may be null, and the master file must contain exactly 262 rows. If any check fails the pipeline stops immediately rather than producing unreliable output silently.

### 4.3.2 Inspection Policy Mapping and Prompt Construction

The policy mapping stage applies the tertile rules defined in Section 4.2.3 to each requirement’s signal vector. Tertile thresholds are computed at runtime from the signal value distributions of the full 262-requirement frozen collection:

the 67th percentile of  $C_{\text{norm}}$  defines the top tertile activation threshold for the dependency impact policy, the 33rd percentile of  $S_{\text{norm}}$  defines the bottom tertile activation threshold for the clarity and measurability policy, and the 67th percentile of  $V_{\text{norm}}$  defines the top tertile activation threshold for the change sensitivity policy. Computing thresholds from the frozen collection at runtime rather than hardcoding them ensures that the tertile boundaries always reflect the actual distribution of the requirements collection being inspected, which is the defining property of collection-relative activation.

The prompt construction stage assembles three fixed elements into a single prompt string for each requirement under each condition. The first element is the four-category inspection task definition, which is identical across all three conditions and defines the categories of Ambiguity, Missing Measurable Criteria, Dependency Risk, and Change Sensitivity along with their operational definitions. The second element is the inspection guidance block, which is the only component that varies between conditions. Under Condition A, the guidance block contains a neutral instruction to review the requirement carefully and report issues supported by the wording. Under Condition B, the guidance block contains the signal-derived policy block produced by the policy mapping stage. Under Condition C, the guidance block contains a single sentence of generic heuristic guidance asking the LLM to apply common requirements engineering inspection heuristics. The third element is the requirement text itself. This three-element structure isolates the contribution of signal-conditioned guidance precisely: conditions A, B, and C differ only in the guidance block, with the task definition and requirement text held constant.

### 4.3.3 LLM Reasoning and Observation Parsing

The LLM reasoning stage submits each assembled prompt to GPT-4o at temperature 0. Temperature 0 was chosen to eliminate sampling variability from the LLM outputs: at this setting, GPT-4o produces deterministic responses for identical inputs, which means that differences in output between conditions reflect differences in the prompts rather than differences in model sampling behavior.

The prompt template enforces the following output structure for every response regardless of condition:

```
Ambiguity: <issues or NONE>
Missing Measurable Criteria: <issues or NONE>
Dependency Risk: <issues or NONE>
Change Sensitivity: <issues or NONE>
```

This prompt-enforced structure is what makes automated observation parsing reliable. Rather than asking human coders to classify free-form LLM responses, which would introduce inter-rater variability and require its own agreement measurement, the parser reads each field by matching the expected label prefix and extracts a binary flag: 1 if the field contains an issue description, 0 if it contains only NONE. This design eliminates inter-rater variability from the coding step entirely. All responses parsed without format errors, confirming that the structured output schema was respected consistently by the model across all conditions and all requirements.

The parsed flags, recording which concern categories each prompting condition identified for each sampled requirement, form the observation record that all evaluation analyses in Chapter 5 are built on.

## 4.4 Experimental Design

### 4.4.1 Dataset and Sampling

The experiment draws on the 262-requirement e-commerce collection introduced in Chapter 3. Metric values for all 262 requirements were pre-computed over the full collection prior to sampling, ensuring that the tertile thresholds used for policy activation in Condition B reflect the distribution of the complete requirements collection rather than the evaluation subset alone.

A stratified sample of 50 requirements was selected using Pareto front strata as sampling units. Stratified sampling was chosen over random sampling for two reasons. First, the Pareto fronts produced by PRISM partition requirements by their combined inspection exposure, and a random sample of 50 from 262 would likely under-represent the extreme fronts, which are the requirements where signal-conditioned prompting is most likely to differ from generic prompting. Second, deterministic selection within each stratum makes the evaluation reproducible without requiring a random seed. Three priority strata partition the collection by inspection exposure: high priority covering Fronts 1 through 3 with 22 requirements selected, moderate priority covering Fronts 4 through 7 with 20 requirements selected, and low priority covering Fronts 8 through 11 with 8 requirements selected. Within each stratum, requirements are selected deterministically by requirement identifier to ensure reproducibility. (Section 3.2.5), The resulting sample of 50 requirements covers all 11 Pareto fronts and spans the full range of signal profiles present in the collection. Table 4.2 reports the per-front allocation.

TABLE 4.2: Per-front sampling allocation across three priority strata.

| Stratum           | Fronts       | Requirements selected |
|-------------------|--------------|-----------------------|
| High priority     | 1, 2, 3      | 22 (5, 8, 9)          |
| Moderate priority | 4, 5, 6, 7   | 20 (7, 5, 4, 4)       |
| Low priority      | 8, 9, 10, 11 | 8 (3, 2, 2, 1)        |
| Total             |              | 50                    |

### 4.4.2 Prompting Conditions

Three conditions isolate the contribution of signal-conditioned guidance. All conditions use GPT-4o at temperature 0. The four-category task definition, the output schema, and the requirement text are identical across conditions. The only variation is the inspection guidance block.

Condition A provides the requirement text and the four-category task definition with no additional guidance. This is the uninstructed baseline, establishing the level of inspection quality achievable by GPT-4o without any structured prompting support.

Condition B adds the signal-derived policy block to the Condition A prompt. The policy block is computed per requirement using the tertile rules defined in Section 4.2.3. Across the 50 sampled requirements, 19 activated the dependency impact policy, 18 activated the change sensitivity policy, 15 activated the clarity and measurability policy, and 8 received the balanced inspection fallback. No other prompt component changes between Condition A and Condition B.

Condition C replaces the policy block with a single sentence of generic heuristic guidance: look for unclear wording, non-testable statements, hidden dependencies, and possible sensitivity to changing assumptions or operational context. This

condition controls for the effect of any additional guidance, isolating the contribution of collection-derived signal specificity over uniform instruction.

### 4.4.3 Reviewer Reference Standard

Five graduate students with requirements engineering coursework independently reviewed all 50 sampled requirements using the same four-category scheme applied to the LLM. Reviews were conducted without access to LLM outputs or signal values. For each requirement and each concern category, a reviewer assigned a binary flag indicating whether they identified a concern of that type. A concern category is included in the reference set for a given requirement if at least three of the five reviewers independently identified it, a majority threshold chosen to filter out idiosyncratic individual judgments while preserving genuine shared concerns.

This majority threshold yielded non-empty reference sets for 31 of the 50 requirements. The remaining 19 requirements received no majority-agreed concern across any category and are excluded from recall computation. Across the 31 requirements with non-empty reference sets, the concern distributions were: ambiguity in 15 requirements, missing measurable criteria in 17, dependency risk in 17, and change sensitivity in 8. The lower frequency of change sensitivity concerns relative to the other categories is consistent with the linguistic characteristics of the collection, since change sensitivity is the hardest concern to identify from the requirement text alone without knowledge of the surrounding project context, which is reflected in the lower inter-rater agreement for this category reported in Chapter 5.

#### 4.4.4 Evaluation Measures

Two measures structure the evaluation.

The first is recall-based alignment. For each requirement with a non-empty reference set, recall is defined as the proportion of reviewer-identified concern categories that the LLM also flagged. A recall of 1.0 means the LLM flagged every concern that the majority of reviewers identified. A recall of 0.0 means it flagged none of them. Mean and median recall are reported per condition across the 31 requirements with non-empty reference sets. Paired Wilcoxon signed-rank tests assess whether differences between conditions are statistically reliable. This measure captures how well each prompting condition aligns with the aggregate human inspection reference at the individual requirement level.

The second is observation profile distribution. For each condition, the proportion of all 50 requirements for which each concern category was flagged is reported. This measure captures how signal-conditioned prompting shifts inspection attention across the full sample, independently of recall alignment. It is particularly relevant to the SCIP contribution because if signal-conditioned prompting concentrates change-sensitivity flags on high-volatility requirements rather than applying them uniformly, this shift would be visible as directional evidence of the targeting effect even when aggregate recall remains unchanged.

