EasyChair Preprint

№ 16033

Chatting with the Factory: Capturing Industrial Data Context for Flexible AI

Nicolas Leberruyer, Mats Ahlskog, Jessica Bruch, Sarmad Bashir, Natalie Agerskans and Stéphane Bouchereau

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

November 2, 2026

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2026) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Conference on Industry of the Future and Smart Manufacturing

Chatting with the Factory: Capturing Industrial Data Context for Flexible AI

Nicolas Leberruyer[a,*], Mats Ahlskog[a], Jessica Bruch[a], Sarmad Bashir[b], Natalie Agerskans[a], Stéphane Bouchereau[c]

[a]*Department of Engineering Sciences, Mälardalen University, Eskilstuna, Sweden*
[b]*Department of Computer Science & Engineering, Mälardalen University, Västerås, Sweden*
[c]*Digital Technology & Operations, Volvo Group, Vénissieux, France*

Abstract

Manufacturing companies generate large volumes of production and operational data, however many industrial AI solutions are limited to a narrow scope and built for a single predefined task, which makes the same underlying data hard to reuse when new questions arise. A key reason is that the knowledge needed to interpret production data, including machine semantics, process relationships, shift structures, and disturbance logs, is largely tacit and resides with experienced personnel rather than in any reusable form. This paper investigates how such contextual knowledge can be captured at low effort and shared across people and AI applications for analysis. Based on a case study at a heavy-duty automotive manufacturing site, we characterize the capture of industrial data context as a knowledge-capture problem and derive technical requirements for addressing it. We then propose a three-stage architecture in which experts' explanations of how the production data and operational records should be read are turned into structured form, held in a knowledge graph, and exposed to AI agents through a standardized interface. Three domain experts explained how the data should be read in roughly 3.5 hours of ordinary conversation, and an AI coding assistant turned those explanations into structured form under developer review, so no expert had to write a data model or a query. The approach is implemented in a machining center and evaluated in its operational setting via a user evaluation with manufacturing employees, a manual KPI cross-check, and a factual benchmark across several off-the-shelf language models. The results indicate that the captured context supports consistent KPI computation and reliable natural-language access for users with different backgrounds and analytical needs, and that on factual KPI questions in this setting the language model behaves as a largely interchangeable component once the context is captured. The paper contributes a reframing of industrial data context, from a data problem to a tacit-knowledge-capture problem, and a proof-of-concept prototype of a context layer in industrial AI.

© 2026 The Authors. Check in the contract: Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the scientific committee of the 8th International Conference on Industry of the Future and Smart Manufacturing.

Keywords: Industrial AI; Industrial Data Contextualization; Manufacturing Analytics; Industrial Systems

* Corresponding author.
E-mail address: nicolas.leberruyer@mdu.se

1877-0509 © 2026 The Authors. Check in the contract: Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the scientific committee of the 8th International Conference on Industry of the Future and Smart Manufacturing.

1. Introduction

Artificial intelligence (AI) is increasingly used in manufacturing to support quality inspection, predictive maintenance, production planning, and process optimization [1, 2]. However, many industrial AI solutions are limited to a narrow scope: developed to solve one predefined problem, their outputs are limited to the analytical logic built into the original application and are difficult to reuse when new questions arise [3].

At the same time, modern production systems generate large volumes of machine and operational data [1]. In principle, the same data could serve operators troubleshooting a stop, maintenance personnel identifying recurring failures, and engineers analyzing shift-level performance, without rebuilding the underlying interpretation logic each time. In practice, raw production data is difficult to reuse because it does not directly expose its operational meaning [4, 5]. Answering even simple questions requires reconstructing how machine tags, process states, shift structures, and disturbance logs relate to each other, knowledge that is largely tacit: it resides with experienced personnel and is rarely captured in any form that others or AI systems can reuse [6, 7]. Existing approaches have clear limitations, as static documentation freezes knowledge without keeping it usable, interoperability standards organize structure while leaving interpretation implicit, and task-specific AI embeds this interpretation directly into the solution's code rather than exposing it as reusable context data. What is missing is a way to capture this interpretation knowledge once, keep it traceable to its source, and share it between people and applications.

This paper addresses this gap through the concept of industrial data context, defined here as the structured representation that captures the often tacit knowledge that links raw manufacturing data to operational meaning. The purpose of this study is to investigate how contextual knowledge can be captured with little effort and shared between people and AI applications for AI-assisted manufacturing analysis. Based on an industrial case study at a heavy-duty automotive manufacturing site, we identify the recurring challenge of capturing industrial data context, formulate practical requirements to address it, and design a three-stage architecture (context extraction, context graph, and standardized context access) that captures this knowledge and exposes it through a standardized AI agent interface [8]. The architecture is implemented in a real machining center using OPC UA data, shift logic, and disturbance logs integrated into a knowledge graph, and is assessed through a structured user evaluation with manufacturing employees, a manual KPI cross-check, and a factual benchmark across several language models.

The individual technologies used here, knowledge graphs, language models, and standardized tool interfaces, are established. The contribution of this paper lies in how they are combined and in what that combination demonstrates: (i) a reframing of industrial data context as a tacit-knowledge-capture problem rather than a data-availability problem, with technical requirements derived from field evidence; (ii) a capture procedure in which what the domain expert has to do is hold an ordinary conversation, with the formalization delegated to an AI coding assistant and checked by a developer, so that the resulting context files stay readable by both people and agents; and (iii) evidence from an operational setting that once context is captured in this form, the language model behaves as a largely interchangeable component for factual analysis, which moves the engineering effort from model selection to context capture. The remainder of the paper is organized as follows. Section 2 reviews work on industrial data context and the limitations of current approaches. Section 3 describes the case study design. Section 4 reports the field findings, the derived requirements, the architecture, its implementation, and its evaluation. Section 5 discusses the implications and limitations and concludes.

