What Are Your Agents Up To? Agent Behavior Mining in Action

Maximilian Körner[1], Hoang Vu[1], Adrian Rebmann[1], Gabriel Kevorkian[1], Tobias Wuttke[1], Michael Perscheid[1], Gregor Berg[1] and Timotheus Kampik[1,2]

[1]SAP, Berlin, Germany
[2]Umeå University, Umeå, Sweden

Abstract

Generative AI agents are increasingly used to automate business processes, yet their behavior is opaque, creating operational risks around cost, compliance, and drift. This paper demonstrates how *agent behavior mining* (ABM) can surface insights into agent performance, execution patterns, and compliance issues using existing process mining tools, exemplified by SAP Signavio Process Intelligence. The analysis is based on the ABM data model, which captures GenAI-agent-specific events and relevant attributes. As this data model is XES-compatible, existing process mining capabilities, including process discovery, conformance checking, performance analysis, and variant exploration, can be applied directly to GenAI agent behavior without additional engineering effort.

Keywords

generative AI, agents, process mining, agent behavior

1. Introduction

Organizations increasingly deploy generative AI (GenAI) agents to automate business processes, ranging from customer-facing interactions to back-office operations. Unlike rule-based automation such as robotic process automation (RPA), a GenAI agent's behavior is not fully determined by its code; we can observe inputs and outputs, but its task execution and inference are non-deterministic and remain opaque. This opacity translates into concrete operational risk: token-inefficient reasoning paths increase cost; repeated tool failures go unnoticed because no systematic view spans multiple executions; agents that progressively deviate from intended workflows are indistinguishable from legitimately adaptive ones without a behavioral baseline; and compliance-relevant steps, such as always invoking a deterministic tool to calculate totals before confirming an order, can be silently skipped.

Closing this gap requires analyzing agent executions, captured as structured data, at a population level across many cases, rather than just debugging individual runs, as current observability tools offer. Building on the agent-agnostic foundations of agent system mining [1, 2], our BPM '26 paper [3] introduces agent behavior mining (ABM), a data model that captures the actions taken by GenAI agents, such as calling a large language model (LLM) or invoking a tool, along with respective sets of relevant attributes, as XES-compatible process event logs. Each conversation constitutes a case with structured events for the observed actions, their attributes, and metadata.

While [3] defines the ABM data model and empirically validates its governance value, this paper demonstrates the analyst experience: how process mining software, using the example of SAP Signavio Process Intelligence (SAP SPI), can be applied to ABM-conformant agent event logs, turning an opaque black box into a visible, measurable process. This shows that process mining tooling already familiar to BPM practitioners can be used to obtain behavioral insights into GenAI agents.

Proceedings of the BPM 2026 Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Forum (BPM 2026), Toronto, Canada, September 27th to October 2nd, 2026.

maximilian.koerner@sap.com (M. Körner); h.vu@sap.com (H. Vu); adrian.rebmann@sap.com (A. Rebmann); gabriel.kevorkian@sap.com (G. Kevorkian); tobias.wuttke@sap.com (T. Wuttke); michael.perscheid@sap.com (M. Perscheid); gregor.berg@sap.com (G. Berg); timotheus.kampik@sap.com (T. Kampik)
0000-0001-9274-3271 (M. Körner); 0009-0002-1656-7207 (H. Vu); 0000-0001-7009-4637 (A. Rebmann); 0009-0005-6589-8135 (G. Kevorkian); 0009-0003-1977-7813 (T. Wuttke); 0000-0002-2331-6775 (M. Perscheid); 0009-0005-6237-3747 (G. Berg); 0000-0002-6458-2252 (T. Kampik)