## Chapter 5

# Evaluation and Results

### 5.1 Overview

This chapter reports the evaluation results for both layers of the two-layer inspection support system presented in Chapters 3 and 4. Section 5.2 presents the PRISM evaluation, covering construct validity, signal independence, structural stability, and human behavioral alignment. Section 5.3 presents the PRISM-Copilot evaluation, covering inter-rater agreement, recall-based alignment, and observation profile distribution. Section ?? synthesizes what the two evaluations together establish about the two-layer architecture as a whole.

### 5.2 PRISM Evaluation

#### 5.2.1 Multi-Objective Structuring Outcome

Applying the five-stage PRISM pipeline to the 262-requirement evaluation collection produces a Pareto-structured inspection ordering consisting of 11 fronts. Table 5.1 reports the size of each front. Front 1 contains 27 requirements, the inspection

priorities for which no other requirement in the collection is simultaneously more structurally central, more linguistically imprecise, and more volatility-exposed. The front sizes decrease gradually from Front 2 onward before tapering toward the later fronts, which is consistent with the expected geometry of Pareto stratification in a three-objective space: the first front captures the requirements at the extreme of the joint risk surface, while subsequent fronts partition the remaining requirements into progressively lower-exposure tiers.

TABLE 5.1: Pareto front distribution for the 262-requirement evaluation collection. Front 1 contains the highest inspection priority requirements.

| Front | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 |
|-------|----|----|----|----|----|----|----|----|----|-----|-----|
| Size  | 27 | 42 | 48 | 34 | 27 | 23 | 18 | 17 | 10 | 11  | 5   |

Figure 5.1

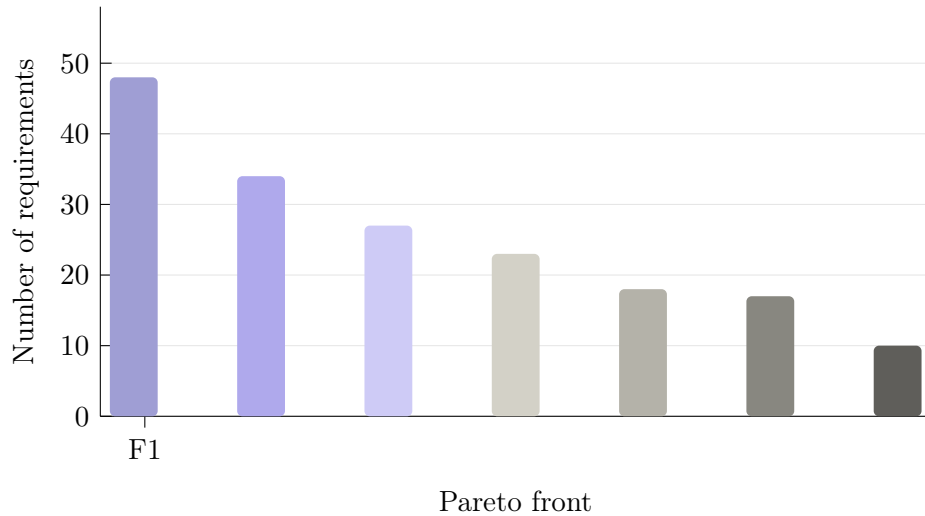


FIGURE 5.1: Pareto front distribution across the 262-requirement evaluation collection. Front 1 contains the 27 requirements for which no other requirement is simultaneously more structurally critical, more linguistically imprecise, and more volatility-exposed.

### 5.2.2 Construct Validity

The construct validity analysis evaluates whether each metric correlates in the expected direction with independent indicators of the construct it is claimed to measure.

For Structural Criticality,  $C(i)$  is compared against three classical graph centrality measures computed over the  $k$ -nearest-neighbor similarity graph of the collection. The Spearman rank correlations are  $\rho = 0.438$  against eigenvector centrality,  $\rho = 0.570$  against closeness centrality, and  $\rho = 0.440$  against betweenness centrality, with corresponding Kendall  $\tau$  values of 0.295, 0.397, and 0.304, respectively. Top-10 overlap rates are 0.70 with eigenvector centrality, 0.60 with closeness centrality, and 0.30 with betweenness centrality. These moderate positive correlations confirm that  $C(i)$  is directionally aligned with established notions of structural influence while remaining non-equivalent to any single classical measure. The non-equivalence is expected:  $C(i)$  combines three geometric primitives derived from the embedding space rather than operating on explicit graph edges, so a perfect correlation would indicate redundancy rather than a novel operationalization.

For Linguistic Specificity,  $S(i)$  is compared against independent surface indicators from the requirement text. The strongest positive correlation is with numeric digit presence ( $\rho = 0.606$ ,  $\tau = 0.496$ ), followed by measurement unit presence ( $\rho = 0.259$ ). Modal verb presence shows a weak positive correlation ( $\rho = 0.131$ ) and vague adjective presence shows a weak negative correlation ( $\rho = -0.086$ ), both consistent with the expected direction. The strong digit correlation is the most informative result: numeric values with measurement units are the most reliable surface indicator of testable, verifiable requirements, and their strong alignment with  $S(i)$  confirms

that the metric captures a genuine dimension of linguistic precision.

For Volatility Exposure,  $V(i)$  is compared against three categories of independent instability markers. Conditional clause presence yields  $\rho = 0.198$ ,  $\tau = 0.162$ . Hedging language presence yields  $\rho = 0.165$ ,  $\tau = 0.135$ . Temporal marker presence yields  $\rho = 0.136$ ,  $\tau = 0.112$ . These correlations are modest but consistently positive across all three categories, confirming directional alignment with multiple independent linguistic signals of instability. The modest magnitudes are expected: the independent markers are binary presence indicators, while  $V(i)$  integrates both lexical and semantic information through geometric mean coupling. A requirement can carry high volatility exposure without explicitly containing any of the three marker types if its semantic content aligns strongly with the volatile anchor prototype, which the lexical-only indicators cannot detect.

### 5.2.3 Signal Independence

The metric independence analysis evaluates whether the three metrics capture genuinely distinct dimensions of inspection risk. All pairwise Spearman correlations are close to zero:  $\rho(C, S) = -0.011$ ,  $\rho(C, V) = -0.104$ , and  $\rho(S, V) = -0.096$ . The corresponding Kendall  $\tau$  values are  $-0.017$ ,  $-0.071$ , and  $-0.065$  respectively. Variance inflation factors are  $VIF(C) = 1.016$ ,  $VIF(S) = 1.013$ , and  $VIF(V) = 1.022$ , all negligibly above 1.0. These results confirm that the three metrics are empirically orthogonal: no metric can be predicted from the others, and no linear combination of any two metrics accounts for meaningful variance in the third. This empirical orthogonality directly justifies the non-compensatory Pareto structuring used in

Stage 5, since the Pareto combination preserves risk information that any scalar ranking would suppress.

### 5.2.4 Structural Stability

The structural stability analysis evaluates whether the Pareto ordering is robust to perturbation of the collection and the embedding representation.

Under bootstrap subsampling, which simulates realistic collection variation by randomly removing 10% of requirements across 300 independent runs, Front 1 shows strong stability. The mean Front 1 Jaccard similarity against the full-collection baseline is 0.852, with a 5th percentile of 0.694 and a 95th percentile of 0.964. Front 1 retention is 1.000 across all runs, meaning that every requirement assigned to Front 1 in the full collection remains in Front 1 whenever it is included in the subsampled collection. The mean absolute front drift is 0.129 positions and the front-size profile L1 distance is 0.089, confirming that both individual assignments and the overall shape of the front distribution are preserved under realistic collection variation.

The embedding perturbation analysis adds Gaussian noise directly to the embedding vectors before recomputing Structural Criticality. Front 1 Jaccard similarity stabilizes around 0.46 across all three noise levels tested ( $\sigma \in \{0.001, 0.005, 0.010\}$ ), with Front 1 retention ranging from 0.629 to 0.663. The consistency of these values across noise levels indicates that the sensitivity is structural rather than noise-magnitude dependent: requirements near the boundary between Front 1 and Front 2 are susceptible to reclassification when their embedding geometry is perturbed, regardless of how much noise is applied. This is an inherent property of

any frontier-based method, since requirements at the dominance boundary are by definition the most contestable assignments, and it motivates treating boundary front assignments as approximate orderings rather than precise classifications. The bootstrap results, which reflect realistic collection variation rather than artificial geometric perturbation, remain the primary evidence for stability under practical deployment conditions.

