Accuracy and Satisfaction in Multi-Turn LLM Dialogues for NFR Assessment

Ali Pourghasemi Fatideh
University of Maine
ali.pourghasemi@maine.edu

Wilder Baldwin
University of Maine
wilder.baldwin@maine.edu

Maria Dhakal
University of Notre Dame
mdhakal@nd.edu

Collin McMillan
University of Notre Dame
cmc@nd.edu

Sepideh Ghanavati
University of Maine
sepideh.ghanavati@maine.edu

Abstract

LLM-based dialogue assistants have become mainstream tools for software developers, yet current evaluation benchmarks focus exclusively on functional correctness. This leaves a critical gap in assessing the quality and accuracy of these conversations when handling Non-Functional Requirements (NFRs), which are inherently vague, context-dependent, and involve many parts of a program. Evaluating how well these systems support collaborative reasoning about NFRs requires methods that go beyond single-turn accuracy to capture both the correctness of the system's outputs and the quality of the multi-turn interaction. In this paper, we investigate the accuracy and quality of multi-turn conversations between developers and an LLM-based agent in the domain of Health Insurance Portability and Accountability Act (HIPAA) regulatory compliance. We hired 49 programmers to interact with GitHub Copilot to assess 148 HIPAA-derived NFRs against the iTrust codebase, a system designed to comply with HIPAA regulations, across three dimensions: requirement satisfaction level, reasoning, and code localization. We find that developers tend to agree with LLM assessments, but accuracy against expert ground truth is low. We model user satisfaction and find that longer system responses and more information-providing turns negatively affect user satisfaction, whereas proactive interactions positively affect it. Our findings provide insights for designing LLM-based dialogue systems that support NFR assessment.

1 Introduction

Interactive dialogue systems have demonstrated strong capabilities across various programming tasks (Fan et al., 2023; Ozkaya, 2023; Zhang et al., 2023), including code generation, documentation, unit testing, fault and code localization, and bug fixing. These capabilities have long been an aspiration in software engineering research, as programmers are most productive through collaborative problem-solving discussions with peers, yet such peer availability is often limited (Finch and Choi, 2020; González and Mark, 2004; LaToza et al., 2006; Perlow and Weeks, 2002; Roehm et al., 2012; Sillito et al., 2008). Recent advances in Large Language Models (LLMs) have enabled these practical interactive dialogue systems for developers.

As these AI systems engage with developers through multiple interaction turns, the success of the human-AI partnership hinges less on the raw correctness of any single generated code block and more on the quality of the collaborative problem-solving process (Kumar et al., 2025; Liu et al., 2024a). This shift from evaluating individual outputs to assessing the collaborative dialogue marks an evolution in how we understand and measure AI assistance in software engineering (Kang et al., 2023; Meng et al., 2024; Nikolov et al., 2025; Rondon et al., 2025; Vaithilingam et al., 2022). Yet despite this shift, current evaluation methods and benchmarks (Jimenez et al., 2023; Jain et al., 2024; Aider, 2024) still focus primarily on single-turn functional correctness (Lin et al., 2025) and do not capture the accuracy or quality of multi-turn collaborative reasoning for Non-Functional Requirements (NFRs) (Arora et al., 2024; Gustavsson et al., 2024). NFRs, unlike functional requirements (FRs), tend to be vaguely defined concerns that span many parts of a program (Glinz et al., 2005; Mairiza et al., 2010), require domain knowledge, such as compliance with regulatory mandates, and may be difficult to test or have complex dependencies. Thus, they require evaluation techniques that could capture all the nuances regarding how well they are satisfied (as a property of the whole system) and their reasoning for such satisfactions.

To this end, we investigate the accuracy and quality of multi-turn conversations in LLM-based Agents, with a focus on the Health Insurance Portability and Accountability Act (HIPAA) (hhs) as a

Proceedings of the 27th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 91–106
August 2–5, 2026.

representative domain of regulatory NFRs. We construct a dataset of HIPAA-derived NFRs grounded in an Electronic Health Record (EHR) system and develop expert-annotated ground truth to capture the requirement satisfaction level, reasoning quality, and code location. We hire 49 programmers to interact with GitHub Copilot through multi-turn conversations and rate their agreement with the agent's assessments of NFRs. We then measure the accuracy of the LLM's assessments and identify which dialogue characteristics influence user satisfaction. Through this study, we address the following research questions:

- **RQ1:** How accurately do LLM-based dialogue assistants assess NFRs, and how do users perceive the response quality?
- **RQ2:** What dialogue characteristics significantly correlate with user satisfaction?

Our findings show that participants agreed with the system's responses 91–94% of the time across requirement satisfaction level, reasoning, and code location, yet accuracy against expert ground truth was low, with F1 scores of 0.381 for requirement satisfaction level, 0.520 (BERTScore) for reasoning, and F1 score of 0.203 for code location. These results suggest that LLM-based assistants produce responses that are perceived as high-quality but are inaccurate. By modeling user satisfaction as a function of dialogue characteristics, we find that user satisfaction is correlated with three metrics. Verbose responses and a higher volume of information-providing turns are negatively associated with user satisfaction, whereas the number of proactive interaction turns is positively associated. Our results provide insights for designing LLM-based dialogue systems that improve the developer experience in NFR assessment. Our data is open-source and available via: https://github.com/sh3rLock3d/DialogueNFR

2 Related Work and Background

This section discusses how our approach relates to prior work. Table 1 summarizes prior work on evaluating LLMs.

2.1 Evaluation of LLM-based Code Assistants

The evaluation of LLM-based code assistants has been shaped by benchmarks targeting functional correctness, such as HumanEval (Chen et al., 2021), APPS (Hendrycks et al., 2021), CoNaLa (Yin et al., 2018) and (Rodriguez-Cardenas et al., 2023), SWE-bench (Jimenez et al., 2023), LiveCodeBench (Jain et al., 2024), and Aider Polyglot (Aider, 2024). More recently, HumanEvalComm (Wu and Fard, 2025) evaluated whether models ask clarifying questions when functional specifications are ambiguous. However, these benchmarks focus on whether models produce correct final outputs, rather than whether they can reason accurately over multiple turns.

Table 1: Summary of related work in evaluating LLMs.

