KPI2KVI: A Multi Agent Workflow for Calculating Key Value Indicators from Service Descriptions

Masoud Shokrnezhad
ICTFICIAL OY
Espoo, Finland
masoud.shokrnezhad@ictficial.com

Tarik Taleb
Ruhr-Universitaet Bochum
Bochum, Germany
tarik.taleb@ruhr-uni-bochum.de

Yan Chen
ICTFICIAL OY
Espoo, Finland
yan.chen@ictficial.com

Qize Guo
ICTFICIAL OY
Espoo, Finland
qize.guo@ictficial.com

***Abstract*—Key Value Indicators (KVIs) provide a decision oriented view of a service by summarizing how operational performance translates into stakeholder value, risk, and outcomes. However, in many domains KVIs are difficult to compute in practice because they require selecting relevant KVI categories, defining measurable Key Performance Indicators (KPIs), collecting KPI values, and applying consistent calculation logic, all of which is typically performed manually and inconsistently from unstructured service documentation. This paper presents KPI2KVI, a tool that transforms a natural language service description into computed KVI estimates by orchestrating a deterministic multi agent workflow powered by Large Language Models (LLMs) that (i) elicits missing service context, (ii) extracts and finalizes relevant KVI categories from a taxonomy, (iii) generates service specific KPIs with units and descriptions, (iv) collects KPI values through an interactive dialogue and also supports intelligent estimation for KPI values that are unavailable, and (v) computes interval valued KVI outputs (minimum, exact, maximum) with traceable explanations for each KVI code. Simulations with representative service descriptions demonstrate that KPI2KVI consistently produces a complete end to end mapping from description to KVI intervals and provides transparent calculation narratives that support post hoc auditing and interactive advisory queries.**

***Index Terms*—key value indicators, key performance indicators, service modeling, multi agent systems, large language models, interval estimation**

I. Introduction

The Sixth Generation (6G) vision increasingly frames networks as critical societal infrastructure, expected not only to deliver advanced capabilities but also to contribute to long-term objectives such as sustainability, inclusion, resilience, and trust [1]–[3]. As these expectations shape research agendas, governance, and procurement, stakeholders need ways to *demonstrate*, *compare*, and *audit* the value impact of services (not just their technical performance) across heterogeneous contexts and lifecycle stages [4]. Key Value Indicators (KVIs) were proposed to make such value outcomes measurable and actionable, but in practice their computation is challenging: value effects are indirect and multi-stakeholder, relevant evidence is often incomplete or only available via proxies (measurements, certifications, surveys), and early-stage designs must still provide credible estimates with transparent assumptions [1]. Without reproducible calculation workflows that link service descriptions to evidence-backed KVIs, value assessment risks becoming ad-hoc, hard to optimize, and prone to inconsistency or "value-washing" [2].

Existing approaches broadly fall into two strands. First, concept and governance frameworks motivate KVIs and propose value-driven assessment processes (e.g., eliciting stakeholders and values, defining indicator candidates, and staging evaluation by maturity), which is valuable for shared language and decision-making, but typically stops short of prescribing fully specified, end-to-end computation pipelines from service descriptions to concrete, reproducible indicator values. Second, operational approaches embed KVIs into orchestration and optimization (e.g., ranking alternatives or trading off performance and value objectives), demonstrating that value-aware decisions are possible when indicators are computable, but often do so for a small, pre-selected KVI set and under strong assumptions about the availability, meaning, and provenance of required inputs. In practice, however, the hard part is often *operationalization* under realistic constraints: deciding which KVIs are actually relevant for a new heterogeneous service, turning narrative requirements into a measurement plan, and then computing results from mixed evidence sources where some inputs are missing, approximate, or only available via proxies. These issues motivate end-to-end traceability and uncertainty-aware outputs with clear, user-facing rationales.