### 5.2.5 Human Behavioral Alignment and Baseline Comparison

The human behavioral alignment analysis evaluates whether the Pareto-structured ordering corresponds to independent human judgments of inspection priority. Three raters with software engineering backgrounds assessed 35 requirements from the collection, assigning binary inspection-worthiness labels without access to any PRISM signal values or front assignments.

Inter-rater agreement was modest, with Fleiss  $\kappa = 0.198$ , which is consistent with the inherently subjective nature of inspection-worthiness judgments: experienced reviewers may legitimately prioritize different risk dimensions when evaluating the same requirement. Despite this disagreement at the individual item level, the aggregate pattern of majority labels shows a statistically significant monotonic association with Pareto front index. The Spearman rank correlation between front index and majority inspection support is  $\rho = -0.478$  ( $p = 0.0037$ , 95% CI  $[-0.681, -0.224]$ ), where the negative sign reflects that earlier fronts carry higher inspection priority. Requirements assigned to earlier fronts are significantly more likely to receive majority inspection-worthiness endorsement from independent human raters than requirements in later fronts.

Table 5.2 reports the baseline comparison results for  $k = 15$ , matching the size of Front 1 in the audited subset. PRISM Front 1 achieves a majority-positive precision of 0.733, compared to 0.867 for Top- $C$ , 0.533 for Top- $V$ , 0.533 for Top- $(1 - S)$ , and a random baseline mean of 0.467 (Figure 5.2). The difference between Top- $C$  and PRISM Front 1 precision does not reach statistical significance (paired permutation test  $p = 0.301$ , McNemar exact test  $p = 0.344$ ). Of the 15 requirements selected by each method, 10 are shared and 5 are selected exclusively by each.

TABLE 5.2: Baseline comparison results for  $k = 15$ . Precision is the proportion of selected requirements receiving majority human endorsement. High- $V$  coverage is the proportion of high-volatility requirements in the audited subset that are selected.

| Method         | Precision | High- $V$ coverage |
|----------------|-----------|--------------------|
| PRISM Front 1  | 0.733     | 0.667              |
| Top- $C$       | 0.867     | 0.333              |
| Top- $V$       | 0.533     | 0.667              |
| Top- $(1 - S)$ | 0.533     | 0.333              |
| Random (mean)  | 0.467     | n/a                |

To understand what these numbers mean in practice, consider the choice an inspection team faces. A team using Top- $C$  selects 15 requirements, of which 13 are judged by independent reviewers as genuinely worth inspecting, a strong result. But of those 15, only 5 carry high volatility exposure, meaning they are the requirements most likely to change during development and cause rework if not caught early. A team using PRISM selects a different 15, of which 11 are judged worth inspecting, slightly fewer than Top- $C$  but the difference is too small to be statistically distinguishable. Of PRISM’s 15 selections, however, 10 carry high volatility exposure. The 5 requirements that PRISM selects exclusively, the ones Top- $C$  misses, are characterized by moderate structural centrality combined with

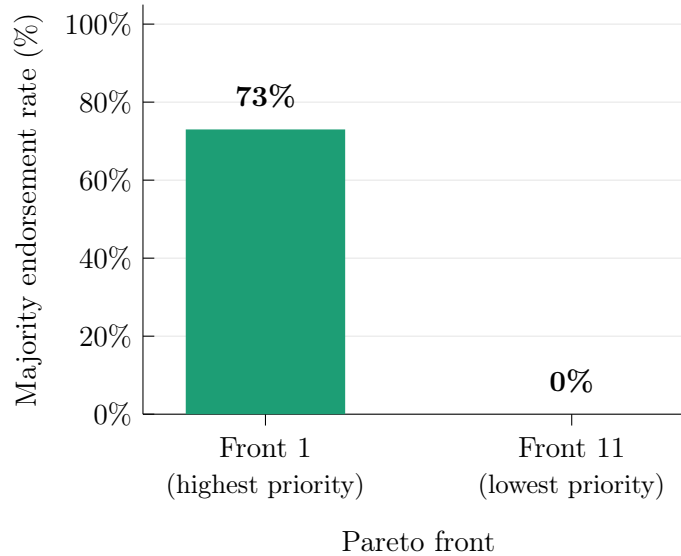


FIGURE 5.2: Human inspection-worthiness endorsement rate for Front 1 versus Front 11. Three independent raters assessed 35 requirements blind to all PRISM metric values and front assignments. Front 1 received majority endorsement from 73% of requirements while Front 11 received 0% (Fisher’s exact  $p < 0.05$ ).

high volatility exposure, a risk profile that single-signal centrality ranking cannot surface. A team using only structural centrality to guide inspection would miss half of the high-volatility requirements that PRISM identifies, inspecting requirements that are important but stable at the expense of requirements that are likely to change. These results confirm that multi-objective structuring surfaces a broader and more complete risk profile than any single-signal approach, with no statistically detectable cost in the precision of its selections.

### 5.2.6 Discussion

The five validation analyses reported in Sections 5.2.2 through 5.2.5 converge on a consistent picture. PRISM produces three metrics that are empirically distinct

from each other, directionally aligned with independent indicators of the constructs they claim to measure, and stable under realistic collection perturbation. The Pareto-structured inspection ordering derived from these metrics shows a statistically significant monotonic association with independent human inspection judgments, and multi-objective structuring surfaces a broader risk profile than any single-metric approach without a statistically detectable cost in precision. Taken together, these results support the core claim of the framework: that inspection readiness is a multi-dimensional, collection-level property, and that a non-compensatory combination of independently computed signals produces an inspection ordering that is both principled and practically aligned with how experienced reviewers think about inspection priority.

The metric independence finding deserves particular attention because it does more than validate a methodological choice. It provides the empirical justification for the entire Pareto structuring mechanism. The near-zero pairwise Spearman correlations ( $|\rho| \leq 0.104$ ) and variance inflation factors negligibly above 1.0 confirm that the three metrics measure genuinely distinct aspects of inspection risk. A requirement can be structurally central without being volatile, precisely worded without being stable, or peripheral without being precise. Each combination represents a genuinely different inspection profile, and the Pareto ordering preserves these distinctions rather than collapsing them into a single number. If the metrics were highly correlated, scalar aggregation would lose little information and the Pareto design would be unnecessary. The empirical orthogonality demonstrated here is what makes the non-compensatory combination not just theoretically motivated but empirically required.

Three limitations of the current evaluation deserve honest acknowledgement.

The first concerns human study agreement. Inter-rater agreement was modest, with Fleiss  $\kappa = 0.198$ , reflecting the inherently subjective nature of inspection-worthiness judgments. Experienced reviewers may legitimately weight structural, linguistic, and volatility risk differently depending on their background and system context. The statistically significant aggregate correlation ( $\rho = -0.478$ ,  $p = 0.0037$ ) provides meaningful evidence of alignment at the population level, but the modest item-level agreement means that individual front assignments should not be interpreted as definitive quality verdicts.

The second concerns boundary sensitivity. The embedding perturbation analysis reveals that requirements near the Front 1 boundary are sensitive to geometric perturbation of the embedding space, indicating that boundary front assignments carry more uncertainty than interior assignments. Teams applying PRISM should treat front membership as an approximate risk stratification rather than a precise ranking, particularly for requirements whose signal values place them close to a front boundary.

The third concerns generalizability. The evaluation is conducted on a single domain-restricted collection of 262 requirements from the tera-PROMISE repository. Domain restriction was methodologically necessary to ensure semantic coherence in the embedding-based structural analysis, but the generalizability of the specific signal distributions and front sizes to other domains and requirement styles remains an open question that future work should address.

Three boundaries of the PRISM contribution are also worth stating explicitly.