2. Literature Review

2.1. Industrial data context

Modern production systems generate large volumes of machine and operational data, but the analytical value of that data depends on knowledge that is rarely written down. Human skills such as visual inspection remain superior for ensuring product quality, but they cannot be formally shared with other workers because they involve a high level of tacit knowledge that exists only in skilled operators' internal cognition [7]. The same applies to interpreting machine signals: knowing what a mode code, a counter, or a particular stop pattern means for a specific machine is experiential knowledge held by the people closest to the process.

This distinction between explicit knowledge that can be written down and tacit knowledge that resides in experience is long established, and the conversion of tacit knowledge into explicit, shareable form, known as externalization, is regarded as the pivotal and most difficult step [6]. In manufacturing this step is now urgent: as companies move to

Industry 4.0, capturing and documenting the tacit knowledge of an aging workforce is a recognized problem, yet there are no systematic procedures to incorporate this knowledge into enterprise systems [9]. Specific knowledge about manufacturing processes is often not available in explicit form but mainly as tacit knowledge of experienced shop-floor workers, so a systematic approach to knowledge externalization and reuse is required to make this operational knowledge available [4].

The practical consequence is that AI initiatives in manufacturing are frequently limited not by data availability but by access to the interpretation knowledge needed to use that data. In this paper, the structured representation that captures this knowledge, linking raw machine signals to machine semantics, process steps, shift structures, and disturbance records, is referred to as industrial data context.

2.2. Limitations of the current approaches

Three families of approaches touch this problem, but each leaves the critical knowledge either uncaptured, frozen, or locked inside a single application.

Documentation captures knowledge but freezes it. The conventional response to knowledge loss is to write it down in manuals, A3 reports, and lessons-learned files. This writes some knowledge down, but what it produces is fixed and does not change. Work on knowledge externalization notes that such documents age and lose currency, and that they capture conclusions rather than the live interpretation logic, so a new question still tends to require the original expert [7, 6].

Interoperability standards organize structure but leave meaning implicit. Reference architectures such as RAMI 4.0, the IEC 62264 hierarchy, and the Asset Administration Shell as its digital-twin information model give manufacturing systems a shared structural vocabulary and support most Industry 4.0 integration efforts. Their strength is standardized structure, but they specify how assets and systems are organized and described rather than how production-specific signals should be interpreted when being analyzed, leaving open how a given signal is to be discovered and interpreted in practice [10]. Such a description can state that a tag exists and carries an integer; it does not explain that increments of that tag count as valid output only in one machine mode. The operational meaning of a signal must still be supplied separately. This limitation is also visible in quality-control settings, where raw machine data collected through standard protocols such as OPC UA lacks the semantic annotation required for automated cross-system interpretation [11].

Task-specific industrial AI builds interpretation logic but hides it. Most industrial AI systems are designed for one predefined task and optimized around a specific model, dataset, or dashboard [3, 2]. Reviews of predictive quality systems consistently find that models are built around fixed datasets, fixed quality targets, and specific process configurations, and cannot easily be transferred to new questions or adapted to different user needs [3]. The contextual knowledge that made the system work is embedded in code or configuration and is difficult to recover or reuse, so each new question tends to require a new application even when the underlying data already exists [12]. A related question is why the event stream is not simply analyzed directly, using clustering, anomaly detection, or process discovery [13]. Such methods can recover structure from the logs, but they cannot say what that structure means: a recurring idle cluster becomes actionable only once it is known that the cluster is a wait for raw material, which requires the disturbance categories and the shift calendar to be joined to it. Data-driven discovery and expert-knowledge capture are therefore complementary, and this paper addresses the second, because the first depends on it for interpretation.

All three approaches must reconstruct this knowledge for every new question, tool, or user.

2.3. A solution that can capture and share this knowledge

Recent technical developments make it possible to address this gap directly, by turning the capture and sharing of contextual knowledge into a low-effort byproduct of analysis rather than a separate documentation task.

A knowledge graph (KG) represents not only entities but the relationships between them, making it a natural container for the contextual links such as machine semantics, shift structures, and disturbance mappings that are otherwise tacit [2, 14]. The relevance of this idea to tacit knowledge is already recognized: recent work captures operators' best practices into an operational KG through an externalization process [9] and uses tacit expert knowledge to support shop-floor operators through KG-based assistance [4]. Reviews show, however, that manufacturing KGs

remain fragmented into task-specific graphs, with limited attention to low-effort capture or reuse across functions [2, 14].

Large language models (LLMs) let users ask questions in everyday language rather than through queries or dashboards, which is important where operators, maintenance staff, and engineers have different backgrounds [15]. Their value depends on the context available when they answer; supplying relevant facts from an external knowledge source at that moment, rather than relying only on what the model learned in training, substantially improves accuracy on knowledge-intensive tasks and supports traceable answers [16, 17]. This pattern, retrieval-augmented generation, has been applied to manufacturing quality control and troubleshooting [18], but typical applications retrieve over unstructured documents at query time and do not capture production-specific knowledge once into a traceable, queryable structure. The architecture proposed here is accordingly a graph-based variant of retrieval-augmented generation, in which the retrieval target is a queryable structure rather than a document corpus, so that what is retrieved is captured interpretation logic rather than text that has to be interpreted again. Finally, codified knowledge is only useful if others can reach it, yet existing solutions often expose context through application-specific integrations that tie it to one tool. The Model Context Protocol (MCP) [19] offers a standardized way for any AI agent to request this captured context through a single interface [8].

Each of these enablers addresses only part of the problem in isolation: knowledge graphs can hold contextual relationships but are typically built per task, language models provide access but depend on the context available to them, and standardized interfaces expose data without specifying what knowledge is captured. They have not been brought together as a single solution that captures contextual knowledge once and makes it available, through a shared interface, to different users and applications.

