How Reliable Are LLM Evaluations of Microservice Architecture Diagrams?

João Luiz Cerqueira
Federal University of Minas Gerais
Belo Horizonte, Brazil
joaolfc@ufmg.br

Mateus Dutra
Federal University of Minas Gerais
Belo Horizonte, Brazil
mateusmdutra@ufmg.br

Eduardo Figueiredo
Federal University of Minas Gerais
Belo Horizonte, Brazil
figueiredo@dcc.ufmg.br

Gustavo Vale
Federal University of Minas Gerais
Belo Horizonte, Brazil
Federal University of Lavras
Lavras, Brazil
gustavovale@dcc.ufmg.br

Abstract

Architecture diagrams play a central role in communicating design intent in microservice systems, but their quality evaluation remains largely manual, subjective, and difficult to scale. This paper investigates the reliability of Large Language Models (LLMs) as automated reviewers of microservice architecture diagrams. We design a guideline-based evaluation framework grounded in five established quality criteria (clarity, consistency, completeness, accuracy, and level of detail) and apply it to assess diagrams using two state-of-the-art multimodal LLMs (GPT-5 and Claude Sonnet 4.6) across both visual (PNG) and textual (SVG/XML) representations. To evaluate reliability, we conducted a human audit with 16 software engineers, which yielded human reference scores for 360 guideline-level LLM evaluations and 253 pairwise comparisons of LLM-generated reasoning. Our results show that LLM-generated scores achieve reasonable agreement with human reference scores, with an overall F1-score of 0.761 and fair ordinal agreement ($\kappa = 0.308$). Agreement varies more clearly by quality criterion than by representation, with completeness showing the strongest alignment and clarity the weakest. Although image-based inputs consistently outperform textual inputs descriptively, the representation effect is not statistically robust for score agreement. In contrast, experts significantly prefer image-based reasoning in pairwise comparisons, selecting it in 60.8% of decided cases. These findings suggest that LLMs can effectively support early-stage architectural reviews by identifying potential architecture issues and prioritizing them for human inspection, but their evaluations still require expert validation.

Keywords

Large Language Model, Software Architecture, Microservices

1 Introduction

Software architecture diagrams are a primary medium for communicating system structure, design decisions, and dependencies among stakeholders [18]. In microservice-based systems, where functionality is decomposed into independently deployable services [19, 28] that are prone to architectural degradation and microservice-specific bad smells over time [13], these diagrams play a critical role in reasoning about service boundaries, communication paths, data ownership, and infrastructure components such as gateways, message brokers, and databases. However, previous studies [10, 20, 21] show that architecture diagrams are often incomplete, inconsistent, and expressed using ad hoc notations, which hinders comprehension and reduces trust in documentation. As a result, architecture review remains a largely manual, subjective, and time-consuming task that depends on scarce expert effort.

Large Language Models (LLMs) have recently demonstrated strong capabilities in software engineering tasks, including code generation, documentation, testing, and program comprehension [15, 16, 24]. More recently, they have also been explored in software architecture contexts, such as collaborative architecture, architectural design decision support, and architecture knowledge evaluation [1, 11, 14, 26]. However, empirical evidence on the use of LLMs to *assess architecture diagrams* remains limited [17]. This task is different from evaluating code or textual documentation because diagrams are inherently visual, often rely on informal notations, and encode meaning through layout, symbols, labels, and spatial relationships [18]. These characteristics raise an important question: **can LLMs provide reliable architectural feedback when the artifact under evaluation is a diagram?**

The closest study to ours [10] evaluates a single LLM on four architecture diagrams and analyzes whether the generated evaluations met expert expectations. Although it provides early evidence that LLMs can support diagram evaluation, important reliability questions remain open. In particular, it does not define a guideline-level scoring framework, compare LLM-generated scores against human reference scores, or separate score agreement from the perceived usefulness of LLM-generated reasoning. Moreover, it remains unclear whether the representation of the diagram affects the evaluation. A model may receive the same architecture as a rendered image, such as a PNG file, or as a textual source representation, such as SVG or draw.io XML. These representations preserve different kinds of information and may influence both the assigned scores and the explanations produced by the model.

In this paper, we investigate the reliability of LLM-based evaluations of microservice architecture diagrams. We define a guideline-based evaluation framework grounded in five established quality criteria: accuracy, clarity, completeness, consistency, and level of detail. For each criterion, we derive explicit guideline-level checks and

require the model to produce both a score and a textual justification. This design allows us to evaluate LLM outputs at a fine-grained level, analyze whether agreement with human judgment varies across quality criteria, and compare evaluations generated from image-based and text-based diagram representations.

We conduct an empirical study with six real-world microservice systems mined from the literature [5] whose architecture diagrams are available in both PNG and textual source representations. The textual representation corresponds to the source files used to generate or document the diagrams, including SVG and draw.io XML formats. To this end, we evaluate two state-of-the-art multimodal LLMs, GPT-5 and Claude Sonnet 4.6, across two diagram representations, five quality criteria, and fifteen guideline-level checks, resulting in 120 LLM calls and 360 guideline-level LLM scores. We also conduct a human audit with software engineers, who provided reference scores for the diagrams and compared anonymized LLM-generated justifications produced from different diagram representations. This design allows us to analyze two complementary dimensions of reliability: whether LLM-assigned scores align with human reference scores and whether experts prefer the reasoning generated from image-based or text-based inputs.

Our results show that LLM evaluations achieve reasonable binary agreement with human reference scores, with an overall F1-score of 0.761 and fair ordinal agreement ($\kappa = 0.308$). Agreement varies more clearly by quality criterion than by representation. Completeness shows the strongest alignment with human judgment, while clarity shows the weakest binary performance. Surprisingly, image-based inputs consistently outperform textual inputs descriptively, especially in precision and ordinal agreementIn contrast, expert preference reveals a clearer representation effect: among decided pairwise comparisons, experts prefer image-based reasoning in 60.8% of cases, with a Wilson 95% confidence interval of [53.6%, 67.5%], and the difference is statistically significant ($p = 0.0034$).

This paper makes the following contributions:

- **A guideline-based evaluation framework** for assessing architecture diagrams with LLMs, enabling fine-grained and auditable analysis of scores and justifications.
- **An empirical study** of LLM-based diagram evaluation across multiple controlled factors, including project, model, diagram representation, quality criterion, and guideline-level checks.
- **Evidence on the reliability of LLM-assigned scores**, comparing LLM outputs against human reference scores and showing how agreement varies across quality criteria.
- **Evidence on the role of diagram representation**, comparing PNG and textual source representations to analyze their effect on both score agreement and expert preference for generated reasoning.
- **A human audit of LLM-generated evaluations**, including human reference scores and pairwise comparisons between justifications generated from different diagram representations.
- **A reproducibility package** containing the data, experimental protocol, systems, prompts, and analysis scripts used in the study [8].

