

Conjecture and Inquiry: Quantifying Software Performance Requirements via Interactive Retrieval-Augmented Preference Elicitation

Shihai Wang^{1*}, Tao Chen^{2†}

¹ School of Computer Science and Engineering, UESTC, Chengdu, China

² IDEAS Lab, University of Birmingham, Birmingham, UK
wsh2130076635@gmail.com, t.chen@bham.ac.uk

Abstract

Since software performance requirements are documented in natural language, quantifying them into mathematical forms is essential for software engineering. Yet, the vagueness in performance requirements and uncertainty of human cognition have caused highly uncertain ambiguity in the interpretations, rendering their automated quantification an unaddressed and challenging problem. In this paper, we formalize the problem and propose IRAP, an approach that quantifies performance requirements into mathematical functions via interactive retrieval-augmented preference elicitation. IRAP differs from the others in that it explicitly derives from problem-specific knowledge to retrieve and reason the preferences, which also guides the progressive interaction with stakeholders, while reducing the cognitive overhead. Experiment results against 10 state-of-the-art methods on four real-world datasets demonstrate the superiority of IRAP on all cases with up to 40× improvements under as few as five rounds of interactions.

1 Introduction

Software project failures often stem from unmet behavioral requirements of the software performance (Eckhardt et al., 2016a; Sayagh et al., 2020). One example is the U.S. Health Care failure in 2013 (U.S. Department of Health and Human Services, 2016), in which the vague performance preferences from the stakeholders have not been properly quantified into precise metrics, causing the software to be unable to cope with the required load surges, leading to severe financial loss.

While quantifying performance requirements is crucial for modeling (Xiang et al., 2026; Gong et al., 2025), tuning (Ye et al., 2026; Chen and Chen, 2026; Chen et al., 2024), testing (Ma

et al., 2025; Du and Chen, 2025), and self-adaptation (Ye et al., 2025; Chen et al., 2018), it is a challenging task since those requirements are often documented in natural language statements, which are vague and imprecise (Eckhardt et al., 2016a; Chen and Li, 2023; Xiong and Chen, 2025). For example, the real-world requirement 4.3.1 for the Puget Sound Enhancements System in the PURE dataset (Ferrari et al., 2017) is:

“The system should support at least 1000 concurrent users.”

Although there is a threshold of 1,000 users, it remains unclear to what extent the throughput is tolerable if it drops below 1,000, nor whether above 1,000 is equally acceptable. Even worse, commonly the stakeholders who give the natural performance requirements cannot precisely quantify their needs without assistance due to the high uncertainty of human cognition (Glinz, 2007; Chung et al., 2000). Such an **uncertain ambiguity** is what makes quantifying the preferences in performance requirements difficult.

Manually quantifying performance requirements with domain-specific language is tedious, expensive, and prone to inconsistency, especially for large projects (Eckhardt et al., 2016b; Whittle et al., 2010). Specific and automated rule-based methods like LQPR (Wang and Chen, 2026) can assist in the quantification, but still, they assume that the stakeholder would give all necessary information in the requirements and hence struggles to fully handle the uncertain ambiguity therein.

Existing generic methods for preference alignment have been leveraging Large Language Models (LLMs), pairing with paradigms such as Reinforcement Learning (Ouyang et al., 2022) or Retrieval-Augmented Generation (RAG) (Borgeaud et al., 2022), some of which also contain limited human feedback. However, their key limitation is they either only work on

*Shihai Wang is also supervised in the IDEAS Lab.

†Tao Chen is the corresponding author.

general preferences which are too costly when used for subjective, case dependent preferences from individual stakeholder in our problem (Howell et al., 2023); or they solely rely on the LLM’s capability to analyze the preferences without explicitly reasoning on the unique properties in the problem, i.e., in our case these are the patterns of the performance requirements. Further, their inability to continually interact with humans for progressive feedback makes coping with the uncertain ambiguity unrealistic.

To overcome the above gaps, in this paper, we propose IRAP, an approach that quantifies software performance requirements as mathematical functions via interactive retrieval-augmented preference elicitation involving the stakeholder—a typical *conjecture and inquiry* loop—which is important to resolve the uncertain ambiguity therein, since even the stakeholder is uncertain about his/her true preferences at the beginning. What makes IRAP unique is that while achieving **retrieval reasoning-guided interaction** based on explicit knowledge of the problem, it also strikes on several aspects to **minimize the cognitive overhead** of the stakeholder, e.g., via suggesting a quantification that is likely similar to one’s true preferences and providing intuitive questions during interaction. Specifically, our contributions are:

- **Theory:** We formulate the preference reasoning problem for quantifying performance requirements based on the defined notions of **precision** and **difficulty** therein.
- **Quantification:** We propose retrieval-generative quantification to automatically quantify an initial draft of a given natural performance requirement via dual classification and generation based on patterns and LLM.
- **Reasoning:** To reduce the cognitive overhead, we propose retrieval-analogical preference reasoning that uses past examples as analogies to convert the quantification of the initial draft, hence it is closer to what is most likely to be preferred by a stakeholder.
- **Interaction:** Using the converted/reasoned quantification as a starting point, we present a tree-based question-answering interaction with the stakeholder to tune the preferences.

To evaluate IRAP, we compare it against 10 state-of-the-art methods under four real-world datasets of performance requirements. The results show that, through all the proposed mech-

anisms, IRAP significantly outperforms the others with quantification up to 40× closer to the ground truth while doing so under minimal cognitive effort. To promote open science, all data and code can be accessed at our anonymous repository: <https://github.com/ideas-labo/irap>.

2 Related Work

Requirements analytics and quantification. Requirements automation has evolved from formal methods (Whittle et al., 2010; Baresi et al., 2010) to deep semantic analysis via neural language models (Hey et al., 2020; Luo et al., 2022) and LLMs (Binkhonain and Alfayaz, 2025). However, these approaches predominantly focus on classification rather than quantification. While LQPR (Wang and Chen, 2026) is a pioneering attempt at rule-based predictive quantification, it permits neither preference reasoning nor interaction at all—the key contributions in this work.

Preference alignment methods based on reinforcement learning. Methods exist for ensuring the alignment between model outputs and human intentions. Current mainstream paradigms leveraging Reinforcement Learning (RL) (Christiano et al., 2017; Ouyang et al., 2022), such as DPO (Rafailov et al., 2023) and WPO (Zhou et al., 2024), aims to reward the model behaviors that are close to the inputs’ needs. A key limitation of those is that they completely leverage LLM to reason the general human preferences (with or without fine-tuning), hence likely to miss the subjective preferences implied in specific cases and cause high uncertainty, especially for performance requirement quantification, which has highly uncertain ambiguity (Wang and Chen, 2026). Although existing works have attempted to achieve personalized alignment via fine-tuned adapters (Cheng et al., 2024)), maintaining independent models for each user incurs unrealistically excessive costs (Howell et al., 2023).

Preference alignment methods based on RAG. Other methods leverage RAG and LLM in-context learning to achieve dynamic alignment. Often, they iteratively instruct LLM with retrievals to effectively guide model behaviors (Borgeaud et al., 2022; Lewis et al., 2020). Nevertheless, existing RAG methods again rely on the implicit instruction of LLMs to integrate bounded context information, without explicitly modeling problem-specific logic of reasoning preferences—this is

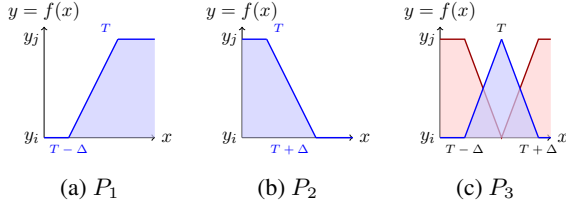


Figure 1: Patterns of the quantification functions for performance requirements.

detrimental for handling uncertain ambiguity in quantifying performance requirements.

3 Theory

3.1 Quantifying Performance Requirements

Inspiring from a recent study (Wang and Chen, 2026), we found that the requirement on a performance metric exhibits three linear patterns of satisfaction ($y \in [0, 1]$), as shown in Figure 1:

- P_1 : Higher value (x) up to a threshold T is the most preferred (y), and there is a certain tolerance if the value is lower than the threshold (e.g., “Throughput needs to support above 100 req/s”).
- P_2 : Lower value down to a threshold T is the most preferred, and there is certain tolerance if the value is higher than the threshold (e.g., “Response time is less than 5s”).
- P_3 : An exact value is the most preferred/non-preferred; anything lower/higher has some tolerance (e.g., “Refresh rate shall be equivalent to 5s/time”).

As such, a performance requirement can be quantified as a piecewise function following one (or more) pattern from the above, centered by a threshold/value. Here, the y-axis represents the stratification of stakeholders across the permissive values of a performance metric at the x-axis. For example, consider the natural requirement statement:

“The software must receive and process ECG signal data at a frequency of no less than 1000Hz.”

Clearly, this requirement is vague, as it is not clear to what extent the data frequency less than 1000Hz is tolerable, nor the preference of those that exceed 1000Hz. One possible natural interpretation can be “anything better than 1000Hz is equally preferred while there is a tolerance of 10%, after which it is unacceptable”. Therefore, the piecewise function of quantification $f(x)$ can be written

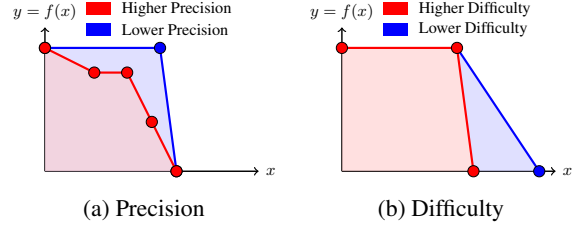


Figure 2: Examples of differences on precision and difficulty on performance requirements.

as:

$$y = f(x) = \begin{cases} 0 & \text{if } x \leq 900 \\ \frac{x-900}{100} & \text{if } 900 < x < 1000 \\ 1 & \text{if } x \geq 1000 \end{cases} \quad (1)$$

The above follows P_1 ($T = 1000$, $\Delta = 10\% \times T$). Note that it is possible to have a performance requirement that exhibits multiple patterns¹, which are joint with several thresholds (and Δ) too (but with their satisfactions adjusted accordingly).

