

An Event-Driven, Agent-Based Reference Architecture for Human-AI Collaboration in MBSE

Markus Feld¹, Oliver von Heißen¹, Abed Alrahman Aldade¹, Lucas Martin¹, Bernhard Hein¹, Isaac Mpidi Biti¹,
Fabian Hanke¹, Aschot Hovemann¹, Roman Dumitrescu²

¹Fraunhofer IEM, Germany, ²University Paderborn - HNI, Germany

Abstract:

Integrating LLM-based agents into Model-Based Systems Engineering (MBSE) promises to reduce coordination overhead across heterogeneous toolchains and data silos, but unsystematic integration can result in agent sprawl, loss of human control, and opaque decision processes. This paper presents an event-driven reference architecture for Human-AI Collaboration in MBSE comprising three pillars: an Event-Driven Information Backbone (EIB) that combines pub/sub coordination with an immutable event log, Domain-Specific Agents (DSAs) that act as intelligent intermediaries between engineering tools and the backbone, and Personalized Interaction Agents (PIAs) that filter event streams into role-relevant decision packages and operationalize human-in-the-loop governance through a Decision Inbox pattern. All participants interact exclusively through explicit event contracts, enabling modular integration without bilateral coupling. Correlation identifiers link causally related events across tools and actors, enabling end-to-end reconstruction of decision provenance, including rejected alternatives and their rationale. A proof-of-concept prototype provides initial evidence of technical feasibility.

Keywords: Model-Based Systems Engineering; Human-in-the-Loop; Event-Driven Architecture; Multi-Agent Systems; Decision Provenance

1 Introduction

1.1 Motivation / Problem statement

Model-Based Systems Engineering (MBSE) requires the coordination of diverse artifacts such as requirements, functions, architecture models, and trace links across multiple stakeholders and heterogeneous toolchains. Integrating information from different sources and collaborating among multi-disciplinary teams are routinely mentioned as major pain points in day-to-day MBSE work, often hindered by a lack of interoperability (Roche et al., 2025). Recent advances in Large Language Models (LLMs) have enabled specialized AI agents capable of supporting engineers with routine tasks. However, a fundamental question remains unanswered: How can a growing population of such agents be systematically integrated into existing development processes? Organizations face three interrelated risks:

- **Integration complexity:** Each additional agent increases integration effort. Ad-hoc integrations lead to unmanageable “agent sprawl” that is difficult to maintain, scale, or adapt.
- **Loss of human control:** Agents may autonomously propose or execute changes without adequate human oversight, which is unacceptable in safety-critical domains.
- **Opaque decision processes:** Existing systems capture *who* made a change and *when*, but not the decision path: What information informed the decision? Which human or AI actors contributed? What alternatives were considered and *why* were they rejected? This becomes critical when decisions rely on AI-generated proposals.

1.2 Research Gap and derived Research Questions

Prior work has extensively addressed *what* AI agents can do for engineering (e.g. generating artifacts, classifying information, analyzing dependencies) (Wang et al., 2024a). Considerably less attention has been paid to *how* agents should be architecturally integrated into MBSE processes such that (a) integration remains scalable, maintainable, and adaptable as agent populations grow, (b) human engineers retain meaningful control over agent-driven changes, and (c) the decision-making process - not just the final outcome - becomes traceable and reconstructable. Current integration approaches typically follow a workflow-centric paradigm that explicitly models control flows between tools and agents. This becomes brittle as tools evolve, new agents are introduced, and process variants proliferate. We argue that a fundamentally different approach is needed: a shift from control-flow modeling to participant contracts and event-based coordination.

To address this gap, we formulate the following research questions:

- **RQ1:** What architectural paradigm enables the scalable, maintainable, and adaptable integration of AI agents into MBSE processes?

- **RQ2:** What design principles and components are required to operationalize human-in-the-loop control in an event-driven agent ecosystem?
- **RQ3:** How can engineering decisions involving human and AI actors be systematically captured such that their rationale and provenance become traceable and reconstructable?

1.3 Contributions and Paper Outline

This paper makes two contributions: First, we propose a **reference architecture for Event-Driven, Agent-Based Human-AI Collaboration** in MBSE that replaces brittle, point-to-point couplings with publish-subscribe coordination through three pillars: an Event-Driven Information Backbone (EIB) with a correlation-based event model for decision traceability, Domain-Specific Agents (DSAs) and Personalized Interaction Agents (PIAs) governed by explicit participant contracts, and the Decision Inbox pattern for scalable human-in-the-loop governance. Second, we provide a **proof-of-concept prototype** that demonstrates the technical feasibility of the architecture through a representative change scenario from the domain of autonomous transport systems.