First, PRISM does not assert that requirements in Front 1 are necessarily

defective. It asserts only that they carry the highest multi-dimensional inspection exposure in the collection, meaning the requirements where the combination of structural influence, linguistic imprecision, and volatility risk is most elevated simultaneously.

Second, PRISM does not claim to replace human inspection judgment. The Pareto ordering identifies which requirements deserve the most careful human attention, but the quality of the inspection depends entirely on the reviewer's expertise and effort once that attention is directed.

Third, PRISM does not claim that its three metrics are the only relevant dimensions of inspection risk. They represent the dimensions for which both prior theoretical grounding and collection-level operationalization are available. Other dimensions, such as requirements traceability coverage or stakeholder disagreement history, could in principle be incorporated into the same Pareto framework if appropriate collection-level metrics could be derived.

The per-requirement metric profiles  $(C(i), S(i), V(i))$  and Pareto front assignments produced by PRISM are not merely an inspection ordering. They are a structured, validated characterization of each requirement's multi-dimensional risk profile that Chapter 4 uses as the foundation for metric-conditioned inspection reasoning. The targeting behavior that PRISM-Copilot demonstrates in Section 5.3 is meaningful precisely because the metrics validated here are reliable indicators of inspection risk. The cross-system discussion in Section 5.4 synthesizes what the two evaluations together establish.

## 5.3 PRISM-Copilot Evaluation

### 5.3.1 Inter-Rater Agreement

Five reviewers independently assessed all 50 sampled requirements using the four-category coding scheme. Fleiss’  $\kappa$  indicated fair agreement for ambiguity ( $\kappa = 0.261$ ), missing measurable criteria ( $\kappa = 0.256$ ), and change sensitivity ( $\kappa = 0.265$ ), and slight-to-fair agreement for dependency risk ( $\kappa = 0.161$ ). The lower agreement for dependency risk reflects the interpretive difficulty of identifying implicit dependencies from requirement text alone: reviewers must infer whether a dependency exists from indirect linguistic cues rather than explicit reference, which produces more variable judgments than the other three categories where the linguistic signals are more direct. Judgments were aggregated by majority vote, requiring at least three of five reviewers to agree for a concern category to enter the reference set for a given requirement.

### 5.3.2 RQ1: Recall-Based Alignment

Table 5.3 reports mean and median recall per condition across the 31 requirements with non-empty reference sets.

TABLE 5.3: Recall-based alignment by prompting condition ( $n = 31$  requirements with non-empty reference sets).

| Condition             | Mean recall | Median recall |
|-----------------------|-------------|---------------|
| A: Baseline           | 0.500       | 0.667         |
| B: Signal-conditioned | 0.532       | 0.667         |
| C: Generic heuristic  | 0.532       | 0.667         |

Both structured conditions raised mean recall from 0.500 to 0.532 over the no-guidance baseline, an increase of 3.2 percentage points. Median recall was unchanged at 0.667 across all three conditions. Signal-conditioned and generic heuristic prompting produced identical mean and median recall. Paired Wilcoxon signed-rank tests found no statistically significant differences between any pair of conditions at this sample size: B versus A ( $W = 0.0$ ,  $p = 0.317$ ), C versus A ( $W = 0.0$ ,  $p = 0.157$ ), and B versus C ( $W = 3.0$ ,  $p = 1.000$ ). These results are treated as directional trends rather than confirmed effects. The equivalent recall of Conditions B and C is expected at this scale: with 31 requirements and modest signal variance within a single collection, the statistical power to detect a difference of 3.2 percentage points is limited. The more informative question, whether signal-conditioned prompting changes where the LLM directs its attention, is addressed by RQ2.

### 5.3.3 RQ2: Observation Profile Distribution

Table 5.4 reports the proportion of all 50 requirements for which each condition flagged each concern category. Figure 5.3

TABLE 5.4: Proportion of requirements flagged per category by condition ( $n = 50$ ). MMC = Missing Measurable Criteria, DR = Dependency Risk, CS = Change Sensitivity.

| Condition             | Amb. | MMC | DR  | CS  |
|-----------------------|------|-----|-----|-----|
| A: Baseline           | 72%  | 16% | 24% | 68% |
| B: Signal-conditioned | 68%  | 18% | 24% | 60% |
| C: Generic heuristic  | 68%  | 18% | 26% | 66% |

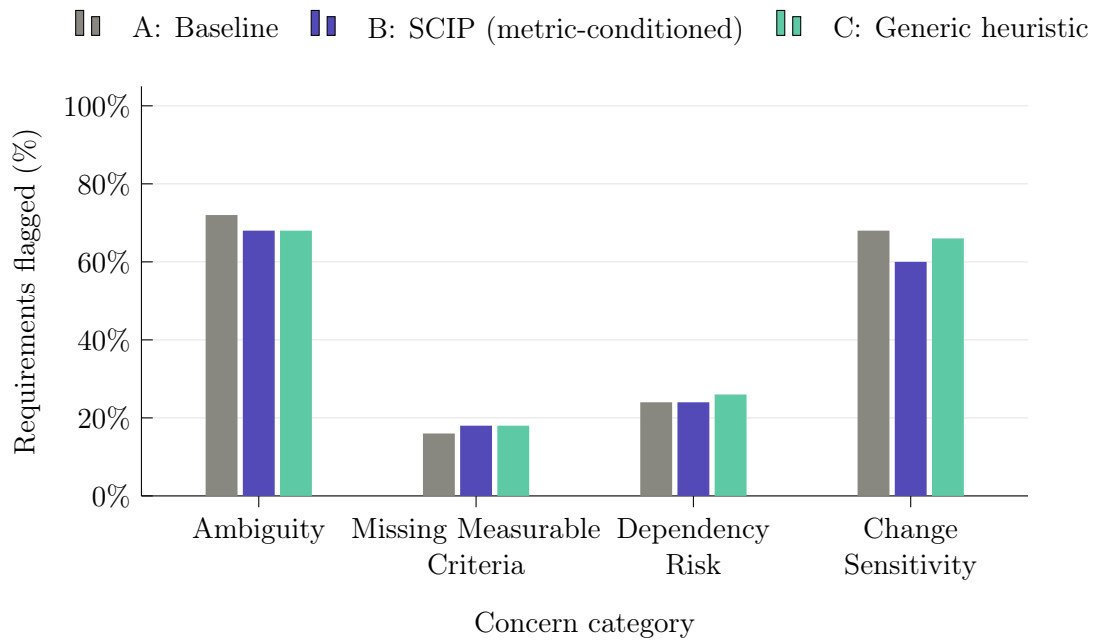


FIGURE 5.3: Proportion of requirements flagged per concern category by prompting condition ( $n = 50$ ). The key result is the Change Sensitivity column: metric-conditioned prompting (Condition B) reduces the overall flag rate from 68% to 60% while concentrating flags on high-volatility requirements, a pattern absent under generic heuristic prompting (Condition C).

The key result is in the change sensitivity column. Signal-conditioned prompting flagged change sensitivity in 60% of requirements, compared to 68% under the no-guidance baseline and 66% under generic heuristics. Understanding what this reduction means requires looking at which requirements received those flags rather than just how many.

Among the 50 sampled requirements, 18 activated the change sensitivity policy under Condition B, meaning they belong to the top tertile of Volatility Exposure with a mean  $V(i)$  of 0.770, where the collection-level signals indicate genuinely elevated change risk. The remaining 32 requirements did not activate the policy, with a mean  $V(i)$  of 0.275. Under Condition B, requirements that activated the change sensitivity policy received a change sensitivity flag rate of 66.7%, while requirements that did not activate the policy received a rate of 56.2%. The 10.5 percentage point difference between these two groups suggests that policy activation is doing meaningful work: the LLM is directing more change sensitivity attention toward the requirements where the collection-level metrics support it and less toward those where they do not.

To put this in plain language: imagine an inspection team reviewing 50 requirements. Without any guidance, the LLM raises change sensitivity concerns on 34 of them, roughly two thirds, regardless of whether those requirements carry genuine volatility signals. With signal-conditioned prompting, the LLM raises change sensitivity concerns on 30 requirements, and those 30 are more concentrated on the requirements where the collection analysis already identified elevated change risk. The team receives fewer change sensitivity flags overall, but the flags they do receive are better targeted. Generic heuristics, by contrast, reduce the overall flag

rate only slightly to 33 requirements without meaningfully concentrating those flags on the high-volatility requirements.