3.2 Preferences in Quantification

Indeed, a performance requirement can be interpreted in different ways, and hence its quantification differs. This is highly dependent on what we call preferences of the stakeholders in the requirements, which can be expressed in two aspects:

- **Precision:** This reflects the number of pattern types (and their thresholds) that can be inferred from a performance requirement. Intuitively, a requirement with more patterns leads to a more complicated piecewise function in the quantification, and hence it can be quantified more precisely. With the same example, if there is an additional threshold, e.g., “...should ideally be 1500Hz”, then the quantification would have four pieces. An illustration has been shown in Figure 2a.
- **Difficulty:** This refers to the value(s) of T , Δ , and/or satisfaction from the pattern type(s) when quantifying the requirements. Using the same example before, $\Delta = 10\% \times T$ is clearly harder to satisfy than setting $\Delta = 50\% \times T$, as illustrated in Figure 2b.

It is worth noting that, commonly, even the stakeholders themselves might not have a clear understanding of their own preferences at the beginning, since without observing the actual quantification, the exact meaning of the performance re-

¹The number of patterns is often consistent with the number of thresholds (Wang and Chen, 2026).

quirement statements remains cognitively fuzzy in one’s mind.

3.3 Problem Formalization

Given the quantification pattern types and the meaning of preferences therein, our goal is to automatically elicit a stakeholder’s truly preferred quantification of a performance requirement in an interactive manner. To that end, for quantifying a given target performance requirement, we formulate this as a finite state transition problem:

$$f_{t,0} \xrightarrow{op_1} f_{t,1} \xrightarrow{op_2} \dots \xrightarrow{op_n} f_t^* \quad (2)$$

whereby $f_{t,0}$ and f_t^* are the initial and the truly preferred quantification (state) of the performance requirement, respectively; all quantifications between those are intermediate ones. op_n denotes the n th operation that includes **ADD**, **REMOVE** pattern types (for controlling precision), and **CHANGE** the values, i.e., T , Δ , and the possible satisfactions y_i/y_j (for controlling difficulty). With this, our aim is to find the quantification that the stakeholder fully agrees on as soon as possible, i.e., reaching f_t^* with the smallest possible n .

4 IRAP Framework

The key property in IRAP is that, with the support of problem-specific knowledge, it not only fosters retrieval reasoning-guided interactive quantification for performance requirements in natural language, but does so with minimized cognitive overhead of the stakeholder, finding a preferred quantification quicker. As in Figure 3, it has three interrelated phases: (1) A retrieval-generative quantification that converts a given natural requirement statement into the mathematical function of an initial draft quantification; (2) a retrieval-analogical preference reasoning that extracts the past examples as analogies to align the initial quantification closer to what is most likely to be preferred by a stakeholder; (3) mining and tuning preferences via interaction with the stakeholder.

4.1 Retrieval-Generative Quantification

From Section 3.1, we can formalize the patterns in the forms of points:

- $P_1: [(T - \Delta, y_i), (T, y_j)]$
- $P_2: [(T, y_i), (T + \Delta, y_j)]$
- $P_3: [(T - \Delta, y_i), (T, y_j), (T + \Delta, y_i)]$

Although a requirement might contain more than one pattern, in practice, they can be decomposed

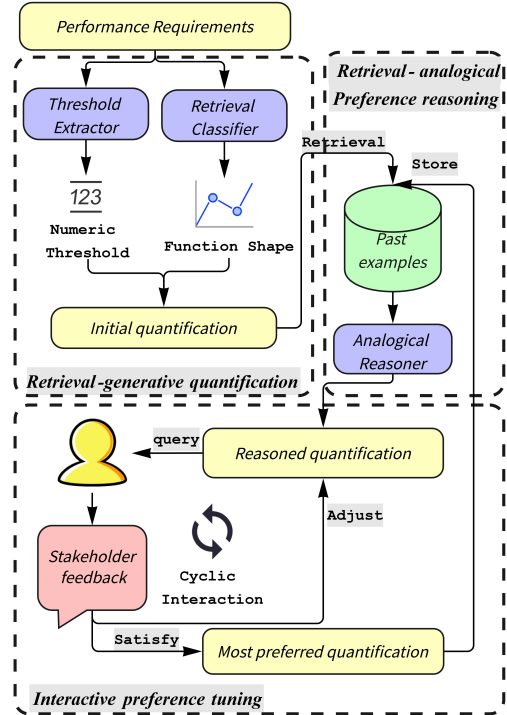


Figure 3: The workflow of IRAP.

into different fragments where each is an independent requirement and involves exactly one such pattern/threshold (Wang and Chen, 2026). Thus, we model the first phase of quantification as dual tasks: (1) firstly classify the requirement into one pattern via retrieval-based classification; and (2) extract the threshold value using LLM generation. Note that, unlike the threshold T , Δ (default to $\Delta = 10\% \times T$) is often implicit and hence can only be adjusted later. This would lead to an initial draft quantification function $f_{t,0}$ for the target.

For example, the performance requirement “The recommendation accuracy should not be less than 85%” can be classified as $P_1: [(85 - \Delta, 0), (85, 1)]$ where $T = 85$.

4.1.1 Retrieval-based Classification

Unlike classic natural language classification, where the class label is formulated as independent one-hot indices without any of their semantic information, IRAP embeds the semantic knowledge of the label into the classification, which has been shown to be superior (Wang et al., 2018). This is because, as studied before (Wang and Chen, 2026), performance requirements exhibit clear phrases that can pinpoint their patterns, serving as a solid foundation for incorporating the semantics of labels. Specifically, we firstly extract 10 key phrases from known performance requirements for each

Pattern Type	Anchor Phrases
P_1	“no less than”, “at least”, “greater than”, ...
P_2	“no more than”, “at most”, “less than”, ...
P_3	“exactly”, “precisely”, “equivalent to”, ...

Table 1: Exemplated anchors; full list is at Appendix A.1.

pattern from Section 3.1, as shown in Table 1, representing their respective semantic information. Those phrases serve as the anchors.

Secondly, since those anchors might not be exhaustive, we seek to extract the semantic meaning from a performance requirement and match it with the semantics of each anchor, which outperforms simple syntactical matching. To that end, an embedding model is needed, and we use the RoBERTa (Liu et al., 2019). However, directly using existing fine-tuning loss is ill-suited, because we found that standard fine-tuning struggles to distinguish between antonymous anchors like “at least” (P_1) and “at most” (P_2) that appear in identical contexts, hence we need stronger, specifically crafted guidance to discriminate those cases. To resolve that, we fine-tune the RoBERTa using a contrastive loss extended from the InfoNCE loss (van den Oord et al., 2018), aiming to globally maximize the average similarity between the input embedding of requirements/anchors and all their matching patterns while penalizing the similarity with all non-matching ones, bridging the requirements and anchors. The loss function is:

$$\mathcal{L}(\mathcal{S}) = \sum_{s_i \in \mathcal{S}} \frac{1}{|\mathcal{P}_i|} \sum_{p \in \mathcal{P}_i} \left(-\log \frac{\exp(\text{sim}(s_i, p)/\tau)}{\sum_{a \in \mathcal{A}} \exp(\text{sim}(s_i, a)/\tau)} \right) \quad (3)$$

where $\text{sim}(u, v)$ denotes cosine similarity, τ is the temperature parameter. s_i denotes a performance requirement/anchor. \mathcal{P}_i is the set of matching patterns for s_i , and \mathcal{A} denotes all patterns. In this way, the obtained semantics of both the requirement statement and the anchor phrases would contain strong semantics of the labeled pattern class.

Finally, we calculate the cosine similarity between the embedding of the given performance requirement and each of the anchors, assigning the pattern of the anchor with the highest similarity.

4.1.2 Generative Threshold Extraction

While we are able to classify a performance requirement statement into patterns, the threshold still needs to be identified. To extract the threshold, IRAP achieves such via fine-tuning a LLM with full parameters, i.e., GPT-2 (774M) in this case, as

not all numbers in the requirement statement are thresholds, and we need to have a comprehensive understanding of the statement. Notably, GPT-2 is chosen since we do not want an over-complex heavy model, and GPT-2 is lightweight, efficient, and with sufficient accuracy. The sample/prompt can be found at Appendix A.2

4.2 Retrieval-Analogical Preference Reasoning

A naive approach would be to directly use the $f_{t,0}$ from retrieval-generative quantification for the stakeholder to start interacting with. However, this could incur large cognitive overhead as the generative one might still be far away from one’s true preference. Since each stakeholder can propose many performance requirements, IRAP seeks to gradually align the subjective preferences of the stakeholder deriving from the previously quantified requirements, hence creating a better starting point of interaction for the newly given requirement statement s_t that can reach f_t^* quicker. Here, assuming that the quantification of a requirement can be represented by z points, e.g., $f_{k,0} : \{(x_{k,1}, y_{k,1}), \dots, (x_{k,z}, y_{k,z})\}$, we retrieve the most semantically-similar² past requirement s_k to the s_t , such that the initial quantification $f_{k,0}$ of s_k has the same number of points as $f_{t,0}$, from those that have been quantified by IRAP. Formally, s_k is expressed as:

$$s_k = \{f_{k,0}, f_k^*\} \quad (4)$$

whereby $f_{k,0}$ and f_k^* denote the initial quantification and the finally accepted, most preferred quantification, respectively. Our goal is to extract the proper operations for converting from $f_{k,0}$ to f_k^* as the analogy, and then apply them to $f_{t,0}$.

Yet, since the set of operations we can use for transferring from one quantification to another is vast especially when they have different pieces/points, finding the operations for converting from $f_{k,0}$ to f_k^* is not straightforward. Suppose that we have $f_{k,0} : \{(9, 0), (10, 1)\}$ and $f_k^* : \{(8.5, 0), (9.5, 0.5), (10.5, 1)\}$. Now, from $f_{k,0}$ to f_k^* we can either add (9.5, 0.5), change (9, 0) to (8.5, 0), and (10, 1) to (10.5, 1); or add (8.5, 0), change (9, 0) to (9.5, 0.5), and (10, 1) to (10.5, 1). Both are valid, but their distances differ, e.g., us-

²We embed each past requirement and s_t using BERT and compute their cosine similarity.

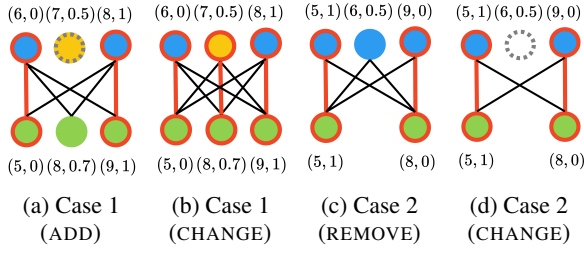


Figure 4: Points alignment (a and c) and changes identification (b and d). Mappings are highlighted.

ing edit distance³, the former has 3 while the latter is 4. Clearly, the former is wiser with smaller magnitudes of changes on the quantification function.