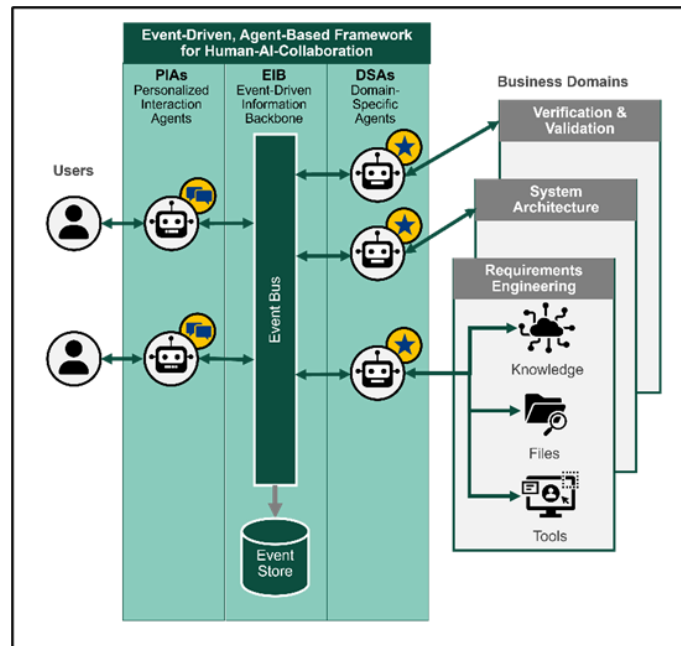


Figure 1: High-level overview of the reference architecture. Personalized Interaction Agents (PIAs) mediate between human users and the Event-Driven Information Backbone (EIB), which combines an event bus for asynchronous pub/sub coordination with a persistent event store for traceability. Domain-Specific Agents (DSAs) connect the backbone to bounded engineering domains, enabling scalable integration.

2 Related Work

2.1. Architectural Patterns for Multi-Agent Collaboration

Recent literature on LLM-based multi-agent systems identifies several recurring coordination patterns (Zou et al., 2025). Centralized orchestration delegates task decomposition to a planner agent (Shen et al., 2023). Programmable conversation topologies model collaboration as agent dialogue graphs with explicit interaction policies (Wu et al., 2023). Role-structured pipelines enforce staged handovers (Hong et al., 2024; Qian et al., 2024), and ensemble architectures aggregate multiple agent proposals for robustness (Wang et al., 2024b). Across these patterns, the dominant evaluation objective is task success and artifact reliability. Governance-relevant properties, such as persistent provenance, cross-tool auditability, and reconstructable decision pathways, are increasingly discussed but not uniformly treated as first-class architectural requirements (Zou et al., 2025). Traceability is typically implemented as an auxiliary logging layer outside the coordination mechanism, which can be incomplete when tools, agents, or workflows change. This motivates architectures where coordination and traceability are designed conjunctively.

2.2. Human-AI-Collaboration

Human-AI collaboration has increasingly shifted from viewing AI as a passive decision-support tool to understanding it as an active collaborator with complementary strengths. Research on human-AI teaming highlights that effective collaboration requires shared goals, mutual predictability, and coordinated adaptation between humans and AI systems (Klein et al., 2004), with hybrid intelligence focusing on complementarity rather than substitution (Dellermann et al.,

2019). Mixed-initiative interaction allows both humans and AI to take initiative and negotiate control (Horvitz, 1999). Human-in-the-loop and human-on-the-loop patterns specify how oversight is distributed: design guidelines stress transparency, feedback channels, and controllability (Amershi et al., 2019), while human-on-the-loop supervision requires that humans define goals, monitor behavior, and intervene on deviation (Kambhampati, 2020). In safety-relevant engineering domains, LLM-based tools reduce manual effort but risk shifting decision-making away from engineers (Vats et al., 2024; Hagemann et al., 2023). Overreliance on AI suggestions can erode critical scrutiny (Schmutz et al., 2024), motivating agentic architectures that couple autonomous execution with escalation policies and human approval checkpoints (Feng et al., 2024; Zou et al., 2025). Regulatory frameworks explicitly demand that humans remain accountable decision-makers in high-impact domains (European Union, 2024; NIST, 2024). Early work on Personal Intelligent Agents (PIAs) introduced per-user agents that filter and integrate information from heterogeneous sources based on profiles and context (Albayrak et al., 2005). However, existing PIA work optimizes personalized information delivery without addressing formal engineering artifacts, decision provenance, or coordination with other AI agents.