To address these gaps, this paper proposes KPI2KVI, a Large Language Model (LLM)-powered multi-agent workflow that computes KVIs from a service description in a general, reproducible, and traceable way. KPI2KVI uses specialized LLM-based agents to (i) conduct a guided interview that elicits service intent, context, stakeholders, and potential value impacts, (ii) map the service to a controlled KVI taxonomy and finalize the KVI scope with human-in-the-loop refinement, (iii) generate a compact, service-specific Key Performance Indicator (KPI) evidence plan for the selected KVIs and collect/structure the resulting measurements with provenance, and (iv) compute each KVI with explicit $\{\text{exact}, \min, \max\}$ bounds and a short rationale that cites the precise KPIs inputs and assumptions used. By combining LLM semantic understanding with a deterministic staged pipeline and persistent structured artifacts, KPI2KVI systematically bridges stakeholder value expectations to measurable evidence and makes uncertainty, assumptions, and computation steps explicit and auditable.

The rest of this paper is organized as follows. Section II

reviews some related work on KVIs concepts, frameworks, and optimization-based operationalizations. Section III presents the KPI2KVI workflow and architecture in detail. Section IV evaluates the approach through simulations. Section V concludes and outlines directions for future work.

II. Literature Review

In 6G, a *service* is an end-to-end capability for users or verticals realized by chaining functions across heterogeneous domains (edge/cloud, terrestrial/non-terrestrial) and governed through Service Level Agreements (SLAs) or intent-based abstractions [1], [5]. A service request therefore combines functional goals with workload characteristics and stringent *requirements* on latency, throughput/data rate, reliability/availability, coverage, positioning, privacy, and security, often context-dependent [1], [5]. In intent-based formulations, services may be decomposed into tasks and mapped to intent categories to make requirements machine-actionable and comparable [5]. KPIs are technical, measurable quantities estimating performance (e.g., delay, throughput, packet loss), typically specified as desired values with tolerable thresholds [1], [6]. KVIs complement KPIs by estimating enabled (or harmed) societal values such as sustainability, inclusion, privacy/confidentiality, and trust [1], [2]. Because many value dimensions are not directly observable at run time, KVIs are often realized via sensor-based measurements, periodic certification/audits, and compositions over lower-level indicators [4]. This KPI–KVI split motivates values-driven service design and evaluation beyond performance-only engineering.

Several works defined KVIs and provided high-level workflows rather than fully specified calculation pipelines. Atzori *et al.* [7] proposed EthicNet/Value of Service (VoS), where stakeholders expressed KVI requirement profiles and KVIs should be monitored and composed end-to-end, but composition operators remained open. Pintor *et al.* [4] (and [8]) systematized architectural formalization (sensor-based vs. certified KVIs, metadata, and parent–child structuring). Wikström *et al.* [2] offered a five-step framework from scenario/value elicitation to KVI formulation and staged assessment, while their white paper defined KVIs as the *scale of effect* of a use case and linked Key Values (KVs)→KVIs→enablers→KPIs, explicitly noting the unknown "exchange rate" between usage and societal value [1]. Ziegler *et al.* [9] provided qualitative KPI-to-value impact mapping, and Osman *et al.* [10] emphasized KPI proxies and business-model-driven prioritization. Other 6G discussions highlighted ecosystem/governance drivers without prescribing calculation rules [11]–[13], while enterprise Business Intelligence (BI) work focused on KPI aggregation/visualization rather than KPI→KVI translation [14]. Across these contributions, assessment was often staged by maturity: early Technology Readiness Levels (TRLs) relied on expert/qualitative evidence, later TRLs relied on measurement and more objective signals [1].

A second group operationalized KVIs through explicit calculation embedded in orchestration/optimization. De Trizio *et al.* [5] modeled intent-mapped service provisioning as a many-to-many matching problem, combining KPI constraints (deadline, throughput) with KVI-related constraints (budget, risk appetite) and ranking providers by entropy-weighted Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) over cost and cyber risk. Sciddurlo *et al.* [6] defined KPI vectors for services/resources, computed KVI components (environmental sustainability, trustworthiness, inclusiveness) via formulas, and optimized KPI–KVI trade-offs via a bi-objective model solved by an exact ϵ-constraint method; they also proposed translating natural-language requests into intents enriched with KPIs and KVIs. Mertens *et al.* [15] proposed Sustainable Development Goal (SDG)-indexed KVIs where "objective" service KVIs could be scored via International Organization for Standardization (ISO)-standards coverage ratios and combined with user preference profiles. Methodologically, these works relied on normalization and aggregation (e.g., relative-closeness ranking in [5], weighted-sum KVI aggregation and Pareto optimization in [6]).