Dependency risk showed no meaningful variation between Conditions A and B (24% in both), confirming that dependency findings are driven by requirement text content rather than prompting strategy for the current collection. Ambiguity remained dominant across all conditions at 68–72%, consistent with the linguistic characteristics of the e-commerce collection. All differences in observation profiles are treated as directional evidence of the targeting mechanism rather than confirmed effects, consistent with the non-significant recall differences reported in Section 5.3.2.

### 5.3.4 Discussion

The results reported in Sections 5.3.1 through 5.3.3 provide directional support for the central claim of this chapter: SCIP changes what inspection attention is directed toward, not only how much the LLM finds. Signal-conditioned prompting matched generic heuristics on aggregate recall while concentrating change-sensitivity inspection flags on the requirements where collection-level signals indicate elevated volatility exposure. Requirements that activated the change sensitivity policy under Condition B received a 10.5 percentage point higher change sensitivity flag rate than those that did not, suggesting that the policy translation layer is doing meaningful work even at proof-of-concept scale. This profile shift, directing inspection attention toward the concerns most relevant for each requirement’s specific risk profile, is the empirical contribution of the chapter. It is directional rather than statistically confirmed at  $n = 31$ , and replication at larger scale is required before drawing strong conclusions.

The most predictable challenge to these results is: if aggregate recall is equivalent between signal-conditioned and generic heuristic prompting, why does signal-conditioned prompting matter? The answer has two parts. The first is that recall equivalence at this scale is expected. The evaluation collection is a single domain-restricted collection of 262 requirements with modest signal variance across the three dimensions. In this setting, generic heuristics designed to cover all four concern categories can approximate the aggregate recall of signal-conditioned prompting because the requirements are not sufficiently heterogeneous for per-requirement targeting to produce a large measurable difference in overall recall. The second part is that the targeting advantage is visible even at this scale in the observation profile, and is expected to grow as collection heterogeneity increases. In a larger industrial requirements collection where requirements span a wider range of structural positions, linguistic precision levels, and volatility profiles, the requirements at the extremes of the signal distribution are likely to be more distinct from each other, and per-requirement targeting should produce larger differentiation in both the observation profile and, eventually, in recall. Generic heuristics, which apply the same inspection logic regardless of each requirement’s collection position, become increasingly inadequate as collection heterogeneity grows. SCIP adapts automatically to each requirement’s position in the collection and requires no manual redesign as the collection grows larger or more diverse.

A secondary finding deserves acknowledgement. Both Condition B and Condition C raised mean recall by 3.2 percentage points over the no-guidance baseline. This suggests that explicit inspection guidance improves LLM behavior, regardless of whether that guidance is collection-derived or manually designed. Inspection

prompts function as reasoning scaffolds, directing the model toward established defect categories rather than relying on implicit language understanding alone. This finding is consistent with prior work on structured prompting for complex reasoning tasks [40], and it supports the requirements engineering community’s interest in structured prompt design for requirements quality tasks [36].

Three limitations of the current evaluation deserve honest acknowledgement. First, the evaluation is conducted on a single domain-restricted collection of 50 requirements drawn from the same e-commerce collection used in Chapter 3. The signal variance in this collection may be lower than in heterogeneous industrial requirements collections, and the results may not generalize to other domains, requirement styles, or LLM configurations. Multi-collection replication with professional reviewers is the primary planned mitigation. Second, the reviewer population consisted of five graduate students with requirements engineering coursework rather than professional practitioners. Their inspection behavior may differ from that of experienced industry requirements engineers, which limits the external validity of the reference standard. Third, the evaluation sample of 50 requirements limits statistical power: the recall differences between conditions did not reach significance under paired Wilcoxon tests, and the observation profile differences are interpreted as directional trends rather than confirmed effects. Larger sample sizes across multiple collections are required to assess statistical reliability.

The findings reported in this section, taken together with the PRISM validation results in Section 5.2, establish the complete empirical picture of the two-layer architecture. Section 5.4 synthesizes what the two evaluations together reveal.

## 5.4 Cross-System Discussion

The two evaluations reported in this chapter address different but logically connected questions. The PRISM evaluation asked whether collection-level signals can be derived, validated, and shown to align with human inspection judgment. The PRISM-Copilot evaluation asked whether those validated metrics can be used to condition LLM inspection reasoning in a way that concentrates attention where the collection evidence says it matters most. The answer to both questions is affirmative, and the relationship between them is not coincidental. The PRISM-Copilot targeting finding is meaningful precisely because the PRISM metrics are valid. A policy layer that routes inspection attention based on unreliable metrics would produce arbitrary rather than targeted behavior. The validation evidence in Section 5.2 is therefore not just a standalone result — it is the prerequisite that makes the PRISM-Copilot evaluation in Section 5.3 interpretable.

The two evaluations also reveal a consistent pattern of evidence at the proof-of-concept level. PRISM delivers strong evidence: the metrics are empirically orthogonal, construct-valid, bootstrap-stable, and significantly aligned with human inspection judgment. PRISM-Copilot delivers directional evidence: the targeting effect is visible in the observation profile but does not reach statistical significance at the current sample size. This asymmetry is expected and honest. PRISM was evaluated on the full 262-requirement collection with multiple independent validation analyses. PRISM-Copilot was evaluated on a proof-of-concept sample of 50 requirements under three prompting conditions. The two evaluations were designed for different purposes — validation versus feasibility demonstration — and their evidence should be interpreted accordingly.

The two evaluations establish what this thesis claims and what it leaves open. What is established is that collection-level readiness metrics can be derived automatically, validated empirically, and used to produce a principled multi-dimensional inspection ordering that aligns with human judgment, and that those metrics can be translated into targeted LLM inspection policies that, at proof-of-concept scale, show directional evidence of changing what the model inspects rather than only how much it finds. What is left open is whether the targeting advantage grows as collection heterogeneity increases, whether the signals generalize across domains and requirement styles, and whether practitioner adoption in realistic development settings produces the efficiency gains that the proof-of-concept results suggest. These open questions define the research agenda identified in Chapter 6.

## Chapter 6

# General Conclusions

### 6.1 Overview

Every software system begins as a list of written requirements, descriptions of what the system should do, how it should behave, and what constraints it must satisfy. Before any code is written, those requirements must be reviewed. This review process, called requirements inspection, is one of the most valuable quality assurance activities in software engineering because defects caught at this stage cost a fraction of what they cost to fix after the system has been built and deployed [17]. The challenge is that inspection is expensive, reviewers are limited, and specifications are large. Teams must make choices about where to focus their attention, but until now no systematic tool existed to help them make those choices. The standard practice is to review requirements one by one, applying the same level of scrutiny to every item in the specification regardless of how structurally important it is, how vaguely it is written, or how likely it is to change.

This thesis established that this uniform approach is not just inefficient. It is based on a false assumption. The assumption is that a requirement's inspection risk

can be assessed from its text alone. In reality, inspection risk is a relational property. Whether a requirement deserves urgent attention depends on where it sits within the full specification: how many other requirements depend on it, how precisely it is worded relative to its peers, and how much change pressure it accumulates from upstream dependencies. None of these properties are visible from the text of any individual requirement. They only become visible when the full collection is analyzed as a whole.

Recognizing this, the thesis proposes and evaluates a two-layer inspection support system that addresses the allocation problem at both the prioritization and the reasoning level. The first layer, PRISM, analyses the full requirements collection and computes three independent readiness metrics, Structural Criticality, Linguistic Specificity, and Volatility Exposure, that together characterize each requirement's multi-dimensional inspection risk profile. The Pareto-structured inspection ordering produced from these metrics tells an inspection team which requirements deserve the most careful attention and why. The second layer, SCIP, takes those metrics and uses them to condition LLM inspection reasoning so that each requirement receives targeted inspection guidance based on what the collection already reveals about its risk. A structurally critical requirement is directed toward dependency-focused inspection. A vaguely worded requirement is directed toward clarity and measurability-focused inspection. A volatile requirement is directed toward change-sensitivity-focused inspection. The two layers answer the question that existing approaches have left open: not just which requirements to inspect more carefully, but how to inspect each one differently based on its specific risk profile.

