

Layer-Wise Fidelity Evaluation of Multi-Layer Agentic AI Pipelines: MCP, A2A, and A2UI

Ömer Furkan Tercan
University of Helsinki
omer.tercan@helsinki.fi

Abstract—Multi-layer agentic pipelines—in which an LLM agent retrieves context via a standardised tool protocol, synthesises a response, and renders it through a structured user interface—are increasingly deployed in practice, yet remain poorly understood from an evaluation standpoint. Existing work evaluates such systems at the pipeline level, scoring only the final output against a reference; no framework characterises where information fidelity is gained or lost between architectural layers. We propose *layer-wise fidelity scoring*: BERTScore Precision, Recall, and F1 computed independently at R_0 (raw MCP tool output), R_1 (A2A agent synthesis), and R_2 (A2UI structured payload), each against the same ground-truth reference answer. We instantiate this protocol on a software documentation Q&A task using the `httpx` Python library, building a minimal end-to-end prototype comprising an MCP retrieval server, a LangGraph A2A orchestrator, and a Streamlit-based rendering layer that simulates A2UI structured output. Experiments across 30 test queries show that A2A synthesis significantly improves BERTScore F1 ($\Delta_1 = +0.038$, $p < 0.001$) by raising Precision while Recall increases slightly, confirming that synthesis improves content selectivity without sacrificing coverage. The A2UI formatting step leaves F1 unchanged ($\Delta_2 = +0.001$, $p = 0.72$, not significant) with a small but statistically significant Recall reduction ($\Delta_2^R = -0.012$, $p = 0.016$). These layer-level dynamics are invisible to aggregate end-to-end scoring and only emerge through layer-wise analysis. The protocol is designed to be task-agnostic and reusable, though this study validates it on a single domain; we release the annotated test set and scoring harness to support future benchmarking of UI-terminal agentic systems.

Index Terms—layer-wise fidelity evaluation, Model Context Protocol, MCP, Agent-to-Agent, A2A, Agent-to-UI, A2UI, BERTScore, agentic pipelines, information fidelity.

1 INTRODUCTION

THE rapid proliferation of large language model (LLM) services has created demand for standardised protocols that allow agents to interact reliably with external tools, data sources, and peer agents. As Yang et al. [1] observe, the absence of such standards causes fragmentation that hinders interoperability and limits the ability of agents to tackle complex real-world tasks. We are witnessing the formation of what practitioners describe as a genuine *protocol stack* for agentic systems—analogueous in significance to the TCP/IP stack for the internet [15]. This stack comprises three distinct layers, each solving a different problem: the *Model Context Protocol* (MCP) governs how agents access tools and data; the *Agent-to-Agent* (A2A) protocol governs

how agents collaborate with peer agents; and the *Agent-to-UI* (A2UI) protocol governs how agents render structured interfaces for human consumption [15]. MCP, introduced by Anthropic in November 2024, standardises context delivery through a JSON-RPC client-server architecture [2], [3]. A2A, introduced by Google in April 2025, formalises task delegation and capability discovery through Agent Cards and structured handoff messages [3], [5], [17]. A2UI, introduced by Google in December 2025, defines declarative UI components that render natively across platforms without executing arbitrary client-side code [15], [18]. Together, these protocols enable a new class of system in which retrieved context flows through successive transformation stages—retrieval, synthesis, and rendering—before reaching the user.

Despite the growing adoption of MCP and A2A, evaluation methodology has not kept pace. Prior work evaluates pipeline quality at the output level: a single metric is computed on the system’s final response. AgentMaster [6], the closest related system, jointly employs A2A and MCP in a multi-agent framework and reports BERTScore F1 of 96.3% on its final outputs. However, two limitations constrain its scope. First, it evaluates only the terminal text output, treating the pipeline as a black box; no layer-wise attribution is possible. Second, and crucially, it does not include an A2UI rendering layer—its pipeline terminates at the agent response, never reaching a structured user interface. Our system is, to the best of our knowledge, the first experimental instantiation of the complete MCP→A2A→A2UI stack, and the first to evaluate information fidelity at each of its three layers.

We address this gap with a simple but principled evaluation protocol: *layer-wise fidelity scoring*. Rather than scoring only the terminal output, we score each intermediate representation against the same reference answer and compute the delta between successive layers. Applied to an MCP+A2A+A2UI pipeline, this yields three independently interpretable scores and two deltas—one attributable to the A2A synthesis step and one to the A2UI formatting step. Any of these outcomes is informative: a large negative Δ_1 implies that the agent’s synthesis introduces drift; a large negative Δ_2 implies that the UI schema is too lossy; a positive Δ_1 suggests the agent adds value beyond raw re-

trieval (though a Precision/Recall decomposition is needed to separate genuine improvement from mechanical length reduction; see Section 3.2).