3. Methodology

This study follows an exploratory single-case study design combined with a prototype implementation and an initial user evaluation [20]. The aim is to identify recurring technical challenges in capturing industrial data context, implement a prototype architecture for capturing it, and assess whether the resulting system can support flexible AI-assisted analysis in a real production setting. This approach is suitable for examining a technical problem in its operational environment while assessing the feasibility of a proposed solution [20].

3.1. Research setting and case selection

The research was conducted over a four-year period at a heavy-duty automotive manufacturing company. The case company was selected because it offers a digitally mature production environment with repeated exposure to AI-related development efforts in factory operations, consistent with the profile of companies engaged in large-scale AI adoption at the shop-floor [1]. While the broader collaboration covered several AI initiatives, this study focuses on the recurring challenge of translating industrial data context into a reusable form for AI-assisted analysis. The empirical work reported here is centered on the final six months of the collaboration. The unit of analysis is the process by which industrial data context is identified, captured, and reused to support different analytical needs.

3.2. Data collection

Data were collected using multiple sources to support triangulation [20] and were organized into three phases, summarized in Table 1.

Phase I: Exploratory characterization

The first phase aimed to characterize how production data is currently interpreted in AI-related work and where recurring challenges arise. The lead author was embedded in the site and participated in AI-related projects, documenting recurring challenges related to data interpretation, reuse of prior analysis logic, and the effort required to relate raw data to production meaning. In addition, structured workshops were conducted with employees from manufacturing engineering, maintenance, and quality. Although initially focused on AI coding assistants, these sessions also revealed the repeated effort required to explain how machine data, process states, and related records should be translated for new analytical tasks. One additional workshop was held with senior digitalization experts from other industrial companies. Notes were taken in all sessions, and a follow-up survey ($n = 8$) was distributed to quantify recurring patterns.

Phase II: Knowledge capture and prototype implementation

Based on the challenges identified in Phase I and the related literature, a prototype architecture was developed and implemented on a machining center at the case company. Production data was collected through OPC UA and combined with the industrial data context required to interpret machine behavior in operational terms. That knowledge was captured in three sessions with the people who held different parts of it: an OT engineer responsible for PLC data logging, a cell expert responsible for the production cell, and a Lean production manager responsible both for Overall Equipment Effectiveness (OEE) follow-up and for the disturbance-log system and its follow-up. Each session was an ordinary conversation, in one case a walkthrough of the cell, recorded only as written notes. No expert was asked to produce a schema, a data model, or a query. The notes were then given to an AI coding assistant, which turned them into structured form and asked clarification questions that the developer answered. Section 4.2 reports the resulting effort and the review procedure in detail. This implementation served as a proof of concept for examining whether industrial data context could be captured once and then reused to support natural-language analysis for different user needs without repeated manual explanation of production-specific data relationships.

Phase III: Initial user evaluation

In the third phase, the prototype was made available to production employees through hands-on interaction with the user interface. Participants were invited to query the system in natural language about machine performance and production disturbances, with the chat interface using the Mistral Large 3 model as the active assistant model. Participants represented production, maintenance, and engineering roles with varying levels of experience in managing machine performance and production disturbances. This phase was intended to examine whether the prototype could make the same production data more accessible to users with different backgrounds, experience levels, and analytical needs. After the sessions, a survey (n = 12) was distributed to collect quantitative and qualitative feedback, including five-point Likert-scale items on perceived usefulness, perceived ease of use, and motivation to sustain use, along with free-text questions for comments and suggestions.

3.3. Data analysis

The analysis followed two steps. First, qualitative data from field notes, workshop notes, and survey comments were reviewed using thematic coding to identify recurring patterns related to data interpretation, context translation, reuse of production knowledge, and perceptions of the prototype [20]. Second, responses of the quantitative survey were summarized using descriptive statistics. Given the small sample size, these results are used only as indicative support for the qualitative findings.

Table 1. Summary of data collection sources

Method	**Participants/Duration**	**Primary data type**
Observation	8 AI projects / 4 years	Field notes
Workshops	10 participants / 8 hours total	Notes & survey
External workshop	5 industry experts / 2 hours	Notes
Expert knowledge capture	3 domain experts / 3.5 hours total	Notes → context files
Prototype implementation	3 weeks machining center data	OPC UA logs (6,928 entries) and disturbances (412 entries)
User evaluation	12 employees	Likert scale & free-text

4. Results

4.1. Findings from the field: the contextualization problem

Industrial data context remains embedded in local interpretation

The study shows that the context required to interpret production data is rarely represented in a reusable form. Survey results (n = 8) indicate that interpretation still depends heavily on direct explanation from the person who originally worked with the data: six of the eight respondents reported that a colleague would understand little or

nothing about their project work without a one-to-one conversation. Visualizations and project outputs may provide partial visibility, but the logic by which machine states, variables, and events should be interpreted remains unclear to others. The pattern is not limited to AI development projects. A conversation with the factory building utilities engineer revealed the same issue in daily operations: sensors and actuators were connected to monitoring dashboards, yet correct interpretation still depended on knowing how specific signals related to physical equipment behavior, operating conditions, and prior production experience. The core issue was not lack of data access, but lack of captured industrial data context reusable across analytical tasks.

The friction of repeated context translation

Because industrial data context is not captured in a reusable form, it must be translated again for each new analytical task, tool, or user. Survey participants rated the effort required to explain data context for an AI-supported session at a mean of 3.5/5, confirming it as a recurring source of friction: while AI coding assistants support local development and code generation, starting a new analysis still requires manually explaining machine manuals, operating logic, variable meanings, and other production-specific relationships. When that effort is too high, users fall back on manual analysis or narrowly scoped local solutions. Creating reusable contextual documentation remained uncommon in the current setting; only one participant had made their work accessible to others, by using an AI assistant to generate structured documentation. Without a dedicated mechanism for context capture, reuse depends on individual initiative rather than shared infrastructure.