published 2026-09-28

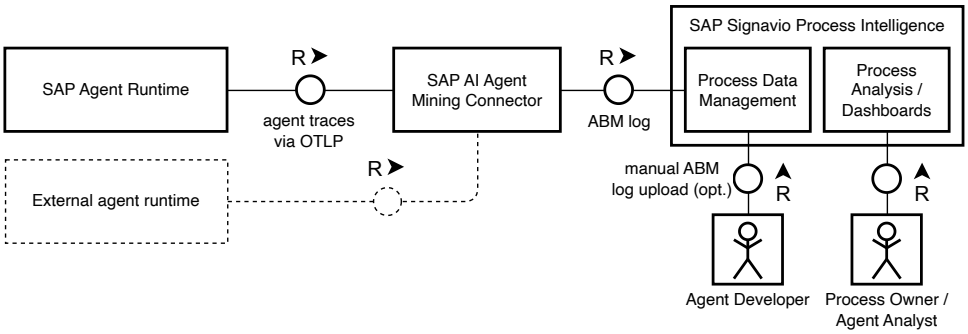

Figure 1: High-level architecture of agent behavior mining in SAP Signavio Process Intelligence. Supporting external agent runtimes (dashed) is a future direction.

2. The Limits of Current Agent Observability

Existing tools for agent observability, such as LangSmith, LangFuse, and Arize Phoenix,[1] were designed for agent developers who need to debug individual executions: they expose each step of a single run in detail, including inputs, outputs, and recorded metadata, enabling developers to trace errors and reconstruct the decisions made in a specific case [4]. Such a per-execution perspective, however, is inadequate for operational governance. Governance-oriented analysis is concerned with understanding how agents behave across the majority of cases: what the typical execution flow looks like, which paths are most frequently taken, and whether execution variants deviate from expected behavior, such as how frequently an agent omits a required step or which execution variants systematically exceed the average cost, none of which a trace debugger can answer.

Process mining tools, by contrast, natively support such analyses, including process discovery, conformance checking, performance analysis, and variant exploration. The ABM data model presented in previous work [3] extends the XES event log standard with event types and attributes specific to common GenAI agent actions. Among others, it defines the event types `call_llm`, `execute_tool`, and `transfer_to_agent`, each with associated attributes, such as token consumption, tool arguments, and the exchanged message. XES compatibility ensures that any process mining software that handles XES can ingest ABM-conformant logs without modification, enabling these tools to produce interpretable process graphs and compute agent-specific performance metrics without custom configuration.

3. Agent Behavior Mining in Practice

In this section, we briefly outline how agent behavior mining is implemented in SAP Signavio Process Intelligence[2] and provide an analysis walkthrough demonstrating how agent behavior mining with standard process mining tooling can yield actionable insights.[3]

3.1. Implementation of Agent Behavior Mining in SAP Signavio

Agent behavior mining, as implemented in SAP SPI, moves agent execution data through three stages, illustrated in Fig. 1: (i) the agent runtime emits OpenTelemetry traces per execution, which are collected and transferred using the OpenTelemetry Protocol (OTLP); (ii) the AI Agent Mining connector[4] normalizes those traces into ABM-conformant event logs; (iii) the logs are then automatically ingested into SAP SPI. The connector is designed to handle the different OpenTelemetry span structures emitted by various agent frameworks, ensuring uniform downstream processing regardless of origin. Agent developers may also upload an ABM-conformant XES log directly to SAP SPI for one-off analysis.

[1] https://www.langchain.com/langsmith-platform, https://langfuse.com, https://arize.com/phoenix
[2] https://www.signavio.com/products/process-intelligence/
[3] A screencast can be found here: https://doi.org/10.6084/m9.figshare.32844872
[4] https://architecture.learning.sap.com/docs/ref-arch/3c8d50