Despite rapid progress, gaps remained for computing *service-relevant* KVIs in a general, reproducible way. Vision/framework works clarified KVI concepts and governance, but often did not specify how to select relevant KVIs for a new service nor how to compute them from available evidence without extensive manual modeling [2], [4], [7]. Qualitative mappings and vertical matrices communicated priorities but were difficult to audit, compare, or optimize because scales and aggregation rules were coarse or subjective [9], [13]. Optimization-oriented work, in contrast, typically fixed a small KVI set and assumed access to non-trivial inputs (e.g., carbon factors, attack likelihoods, certification mappings) that might be unavailable or ambiguous at design time [5], [6], [15]. Across strands, there was limited support for missing/uncertain evidence, end-to-end traceability from service description to computed indicators, and systematic bridging from stakeholder value expectations to measurable KPI evidence [1], [10]. Moreover, many approaches provided neither uncertainty bounds (e.g., intervals) nor concise, user-facing rationales connecting results to underlying evidence. These limitations motivated more operational and traceable KPI-to-KVI computation workflows that remained usable under uncertainty and heterogeneous services.

III. Approach

This section presents KPI2KVI, a multi-agent workflow that calculates KVIs for a given service. We first define the nine-stage workflow and the roles of its LLM-based agents, then describe the system architecture and orchestration logic that implement this workflow, and finally walk through a concrete example that shows how a small set of KVIs can be calculated for a cloud-based telemedicine video consultation service.

A. Workflow and Agent Responsibilities

The end-to-end workflow is a deterministic pipeline composed of nine steps, alternating between conversational steps (to elicit information and allow corrections) and structured steps (to produce machine-readable artifacts). The progression

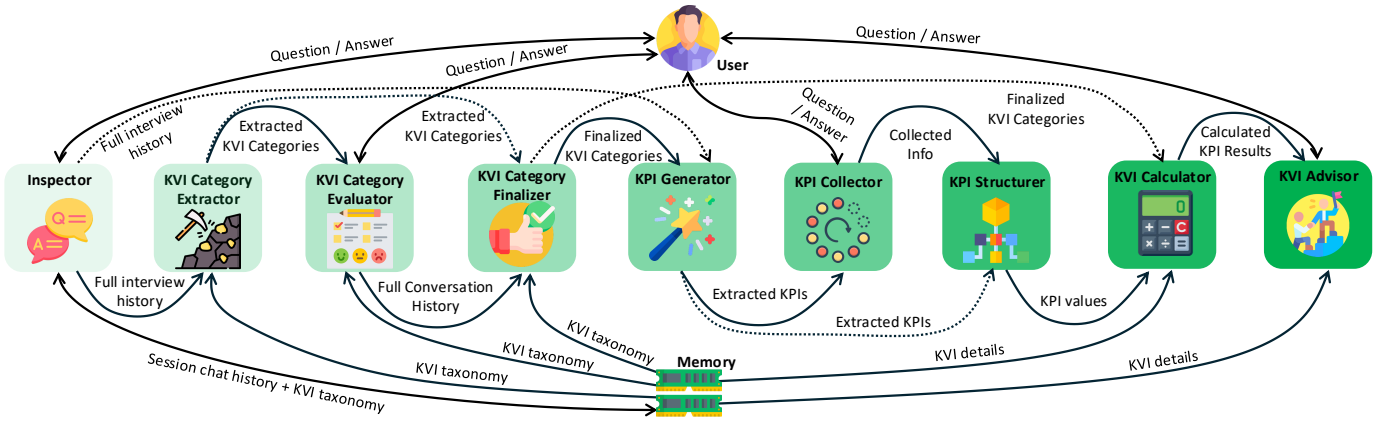

Fig. 1. KPI2KVI pipeline: from service interview to KVI category selection, KPI generation and value collection, per-KVI calculation with explicit bounds, and post-hoc advising, with shared memory storing reusable artifacts across stages.

of stages and the shared memory that carries artifacts across them are shown in Fig. 1.

