Prompt-to-Metric: LLMs and Graph Algorithms for Platform Ecosystem Health Monitoring

Shady Hegazy, Muhammad Ammar, Christoph Elsner
Siemens Foundational Technologies
Siemens AG
Munich, Germany
Firstname.lastname@siemens.com

Jan Bosch
Department of Computer Science and Engineering
Chalmers University of Technology
Göteborg, Sweden
Jan.bosch@chalmers.se

Helena Holmström-Olsson
Department of Computer Science and Media Technology
Malmö University
Malmö, Sweden
Helena.holmstrom.olsson@mau.se

Abstract—**Platform ecosystems are networks of interconnected actors co-creating value through a shared technological platform. Such socio-technical systems require unique key performance indicators and health evaluation metrics to address the unique characteristics and value-creation modes they entail. Several platform ecosystems health evaluation models have been suggested in literature, along with a plethora of metrics. This study presents Prompt-to-Metric, a system that allows users, mainly platform orchestrators and decision-makers, to monitor the health of a platform ecosystem through natural language queries. The system relies on a KPI network of approximately 400 health metrics classified across four levels of hierarchy according to a model developed through a systematic literature review on the topic. In addition, the pipeline uses graph algorithms to enhance the relevancy of the responses and uncover insights regarding metrics relatedness. The system was implemented as a prototype and is being evaluated for feasibility in real-world application scenarios using data from an operational platform ecosystem. Future work includes expanding the set of calculable metrics, improving response relevance, and further evaluation in real-world settings.**

Keywords-platform ecosystem; software ecosystem; performance evaluation; analytics; graph algorithms; large language models

I. INTRODUCTION

Platform ecosystems are socio-technical networks in which diverse actors such as developers, users, and organizations collaborate and co-create value around a shared technological platform [1]. Examples of such ecosystems include open-source software communities, cloud service marketplaces, and mobile app stores. Assessing and monitoring the health of platform ecosystems presents significant challenges. Unlike traditional software systems, platform ecosystems involve complex interdependencies among actors, diverse contribution patterns, and evolving value-creation models. These characteristics require specialized key performance indicators (KPIs) and health evaluation metrics that capture both technical and socio-economic dimensions [2]. Although several health evaluation models and metrics have been proposed in the literature, they remain fragmented, tool-specific, and often inaccessible to non-technical stakeholders due to lack of attention to ecosystems unique complexities in standard analytics and performance monitoring solutions [3]. As a result, platform orchestrators and decision-makers struggle to gain actionable insights into ecosystem health. This paper presents Prompt-to-Metric, a work-in-progress system designed to address these challenges by enabling natural language access to platform ecosystem health analytics. The system integrates a hierarchical KPI network of approximately 400 health metrics, derived from a systematic literature review, and organizes them across four abstraction levels. By leveraging graph algorithms, Prompt-to-Metric retrieves relevant metrics in response to user queries and uncovers relationships among KPIs to support deeper analytical insights.

An initial prototype has been developed and deployed in a real-world platform ecosystem to evaluate its feasibility. Preliminary findings suggest that Prompt-to-Metric can bridge the gap between complex platform ecosystem health analytics and brevity and accessibility requirements of non-technical stakeholders. The contributions of this study are three-fold. First, it presents a four-tier network-based data model for ecosystem health evaluation metrics. Second, it presents a pipeline for natural language-based platform ecosystem health monitoring. Third, it presents preliminary findings from evaluations in real-world settings. The remainder of this paper is structured as follows. Section II presents details on the KPI network used within the system. Section III presents details on the Prompt-to-Metric pipeline. Section IV presents discussion of the findings from the preliminary evaluation of the system.

II. KPI NETWORK FOR PLATFORM ECOSYSTEM HEALTH EVALUATION

A. Health Metrics Elicitation

To elicit the health metrics to be integrated in the system, we conducted a systematic literature review with a focus on data-