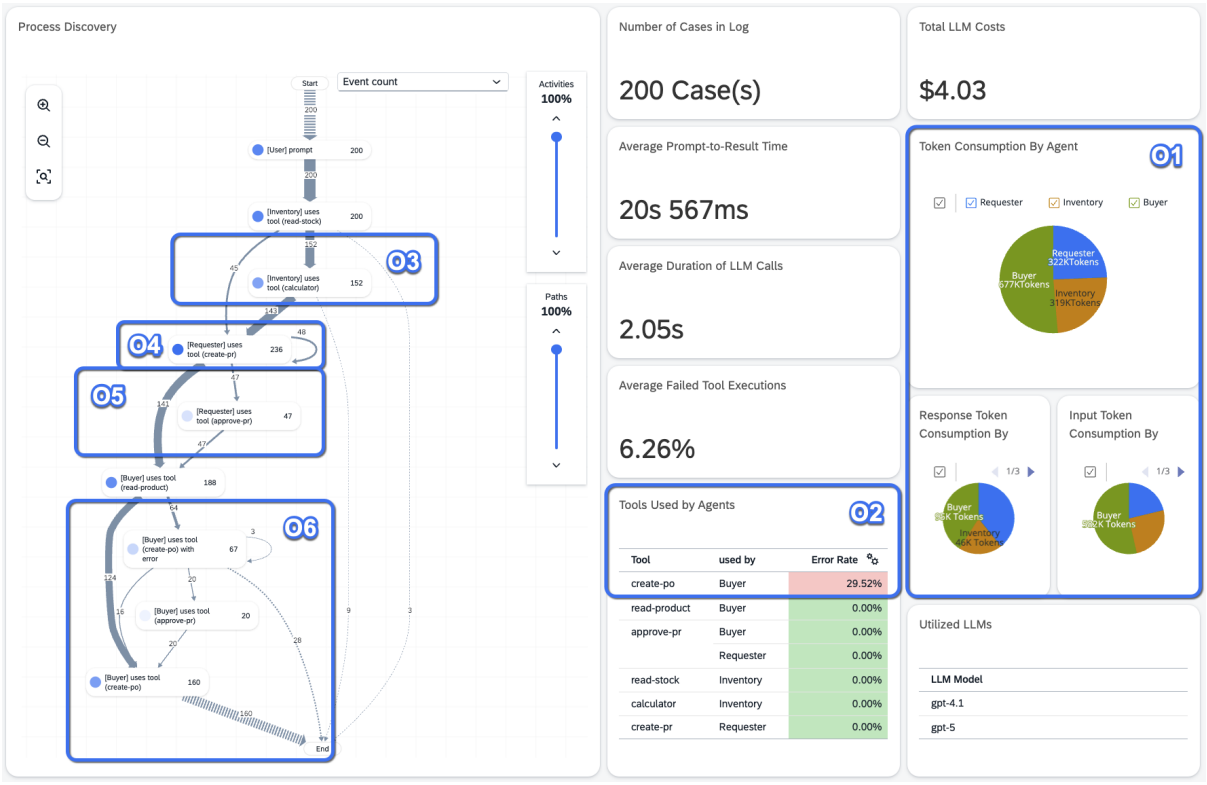

Figure 2: Process Mining Dashboard for the procurement agent system showing some ABM-specific analysis widgets, such as the distribution of token consumption. Observations **O1** to **O6** are annotated in blue.

Predefined dashboards for agent analysis are provided out of the box, configured against the event types and attributes specified by ABM, such as `call_llm` events with `ai:*_tokens` attributes, such that no manual setup is required. The widget infrastructure in SAP SPI supports further tailoring of the analysis: analysts can add, filter, or rearrange widgets to reflect the specifics of a given agent and its domain. Because ABM's event types and attributes apply uniformly across agent implementations, dashboards are reusable across different agent systems.

3.2. Agent Behavior Mining Analysis Walkthrough

The analysis is illustrated using a three-agent procurement system connected to the company's procurement software via a *model context protocol* server, with 200 recorded execution traces.[5] The system comprises an *Inventory Agent* that checks current stock levels and calculates reorder quantities; a *Requester Agent* that handles purchase requisitions (PRs); and a *Buyer Agent* that determines the optimal order quantity and issues the corresponding purchase order (PO), with discretion to order more when a product carries a high rating and bulk pricing applies. While the walkthrough focuses on the procurement case, the shown features and analyses can be applied identically to any ABM log.