## 6.2 Answers to the Research Questions

### 6.2.1 RQ1: Collection-Level Metric Validity

Can analysis of a full collection of natural language requirements produce independent, valid, and stable multi-dimensional readiness metrics that align with human inspection judgment and support principled inspection prioritisation?

The evidence from Chapter 3 answers this question affirmatively and on multiple independent dimensions. The three metrics are empirically orthogonal: pairwise Spearman correlations do not exceed  $|\rho| = 0.104$  and variance inflation factors are negligibly above 1.0, confirming that each metric captures a genuinely distinct aspect of inspection risk. Each metric is construct-valid:  $C(i)$  correlates positively with classical graph centrality measures,  $S(i)$  correlates strongly with numeric digit presence ( $\rho = 0.606$ ,  $\tau = 0.496$ ), and  $V(i)$  correlates positively with all three independent instability marker categories. The Pareto ordering is stable under realistic collection variation: across 300 bootstrap runs removing 10% of requirements, Front 1 Jaccard similarity averages 0.852 with a retention rate of 1.000. Most importantly, the ordering aligns with independent human inspection judgment: the Spearman rank correlation between Pareto front index and majority human inspection endorsement is  $\rho = -0.478$  ( $p = 0.0037$ ), confirming that requirements in earlier fronts are significantly more likely to be judged worthy of careful inspection by experienced reviewers.

The answer to RQ1 is yes. Collection-level analysis can produce readiness metrics that are independent, construct-valid, bootstrap-stable, and aligned with human inspection judgment. The non-compensatory Pareto combination of these metrics

produces a principled inspection ordering that reflects all three dimensions of risk simultaneously without any dimension masking the others.

### 6.2.2 RQ2: Metric-Conditioned Inspection Reasoning

Can readiness metrics derived from a full collection of requirements be used to condition LLM inspection reasoning in a way that concentrates inspection attention on the concerns most relevant to each requirement’s specific risk profile, rather than applying uniform inspection logic across the full collection?

The evidence from Chapter 4 answers this question directionally at proof-of-concept scale. Metric-conditioned prompting matched generic heuristics on aggregate recall, both raising mean recall by 3.2 percentage points over the no-guidance baseline, with no statistically significant difference between conditions at  $n = 31$ . The more informative finding is in the observation profile. Requirements that activated the change sensitivity policy under Condition B received a 10.5 percentage point higher change sensitivity flag rate than those that did not, providing directional evidence that the SCIP policy translation layer concentrates inspection attention where the collection metrics support it. Generic heuristics applied change sensitivity attention uniformly regardless of each requirement’s volatility profile, producing a higher overall flag rate without the targeting precision that metric-conditioned prompting delivers.

The answer to RQ2 is directionally yes. Metric-conditioned prompting changes what inspection attention is directed toward, not only how much the LLM finds. This targeting effect is visible even at the modest scale of the proof-of-concept evaluation. Whether it grows as collection heterogeneity increases, the primary

hypothesis motivating the research agenda in Section 6.5, remains the central open question.

### 6.2.3 The Unified Thesis Contribution

The two research questions are logically ordered. RQ2 is only meaningful because RQ1 was answered affirmatively: a metric-conditioned inspection policy is only as trustworthy as the metrics it is conditioned on. The validation evidence in Chapter 3 is therefore not a standalone contribution but the prerequisite that makes Chapter 4 defensible.

Taken together, the two research questions and their answers establish a complete computational account of collection-level inspection readiness at the proof-of-concept level. RQ1 provided evidence that readiness metrics can be derived from collection geometry, validated against independent indicators, and shown to be empirically orthogonal in this evaluation collection, supporting the interpretation that inspection risk is genuinely multi-dimensional and cannot be reduced to a single score without losing information. RQ2 provided directional proof-of-concept evidence that those validated metrics can be translated into targeted inspection reasoning through a deterministic policy layer, and that the resulting system concentrates inspection attention where the collection evidence says it matters most. The validity of the metrics is what makes the conditioning meaningful. The targeting demonstration is what makes the metric validation practically consequential. The two layers form a principled, end-to-end approach to collection-aware inspection support, one that addresses the allocation problem not just by ranking requirements, but by changing how each one is inspected based on what the collection already reveals about its risk.

## 6.3 Contributions

This thesis makes four contributions to the requirements inspection literature.

The first is the operationalization of inspection readiness as a collection-level, multi-dimensional property that can be derived without manual annotation. Prior research studied structural criticality, linguistic specificity, and volatility exposure as separate research threads, and when these dimensions were combined, scalar aggregation was used in ways that destroyed their independence. PRISM is the first framework to operationalize all three as collection-level metrics derived from transformer embedding geometry and lexical pattern matching on requirement text alone, and to combine them through non-compensatory Pareto dominance that preserves their independence in the inspection ordering. The contribution is not the individual metric concepts, each of which has prior research behind it, but their joint operationalization as independent, reproducible, collection-level computations and the empirical evidence that the resulting ordering is valid, stable, and aligned with human inspection judgment.

The second is the empirical demonstration that structural criticality, linguistic specificity, and volatility exposure are genuinely orthogonal dimensions of inspection risk in real-world requirements specifications. The near-zero pairwise correlations ( $|\rho| \leq 0.104$ ) and variance inflation factors negligibly above 1.0 confirm that these dimensions are not projections of a shared underlying construct. This is not a methodological check. It is a finding about the nature of inspection risk. It establishes that the three dimensions must be combined non-compensatorily, because collapsing them into a scalar score would destroy information that their independence makes it possible to preserve. The Pareto structuring mechanism is the empirically correct

response to this finding, not an arbitrary design choice.

The third is the SCIP architectural pattern: a reusable design for translating collection-derived readiness metrics into targeted LLM inspection policies through a deterministic policy layer, without exposing raw numeric scores to the model. The pattern introduces a principled separation between metric computation and inspection reasoning. The policy layer converts metric values into inspection-relevant language that the LLM can act on directly, decoupling the prompting strategy from any specific metric computation method. Any collection analysis approach that produces normalized per-requirement scores on the three metric dimensions can drive the same policy mapping. This generalizability is what makes SCIP an architectural pattern rather than a feature specific to PRISM-Copilot.

The fourth is the proof-of-concept targeting demonstration: initial empirical evidence, on this evaluation collection, that collection-level metric conditioning changes the qualitative character of LLM inspection output. Metric-conditioned prompting showed directional evidence of concentrating change-sensitivity inspection flags on the requirements where collection metrics indicated elevated volatility exposure, a pattern absent under generic heuristic prompting despite equivalent aggregate recall. Requirements that activated the change sensitivity policy received a 10.5 percentage point higher change sensitivity flag rate than those that did not. This demonstration provides initial evidence that the SCIP pattern does meaningful work at proof-of-concept scale and identifies the conditions, larger collections, greater metric heterogeneity, and professional reviewers, under which the targeting advantage is most likely to grow into a statistically confirmed effect.

## 6.4 Limitations

Three limitations apply to the thesis as a whole and are worth stating precisely, both to bound the claims made in Sections 6.2 and 6.3 and to motivate the research agenda in Section 6.5.

The first limitation concerns generalizability across domains and specification styles. Both PRISM and SCIP were evaluated on a single domain-restricted collection of e-commerce requirements derived from the *tera-PROMISE* repository. This restriction was a deliberate methodological decision: embedding-based structural analysis requires semantic coherence across the collection, and combining requirements from unrelated systems would introduce artificial structural interactions that distort all three readiness metrics. The restriction was therefore necessary for the evaluation to be valid. What it creates, however, is an open question about whether the specific metric distributions, Pareto front sizes, policy activation rates, and observation profiles reported in this thesis generalize to requirements from other domains, safety-critical systems, enterprise software, embedded systems, or to specifications written in different styles, at different levels of abstraction, or by different authoring communities. The contributions of this thesis are claims about what collection-level inspection support can do in principle. Whether they hold across the full diversity of real-world specifications requires multi-collection replication.