Manuscript received July 16, 2025; revised July 26, 2025; accepted August 13, 2025. Published September 2, 2025.
Issue category: Special Issue on DSD/SEAA 2025 on Works in Progress (WiP) Session, Salerno, Italy, Sept. 2025
Paper category: Short
DOI: doi.org/10.64552/wipiec.v11i1.97

driven quantitative metrics and performance indicators of platform ecosystems [4]. The search was executed on three major scientific databases: IEEE Xplore; ACM Digital Library; and Scopus. Application of the inclusion and exclusion criteria, followed by a thorough quality appraisal, resulted in the inclusion of 52 primary studies in the review process after full-text screening and analysis. The reviewed studies presented a mix of empirical analyses, metric proposals, and framework developments targeting ecosystem robustness, resilience, productivity, niche-creation, and evolution. Our review distilled 416 health metrics ranging from low-level activity indicators to abstract ecosystem-level constructs, and varying significantly in abstraction, measurability, and scope. In addition to cataloguing these metrics, we inferred, from the overall approach of the reviewed studies, a structuring of the extracted metrics into a formal, hierarchical graph, serving as both an ontology and a computable model for automated health assessment which was integrated and operationalized through the Prompt-to-Metric system. Figure 1 shows the distribution of extracted metrics across the inferred four levels of abstraction hierarchy elaborated below.

- Level 1: Metrics at this level represent broad strategic categories that constitute a major distinct domain of performance for the ecosystem and frames a different perspective on its health. Examples include technical health; productivity; niche-creation; and network health.
- Level 2: Metrics at this level represent conceptual characteristics and qualities that indicate performance regarding specific goals or capabilities within a domain of performance. While not directly measurable, they guide the formulation of composite indicators. Examples include visibility; scalability; and robustness.
- Level 3: Metrics at this level represent tangible qualities and quantifiable indicators which indicate specific modular performance aspects. These metrics are neither abstract nor directly quantifiable but can rather be estimated by aggregating lower-level directly quantifiable metrics. Examples include developer activeness; communities' growth; contribution satisfaction; and profit focus.
- Level 4: Metrics at this level represent directly measurable and quantifiable performance indicators. Examples include bug fix time; active projects count; cash flow; and network transitivity.

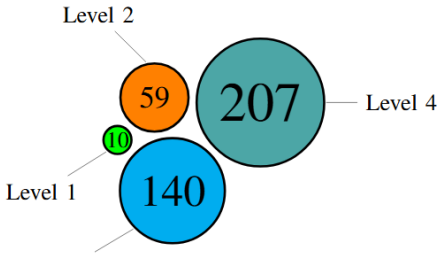


Figure 1. Distribution of health metrics per hierarchy level.

B. *Metric Graph Representation*

Each metric was modeled as a node in a directed property graph stored in a Neo4j graph database. Edges encode parent-child relationships that reflect conceptual aggregation, dependency, or influence between metrics as suggested by their origin studies. Nodes represent individual metrics, and store metric-specific attributes such as desired direction, quantifiability, measurement unit, among others. Figure 2 illustrates a subsection of this graph with each color representing a different abstraction level. The hierarchical layout allows traversing from abstract ecosystem performance aspects the tangible metrics that quantify them. The graph structure serves both analytical and operational purposes. Within the Prompt-to-Metric pipeline, this graph also acts as the lookup structure for matching user prompts to valid health metrics and their associated computation logic. For example, it enables queries such as "find all measurable indicators that contribute to developer engagement" or "identify all financial metrics associated with ecosystem maturity."

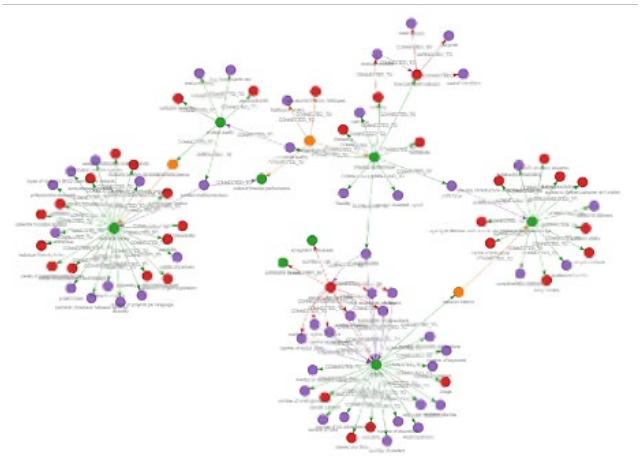


Figure 2. Visualization of a sub-graph of the KPI network.

C. *Metric Operationalization and Data Mapping*

Prompt2Metric draws on three categories of complementary data sources. Each category is ingested or queried in a way that supports real-time calculation of its associated metrics. Each metric is associated with a calculation script that executed on different data sources according to the metric genre.