Static documents do not provide reusable context

The workshops also revealed a failure mode related to the limited reusability of static documentation. Participants estimated that A3 reports, lessons-learned files, and project summaries often lose operational visibility within six months of creation, and one respondent noted that although documentation is considered important, most people do not read it in practice.

The same limitation was evident across the eight AI-related projects. Although capturing industrial data context was often more critical to success than model development itself, the interpretation logic was rarely embedded in the delivered technical solution; it remained buried in reports, scripts, or project-specific outputs. The consequence became clear when a second team was asked to continue development of an existing AI solution one year later. Despite having access to the previous team's code and final report, they could not move forward without a renewed explanation of how the production data should be interpreted, because the relevant data relationships were visible in neither the raw data nor the code.

Technical requirements for captured industrial data context

Five technical requirements emerge for reusable AI-assisted analysis in manufacturing.

1. **Low-effort context capture:** The process of capturing industrial data context must provide immediate practical value and should not depend on extensive manual modeling for each new use case.
2. **Assisted structuring:** Turning an expert's plain-language explanation into a usable structure should be supported by the tooling rather than done by hand, and existing operational documents and records should be reused where they already state part of the logic, reducing the need to restate the same knowledge manually.
3. **Flexible accessibility:** Any authorized user or AI agent should be able to query captured context without database expertise and without manually redefining the relevant production relationships for each session.
4. **Traceability and verifiability:** The contextual information used in AI-assisted analysis must remain traceable to its original source so that users can verify, inspect, and correct it when needed.
5. **Incremental extensibility:** New data sources should be added without rebuilding the entire solution, allowing the system to support new analytical needs over time.

4.2. The context layer

The five requirements derived above motivate the design of a context layer. We first describe the three stages and then their implementation in a real machining center.

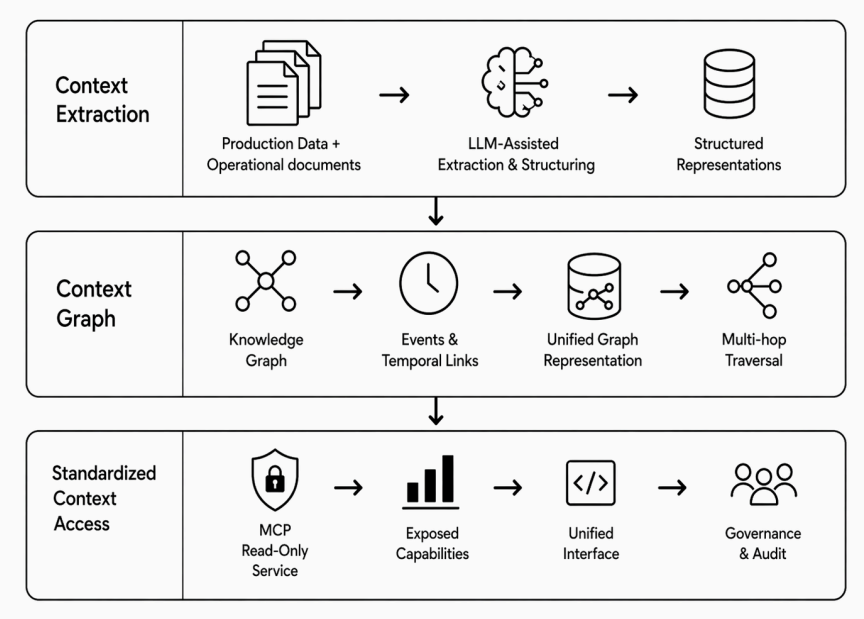


Fig. 1. Proposed three-stage architecture for capturing industrial data context.

The three stages

To address the limitations identified above, we propose a three-stage architecture for capturing industrial data context for AI-assisted analysis. Figure 1 illustrates how context extraction, graph-based codification, and standardized access transform raw production signals into a reusable, AI-accessible context layer. The objective is to avoid rebuilding the interpretation logic of production data for each new question, user, or application: relevant production relationships are captured once, stored in structured form, and made reusable for new tasks.

The first stage is **Context Extraction**. Its input is not the production data alone but the explanation of how that data should be read: what the people who work with the machine know about tag semantics, cycle-time definitions, shift schedules, and disturbance categories, written down in plain language and paired with the raw exports and operational records those explanations refer to. The stage turns that pairing into structured representations of the factory-specific relationships needed to interpret machine behavior. The second stage, **Context Graph**, represents the extracted information as a knowledge graph linking operational concepts, machine events, temporal relations, and contextual definitions; unlike a document repository, it supports multi-hop traversal (navigating across multiple connected relationships), so a single query can connect a machine event to its production context, shift schedule, and operator-recorded disturbance explanation. The third stage, **Standardized Context Access**, exposes the graph via an MCP-based read-only service, separating client applications from the underlying implementation so that multiple systems reach the same context layer through a consistent interface. This eliminates the need for AI agents to reconstruct production logic from raw data, since the graph already explicitly captures machine semantics, temporal relationships, and disturbance mappings.

Architecture implementation in the machining center case

First stage: Context extraction. The proposed architecture was implemented on a large-format machining center, anonymized as MC-01, connected via OPC UA to a local logging service. The case covers a 22-day observation window (11 March–1 April 2026), during which the system recorded 6,928 timestamped events. Each event captured a change in one of three tag values: **Mode** (1 = Production, 2 = Setup, 3 = Idle/Stop), **Program** (active machining program, encoding both article identifier and tempo, i.e. the machining operation sequence, e.g. `ART-XX_T2`), and **Counter** (an integer representing the active process step).