Overall, our findings suggest that LLMs can support architecture review as first-pass assessors. They can help identify potential quality issues, organize feedback around explicit criteria, and prioritize diagrams or guidelines that require closer human inspection. However, their outputs should not be treated as authoritative judgments. The modest ordinal agreement, the criterion-dependent reliability, and the stronger expert preference for image-based reasoning indicate that LLM-based diagram evaluation is most appropriate in a human-in-the-loop workflow, where automated feedback supports expert review rather than replacing it.

2 Background And Related Work

This section situates our work at the intersection of software architecture documentation, architecture diagram quality, and LLM-based support for software architecture tasks.

2.1 Software Architecture Diagrams

Software architecture descriptions capture the key elements of a system, their relationships, and the rationale behind design decisions [7, 25]. Diagrams are a central part of these descriptions because they provide a compact and visual representation of architectural structure, enabling stakeholders to reason about components, dependencies, interactions, and deployment concerns [21, 25]. In microservice-based systems, this role becomes particularly important because architectural understanding depends on how services, APIs, databases, message brokers, gateways, and external actors interact [30]. This reliance on diagrams is also observed in practice: in an industrial survey on migration to microservice architectures, practitioners reported box-and-line diagrams as one of the main forms used to document the new architecture [12, 23, 27].

Despite their importance, architecture diagrams are often difficult to interpret and maintain [25]. Empirical studies [20] show that the diagrams in software projects are frequently informal, heterogeneous, and only partially aligned with the implemented system. In open-source projects, diagrams may mix abstraction levels, omit relevant architectural elements, and rely on ad hoc notation, making consistent interpretation difficult [20]. Previous work also shows that ambiguous or incomplete diagrams can lead to misunderstandings between stakeholders, negatively affecting communication, maintenance, and architectural decision-making [18, 21]. These challenges motivate the need for more systematic and scalable approaches to assess the quality of architecture diagrams.

2.2 Quality Criteria For Architecture Diagrams

The literature on software architecture documentation emphasizes that diagrams should support both structural correctness and effective communication [21]. Previous studies have discussed quality properties such as accuracy, clarity, completeness, consistency, and an appropriate level of detail [10, 20, 21]. These criteria capture complementary aspects of diagram quality. Table 1 summarizes their meaning and the corresponding guideline-level checks. Although these criteria are useful for characterizing diagram quality, they are broad dimensions rather than concrete evaluation units. This fact creates a challenge for empirical evaluation because two reviewers may agree that clarity or completeness is important, but

Table 1: Quality criteria and guideline-level checks used in the evaluation framework.

Criterion	Meaning	Guidelines
Accuracy [10]	Whether the diagram correctly reflects the architecture described in the text, including components, interactions, and technologies.	1. Component matching 2. Interaction matching [21]
Clarity [10]	Whether a technical or semi-technical reader can easily understand the architecture from the diagram and description.	3. Element identification 4. Reading flow 5. Symbol and label readability 6. Specific symbology [21]
Completeness [10]	Whether the diagram includes the main architectural elements and interactions needed to understand the system as a whole.	7. Coverage of main components 8. External actors coverage 9. Main interactions coverage [21]
Consistency [10]	Whether the diagram uses a clearly defined notation and applies it uniformly, including consistent representation of specific technologies.	10. Notation definition and uniformity 11. Technology symbols 12. Naming and representation consistency [21]
Level of Detail [10]	Whether the diagram presents elements and interactions with an appropriate amount of detail for developers and architects.	13. Element detail 14. Interaction detail 15. Overall appropriateness for developers and architects [21]

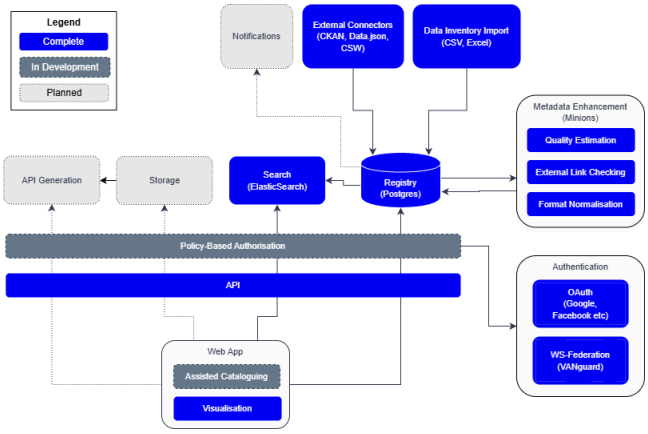

Figure 1: Magda.io architecture diagram used to illustrate the guideline-level evaluation.

still assess these criteria differently if they are not operationalized into explicit checks. To address this issue, our study builds on practitioner-oriented recommendations from an industrial survey on architecture diagram construction [21]. We map these recommendations to the five quality criteria and derive 15 guideline-level checks, as shown in Table 1. This mapping provides the conceptual basis for the evaluation framework used in our empirical study.

Figure 1 shows the diagram of the Magda.io software architecture used in our study. As a brief example of the criteria in practice, the diagram shows a lack of completeness, especially in the external actors coverage guideline. This problem occurs because end users are not represented and external data sources are only implied by connector components rather than shown as explicit actors. It also shows limitations in level of detail, since several arrows indicate direction but do not specify whether interactions are synchronous calls, events, batch jobs, or data updates. At the same time, the diagram performs well in consistency for notation definition and uniformity, because it uses a legend to distinguish complete, in-development, and planned components and applies this visual convention across the diagram. This example illustrates how the same diagram can satisfy some guideline-level checks while failing others, motivating our use of fine-grained evaluation units rather than only broad criterion-level judgments.

By using guideline-level checks, we make the criteria more explicit and auditable. This detailed evaluation is important because holistic criterion-level judgments may hide partial failures, especially when a diagram satisfies some aspects of a criterion but fails others. In our study, these guidelines serve as the unit of evaluation for both LLM-based assessment and human reference judgments.

2.3 LLMs In Software Architecture And Diagram Evaluation

Large Language Models (LLMs) have been widely applied to software engineering tasks, including code generation, testing, explanation, documentation, and refactoring support [2, 3, 15, 16, 22, 24]. Recent research [1, 6, 11, 14, 26] has also explored their use in software architecture, including collaborative architecting, architectural design decision generation, architecture decision records, and generation of architectural artifacts. These studies suggest that LLMs can reason about high-level architectural concepts and produce useful architectural text. However, most existing work focuses on textual artifacts [1, 11, 14], such as requirements, design decisions, documentation, or generated descriptions.