2.3. Event-Driven Architecture (EDA)

Event-driven architecture (EDA) is widely adopted in distributed systems to enable loose coupling and asynchronous coordination between heterogeneous producers and consumers. Event streaming platforms such as Kafka provide a durable log supporting replay, fan-out consumption, and scalable processing (Kreps et al., 2011). Beyond scalability and decoupling, EDA is relevant in this context because it enables persistent trace reconstruction. For provenance and reconstructability, many EDA-based systems adopt Event Sourcing Patterns, where state changes are recorded as an append-only sequence of events and materialized into projections for read use. The empirical software engineering literature highlights that event sourcing improves auditability but introduces hard engineering problems, most notably schema evolution, projection rebuilding, and operational complexity (Overeem et al., 2021). Newer work continues to address practical gaps, including benchmarking and technology selection for event-sourcing databases under realistic workloads (Malyi, Serdyuk, 2024). In engineering contexts, EDA is increasingly discussed as an enabling mechanism for digital thread continuity across lifecycle tools, where asynchronous propagation and replay support cross-domain synchronization and traceability (Wu et al., 2025; Zhang et al., 2025).

3 Research Methodology

The reference architecture was developed following the **Design Science Research Methodology (DSRM)** of Peffers et al. (2007), applied iteratively across the six steps of the DSRM Design Cycle: **Problem identification** combined reported MBSE pain points, particularly interoperability and cross disciplinary coordination challenges (Roche et al., 2025), with insights gained from industrial MBSE integration projects, resulting in the three risks that motivate this work. In **objective definition**, these risks and the research questions were translated into seven observable solution objectives (Table 1). **Design and development** operationalized the objectives through six design principles and the three component types presented in Section 4. **Demonstration** was conducted through the proof of concept scenario in Section 5. **Evaluation** maps the observed behavior back to the objectives in Section 6. **Communication** is provided through this paper and the planned open source release of the prototype.

Table 1: Solution objectives derived from the research questions.

ID	Objective	RQ	Facet
O1	Adding or removing an agent requires no modification to existing participants or workflows.	RQ1	Modularity
O2	An engineering tool can be replaced behind a DSA without affecting other participants in the ecosystem.	RQ1	Adaptability
O3	No agent-proposed change is executed without an explicit human disposition (approve, reject, or override).	RQ2	Control guarantee
O4	The system filters and aggregates events into role-relevant decision packages to manage cognitive load.	RQ2	Cognitive relief
O5	The level of human involvement is configurable per risk level and confidence threshold, without architectural changes.	RQ2	Progressive autonomy
O6	The complete decision path (trigger, analyses, alternatives, human dispositions, and execution) is reconstructable.	RQ3	End-to-end traceability
O7	Rejected proposals and their rationale are preserved with the same fidelity as approved ones.	RQ3	Decision completeness

4 Reference Architecture Design

This section introduces an event-driven reference architecture for Human-AI Collaboration in MBSE. It integrates principles from event-driven architecture and event sourcing (Kreps et al., 2011; Overeem et al., 2021), multi-agent

coordination (Zou et al., 2025), human-AI interaction guidelines (Amershi et al., 2019), personal intelligent agents (Albayrak et al., 2005), and digital thread approaches in engineering (Wu et al., 2025), adapting them to the governance and provenance requirements of MBSE. As shown in Figure 1, the architecture comprises three core component types: **an Event-Driven Information Backbone (EIB)**, **Domain-Specific Agents (DSAs)**, and **Personalized Interaction Agents (PIAs)**. Their interactions are governed by explicit **participant contracts** and a **correlation-based event model**. The section first derives the design principles from the research questions and related work, then describes each component type, the contract and event model, and the Decision Inbox pattern for human-in-the-loop governance.

4.1 Design Principles

The architecture is guided by six design principles that operationalize the solution objectives defined in Section 3. Table 2 summarizes each principle, its rationale, and the objectives it addresses. DP1 and DP2 operationalize O1 and O2 (modularity and adaptability): because participants are coupled only through event contracts, adding or removing a participant requires no workflow redesign, and tools can be replaced behind their DSA without affecting others. DP3 and DP6 operationalize O3 and O5 (human control): treating human decisions as first-class events ensures the control guarantee (O3), while separating the governance layer (PIA) from domain logic (DSA) enables independent evolution of filtering and escalation logic (O4, O5). DP4 and DP5 operationalize O6 and O7 (traceability): an immutable log with correlation identifiers enables end-to-end trace reconstruction (O6), and the combination of DP3 with DP4 ensures that rejected proposals and their rationale are preserved alongside approved ones (O7).