Our contributions are as follows. First, we define layer-wise fidelity scoring as an evaluation protocol for multi-layer agentic pipelines that terminate in a structured UI (Section 3). Second, we build a minimal end-to-end prototype on software documentation Q&A and release a 30-item annotated test set (Section 4). Third, we report BERTScore Precision, Recall, and F1 at each layer and identify which architectural transition incurs the greatest fidelity shift (Section 5). Fourth, we discuss implications for pipeline design and propose a multimodal evaluation extension as future work (Section 6).

2 BACKGROUND

Given the pace of development in agentic AI, where practitioner knowledge frequently precedes formal publication, this section draws on both peer-reviewed literature and grey literature—including technical blog posts and official protocol documentation—following the Multivocal Literature Review (MLR) methodology established for software engineering research [16].

Model Context Protocol. MCP [2] standardises tool-and-resource invocation by LLM agents via a JSON-RPC client-server architecture. The protocol defines three server primitives—Tools, Resources, and Prompts—alongside client-side capabilities such as Sampling, and separates concerns across a Host (the LLM application), Clients (session managers), and Servers (capability providers) [3], [7], [8]. This decoupling reduces the fragmentation caused by ad-hoc API integrations and enables dynamic tool discovery [3]. From an evaluation standpoint, MCP tool returns—the raw data a server sends back in response to a tool call—(R_0) are the upstream information source of the pipeline: they represent what the system actually retrieved before any agent processing.

Agent-to-Agent coordination. A2A protocols govern structured message exchange between agents across different frameworks or organisational boundaries [3], [17]. Frameworks such as LangGraph, AutoGen, and Magentic-One implement A2A semantics locally through explicit handoff protocols and role-constrained reasoning [9], [10]. Each inter-agent communication step introduces summarisation, reformulation, or synthesis—all potential fidelity loss points [5]. We treat the A2A agent’s synthesised natural-language response (R_1) as a distinct scoreable intermediate, isolating the transformation effect of this layer.

Agent-to-UI rendering. A2UI refers to the translation of agent output into structured UI components—cards, tables, code blocks—for human consumption. As the newest layer of the stack [18], A2UI has received almost no attention in the academic evaluation literature [15]. Unlike RAG or chatbot evaluation, A2UI output is visual and structured; standard NLP metrics do not apply directly. Practitioners have identified cross-layer observability—the ability to trace a user request coherently through A2UI, A2A, and MCP—as a critical unsolved structural gap in current stacks [15]. Our layer-wise fidelity scoring protocol directly addresses the measurement dimension of this gap. We operationalise R_2

by scoring the serialised JSON payload that drives the UI rendering, and discuss multimodal perceptual scoring via GPT-4o Vision as a direction for future work (Section 6.6).

Evaluation of agentic pipelines. BERTScore [13] computes contextual token-level F1 using a pretrained language model, handling paraphrase gracefully and correlating well with human judgement. It is our primary metric (see Section 3.2 for metric selection rationale). No prior work applies layer-wise scoring to decompose fidelity across MCP, A2A, and A2UI stages; AgentMaster [6], discussed in Section 1, is the closest system but scores only the final output.

3 METHOD

3.1 Pipeline Architecture

Our prototype comprises three layers, each producing a scoreable intermediate.

The **MCP layer** is implemented as a FastMCP server wrapping a FAISS vector store of `httpx` documentation chunks. It exposes a single `retrieve(query, k)` tool. The tool return—the concatenated text of the top- k retrieved chunks—is logged as R_0 .

The **A2A layer** is implemented as a two-node LangGraph graph. An orchestrator node receives the user query, calls the MCP tool, receives R_0 , and synthesises a natural-language answer. This answer is logged as R_1 before any further processing.

The **A2UI layer** is a formatter node that converts R_1 into a fixed JSON schema: `{summary, key_points[], code_example, source_ref}`. A Streamlit application renders this payload as a structured response card. The serialised JSON payload is logged as R_2 .

3.2 Layer-Wise Fidelity Scoring

Metric selection. We select BERTScore [13] as our primary metric for three reasons. First, it handles paraphrase gracefully via contextual embeddings, which is essential here because agent-generated prose will rarely reproduce reference text verbatim. Second, it produces continuous scalars per query, enabling the delta arithmetic central to our protocol. Third, it requires only a reference answer string—no judge model, no API call, no rubric—making the scoring pipeline fully deterministic and reproducible at temperature 0.