Architecture diagrams pose a different challenge [10, 20]. They are visual artifacts whose meaning often depends on layout, grouping, labels, symbols, and spatial relationships. Even when a textual source representation is available, such as SVG or draw.io XML, the representation may not preserve the same information that a

human perceives when inspecting the rendered diagram. Therefore, evaluating architecture diagrams with LLMs requires understanding not only whether models can produce plausible architectural feedback, but also whether their scores align with human judgment and whether the input representation affects the usefulness of their reasoning.

The closest work to ours, by Oliveira and Mendonça [10], evaluated diagrams from four systems using a single LLM and five quality criteria. Their study provides initial evidence that LLMs can identify relevant issues in architecture diagrams, but also reports limited agreement with researcher judgment. Our work extends this line of research in three ways. First, we evaluate two state-of-the-art multimodal LLMs instead of a single model. Second, we introduce a guideline-level scoring framework and compare LLM-generated scores against human reference scores, allowing a more structured analysis of reliability. Third, we investigate the role of diagram representation by comparing rendered PNG diagrams with textual source representations, including SVG and draw.io XML. By combining these dimensions, our study provides a controlled and fine-grained evaluation of the strengths and limitations of LLM-based architecture diagram assessment.

3 Experimental Design

This section describes the design of our empirical study. We first present the study goal and research questions. We then organize the remaining subsections according to the four stages represented in Figure 2: data collection, LLM-based diagram evaluation, human architectural evaluation, and data analysis. This structure connects the methodological description to the experimental pipeline used in the study. Figure 2 uses a simple pipeline notation: labeled boxes denote the four sequential stages, and arrows denote the flow of artifacts (diagrams, LLM outputs, and human judgments) between stages, each detailed in the corresponding subsection.

3.1 Study Goal And Research Questions

The goal of this study is **to evaluate the reliability of LLM-based assessments of microservice architecture diagrams**. We analyze reliability by comparing LLM-generated scores with human reference scores and by examining how software engineers judge the reasoning produced by LLMs.

Our study addresses the following research questions:

RQ$_1$. To what extent do LLM-generated scores align with human reference scores when assessing microservice architecture diagrams?

RQ$_2$. Which quality criteria show stronger or weaker agreement between LLM-generated scores and human reference scores?

RQ$_3$. Do software engineers prefer the reasoning generated from image-based or text-based diagram representations?

RQ$_1$ evaluates the overall score-level reliability of LLM-based diagram assessment, indicating whether LLMs can produce scores aligned with human judgment. **RQ$_2$** refines this analysis by examining whether reliability varies across quality criteria, helping identify which aspects of diagram quality are more or less suitable for LLM-supported assessment. **RQ$_3$** complements the score-based

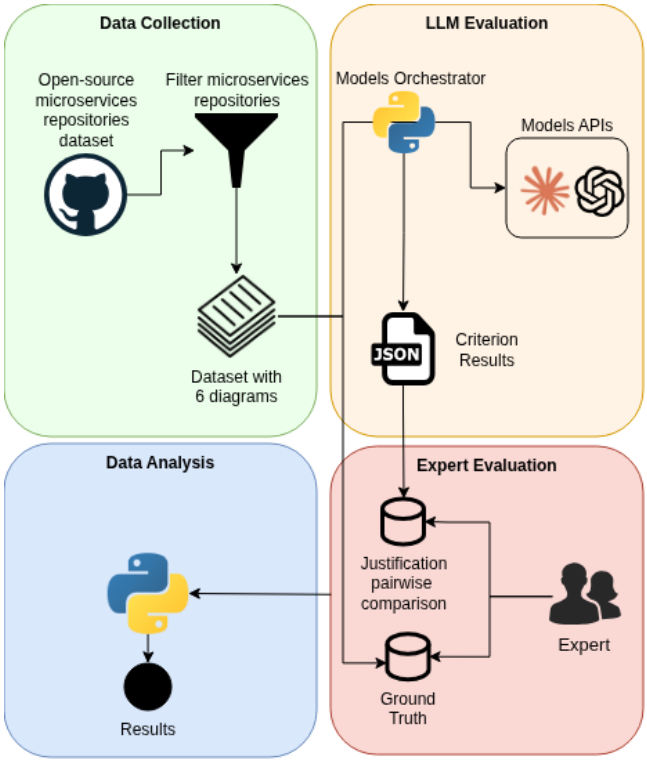

Figure 2: Study design overview

analysis by focusing on the perceived usefulness of the generated reasoning, comparing whether experts prefer explanations produced from image-based or text-based representations. Together, these questions characterize LLM-based architecture diagram assessment in terms of both agreement with human judgment and usefulness for human reviewers, clarifying whether LLMs can support architecture review as first-pass assessors and where human validation remains necessary.

3.2 Data Collection And Study Objects

We conduct the study with six real-world projects based on microservice architectures. The projects were selected from a curated dataset of open-source microservice systems [5]. We selected systems that (i) are based on a microservice architecture, (ii) provide an architecture diagram together with an available textual source representation (SVG or draw.io XML), and (iii) include repository-level documentation, allowing us to evaluate the same architectural information across formats. Table 2 summarizes the selected systems and their descriptions. The repository URLs and collected artifacts are available in our replication package [8].

For each project, we collected three artifacts: (i) a rendered architecture diagram in PNG format, (ii) the textual source representation used to generate or document the same diagram, and (iii) the official textual description available in the project repository. The textual source representations include SVG and draw.io XML files. Three diagrams use SVG and three use draw.io XML. By collecting the rendered image and its corresponding source representation, we

Table 2: Subject Open-Source Systems.

Project	Description
Appwrite	Open-source backend platform that provides APIs and infrastructure services for web, mobile, and server applications.
GeoServer Cloud	Cloud-native distribution of GeoServer organized as a set of dockerized microservices.
Magda	Open-source data catalog platform for cataloging, enriching, searching, and managing data assets.
Research Modifiability Pattern Experiment	Research repository containing service-based WebShop systems used to study architectural modifiability.
Sentry	Self-hosted distribution of Sentry, an error monitoring platform composed of multiple supporting services.
Spryker	Demo deployment of a Spryker-based commerce platform, including a containerized application stack.

preserve the architectural content and vary the format provided to the LLM. This design allows us to analyze whether the representation of the same diagram influences LLM-based evaluation.

The collected artifacts are evaluated according to the guideline-level framework described in Section 2.2. For each guideline, both LLMs and human participants assign a score on a 4-point scale: 4 = Excellent, 3 = Good, 2 = Fair, and 1 = Poor. We use this scale to avoid a neutral midpoint and to keep the scoring task consistent across LLM evaluations and human reference scores. Each LLM evaluation also includes a textual justification for the assigned score, enabling us to analyze not only score agreement but also the perceived usefulness of the generated reasoning.