Table 2: Design principles and their mapping to solution objectives.

Principle	Rationale	Objectives
DP1: Loose coupling through events	Participants interact exclusively through asynchronous events; no direct point-to-point dependencies	O1, O2
DP2: Explicit participant contracts	Subscriptions, publications, and permissions are formally declared, making integration inspectable and versionable	O1, O2, O5
DP3: Human decisions as first-class events	Approvals, rejections, and overrides are captured as immutable events with full attribution and provenance	O3, O7
DP4: Immutable event log	All events are persisted in an append-only store; system state is derived from event replay	O6, O7
DP5: Correlation-based traceability	Causally related events share a correlation identifier for end-to-end trace reconstruction	O6
DP6: Separation of concerns	Domain logic (DSA), user interaction/governance (PIA), and infrastructure (EIB) are architecturally distinct	O1, O4, O5

4.2 Event-Driven Information Backbone (EIB)

The EIB forms the **central nervous system of the architecture** by combining two functions. As an **Event Bus**, the EIB facilitates asynchronous, decoupled communication using a publish/subscribe model (Kreps et al., 2011). Any participant can publish events to named topics and subscribe to relevant topics without knowledge of other consumers or producers. This enforces DP1, promotes modularity, and supports horizontal scaling. As a **persistent Event Store**, the EIB archives every event in an immutable, append-only log following the **event sourcing pattern** (Overeem et al., 2021), fulfilling DP4. The log enables replay for recovery, debugging, or simulation, supports temporal queries for audit, and allows late-joining participants to reconstruct historical state. Together, the event bus and event store ensure the EIB is not merely a transport layer but a **coordination substrate** carrying structured change events, causal links, and decision provenance while providing a **reconstructable record of collaboration**.

4.3 Domain-Specific Agents (DSAs)

DSAs act as **intelligent intermediaries** between engineering tools and the EIB. Each DSA is responsible for a bounded engineering domain such as requirements management or system architecture. As event publishers, DSAs **monitor connected tools for state changes**, transform them into the canonical event schema (Section 4.6), and publish them to the EIB. As event consumers, they subscribe to relevant topics and **translate incoming events into tool-specific actions**. A key principle is data sovereignty: authoritative data remains in the tools, while DSAs expose **state changes as standardized events** rather than replicating raw tool data. Operationally, a DSA follows an **observe-analyze-act cycle**. It observes by consuming events and monitoring tool changes. It analyzes using domain knowledge, rules, retrieval-augmented generation, or LLM-based reasoning (cf. Wang et al., 2024a). It acts by publishing new events such as impact reports, change proposals, or validations back to the EIB. Critically, DSAs do not communicate directly with other DSAs or PIAs; all coordination is mediated through events on the backbone (DP1). A DSA’s internal implementation remains opaque to the architecture as long as its event contract is honored (DP2, DP6).

4.4 Personalized Interaction Agents (PIAs)

PIAs serve as the **interaction and governance layer** for human engineers, extending the Personal Intelligent Agent concept (Albayrak et al., 2005) from personalized information delivery to engineering decision governance. Each engineer is supported by an individualized PIA that **adapts to personal preferences, user roles, and current task context**. A PIA subscribes to all EIB topics relevant to its user and continuously monitors incoming events. It filters the event stream to surface only role-relevant information, aggregates related events along the engineering thread into **coherent information packages**, enriches these packages with additional context (e.g., impact reports, summaries), and presents them through the Decision Inbox Pattern (Section 4.7). This mediated interaction design ensures **human-in-the-loop control** and safeguards against uncontrolled automation. The PIA **captures human decisions**, such as approvals, rejections, modifications, or escalations, and publishes them as first-class events on the EIB, **creating an auditable decision trail** recording the actor, rationale, and information basis (DP3). By separating the governance layer (PIA) from domain logic (DSA), the architecture ensures that human oversight is independent of the specific engineering domain: a new DSA can be introduced without modifying the PIA, and the PIA's escalation policies can evolve without affecting domain agents (DP6).

4.5 Participant Contracts

Every participant is defined by an **explicit participant contract** declaring five elements: a **unique participant identity**, the set of event topics and schema versions it subscribes to, the **set of event types and schema versions** it may publish, its **permissions** (propose, approve, or execute), and a **behavioral description** for documentation purposes. Contracts serve as the **unit of modularity** (DP2): introducing a new participant requires only registering its contract; removing one requires only unsubscribing. No workflow, orchestrator, or other participant needs modification. This property is central to the scalability claim of the architecture (O1).