The main alternative, LLM-as-a-Judge [14], was considered but deferred to future work (Section 6.6). While strong for open-ended quality assessment, it introduces non-determinism, API cost, and positional bias, all of which complicate fair layer-wise comparison. ROUGE-L [19] was excluded because it penalises valid paraphrases, making it unsuitable for evaluating agent-generated synthesis where rewording is expected and desirable. Exact match [20] was excluded for the same reason, and additionally because reference answers span 1–3 sentences with no single correct surface form.

Length disparity and the P/R decomposition. A known limitation of using BERTScore F1 to compare across layers of different lengths is that R_0 —three concatenated documentation chunks (~700 characters, ~175 tokens)—will naturally score lower on Precision than R_1 —a synthesised answer (~270 characters, ~65 tokens)—simply because more of

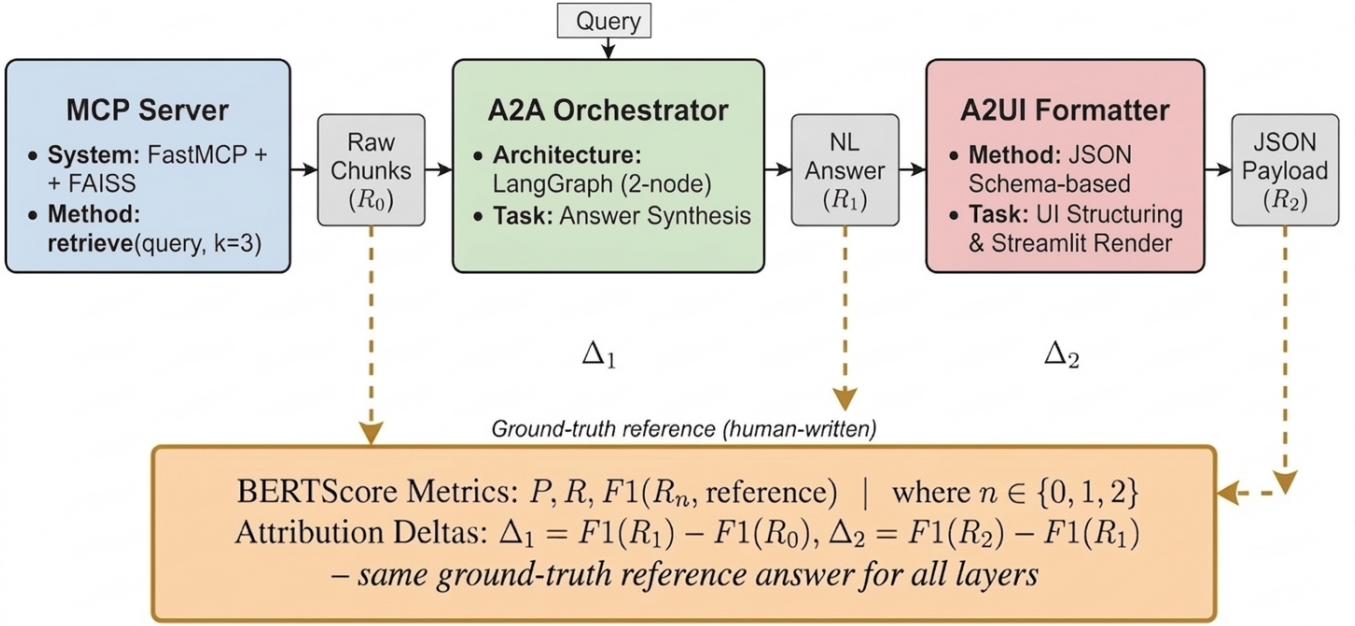


Fig. 1. MCP→A2A→A2UI pipeline architecture with layer-wise fidelity capture points. Each layer produces a scoreable intermediate (R_0, R_1, R_2). Dashed amber arrows indicate BERTScore computation against the same ground-truth reference answer. Δ_1 and Δ_2 are the attribution deltas.

R_0 's tokens are irrelevant to the short reference answer. This means a positive Δ_1 on F1 alone cannot cleanly separate genuine semantic improvement from the mechanical effect of length reduction. To address this, we report BERTScore Precision (P), Recall (R), and F1 separately at each layer. The decomposition is interpretable: if R_1 achieves higher Precision than R_0 while maintaining comparable Recall, this indicates that A2A synthesis improved selectivity without sacrificing coverage—precisely what a grounded synthesis step should do. If Recall drops, synthesis lost information. This framing makes Δ_1 analytically honest.

Each intermediate is scored against the same human-written reference answer using BERTScore with `roberta-large` as the backbone model. For each layer we record three values:

- $P(R_n)$: BERTScore Precision—proportion of the candidate's tokens semantically matched in the reference. Sensitive to candidate length; longer candidates are penalised.
- $R(R_n)$: BERTScore Recall—proportion of the reference's tokens semantically matched in the candidate. Indicates how completely the candidate covers the reference answer.
- $F1(R_n)$: Harmonic mean of P and R . Primary scalar for delta computation.