3.3 LLM-Based Diagram Evaluation

The LLM evaluation stage consists of evaluating the collected architecture diagrams with LLMs. We evaluate two state-of-the-art multimodal models: GPT-5, accessed through the OpenAI API, and Claude Sonnet 4.6, accessed through the Anthropic API. We chose GPT-5 and Claude Sonnet 4.6 because, at the time of the study, they were among the most capable multimodal LLMs for visual reasoning.

Each model evaluates the diagrams under two representation conditions. In the image-based condition, the model receives the rendered PNG diagram. In the text-based condition, the model receives the textual source representation of the same diagram, either SVG or draw.io XML. In both conditions, the model also receives the official textual description of the project collected from the repository. Thus, the two conditions differ only in the representation of the diagram, allowing us to compare whether image-based and text-based inputs lead to different scores and justifications.

We use a criterion-specific prompting protocol. Each prompt includes: (i) the evaluation task, (ii) the definition of the target quality criterion, (iii) the guideline-level checks associated with that criterion, (iv) the architecture diagram in the corresponding representation, and (v) the official textual description of the system. The prompt requires the model to return strict JSON with one score and one textual justification for each guideline. Figure 3 shows an example prompt for the completeness criterion.

We evaluate every combination of project, model, diagram representation, and quality criterion. The study includes six projects, two models, two diagram representations, and five quality criteria, resulting in 120 LLM calls ($6 \times 2 \times 2 \times 5$). Since the evaluation framework contains 15 guideline-level checks, these calls produce 360 guideline-level LLM scores ($6 \times 2 \times 2 \times 15$). This design allows direct comparison between image-based and text-based evaluations for the same project, model, criterion, and guideline.

We do not use fine-tuning, few-shot examples, or explicit chain-of-thought instructions. Our goal is to evaluate the out-of-the-box capability of current LLMs in a practical architecture review setting. The models are instructed to provide concise, evidence-based justifications, but not to expose intermediate reasoning steps. We conducted two trials to inspect output stability and used the last trial in the analysis. A systematic quantification of run-to-run variability across trials is left for future work.

3.4 Human Architectural Evaluation Audit

The human evaluation stage consists of a human audit designed to evaluate LLM outputs against software engineers' judgments. The audit has two complementary purposes. First, it provides human reference scores for the guideline-level assessment of each architecture diagram. Second, it captures expert preferences between LLM-generated justifications produced from image-based and text-based diagram representations.

The audit was conducted with 16 software engineers, who reported their academic background, software development experience, architecture analysis experience, and technical expertise. Half held an undergraduate degree, 31% held a master's degree, and 19% held a PhD in computer science. Most participants had more than five years of software development experience (56%) and medium (56%) or extensive (25%) architecture analysis experience. All participants selected Software Architecture as an area of expertise, followed by service/API design (69%), distributed systems (56%), and microservices (56%).

To construct human reference scores, each participant evaluated one assigned project by inspecting its architecture diagram and textual description and scoring the 15 guideline-level checks from 1 to 4, similar to the procedure done with the LLMs. Although 16 participants completed the audit, we selected 12 responses to maintain a balanced design with two evaluators per diagram. When three participants evaluated the same diagram, we selected the two with higher software development experience. This procedure avoids overrepresenting diagrams with more respondents and yields a balanced, expert-informed reference set rather than an absolute ground truth.

Participants also compared pairs of LLM-generated justifications. Each pair referred to the same project, criterion, and guideline, but differed in the diagram representation used by the LLM: PNG or textual source. Table 3 reports the number of comparisons by diagram. Participants saw the score and justification for both evaluations and selected the better one, or marked a tie. We retained all 16

Prompt: Architecture Completeness Evaluation

Context. You are evaluating the **COMPLETENESS** of a software architecture diagram for a microservices-based system, based on the given diagram and its textual description.
Diagram. [Insert diagram here: attached PNG image or inline SVG/XML content.]
Description. [Insert official project description here.]
Criterion: Completeness. Completeness is about whether the diagram includes the main architectural elements and interactions needed to understand the system as a whole.
Evaluate the following guidelines:

(1) **Coverage of Main Components** — Are all major components represented in the diagram?
(2) **External Actors Coverage** — Are relevant external actors shown when they are important to understanding how the system works end-to-end?
(3) **Main Interactions Coverage** — Are the primary interactions between components represented so that the overall behavior of the system can be understood?

Output Format (IMPORTANT). Return strict JSON only, without markdown, prose, or code fences:

```
{
  "criterion": "completeness",
  "guidelines": [
    {
      "id": "coverage_of_main_components",
      "score": "4 (Excellent) | 3 (Good) | 2 (Fair) | 1 (Poor)",
      "justification": "<concise, evidence-based explanation>"
    }
  ]
}
```

Figure 3: Criterion-specific prompt example for Completeness.

Table 3: Pairwise preference comparisons by diagram.

Diagram	Comparisons
magda-io	74
appwrite	42
spryker	39
research-modifiability-pattern-experiment	35
geoserver-cloud	32
sentry	31
Total	253

participants for this preference analysis because each comparison contributes directly to the preference distribution.

3.5 Data Analysis And Operationalization

The final stage of our study analyzes LLM reliability according to the three research questions. We use score-agreement metrics for RQ_1 and RQ_2 and preference-based analysis for RQ_3. Across the analyses, we focus on diagram representation and quality criteria. We pool the outputs of both LLMs because our goal is not to compare models, but to evaluate whether LLM-based assessments align with human judgment and whether input representation affects the evaluation.

To answer RQ_1, we compare LLM-generated scores with human reference scores. We first convert the original 4-point scale into binary labels: scores 1 and 2 represent low-quality assessments, while scores 3 and 4 represent high-quality assessments. This threshold reflects the practical goal of first-pass review, distinguishing diagrams or guidelines that require closer inspection from those that are broadly acceptable. When the two human evaluators fall on opposite sides of this boundary for the same item, the researchers determine the final reference label. We then compute precision, recall, and F1-score for each representation. To preserve the ordinal nature of the original scores, we also compute quadratic weighted Cohen's Kappa [9], which penalizes larger disagreements more strongly than smaller ones. Finally, we use bootstrap resampling to estimate the 95% confidence intervals for the representation differences.

To answer RQ_2, we apply the same agreement metrics at the criterion level. For each quality criterion, we compute precision, recall, and F1-score using the binary labels, and quadratic weighted Cohen's Kappa using the original 1–4 scores. This analysis allows us to identify which criteria show stronger or weaker alignment between LLM-generated scores and human reference scores. We also compare image-based and text-based representations within each criterion to examine whether representation effects are concentrated in specific aspects of diagram quality.