The second limitation concerns the reviewer populations used in both human studies. The 35-requirement inspection judgment study in Chapter 3 and the 50-requirement reviewer reference standard in Chapter 4 both used graduate students with requirements engineering coursework rather than professional practitioners

with industry experience. This was a practical scoping decision consistent with the proof-of-concept framing of both studies. However, practitioner inspection behavior may differ from that of graduate students in ways that matter for the results. Experienced requirements engineers may apply different weighting to structural, linguistic, and volatility concerns than graduate students, which could affect both the human alignment result for PRISM and the reference standard quality for SCIP. The modest inter-rater agreement observed in both studies, Fleiss  $\kappa$  ranging from 0.161 to 0.265 in Chapter 4, is partly a reflection of the inherent subjectivity of inspection judgments, but it may also reflect differences in expertise that a practitioner study would help to disentangle.

The third limitation concerns the statistical power of the SCIP evaluation. The Chapter 4 evaluation was designed as a proof-of-concept demonstration rather than a confirmatory study, and its sample size of 31 requirements with non-empty reference sets limits the statistical power available to detect recall differences between conditions. The recall differences observed, 3.2 percentage points over baseline, did not reach statistical significance under paired Wilcoxon tests, and all observation profile differences are interpreted as directional trends. This limitation is not a flaw in the evaluation design. It is an honest acknowledgement of what a proof-of-concept study can and cannot establish. The targeting effect visible in the observation profile is real and consistent with the SCIP design, but its magnitude and reliability under varying conditions remain to be confirmed at larger scale.

## 6.5 Future Work

The contributions and limitations of this thesis point toward four concrete research directions. Each one follows directly from what the thesis established and what it left open.

The first direction is multi-collection replication of PRISM. The metric operationalizations, construct validity results, and Pareto stability findings reported in Chapter 3 were established on a single domain-restricted collection of e-commerce requirements. The immediate next step is to apply the same pipeline to specifications from different domains, safety-critical systems, enterprise software, embedded systems, and financial applications, and assess whether the metric distributions, pairwise independence, and bootstrap stability results hold across diverse requirement styles and authoring communities. Replication with professional requirements engineers as human raters, rather than graduate students, would also strengthen the external validity of the human alignment result and test whether the Spearman correlation between Pareto front index and human inspection endorsement generalizes beyond the current study population. Multi-collection replication is the primary empirical step required to move the PRISM contribution from a validated proof-of-concept to a broadly applicable inspection support tool.

The second direction is large-scale evaluation of SCIP across heterogeneous industrial specifications. The central hypothesis motivating the SCIP design is that the targeting advantage of metric-conditioned prompting grows as collection heterogeneity increases: as requirements span a wider range of structural positions, linguistic specificity levels, and volatility profiles, per-requirement targeting should produce larger differentiation in both the observation profile and, eventually, in

aggregate recall. Testing this hypothesis requires evaluation across multiple collections with professional reviewers, larger sample sizes sufficient to detect statistically significant recall differences, and metric distributions that span a wider range than the single e-commerce collection used in Chapter 4. The primary question is not whether metric-conditioned prompting is better than generic heuristics on average. It is whether the advantage is largest for the requirements at the extremes of the metric distribution, which is precisely where the policy layer is designed to make the biggest difference.

The third direction is practitioner integration and deployment evaluation. PRISM-Copilot currently operates as a standalone research pipeline that requires command-line execution and manual inspection of output files. Integrating it into existing requirements tooling, requirements management systems, IDE plugins, or review workflow platforms, and conducting a field study with professional engineers would bridge the gap between the proof-of-concept demonstration presented here and the industrial validation that the research agenda requires. The key evaluation questions at this stage are not statistical. They are practical. Do practitioners find the metric-conditioned inspection guidance useful? Does it change their inspection behavior in ways that reduce rework and defect escape rates? Does the additional automation overhead justify the targeting improvement in real development contexts? These questions can only be answered through deployment in realistic settings with practitioners who have genuine inspection responsibilities.

The fourth direction is metric set extension. The SCIP architectural pattern is not tied to the three metrics that PRISM computes. Any collection analysis method that produces normalized per-requirement scores on additional inspection-relevant

dimensions can drive the same policy mapping layer. Requirements traceability coverage, how completely a requirement is linked to downstream design and test artifacts, is one candidate dimension that has prior research support and could be operationalized at the collection level without manual annotation. Stakeholder disagreement history, where available from version control or issue tracking systems, is another. Review velocity, how long requirements at different collection positions have historically taken to pass inspection, is a third. Extending the metric set and testing whether additional dimensions improve the targeting precision of SCIP prompts is a natural next step that builds directly on the architectural separation between metric computation and inspection reasoning that this thesis established.

thesis

# Bibliography

- [1] Philip Achimugu et al. “A Systematic Literature Review of Software Requirements Prioritization Research”. In: *Information and Software Technology* 56.6 (2014), pp. 568–585.
- [2] Sali Hani Alsafadi et al. *PRISM Replication Package*. <https://doi.org/10.5281/zenodo.18796307>. Anonymized for double-blind review. Author identifiers will be added upon acceptance. 2026.
- [3] Aybuke Aurum and Claes Wohlin. *Engineering and Managing Software Requirements*. Springer, 2005.
- [4] Anthony J. Bagnall, Victor J. Rayward-Smith, and Ian M. Whittley. “The Next Release Problem”. In: vol. 43. 14. 2001, pp. 883–890.
- [5] Sarmad Bashir, Alessio Ferrari, et al. “Requirements Ambiguity Detection and Explanation with LLMs: An Industrial Study”. In: *Proceedings of ICSME*. 2025.
- [6] Victor R. Basili et al. “The Empirical Investigation of Perspective-Based Reading”. In: *Empirical Software Engineering* 1.2 (1996), pp. 133–164.
- [7] Daniel M. Berry, Erik Kamsties, and Michael M. Krieger. *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity*. Tech. rep. University of Waterloo, 2003.

- [8] Barry W. Boehm and Philip N. Papaccio. “Understanding and Controlling Software Costs”. In: vol. 14. 10. 1988, pp. 1462–1477.
- [9] Lionel C. Briand, Christof Differding, and H. Dieter Rombach. “Practical Guidelines for Measurement-Based Process Improvement”. In: *Empirical Software Engineering* 2.1 (1997), pp. 65–110.
- [10] Bill Brykczynski. “A Survey of Software Inspection Checklists”. In: *ACM SIGSOFT Software Engineering Notes* 24.1 (1999), pp. 82–89.
- [11] Pär Carlshamre et al. “An industrial survey of requirements interdependencies in software product release planning”. In: *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*. 2001, pp. 84–91. DOI: 10.1109/ISRE.2001.948547.
- [12] Francis Chantree et al. “Identifying nocuous ambiguities in natural language requirements”. In: *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE’06)*. Minneapolis, USA, 2006, pp. 59–68.
- [13] Jane Cleland-Huang et al. “Best Practices for Automated Traceability”. In: vol. 40. 6. 2007, pp. 27–35.
- [14] Åsa G. Dahlstedt and Anne Persson. “Requirements Interdependencies: State of the Art and Future Challenges”. In: *Engineering and Managing Software Requirements*. Springer, 2005, pp. 95–116.
- [15] Kalyanmoy Deb et al. “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