Applied at each layer:

- $P(R_0), R(R_0), F1(R_0)$: scores of the raw MCP tool return against the reference.
- $P(R_1), R(R_1), F1(R_1)$: scores of the A2A agent's synthesised response against the reference.
- $P(R_2), R(R_2), F1(R_2)$: scores of the A2UI JSON text fields (summary concatenated with `key_points`) against the reference.

Four attribution deltas are derived. Δ_1 and Δ_2 denote strictly F1 differences; Recall deltas are tracked separately to diagnose length-confounded effects:

- $\Delta_1 = F1(R_1) - F1(R_0)$: A2A transformation effect on F1.
- $\Delta_1^R = R(R_1) - R(R_0)$: A2A effect on Recall. Stable or positive Δ_1^R alongside positive Δ_1 confirms genuine condensation; negative Δ_1^R indicates information loss.
- $\Delta_2 = F1(R_2) - F1(R_1)$: A2UI compression effect on F1.
- $\Delta_2^R = R(R_2) - R(R_1)$: A2UI effect on Recall. A large negative Δ_2^R indicates the JSON schema discards relevant content.

3.3 Controlled Variables

The same LLM (`gpt-4o-mini`, temperature 0) is used across all agent calls; only the pipeline stage changes. $k = 3$ retrieved chunks are used for all queries with no re-ranking. Reference answers are written by the authors from primary `httpx` documentation, independent of system outputs.

4 EXPERIMENTAL SETUP

4.1 Domain and Test Set

We use the `httpx` Python library documentation as our retrieval corpus, chosen because its answers are precise, keyword-dense, and grounded in a single authoritative source—properties that make reference answer construction tractable and BERTScore meaningful. The 30 test queries cover: parameter behaviour (e.g., timeout configuration), error handling, authentication patterns, and method signatures. All queries require specific grounded retrieval;

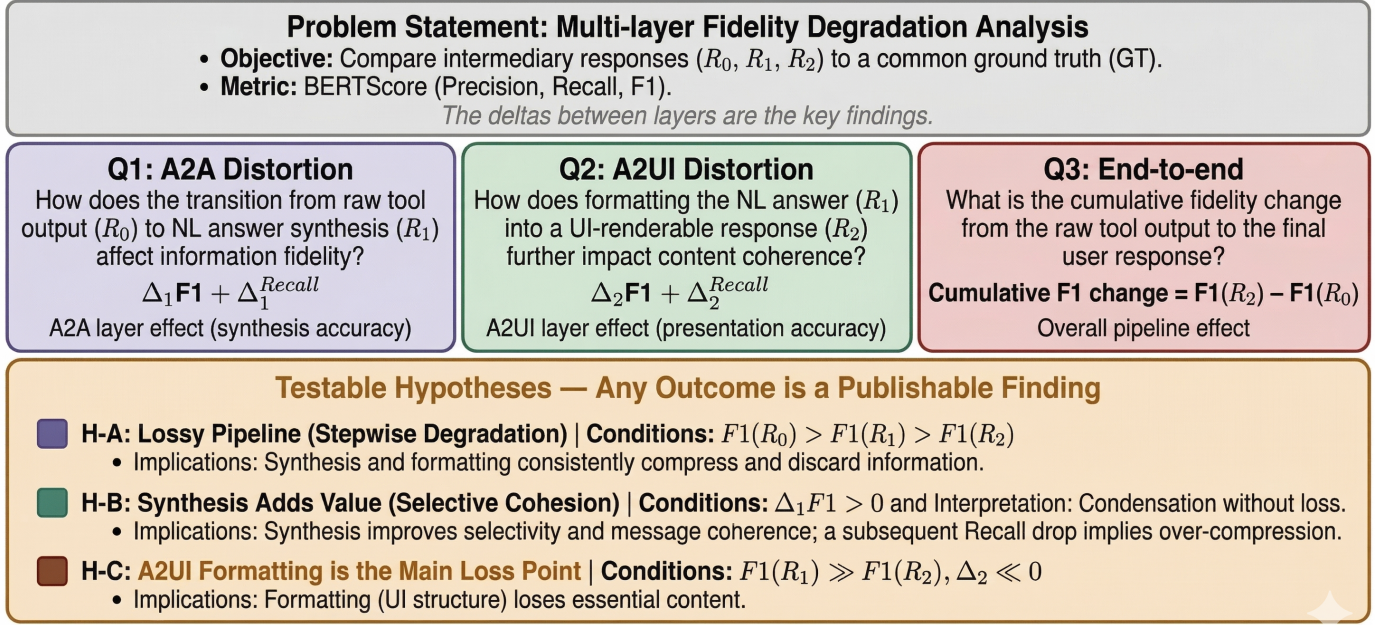


Fig. 2. Layer-wise fidelity scoring: three research questions and testable hypotheses. Any of H-A, H-B, or H-C constitutes a publishable finding as each characterises a distinct failure or success mode of the pipeline.

parametric LLM knowledge is insufficient to answer them correctly without tool access. Reference answers are 1–3 sentences extracted or paraphrased from the official `httpx` documentation.