a) KVI Category Selection: The workflow starts with the *inspector* (Step 1), whose *inputs* are the initial service description and any subsequent clarifications. Its *process* is a guided interview that elicits service intent, operational context, stakeholders, and potential value impacts (e.g., privacy/security expectations, sustainability concerns, accessibility constraints), and its *output* is an interview transcript stored in memory and used as the primary evidence for downstream selection. Next, the *kvi category extractor* (Step 2) takes as *inputs* the full inspector transcript and the global KVI taxonomy, *processes* them by mapping the service to the most relevant KVI categories using the taxonomy as a controlled vocabulary, and *outputs* a ranked structured list of candidate categories for refinement. The taxonomy is extracted from the KVIs proposed in [16]. The *kvi category evaluator* (Step 3) then uses as *inputs* the extracted categories and taxonomy context, *processes* them via a short conversational loop that justifies the proposal and invites the user to add/remove/replace categories based on domain knowledge and stakeholder priorities, and *outputs* a refinement transcript capturing the user's decisions. Finally, the *kvi category finalizer* (Step 4) takes as *inputs* the extractor output and evaluator transcript, *processes* them by consolidating the final category set (resolving duplicates and ensuring consistent IDs), and *outputs* a structured finalized list of categories that defines the KVI scope for subsequent KPI generation and KVI calculation; this contract is critical for auditability because it fixes *which* values will be computed before numerical evidence is collected.

b) KPI Generation: Given the finalized KVI scope, KPI2KVI next produces the measurement evidence needed for computation. The *kpi generator* (Step 5) takes as *inputs* the inspector transcript (service context) and the finalized KVI categories (value scope), *processes* them by proposing a compact set of service-specific KPIs that can serve as measurable evidence for the selected value indicators (including name, description, and unit for each KPI), and *outputs* a structured KPI list stored in memory and presented to the user as the measurement plan. The *kpi collector* (Step 6) then uses as *inputs* this KPI list and the ongoing user chat, *processes* them by collecting one KPI value at a time while explicitly supporting missing evidence (the user may provide a value or delegate it to the system, which is recorded as an assumption rather than an observation), and *outputs* a collection transcript capturing raw values, units, and whether each value was user-provided or system-decided. Finally, the *kpi structurer* (Step 7) takes as *inputs* the KPI list and the collector transcript, *processes* them by converting free-form conversation into a machine-readable table of KPI values with provenance flags and normalized representation (e.g., consistent numeric parsing and unit alignment), and *outputs* a structured KPI table that becomes the single source of truth for downstream KVI calculations.

c) KVI Calculation: Given the structured KPI table and finalized KVI scope, KPI2KVI computes the target indicators and makes the results explainable. The *kvi calculator* (Step 8) takes as *inputs* (i) one target KVI definition, including a step-by-step calculation narrative (a reasoning chain) describing how to derive the KVI and (ii) the structured KPI table, *processes* them by producing an estimated value with explicit bounds and a short rationale that links the result to the exact KPIs and assumptions used, and *outputs* per-KVI calculation artifacts $\{\text{exact}, \text{min}, \text{max}, \text{rationale}\}$ stored in memory. The orchestrator loops this step over all KVIs implied by the finalized categories and stores results so later stages can cite them precisely. Finally, the *kvi advisor* (Step 9) takes as *inputs* a consolidated advisor context containing the finalized categories, the structured KPI table, and all per-KVI calculation artifacts, *processes* them by answering user questions and explaining trade-offs with traceable justifications, and *outputs* user-facing explanations and follow-up guidance (e.g., which KPIs would most reduce uncertainty if measured more precisely).

B. Architecture

The KPI2KVI workflow is implemented as a two-tier system with a chat interface and a backend workflow controller. The frontend provides a conversational User Interface (UI) and consumes a Server-Sent Events stream to incrementally render

intermediate status updates and agent outputs. The backend exposes a streaming Application Programming Interface (API) endpoint and delegates each user turn to a workflow engine that selects and executes the appropriate step of the workflow, persists the session state, and emits progress and content events for the UI. The runtime architecture is shown in Fig. 2. The workflow controller (orchestrator) is the central component that realizes the nine-stage process described above. It discovers agent modules from a registry, executes them through an LLM interface, and maintains cross-agent memory. Concretely, the orchestrator implements a staged state machine where each session stores (i) the complete chat history, (ii) the name of the *current* agent responsible for the next user message, and (iii) a key–value store of artifacts accumulated so far. Artifacts include both human-readable text (assistant messages shown to the user) and structured objects (e.g., lists of selected KVI category IDs, a KPI list with units/descriptions, and a table of collected KPI values with provenance). This design is what enables the system to advance from one workflow step to the next without losing context, while still keeping each step's output explicit and auditable.