- [16] Fabrizio Fabbri et al. “The Linguistic Approach to the Natural Language Requirements Quality: Benefit of the Use of an Automatic Tool”. In: *Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop*. 2001, pp. 97–105.
- [17] Michael E. Fagan. “Design and Code Inspections to Reduce Errors in Program Development”. In: *IBM Systems Journal* 15.3 (1976), pp. 182–211.
- [18] Alessandro Fantechi, Stefania Gnesi, and Laura Semini. “Rule-Based NLP vs ChatGPT: A Preliminary Study on Ambiguity Detection in Requirements”. In: *Proceedings of the NLP4RE Workshop, co-located with REFSQ*. 2023.
- [19] Alessandro Fantechi et al. “Assisting Requirement Formalization by Means of Natural Language Translation”. In: vol. 4. 3. 1994, pp. 243–263.
- [20] Henning Femmer et al. “Rapid Quality Assurance with Requirements Smells”. In: *Journal of Systems and Software*. Vol. 123. 2017, pp. 190–213.
- [21] Alessio Ferrari et al. “LLMs for Requirements Engineering: A Systematic Literature Review”. In: *arXiv preprint arXiv:2509.11446* (2025).
- [22] Des Greer and Günther Ruhe. “Software Release Planning: An Evolutionary and Iterative Approach”. In: vol. 46. 4. 2004, pp. 243–253.
- [23] Mark Harman, S. Afshin Mansouri, and Yuanyuan Zhang. “Search-Based Software Engineering: Trends, Techniques and Applications”. In: *ACM Computing Surveys* 45.1 (2012), 11:1–11:61.
- [24] ISO/IEC/IEEE. *Systems and Software Engineering — Life Cycle Processes — Requirements Engineering*. Tech. rep. ISO/IEC/IEEE 29148:2018. International Organization for Standardization, 2018.

- [25] Abdelkarim Jahi and Abdelkarim Sami. “Evaluating LLMs for User Story Prioritization”. In: *arXiv preprint arXiv:2409.00038* (2024).
- [26] Joachim Karlsson and Kevin Ryan. “A cost-value approach for prioritizing requirements”. In: *IEEE Software* 14.5 (1997), pp. 67–74. DOI: 10.1109/52.605933.
- [27] Marcos Lima et al. “Software Engineering Repositories: Expanding the PROMISE Database”. In: *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. 2019, pp. 427–436.
- [28] Samuel Lubos, Alexander Felfernig, et al. “Leveraging LLMs for the Quality Assurance of Software Requirements”. In: *Proceedings of the IEEE International Requirements Engineering Conference (RE)*. 2024.
- [29] Nurul Nurmuliani, Didar Zowghi, and Steve Powell. “Analysis of Requirements Volatility During Software Development Life Cycle”. In: *Proceedings of the Australian Software Engineering Conference*. 2004, pp. 28–37.
- [30] Vilfredo Pareto. *Manuale di Economia Politica*. Milan: Società Editrice Libreria, 1906.
- [31] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. 2019, pp. 3982–3992.
- [32] Abdelkarim Sami et al. “Prioritizing Software Requirements Using Large Language Models”. In: *arXiv preprint arXiv:2405.01564* (2024).

- [33] Unnati S. Shah and Devesh C. Jinwala. “Resolving Ambiguities in Natural Language Software Requirements: A Comprehensive Survey”. In: *ACM SIGSOFT Software Engineering Notes* 40.5 (2015), pp. 1–7.
- [34] Forrest Shull, Ioana Rus, and Victor Basili. “How Perspective-Based Reading Can Improve Requirements Inspections”. In: *Computer*. Vol. 33. 7. 2000, pp. 73–79.
- [35] Rahul Thakurta and Frederik Ahlemann. “Understanding Requirements Volatility in Software Projects: An Empirical Investigation of Volatility Awareness, Management Approaches and Their Applicability”. In: *Proceedings of the 43rd Hawaii International Conference on System Sciences*. 2010, pp. 1–10.
- [36] Andreas Vogelsang. “From Specifications to Prompts: On the Future of Generative LLMs in Requirements Engineering”. In: *IEEE Software* (2024).
- [37] Andreas Vogelsang, Maximilian Korn, et al. “On the Impact of Requirements Smells in Prompts: The Case of Automated Traceability”. In: *Proceedings of ICSE-NIER*. New Ideas and Emerging Results Track. 2025.
- [38] Jingdong Wang and Qing Wang. “Analyzing and Predicting Software Integration Bugs Using Network Analysis on Requirements Dependency Network”. In: *Requirements Engineering* 21.2 (2016), pp. 161–184.
- [39] Liang Wang et al. “Text Embeddings by Weakly-Supervised Contrastive Pre-training”. In: *arXiv preprint arXiv:2212.03533* (2022).

- [40] Jason Wei et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”. In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022, pp. 24824–24837.
- [41] Yuanyuan Zhang, Mark Harman, and S. Afshin Mansouri. “The Multi-Objective Next Release Problem”. In: *Proceedings of GECCO*. 2007, pp. 1129–1137.
- [42] Didar Zowghi and Nur Nurmuliani. “A study of the impact of requirements volatility on software project performance”. In: *Proceedings of the Ninth Asia-Pacific Software Engineering Conference (APSEC 2002)*. Gold Coast, Australia, 2002, pp. 3–11.

## Appendix A

# Replication Package

The complete replication package for both studies is publicly available on Zenodo.

- **PRISM** (Chapter 3): <https://doi.org/10.5281/zenodo.18796307>
- **PRISM-Copilot** (Chapter 4): <https://doi.org/10.5281/zenodo.19211675>

The package includes:

- The 262-requirement canonical corpus (`scoring_set.csv`)
- Pre-computed signal vectors (`C_scores.csv`, `S_scores.csv`, `V_scores.csv`)
- All pipeline scripts (`step1` through `step5` and all validation scripts)
- PRISM-Copilot experiment scripts (`01_build_corpus_master.py` through `11_paired_tests.py`)
- All prompt templates (`prompt_A.txt`, `prompt_B.txt`, `prompt_C.txt`)
- LLM outputs and practitioner coding sheets

## Appendix B

# Anchor Statements

The following anchor statements are used in the Specificity and Volatility signal computations (Stage 4, Section 3.2.4). All anchors were fixed prior to evaluation and are reproduced here for transparency and replication.

### C.1 Specificity Signal — Specific Anchors ( $S^+$ )

1. The system shall respond within 50 ms of a user click.
2. The temperature sensor shall report values with  $\pm 0.1$  degree accuracy.
3. The database shall handle 10,000 transactions per second at 99% uptime.
4. The interface shall encrypt stored records using AES-256.
5. The system shall retain audit logs for a minimum of one year.
6. The service shall achieve 99.9% availability per month.

## C.2 Specificity Signal — Vague Anchors ( $S^-$ )

1. The system should be fast and user friendly.
2. The software shall have a high quality interface.
3. The product should be better than the previous version.
4. The application must be as efficient as possible.
5. The system should be robust and reliable in most environments.
6. The product must comply with all relevant laws and regulations.

## C.3 Volatility Signal — Stable Anchors ( $V^-$ )

1. The system shall store a 64-bit transaction identifier.
2. The product shall utilize PostgreSQL version 14.
3. The API shall return data in a validated JSON format.
4. The system shall utilize AES-256 for data encryption.
5. The application shall display the logo in the header.
6. The system shall support the IPv6 protocol stack.
7. The middleware shall implement a persistent message queue.
8. The server shall listen for incoming traffic on port 443.
9. The audit trail shall record an immutable timestamp.

10. The system shall respond to ICMP heartbeat pings.
11. The application shall use UTF-8 character encoding.
12. The database shall maintain a connection pool of 20.
13. The system shall backup data to a redundant S3 bucket.
14. The product shall implement Role-Based Access Control.
15. The system shall be deployed on a Linux-based kernel.

## C.4 Volatility Signal — Volatile Anchors ( $V^+$ )

1. The requirement is subject to further stakeholder review.
2. The feature will be developed in Phase 2 or later.
3. The integration depends on a third-party vendor's API.
4. The system currently uses a library which may be replaced.
5. Subject to branding guidelines, the UI might change.
6. If the budget allows, the system should include analytics.
7. The product may need to comply with upcoming regulations.
8. The integration is contingent upon external hardware delivery.
9. The capacity is initially set to 100 concurrent users.
10. Unless the license is revoked, we will use this framework.

11. The system is expected to accommodate future tech shifts.
12. The final architecture depends on the security audit results.
13. The software aims to support legacy formats where possible.
14. This requirement is pending approval from the legal team.
15. The implementation might change during the prototype phase.