4.6 Correlation-Based Event Model

All events conform to a **canonical two-part structure**. The **header** carries metadata for **routing, traceability, and attribution**: a globally unique *event_id*, UTC timestamp, *event_type* following a hierarchical namespace (e.g., *Requirement.Created*, *Architecture.Change.Proposed*), *source_agent*, optional *actor_id* identifying the triggering human, a *correlation_id* **grouping all events in the same end-to-end scenario**, and a *schema_ref* for versioned deserialization. The **payload** contains a *subject_id* identifying the **domain entity and a data object** with domain-specific content. The *correlation_id* is the **primary traceability mechanism** (DP5, O6). Assigned when a scenario begins and propagated to every downstream event, it **links all causally related events across** process chains. Querying the event store by *correlation_id* yields the complete trace: the triggering event, analyses, proposed alternatives, human decisions, and executed changes.

4.7 Decision Inbox Pattern

The Decision Inbox operationalizes **human-in-the-loop governance at scale** through a five-step process. First, the PIA aggregates all proposals and supporting events sharing a *correlation_id* into a **single decision package**. Second, it **contextualizes the package** with background information including affected artifacts, confidence scores, and risk levels. Third, the engineer can approve or reject each proposal individually, request additional analysis, or override a recommendation. Fourth, each **decision is captured as an immutable event** (**.Approved*, **.Rejected*, **.Overridden*) with the actor and optional rationale, and published to the EIB. Fifth, approved proposals trigger downstream execution by the responsible DSA, while rejected proposals are preserved with their rationale for auditability. The Decision Inbox further supports **progressive autonomy through configurable escalation policies**, building on mixed-initiative interaction (Horvitz, 1999) and the human-on-the-loop pattern (Kambhampati, 2020). Low-risk proposals with high agent confidence may be auto-approved (with the auto-approval recorded as an event), while high-risk or low-confidence proposals are always escalated to human review. This bridges the human-in-the-loop and human-on-the-loop patterns identified in Section 2.2, allowing organizations to calibrate human involvement to their risk tolerance and regulatory requirements. Figure 2 illustrates a decision package within the Decision Inbox of the PoC prototype.

5 Proof-of-Concept: Implementation and Demonstration

To demonstrate the reference architecture's technical feasibility, we implemented a proof-of-concept (PoC) prototype and exercised it through a representative change scenario from the domain of autonomous transport systems. The PoC prototype will be made available as open-source software upon publication. Section 5.1 describes the technology stack and its mapping to the reference architecture components. Section 5.2 presents the generic four-phase walkthrough that the architecture follows for cross-domain change propagation. Section 5.3 instantiates this walkthrough with a concrete demonstration scenario and reports the results.

5.1 Technology Stack

The prototype instantiates all three component types of the reference architecture as independently deployable, containerized services orchestrated via Docker Compose.

Table 3: PoC technology stack and mapping to reference architecture components (Arch. Type)

Component	Arch. Type	Technology	Role in Architecture
EIB	EIB	Redpanda + Schema Registry	Kafka-compatible event bus, event store, and versioned JSON schemas
REA (Requirements Engineering Agent)	DSA	Python/FastAPI	DSA for requirements management
SAA (System Architecture Agent)	DSA	Python/FastAPI, CrewAI, OpenAI API	DSA for system architecture with LLM-based impact analysis
PIA Backend	PIA	Python/FastAPI, WebSocket	PIA: event aggregation and decision capture
PIA Frontend		Next.js (React)	PIA: Decision Inbox, Trace View, and model exploration UI

All agent services share a common library providing base classes, canonical event schemas, and Kafka integration utilities. Each DSA interacts with its domain through a ToolAdapter interface that decouples agent logic from specific tool implementations. In the prototype, REA and SAA both use JSON-based model stores simulating production tools; replacing these with real connectors requires only a new adapter implementation (DP2, DP6). The PIA is split into two containers: the backend communicates with the EIB and implements all business logic; the frontend provides the user-facing views, including the Decision Inbox for reviewing and acting on proposals, a Trace View for auditing decision chains, and model exploration views for requirements, functions, and architecture. The frontend communicates exclusively with the backend via REST and WebSocket and has no direct EIB access. The prototype operates on a synthetic dataset modeled after a real-world Driverless Transport System (DTS). The dataset comprises a representative set of MBSE artifacts, including requirements, system functions, and logical as well as physical architecture components.

5.2 Generic Scenario Walkthrough

The architecture handles cross-domain change propagation through four distinct phases. The following paragraphs detail each phase to provide a general understanding of the workflow.