Operationally, the orchestrator performs three actions on every user turn. First, it *builds context*: it constructs the next prompt by combining the new user message with the relevant subset of stored artifacts (e.g., inspector transcript for category extraction; finalized categories for KPI generation; structured KPI table for KVI calculation). Second, it *runs and routes*: it executes the current agent, inspects the response for completion cues (for conversational steps), and when a step is complete it automatically triggers the next structured step(s) without requiring another user message. Third, it *stores and streams*: it writes outputs back to memory under explicit keys and streams progress/content events so the frontend can render multi-agent turns coherently. Finally, the architecture separates *domain knowledge* from *interaction logic*: the orchestrator loads the shared KVI taxonomy and canonical KVI definitions from data assets and passes them into relevant agents, ensuring stable IDs/codes and enabling reuse of intermediate structured artifacts across runs.

C. Illustrative Example and Addressing Literature Gaps

Consider a simple service: a cloud-based video consultation platform that enables remote medical appointments. The service description typically includes performance intent (stable video/audio, low delay, high availability) and operational constraints (sensitive personal data, multi-tenant deployment, and varying client devices). In KPI2KVI, the inspector first elicits missing context (e.g., what data are processed and stored, who can access recordings, which regulations apply, and the expected user base). Based on this transcript, the category extractor selects a single relevant KVI category, for example *User Trust, Perception, & Requirement Compliance*, which in the KPI2KVI taxonomy maps to KVIs such as `PUC-UPCA` (share of identified user privacy concerns addressed, %), `PUC-USCA` (share of identified user security concerns addressed, %), and `RPS-DDSS` (user-reported perceived security of the service, %).

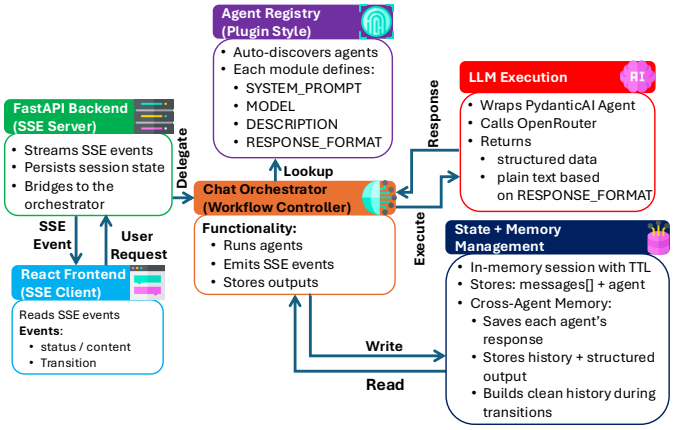

Fig. 2. TheKPI2KVI architecture: a streaming frontend/backend setup with a workflow controller that discovers agents, executes them via an LLM provider, and manages cross-agent state and reusable structured artifacts.

The KPI generator then proposes a small evidence set sufficient to compute these KVIs. For this example, consider three KPIs: N_p = number of privacy concerns collected during requirements elicitation (count), A_p = number of those privacy concerns addressed by implemented controls (count), and r_s = average user perceived-security score on a 1–5 Likert scale from a short pilot survey (dimensionless). Suppose the KPI collection yields $N_p = 10$, and because some controls are still under implementation the collector records $A_p \in [7, 9]$ (delegated estimate). From a pilot with limited respondents, assume $r_s \in [3.8, 4.4]$ with a nominal mean $r_s = 4.1$. These values are stored in the structured KPI table with provenance (the interval-valued entries explicitly reflecting uncertainty).