To accurately measure the converting distance from $f_{k,0}$ to f_k^* and obtain their operations, we propose a path-aware operation extraction (PAOE), such that not only the actual change, but also the weights/costs with the change, are included. We formalize this as a bipartite graph maximum weight matching problem (Kuhn, 2010) based on a graph for $f_{k,0}$ and f_k^* :

- **Vertices:** As in Figure 4, we list the points in $f_{k,0}$ as the top vertex set $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$, and list those in f_k^* to the bottom vertex set $\mathcal{V} = \{v_1, v_2, \dots, v_m\}$.
- **Edges and weights:** The set \mathcal{E} contains edges connecting any node in \mathcal{U} with any node in \mathcal{V} . The weight w_{ij} of the edge (u_i, v_j) represents the “matching preference” between point u_i and v_j . We define the edge weight as the negative Euclidean distance:

$$w_{ij} = -\sqrt{(x_{u_i} - x_{v_j})^2 + (y_{u_i} - y_{v_j})^2}.$$

We seek to find a matching \mathcal{M} in the graph (i.e., a subset of edges, where any two edges do not share a vertex), such that the sum of the weights of all edges within the matching is maximized:

$$\arg \max_{\mathcal{M}} \sum_{(u_i, v_j) \in \mathcal{M}} w_{ij} \quad (5)$$

The above can be solved by the well-known Kuhn-Munkres (KM) algorithm (Kuhn, 2010).

Drawing on the most semantically-similar requirement s_k and PAOE, the key steps of retrieval-analogical preference reasoning in IRAP are:

1. **Points alignment:** We firstly employ the KM algorithm to find the optimal matching be-

³We count adding/ removing the whole point and changing each value as independent operations, e.g., adding $(9, 0)$ is one operation; changing $(9, 0)$ to $(9.5, 0.5)$ would be two operations as there are two value changes.

tween points in $f_{k,0}$ and f_k^* . If there are more points in $f_{k,0}$ than in f_k^* , then the unmatched points in $f_{k,0}$ are subject to REMOVE (Figure 4c). When there are more points in f_k^* than in $f_{k,0}$, we identify the unmatched points in f_k^* as those that require ADD the corresponding points in $f_{k,0}$ and set their x/y by averaging those of the two adjacent points, e.g., $(7, 0.5)$ in Figure 4a.

2. **Changes identification:** The KM algorithm is then run again on the aligned points. The final matching of points with different values (including newly added points) would indicate those that need CHANGE on T , Δ , and/or the satisfaction y_i/y_j (Figures 4b and 4d).
3. **Operations sequencing:** All the operations incurred from the previous two steps serve as the analogy, in which the ADD and REMOVE are always placed before CHANGE⁴.

As such, the extracted operations can be applied to the $f_{t,0}$ for obtaining a new initial and reasoned quantification $f'_{t,0}$ for the interaction phase. The detailed pseudo code can be found at Appendix B.

Taking Case 2 in Figure 4 as an example, the operations are REMOVE (the second point) and CHANGE (decrease the x of the new second point⁵). Suppose that $f_{t,0} : \{(25, 1), (30, 0.5), (40, 0)\}$, then applying the two operations has:

$$\{(25, 1), (30, 0.5), (40, 0)\} \xrightarrow{op1} \{(25, 1), (40, 0)\} \\ \xrightarrow{op2} \{(25, 1), (36, 0)\}$$

leading to the new $f'_{t,0} : \{(25, 1), (36, 0)\}$.

PAOE is also efficient, e.g., for processing requirement under 40+ past examples, it takes < 0.2 seconds on a 2.8GHz and 8GB RAM machine.

4.3 Interactive Preference Tuning

With the converted/reasoned $f'_{t,0}$, IRAP can then start iteratively interactive quantification with the stakeholder. Here, we follow tree-based multiple choice questions-answering for preference tuning, since this can significantly reduce stakeholders’ cognitive load and improve efficiency (Polat, 2020), leading to a most preferred quantification f_t^* quicker. A snippet of the questions has been

⁴This is a simple and pragmatic choice to eliminate the chance of having an invalid operation sequence, e.g., we cannot change a value before it is added. Without this, we would need a more expensive strategy to repair the above case, which is not ideal.

⁵The extent of change is adjusted and bounded by the function, starting with 10% of the original value.

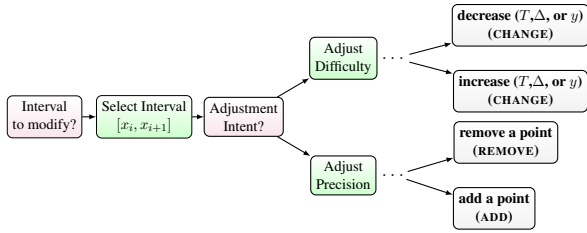


Figure 5: A snippet of the question tree; the full tree can be found at Appendix C.

shown in Figure 5. In total, there are 5 levels with 7 (alternative) questions, which can be queried iteratively over rounds (a round runs from the root to one of the leaves). Each leaf denotes an operation related to the precision or difficulty, i.e., ADD (values are set by averaging adjacent points as before), REMOVE, or CHANGE (T , Δ , and/or satisfaction by a small step). The chosen operation is applied to the quantification at the end of a round.

Note that if we allow infinite interaction rounds, then certainly IRAP would always reach the most preferred quantification, but it is too costly. As such, we set a hyperparameter N which bounds the maximum number of rounds in the interaction.

5 Evaluation

5.1 Experiment Setup

Datasets and Procedure: As in Table 2, we use four carefully curated datasets, extracting the performance requirements documented in real-world projects, as the testing sets. Since real-world samples are limited, we leveraged GPT-4 to generate synthetic data based on examples from 34 software engineering domains (Ferrara et al., 2024), leading to 2,560 (labeled) performance requirements as the initial quantification examples for the fine-tunings and retrieval knowledge in IRAP. For each dataset, the ground truth—the most preferred quantification of the requirements—is manually agreed and annotated by the authors and collaborators who are experienced software engineers. This is important for us to conduct fair comparisons and they still represent real-world preferences.

The annotation follows two steps:

- Determine the number of “break points” that break the quantification into two extra fragments. This is basically a classification task, and we computed Cohens κ (Landis and Koch, 1977) as an indicator of inter-annotator agreement rates, then iterated the process until there is a $\kappa > 0.7$ (a pragmatic threshold of

Dataset	# Projects	# Perf. Req.	Source
PROMISE (pro, 2005)	15	45	real-world
PURE (Ferrari et al., 2017)	79	23	real-world
SRS (Shaukat et al., 2018)	4	15	real-world
FQ (Alhoshan et al., 2025)	5	10	real-world
LLM-generated dataset	N/A	2560	synthetic

Table 2: Details of the datasets studied.

reliable common agreement in prior works).

- Determine the value of each break point. For each of those points, we then voted for eliminating one value in turn, until only one final value is left, which would be chosen.

For interaction, we ask several experienced human software engineers (five per dataset) to use IRAP with the annotated ground truth in mind. There are 5 repeats for all methods, even if no humans are involved.

State-of-the-art Methods: We compare IRAP against four categories of 10 state-of-the-art methods: (1) domain-specific method: the rule- and pattern-based LQPR (Wang and Chen, 2026); (2) the vanilla LLMs, i.e., DeepSeek-V3, Qwen3-Coder, gpt-5-mini, and llama-4; (3) RAG-based methods: Naive RAG (Lewis et al., 2020), Hybrid RAG (Bruch et al., 2024), and the noise-resilient ASTUTE RAG (Wang et al., 2025), for all of which Qwen3-Coder (480B) is used as the backbone LLM; (4) the preference-optimized methods with RL: DPO (Rafailov et al., 2023) and WPO (Zhou et al., 2024), which are paired with a fine-tuned Qwen-7B via LoRA (Hu et al., 2022). As with IRAP, the applicable methods use the same synthetic dataset for fine-tuning and knowledge retrieval. The prompt details are at Appendices D.3–D.5. For methods that support direct human interactions on-the-fly (with any questions), e.g., the vanilla LLMs, we ask the same human software engineers and set the same maximum interaction rounds as IRAP ($N = 5$).

Evaluation Metrics: We use several metrics to assess the difference between the produced and the ground truth quantification for each test requirement: we use Point-to-Point Distance (P2P) to assesses the structure similarity (Kuhn, 2010) and Maximum Deviation (Chebyshev) that quantifies the worst-case vertical discrepancy (Powell, 1981) as two *geometric* metrics; we also use two *scalar* metrics Root Mean Square Error (RMSE) that measures the average deviation of y values (Chai and Draxler, 2014); and Integrated Area Difference (IAD) that evaluates the difference of require-