To answer RQ_3, we analyze expert preferences between pairs of LLM-generated justifications. Each comparison presents two evaluations for the same project, criterion, and guideline: one generated from the PNG representation and one generated from the textual source representation. We code each decision as a preference for image, a preference for text, or a tie. We report the overall preference distribution and the distribution by criterion. For inferential analysis, we exclude ties and test whether experts choose image and text with equal probability among decided comparisons. We use a chi-square goodness-of-fit test and report a Wilson 95% confidence interval for the image preference share.

4 Results

This section presents our results by research question. Section 4.1 (RQ_1) evaluates LLM score alignment with human references across diagram representations. Section 4.2 (RQ_2) examines agreement variation across quality criteria. Section 4.3 (RQ_3) analyzes expert preferences for LLM-generated justifications.

Table 4: Agreement metrics by diagram representation.

Representation	Precision	Recall	F1	κ
Image	0.752	0.802	0.776	0.387
Textual	0.706	0.792	0.747	0.225
Pooled	0.728	0.797	0.761	0.308

4.1 RQ_1: Agreement By Representation

RQ_1 evaluates the overall agreement between LLM-generated scores and human reference scores. Figure 4 presents the distribution of precision, recall, F1, and quadratic weighted Kappa by diagram representation (image and textual) at the diagram level. Table 4 summarizes the pooled values for each representation and for the complete set of evaluations.

In general, LLM evaluations achieve 0.728 precision, 0.797 recall, and 0.761 F1 when compared with human reference labels. The higher recall indicates that LLMs identify human-positive cases more often than they avoid false positives. In practical terms, this suggests that LLM-based evaluations tend to be more permissive than conservative: they are more likely to classify a guideline as satisfactory when humans also do so, but they may also assign positive scores to some cases that humans judge less favorably.

The image-based representation shows a small descriptive advantage over the textual representation in all agreement metrics. Image-based evaluations reach 0.752 precision, 0.802 recall, and 0.776 F1, while text-based evaluations reach 0.706 precision, 0.792 recall, and 0.747 F1. The largest binary difference appears in precision, suggesting that image-based evaluations produce fewer false positives than text-based evaluations. However, bootstrap analysis does not support a robust representation effect for precision, recall, or F1. Therefore, the observed binary advantage of image-based inputs should be interpreted as a descriptive trend rather than as conclusive evidence that representation changes score agreement.

The ordinal agreement results follow the same direction. The pooled quadratic weighted Kappa reaches $\kappa = 0.308$, indicating fair agreement on the original 1–4 scale. Image-based evaluations reach $\kappa = 0.387$, while text-based evaluations reach $\kappa = 0.225$. This difference suggests that image-based inputs may better preserve the graded structure of human scores. However, as with the binary metrics, bootstrap analysis does not provide conclusive evidence of a robust representation effect. Thus, image-based inputs appear more favorable descriptively, but the score-level evidence is not strong enough to claim a statistically robust representation advantage.

Answer to RQ_1. LLM evaluations show reasonable agreement with human reference scores, with pooled F1 equal to 0.761 and pooled quadratic weighted Kappa equal to 0.308. Image-based inputs show a consistent descriptive advantage over textual inputs, especially in precision and ordinal agreement, but the bootstrap analysis does not support a robust representation effect for score agreement.

Table 5: Criterion-level agreement between LLM-generated scores and human reference scores.

Criterion	F1	κ
Accuracy	0.822	0.257
Clarity	0.697	0.336
Completeness	0.857	0.396
Consistency	0.713	0.341
Level of Detail	0.738	0.259

4.2 RQ_2: Agreement By Criterion

RQ_2 examines whether agreement between LLM-generated scores and human reference scores varies between quality criteria. Figure 5 presents the distribution of F1, precision, and recall by criterion after pooling representations. Table 5 summarizes the F1-score and quadratic weighted Kappa for each criterion.

The agreement varies more clearly between criteria than between diagram representations. Completeness achieves the highest F1-score, reaching 0.857, followed by accuracy with 0.822. Level of detail reaches 0.738, consistency reaches 0.713, and clarity shows the lowest F1-score, reaching 0.697. This ranking suggests that LLM evaluations align better with human reference labels for criteria related to the presence of architectural information, such as components and interactions, than for criteria that depend more strongly on subjective interpretation, such as clarity.

The separation between precision and recall reveals an important criterion-level pattern. Recall is especially high for completeness and accuracy, reaching 0.975 and 0.938, respectively. This indicates that LLM evaluations rarely miss cases that humans classify as positive for these criteria. In contrast, clarity has the lowest recall, reaching 0.679, suggesting that LLMs more often fail to identify diagrams or guidelines that humans consider satisfactory for this criterion. Precision varies less sharply across criteria, ranging from 0.705 to 0.765, which indicates that the main criterion-level differences are more associated with missed positives than with false positives.

The ordinal analysis provides a complementary interpretation. Completeness also shows the strongest quadratic weighted Kappa, with $\kappa = 0.396$. Consistency and clarity follow with $\kappa = 0.341$ and $\kappa = 0.336$, respectively. Accuracy and level of detail show lower ordinal agreement, with $\kappa = 0.257$ and $\kappa = 0.259$. The contrast between accuracy's high F1-score and lower Kappa suggests that LLMs often identify the correct binary category for this criterion, but still diverge from humans on the exact score in the original 1–4 scale.

When comparing representations within each criterion, image-based input leads descriptively in four of the five criteria: accuracy, clarity, completeness, and level of detail. Text-based input leads only in consistency, and the difference is small. The largest descriptive image advantages occur for accuracy and clarity, while level of detail remains nearly tied between representations. However, bootstrap analyses do not support a robust representation advantage for any individual criterion. Therefore, criterion type explains agreement differences more clearly than input representation.

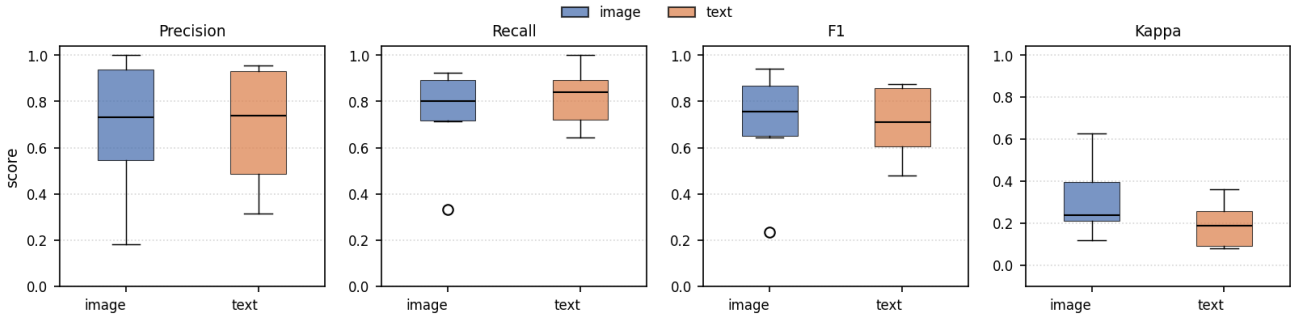

Figure 4: Precision, recall, F1, and quadratic weighted Kappa by representation at the diagram level.