The KVI calculator then produces transparent formulas and interval results. For `PUC-UPCA`, a natural operationalization is the percentage of addressed privacy concerns:

$$\texttt{PUC-UPCA} = 100 \cdot \frac{A_p}{N_p}.$$

With $N_p = 10$ and $A_p \in [7, 9]$, the result is $\texttt{PUC-UPCA}_{\min} = 70$, $\texttt{PUC-UPCA}_{\max} = 90$, and $\texttt{PUC-UPCA}_{\text{exact}} = 80$ (using the midpoint). For perceived security `RPS-DDSS`, the calculator can map the Likert score to a 0–100 scale:

$$\texttt{RPS-DDSS} = 100 \cdot \frac{r_s - 1}{4}.$$

With $r_s \in [3.8, 4.4]$, we obtain $\texttt{RPS-DDSS}_{\min} = 70$, $\texttt{RPS-DDSS}_{\max} = 85$, and $\texttt{RPS-DDSS}_{\text{exact}} = 77.5$. Importantly, the advisor can later explain that the uncertainty comes from delegated control-coverage assumptions and limited survey evidence, and can recommend which KPIs (e.g., tightening A_p via control verification or increasing the survey sample for r_s) would most reduce the KVI bounds. Overall, the example demonstrates a concrete chain from service description to bounded, explainable KVIs outputs grounded in an explicit KPI evidence set.

D. Addressing Literature Gaps

KPI2KVI directly addresses the main gaps identified in Sec. II by making KPI-to-KVI computation operational, traceable, and usable under uncertainty, while keeping the workflow largely automatic through LLMs. First, instead of assuming a fixed, small KVI set or requiring extensive manual modeling, LLMs perform taxonomy-grounded category extraction and refinement, and the inspector/evaluator loops explicitly incorporate human feedback before a finalized category contract is recorded. Second, it systematically bridges stakeholder value expectations to measurable evidence by using LLMs to propose a compact, service-specific KPI set and by structuring collected values into a single machine-readable table that can be reused and compared across runs. Third, computation and explanation are supported by the KVI definitions and the calculator's step-by-step reasoning: for each KVI, the system produces bounded results with short rationales tied to the exact KPIs and assumptions, and the advisor can answer questions by citing stored artifacts. Together, these mechanisms improve end-to-end traceability from the service description to computed KVIs, and make uncertainty and human-in-the-loop corrections explicit rather than implicit.

IV. Simulations

We evaluate KPI2KVI by testing four variants: (1) *monolithic LLM* (DeepSeek-R1), where a single system prompt describes the end-to-end KPI2KVI logic and embeds the KVI taxonomy; (2) *agentic SLM (no taxonomy, no CoT)*, where the KPI2KVI agents are implemented with an Small Language Model (SLM) (Gemini 2.5 Flash Lite) and have no access to the taxonomy and no explicit Chain-of-Thought (CoT) calculation prompting; (3) *agentic SLM (+ taxonomy, no CoT)*, identical to (2) but with access to the taxonomy artifacts; and (4) *KPI2KVI (full)*. To produce the measurements, we execute each method end-to-end on a suite of service cases and systematically vary three experimental conditions: (i) the computational difficulty of KVI derivation, (ii) the scope of requested KVIs, and (iii) the quality of taxonomy grounding available to the method. Concretely, calculation complexity is controlled by selecting KVI instances whose definitions require different formula depths and different numbers of KPIs per KVI; the KVI scope is controlled by varying the number of requested KVI categories; and taxonomy quality $q \in [0, 1]$ is controlled by applying a reproducible degradation procedure to the taxonomy artifacts (e.g., removing a fraction $1 - q$ of entries or fields such as aliases, descriptions, and IDs), thereby modulating how reliable taxonomy grounding is. For each x-axis point, we repeat each method 10 times (different seeds and prompt paraphrases); the plotted curves show the mean and the shaded regions show the variance across runs.

Fig. 3 highlights complementary failure modes. In Fig. 3–A, increasing complexity amplifies run-to-run variation in computed KVIs. KPI2KVI (4) remains the most stable because it structures the KPI table, applies explicit calculation steps with bounded outputs, and enables the verifier to catch inconsistencies. The SLM baselines without CoT (2–3) show