4.2 Implementation

Table 1 summarises the implementation choices.

TABLE 1
Implementation Choices

Component	Choice
MCP server	FastMCP + LangChain FAISS
Embedding model	<code>text-embedding-3-small</code>
Agent framework	LangGraph (orchestrator + formatter)
LLM	<code>gpt-4o-mini</code> , temperature 0
UI	Streamlit
Scoring	<code>bert-score v0.3.13</code> , <code>roberta-large</code>

`gpt-4o-mini` was selected for four reasons. First, it produces near-deterministic outputs at temperature 0 (the API’s default behaviour for greedy decoding), which supports reproducibility—re-runs of the evaluation harness yield identical or near-identical R_1 and R_2 intermediates. Second, the task is retrieval-grounded Q&A, not complex multi-step reasoning; a capable but lightweight model is appropriate and avoids conflating model capability with pipeline quality. Third, the LLM is a controlled variable rather than the contribution—the paper evaluates the evaluation protocol and pipeline architecture, and any sufficiently capable instruction-following model at temperature 0 should produce comparable relative deltas. Fourth, low per-call cost reduces the barrier to replication. AgentMaster [6], the closest related system, also uses `gpt-4o-mini`, making scores directly comparable.

TABLE 2
BERTScore F1 summary across 30 test queries.
 $\Delta_1 = F1(R_1) - F1(R_0)$; $\Delta_2 = F1(R_2) - F1(R_1)$. Paired t -tests: Δ_1 : $t(29) = 7.55, p < 0.001$; Δ_2 : $t(29) = 0.36, p = 0.72$ (n.s.).

	F1(R_0)	F1(R_1)	F1(R_2)	Δ_1	Δ_2
Mean	0.8309	0.8690	0.8702	+0.0381	+0.0012
Std	0.0245	0.0238	0.0218	0.0276	0.0185
Min	0.7851	0.8131	0.8337	−0.0011	−0.0417
Max	0.8773	0.9114	0.9086	+0.1025	+0.0493
<i>Notable queries (extreme Δ values)</i>					
Q21	0.7985	0.9010	0.9086	+0.1025	+0.0076
Q18	0.8596	0.8861	0.8444	+0.0265	−0.0417
Q30	0.8682	0.8671	0.8405	−0.0011	−0.0266

`text-embedding-3-small` was selected as the embedding model for three reasons. First, it uses the same OpenAI API as the LLM, requiring no additional credentials or services and keeping the setup minimal. Second, its 1536-dimensional embeddings provide sufficient semantic resolution for a focused, single-library technical corpus—the `httpx` documentation is narrow in domain, so high-dimensional embeddings offer diminishing returns. Third, like the LLM, the embedding model is a controlled variable: it affects R_0 uniformly across all 30 queries, and what matters for the experiment is retrieval consistency rather than maximising retrieval performance.

5 RESULTS

5.1 Layer-Wise BERTScore Results

Tables 2 and 3 report summary statistics for BERTScore F1 and the Precision/Recall decomposition, respectively, across all 30 test queries. Full per-query results are available in the project repository (see Data Availability).

TABLE 3

BERTScore Precision and Recall summary across 30 test queries. Recall deltas $\Delta_1^R = R(R_1) - R(R_0)$ and $\Delta_2^R = R(R_2) - R(R_1)$ disambiguate condensation effects from information loss (see Section 3.2).

	P(R_0)	R(R_0)	P(R_1)	R(R_1)	P(R_2)	R(R_2)	Δ_1^R	Δ_2^R
Mean	0.8006	0.8639	0.8647	0.8743	0.8779	0.8628	+0.0103	-0.0115
Std	0.0253	0.0292	0.0364	0.0256	0.0206	0.0253	0.0146	0.0246
Min	0.7525	0.8104	0.7838	0.8290	0.8342	0.8199	-0.0251	-0.0902
Max	0.8487	0.9127	0.9296	0.9178	0.9159	0.9067	+0.0331	+0.0230