Raw machine data alone is insufficient: interpreting the event stream requires knowing what the tags mean, how the machine is scheduled, and what counts as normal throughput. This knowledge was held by three people rather than in any document, and was captured as summarized in Table 2. In each session the expert's only output was an

explanation, which the researcher wrote down in plain language. Those notes, together with the raw tabular exports, were then given to an AI coding assistant (GitHub Copilot), which proposed a structured form and asked clarification questions where the notes were incomplete. The developer answered these and corrected proposals that conflicted with known machine behavior, for instance that a `Counter` increment is valid output only when `Mode = 1`, and that the tempo component of the program identifier tracks the active article. Over a few such exchanges this produced two context files, a machine-and-tag ontology and an OEE calculation specification, each confirmed by the developer before the graph was built and each extended later as further sources were added, as Table 2 shows. What distinguishes this procedure is how the work is divided: the domain experts contributed roughly 3.5 hours of conversation and no technical work, while turning it into a schema required an AI coding assistant and a developer able to recognize an incorrect proposal, but no knowledge-engineering specialist. The coding assistant is a development tool and not part of the running system, so what matters here is this division of work rather than the particular assistant used. Effort figures are reconstructed afterwards from session records and development history rather than measured at the time, and should be read as approximate.

Table 2. Context capture by source: expert input, structuring step, and resulting context file

Knowledge source	**Expert input**	**Structuring step**	**Context file**
PLC tag semantics and logging behavior	OT engineer, ≈1 h meeting	Notes to assistant; clarified mode/counter validity	Machine-and-tag ontology
Cell structure and machine roles	Cell expert, ≈1.5 h incl. walkthrough	Written description; assistant follow-up questions	Tag ontology extension
OEE follow-up and disturbance reporting	Lean production manager, ≈1 h meeting	Notes to assistant; clarified shift calendar	OEE calculation specification
Disturbance log (added later)	Same Lean production manager, owner of the log system; follow-up only, a few minutes	Tabular export and column description; clarified disturbance categories; incremental schema dialogue	Extended graph schema

Adding the disturbance log later used the same procedure: the Lean production manager's earlier explanation carried into the schema extension, which the developer checked before it was applied. The assistant can propose a relationship that looks reasonable but is wrong, and so can an explanation that leaves out something everyone treats as obvious. The shift calendar, for instance, was captured correctly as Monday 06:00 to Friday 14:00, but nobody stated that these are local times while the machine log is timestamped in UTC, which displaced planned production time by one hour, or two under daylight saving. The context file was consistent with itself, so reading it would not have revealed this; only the figures at the start and end of shifts did, and all results reported here postdate the correction. Every proposal was therefore checked against known machine behavior and against the figures produced when shifts and programs changed, and the loading routine was written so a corrected interpretation can be re-run over the whole dataset.

Second stage: Context graph. We implemented a loading routine to populate and maintain the knowledge graph in Neo4j. A knowledge graph is a way of organizing meaning rather than a specific storage technology: the same relationships can be held in a relational database, a document store, or a vector store, although a graph database or triple store supports traversal most directly. A plant already running a relational system can therefore capture context this way without replacing it.

The graph is built around one central node type, the `LogEntry`: one node for each recorded change in mode, program, or counter. Each entry points to the next, and the time between the two is stored on that link, so the time a machine spent in any state can be read straight from the graph instead of being recalculated. Each entry is also connected to the machine it came from, to the `Shift` it falls within, which marks planned production time, and to the `Program` running at that moment, which supplies the ideal cycle time used for Performance. Disturbances are linked to a shift rather than to individual log entries, because each one carries the reason the operator wrote, a duration, and the machine or system affected, but no start time. They are therefore matched to machine states by day rather than to a particular idle period.

The disturbance nodes come from a second operational source added after the initial build, the *shift disturbance log*, which operators maintain in a structured database and which records the scope of each event, whether machine-specific, raw-material-related, or cell-wide. Adding it meant the graph could show not only what the machine did but also why it stopped.

Third stage: Standardized context access. To meet the flexible accessibility requirement, the knowledge graph is exposed through a lightweight MCP-based server, a single access point through which any compatible AI agent can query the production knowledge base without custom database integration. To protect data integrity the server is read-only. It exposes thirteen tools in five groups: six that return OEE or one of its components for a day, an hour, or a date range; two that break a day down by hour; two for disturbance lookup, linking idle time to documented stop reasons; one that lists the dates for which data exists; and two for open-ended investigation, namely a chronological state-change log (`get_state_transitions`) and a read-only query interface (`execute_read_query`) for iterative exploration, much as a human analyst would narrow down an anomaly.

Captured context reaches a client in two ways. The tool descriptions returned by the server carry the graph schema, the intended use of each tool, and warnings about how the data must be read, such as the absence of start times on disturbance records, so any connecting client receives them. The dashboard then adds a system prompt of its own, shown in Table 3, which restates definitions already held in the graph, sets agent policy, requiring a tool call before any figure is reported and a fixed refusal outside this cell and its data, and adds the one thing held nowhere else: the order in which to investigate an abnormal KPI.

Table 3. Extract from the agent system prompt

Scope. You may only answer questions about: OEE metrics and their calculation; production data for this cell (dates, shifts, disturbances, log entries); how this dashboard works. If a question is outside this scope, respond only with: "I can only help with questions about the OEE dashboard and its production data."
Data context. Schedule: 24/5 (Mon 06:00 → Fri 14:00). Mode 1 = Production, Mode 2 = Setup, Mode 3 = Idle/Stop. Always use a tool to fetch data before answering questions about specific metrics or dates. When a KPI looks abnormal, use `get_state_transitions` to trace the timeline, then `execute_read_query` to drill deeper. Disturbances for the whole cell are stored as nodes linked to shifts, each with a reason, a duration, and the machine affected; they have no exact start time, only a date and total duration. Use `get_disturbances_for_date` when availability is low to identify likely causes.