Work	MT	SM	Evaluation Target
Chen et al. (2021)	–	–	Code generation
Hendrycks et al. (2021)	–	–	Code generation
Yin et al. (2018)	–	–	Code generation
Jimenez et al. (2023)	–	–	Bug fixing
Jain et al. (2024)	–	–	Code generation
Aider (2024)	–	–	Code editing
Wu and Fard (2025)	–	–	Clarification ability
Laban et al. (2025)	✓	–	MT degradation
Vaithilingam et al. (2022)	✓	–	User expectations
Kumar et al. (2025)	✓	–	Agent collaboration
Liu et al. (2024b)	✓	–	Code quality
Sarkar et al. (2022)	✓	–	User experience
Liang et al. (2024)	✓	–	Usability
This work	✓	✓	NFR assessment

MT=Multi-turn, SM=Satisfaction Modeling

2.2 Dialogue System Evaluation

The most widely adopted framework for task-oriented dialogue evaluation is PARADISE (PARAdigm for DIalogue System Evaluation) (Walker et al., 1997), which predicts overall user satisfaction as a function of task success and dialogue costs such as elapsed time and number of turns. PARADISE has been applied across diverse domains (Walker et al., 2002; Forbes-Riley and Litman, 2006; Bickmore and Giorgino, 2006), but to the best of our knowledge, no prior work has applied the PARADISE framework or similar user satisfaction-based dialogue evaluation methods to assess LLM-based dialogue systems. With the rise of LLMs, automatic metrics like BERTScore (Zhang et al., 2019) or LLM-as-judge (Zhuge et al., 2024) have been proposed for utterance-level quality assessment. Yet these metrics evaluate static output quality and do not model how dialogue characteristics influence user satisfaction. Laban et al. (2025) further show that LLM performance degrades in multi-turn conversations, which reinforces the need for evaluation methods that go beyond single-turn quality. In this work, we apply the PARADISE framework to an LLM-based dialogue system and assess the final response obtained through multi-turn collaboration.

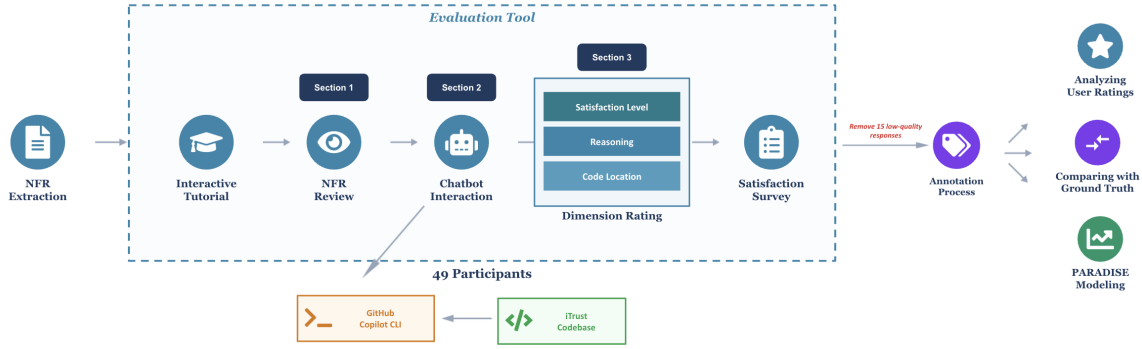

Figure 1: Methodology overview.

2.3 Developer–AI Interaction Studies

A growing body of work examines how developers interact with AI coding tools. Vaithilingam et al. (2022) found gaps between developer expectations and actual experience with LLM-generated code, while Kumar et al. (2025) studied how developers wield agentic AI in real software engineering tasks. Studies have identified issues with overconfidence (Liu et al., 2024b), poor intent specification (Sarkar et al., 2022), and usability limitations (Liang et al., 2024) that functional benchmarks cannot capture. However, these works fall short of identifying which LLM behaviors or dialogue characteristics influence developer satisfaction.

2.4 iTrust

HIPAA is a U.S. federal regulation that establishes standards for protecting the privacy and security of individually identifiable health information, known as electronic protected health information (ePHI). We use HIPAA as a representative domain of regulatory NFRs. To provide a realistic codebase for NFR assessment, we use an EHR system called iTrust (Meneely et al., 2012). The goal of iTrust is to provide students with a project that is real-world relevant and sufficiently deep and complex to simulate industrial systems. iTrust is well-suited for this study, as it serves as a testbed for understanding security and privacy requirements, with a focus on HIPAA rules, making it an appropriate context for evaluating HIPAA-derived NFRs.

3 Methodology

Figure 1 illustrates our methodology. We first extract 148 HIPAA-derived NFRs and construct ground truth assessments against the iTrust codebase. We then recruit 49 programmers to use our evaluation tool, where they review the NFRs, interact with a GitHub Copilot agent to assess them, and rate the agent's responses. They then complete a user satisfaction survey. We annotate the dialogue turns based on dialogue characteristics.

3.1 Ground Truth Creation

To construct the NFR corpus, a privacy and software engineering expert reviewed the HIPAA regulations and extracted 650 statements. Next, to ensure that the NFRs are traceable in iTrust, we applied two inclusion criteria: (1) the statement must pertain to only software construction defined in SWEBOK (Bourque et al., 1999), and (2) it must fall within iTrust's scope. These criteria reduced 650 statements to 180 NFRs. Then, these NFRs were reviewed by another software engineering expert to ensure they satisfied the criteria and met the characteristics of a good NFR as defined by Sommerville (Sommerville, 2011). The other reviewer proposed some edits to the NFR texts and identified 32 NFRs that remain outside the scope of iTrust. These steps reduced 650 statements to the final list of 148 NFRs, covering topics such as technical safeguards and use of ePHI (e.g., *The system shall ensure the integrity of ePHI that it creates.*).

As mentioned earlier, our goal is to evaluate whether LLMs can provide an analyses that match human understanding. Specifically: what is the compliance status (requirement satisfaction level), why does that status hold (reasoning), and where in the code is the requirement addressed (localization). We then manually create a ground truth to compare LLM responses and assess the LLM's accuracy and quality (RQ1). This ground truth includes an answer for each NFR across three dimensions. These

three dimensions are (i) requirement satisfaction Level, which has four values as Satisfied, Weakly Satisfied, Weakly Denied, or Denied (Amyot et al., 2010); (ii) reasoning behind the satisfaction level; and (iii) code locations that address the NFR, including file name and line number(s). We choose these dimensions because, for finding code location related to an NFR, binary links (i.e., related vs. non-related) do not capture how code implements, or relates to requirements (Alor et al., 2025).

To construct the ground truth, the first expert assessed each NFR with respect to iTrust, documenting the satisfaction level, reasoning behind their assessment, and relevant code locations. A second expert then reviewed each annotation to verify its correctness or provide feedback on any issues identified. The first expert either accepted the feedback or, in cases of disagreement, engaged in discussion with the second expert to reconcile differing assessments. Discussions continued until both experts reached an agreement, and the agreed-upon annotations were recorded as the ground truth.

3.2 Survey Design

We design a survey to evaluate LLM performance in assessing NFRs through multi-turn dialogue. In this survey, developers are asked to review a subset of NFRs, query an LLM-based chatbot to assess NFRs, rate the chatbot's responses, and finally complete a user satisfaction survey (adopted from the PARADISE framework (Walker et al., 1997)).