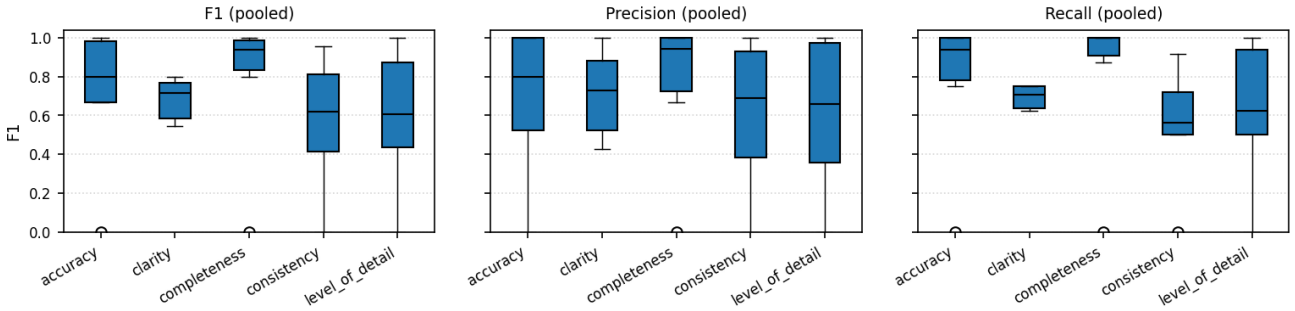

Figure 5: F1, precision, and recall by criterion after pooling representations at the diagram level.

Answer to RQ$_2$. Agreement depends more on the quality criterion than on diagram representation. Completeness aligns best with human scores, while clarity has the weakest binary performance. LLMs perform better on information-presence criteria than on subjective ones. Image inputs lead descriptively in most criteria, but without a robust representation advantage.

4.3 RQ$_3$: Expert Preference For Reasoning

RQ$_3$ evaluates whether experts prefer LLM-generated reasoning produced from image-based or text-based diagram representations. Unlike RQ$_1$ and RQ$_2$, which focus on score agreement, this analysis captures the perceived usefulness of the justifications produced by the models. Table 6 summarizes the overall preference distribution.

Experts preferred image-based reasoning in 113 comparisons and text-based reasoning in 73 comparisons. They marked a tie in 67 comparisons, corresponding to 26.4% of all cases. This tie rate indicates that experts often considered both justifications similarly useful, or similarly limited, for supporting diagram review. However, among decided comparisons, image-based reasoning accounts for 60.8% of preferences. The Wilson 95% confidence interval for the image preference share is [53.6%, 67.5%], and the chi-square goodness-of-fit test rejects the equal-preference baseline ($p = 0.0034$). Therefore, while many comparisons remain close, experts significantly prefer image-based reasoning when they identify a better justification.

Table 6: Overall expert preference distribution.

Preference	Count	Percentage
Image	113	44.7%
Text	73	28.9%
Tie	67	26.4%
Total	253	100.0%

Table 7 reports expert preference by criterion among decided comparisons. Image-based reasoning receives more preferences than text-based reasoning for accuracy, clarity, completeness, and consistency. Level of detail is the only criterion where text-based reasoning has a small numerical advantage. The strongest image preference appears for consistency and completeness, while accuracy shows a more balanced distribution. This pattern suggests that the overall preference for image-based reasoning is not restricted to a single criterion, although its magnitude varies across criteria.

Table 7: Expert preference by criterion among decided comparisons.

Criterion	Image	Text	Image share
Accuracy	18	15	54.5%
Clarity	29	18	61.7%
Completeness	23	11	67.6%
Consistency	28	13	68.3%
Level of Detail	15	16	48.4%

Answer to RQ_3. Experts prefer image-based reasoning in pairwise comparisons. Among decided comparisons, image-based reasoning receives 60.8% of preferences, with a Wilson 95% confidence interval of [53.6%, 67.5%], and the chi-square goodness-of-fit test rejects the equal-preference baseline. The preference appears across most criteria, while the 26.4% tie rate shows that many image-based and text-based justifications remain comparable.

5 Discussion

This section organizes into six topics the implications of our findings for LLM-supported architecture diagram review. We synthesize the evidence around broader discussion points: the role of LLMs as structured review assistants, the limits of their reliability across quality criteria, the different effects of diagram representation on scores and explanations, and the need for human-in-the-loop workflows. We also discuss how guideline-level evaluation can make LLM-based feedback more auditable and actionable for architects, developers, managers, tool builders, and researchers.

LLMs can support architecture diagram review as structured assistants, but not as authoritative evaluators. Our results suggest that LLMs can help architecture review mainly by supporting preliminary inspection. The pooled F1-score of 0.761 shows that LLM-generated scores often align with human reference labels at a coarse level, while the fair ordinal agreement ($\kappa = 0.308$) shows that they still diverge from humans when finer score distinctions are considered. This distinction is important for practice: LLMs can help architects and developers identify diagrams or guidelines that deserve attention, but their assessments should not be interpreted as final judgments about architectural quality. In this sense, LLM-based evaluation is better suited for review triage than for review certification.

LLMs are more reliable for checklist-like inspection than for subjective diagram interpretation. The criterion-level results indicate that LLMs are more useful when diagram quality can be assessed through explicit evidence. Completeness achieves the highest F1-score (0.857) and the strongest ordinal agreement ($\kappa = 0.396$), while accuracy also shows high binary agreement (F1 = 0.822). These criteria are closely related to checking whether expected components, interactions, actors, and technologies are represented. In contrast, clarity shows the lowest F1-score (0.697) and recall (0.679), and level of detail has one of the lowest ordinal agreements ($\kappa = 0.259$). This suggests that LLMs are more reliable for checklist-like inspection than for criteria that depend on subjective interpretation, visual communication, and audience expectations. For practitioners, this means that LLM feedback is more actionable when it points to missing or inconsistent architectural information. For researchers, it reinforces the importance of evaluating LLMs by quality criterion rather than relying only on aggregate performance.