Three properties follow from this configuration. First, figures are computed from the graph rather than estimated by the model, because the prompt requires a tool call before any metric is reported. Second, the context layer also sets the limits of what the agent will answer: a question it cannot support returns a fixed refusal rather than a plausible-looking number. Third, the interface lists every tool the agent called and the arguments it used, and for open-ended investigation it shows the Cypher query that was executed, so a user can see which part of the context layer produced a figure without leaving the dashboard.

The demonstration interface: "Chatting with the factory". A dashboard serves as the human-machine interface, presenting four OEE gauges (Availability, Performance, Quality, OEE) configurable over arbitrary date ranges, alongside daily and hourly trend visualizations. A natural-language chat interface, powered by an LLM and supported by the MCP toolset, enables users without database expertise to query production performance in natural language. For this pilot, Quality was fixed at 100% because reliable scrap data was not yet available, so reported OEE was simplified to Availability × Performance; the full OEE structure is retained in the schema and interface so a quality source can be added later without redesign. Table 4 shows a representative interaction.

The user specifies no shift structure, tag semantics, article-tempo mapping, or disturbance category: the industrial data context is captured at ingestion and reused across queries.

4.3. Evaluation

Pilot observations

Three observations emerged from the pilot with direct implications for industrial deployment. First, the context layer was reused across every query, with no repeated explanation of machine semantics, shift logic, or disturbance categories, so the work identified earlier was done once instead of being repeated for every analysis. Second, the layer

Table 4. Example interaction with the chat interface

Query. "Why was OEE low on March 20th?"

Tools called, shown to the user. `calculate_oee`, `get_state_transitions`, `get_disturbances_for_date`, each with the date as argument.

Answer. OEE of 13.9%, driven almost entirely by Availability. The agent attributed the loss to a cell-wide rebuild recorded against that date, with staffing and operator-maintenance entries as secondary contributors, and confirmed the explanation against the state log, which showed the machine in Setup or Idle for most of the period with roughly two hours in Production. The user did not have to know what Mode 3 means, where disturbances are stored, or how the two sources relate.

extended without being rebuilt: introducing the operator disturbance log extended the schema incrementally, and the same layer then served different needs, with maintenance personnel asking about repeated stop reasons, operators about machine behavior during a shift, and engineers about performance across articles or periods.

The third observation concerns where flexibility comes from. Flexibility comes from open querying over a well-modeled graph, not from a fixed tool set. Twelve of the thirteen tools return predefined results and, on their own, answer only the questions someone thought of in advance. The flexibility we observed comes from the thirteenth, the open query tool `execute_read_query`, working over a graph in which machine semantics, timing, and disturbances are all spelled out. When asked about an availability decrease, the agent did not call a pre-built view; it composed its own query to inspect the sequence of machine states around the period, located the loss, and explained it in natural language. Storing state durations directly on graph edges is what made this possible: the agent can compute time spent in each state on demand, rather than relying on a separate hard-coded formula for each anticipated question. The predefined tools make the common KPIs convenient; being able to query the context layer freely is what lets the system answer questions nobody designed it for.

KPI calculations check

To complement the qualitative findings, Availability and Performance were recomputed manually from the raw machine event log for five selected production days, using the same shift schedule and ideal cycle-time assumptions as the prototype. The manually computed values aligned with the application outputs for all five days, indicating that KPI computation is internally consistent on the studied cases. This is a self-consistency check against the same assumptions since no alternative working system for monitoring this machine's OEE was available during the study period.

Answering over the context layer across models

A practical concern for deployment is whether the context layer can be read by different language models without per-vendor engineering. Because access is mediated by the MCP interface rather than model-specific SDK functions, any MCP-compatible model can in principle consume the same context. We tested this directly, varying model size to see whether smaller (cheaper) models could answer factual KPI questions as reliably as larger ones. Five off-the-shelf models from two vendors were evaluated, with the same system prompt and tool set held constant, on a 12-question factual benchmark (8 percentages, 3 dates, 1 boolean) derived from the manually cross-checked KPI facts. Accuracy and latency were computed on successful numeric responses; reliability is reported as successful runs over total attempts including retries (Table 5).

Table 5. Factual KPI benchmark over 12 questions (accuracy/latency on successful numeric responses)

Model	Accuracy	Median latency (s)	Successful / total	Est. monthly cost (EUR)
Gemini 2.5 Flash	1.0	4.2	12/13	0.11
Gemini 2.5 Flash-Lite	1.0	8.9	8/36	0.02
Mistral Large 3	1.0	2.7	12/12	0.11
Mistral Medium 3.5	0.83	2.3	12/12	0.35
Mistral Small 4	1.0	2.0	12/12	0.02

Accuracy was high and largely independent of model choice: four of five models answered all questions correctly and the fifth missed two (one deliberately ambiguous yes/no item and one numeric answer off by about 3.7%). We read this cautiously: with only 12 questions on a single machine, the result shows that the context layer supports accurate factual answering across a range of models in this setting, not that any model is universally interchangeable. The clearest differences were practical. Response time ranged from 2.0 s to 8.9 s per question, and reliability varied substantially: the three Mistral models answered every question on the first attempt, whereas the cheapest Gemini model required 36 attempts to obtain 8 complete runs, due to repeated "service unavailable" errors from the provider. Any candidate model must therefore be tested for speed and reliability in the target setting before it is used for live, on-demand querying. Cost, by contrast, was negligible: for a single machine the estimated monthly usage cost was at most EUR 0.35 (at June 2026 provider prices), small relative to the development and licensing costs of purpose-built analytics tools.