Before starting the surveys, participants complete an interactive tutorial that walks them through the codebase structure and asks them to evaluate three NFRs. Upon successful completion of the tutorial, they proceed with the main survey, which has three stages. In the first stage, developers review 10 assigned NFRs to familiarize themselves with the requirements before interacting with the chatbot. For each NFR, they need to acknowledge that they reviewed it. We also include two attention-check questions to ensure developers read the NFRs carefully. In the second stage, developers query the chatbot to assess the NFRs across the three dimensions. We ask developers to query all NFRs in a single initial prompt, and then follow up based on the agent's initial response to elicit more targeted subsequent prompts. In the third stage, developers rate each NFR's responses on the three dimensions on a Likert scale of 1–4, where 1 is "Strongly Disagree" and 4 is "Strongly Agree". We exclude a neutral option to nudge developers to provide a rating (Jain et al., 2026; Su and McMillan, 2024); however, we provide a text box where developers can explain if they are unable to make a selection. Here, we also include two additional attention-check questions to ensure participants do not respond randomly. After completing the main survey, they are asked to complete a user satisfaction survey and a demographic questionnaire. The satisfaction survey contains questions adopted from the PARADISE framework, which we use to model the system's performance. We ensured that each NFR is assigned to at least two developers.

3.3 Survey Tool

We developed a tool to support NFR review, chatbot interaction, and response rating. Figure 2 shows our tool, which consists of three panels (one for each stage described in Section 3.2).

The chatbot is powered by GitHub Copilot, and we use the GitHub Copilot CLI (GitHub, Inc.) to relay the agent's responses to developers. We use gpt-5.1-codex-max (OpenAI) as the underlying model.

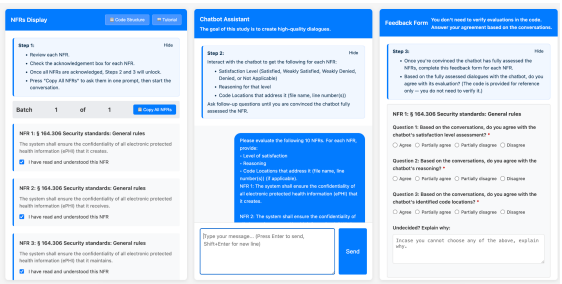

Figure 2: User interface of the tool: review NFRs (left), chatbot interaction (middle), and response rating (right).

3.4 Participants

We first conducted a pilot study with 8 participants: 4 computer science students from our university and 4 recruited through Prolific (Prolific), to validate the study design. The pilot confirmed that the task could be completed in a reasonable time (6 participants completed it within an hour) and that the instructions were clear (everyone who interacted with the chatbot successfully completed the study). For the main study, we recruited 41 participants through Prolific. We required participants to hold a degree in Computer Science or Mathematics, work in a software engineering-related role, and have an approval rating of 95–100% on the platform. Pilot and main study participants were compensated $15 and $20, respectively. This study was approved by our Institutional Review Board (IRB) (see Section

8 for more details). After screening for quality, we removed 15 responses in which participants failed the attention-check questions, user turns were unrelated to the given NFRs, or the dialogues lacked a satisfaction level, reasoning, or code location, resulting in 34 participants.

3.5 Annotation Process

To analyze dialogue characteristics that correlate with user satisfaction (RQ2), we annotate each turn in the collected dialogues using labels drawn from three sources: the PARADISE framework, which provides dialogue cost metrics, MT-Eval (Kwan et al., 2024), and MT-Bench-101 (Bai et al., 2024), which show the interaction patterns in the turns. Appendix B provides a definition of the annotation labels used in this study.

Two experts independently reviewed a subset of 50 turns and developed heuristics for annotation based on these taxonomies. In the first round of annotation, the average inter-rater reliability calculated using Cohen's Kappa (Cohen, 1960) between the two annotators was 0.32, indicating fair agreement. The annotators then resolved disagreements and refined the annotation heuristics. Using the updated heuristics, they re-annotated the same set and achieved a Cohen's kappa of 0.83, indicating almost perfect agreement. After resolving the disagreements, a next round of 54 additional turns yielded an average Cohen's kappa of 0.81, indicating almost perfect agreement again. Given this high level of agreement, the two annotators independently annotated the remaining turns.

4 Results

In this section, we present the results of our study. We outline the basic structure and descriptive statistics of the collected dialogues, and examine the accuracy and perceived quality of the LLM-based assistant's NFR assessments. We then apply the PARADISE framework to identify which dialogue characteristics significantly influence user satisfaction. Table 3 summarizes the key statistics of the collected dialogues.

4.1 RQ1: Response Accuracy and Quality

We evaluate the system's NFR assessments from two perspectives. First, we measure accuracy by comparing the system's outputs against the ground truth in Section 3.1. Second, we measure response quality through participant agreement ratings done in stage three of the survey.

Label	R1	R1-redo	R2
Recollections	0.000	0.640	0.772
Expansion	0.129	0.602	0.666
Refinement	0.176	0.729	0.615
Follow-up	0.289	0.796	0.884
user initiatives	0.668	0.898	1.000
unsuitable-request	–	–	–
inappropriate responses	–	–	–
Error	–	–	–
help messages	0.000	–	–
given-data	0.408	1.000	–
check move	0.369	0.638	0.791
relevant data	–	–	–
abandoned requests	0.000	1.000	–
Anaphora Resolution	0.000	1.000	1.000
Separate Input	–	–	–
Content Confusion	–	–	
Content Rephrasing	0.370	0.912	0.813
Format Rephrasing	0.000	–	–
Self-correction	0.558	0.811	–
Self-affirmation	0.728	0.878	0.769
Instruction Clarification	0.479	0.790	–
Proactive Interaction	1.000	1.000	–
Size	50	50	54
Average κ	0.323	0.836	0.812

Table 2: Inter-rater reliability for dialogue turn labels across annotation rounds. Labels with – indicate insufficient variation in annotations to compute κ.

Statistic	Value
Total # Dialogues	34
Total # Turns	406
Avg. # Turns per Dialogue	11.94
Avg. # Words in Prompt	60.90
Max. # Words in Prompt	496
Avg. # Words in Response	129.46
Max. # Words in Response	877
Avg. # Words per Turn	190.36
Max. # Words per Turn	1206

Table 3: Key statistics of the collected dialogues.

4.1.1 Ground Truth Accuracy

To measure the accuracy of the system's outputs, we manually extracted the system's final responses on the evaluation dimensions from each dialogue and compared them against the ground truth.

We compute accuracy differently for each dimension, due to the nature of each output. Satisfaction level is a categorical label, so we compute precision, recall, and F1 score. Reasoning is free-form text, so we measure semantic similarity between the system's reasoning and the ground truth using BERTScore (Zhang et al., 2019), report mean precision, recall, and F1 across all responses, and additionally report cosine similarity and ROUGE (Lin, 2004) as complementary measures of semantic alignment. Code location consists of sets of file names and line numbers; we define true positives as $|G \cap P|$, false positives as $|P \setminus G|$, and false neg-

Dimension	Metric	Score
Satisfaction Level	Precision	0.438
Satisfaction Level	Recall	0.418
Satisfaction Level	F1	0.381
Reasoning	Precision	0.521
	Recall	0.529
	F1	0.520
	Cosine Similarity	0.774
	ROUGE-1	0.204
	ROUGE-2	0.037
	ROUGE-L	0.156
Code Location	Precision	0.151
Code Location	Recall	0.142
Code Location	F1	0.203

Table 4: Accuracy of LLM against ground truth.