Type	Method	P2P	Chebyshev	RMSE	IAD	P2P	Chebyshev	RMSE	IAD
		PROMISE Dataset				PURE Dataset			
Domain-specific	LQPR (Wang and Chen, 2026)	0.950 (0.340)	0.460 (0.280)	0.300 (0.210)	0.240 (0.190)	0.870 (0.380)	0.570 (0.320)	0.320 (0.240)	0.260 (0.210)
Vanilla LLMs	DeepSeek-V3 (671B)	0.774 (0.451)	0.491 (0.331)	0.274 (0.231)	0.201 (0.191)	0.651 (0.451)	0.481 (0.391)	0.251 (0.231)	0.201 (0.211)
	Qwen3-Coder(480B)	0.851 (0.584)	0.481 (0.371)	0.224 (0.191)	0.141 (0.145)	0.651 (0.491)	0.431 (0.341)	0.231 (0.201)	0.191 (0.181)
	gpt-5-mini (300B)	0.751 (0.471)	0.474 (0.281)	0.264 (0.191)	0.191 (0.154)	0.791 (0.311)	0.391 (0.371)	0.201 (0.211)	0.161 (0.191)
RAG-based (Qwen3-Coder 480B)	llama-4 (300B)	0.864 (0.381)	0.341 (0.391)	0.191 (0.231)	0.141 (0.181)	0.791 (0.311)	0.391 (0.371)	0.201 (0.211)	0.161 (0.191)
	RAG (Lewis et al., 2020)	0.549 (0.536)	0.310 (0.394)	0.178 (0.247)	0.140 (0.202)	0.326 (0.559)	0.217 (0.355)	0.126 (0.208)	0.103 (0.178)
	Hybrid RAG (Bruch et al., 2024)	0.699 (0.542)	0.349 (0.404)	0.178 (0.228)	0.125 (0.175)	0.949 (0.511)	0.700 (0.379)	0.418 (0.278)	0.300 (0.268)
Preference-optimized with RL (Qwen-7B)	ASTUTE RAG (Wang et al., 2025)	0.760 (0.569)	0.451 (0.339)	0.227 (0.182)	0.147 (0.140)	0.617 (0.369)	0.463 (0.267)	0.251 (0.169)	0.185 (0.162)
	DPO (Rafailov et al., 2023)	1.252 (0.866)	0.774 (0.328)	0.480 (0.277)	0.338 (0.287)	1.356 (1.292)	0.870 (0.221)	0.546 (0.208)	0.374 (0.265)
IRAP (Ours)	WPO (Zhou et al., 2024)	1.470 (0.753)	0.931 (0.205)	0.439 (0.318)	0.323 (0.277)	1.672 (0.918)	0.992 (0.039)	0.605 (0.170)	0.428 (0.164)
	IRAP (Ours)	0.239 (0.385)	0.029 (0.105)	0.014 (0.052)	0.010 (0.042)	0.041 (0.054)	0.136 (0.218)	0.055 (0.085)	0.034 (0.050)
		SRS Dataset				FQ Dataset			
Domain-specific	LQPR (Wang and Chen, 2026)	0.740 (0.310)	0.400 (0.260)	0.160 (0.180)	0.210 (0.150)	0.880 (0.350)	0.760 (0.290)	0.510 (0.220)	0.340 (0.190)
Vanilla LLMs	DeepSeek-V3 (671B)	0.621 (0.381)	0.401 (0.331)	0.221 (0.211)	0.161 (0.171)	0.854 (0.461)	0.642 (0.342)	0.322 (0.185)	0.241 (0.192)
	Qwen3-Coder (480B)	0.861 (0.291)	0.551 (0.271)	0.291 (0.221)	0.224 (0.201)	1.084 (0.742)	0.741 (0.371)	0.451 (0.271)	0.371 (0.268)
	gpt-5-mini (300B)	0.761 (0.291)	0.531 (0.271)	0.241 (0.141)	0.161 (0.125)	0.932 (0.401)	0.651 (0.351)	0.374 (0.242)	0.301 (0.265)
RAG-based (Qwen3-Coder 480B)	llama-4 (400B)	0.691 (0.231)	0.281 (0.311)	0.131 (0.161)	0.098 (0.125)	0.801 (0.374)	0.684 (0.351)	0.391 (0.201)	0.322 (0.171)
	RAG (Lewis et al., 2020)	0.415 (0.600)	0.218 (0.355)	0.114 (0.187)	0.081 (0.140)	0.286 (0.410)	0.026 (0.132)	0.014 (0.070)	0.013 (0.063)
	Hybrid RAG (Bruch et al., 2024)	1.092 (0.580)	0.760 (0.336)	0.401 (0.219)	0.275 (0.191)	0.332 (0.470)	0.093 (0.260)	0.059 (0.168)	0.046 (0.143)
Preference-optimized with RL (Qwen-7B)	ASTUTE RAG (Wang et al., 2025)	0.689 (0.569)	0.395 (0.278)	0.202 (0.167)	0.138 (0.148)	0.747 (0.677)	0.516 (0.417)	0.276 (0.242)	0.188 (0.212)
	DPO (Rafailov et al., 2023)	1.342 (1.353)	0.878 (0.257)	0.542 (0.237)	0.375 (0.284)	1.373 (0.821)	0.831 (0.271)	0.526 (0.256)	0.418 (0.308)
IRAP (Ours)	WPO (Zhou et al., 2024)	1.903 (1.280)	0.981 (0.052)	0.645 (0.124)	0.539 (0.148)	0.956 (0.408)	0.973 (0.044)	0.532 (0.213)	0.355 (0.142)
	IRAP (Ours)	0.100 (0.221)	0.130 (0.238)	0.051 (0.094)	0.030 (0.056)	0.270 (0.412)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)

Table 3: Mean (deviation) performance over all repeats and requirements. **Bold** highlights the best result per case.

ment difficulty by area (Alt and Godau, 1995). For all metrics, the smaller the value, the better. More details can be found at Appendix D.1.

5.2 Main Results

Table 3 presents the results over all repeats and requirements. As can be seen, in general, IRAP performs remarkably better than the others, ranking the first for all datasets/metrics with up to 40× improvements. Rule-based methods like LQPR can only quantify performance requirements as patterns and lack preference reasoning capabilities. Directly instructing LLMs (e.g., DeepSeek-V3) performs poorly too, since it remains challenging to ensure that the LLM can fully understand the complexity of requirement quantification even with interactions. Preference-optimized methods with RL, such as DPO are also not performing well, since they fail to capture the imprecise and vague preference implied in a performance requirement. The RAG-based methods perform the best among the state-of-the-art methods, since the retrieval is important for quantification. However, they remain inferior to IRAP because they do not explicitly handle the problem-specific patterns/knowledge as IRAP. A qualitative case study can be found at Appendix E.

5.3 Ablation Study

We respectively remove the retrieval-analogical preference reasoning (w/o-A), interactive preference tuning (w/o-T), and both (w/o-AT) from IRAP for comparisons. As we can see from Fig-

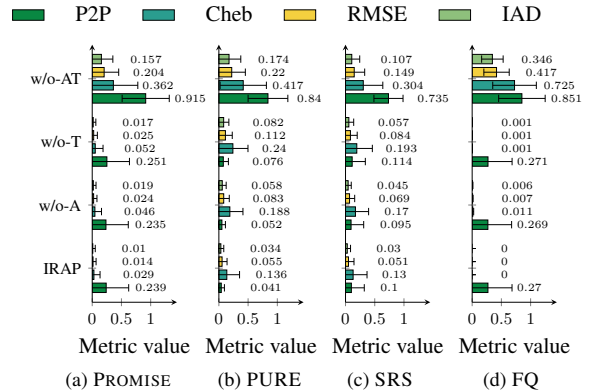


Figure 6: Mean (deviation) performance of ablation over all repeats and requirements.

ure 6, IRAP indeed generally performs the best compared with the others. Notably, w/o-AT performs much worse than w/o-A and w/o-T, meaning that while both retrieval-analogical preference reasoning and interactive preference tuning make non-trivial contributions to IRAP individually, their combination is the key to its success.

5.4 Cognitive Overhead

To examine the effectiveness of IRAP in reducing cognitive load, we quantify the cognitive load as the number of human interactions required to achieve a goal. To this end, we measure the cognitive overhead saved by IRAP relative to a compared method that permits human interaction. Suppose that we compare IRAP with another method B :

- For each requirement, we find the smallest interaction count $N_{b,i}$ that B reaches its

Dataset	DeepSeek-V3	Qwen3-Coder	gpt-5-mini	llama-4
P2P				
PROMISE	0.4044	0.3378	0.4133	0.2889
PURE	0.2261	0.2522	0.2522	0.2087
SRS	0.2667	0.2267	0.2133	0.2800
FQ	0.3600	0.2000	0.2400	0.4200
Chebyshev				
PROMISE	0.2667	0.2578	0.2667	0.3067
PURE	0.5826	0.5478	0.4783	0.5565
SRS	0.3733	0.2267	0.2133	0.5600
FQ	0.2200	0.2400	0.2000	0.2000
RMSE				
PROMISE	0.2089	0.2489	0.2178	0.2711
PURE	0.2522	0.2609	0.2609	0.2696
SRS	0.2400	0.2400	0.2267	0.5867
FQ	0.2800	0.2200	0.2400	0.2200
IAD				
PROMISE	0.2178	0.2489	0.2178	0.2800
PURE	0.2522	0.2783	0.2783	0.2696
SRS	0.2400	0.2400	0.2533	0.3867
FQ	0.2800	0.2200	0.2200	0.2400

Table 4: Cognitive overhead saving achieved by IRAP.

best metric value b averaged over 5 repeats, then averaging the count over k requirements:

$$N_{b\text{-avg}} = \frac{1}{k} \sum_{i=1}^k N_{b,i}.$$

- We then find the smallest interaction count $N_{a,i}$ for IRAP to reach the same b averaged over 5 repeats on the same requirement, compute the average count over k requirement: $N_{a\text{-avg}} = \frac{1}{k} \sum_{i=1}^k N_{a,i}$.
- Quantify the relative cognitive overhead of IRAP via the ratio $\frac{N_{a\text{-avg}}}{N_{b\text{-avg}}}$.

The smaller the ratio, the more savings can be achieved by IRAP, e.g., for k requirements, if DeepSeek-V3 reaches their best average metric b at average $N_{b\text{-avg}} = 5$. We find at which average smallest interaction count that IRAP can reach b , says $N_{a\text{-avg}} = 3$; we can finally have $\frac{N_{a\text{-avg}}}{N_{b\text{-avg}}} = \frac{3}{5} = 0.6$ —IRAP has only 60% overhead to that of DeepSeek-V3.

The results are in Table 4: IRAP significantly reduces the cognitive overhead in all cases with as low as 20% of the other method.

5.5 Sensitivity to the Interaction Rounds N

To understand the sensitivity of IRAP to N , we set $N \in [1, 2, \dots, 9]$ and plot the mean metric values standardized over all datasets. From Figure 7, as expected, a larger N indicates better results, but clearly, more interaction rounds could incur higher cognitive overhead for the stakeholder, which is not ideal. However, we see that the default $N = 5$ reaches a well-balanced trade-off between performance and cognitive overhead, which is safer.

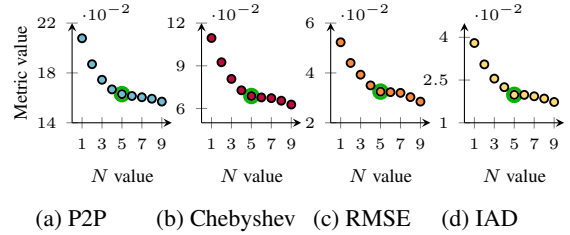


Figure 7: Sensitivity of IRAP to N over all datasets (detailed results can be found at Appendix F).

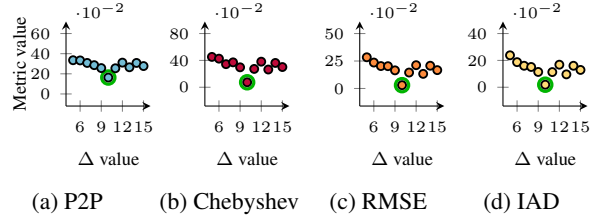


Figure 8: Sensitivity of IRAP to Δ over all datasets (detailed results can be found at Appendix G).

5.6 Sensitivity to the Changing Step Δ

By default, IRAP sets $\Delta = 10\% \times T$ to construct the quantification $f_{t,0}$. To verify the rationality of this heuristic setting, we conduct a sensitivity analysis by performing a parameter sweep for Δ across all datasets. We vary Δ from 5% to 15% of the threshold T with a step size of 1%.