Beyond factual KPI questions, we gave each model an open-ended prompt asking what concrete actions would improve OEE, repeated three times per model. All models produced consistent recommendations organized around the OEE components, but asked once they mostly gave advice, pointing to the tools a user could run. Asked to carry the analysis out, the agent fetched the disturbance record for the whole period and the state log for one date, and connected a multi-hour idle window to a machine collision logged that day. It also attached an expected improvement percentage to each proposed action, which nothing in the retrieved data supports; the useful part is the links it finds, not the numbers it predicts. Connecting machine events to recorded stop reasons is what the company's existing IoT platform could not do, because it only received data from the PLC.

Manufacturing employees' feedback

The quantitative results, summarized by work position in Table 6, indicate high overall acceptance, with aggregate means of 4.25 for Perceived Usefulness, 4.39 for Perceived Ease of Use, and 4.53 for Motivation to Sustain Use. Operational roles such as maintenance and production management reported solid perceived usefulness, both averaging 4.17. Respondents in IT and Lean functions were more critical, specifically regarding the system's current capacity to provide complex actionable insights. Trainees, representing users with limited experience in production analysis, reported the highest Perceived Ease of Use (4.67), together with a high Motivation to Sustain Use (4.67) and Perceived Usefulness (4.57). Given the small per-group sample sizes (in particular $n = 1$ for production management and $n = 2$ for IT and Lean), the group-level figures should be read as indicative only.

Table 6. Mean survey scores by work position ($n = 12$)

Work Position	**n**	**Perceived Usefulness**	**Perceived Ease of Use**	**Motivation to Sustain Use**
Production Manager	1	4.17	4.00	4.33
Maintenance	4	4.17	4.33	4.75
Digital & IT / Lean	2	3.67	4.00	3.83
Trainee	5	4.57	4.67	4.67

Three qualitative themes emerged from the open-text responses. First, the system was perceived as a significant efficiency enabler, with users highlighting its ability to generate quick ad-hoc reports and its role as a time-efficient and smooth alternative to manual data exploration. Second, participants identified high potential for analytical depth, noting that the interface helped explain production issues at both basic and advanced levels and supported investigations into why availability was low. Finally, respondents emphasized the critical need for trust and transparency, stating that long-term reliance in a production environment requires clear visibility into the sources of data and concrete references for generated insights. An additional theme emerged from field observations: three production managers noted that broader access to production analysis could motivate operators to take a more active role in continuous improvement, since analytical insights currently originate from engineers and managers rather than from the operators closest to the machines.

5. Discussion and Conclusion

The findings point to a recurring limitation in industrial AI practice: the knowledge needed to interpret production data, such as machine-tag semantics, shift logic, and disturbance mappings, tends to remain tacit or embedded in local documents and individual expertise rather than in a reusable, shareable form. Treating this as a knowledge-capture problem rather than a data problem reframes what the architecture contributes. It performs the externalization step regarded as pivotal and most difficult [6, 9], at a cost that falls almost entirely outside the expert's working day: three conversations, no data model written by any expert, and the structuring work delegated to an AI coding assistant under developer review. Where task-specific industrial AI hides its interpretation logic in code [3, 12], standards organize structure while leaving factory-specific meaning implicit [10, 11], and static documentation freezes conclusions [7], the context layer keeps the live interpretation logic queryable and readable by both people and agents, so each new question no longer requires the context to be reconstructed. Low-effort capture does not remove the need for validation against machine behavior, but it changes who performs it and how often: from an expert re-explaining context for each analysis to a developer checking a proposal once per source.

The user evaluation supports the practical argument. Maintenance reported the highest motivation to sustain use, consistent with the value of on-demand access to contextualized data for troubleshooting [12], and broader access may move analytical ownership from engineers and managers toward the operators closest to the machine. IT and management respondents were more critical of the depth of insights currently available, while trainees' high ease-of-use scores suggest the interface can lower barriers for less experienced users. Across roles, trust and transparency emerged as a shared prerequisite, in line with research identifying interpretability and source traceability as necessary conditions for deployment [21]. This feedback led directly to a design change: the interface now shows the tools and queries behind each answer, so a figure can be traced back to the query that produced it. Three things therefore support trust in an answer. Figures come from the graph rather than from the model. Questions outside of the scope are refused. And the steps taken to reach an answer are visible. What is still missing is a pointer to where each definition came from.

The quantitative validation adds two results. The prototype's KPI outputs agreed with manual recomputation on all five cross-check days, and the factual benchmark showed that accuracy did not depend on which language model was used. This agrees with what the retrieval-augmented generation literature reports: on knowledge-intensive tasks, accuracy comes from the external source the model reads rather than from the model itself [16, 17]. Here that source is a queryable graph [14] reached with a standard interface [19, 8] rather than a collection of documents [18], which supports treating the language model as a largely interchangeable part sitting above a stable context layer. The differences that did emerge, latency and reliability, are deployment considerations rather than properties of the architecture, and cost was negligible even at scale. An existing commercial IoT platform used at the case company provides a reference point: it could report only previous-day figures, and since it only receives PLC data it could not explain why a KPI was low. The context layer brings machine events and disturbance records together in one place that can be queried, so an off-the-shelf model can reason across both.