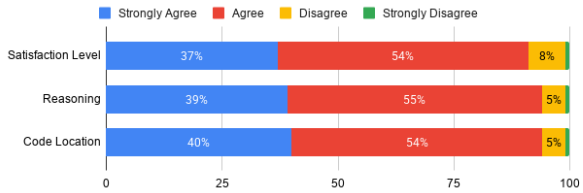

Figure 3: Distribution of participant agreement ratings across the three evaluation dimensions.

Dimension	Mean Agr.	High AD
Satisfaction Level	3.27	19
Reasoning	3.31	13
Code Location	3.34	15

Table 5: Mean participant agreement and number of NFRs where participant rating diverged ($AD > 0.67$).

atives as $|G \setminus P|$, where G is the ground truth set and P is the system's predicted set, and compute the mean precision, recall, and F1 across NFRs.

We present these results in Table 4. The system achieved an F1 score of 0.381 for satisfaction level, only modestly above the 0.25 expected from random assignment across four categories. This suggests that while the system can produce responses with high perceived quality, it struggles to correctly determine the satisfaction status of NFRs. For reasoning, BERTScore F1 is 0.520, cosine similarity is 0.774, and ROUGE-1/2/L F1 scores are 0.204, 0.037, and 0.156, respectively. The relatively higher cosine similarity is expected, as both the system and ground truth address the same NFR and are in the same topical space. However, the lower BERTScore indicates that the generated reasoning diverges from the ground truth. Code location yields a mean F1 of 0.203, indicating that the system also struggles to identify where in the codebase NFRs are addressed.

4.1.2 Participant Perceived Response Quality

Distribution of Ratings. Figure 3 shows the distribution of participant ratings across the three dimensions. The majority of participants agreed or strongly agreed with the system's responses: 91% for satisfaction level, 94% for reasoning, and 94% for code location. Disagreement rates were low, at 8% for satisfaction level, 6% for reasoning, and 5% for code location. "Strongly disagree" ratings were rare, with only 4, 5, and 2 instances for satisfaction level, reasoning, and code location, respectively. Mean agreement was 3.27 for satisfaction level, 3.31 for reasoning, and 3.34 for code location, all above the "agree" threshold (Table 5). To assess whether mean agreement was significantly above the "agree" threshold of 3, we conducted one-sample t-tests with H_0: $\mu \leq 3$ versus H_1: $\mu > 3$. All three dimensions were statistically significant at $p < 0.001$. These results confirm that participants' rating of the system's responses is significantly above the "agree" threshold across all three dimensions, indicating that users generally perceive response quality as high. Table 8 in Appendix C reports the full test statistics. These results stand in contrast to the low accuracy, suggesting that the system produces responses perceived as high-quality by developers but are inaccurate.

Rating Divergence. To identify NFRs where participants diverged in their ratings, we computed the Average Deviation (AD) index (Burke et al., 1999) for each NFR across participants and flagged those with $AD > 0.67$, the recommended cutoff for 4-point Likert scales indicating a lack of consensus (Burke and Dunlap, 2002). We adopt AD because, unlike other agreement metrics, it does not require specification of a null distribution, thereby avoiding assumptions about the nature of random responding (O'Neill, 2017). This is particularly important for our data, where ratings are skewed toward "agree" and "strongly agree," making metrics that assume balanced distributions less appropriate. Of the 148 NFRs, 19 exhibited high divergence on satisfaction level, 13 on reasoning, and 15 on code location (Table 5). Of these, 6 NFRs had high divergence on all three dimensions and 6 on two dimensions. Among single-dimension divergences, code location accounted for 9 cases, satisfaction level for 7, and reasoning for only 1. These results suggest that system quality is less consistent across satisfaction levels for the same NFR than other dimensions.

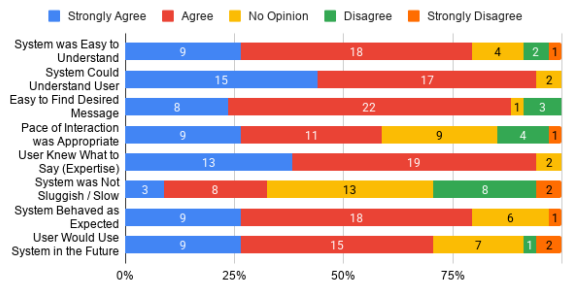

Figure 4: Participants' PARADISE ratings for the LLM.

Label	Count
Given Data	370
Follow-up	195
Recollections	156
Expansion	152
Refinement	53
User Initiatives	37
Proactive Interaction	36
Content Rephrasing	29
Check Move (CM)	27
Self-affirmation	23
Self-correction	16
Separate Input	11
Instruction Clarification	10
Anaphora Resolution	9
Unsuitable Request	3
Abandoned Requests	2
Help Messages	2
Format Rephrasing	2
Inappropriate Responses	1
Error	1
Content Confusion	0
Relevant Data	0

Table 6: Total counts of dialogue act labels.

4.2 RQ2: Dialogue Characteristics and User Satisfaction

To investigate which dialogue characteristics influence user satisfaction, we applied the PARADISE framework, following the methodological considerations outlined by Hajdinjak and Mihelič (2006) and using predictors derived from annotations.

User Satisfaction. After completing the main evaluation, participants filled out a satisfaction survey adopted from the PARADISE framework. Figure 4 summarizes the distribution of user satisfaction ratings. The user satisfaction score (US), computed as the sum of individual Likert responses, ranged from 20 to 37, with a mean of 31.26 out of 40, indicating that users were generally satisfied with the system. The features related to the system's ability to understand the user and the user's ability to know what to say at each point were the most highly rated, averaging 4.3. The features related to the system's speed received the lowest ratings, averaging around 3. These ratings suggest that the system created a fluent and intuitive conversational experience, which may have contributed to the high agreement rates observed in RQ1.

All Predictors. We construct a set of predictors that may correlate with user satisfaction. These predictors include task success, task completion, dialogue costs (Hajdinjak and Mihelič, 2006), and patterns (Kwan et al., 2024; Bai et al., 2024). Task success is computed as the sum of F1 scores across the three evaluation dimensions described in Section 4.1.1. Task completion is computed as the proportion of NFRs for which the system provided all three outputs, except when the NFR is assessed as Denied, in which case we require only requirement satisfaction level and reasoning, since code location is not applicable. Dialogue costs and patterns can be obtained by metadata (e.g., elapsed time) and annotated data. Table 6 summarizes the results of annotated data.

In Table 11 in Appendix D, we report the descriptive statistics for all predictor variables. We first removed predictors with zero variance (*content confusion*, *relevant data count* and *ratio*), as these provide no discriminative information. We then identified highly correlated predictor pairs (i.e., the absolute values of the correlation coefficients $|r| > 0.7$) and, for each pair, retained the predictor with the lower p-value against US. This is because with highly correlated predictors, small errors or variations in the observed values can lead to large changes in the weights of the performance function Hajdinjak and Mihelič (2006). In Table 9 in Appendix D, we show the 9 removed predictors and their correlated counterparts, leaving 21 predictors for the regression. We use these 21 predictors as candidate dialogue characteristics that may correlate with user satisfaction.