As from Figure 8, clearly, 10% is a reasonable setting for Δ , leading to the best results in general.

6 Conclusion

This paper formalizes the problem of quantifying performance requirements and presents a *conjecture and inquiry* approach via interactive retrieval-augmented preference elicitation, dubbed IRAP. The key is that IRAP not only performs interactive quantification via retrieval reasoning of preference using problem-specific knowledge, but also does so with minimized stakeholders' cognitive overhead in several aspects. Evaluation against 10 state-of-the-art methods and four real-world datasets reveals that IRAP outperforms the others on both efficiency and efficacy in general.

Looking forward, the above provides a formal problem formulation and solution for quantifying performance requirements, and more importantly, it paves the way towards a research avenue of interactive preference elicitation in the field.

Limitations

Specific to performance requirement quantification: IRAP is an approach that is specifically designed for preference reasoning and tuning under software performance requirement quantification, taking many of its characteristics into account. Further, the linear slope in IRAP can be easily replaced by, e.g., a nonlinear sigmoid/error function, and hence the linearity is not a constraint nor a hard assumption. While many of its concepts are general and can be transferred to other problems, this remains subject to future work.

Quantifying initially complex requirements with multiple patterns: While the three patterns formulated in IRAP are generic enough to represent the practical performance requirements, there exist complex ones that contain multiple patterns. To quantify those, currently IRAP assumes that the initially given requirements need to be fragmented, such that each fragment serves as an independent requirement that naturally contains only one pattern; though they might become more complex following the preference reasoning and interactive preference tuning. Although this limitation implies the need for extra efforts when using IRAP, the fragmentation is often straightforward and practically only 22% real-world performance requirements contain more than one pattern/threshold (Wang and Chen, 2026). Hence we anticipate that IRAP would work fine for most of the real-world cases.

The availability of past quantification examples: The retrieval-analogical preference reasoning relies on past examples as the analogy to convert an initial draft of quantification. However, at the beginning, when there are too few accumulated past examples, the analogies to be retrieved are limited. However, this would be relieved as the quantification and reasoning proceed with more performance requirements. Further, it is possible to generate synthetic data, as we did in this work, to serve as the prior knowledge therein.

Threats to validity: Threats to construct validity might arise from the metrics used in the evaluation. To mitigate this, in this work, we use all the common metrics that measure the distance between the produced quantification and the ground truth. Also, for all experiments, the setting of parameters might affect internal validity. We have considered that by using the same/default settings for state-of-the-art methods, while exam-

ining the hyperparameter N for IRAP; we have also maintained consistency on all other setups for better fairness. Yet, admittedly, we cannot ensure that those are optimal for all cases. Finally, the subject datasets and methods compared could cause threats to external validity. Although we have examined 10 state-of-the-art methods, covering four diverse categories, and under four widely-known available real-world performance requirements datasets (as there are not much publicly available datasets for the problem), it remains difficult to guarantee that the same results would always be observed in all new cases. As such, we acknowledge that examining more datasets/methods, if publicly available, might prove more fruitful.

Ethical Considerations

All human-involved processes in this paper, i.e., the manual annotation of the most preferred quantification of the requirements in the datasets and the interactive adjustments in experiments, are strictly conducted in compliance with the ethical standards for academic research. The participants in this study are authors and collaborators of this paper/work, who are experienced software engineers, and their involvement is carried out in full accordance with legal and regulatory policies, limiting to content related to the performance requirements only. Notably, the participants legitimate right—includes the right to withdraw from the study at any time—has been fully protected.

The real-world performance requirement dataset used in this study, as well as the synthetic data generated via GPT-4, do not involve any personal privacy information, commercially sensitive data, or confidential information targeting specific users. The requirement content mainly focuses on general performance metrics in software engineering (e.g., response time, throughput, and resource utilization). Furthermore, all open-source LLMs employed in this study (e.g., Qwen and GPT-2) are used in strict compliance with their respective open-source licenses and usage policies.

We hereby commit that all steps of data collection/labeling, data generation, and human-machine interaction in this study are fully aligned with scientific research ethics.

Acknowledgment

This work was supported by a NSFC Grant (62372084).

References

2005. The promise repository of software engineering databases. <https://openscience.us/repo/requirements/requirements-other/nfr.html>. Retrieved on Jan 01, 2025.
- Waad Alhoshan, Alessio Ferrari, and Liping Zhao. 2025. How effective are generative large language models in performing requirements classification? *CoRR*, abs/2504.16768.
- Helmut Alt and Michael Godau. 1995. Computing the fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91.
- Luciano Baresi, Liliana Pasquale, and Paola Spoletini. 2010. Fuzzy goals for requirements-driven adaptation. In *RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010*, pages 125–134. IEEE Computer Society.
- Manal Binkhonain and Reem Alfayaz. 2025. Are prompts all you need? evaluating prompt-based large language models (llm)s for software requirements classification. *CoRR*, abs/2509.13868.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.
- Sebastian Bruch, Siyu Gai, and Amir Ingber. 2024. An analysis of fusion functions for hybrid retrieval. *ACM Trans. Inf. Syst.*, 42(1):20:1–20:35.
- Tianfeng Chai and Roland R Draxler. 2014. Root mean square error (rmse) or mean absolute error (mae)?—arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250.
- Pengzhou Chen and Tao Chen. 2026. Promisetune: Unveiling causally promising and explainable configuration tuning. In *48th IEEE/ACM International Conference on Software Engineering (ICSE)*. ACM.
- Pengzhou Chen, Tao Chen, and Miqing Li. 2024. MMO: meta multi-objectivization for software configuration tuning. *IEEE Trans. Software Eng.*, 50(6):1478–1504.
- Tao Chen, Ke Li, Rami Bahsoon, and Xin Yao. 2018. FEMOSAA: feature-guided and knee-driven multi-objective optimization for self-adaptive software. *ACM Trans. Softw. Eng. Methodol.*, 27(2):5:1–5:50.
- Tao Chen and Miqing Li. 2023. Do performance aspirations matter for guiding software configuration tuning? an empirical investigation under dual performance objectives. *ACM Trans. Softw. Eng. Methodol.*, 32(3):68:1–68:41.
- Daixuan Cheng, Shaohan Huang, and Furu Wei. 2024. Adapting large language models via reading comprehension. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307.
- Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. 2000. *Non-Functional Requirements in Software Engineering*, volume 5 of *International Series in Software Engineering*. Springer.
- Chengwen Du and Tao Chen. 2025. Causally perturbed fairness testing. *ACM Transactions on Software Engineering and Methodology*.
- Jonas Eckhardt, Andreas Vogelsang, Henning Femmer, and Philipp Mager. 2016a. Challenging incompleteness of performance requirements by sentence patterns. In *24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, September 12-16, 2016*, pages 46–55. IEEE Computer Society.
- Jonas Eckhardt, Andreas Vogelsang, and Daniel Méndez Fernández. 2016b. Are "non-functional" requirements really non-functional?: an investigation of non-functional requirements in practice. In *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pages 832–842. ACM.
- Carmine Ferrara, Francesco Casillo, Carmine Gravino, Andrea De Lucia, and Fabio Palomba. 2024. Re-fair: Toward a context-aware recommender for fairness requirements engineering. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*, pages 213:1–213:12. ACM.
- Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2017. PURE: A dataset of public requirements documents. In *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, pages 502–505. IEEE Computer Society.
- Martin Glinz. 2007. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference, RE 2007, October 15-19th, 2007, New Delhi, India*, pages 21–26. IEEE Computer Society.

- Jingzhi Gong, Tao Chen, and Rami Bahsoon. 2025. [Dividable configuration performance learning](#). *IEEE Trans. Software Eng.*, 51(1):106–134.
- Tobias Hey, Jan Keim, Anne Koziolok, and Walter F. Tichy. 2020. [Norbert: Transfer learning for requirements classification](#). In *28th IEEE International Requirements Engineering Conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020*, pages 169–179. IEEE.
- Kristen Howell, Gwen Christian, Pavel Fomitchov, Gitit Kehat, Julianne Marzulla, Leanne Rolston, Jadin Tredup, Ilana Zimmerman, Ethan Selfridge, and Joseph Bradley. 2023. [The economic trade-offs of large language models: A case study](#). *CoRR*, abs/2306.07402.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Harold W. Kuhn. 2010. [The hungarian method for the assignment problem](#). In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 29–47. Springer.
- J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Xianchang Luo, Yinxing Xue, Zhenchang Xing, and Jiamou Sun. 2022. [PRCBERT: prompt learning for requirement classification using bert-based pre-trained language models](#). In *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*, pages 75:1–75:13. ACM.
- Youpeng Ma, Tao Chen, and Ke Li. 2025. [Faster configuration performance bug testing with neural dual-level prioritization](#). In *47th IEEE/ACM International Conference on Software Engineering, ICSE 2025, Ottawa, ON, Canada, April 26 - May 6, 2025*, pages 988–1000. IEEE.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Murat Polat. 2020. [Analysis of multiple-choice versus open-ended questions in language tests according to different cognitive domain levels](#). *Novitas-ROYAL (Research on Youth and Language)*, 14(2):76–96.
- Michael James David Powell. 1981. *Approximation theory and methods*. Cambridge university press.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. [Direct preference optimization: Your language model is secretly a reward model](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Mohammed Sayagh, Noureddine Kerzazi, Bram Adams, and Fábio Petrillo. 2020. [Software configuration engineering in practice interviews, survey, and systematic literature review](#). *IEEE Trans. Software Eng.*, 46(6):646–673.
- Zain Shaukat Shaukat, Rashid Naseem, and Muhammad Zubair. 2018. [A dataset for software requirements risk prediction](#). In *2018 IEEE International Conference on Computational Science and Engineering, CSE 2018, Bucharest, Romania, October 29-31, 2018*, pages 112–118. IEEE Computer Society.
- U.S. Department of Health and Human Services. 2016. [HealthCare.gov: Case study of CMS management of the federal marketplace](#). Office of Inspector General Report. Accessed online.
- Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. [Representation learning with contrastive predictive coding](#). *CoRR*, abs/1807.03748.
- Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö. Arik. 2025. [Astute RAG: overcoming imperfect retrieval augmentation and knowledge conflicts for large language models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 30553–30571. Association for Computational Linguistics.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. [Joint embedding of words and labels for text classification](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2321–2331. Association for Computational Linguistics.