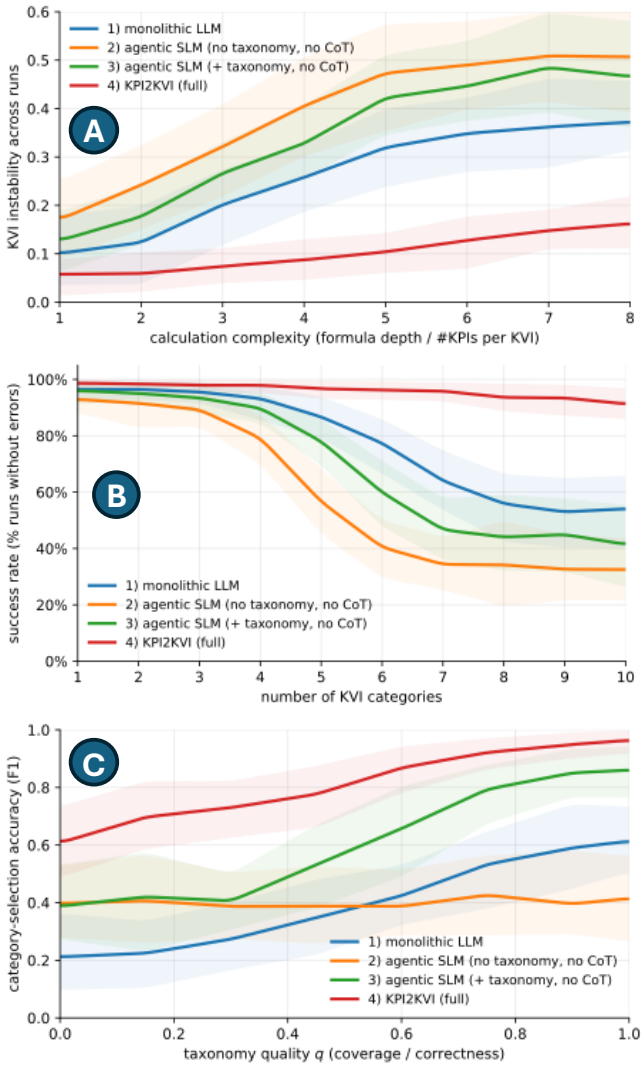

Fig. 3. Simulation results for the four method variants. (A) KVI instability across repeated runs versus calculation complexity (formula depth / #KPIs per KVI). (B) end-to-end success rate (runs without verifier-flagged errors) versus the number of requested KVI categories. (C) category-selection accuracy (F1) versus taxonomy quality q (coverage/correctness). Curves show mean over 10 runs; shaded regions show variance.

higher instability as multi-step arithmetic and unit handling become harder, while the monolithic LLM (1) is typically more stable than the SLM variants due to higher model capacity but less stable than the fully staged pipeline because it lacks intermediate structured artifacts and per-KVI contracts. In Fig. 3–B, the success rate drops as more KVI categories are requested, reflecting the difficulty of maintaining a consistent category scope, collecting sufficient KPI evidence, and completing all computations at scale. Taxonomy access delays the collapse for the agentic SLM (3) relative to (2), and the monolithic LLM (1) degrades more gracefully than the SLM baselines but still fails earlier than KPI2KVI (4), which stays robust due to its staged contract (finalized categories), shared memory artifacts, and per-KVI computation loop. Finally, Fig. 3–C isolates taxonomy dependence: method (2) is largely insensitive to q (no taxonomy), method (1) improves with q

because the taxonomy is embedded in its prompt, and method (3) benefits strongly from a high-quality taxonomy but can be misled when q is low; KPI2KVI (4) consistently achieves the best category-selection accuracy by combining taxonomy grounding with the inspector/evaluator refinement loop and explicit finalization of categories.

V. Conclusion

In this paper, we presented KPI2KVI, a multi-agent workflow that calculated KVIs for a given service from its description by translating stakeholder intent into measurable KPIs and then producing traceable, bounded KVI results. We defined a deterministic nine-stage pipeline, specified the responsibilities of each LLM-based agent, and implemented an orchestrated architecture with shared memory that stored reusable artifacts such as the interview transcript, the finalized KVI category contract, and a structured KPI value table. Using taxonomy-grounded category selection and a human-in-the-loop refinement loop, the workflow fixed the KVI scope before evidence collection, which improved auditability and reduced ambiguity in downstream calculations. We then computed each KVI with explicit formulas, interval bounds, and short rationales that linked outputs to the exact KPIs and assumptions used. In simulations, KPI2KVI achieved the best overall robustness across increasing calculation complexity, larger requested KVI scopes, and varying taxonomy quality. As future work, we planned to fine-tune SLMs so they no longer required a taxonomy and explicit CoT prompting for reliable KVI derivation, and to integrate Retrieval-Augmented Generation (RAG) to improve long-horizon memory management and reuse of past artifacts across sessions and services.