Predictors of User Satisfaction. We analyzed which predictors significantly correlate with user satisfaction and can be used to estimate it. To this end, we developed a performance function for the LLM system that predicts user satisfaction based on the predictors. The process for creating this function can help us identify the predictors that are significantly correlated Hajdinjak and Mihelič (2006). Before creating the function, we removed outliers based on deviations from the mean. One dialogue was excluded because its user satisfaction value exceeded two standard deviations from the

Predictor	β	SE	t	p
Mean Words per Response	-0.583	0.174	-3.35	.002
Given data	-0.485	0.162	-2.99	.006
Proactive Interaction	0.422	0.163	2.60	.015

β: standardized coefficient; SE: standard error; t: test statistic.

Table 7: PARADISE performance function coefficients.

mean. Additionally, one dialogue was removed due to an extreme predictor value, with one predictor exceeding five standard deviations from the mean. After removing outliers, 32 dialogues remained in the dataset. We then z-score-normalized all variables to ensure that the resulting regression coefficients reflect each predictor's relative contribution to user satisfaction. We fit the PARADISE performance function using backward elimination (Seber and Lee, 2003) and iteratively removing the least significant predictor until all remaining predictors were significant. The full elimination sequence is reported in Table 10 in Appendix F. The elimination process removed 18 predictors. The final model retained three significant predictors, which result in the following performance function:

$$\begin{aligned}\hat{N}(\text{US}) = &-0.583 \cdot N(\text{Mean Words per Response}) \\ &- 0.485 \cdot N(\text{Given Data}) \\ &+ 0.422 \cdot N(\text{Proactive Interaction})\end{aligned}$$

In this function, N indicates the normalized value of the metric. The results show that user satisfaction is significantly influenced by only these dialogue characteristics. We report the coefficient details in Table 7. The negative coefficients for *mean bot words* ($\beta = -0.583$) and *given data count* ($\beta = -0.485$) suggest that verbose responses are associated with lower user satisfaction, even when the information itself may be relevant. In contrast, *proactive interaction* ($\beta = 0.422$) as the only positive predictor, suggests that participants value the system taking the initiative in the dialogue and contribute positively to user satisfaction.

5 Discussion

5.1 The Gap between Rating and Accuracy

LLM-based code assistants have demonstrated strong performance on functional correctness (Chen et al., 2021; Li et al., 2022; Jain et al., 2024). However, our results reveal that this competence does not extend to NFRs. Participants agreed or strongly agreed with the system's outputs over 91% of the time, yet F1 scores against expert ground truth were low. As an example, an NFR requires the system to implement procedures for creating passwords. The ground truth provides its reasoning based on dedicated creation actions. However, the system reasons based on `ResetPasswordAction.java` as evidence and conflates password reset with password creation. Here, the system failed to discern semantically similar yet functionally distinct operations. Despite this mistake, the system produced a sufficiently convincing response that the participant strongly agreed with the incorrect assessment. This example shows that while LLMs can generate convincing NFR assessments that satisfy developers, their actual accuracy remains low. Several factors, such as LLM's confidence in responses, NFRs' attributes (e,g,. ambiguity), or lack of expertise may have contributed to this gap. Understanding precisely which factors drive the gap, and how to mitigate them, is a direction for future work.

5.2 Dialogue Design Implications

By applying the PARADISE framework, we derived a function that measures the performance of LLM agents for evaluating NFRs. This function provides actionable insights for developing effective LLM-based dialogue assistants. Beyond measuring the performance, our data enables a further step toward fully automated evaluation. Creating a dialogue system still requires human participants to interact with the system, whereas user simulation can automate the evaluation process entirely. Our annotated dialogue corpus provides the foundation for training user simulators that can evaluate LLM-based agents without requiring human participants, which we plan to pursue in future work.

6 Conclusions

This study investigated multi-turn dialogues between developers and an LLM-based agent for evaluating NFRs. Our results reveal that participants agreed with the system's outputs over 91% of the time, yet actual accuracy was low (F1 of 0.381 for satisfaction level, 0.520 for reasoning, and 0.203 for code localization). By applying the PARADISE framework, we found that a high number of proactive interactions positively influence user satisfaction, while verbose responses and a high number of information-providing responses detract from it. These findings provide guidance on how LLM-based dialogue agents should interact with developers in NFR assessment tasks.

7 Limitations

This study has several limitations that should be considered when interpreting the results.

First, the NFR corpus was initially extracted by a single privacy and software engineering expert who reviewed the HIPAA regulations and derived 180 NFRs from 650 extracted statements. A second expert reviewed and validated these extracted NFRs, which raises the possibility that some relevant requirements were missed. HIPAA is a complex and extensive regulatory framework, and a single reviewer may overlook statements that a second independent extractor would have identified. In the future, we will ensure that two experts independently extract NFRs from the regulations and reconcile their results to ensure a more comprehensive coverage of the regulatory space.

Second, we evaluated a single LLM-based agent (GitHub Copilot with gpt-5.1-codex-max) against a single codebase (iTrust) in one regulatory domain (HIPAA). While iTrust is a testbed for privacy and security requirements, the generalizability of our findings to other codebases, regulatory frameworks, or LLM-based agents remains an open question.

Third, like most survey-based studies in software engineering, this work faces several validity threats. Variations in the source code and NFRs, participant selection, or the survey interface could affect outcomes. To address these concerns, we use the iTrust EHR system, a well-established codebase specifically designed to comply with HIPAA regulations, from which we draw our NFRs, and a simple web interface to avoid the complexity of tools like IDEs. Another problem is that our participant pool, recruited through Prolific with screening filters to target individuals with software development experience, may not fully represent all developers who engage with NFR assessment in practice or have knowledge in HIPAA compliance, and may be subject to selection bias. The study is also subject to self-report bias and recall bias, as participants rated the system's responses based on their own understanding of HIPAA compliance, which may not reflect expert-level judgment, and may have misremembered details of the interaction when completing the satisfaction survey.

Fourth, our analysis was conducted on a relatively small dataset, i.e., 32 dialogues, which limits the statistical power of the analysis. Although we applied standard procedures, including predictor filtering, normalization, and backward elimination, the identified predictors and their coefficients may not be stable or generalizable beyond this dataset.

Finally, our user satisfaction function should not be used to increase or optimize user satisfaction in an LLM agent, because it may make inaccurate responses seem more convincing.

8 Ethics Consideration

This study was approved by the University of Maine's Institutional Review Board (IRB) and conducted in compliance with institutional ethical standards. Prior to participation, all individuals were provided informed consent through a form detailing the investigators, risks, benefits, compensation, and confidentiality. They were informed that participation was voluntary, and individuals were free to withdraw at any point. Contact information for the investigators and the IRB was provided in the consent form. No participants contacted either party during the study.

9 Acknowledgement

This research was supported by NSF Award #2442683.