Shihai Wang and Tao Chen. 2026. [Light over heavy: Automated performance requirements quantification with linguistic inducement](#). In *Proceedings of the 48th IEEE/ACM International Conference on Software Engineering (ICSE)*. ICSE 2026.

Jon Whittle, Peter Sawyer, Nelly Bencomo, Betty H. C. Cheng, and Jean-Michel Bruel. 2010. [RELAX: a language to address uncertainty in self-adaptive systems requirement](#). *Requir. Eng.*, 15(2):177–196.

Zezen Xiang, Jingzhi Gong, and Tao Chen. 2026. [Dually hierarchical drift adaptation for online configuration performance learning](#). In *48th IEEE/ACM International Conference on Software Engineering (ICSE)*. ACM.

Gangda Xiong and Tao Chen. 2025. [Cotune: Co-evolutionary configuration tuning](#). In *Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. ASE 2025.

Yulong Ye, Tao Chen, and Miqing Li. 2025. [Distilled lifelong self-adaptation for configurable systems](#). In *47th IEEE/ACM International Conference on Software Engineering, ICSE 2025, Ottawa, ON, Canada, April 26 - May 6, 2025*, pages 1333–1345. IEEE.

Yulong Ye, Hongyuan Liang, Chao Jiang, Miqing Li, and Tao Chen. 2026. [Revealing domain-spatiality patterns for configuration tuning: Domain knowledge meets fitness landscapes](#). *ACM Transactions on Software Engineering and Methodology*.

Wenxuan Zhou, Ravi Agrawal, Shujian Zhang, Sathish Reddy Indurthi, Sanqiang Zhao, Kaiqiang Song, Silei Xu, and Chenguang Zhu. 2024. [WPO: enhancing RLHF with weighted preference optimization](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 8328–8340. Association for Computational Linguistics.

Appendix

A Additional Details for Retrieval-Generative Quantification

A.1 Full Anchor List in Retrieval-based Classification

Here, the anchor phrases serve as important domain-specific knowledge for incorporating label semantics in the retrieval-based classification.

To that end, we extract and prepare the anchors following the steps below:

1. Select highly representative and relevant phrases from the keyword sets of performance requirements summarized in prior work (Wang and Chen, 2026).
2. Drawing on those, supplement more synonymous phrases via the English lexical database.
3. Carefully assign those phrases into the correct pattern types through agreements among all authors. To balance the effort and representativeness, we cap 10 phrases for each pattern.

Table 5 lists all the 30 anchor phrases corresponding to the three defined patterns. Note that when fine-tuning the RoBERTa, all requirements and anchors are used as the training samples.

A.2 Sample/Prompt in Generative Threshold Extraction

To extract the relevant threshold T in a requirement, IRAP fine-tunes the lightweight GPT-2 with only 774M parameters. Each sample consists of an instruction-based prompt and the expected ground truth of the threshold, e.g.:

Fine tuning sample

Input: Please extract the numeric threshold from the following performance requirements: “The response time must not exceed 200ms.”
Output: 200

Upon extraction, we use the same prompt to obtain the threshold of the target requirement, e.g.:

Prompt

Please extract the numeric threshold from the following performance requirements.
Requirement: “The system should support at least 500 concurrent stakeholders for 7 days, 24 hours.”

B Additional Details for Retrieval-Analogical Preference Reasoning

The pseudo code for the retrieval-analogical preference reasoning has been shown in Algorithm 1, where the key steps discussed are highlighted. The

Pattern Type	Anchor Phrase
P_1	“no less than”, “at least”, “greater than”, “minimum of”, “not below”, “above”, “exceeding”, “no fewer than”, “greater than or equal to”, “at minimum”
P_2	“no more than”, “at most”, “less than”, “maximum of”, “not exceeding”, “under”, “below”, “up to”, “at maximum”, “with in”
P_3	“exactly”, “equal to”, “precisely”, “specifically”, “fixed at”, “set to”, “equivalent to”, “identical to”, “precisely at”, “designated as”

Table 5: The anchor phrases for all pattern types.

Algorithm 1 RETRIEVAL_ANALOGICAL_PREFERENCE_REASONING

```

1: Input: Performance requirement  $s_t$ , initial draft quantification for the target  $f_{t,0}$  and set of past quantification examples  $\mathcal{S} : \{s_i = \{f_{i,0}, f_i^*\} | i \in [1, t-1]\}$ 
2: Output: Converted/reasoned quantification  $f'_{t,0}$ 
3:  $s_k = \{f_{k,0}, f_k^*\} \leftarrow \arg \max_{s_i \in \mathcal{S}, |f_{i,0}|=|f_{t,0}|} \text{SEMANTIC\_SIM}(s_t, s_i)$ 
4:  $\mathcal{O} = \emptyset$ 
    $\triangleright$  Points alignment
5:  $\mathcal{M} \leftarrow \text{MATCHING\_BY\_KM}(f_{k,0}, f_k^*)$ 
6: if points in  $f_{k,0} >$  points in  $f_k^*$  then
7:    $\mathcal{O} \leftarrow \mathcal{O} \cup \{\text{REMOVE}(u) | u \in f_{k,0}; u \notin \mathcal{M}\}$ 
8: else if points in  $f_{k,0} <$  points in  $f_k^*$  then
9:    $\mathcal{O} \leftarrow \mathcal{O} \cup \{\text{ADD}(u) | u \in f_k^*; u \notin \mathcal{M}\}$ 
10: end if
    $\triangleright$  Changes identification
11:  $f'_{k,0} \leftarrow \text{apply } \mathcal{O} \text{ to } f_{k,0}$ 
12:  $\mathcal{M}' \leftarrow \text{MATCHING\_BY\_KM}(f'_{k,0}, f_k^*)$ 
13: for each matched pair  $(u_i, v_j) \in \mathcal{M}'$  do
14:   if  $u_i \neq v_j$  then
15:      $\mathcal{O} \leftarrow \mathcal{O} \cup \{\text{CHANGE}(u_i, v_j)\}$ 
16:   end if
17: end for
    $\triangleright$  Operations sequencing
18: Place the CHANGE in  $\mathcal{O}$  as after ADD and REMOVE
19:  $f'_{t,0} \leftarrow \text{apply } \mathcal{O} \text{ to } f_{t,0}$ 
20: return  $f'_{t,0}$ 

```

Algorithm 2 MATCHING_BY_KM

```

1: Input: Initial points  $f_{k,0} : \mathcal{U} = \{u_1, \dots, u_n\}$ ; final points  $f_k^* : \mathcal{V} = \{v_1, \dots, v_m\}$ 
2: Output: Optimal matching  $\mathcal{M}$ 
3:  $\mathcal{W} \leftarrow$  a  $n \times m$  matrix
4: for  $i \leftarrow 1$  to  $n$  do
5:   for  $j \leftarrow 1$  to  $m$  do
6:      $\mathcal{W} \leftarrow w_{ij} = -\sqrt{(x_{u_i} - x_{v_j})^2 + (y_{u_i} - y_{v_j})^2}$ 
7:   end for
8: end for
9:  $\mathcal{M} \leftarrow \text{KMAlgorithm}(\mathcal{W})$   $\triangleright$  Solving max weight matching
10: return  $\mathcal{M}$ 

```

optimal matching via the KM algorithm has been illustrated as the pseudo code in Algorithm 2.

C Full Question Tree of Interactive Preference Tuning

The complete tree-based multiple-choice questions used in interactive preference tuning for question-answering are shown in Figure 9. In particular, the designs follow three levels of hierarchy to progressively prompt the stakeholder for intuitive and iterative feedback.

Note that, in IRAP, completing one round means answering all questions from the root to one of the leaves, leading to one operation; the number of rounds is bounded by the hyperparameter $N = 5$. In all experiments, we have rarely observed that the quantification matches the ground truth with fewer than 5 interaction rounds.

D Other Experiment Setting Details

D.1 Additional Details of Evaluation Metrics

The specific formulas for calculating the evaluation metrics are discussed below:

- **Point-to-Point Distance (P2P):** This metric assesses the structural and positional similarity of the functions’ inflection point sets. We normalize the domain to $[0, 1]$. Optimal matching is achieved via the KM algorithm (Kuhn, 2010), accumulating the Euclidean distances of matched pairs:

$$\text{P2P} = \sum_{(u_i, v_j) \in \mathcal{M}} w_{ij}$$

- **Maximum Deviation (Chebyshev):** This metric quantifies the worst-case vertical discrepancy between the two quantification. It is calculated as the maximum absolute difference between function values across the domain, determined by searching all inflection

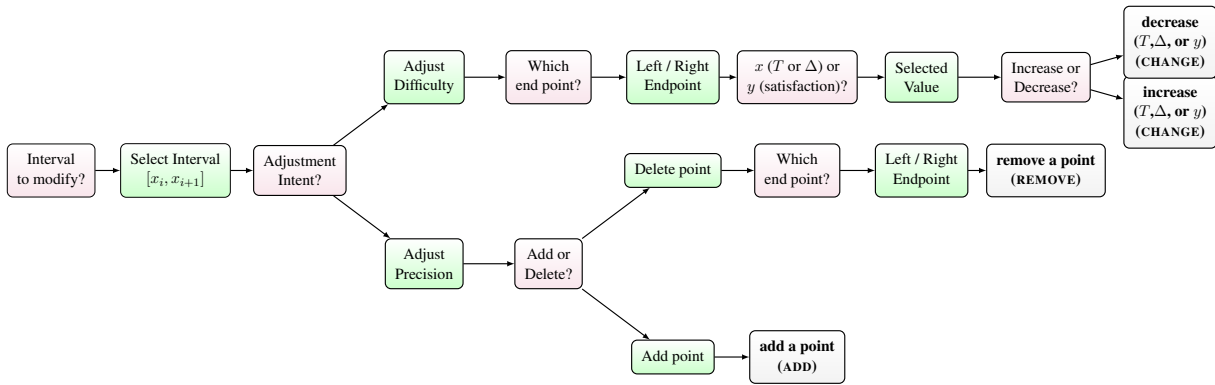


Figure 9: The complete question tree in IRAP for interaction.

Prompt used for vanilla LLMs

For a performance requirement statement, we can quantify it. Specifically, we expect to derive a function expression that takes a performance metric as input and outputs the stakeholder satisfaction level corresponding to that metric. We may simply assume that the function expression is a piecewise linear function. For example, given the requirement statement:

> In the scenario of real-time electrocardiogram (ECG) monitoring, the software shall receive and process ECG signal data at a sampling frequency no lower than 1000 Hz.