3.2.1. Performance Overview

The analysis dashboard provides a set of metrics derived directly from standard ABM, including durations and token-based costs. Across the 200 executions, the agent system operated for an average of 21 seconds end-to-end, with individual LLM calls averaging 2.05 seconds, at a total LLM cost of $4.03. Token consumption broken down by agent reveals a first Observation (**O1**): the Buyer Agent consumes 677k tokens in total (582k input, 95k output), nearly twice the 322k consumed by the Requester Agent (231k input, 91k output) and the 319k consumed by the Inventory Agent (274k input, 46k output).

[5]The ABM event log and dashboard configuration are available at https://github.com/a-rebmann/ABM/tree/main/procurement-demo.

The widgets show that the discrepancy is mainly due to input tokens. A likely explanation is that the Buyer executes last in the pipeline and receives the full preceding conversation history as context; here, context engineering to pass only information relevant to the Buyer might reduce costs. Alongside the metrics, the dashboard lists the LLM models used by the system and an overview of the agents present in the log, including the tools each agent invokes, and the per-tool error rate. The latter surfaces **O2**: the `create-po` tool of the Buyer Agent has an error rate of 29.52%.

3.2.2. Process Discovery

The process discovery widget renders the agent event log as a Directly-Follows Graph (DFG), with edge thickness encoding execution frequency. For the procurement log, the DFG confirms that the observed executions reflect the overall intended design. The Inventory Agent runs first; in 188 (45 + 143) of 200 executions, it determines that items require reordering, and execution continues to the Requester and Buyer Agents; in the remaining 12 (9 + 3) cases with sufficient stock, execution terminates after the Inventory Agent.

Several behavioral peculiarities are directly visible in the graph. **O3**: In 48 (45 + 3) cases, the Inventory Agent omits the calculator tool entirely. **O4**: The `create-pr` tool of the Requester Agent exhibits a self-loop 48 times, meaning the tool is invoked multiple times within the same execution without a preceding tool error, potentially creating duplicate purchase requisitions. **O5**: In only 47 of the 188 ordering cases does the Requester Agent call `approve-pr`; in the remaining 141, it does not. **O6**: In 64 of the 188 ordering cases the `create-po` tool fails; the graph further shows 3 unsuccessful retry attempts. From those 64 failure cases, three paths are visible: in 20 cases the Buyer Agent calls `approve-pr` and then successfully creates the PO; in 16 cases a subsequent retry of `create-po` succeeds; and in 28 cases the execution terminates without a successfully created PO.

Observations O1 (suspicious token consumption), O4 (redundant tool calls), and partially O6 (unresolved tool errors) are directly actionable. Observations O3, O5, and O6 additionally require more business context to resolve: Is it ever legitimate for the Inventory Agent to omit the calculator? Is the Requester Agent skipping PR approval an unintended behavior? And is the Buyer Agent authorized to approve PRs? These questions motivate conformance checking against expected behavior.

3.2.3. Conformance Checking

For conformance checking, two complementary approaches are described below.

The first is alignment-based conformance against a *normative process model* specified in BPMN and defining the expected behavior. This analysis reveals that the Inventory Agent must always invoke the calculator tool (addressing O3) and that only the Requester Agent may approve purchase requisitions (addressing O5 and O6). Only 2 execution variants are fully conformant, accounting for approximately 28% of the 188 ordering cases, while the remaining 72% deviate in at least one aspect. Based on these findings, improvements to the agent system can be derived: the Inventory Agent's instructions should emphasize the mandatory use of the calculator; the Requester Agent's instructions should explicitly assign PR approval as its responsibility; and access to the `approve-pr` tool must be revoked from the Buyer Agent.