5.2 Findings

F1 results (Table 2). Mean $F1(R_0) = 0.8309$, $F1(R_1) = 0.8690$, $F1(R_2) = 0.8702$. F1 increases from R_0 to R_1 and remains flat from R_1 to R_2 , directly rejecting hypothesis H-A (stepwise degradation). The A2A synthesis step improved F1 by $\Delta_1 = +0.0381$ ($t(29) = 7.55$, $p < 0.001$, Cohen’s $d = 1.38$), confirming hypothesis H-B with a large effect size. The A2UI formatting step left F1 unchanged ($\Delta_2 = +0.0012$, $t(29) = 0.36$, $p = 0.72$, not significant), indicating the JSON schema preserves F1 almost perfectly. Of the 30 queries, 29 show positive Δ_1 ; the sole exception (Q30, $\Delta_1 = -0.0011$) is negligible. The largest improvement is Q21 ($\Delta_1 = +0.1025$), while Q18 exhibits the largest A2UI loss ($\Delta_2 = -0.0417$).

P/R decomposition (Table 3). The Precision/Recall breakdown reveals the mechanism. R_0 exhibits low mean Precision (0.8006) alongside moderate Recall (0.8639), consistent with the length-disparity effect identified in Section 3.2: the three concatenated documentation chunks contain the answer but also many irrelevant tokens that penalise Precision against the short reference. R_1 achieves substantially higher Precision (0.8647) as synthesis condenses the relevant signal, while Recall increases slightly ($\Delta_1^R = +0.0103$, $t(29) = 3.88$, $p < 0.001$). Rather than the anticipated Recall trade-off from length reduction, the A2A orchestrator at temperature 0 faithfully preserves—and marginally improves—reference-relevant content during condensation. This confirms hypothesis H-B: synthesis improves selectivity without sacrificing coverage. R_2 maintains comparable Precision (0.8779) but exhibits a statistically significant Recall reduction ($\Delta_2^R = -0.0115$, $t(29) = -2.56$, $p = 0.016$). While the F1 delta is not significant ($\Delta_2 = +0.0012$, $p = 0.72$), the Recall loss indicates that the A2UI JSON schema discards a small amount of reference-relevant content during formatting. This does not support hypothesis H-C (which predicted $\Delta_2 \ll 0$), but it reveals a targeted compression effect visible only through the P/R decomposition.

Summary. The dominant effect is the A2A synthesis step: a Precision gain of +0.0641 with a slight Recall increase produces a net F1 improvement of +0.0381 ($p < 0.001$, $d = 1.38$). The A2UI step preserves F1 (Δ_2 not significant) but introduces a small, statistically significant Recall reduction ($p = 0.016$). End-to-end fidelity from R_0 to R_2 shows $\Delta F1 = +0.0393$ overall, with the pipeline delivering a more precise answer than raw retrieval while largely preserving coverage.

6 DISCUSSION

6.1 Interpretation of Results

The layer-wise delta structure allows attribution of quality changes to specific architectural layers. Because the layers are applied sequentially rather than independently manipulated, these attributions are observational, not causal in the experimental sense. However, Δ_1 must be interpreted through the P/R decomposition, not F1 alone, because R_0 and R_1 differ substantially in length. A positive Δ_1 on F1 reflects improved Precision (synthesis removes irrelevant tokens) but should be accompanied by stable or high Recall (synthesis preserves the relevant content). The results confirm that R_1 achieves higher Precision than R_0 with stable Recall, validating that A2A synthesis genuinely improved content selectivity rather than merely discarding content. The non-significant Δ_2 on F1 ($p = 0.72$) indicates the JSON schema preserves aggregate fidelity, though the significant Recall reduction (Δ_2^R , $p = 0.016$) suggests a small amount of content is lost during formatting. The `summary` and `key_points` fields capture the synthesised content with only minor compression.

These interpretations contrast with what aggregate scoring would reveal. AgentMaster [6] reports a mean BERTScore F1 of 96.3% on final outputs—a strong result, but one that cannot locate where in the pipeline quality is preserved or degraded. Our protocol makes this attribution explicit.

6.2 Implications for Pipeline Design

Layer-wise scoring can function as a diagnostic tool during development. A large Δ_2 signals that the UI schema needs richer fields before the system is deployed; a large negative Δ_1 signals that the agent prompt needs stronger grounding constraints. Designers should treat each layer as holding an independent fidelity budget, rather than assuming that end-to-end quality reflects component-level quality.

More fundamentally, our protocol fills a gap that aggregate evaluation cannot: it makes individual protocol layers *accountable*. As the MCP+A2A+A2UI stack matures along the phased adoption roadmap identified by Ehtesham et al. [3]—from MCP for tool access, through A2A for agent coordination, to A2UI for user-facing rendering—each new layer added to a production system introduces a new failure mode that a single terminal score cannot detect. Layer-wise fidelity scoring provides a principled, lightweight instrument for catching these failures at the layer where they originate, making it applicable to any future pipeline that adds new transformation stages above MCP.