References

Aider. 2024. o1 tops aider's new polyglot leaderboard. https://aider.chat/2024/12/21/polyglot.html. Accessed: 2026-01-08.

Ebube Alor, SayedHassan Khatoonabadi, and Emad Shihab. 2025. Evaluating the use of llms for documentation to code traceability. *arXiv preprint arXiv:2506.16440*.

Daniel Amyot, Sepideh Ghanavati, Jennifer Horkoff, Gunter Mussbacher, Liam Peyton, and Eric Yu. 2010. Evaluating goal models within the goal-oriented requirement language. *International Journal of Intelligent Systems*, 25(8):841–877.

Chetan Arora, John Grundy, and Mohamed Abdelrazek. 2024. Advancing requirements engineering through generative ai: Assessing the role of llms. In *Generative AI for Effective Software Development*, pages 129–148. Springer.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and 1 others. 2024. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7421–7454.

Timothy Bickmore and Toni Giorgino. 2006. Health dialog systems for patients and consumers. *Journal of biomedical informatics*, 39(5):556–571.

Pierre Bourque, Robert Dupuis, Alain Abran, James W Moore, and Leonard Tripp. 1999. The guide to the software engineering body of knowledge. *IEEE software*, 16(6):35–44.

Michael J Burke and William P Dunlap. 2002. Estimating interrater agreement with the average deviation index: A user's guide. *Organizational research methods*, 5(2):159–172.

Michael J Burke, Lisa M Finkelstein, and Michelle S Dusig. 1999. On average deviation indices for estimating interrater agreement. *Organizational Research Methods*, 2(1):49–68.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. 2023. Large language models for software engineering: Survey and open problems. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, pages 31–53. IEEE.

James D Finch and Jinho D Choi. 2020. Emora stdm: A versatile framework for innovative dialogue system development. *arXiv preprint arXiv:2006.06143*.

Kate Forbes-Riley and Diane Litman. 2006. Modelling user satisfaction and student learning in a spoken dialogue tutoring system with generic, tutoring, and user affect parameters. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 264–271.

GitHub, Inc. GitHub Copilot CLI. Accessed: March 2026.

Martin Glinz and 1 others. 2005. Rethinking the notion of non-functional requirements. In *Proc. Third World Congress for Software Quality*, volume 2, pages 55–64.

Victor M González and Gloria Mark. 2004. " constant, constant, multi-tasking craziness" managing multiple working spheres. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120.

Henrik Gustavsson, Damir Bilic, Jan Carlson, and Eduard Paul Enoiu. 2024. Success factors in the specification of operational scenarios-an industrial perspective. In *2024 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE.

Melita Hajdinjak and France Mihelič. 2006. The paradise evaluation framework: Issues and findings. *Computational Linguistics*, 32(2):263–272.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and 1 others. 2021. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*.

hhs. Health information privacy. https://www.hhs.gov/hipaa/index.html. Accessed: 2026-04-19.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Vijayanta Jain, Sepideh Ghanavati, Sai Teja Peddinti, and Collin McMillan. 2026. Automated generation of accurate privacy captions from android source code using large language models. *arXiv preprint arXiv:2601.06276*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Sungmin Kang, Juyeon Yoon, and Shin Yoo. 2023. Large language models are few-shot testers: Exploring llm-based general bug reproduction. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2312–2323. IEEE.

Aayush Kumar, Yasharth Bajpai, Sumit Gulwani, Gustavo Soares, and Emerson Murphy-Hill. 2025. Sharp tools: How developers wield agentic ai in real software engineering tasks. *arXiv e-prints*, pages arXiv–2506.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2024. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20153–20177.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.

Thomas D LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining mental models: a study of developer work habits. In *Proceedings of the 28th international conference on Software engineering*, pages 492–501.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, and 1 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Jenny T Liang, Chenyang Yang, and Brad A Myers. 2024. A large-scale survey on the usability of ai programming assistants: Successes and challenges. In *Proceedings of the 46th IEEE/ACM international conference on software engineering*, pages 1–13.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Feng Lin, Dong Jae Kim, Zhenhao Li, Jinqiu Yang, and 1 others. 2025. Robunfr: Evaluating the robustness of large language models on non-functional requirements aware code generation. *arXiv preprint arXiv:2503.22851*.

Junwei Liu, Kaixin Wang, Yixuan Chen, Xin Peng, Zhenpeng Chen, Lingming Zhang, and Yiling Lou. 2024a. Large language model-based agents for software engineering: A survey. *arXiv preprint arXiv:2409.02977*.

Yue Liu, Thanh Le-Cong, Ratnadira Widyasari, Chakkrit Tantithamthavorn, Li Li, Xuan-Bach D Le, and David Lo. 2024b. Refining chatgpt-generated code: Characterizing and mitigating code quality issues. *ACM Transactions on Software Engineering and Methodology*, 33(5):1–26.

Dewi Mairiza, Didar Zowghi, and Nurie Nurmuliani. 2010. An investigation into the notion of non-functional requirements. In *Proceedings of the 2010 ACM symposium on applied computing*, pages 311–317.

Andrew Meneely, Ben Smith, and Laurie Williams. 2012. Appendix b: itrust electronic health care system case study. *Software and Systems Traceability*, 425.

Xiangxin Meng, Zexiong Ma, Pengfei Gao, and Chao Peng. 2024. An empirical study on llm-based agents for automated bug fixing. *arXiv preprint arXiv:2411.10213*.

Stoyan Nikolov, Daniele Codecasa, Anna Sjövall, Maxim Tabachnyk, Satish Chandra, Siddharth Taneja, and Celal Ziftci. 2025. How is google using ai for internal code migrations? In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 481–492. IEEE.

Thomas A O'Neill. 2017. An overview of interrater agreement on likert scales for researchers and practitioners. *Frontiers in psychology*, 8:777.

OpenAI. Building more with GPT-5.1-Codex-Max. Accessed: March 2026.

Ipek Ozkaya. 2023. Application of large language models to software engineering tasks: Opportunities, risks, and implications. *IEEE Software*, 40(3):4–8.

Leslie Perlow and John Weeks. 2002. Who's helping whom? layers of culture and workplace behavior. *Journal of Organizational Behavior: The International Journal of Industrial, Occupational and Organizational Psychology and Behavior*, 23(4):345–361.

Prolific. Prolific. https://www.prolific.com. Accessed: 2026-04-17.

Daniel Rodriguez-Cardenas, David N Palacio, Dipin Khati, Henry Burke, and Denys Poshyvanyk. 2023. Benchmarking causal study to interpret large language models for source code. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 329–334. IEEE.

Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. 2012. How do professional developers comprehend software? In *2012 34th International Conference on Software Engineering (ICSE)*, pages 255–265. IEEE.

Pat Rondon, Renyao Wei, José Cambronero, Jürgen Cito, Aaron Sun, Siddhant Sanyam, Michele Tufano, and Satish Chandra. 2025. Evaluating agent-based program repair at google. *arXiv preprint arXiv:2501.07531*.