Phase 1: Requirement Creation. When a requirements engineer captures a new requirement in the requirements management tool, the REA detects this creation through its ToolAdapter, translates it into a *Requirement.Created* event conforming to the canonical event schema, and publishes it to the EIB. This event carries a newly assigned *correlation_id* that links all subsequent events in the resulting process chain.

Phase 2: AI-Powered Impact Analysis. The System Architecture Agent (SAA), subscribed to requirement topics, receives the *Requirement.Created* event. The SAA retrieves the current function and architecture models and invokes a CrewAI-based analysis pipeline that leverages an LLM to assess impacts on existing functions and propose necessary changes. The analysis distinguishes between four types of actions: *create* (new function is required), *modify* (logic of existing function changes), *extend* (new capability added to existing function), and *delete* (function becomes obsolete). For each proposed change, an individual *Architecture.Change.Proposed* event is published, containing the specific target, action, rationale, and detailed assessment (risk, confidence, effort). Additionally, an *Impact.Report.Created* event is published that consolidates all proposed changes and provides an overall assessment of the requirement's impact on the architecture, including aggregate risk evaluation and confidence metrics for decision-making. All published events carry the *correlation_id* from the original *Requirement.Created* event, ensuring full traceability and enabling to link analysis results back to the initiating requirement.

Phase 3: Human-in-the-Loop Decision. The System Architect's (SA) PIA Backend subscribes to all architecture-related events on the EIB. It aggregates incoming *Impact.Report.Created* and *Architecture.Change.Proposed* events by user role and *correlation_id*, assembling them into a coherent decision package. The PIA Frontend renders this package in the Decision Inbox, presenting the SA with the overall impact assessment, affected functions, risk levels, and confidence scores alongside individual proposals. The architect can review and decide on each proposal individually or approve/reject them in bulk. Figure 2 shows a screenshot of the prototype's Decision Inbox. For each decision, the PIA Backend publishes either an *Architecture.Change.Approved* or *Architecture.Change.Rejected* event to the EIB, recording the SA's *actor_id* and optional rationale. Both event types inherit the *correlation_id* from the original requirement, maintaining the end-to-end traceability.