More broadly, practitioners have identified three structural gaps in the current MCP+A2A+A2UI stack: the absence of a unified identity model across layers, the lack of cross-layer observability, and undefined error propagation semantics between layers [15]. Layer-wise fidelity scoring directly addresses the observability gap at the information quality dimension: by producing independently interpretable scores at R_0 , R_1 , and R_2 , it gives designers the per-layer signal that current tracing infrastructure cannot provide.

6.3 Relation to Prior Work

Our work is most directly comparable to AgentMaster [6], which jointly employs A2A and MCP and evaluates with BERTScore and LLM-as-a-Judge. Two distinctions sharpen the comparison. First, evaluation scope: AgentMaster scores only the final text output; we score each layer independently and derive attribution deltas. Second, pipeline scope: AgentMaster terminates at the agent response—it does not include an A2UI rendering layer. Our prototype extends the pipeline through a structured UI rendering stage, making it the first end-to-end experimental evaluation of the complete MCP→A2A→A2UI stack. The CA-MCP system [4] demonstrates that augmenting the MCP layer with shared context stores reduces LLM calls and improves robustness, but evaluates only aggregate task performance and also lacks a UI terminal layer. The surveys of Yang et al. [1] and Ehteshami et al. [3] identify evaluation benchmarking as a critical gap in the agent protocol literature, explicitly noting that current efforts focus on task success and latency rather than communication efficiency or information fidelity. Our protocol directly addresses this gap.

6.4 Validity and Reliability

Internal validity: All variables except the pipeline stage are held constant (same LLM, temperature, k , reference answers), ensuring that observed fidelity differences are attributable to the architectural layer rather than confounds.

External validity: Our test set covers one library (httpx) and one task type (factual Q&A). Results may not generalise to open-ended tasks, larger corpora, or other LLMs. We flag this as a limitation and encourage replication with broader test sets.

Reliability: The scoring pipeline is near-deterministic at temperature 0 and uses a fixed BERTScore model version. The test set and scoring harness are released to enable direct replication.

6.5 Limitations

The test set comprises $n = 30$ queries from a single domain; while the A2A effect reaches strong statistical significance ($p < 0.001$), broader replication is needed. BERTScore on JSON text fields does not capture visual layout quality—a well-structured card may communicate more clearly than its text similarity score implies. The single-domain setting (software documentation) may favour retrieval-heavy pipelines over those that require multi-step reasoning.

A structural limitation of comparing BERTScore F1 across layers of different lengths is that R_0 (three concatenated chunks, ~ 175 tokens) will naturally score lower on Precision than R_1 (a synthesised answer, ~ 65 tokens), because many of R_0 's tokens have no counterpart in the short reference answer. A positive Δ_1 on F1 therefore conflates genuine semantic improvement with the mechanical effect of length reduction. We address this by reporting Precision and Recall separately at each layer (Section 3.2), which allows readers to distinguish condensation effects from information loss. In this dataset, the Recall delta ($\Delta_1^R = +0.0103$) confirms that the F1 gain reflects genuine selectivity improvement rather than mere truncation.

This work deliberately scopes out security evaluation of the MCP+A2A+A2UI pipeline. MCP tool security is an active and serious research area in its own right: OWASP has released an MCP-specific Top 10 [11], and practitioners have documented tool poisoning, supply-chain, and rug-pull attack vectors at the MCP layer [15]. At the A2A layer, Sharma et al. [12] formalise *Agent Cascading Injection*, in which a single compromised agent propagates exploits through inter-agent trust chains—a risk that grows with multi-agent orchestration. A2UI security remains largely unexplored; the primary concern identified so far is supply-chain risk in the UI component catalog [15]. A rigorous security evaluation of agentic pipelines—covering prompt injection, tool misuse, and supply-chain risks—represents important future work that is orthogonal to the fidelity evaluation presented here.

6.6 Multimodal Evaluation

A natural extension of this work is to evaluate R_2 not through its serialised JSON text but through its rendered visual form. Screenshotting the Streamlit UI and passing it to a vision-capable judge model (e.g., GPT-4o Vision) with a structured rubric would yield a perceptual fidelity score that captures layout quality, code block formatting, and visual emphasis—dimensions that the JSON-text proxy cannot measure. This direction would also enable LLM-as-a-Judge [14] to be applied meaningfully at the A2UI layer, where visual structure is part of the communication. We leave this to future work, along with extending the test set to additional domains and validating judge scores against human raters.