Score agreement and reasoning usefulness capture different dimensions of LLM reliability. The comparison between image-based and text-based inputs shows that these two dimensions should be analyzed separately. Image-based inputs show a consistent descriptive advantage in precision (0.752 vs. 0.706), F1-score (0.776 vs. 0.747), and ordinal agreement ($\kappa = 0.387$ vs. $\kappa = 0.225$), but these differences are not statistically robust. However, experts prefer image-based reasoning in 60.8% of decided comparisons, with a significant preference over the equal-preference baseline ($p = 0.0034$). This means that a representation may not substantially improve the final score while still improving the usefulness of the explanation. This distinction is consistent with prior findings that LLMs can reach high agreement with human decisions while relying on shallow reasoning [4]. For tool builders, this suggests that diagram-review tools should not optimize only for score prediction; they should also evaluate whether explanations are useful, grounded, and actionable for human reviewers.

Rendered diagrams should be treated as primary inputs when explanations are intended for human review. The preference for image-based reasoning highlights the role of visual information in architecture understanding. Architecture diagrams communicate through layout, spatial grouping, visual hierarchy, symbols, labels, and proximity between elements, aspects that are central to graphical software design communication [18] and to architecture documentation practice [21, 25]. Textual representations such as SVG or draw.io XML may expose labels and internal structure, but they can obscure how the diagram is perceived as a visual artifact. Therefore, tools that evaluate diagrams with LLMs should treat the rendered diagram as a primary input when the goal is to support human review. Textual representations remain useful as complementary inputs, especially for recovering labels or hidden metadata, but they should not replace the visual artifact that stakeholders actually inspect.

Guideline-level evaluation makes LLM-supported review more auditable and actionable. Previous work has shown that architecture diagrams are often informal, heterogeneous, and partially aligned with the implemented system [20], which makes holistic judgments difficult to interpret. By decomposing quality into guideline-level checks, the evaluation becomes more transparent: architects can inspect whether the model disagrees on component coverage, interaction matching, notation consistency, reading flow, or level of detail. This decomposition is useful for practitioners because it turns a vague assessment such as "the diagram is unclear" into concrete review items. It is also useful for researchers because it exposes where LLMs succeed or fail instead of hiding these differences behind a single overall score.

LLM-supported architecture diagram review should be designed as a human-in-the-loop process. A practical workflow would provide the LLM with the rendered diagram, its textual description, and explicit guideline-level checks; the model would then assign preliminary scores, flag low-scoring or uncertain items, and

generate concise evidence-based comments. Architects, developers, or technical leads would validate the comments, reject misleading feedback, and decide whether the diagram satisfies the communication needs of its audience. This workflow preserves the scalability of LLMs while keeping responsibility for architectural interpretation with humans, which is especially important for criteria such as clarity and level of detail where human judgment remains central.

6 Threats To Validity

We discuss threats to validity following guidelines for experimentation in software engineering [29].

Internal validity. Internal validity concerns whether the observed effects can be attributed to the studied factors [29]. One threat is that the evaluated systems are public open-source projects, and the LLMs may have encountered project documentation, repository content, or related discussions during training. This creates a potential training-data bias, since model outputs may reflect memorized or previously learned information rather than only the diagram representation provided in the prompt. We cannot fully rule out this threat because the training data of proprietary LLMs is not publicly available. Therefore, we interpret our results as evidence about LLM behavior when evaluating public architecture documentation, not as a pure test of diagram understanding from unseen systems. Another internal threat concerns the human audit process. Participants evaluated multiple items, which may introduce learning or fatigue effects, especially in the pairwise comparison phase. To reduce this risk for the reference scores, each participant evaluated only one assigned diagram.

Construct validity. Construct validity concerns whether the measures capture the intended concepts [29]. Diagram quality is multidimensional and subjective, and our five criteria may not capture all aspects relevant to architecture understanding. To mitigate this threat, we decomposed each criterion into guideline-level checks, making the evaluation more explicit and auditable. Another threat comes from binarizing the original 1–4 scale. Mapping scores 1 and 2 to low quality and scores 3 and 4 to high quality enables precision, recall, and F1 analysis, but may hide meaningful differences between adjacent categories. For this reason, we also report quadratic weighted Kappa on the original ordinal scale, and we leave a sensitivity analysis of alternative thresholds to future work. Finally, expert preference in RQ3 captures perceived reasoning quality rather than factual correctness, so we interpret preference as perceived usefulness rather than as a direct correctness measure.

External validity. External validity concerns the generalizability of the results [29]. Our study uses six open-source microservice systems and their available architecture diagrams. The results may not generalize to other architectural styles, industrial diagrams, proprietary documentation, or diagrams with different notation conventions. The textual representation also depends on the available source format, which in our study includes SVG and draw.io XML. Other textual diagram formats, such as PlantUML, Mermaid, or manually written architecture descriptions, may lead to different model behavior. In addition, we evaluated two state-of-the-art LLMs, and the absolute performance values may change as newer models become available.

Conclusion validity. Conclusion validity concerns the reliability of the statistical analysis and the strength of the conclusions drawn [29]. The score-agreement analyses use 360 guideline-level LLM evaluations, but these observations are not fully independent because they come from repeated evaluations of the same diagrams, criteria, and models. The 253 pairwise preference comparisons also include multiple responses from the same participants and across the same diagrams. We therefore interpret representation effects with caution, especially when bootstrap analyses do not support robust differences. To reduce the risk of overclaiming, we distinguish descriptive trends from robust findings and avoid treating non-significant representation differences as conclusive. We report agreement pooled across the two LLMs and evaluators. Per-model and inter-rater breakdowns, along with hierarchical models for repeated measures, are left for future work.

7 Conclusion And Future Work

This paper investigated the reliability of LLMs for assessing microservice architecture diagrams using a guideline-based evaluation framework and a human audit. We evaluated two LLMs across six microservice systems, two diagram representations, five quality criteria, and 15 guideline-level checks.

Our results show that LLM-generated scores achieve reasonable agreement with human reference scores, with stronger alignment for completeness and weaker alignment for clarity. Image-based inputs show a consistent descriptive advantage over textual inputs, but this effect is not statistically robust for score agreement. However, experts significantly prefer image-based reasoning in pairwise comparisons, suggesting that visual representations may produce justifications that are more useful or convincing to human reviewers.

Overall, these findings indicate that LLMs can support architecture diagram review as first-pass assessors, helping identify potential quality issues and prioritize human inspection. Nevertheless, their evaluations should remain part of a human-in-the-loop workflow rather than replacing expert judgment.

Future work should extend this evaluation to additional systems, architectural styles, and diagram notations to assess whether the observed results generalize beyond open-source microservice projects. Further studies should investigate how richer project context, different prompting strategies, or fine-tuning can improve score alignment and reasoning quality across diagram types and quality criteria. Finally, future work should examine interactive review workflows in which LLMs act as first-pass assessors and architects validate, refine, or reject the generated evaluations during real documentation review tasks.

Artifact Availability

We make the research artifacts associated with this study available in our replication package [8].