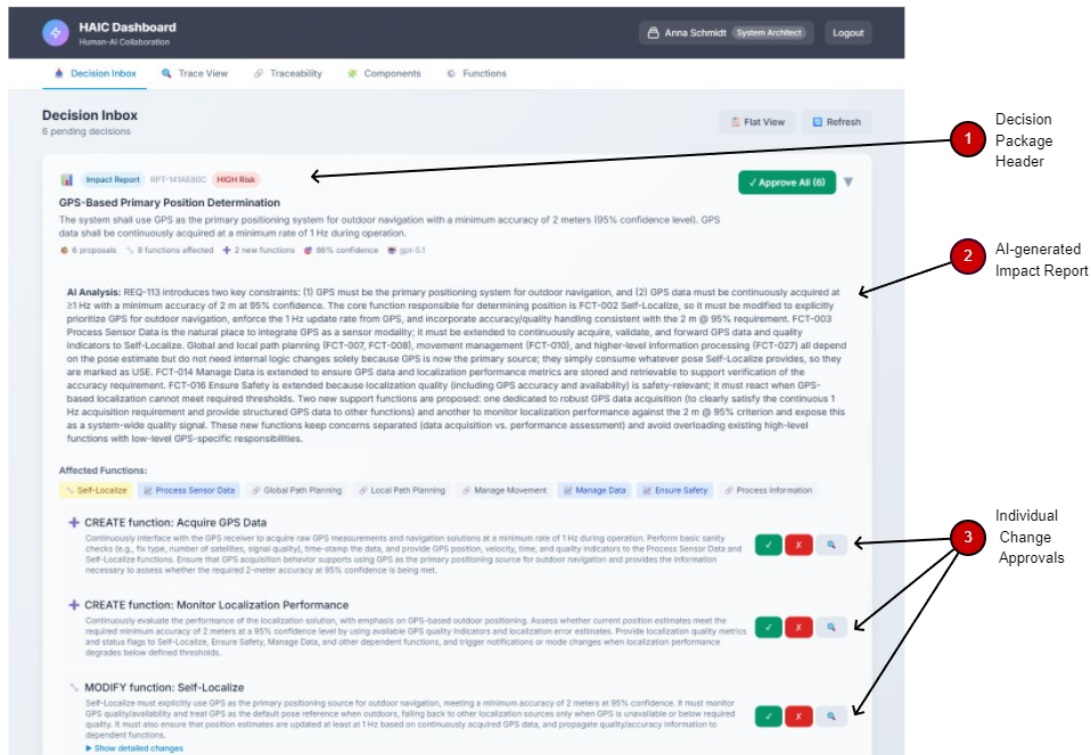


Figure 2: Decision Inbox of the proof-of-concept prototype showing the aggregated decision package for REQ-113 (excerpt). The PIA consolidates all events sharing the same correlation_id into a single view comprising (1) the decision package header with metadata, risk level, and batch approval, (2) the LLM-generated impact analysis with affected functions and confidence assessment, and (3) the individual change proposals, each with approve, reject, and inspect actions.

Phase 4: Execution. The SAA consumes the *Architecture.Change.Approved* events and executes the approved proposals against the architecture model, thereby implementing the actual architectural changes in the system. For each executed change, the SAA publishes corresponding events to the EIB: an *Architecture.Function.Created* event for newly created functions, and *Architecture.Function.Modified* events for functions that have been modified or extended. All published events inherit the *correlation_id* from the original requirement, enabling downstream agents to react to the architectural changes and maintain full traceability throughout the workflow. Rejected proposals, if any, are not executed but remain in the event log with their rationale and decision record, ensuring completeness and auditability of all architectural decisions.

5.3 Demonstration Scenario and Results

To validate the architecture, the generic walkthrough from Section 5.2 was instantiated as a concrete demonstration scenario. It shows a complete change propagation and verifies that the architectural components interact as intended. Figure 3 presents the corresponding interactions and process flow as a sequence diagram, which is detailed in the following section.

Phase 1: A Requirements Engineer creates the requirement “REQ-113: The system shall use GPS as the primary positioning system for outdoor navigation with a minimum accuracy of 2 meters (95% confidence level). GPS data shall be continuously acquired at a minimum rate of 1 Hz during operation.”. The REA detects the creation and publishes a *Requirement.Created* event to the EIB. **Phase 2:** The *Requirement.Created* event triggers the SAA, which invokes its CrewAI-based analysis pipeline. The analysis pipeline assesses the impact and provides six change proposals (two new functions, one modification, three extensions), each published as an individual *Architecture.Change.Proposed* event. The pipeline also creates an Impact Report reflecting the overall assessment of the requirement's impact and publishes it as an *Impact.Report.Created* event. **Phase 3:** The PIA receives these events and consolidates them into a coherent decision package based on their shared *correlation_id*, then presents it in the Decision Inbox of the SA. In our scenario, the SA reviews and approves all six proposals. The PIA translates these approvals into corresponding *Architecture.Change.Approved* events back to the EIB. **Phase 4:** The SAA receives these events, applies the approved changes to the Architecture model, and emits two *Architecture.Function.Created* and four *Architecture.Function.Modified* events (one modification, three extensions) as a result. In a real-world environment, these events could in turn trigger further agents to perform downstream validations or adjustments. For the purposes of this demonstration the event chain ends here.