- Network health metrics: Data from the platform's services and interfaces are periodically collected, transformed and loaded into a Neo4j graph database according to an integrative schema. This enables the execution of graph algorithms necessary for computing network metrics such as degree centrality, eigenvector centrality, community evolution, among others.
- Technical health metrics: Data from DevOps, version control, bug and issue tracking, and collaborative coding systems of the platform are fetched on-demand

to estimate metrics such as developer activeness, bug-fix time, and similar metrics. Additionally, analytics data of the user-facing interfaces of the platform are utilized for calculating relevant technical metrics such as crash-related metrics.

- Financial health metrics: Indicators related to financial health of the ecosystem are computed from figures extracted out of periodic financial statements.

III. PROMPT2METRIC SYSTEM

A. System Architecture

The Prompt-to-Metric system comprises five main components:

1) Metrics graph database: As described in Section II.

2) Query-to-Metric mapping engine: A language model-based component that serves as the technical bridge between natural language user inquiries and ecosystem health metrics. This component achieves the following functions.
   - Receiving the user's query from the interface.
   - Accessing the comprehensive list of available metrics.
   - Semantically analyzing the query to identify intent and required metrics.
   - Selecting the most relevant metrics from the metrics database.
   - Executing node centrality and community detection analyses on the metrics database to identify strongly relevant or closely related metrics to the initial list of relevant metrics.
   - Generating the corresponding code for the selected metrics.

3) Metric execution pipeline: A service that achieves the following functions.
   - Generates the concrete data request for the chosen metric, either as a Cypher query (for network data) a GitLab REST call (for technical data), or for other data sources, according to the calculation scripts associated with the selected metrics.
   - Retrieves and executes that request against the corresponding data source.
   - Invokes Neo4j Graph Data Science algorithms when network metrics require centrality or community detection.
   - Post-processes raw results into user-friendly tables or charts and attaches a textual interpretation
   - Logs the prompt, query, runtime and output to our MLflow instance for traceability and evaluation.
   - Streams the formatted response to the Streamlit chat interface.
   - Optionally persists snapshots for longitudinal analysis.

4) User interface: Prompt-to-Metric is delivered through a single-page Streamlit chat application that runs entirely in the browser. All conversational context is kept inside the session state containing the alternating user-assistant message list, the identifier of the last metric served, and a small cache of recently generated Cypher and REST queries. Because this state object is scoped to the browser session, no external store is needed to preserve the context between prompts. UI components include data visualizations widgets, in addition to feedback elicitation buttons. Figure 3 presents a screenshot of the interface after returning metric statistics in response to a query.

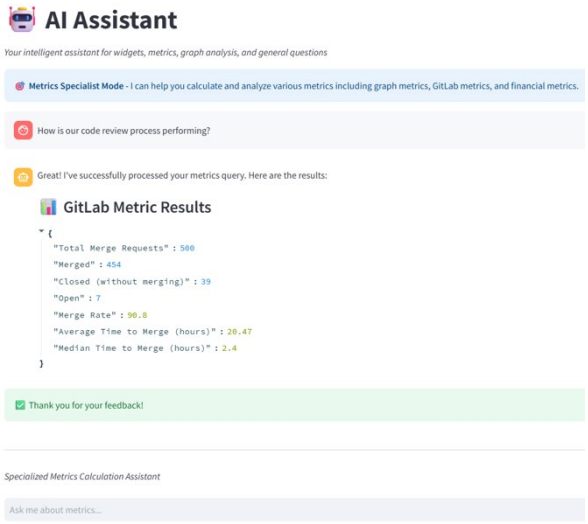


Figure 3. Prompt-to-Metric *Streamlit Chatbot UI* interface.

5) Evaluation and logging: Every user interaction is tracked by a two–stage feedback pipeline with two components.
   - Automatic run logging: As soon as a workflow is completed, a background thread generates a new MLflow run, which is hosted inside the team's GitLab instance, that records: the raw prompt and timestamp; the selected metric ID and definition; the generated calculation script; execution time; success flag; results row count; and the LLM model version that served the request.
   - Explicit user feedback: If the pipeline completes successfully, the interface displays thumbs-up and thumbs- down buttons. A thumbs-up is logged as feedback = 2, a thumbs-down as feedback = 1. If the pipeline fails before a result is shown, feedback = 0 is logged automatically. The feedback value is appended to the same MLflow run ID in a separate thread so that user experience remains unaffected. This three-point Likert-style scale provides a lightweight but actionable quality signal for longitudinal analysis, regression testing, and future fine-tuning of the LLM components.