The corresponding piecewise linear quantification function is:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 900 \\ 1/100(x - 900) & \text{if } 900 < x < 1000 \\ 1 & \text{if } x \geq 1000 \end{cases}$$

A piecewise linear function has multiple "inflection points". The set of inflection points for the above piecewise linear function is:

$$\{(900, 0), (1000, 1)\}$$

However, this form of quantification is imprecise. You need to ask questions to the requirement setter to adjust the quantification form. The quantification form shall be uniformly represented in the form of a point list. You may ask the requirement setter about the aspects of the current quantification form that they are dissatisfied with, and then adjust the quantification form based on their responses. Your inquiries are subject to constraints: you are not allowed to directly ask for the exact ideal quantification form they have in mind. Instead, you can only ask ambiguous questions, such as whether the coordinate value of a certain point is too large or too small (you cannot directly ask for the specific coordinate value), and whether the number of segments is too many or too few (you cannot directly ask for the exact number of segments). Within these constraints, you may ask any questions you want. Your goal is to guess the quantification form that satisfies the stakeholder. Note that you are only allowed to ask 5 questions in total, with one question per round. After I have answered your 5th question, you need to output the final confirmed quantification form. The performance requirement quantification task you need to handle is as follows:

Table 6: Prompt used for vanilla LLMs.

points, endpoints, and sampled intervals:

$$\text{Chebyshev} = \max_x |f_1(x) - f_2(x)|$$

ences:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (f_1(x_i) - f_2(x_i))^2}$$

- **Root Mean Square Error (RMSE) for y :** This metric measures the robust average functional deviation along the curve. After normalizing the domain to $[0, 1]$, we select N uniformly sampled points x_i and compute the root mean square of the function value differ-

- **Integrated Area Difference (IAD):** This metric evaluates the overall difference in the implied requirement difficulty, as the integral area represents this dimension. It is the absolute difference between the areas enclosed by two functions and the x-axis, within the

normalized common domain $[0, 1]$:

$$\text{IAD} = \left| \int f_1(x) dx - \int f_2(x) dx \right|$$

D.2 Hyperparameters

For all state-of-the-art methods, their hyperparameters are set as the default; for IRAP, the hyperparameters are also used the default, e.g., $\tau = 0.07$ for fine-tuning the GPT-2. The number of interactions N is set to 5 unless otherwise stated.

D.3 Prompt used for vanilla LLMs

The prompt used for vanilla LLMs is shown in Table 6.

D.4 Prompt used for RAG-based methods

The prompt used for RAG-based methods is shown in Table 7.

D.5 Prompt used for preference-optimized methods with RL

The prompt used for RAG-based methods is shown in Table 8. Note that the prompts are optimized via reinforcement learning to make the quantitative forms generated by LLMs more aligned with the preferences of stakeholders.

E Additional Qualitative Case Study

Commonly effective cases for IRAP: A typical kind of performance requirements that IRAP handles well is: “*The system requests per second (req/s) shall support at least 200.*” The stakeholder’s most preferred quantification is $f_t^* : \{(180, 0), (195, 0.6), (198, 0.8), (200, 1)\}$. IRAP initially generates $f_{t,0} : \{(180, 0), (200, 1)\}$. Subsequently, the retrieval-analogical preference reasoning phase retrieves a relevant historical example s_k : “*The number of concurrent users shall reach 100,*” which followed the trajectory $f_{k,0} : \{(90, 0), (100, 1)\} \rightarrow f_k^* : \{(98, 0), (100, 1)\}$, as the most appropriate analogy, i.e., stricter quantification (higher difficulty) is more likely to be preferred by the stakeholder. As such, IRAP then accordingly convert $f_{t,0}$ into the reasoned state $f'_{t,0} : \{(195, 0), (200, 1)\}$. In the interaction, IRAP guides the stakeholder through the question tree illustrated in Figure 9:

- **Round 1 (Adjusting Difficulty):** The stakeholder finds the reasoning result too strict and selects the interval $[196, 200]$. The path

is: LEFT ENDPOINT \rightarrow SELECT X \rightarrow DECREASE. Applying the step size of 10%, the x value decreases from 196 to 175.5. Thus, the current state is: $\{(175.5, 0), (200, 1)\}$.

- **Round 2 (Adjusting Difficulty):** Finding 175.5 slightly too loose/relaxed, the stakeholder reverses the direction to INCREASE x . Due to the direction reversal on the same point, the step size attenuates to 5%. The x value increases to $175.5 \times (1 + 5\%) = 184.275$, leading to the current state: $\{(184, 0), (200, 1)\}$.
- **Round 3 (Adjusting Precision):** The stakeholder adds a transitional satisfaction level to increase the precision of the curve. The path is: INTERVAL TO MODIFY \rightarrow ADJUST PRECISION \rightarrow ADD POINT. IRAP inserts a new point at the mean of $[184, 200]$, resulting in $(192, 0.5)$. Thus, the current state is: $\{(184, 0), (192, 0.5), (200, 1)\}$.
- **Round 4 (Adjusting Precision):** The stakeholder decides to further increase the precision by adding a point between the intermediate point and the endpoint. The path is: INTERVAL TO MODIFY \rightarrow ADJUST PRECISION \rightarrow ADD POINT. IRAP inserts a new point at the mean of $[192.138, 200]$, resulting in $(196, 0.75)$. The current state becomes: $\{(184, 0), (192, 0.5), (196, 0.75), (200, 1)\}$.
- **Round 5 (Adjusting Difficulty):** The stakeholder further refines the satisfaction level y for the intermediate point. The path is: ADJUST DIFFICULTY \rightarrow SELECT Y \rightarrow INCREASE. The y value increases by 10% to $0.5 \times (1 + 10\%) = 0.55$. Thus, the current state is: $\{(18, 0), (192, 0.55), (196, 0.75), (200, 1)\}$.

Finally, IRAP reaches a result relatively close to f_t^* through five rounds of tree path question-answering based on the question tree.

In contrast, the domain-specific method (LQPR) enables automated mapping from natural language to quantitative forms; however, it essentially relies on static rule matching and completely neglects the dynamic impacts of stakeholders’ subjective preferences on the tolerance margin Δ and satisfaction curves: it at most output a quantification similar to $f_{t,0} : \{(180, 0), (200, 1)\}$.

Although the vanilla LLMs support interactive capabilities, without the guidance of the quantitative theoretical framework proposed in this

Prompt used for RAG-based methods

Accurately converting performance requirements described in natural language into computable quantitative indicators is a key challenge in the field of software engineering, shifting from qualitative evaluation to quantitative analysis. Our goal is to establish a stakeholder satisfaction function $f(x)$ for any performance indicator x , where the function outputs the stakeholder satisfaction (ranging from $[0, 1)$) corresponding to the value of x .

We adopt piecewise linear functions to characterize such satisfaction curves, as they offer simplicity, ease of interpretation, and sufficient expressive power. This type of piecewise linear function can be uniquely determined by its set of inflection points $\{(x_1, y_1), \dots, (x_n, y_n)\}$.

For example, consider the requirement statement:

> In real-time ECG monitoring scenarios, the software must receive and process ECG signal data at a sampling frequency of no less than 1000Hz.

With a preset tolerance range of 10%, the set of inflection points for this requirement is:

$$\{(900, 0), (1000, 1)\}$$

This set can directly restore the corresponding quantitative function $f(x)$:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 900 \\ 1/100(x - 900) & \text{if } 900 < x < 1000 \\ 1 & \text{if } x \geq 1000 \end{cases}$$

Thus, our task is formally defined as a sequence-to-sequence conversion problem:

$$\text{Performance Requirement Statement} \rightarrow [(x_1, y_1), \dots, (x_n, y_n)]$$

However, the quantitative form directly derived from performance requirement statements is not precise and does not fully align with the quantitative form expected by the requirement setters. The following examples illustrate the differences between the quantitatively derived form from the literal meaning and the expected quantitative form by the requirement setters:

[Place the retrieved historical samples here]

Please refer to the above difference samples. ****Note that you need to analyze the differences from the base form to the prefer form and apply them to your current task**** to provide a more preference-aligned quantitative form (prefer form) for the following performance requirement. The base form is provided; you need to infer the prefer form. (Output the result only in the form of a list of points, and submit just the final prefer form):

Table 7: Prompt for RAG-based methods.

paper, the active inquiries generated by LLMs tend to be divergent and fail to capture the key aspects of stakeholder preferences. For instance, the result obtained using Qwen3-coder is: $\{(199, 0), (200, 1)\}$, which oversimplifies the stakeholder's complex satisfaction decay into a binary threshold and fails to explore the tolerance margin between 180 and 200 req/s.

RAG-based methods incorporate historical contextual information, yet they overly depend on the model's inherent reasoning capabilities to interpret stakeholder preferences. Such an analysis approach is both unstable and indirect. In this case, it produces inconsistent and unstable numerical mappings $\{(180, 0), (192, 0.8), (200, 1)\}$, where the satisfaction values are biased by loosely related historical context rather than the actual fine-

grained preferences elicited through structured interaction.

Preference-optimized methods with RL struggle to accurately capture fine-grained numeric preferences when faced with sparse interactive data, and even learn incorrect preferences due to overfitting noisy data, ultimately resulting in counterproductive quantitative outcomes. For this example, DPO generates a non-monotonic quantification $\{(180, 0), (190, 0.7), (195, 0.5), (200, 1)\}$, where satisfaction counter-intuitively drops as throughput improves, clearly violating the fundamental monotonic property/meaning implied by the P_1 performance pattern.

Cases when IRAP does not work effectively:

There exists a small number of cases where IRAP is

Prompt used for preference-optimized methods with RL

Accurately converting performance requirements described in natural language into computable quantitative indicators is a key challenge in the field of software engineering, transitioning from qualitative evaluation to quantitative analysis. Our goal is to establish a stakeholder satisfaction function $f(x)$ for any performance indicator x , where the function outputs the stakeholder satisfaction (within the range $[0, 1]$) corresponding to the value of x .

We adopt a piecewise linear function to characterize this satisfaction curve, which offers simplicity, interpretability, and sufficient expressive power. Such a piecewise linear function can be uniquely determined by its set of inflection points $\{(x_1, y_1), \dots, (x_n, y_n)\}$.

For example, consider the requirement statement:

> In real-time ECG monitoring scenarios, the software must receive and process ECG signal data at a sampling frequency of no less than 1000Hz.