The study has clear limitations. The evaluation rests on a single industrial case, one machine, a static 22-day dataset, and a small user sample ($n = 12$), and Quality was fixed at 100% for lack of reliable scrap data. Capture effort is reconstructed afterwards rather than measured, and covers the conversations and the structuring step only, not the implementation work that followed. Disturbances are recorded with a date and a duration but no start time, so machine states and stop reasons are matched at day resolution rather than aligned in time. We ran no controlled comparison, either against the existing platform or against data-driven alternatives such as process mining or unsupervised anomaly detection, so our claim that these methods and expert-knowledge capture complement each other, because the patterns they find still need captured context to say what they mean, remains an argument rather than a result. The user evaluation was carried out before the tool and query trace was added, so the reported acceptance scores reflect the interface without it. The workflow that directs the agent from an abnormal KPI to the state log and then to graph queries sits in the dashboard's own prompt rather than on the server, so another client would receive the tools and their descriptions but not that policy. When asked open-ended questions, the agent adds expected improvement percentages that the data does not contain. The trace makes these visible to the user, but does not stop the agent from producing them. Finally, the implementation captures neither the logic of a whole production system nor the rationale for past actions. These results should be read as evidence of feasibility rather than general validation.

Future work should extend the context layer with maintenance, quality, and work-instruction records, to test whether incremental schema extension scales as sources accumulate, and move from the static dataset to continuous event-driven OPC UA ingestion so the graph can be maintained and evaluated live. Two additional steps follow from the limitations above: pointing to the context file behind each definition, so that a user can check not only how a figure was computed but how the underlying term was defined, and a controlled comparison against data-driven discovery methods on the same event stream. MCP also lets a server publish prompts alongside tools, so that the investigation procedure travels via the same interface as the data; it does not, however, force a model to follow it.

In conclusion, the central challenge to flexible industrial AI in this case was not a lack of data but a lack of captured, shareable knowledge. Capturing industrial data context as a byproduct of ordinary conversation, rather than as a modeling exercise, produced a layer that was consistent with manual KPI recomputation, usable by five off-the-shelf language models without per-model engineering, and able to bring machine events together with recorded stop reasons in a way the existing platform could not. The same knowledge was readable by employees and usable by agents, which is what makes it shareable and not just reusable. Initial user feedback confirms the operational relevance of the approach and the importance of trust and transparency to any future deployment.

Acknowledgements

This research work has been funded by the Knowledge Foundation within the framework of the INDTECH Research School project (Grant Number 20220132), participating companies and Mälardalen University.

References

[1] World Economic Forum, Global lighthouse network: Adopting AI at speed and scale, Tech. rep., World Economic Forum (2023).
[2] Y. Wan, Y. Liu, Z. Chen, C. Chen, X. Li, F. Hu, M. Packianather, Making knowledge graphs work for smart manufacturing: Research topics, applications and prospects, Journal of manufacturing systems 76 (2024) 103–132.
[3] H. Tercan, T. Meisen, Machine learning and deep learning based predictive quality in manufacturing: a systematic review, Journal of Intelligent Manufacturing 33 (7) (2022) 1879–1905.
[4] L. Hoerner, M. Schamberger, F. Bodendorf, Using tacit expert knowledge to support shop-floor operators through a knowledge-based assistance system: Using tacit expert knowledge to support shop-floor operators through a knowledge-based assistance system, Computer Supported Cooperative Work (CSCW) 32 (1) (2023) 55–91.
[5] L. Leoni, M. Ardolino, J. El Baz, G. Gueli, A. Bacchetti, The mediating role of knowledge management processes in the effective use of artificial intelligence in manufacturing firms, International Journal of Operations & Production Management 42 (13) (2022) 411–437.
[6] I. Nonaka, H. Takeuchi, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, Oxford University Press, 1995.
[7] T. L. Johnson, S. R. Fletcher, W. Baker, R. L. Charles, How and why we need to capture tacit knowledge in manufacturing: Case studies of visual inspection, Applied ergonomics 74 (2019) 1–9.
[8] R. Sapkota, K. I. Roumeliotis, M. Karkee, Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges, Information Fusion (2025) 103599.
[9] E. Fenoglio, E. Kazim, H. Latapie, A. Koshiyama, Tacit knowledge elicitation process for industry 4.0, Discover Artificial Intelligence 2 (1) (2022) 6.
[10] T. Burns, J. Cosgrove, F. Doyle, A review of interoperability standards for industry 4.0., Procedia Manufacturing 38 (2019) 646–653.
[11] D. Shi, P. Liedl, T. Bauernhansl, Interoperable information modelling leveraging asset administration shell and large language model for quality control toward zero defect manufacturing, Journal of manufacturing systems 77 (2024) 678–696.
[12] D. Romero, J. Stahre, M. Taisch, The operator 4.0: Towards socially sustainable factories of the future, Computers & industrial engineering 139 (2020) 106128.
[13] W. M. P. van der Aalst, Process Mining: Data Science in Action, 2nd Edition, Springer, Berlin, Heidelberg, 2016.
[14] G. Buchgeher, D. Gabauer, J. Martinez-Gil, L. Ehrlinger, Knowledge graphs in manufacturing and production: a systematic literature review, Ieee Access 9 (2021) 55537–55554.
[15] S. Kernan Freire, C. Wang, M. Foosherian, S. Wellsandt, S. Ruiz-Arenas, E. Niforatos, Knowledge sharing in manufacturing using llm-powered tools: user study and model benchmarking, Frontiers in Artificial intelligence 7 (2024) 1293084.
[16] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in neural information processing systems 33 (2020) 9459–9474.
[17] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, arXiv preprint arXiv:2312.10997 (2023).
[18] J. A. H. Álvaro, J. G. Barreda, An advanced retrieval-augmented generation system for manufacturing quality control, Advanced Engineering Informatics 64 (2025) 103007.
[19] Anthropic, Model context protocol (mcp) specification, https://modelcontextprotocol.io, accessed: 2026-04-02 (2024).
[20] R. K. Yin, Case study research and applications, Vol. 6, Sage Thousand Oaks, CA, 2018.
[21] G. Mentzas, M. Fikardos, K. Lepenioti, D. Apostolou, Exploring the landscape of trustworthy artificial intelligence: Status and challenges, Intelligent Decision Technologies 18 (2) (2024) 837–854.