The second approach uses *process atoms* [5], concise declarative specifications of expected behavior that are evaluated independently, which is especially useful when execution behavior is too diverse to be captured in a single happy-path model. Three process atoms have been defined for the agent system: (1) *for every PR created, a PO is successfully created* is satisfied in 85% of ordering cases (related to O6); (2) *the calculator tool is used in every execution* is satisfied in 76% of ordering cases (O3); and (3) *a PR is approved before a PO creation is attempted* is satisfied in only 25% of ordering cases (related to O5). These process atoms enable targeted, ongoing governance monitoring: after applying corrective actions, analysts can track the same atoms across deployments to confirm each change's effect.

Both approaches show compliance rates and violations that directly guide agent design improvements, allowing prioritization by frequency and severity and tracking progress over time.

4. Maturity

Agent behavior mining is a feature available in SAP Signavio Process Intelligence, an enterprise-grade process mining and BPM platform used by organizations worldwide across various industries. ABM and the demonstrated analyses reuse existing functionality, such as pipelines, dashboards, and widgets, thus inheriting the platform's reliability, security, and scalability. The governance insights derived from agent behavior mining using process mining techniques have been evaluated in a structured study with 18 industry practitioners [3], who reported perceived improvements in agent transparency and rated the insights as helpful for governance decisions. ABM was further applied to multi-agent systems implemented in various agent frameworks, including LangGraph, Google ADK, SAP Joule, and customer implementations, demonstrating that agent behavior mining can be applied across platforms and scenarios.

5. Conclusion and Future Work

We showed that mining agent behavior using a process mining platform turns GenAI-agent execution from a black box into a measurable, governable process, without requiring excessive engineering on the analyst tooling side. The observations surfaced in the procurement walkthrough (excess token cost concentrated in one agent, redundant tool invocations, missing approvals, and recoverable failures) illustrate a capability rather than a single system: cost, conformance, and drift signals can be derived from any ABM-conformant log, building the foundation for actively steering agent behavior over time.

A natural next step is to close the loop between observation and behavior to enable continuous improvement. Process atoms, already used here for declarative conformance checking, can also serve as the agent-consumable representation of organization-specific procedural knowledge, a shared, governed organizational memory that agents consult at runtime. ABM observations, recurring violations, successful recovery paths, and novel yet valid execution variants then become candidate updates to the memory. After human curation, these updates can be used to guide subsequent executions.

Declaration on Generative AI

The authors used Claude by Anthropic in order to: Paraphrase and reword, Peer review simulation. The authors reviewed and edited the content and take full responsibility for the publication's content.

References

[1] A. Tour, A. Polyvyanyy, A. A. Kalenkova, Agent system mining: Vision, benefits, and challenges, IEEE Access 9 (2021) 99480–99494.

[2] A. Tour, A. Polyvyanyy, A. A. Kalenkova, A. Senderovich, Agent miner: An algorithm for discovering agent systems from event data, in: BPM, Springer, 2023, pp. 284–302.

[3] H. Vu, M. Körner, A. Rebmann, G. Kevorkian, M. Perscheid, G. Berg, T. Kampik, Agent behavior mining: Generative AI agent governance in business processes, in: BPM, Springer, 2026.

[4] D. Moshkovich, H. Mulian, S. Zeltyn, N. Eder, I. Skarbovsky, R. Abitbol, Beyond black-box benchmarking: Observability, analytics, and optimization of agentic systems, 2025. arXiv:2503.06745.

[5] T. Kampik, C. Warmuth, A. Rebmann, R. Agam, L. N. P. Egger, A. Gerber, J. Hoffart, J. Kolk, P. Herzig, G. Decker, H. van der Aa, A. Polyvyanyy, S. Rinderle-Ma, I. Weber, M. Weidlich, Large process models: A vision for business process management in the age of generative AI, KI – Künstliche Intelligenz 39 (2025) 81–95.