B. Workflow

Figure 4 visualizes the end-to-end flow, which unfolds in six steps:

- User prompt: The user submits a natural-language inquiry through the Streamlit chat interface.
- Intent router: An LLM classifier decides whether the prompt requests a health metric or general conversation. Non-metric prompts are handled conversationally; metric prompts advance to Step 3.
- Metric mapper: A second LLM request compares the prompt with all metric descriptions in the unified graph and selects the best-matching metric node. The metric is then analyzed using graph algorithms to suggest closely related or strongly relevant metrics.
- Code generator: The selected metrics calculation scripts are retrieved and adapted to the prompt context. The generated code is executed on the targeted data sources, and the responses are fetched and passed forward for post-processing.
- Result delivery: Raw results are post-processed and displayed in the Streamlit UI and optionally persisted as CSV snapshots.
- MLflow logging: The prompt, chosen metric, generated query, execution metadata, output summary, and user feedback are logged to MLflow for traceability and future evaluation.

IV. Discussion

The preliminary evaluation of the Prompt-to-Metric prototype in a real-world platform ecosystem has provided valuable insights into its feasibility and potential impact. Early deployments with platform orchestrators and technical managers suggest that natural language access to ecosystem health metrics can significantly lower the barrier to understanding complex platform dynamics. Users were able to formulate high-level queries about ecosystem performance and receive actionable responses without requiring prior knowledge of underlying data structures or query languages. The hierarchical KPI network, comprising over 400 metrics organized across four abstraction levels, proved effective in structuring a wide range of health indicators. The integration of graph algorithms enabled the system to identify related metrics and suggest complementary indicators, which was perceived as particularly helpful for exploring unfamiliar dimensions of ecosystem health. However, the evaluation also surfaced key challenges. In particular, financial metrics were found to be difficult to estimate due to data clearance issues, which limited the system's ability to provide a complete view of ecosystem health in some scenarios. Furthermore, certain queries involving large-scale network computations introduced noticeable response latency, and ambiguous prompts sometimes led the language model to select metrics that were technically relevant but not fully aligned withthe user's intent. Despite these challenges, the evaluation

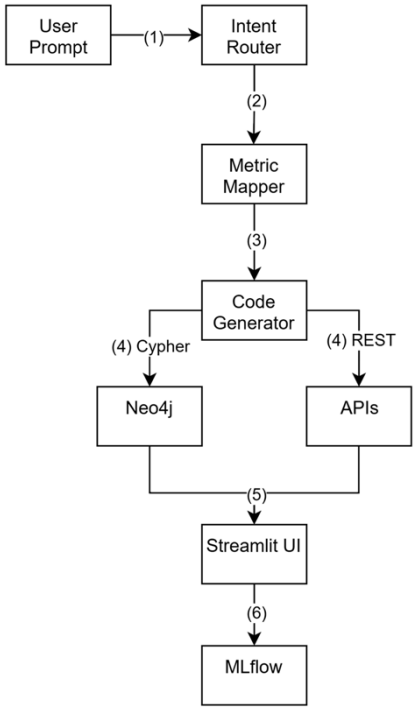


Figure 4. Overview of the Prompt-to-Metric system workflow.

confirmed the system's potential to bridge complex health analytics with the accessibility needs of non-technical stakeholders. Future work will involve expanding the evaluation to additional real-world application scenarios to assess the system's generalizability across diverse ecosystem types. We also plan to incorporate longitudinal analysis of metric time-series data by extending the system's storage to maintain historical values, enabling temporal trend analysis and proactive ecosystem governance. Additionally, an extended usefulness study [5] is planned to be carried out in collaboration with stakeholders and orchestrators of two different platform ecosystems.

References

[1] J. Bosch, "From software product lines to software ecosystems," in *Proceedings of the 13th International Software Product Line Conference*, in SPLC '09. USA: Carnegie Mellon University, Aug. 2009, pp. 111–119.

[2] C. Alves, J. Oliveira, and S. Jansen, *Understanding Governance Mechanisms and Health in Software Ecosystems: A Systematic Literature Review*. 2018. doi: 10.1007/978-3-319-93375-7_24.

[3] F. Fotrousi, S. A. Fricker, M. Fiedler, and F. Le-Gall, "KPIs for Software Ecosystems: A Systematic Mapping Study," in *Software Business. Towards Continuous Value Delivery*, C. Lassenius and K. Smolander, Eds., in Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2014, pp. 194–211. doi: 10.1007/978-3-319-08738-2_14.

[4] B. A. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report / Keele University, EBSE 2007-001, Jul. 2007.

[5] F. Davis and F. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Quarterly*, vol. 13, p. 319, Sep. 1989, doi: 10.2307/249008.