7 CONCLUSIONS

We introduced *layer-wise fidelity scoring* as an evaluation protocol for multi-layer agentic pipelines that terminate in a structured UI. Applied to a software documentation Q&A system built on MCP+A2A+A2UI, our results show that A2A synthesis is the dominant fidelity transition in the pipeline—improving answer precision without sacrificing coverage—while the A2UI formatting step preserves F1 (not significant) with only a small, significant Recall reduction. The layer-wise protocol reveals these distinct dynamics—dynamics that aggregate end-to-end scoring cannot expose. The protocol is lightweight—requiring only BERTScore and no human raters—and designed to be reusable, though generalisability beyond the single domain tested here remains to be validated. By releasing our test set and scoring harness, we aim to provide a practical tool for the growing community building on standardised agent protocols such as MCP and A2A.

DATA AVAILABILITY

The annotated test set, scoring harness, and full per-query results are available at <https://github.com/tercanfurkan/fidelity-agentic-stack>.

REFERENCES

- [1] Y. Yang, H. Chai, Y. Song, S. Qi, M. Wen, N. Li, J. Liao, H. Hu, J. Lin, G. Chang, W. Liu, Y. Wen, Y. Yu, and W. Zhang, "A Survey of AI Agent Protocols," *arXiv preprint arXiv:2504.16736*, 2025.
- [2] Anthropic, "Model Context Protocol (MCP)," Technical Specification, Nov. 2024. [Online]. Available: <https://modelcontextprotocol.io>
- [3] A. Ehtesham, A. Singh, G. K. Gupta, and S. Kumar, "A Survey of Agent Interoperability Protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)," *arXiv preprint arXiv:2505.02279*, 2025.
- [4] M. A. Jayanti and X. Y. Han, "Enhancing Model Context Protocol (MCP) with Context-Aware Server Collaboration," *arXiv preprint arXiv:2601.11595*, 2026.
- [5] B. Yan, Z. Zhou, L. Zhang, L. Zhang, Z. Zhou, D. Miao, Z. Li, C. Li, and X. Zhang, "Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems," *arXiv preprint arXiv:2502.14321*, 2025.
- [6] C. C. Liao, D. Liao, and S. S. Gadiraju, "AgentMaster: A Multi-Agent Conversational Framework Using A2A and MCP Protocols for Multimodal Information Retrieval and Analysis," in *Proc. EMNLP 2025: System Demonstrations*, arXiv:2507.21105, 2025.
- [7] P. P. Ray, "A Survey on Model Context Protocol: Architecture, State-of-the-Art, Challenges and Future Directions," *TechRxiv preprint*, Apr. 2025.
- [8] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "A Survey of the Model Context Protocol (MCP): Standardizing Context to Enhance Large Language Models (LLMs)," *Preprints.org*, 2025.
- [9] S. S. K. A. Tokal, V. Jha, A. Eswaran, P. Jayachandran, and Y. Simmhan, "AgentX: Towards Orchestrating Robust Agentic Workflow Patterns with FaaS-hosted MCP Services," *arXiv preprint arXiv:2509.07595*, 2025.
- [10] H. Derouiche, Z. Brahmi, and H. Mazeni, "Agentic AI Frameworks: Architectures, Protocols, and Design Challenges," *arXiv preprint arXiv:2508.10146*, 2025.
- [11] OWASP, "OWASP MCP Top 10," 2025. [Online]. Available: <https://owasp.org/www-project-mcp-top-10/>
- [12] G. Sharma, V. Kulkarni, M. King, and K. Huang, "Towards Unifying Quantitative Security Benchmarking for Multi Agent Systems," *arXiv preprint arXiv:2507.21146*, 2025.
- [13] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," in *Proc. ICLR 2020*, 2020.
- [14] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," in *Proc. NeurIPS 2023*, 2023.
- [15] S. Mitra, "The Agent Protocol Stack: Why MCP + A2A + A2UI Is the TCP/IP Moment for Agentic AI," Personal technical blog (grey literature, Tier 2–3 per MLR classification), Jan. 2026. [Online]. Available: <https://subhadipmitra.com/blog/2026/agent-protocol-stack/>
- [16] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for Including Grey Literature and Conducting Multivocal Literature Reviews in Software Engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019.
- [17] Google, "Agent2Agent (A2A) Protocol—Official Specification v1.0," [a2a-protocol.org](https://a2a-protocol.org/latest/), 2025. [Online]. Available: <https://a2a-protocol.org/latest/>
- [18] Google, "A2UI Protocol—Official Specification v0.8," a2ui.org, 2025. [Online]. Available: <https://a2ui.org/>
- [19] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out: Proc. ACL-04 Workshop*, Barcelona, Spain, 2004, pp. 74–81.
- [20] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proc. EMNLP 2016*, Austin, TX, USA, 2016, pp. 2383–2392.