Acknowledgements

This research was partially supported by the Brazilian funding agencies: CAPES, CNPq (Grants 446729/2024-8 and 406089/2025-6), and FAPEMIG (Grant APQ-01488-24). ChatGPT was used to refine wording, improve clarity, and identify relevant references.

References

[1] Aakash Ahmad, Muhammad Waseem, Peng Liang, Mahdi Fahmideh, Mst Shamima Aktar, and Tommi Mikkonen. 2023. Towards Human-Bot Collaborative Software Architecting with ChatGPT. In *Proc. of the 27th International Conference on Evaluation and Assessment in Software Engineering (EASE)* (Oulu, Finland). 279–285.

[2] Gabriel Amaral, Henrique Gomes, Eduardo Figueiredo, Carla Bezerra, and Larissa Rocha. 2025. Improving JavaScript Test Quality with Large Language Models: Lessons from Test Smell Refactoring. In *Brazilian Symposium on Software Engineering* (Recife/PE). 776–782.

[3] Larisse Amorim, Ivandeclei Costa, Leticia Alves, and Eduardo Figueiredo. 2025. Bad Smell Detection using Google Gemini. In *49th IEEE Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 1637–1642.

[4] Larisse Amorim, Caique Fortunato, Gustavo Vale, and Eduardo Figueiredo. 2026. High Agreement, Shallow Reasoning: A Mixed-Method Study of LLMs in Refactoring Reviews. In *22nd International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE)*.

[5] Dario Amoroso d'Aragona et al. 2024. A Dataset of Microservices-based Open-Source Projects. In *International Conference on Mining Software Repositories (MSR)*. 504–509.

[6] Shrikara Arun, Meghana Tedla, and Karthik Vaidhyanathan. 2025. LLMs for generation of architectural components: an exploratory empirical study in the serverless world. In *2025 IEEE 22nd International Conference on Software Architecture (ICSA)*. IEEE, 25–36.

[7] Len Bass, Paul Clements, and Rick Kazman. 2012. *Software Architecture in Practice* (3 ed.). Addison-Wesley.

[8] João Luiz Cerqueira, Mateus Dutra, Eduardo Figueiredo, and Gustavo Andrade Vale. 2026. Artifact and Replication Package: How Reliable Are LLM Evaluations of Microservice Architecture Diagrams? (SBCARS 2026). https://doi.org/10.5281/zenodo.21811719

[9] Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin* 70, 4 (1968), 213.

[10] Glauber Queiroz de Oliveira and Nabor C Mendonça. 2025. LLM-Based Quality Assessment of Software Architecture Diagrams: A Preliminary Study with Four Open-Source Projects. In *European Conference on Software Architecture (ECSA)*. Springer, 116–123.

[11] Rudra Dhar, Karthik Vaidhyanathan, and Vasudeva Varma. 2024. Can LLMs Generate Architectural Design Decisions? - An Exploratory Empirical Study. In *2024 IEEE 21st International Conference on Software Architecture (ICSA)*. 79–89.

[12] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2018. Migrating towards Microservice Architectures: An Industrial Survey. In *2018 IEEE International Conference on Software Architecture (ICSA)*. IEEE.

[13] Mateus Dutra, Denis Pinheiro, Johnatan Oliveira, and Eduardo Figueiredo. 2025. From Detection to Refactoring of Microservice Bad Smells: A Systematic Literature Review. *Journal of Software Engineering Research and Development (JSERD)* 13, 2 (2025), 13:172–13:183.

[14] Matteo Esposito, Xiaozhou Li, Sergio Moreschini, Noman Ahmad, Tomas Cerny, Karthik Vaidhyanathan, Valentina Lenarduzzi, and Davide Taibi. 2026. Generative AI for software architecture. Applications, challenges, and future directions. *Journal of Systems and Software* 231 (2026).

[15] 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)*. 31–53.

[16] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024).

[17] Jasmin Jahić and Ashkan Sami. 2024. State of practice: Llms in software engineering and software architecture. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 311–318.

[18] Rodi Jolak, Maxime Savary-Leblanc, Manuela Dalibor, Andreas Wortmann, Regina Hebig, Juraj Vincur, Ivan Polášek, Xavier Pallec, and Sébastien Gérard. 2020. Software engineering whispers: The effect of textual vs. graphical software design descriptions on software design communication. *Empirical Software Engineering* 25 (2020).

[19] James Lewis and Martin Fowler. 2014. Microservices: A Definition of This New Architectural Term. https://martinfowler.com/articles/microservices.html. Accessed: 2026-06-25.

[20] Sofia Migliorini, Roberto Verdecchia, Ivano Malavolta, Patricia Lago, and Enrico Vicario. 2024. Architectural views: the state of practice in open-source software projects. In *European Conference on Software Architecture (ECSA)*. Springer, 396–415.

[21] Jalves Nicacio and Fabio Petrillo. 2022. An approach to build consistent software architecture diagrams using devops system descriptors. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings* (Montreal, Quebec, Canada) *(MODELS '22)*. ACM, New York, NY, USA, 312–321.

[22] Henrique Nunes, Eduardo Figueiredo, Larissa Rocha Soares, Sarah Nadi, Fischer Ferreira, and Geanderson Santos. 2025. Evaluating the Effectiveness of LLMs in Fixing Maintainability Issues in Real-World Projects. In *32nd IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 669–680.

[23] Flavio Oquendo, Jair Leite, and Thais Batista. 2016. *Software Architecture in Action*. Springer.

[24] Ipek Ozkaya. 2023. Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. *IEEE Software* 40, 3 (2023), 4–8.

[25] N. Rozanski and E. Woods. 2012. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley.

[26] Mohamed Soliman and Jan Keim. 2025. Do Large Language Models Contain Software Architectural Knowledge? : An Exploratory Case Study with GPT. In *2025 IEEE 22nd International Conference on Software Architecture (ICSA)*. 13–24.

[27] Markos Viggiato, Ricardo Terra, Henrique Rocha, Marco Tulio Valente, and Eduardo Figueiredo. 2018. Microservices in Practice: A Survey Study. In *6th Workshop on Software Visualization, Evolution and Maintenance (VEM), co-allocated with CBSoft*.

[28] Muhammad Waseem, Peng Liang, and Mojtaba Shahin. 2020. A systematic mapping study on microservices architecture in devops. *Journal of Systems and Software* 170 (2020), 110798.

[29] Claes Wohlin, Per Runeson, Martin Höst, Magnus Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering* (1st ed.). Springer Science & Business Media, New York.

[30] He Zhang, Shanshan Li, Zijia Jia, Chenxing Zhong, and Cheng Zhang. 2019. Microservice Architecture in Reality: An Industrial Inquiry. *2019 IEEE International Conference on Software Architecture (ICSA)* (2019), 51–60.