Acknowledgment

The work in this paper was supported in part by the Federal Ministry of Research, Technology, and Space (BMFTR), Germany, through the Project 6GEM+ under Grant 16KIS2411; and in part by the European Union through the 6G-SANDBOX project (Grant No.101096328) and the 6G-Path project (Grant No. 101139172).

References

[1] G. Wikström, A. Schuler Scott, I. Mesogiti *et al.*, "What societal values will 6G address?" Zenodo, Tech. Rep., May 2022.

[2] G. Wikström, N. Bledow, M. Matinmikko-Blue *et al.*, "Key value indicators: A framework for values-driven next-generation ICT solutions," *Telecommunications Policy*, vol. 48, no. 6, p. 102778, Jul. 2024.

[3] Q. Wang, A. Diaz Zayas, L. Cordeiro *et al.*, "6G-PATH Open Experimentation Platform for B5G Enablers and Diverse Vertical Applications," in *2025 IEEE International Conference on Communications Workshops (ICC Workshops)*, Glasgow, Scotland, UK, May 2026, pp. 1–6.

[4] L. Pintor, L. Atzori, and A. Iera, "Sustainability in telecommunication networks and Key Value Indicators: A survey," *Computer Networks*, vol. 271, p. 111466, Oct. 2025.

[5] F. de Trizio, G. Sciddurlo, I. Cianci *et al.*, "Optimizing Key Value Indicators in Intent-Based Networks through Digital Twins aided service orchestration mechanisms," *Computer Communications*, vol. 228, p. 107977, Dec. 2024.

[6] G. Sciddurlo, F. de Trizio, G. Piro *et al.*, "A value-driven system design framework for sustainable 6G networks," *Computer Networks*, vol. 269, p. 111477, Sep. 2025.

[7] L. Atzori, C. Campolo, A. Iera *et al.*, "Toward the EthicNet: Challenges and Enablers for Ethics-Aware Networks," *IEEE Communications Magazine*, vol. 61, no. 11, pp. 192–198, Nov. 2023.

[8] L. Pintor, L. Atzori, and A. Iera, "Building the Foundations of Ethical Networks: Integrating Key Value Indicators for Social, Economic, and Environmental Impact," in *2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2024, pp. 1–6, iSSN: 2166-9589.

[9] V. Ziegler and S. Yrjola, "6G Indicators of Value and Performance," in *2020 2nd 6G Wireless Summit (6G SUMMIT)*, Mar. 2020, pp. 1–5.

[10] H. Osman, J. Bradford, and S. Mitchell, "Bridging the Gap Between 6G Technologies and Societal Values: A Comprehensive Analysis of Key Value Indicators (KVIs) and Business Models," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2024, pp. 1–5, iSSN: 1558-2612.

[11] S. Seppo Yrjölä, P. Ahokangas, and M. Matinmikko-Blue, "Value Creation and Capture From Technology Innovation in the 6G Era," *IEEE Access*, vol. 10, pp. 16 299–16 319, 2022.

[12] C. Christophorou, I. Ioannou, V. Vassiliou *et al.*, "ADROIT6G DAI-Driven Open and Programmable Architecture for 6G Networks," in *2023 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2023, pp. 744–750.

[13] A. Pouttu, "6G white paper on validation and trials for verticals towards 2030's," Jun. 2020.

[14] V. D. Kolychev and A. A. Shebotinov, "Application of Business Intelligence instrumental tools for visualization of key performance indicators of an enterprise in telecommunications," *Scientific Visualization*, vol. 11, no. 1, 2019.

[15] J. S. Mertens, L. Galluccio, and A. Lombardo, "A deep dive into KVIs for ethics-aware networks," in *2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2024, pp. 1–6, iSSN: 2166-9589.

[16] I. Patsouras, A. Charemis, I. Wedikkara Gedara *et al.*, "6G KVIs – SNS Projects Initial Survey Results 2025," Zenodo, Tech. Rep., Apr. 2025.