Advait Sarkar, Andrew D Gordon, Carina Negreanu, Christian Poelitz, Sruti Srinivasa Ragavan, and Ben Zorn. 2022. What is it like to program with artificial intelligence? *arXiv preprint arXiv:2208.06213*.

George AF Seber and Alan J Lee. 2003. *Linear regression analysis*. John Wiley & Sons.

Jonathan Sillito, Gail C Murphy, and Kris De Volder. 2008. Asking and answering questions during a programming change task. *IEEE Transactions on Software Engineering*, 34(4):434–451.

Ian Sommerville. 2011. Software engineering 9th edition. *ISBN-10*, 137035152:18.

Chia-Yi Su and Collin McMillan. 2024. Distilled gpt for source code summarization. *Automated Software Engineering*, 31(1):22.

Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*, pages 1–7.

Marilyn Walker, Diane Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 271–280.

Marilyn A Walker, Alexander I Rudnicky, John S Aberdeen, Elizabeth Owen Bratt, John S Garofolo, Helen Wright Hastie, Audrey N Le, Bryan L Pellom,

Alexandros Potamianos, Rebecca J Passonneau, and 1 others. 2002. Darpa communicator evaluation: progress from 2000 to 2001. In *Interspeech*, pages 273–276.

Jie JW Wu and Fatemeh H Fard. 2025. Humanevalcomm: Benchmarking the communication competence of code generation for llms and llm agents. *ACM Transactions on Software Engineering and Methodology*, 34(7):1–42.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In *Proceedings of the 15th international conference on mining software repositories*, pages 476–486.

Quanjun Zhang, Chunrong Fang, Yang Xie, Yaxin Zhang, Yun Yang, Weisong Sun, Shengcheng Yu, and Zhenyu Chen. 2023. A survey on large language models for software engineering. *arXiv preprint arXiv:2312.15223*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, and 1 others. 2024. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*.

A Participants Demographic

We recruited a total of 49 participants for our study: 8 from a pilot study and 41 from the main study. Figure 5 shows the demographic distribution of participants in each group.

B Annotation Definitions

Table 12 defines the annotation labels used in this study, along with their definitions and examples from our corpus.

C Statistical Significance Tests

In Table 8, we report the one-sample t-test results for participant agreement across the three evaluation dimensions.

Dimension	Mean	SD	t	p
Satisfaction Level	3.27	0.66	7.49	$< .001$
Reasoning	3.31	0.63	9.05	$< .001$
Code Location	3.34	0.60	10.41	$< .001$

Table 8: One-sample t-test results for participant agreement (H_0: $\mu \leq 3$).

Removed	Retained	r
Errors	Inappropriate Resp.	1.000
Inappropriate Resp. Ratio	Inappropriate Resp.	1.000
Abandoned Requests	Abandoned Req. Ratio	0.999
Help Message Ratio	Help Messages	0.996
Number of Turns	Given data	0.981
Expansion	Given data	0.867
Follow-up	Given data	0.860
Recollections	Given data	0.853
Unsuitable Req. Ratio	Unsuitable Requests	0.806

Table 9: Predictors removed due to high correlation ($|r| > 0.7$). For each pair, the predictor with the higher p-value against US was removed.

Step	Removed Predictor	p	R^2
1	Mean Words Request	0.853	0.728
2	Given data Ratio	0.802	0.727
3	Mean Response Time	0.865	0.726
4	Task Completion	0.803	0.725
5	Instruction Clarification	0.725	0.724
6	Self-correction	0.681	0.721
7	Check Move Ratio	0.652	0.718
8	Check Moves	0.688	0.714
9	Unsuitable Requests	0.534	0.711
10	Inappropriate Responses	0.484	0.705
11	Anaphora Resolution	0.446	0.697
12	User Initiatives	0.138	0.688
13	Elapsed Time	0.074	0.652
14	Refinement	0.111	0.597
15	Help Messages	0.150	0.549
16	Self-affirmation	0.157	0.507
17	Abandoned Request Ratio	0.166	0.465
18	Task Success	0.134	0.423

Table 10: Backward elimination steps. At each step, the predictor with the highest p-value was removed. The process terminated when all remaining predictors were significant ($p < 0.05$).

D Predictors

In Table 11, we report the descriptive statistics for user satisfaction and all predictor variables used in the PARADISE regression model.

E High Correlated Predictors

Table 9 lists the predictors removed due to high correlation ($|r| > 0.7$)

F Backward Elimination Steps

In Table 10, we report the full backward elimination sequence.

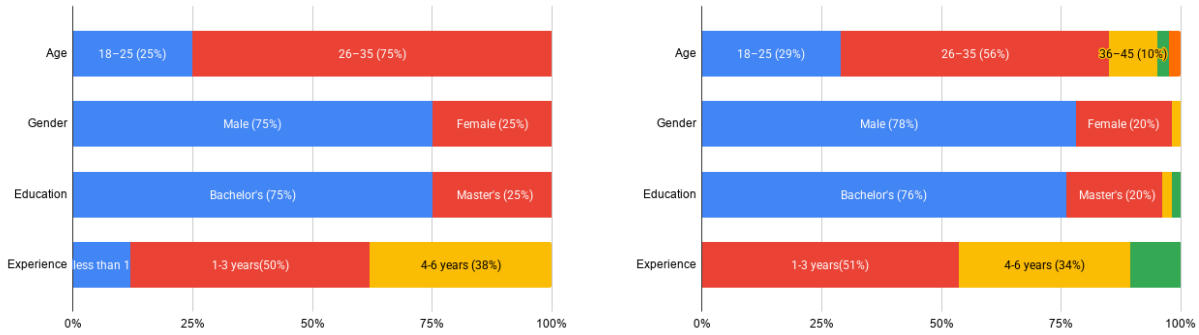

Figure 5: Demographic distribution of pilot study participants (n=8, left) and main study participants (n=41, right).

Variable	**Mean**	**SD**	**Min**	**Max**
User Satisfaction (US)	31.26	3.65	20.00	37.00
Dialogue Cost Metrics - meta-data (Hajdinjak and Mihelič, 2006)				
Number of Turns	11.94	7.95	2.00	27.00
Mean Response Time (s)	45.80	33.05	12.59	157.77
Elapsed Time (s)	3496.42	9230.81	312.92	55369.17
Mean Words per Request	60.90	42.13	9.41	229.50
Mean Words per Response	129.46	105.49	50.91	548.00
Dialogue Cost Metrics - Counts (Hajdinjak and Mihelič, 2006)				
User Initiatives	1.09	0.38	1.00	3.00
Unsuitable Requests	0.09	0.29	0.00	1.00
Inappropriate Responses	0.03	0.17	0.00	1.00
Errors	0.03	0.17	0.00	1.00
Help Messages	0.06	0.24	0.00	1.00
Given data	10.88	7.47	2.00	26.00
Check Moves	0.79	1.09	0.00	4.00
Relevant Data	0.00	0.00	0.00	0.00
Abandoned Requests	0.06	0.24	0.00	1.00
Dialogue Cost Metrics - Ratios (Hajdinjak and Mihelič, 2006)				
Unsuitable Request Ratio	0.009	0.036	0.000	0.200
Inappropriate Response Ratio	0.004	0.021	0.000	0.125
Help Message Ratio	0.003	0.014	0.000	0.063
Given data Ratio	0.907	0.121	0.583	1.000
Check Move Ratio	0.064	0.105	0.000	0.500
Relevant Data Ratio	0.000	0.000	0.000	0.000
Abandoned Request Ratio	0.002	0.009	0.000	0.040
Interaction Pattern Labels (Kwan et al., 2024; Bai et al., 2024)				
Recollections	4.59	3.64	0.00	11.00
Expansion	4.47	4.01	0.00	12.00
Refinement	1.56	2.27	0.00	10.00
Follow-up	5.74	5.48	0.00	18.00
Anaphora Resolution	0.26	0.57	0.00	2.00
Content Confusion	0.00	0.00	0.00	0.00
Self-correction	0.47	1.11	0.00	5.00
Self-affirmation	0.68	1.39	0.00	5.00
Instruction Clarification	0.29	0.52	0.00	2.00
Proactive Interaction	1.06	2.17	0.00	12.00
Task Metrics				
Task Success	0.95	0.26	0.55	1.80
Task Completion	0.90	0.20	0.10	1.00