Given a preset tolerance range of 10

$$\{(900, 0), (1000, 1)\}$$

This set can directly restore the corresponding quantitative function $f(x)$:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 900 \\ 1/100(x - 900) & \text{if } 900 < x < 1000 \\ 1 & \text{if } x \geq 1000 \end{cases}$$

Thus, our task is formally defined as a sequence-to-sequence conversion problem:

$$\text{Performance requirement statement} \rightarrow [(x_1, y_1), \dots, (x_n, y_n)]$$

Please convert the following performance requirement into a quantitative form based on the above theory (output the result as a list of 2D points, such as $[[10.0, 1.0], [11.0, 0.0]]$). Please strictly follow the specified format for output, and do not include any additional content.):

Table 8: Prompt for preference-optimized methods with RL.

N value	P2P	Chebyshev	RMSE	IAD	P2P	Chebyshev	RMSE	IAD
	PROMISE Dataset				PURE Dataset			
1	0.315 (0.145)	0.038 (0.011)	0.018 (0.002)	0.012 (0.001)	0.108 (0.010)	0.257 (0.069)	0.133 (0.017)	0.104 (0.011)
2	0.284 (0.147)	0.034 (0.008)	0.017 (0.002)	0.013 (0.001)	0.073 (0.003)	0.180 (0.041)	0.096 (0.006)	0.073 (0.002)
3	0.264 (0.147)	0.028 (0.008)	0.016 (0.002)	0.010 (0.001)	0.059 (0.005)	0.184 (0.045)	0.097 (0.009)	0.066 (0.005)
4	0.252 (0.148)	0.028 (0.013)	0.016 (0.003)	0.011 (0.001)	0.044 (0.004)	0.117 (0.039)	0.060 (0.006)	0.052 (0.003)
5	0.238 (0.384)	0.032 (0.111)	0.014 (0.048)	0.011 (0.038)	0.040 (0.052)	0.131 (0.210)	0.053 (0.082)	0.033 (0.048)
6	0.244 (0.149)	0.022 (0.006)	0.010 (0.001)	0.010 (0.001)	0.023 (0.003)	0.126 (0.048)	0.065 (0.008)	0.041 (0.003)
7	0.243 (0.148)	0.031 (0.012)	0.016 (0.002)	0.011 (0.001)	0.025 (0.003)	0.104 (0.037)	0.047 (0.006)	0.032 (0.002)
8	0.241 (0.149)	0.024 (0.013)	0.015 (0.003)	0.010 (0.001)	0.025 (0.002)	0.117 (0.054)	0.043 (0.008)	0.028 (0.003)
9	0.237 (0.150)	0.018 (0.005)	0.009 (0.001)	0.007 (0.001)	0.022 (0.003)	0.097 (0.039)	0.042 (0.006)	0.026 (0.002)
	SRS Dataset				FQ Dataset			
1	0.138 (0.048)	0.143 (0.052)	0.058 (0.008)	0.036 (0.003)	0.270 (0.170)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
2	0.121 (0.049)	0.156 (0.055)	0.063 (0.008)	0.036 (0.003)	0.270 (0.170)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
3	0.105 (0.049)	0.111 (0.029)	0.044 (0.005)	0.026 (0.002)	0.270 (0.170)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
4	0.101 (0.049)	0.146 (0.035)	0.064 (0.005)	0.027 (0.002)	0.270 (0.170)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
5	0.094 (0.221)	0.087 (0.174)	0.034 (0.069)	0.021 (0.042)	0.270 (0.412)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
6	0.109 (0.049)	0.123 (0.047)	0.054 (0.007)	0.028 (0.002)	0.270 (0.170)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
7	0.104 (0.049)	0.134 (0.050)	0.065 (0.008)	0.034 (0.003)	0.270 (0.170)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
8	0.101 (0.049)	0.121 (0.062)	0.063 (0.010)	0.036 (0.003)	0.270 (0.170)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
9	0.099 (0.049)	0.136 (0.063)	0.063 (0.010)	0.036 (0.004)	0.270 (0.170)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)

Table 9: Detailed sensitivity analysis of IRAP with varying N value.

less effective, such as the requirement⁶: “The video stream must maintain 60 FPS, but can drop to 30 FPS in power-saving mode, and should never exceed 120 FPS to save bandwidth.” This interpretation of this requirement is logically a com-

bination of at least P_1 and P_2 patterns. Since currently IRAP works on the given performance requirement with one pattern only each time, the above, although can be fragmented, might alter the original semantics such that IRAP cannot detect. In the above, the entire semantics implies that any-

⁶Naturally, a higher FPS is better.

Δ value	P2P	Chebyshev	RMSE	IAD	P2P	Chebyshev	RMSE	IAD
	PROMISE Dataset				PURE Dataset			
5%	0.419 (0.515)	0.436 (0.358)	0.296 (0.309)	0.257 (0.303)	0.153 (0.220)	0.338 (0.268)	0.168 (0.134)	0.111 (0.107)
6%	0.422 (0.472)	0.441 (0.310)	0.273 (0.243)	0.232 (0.244)	0.120 (0.076)	0.280 (0.197)	0.137 (0.090)	0.086 (0.075)
7%	0.454 (0.501)	0.471 (0.292)	0.303 (0.274)	0.260 (0.280)	0.157 (0.215)	0.265 (0.200)	0.137 (0.113)	0.100 (0.105)
8%	0.422 (0.477)	0.426 (0.283)	0.271 (0.245)	0.225 (0.246)	0.122 (0.060)	0.318 (0.184)	0.155 (0.083)	0.101 (0.072)
9%	0.404 (0.463)	0.365 (0.277)	0.232 (0.222)	0.183 (0.212)	0.097 (0.047)	0.272 (0.211)	0.123 (0.085)	0.071 (0.052)
10%	0.239 (0.385)	0.029 (0.105)	0.014 (0.052)	0.010 (0.042)	0.041 (0.054)	0.136 (0.218)	0.055 (0.085)	0.034 (0.050)
11%	0.355 (0.437)	0.256 (0.277)	0.151 (0.215)	0.121 (0.209)	0.096 (0.077)	0.281 (0.206)	0.135 (0.105)	0.100 (0.092)
12%	0.388 (0.459)	0.369 (0.315)	0.210 (0.225)	0.175 (0.214)	0.108 (0.073)	0.283 (0.212)	0.121 (0.087)	0.078 (0.072)
13%	0.354 (0.402)	0.254 (0.256)	0.126 (0.148)	0.089 (0.137)	0.097 (0.075)	0.287 (0.234)	0.130 (0.096)	0.076 (0.049)
14%	0.377 (0.428)	0.315 (0.248)	0.178 (0.168)	0.132 (0.160)	0.135 (0.078)	0.332 (0.197)	0.158 (0.085)	0.106 (0.069)
15%	0.381 (0.420)	0.306 (0.267)	0.174 (0.201)	0.137 (0.197)	0.154 (0.212)	0.333 (0.243)	0.167 (0.130)	0.122 (0.103)
	SRS Dataset				FQ Dataset			
5%	0.223 (0.316)	0.435 (0.265)	0.243 (0.207)	0.193 (0.210)	0.545 (0.505)	0.597 (0.354)	0.428 (0.318)	0.392 (0.315)
6%	0.198 (0.252)	0.334 (0.252)	0.173 (0.134)	0.125 (0.118)	0.592 (0.437)	0.650 (0.306)	0.361 (0.186)	0.305 (0.171)
7%	0.237 (0.331)	0.345 (0.241)	0.167 (0.122)	0.105 (0.104)	0.385 (0.517)	0.294 (0.312)	0.210 (0.274)	0.175 (0.266)
8%	0.199 (0.259)	0.344 (0.218)	0.172 (0.116)	0.112 (0.104)	0.401 (0.447)	0.397 (0.261)	0.212 (0.142)	0.165 (0.125)
9%	0.171 (0.234)	0.288 (0.194)	0.139 (0.091)	0.083 (0.075)	0.355 (0.479)	0.261 (0.278)	0.164 (0.201)	0.119 (0.183)
10%	0.100 (0.221)	0.130 (0.238)	0.051 (0.094)	0.030 (0.056)	0.270 (0.412)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
11%	0.160 (0.211)	0.259 (0.190)	0.115 (0.094)	0.081 (0.083)	0.409 (0.392)	0.294 (0.281)	0.178 (0.183)	0.152 (0.169)
12%	0.171 (0.259)	0.340 (0.219)	0.166 (0.118)	0.104 (0.109)	0.577 (0.467)	0.533 (0.290)	0.349 (0.196)	0.315 (0.184)
13%	0.151 (0.209)	0.278 (0.245)	0.128 (0.098)	0.092 (0.069)	0.456 (0.427)	0.238 (0.266)	0.147 (0.200)	0.128 (0.184)
14%	0.240 (0.349)	0.332 (0.192)	0.190 (0.135)	0.142 (0.134)	0.480 (0.442)	0.470 (0.260)	0.302 (0.189)	0.259 (0.190)
15%	0.182 (0.203)	0.290 (0.180)	0.153 (0.091)	0.110 (0.076)	0.389 (0.443)	0.273 (0.194)	0.174 (0.163)	0.148 (0.156)

Table 10: Detailed sensitivity analysis of IRAP with varying Δ values.

thing less than 60 FPS remains highly tolerable, but if we fragment only the first part, such a meaning becomes blurred, which causes IRAP to produce an initial draft that is far away from the stakeholder’s true preferences (and hence might not be effectively quantified to be sufficiently close to the true preference even following the analogy reasoning and preference tuning).

In particular, a typical case we observed is related to the Retrieval-Analogical Preference Reasoning: when two performance requirements are semantically similar, but stakeholders preferences differ drastically. An example could be:

- **Sample 1:**

- **Performance requirement:** “*In the Online Bookstore System, the search results for book titles shall be returned to the user within 5 seconds to ensure a smooth browsing experience.*”

- **Initial quantification:** $f_{1,0} = (5.0, 1.0), (5.5, 0.0)$

- **Ideal quantification:** $f_1^* = (5.0, 1.0), (6.05, 0.0)$

- **Sample 2:**

- **Performance requirement:** “*In the Nuclear Power Plant Monitoring System, the status feedback for reactor cooling valves shall be returned to the console within 5 seconds to ensure real-time safety tracking.*”

- **Initial quantification:** $f_{2,0} =$

$(5.0, 1.0), (5.5, 0.0)$

– **Ideal quantification:** $f_2^* = (5.0, 1.0), (5.1, 0.0)$

Here, **sample 1** is chosen as the example, but using its transition makes the initial of **sample 2** even more distant from its ideal quantification.

F Full Results of the Sensitivity to Interaction Rounds N

Table 9 presents the detailed experimental results of the sensitivity analysis on the number of interactions N with respect to each dataset.

G Full Results of the Sensitivity to Δ in Retrieval-Generative Quantification

Table 10 presents the detailed experimental results of the sensitivity analysis on the value of Δ with respect to each dataset.