Table 11: Descriptive statistics for user satisfaction and all predictor.

Table 12: Annotation schema with definitions and examples from our study.

Metric	**Definition**	**Notes / Example**
Recollections	User returns to a topic or detail from earlier in the dialogue (not the immediately preceding turn), requiring the system to retrieve information from its prior context.	Applies when the user explicitly navigates back to a previously discussed topic. E.g., *"Ok, back to NFR1: What exactly constitutes a 'discrepancy' in login attempts—e.g., multiple failures from one IP, geolocation mismatches, or failed 2FA?"*
Expansion	User explores a new facet or sub-aspect of the current topic without changing to a different topic entirely.	Often occurs when the user has already asked about one dimension (e.g., satisfaction level) and then asks about another (e.g., code location) for the same NFR. E.g., *"What about access of authorized software programs?"* (a new facet of NFR 12 after discussing human-role access). May co-occur with Follow-up.
Refinement	User modifies, narrows, or corrects their previous instructions after seeing the system's output.	The user reviews the output and asks for a changed version. May trigger Content Rephrasing, Format Rephrasing, or Self-correction on the system side. E.g., *"Shouldn't the office visit details be covered within 'patient records'? Dig deeper into office visit details to clarify this."*
Follow-up	User asks a question that builds directly on the system's most recent response, referencing specific details or opinions from it.	The key criterion is *most recent response*. If the user refers back to an earlier turn, annotate as Recollections instead. E.g., *"Where in code can this be found?"* (referencing the role-enforcement explanation just given)
User Initiative	User starts a new information-seeking request or introduces a new topic.	Includes the first turn of a session as well as mid-conversation topic shifts (e.g., when the user moves from one NFR to the next). E.g., *"NFR 5: The system shall support implementing policies..."* (introducing a new NFR for evaluation)
Unsuitable Request	User's request is out of scope or context of the system's domain.	No instances in our corpus.

Continued on next page

Table 12 (continued)

Metric	Definition	Notes / Example
Inappropriate Response	System produces an unexpected or erroneous response, including pardon moves.	No instances in our corpus.
Error	System errors, including malformed or incoherent natural-language output.	No instances in our corpus.
Help Message	System proactively provides guidance, recommendations, or actionable steps beyond what was directly requested.	Distinct from Given Data: GD answers the question asked; HM goes further by offering unsolicited help. E.g., user asks *"Why is NFR 38 denied? Indicate where this check needs to occur."* and the system not only explains the gap but recommends where authorization checks should be added.
Given Data	System provides the information the user directly requested.	Present in the majority of turns. E.g., system returns satisfaction level, reasoning, and code locations for each NFR in the batch.
Check Move	User or system requests verification or confirmation of something stated in a prior turn. Extended from the original PARADISE definition (system-only) to include user-initiated checks.	E.g., *"Can you validate this again to make sure the roles are correct?"* or *"So nothing was accomplished at this NFR, right? Not a single thing."* or user challenges: *"Are you sure that is true?"*
Relevant Data	System directs the user to select from relevant, available data.	No instances in our corpus.
Abandoned Request	User drops a request without receiving a satisfying answer and moves on.	E.g., *"Let's start again from 0: Please evaluate the following 10 NFRs..."* (abandoning the prior single-NFR approach)
Anaphora Resolution	The user's query contains a pronoun or demonstrative whose referent must be resolved from prior turns.	E.g., *"How is this varified?"* ("this" refers to ePHI integrity from the prior turn). Also: *"I have the same question for NFR 39."* ("the same question" refers to the NFR 38 query pattern)
Separate Input	Task requirements and the actual input data are provided across different turns rather than in a single message.	E.g., Turn 1: user establishes evaluation criteria (satisfaction level, reasoning, code locations). Turn 7: *"With the experience evaluating NFR 11, analyze these NFRs as well. [NFR 12–20]"*
Content Confusion	User's query conflates superficially similar but semantically distinct concepts, requiring the system to disambiguate.	E.g., *"There are encryption terms for the user authentication, check again."* (user conflates password hashing with ePHI encryption; system correctly distinguishes SHA-256 hashing from data-at-rest encryption)

Continued on next page

Table 12 (continued)

Metric	Definition	Notes / Example
Content Rephrasing	System rephrases the content of its previous response to provide more depth or detail, triggered by user feedback.	Describes the system's response; often co-occurs with Refinement on the user side. E.g., *"I want Reasoning to be a little more detailed, thank you."* System regenerates with expanded justifications.
Format Rephrasing	System restructures the format or presentation of its previous response without changing the substance, triggered by user feedback.	Describes the system's response; often co-occurs with Refinement on the user side. E.g., *"Ok but you didn't provide line numbers for those files."* and *"Nope, line number. I need exact code not just file names."*
Self-correction	System revises its previous answer based on the user's feedback or challenge.	E.g., user asks *"So is that NFR satisfaction level correct for NFR 12 with the part of the authorized software programs?"* System downgrades NFR 12 from Satisfied to Weakly Satisfied.
Self-affirmation	System maintains its previous position when challenged by the user.	E.g., user asks *"Is this not just about getting the information during an emergency for the ER roles?"* System maintains Weakly Satisfied, explaining the gap between ER-role access and broader emergency override.
Instruction Clarification	System asks questions because the user's request is too ambiguous to proceed without more information.	E.g., user says *"Ok but that's not implemented, I don't see the point to that."* System responds: *"Can you tell me what behavior you're seeing? Are you testing through the normal /login.jsp flow or elsewhere?"*
Proactive Interaction	System voluntarily asks probing questions to deepen the conversation, even though it could answer without them.	Distinct from Instruction Clarification: the system does not *need* the information to proceed but asks to encourage exploration. E.g., after listing missing PHI coverage areas, the system asks: *"Which PHI areas matter most to you so I can dig deeper?"*