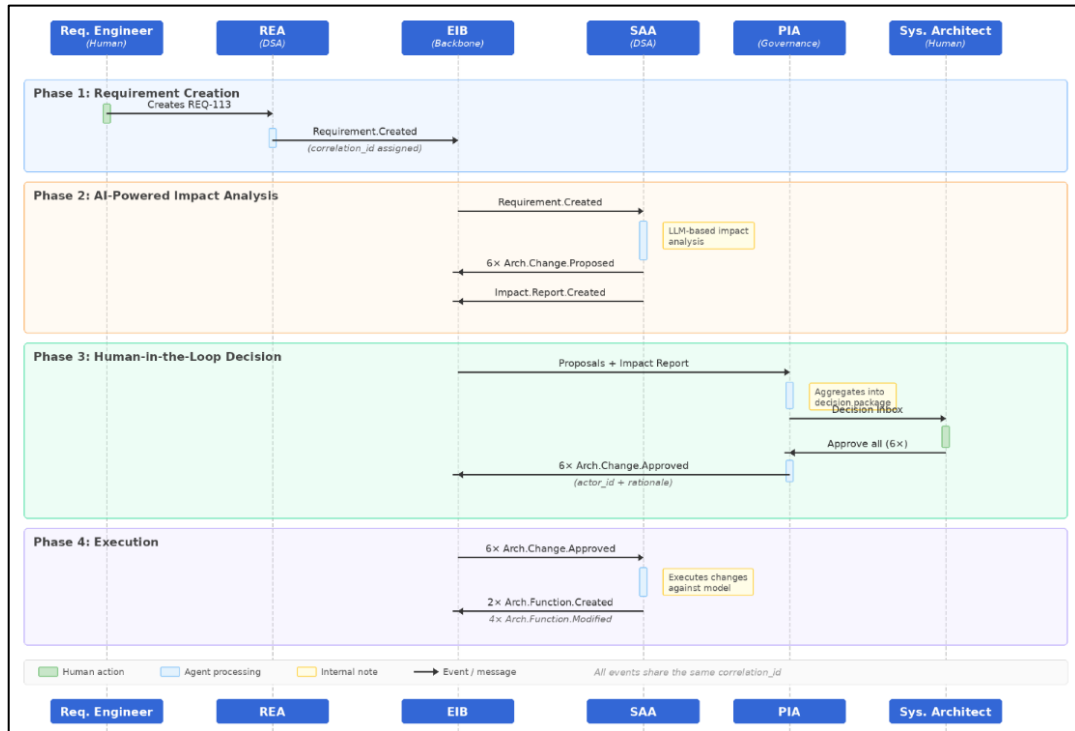


Figure 3: Sequence diagram of the PoC scenario walkthrough, illustrating the end-to-end event chain of architectural change propagation across four phases: requirement creation, LLM-based impact analysis with proposed architecture changes, human-in-the-loop approval, and automated execution of approved changes.

The demonstration confirmed the architecture’s technical feasibility across all four phases, from the initial requirement creation through impact analysis, decision-making, and execution. The complete end-to-end flow executed without manual intervention between phases. All events were delivered via the EIB with correct topic routing, and all events within the scenario consistently shared the same *correlation_id*, confirming correlation integrity across participant boundaries. The Decision Inbox (Figure 2) correctly aggregated the six change proposals and the impact report into a single decision package, and no architectural change was executed without prior human approval. Finally, the Trace View (Figure 4) successfully reconstructed the complete event chain from the single *correlation_id*, including the decision rationale provided by the architect, demonstrating that the provenance record is both complete and queryable.

6 Discussion

6.1 Evaluation Against Solution Objectives and Scalability Considerations

This section evaluates the reference architecture and its proof-of-concept instantiation against the seven solution objectives defined in Section 3. Regarding modularity (O1), the proof-of-concept confirmed that agents can be integrated without bilateral coupling. The REA and SAA operate independently and are connected only through the EIB; adding the SAA required only a contract registration and event subscription, with no modifications to the REA. Tool exchangeability (O2) was demonstrated in a simulated setting: both DSAs interact with their domain through a ToolAdapter interface, and the JSON-based model store is a pluggable implementation. While no production tool was connected, the architecture confines tool replacement to the adapter layer without affecting other participants. The control guarantee (O3) was directly demonstrated: all six architecture change proposals were routed to the Decision Inbox (Figure 2), and execution occurred only after explicit human approval. Cognitive relief (O4) was likewise confirmed, as the PIA aggregated the six proposals and one impact report into a single decision package filtered by role and *correlation_id*, rather than presenting individual events. Progressive autonomy (O5) is architecturally supported through configurable escalation policies in the PIA, including an auto-approval path for low-risk, high-confidence proposals. However, this path was not exercised in the demonstration scenario, which followed a strict human-in-the-loop workflow. End-to-end traceability (O6) was demonstrated through the Trace View (Figure 4), which reconstructed the full event chain from a single *correlation_id*, spanning requirement creation, impact analysis, human decisions, and execution. Decision completeness (O7) is ensured by design: approved and rejected events use identical schemas and are both persisted in the immutable log with actor and rationale fields, ensuring that the decision record is complete regardless of outcome. In summary, five objectives (O1, O3, O4, O6, O7) were directly demonstrated, O2 was demonstrated in a simulated setting, and O5 was confirmed as architecturally supported. Beyond the demonstrated scenario, the architecture combines three complementary scaling dimensions to address the complexity of real-world MBSE settings. *Technical scalability* rests on the EIB, which inherits

the horizontal scalability of Kafka-compatible event platforms. *Integration scalability* follows from the contract-based modularity established under O1: adding any participant requires only registering its contract, without modifications to existing participants or workflows, so integration effort scales linearly with the agent population. *Cognitive scalability* builds on the PIA-side filtering, aggregation, and auto-approval established under O4 and O5; beyond these, the Correlation-Based Event Model binds causally related events and decisions into coherent threads, which becomes all the more important as scenarios grow more complex, with non-linear, iterative decision paths. Empirical validation of these properties at production scale, however, remains necessary (see Section 6.2).

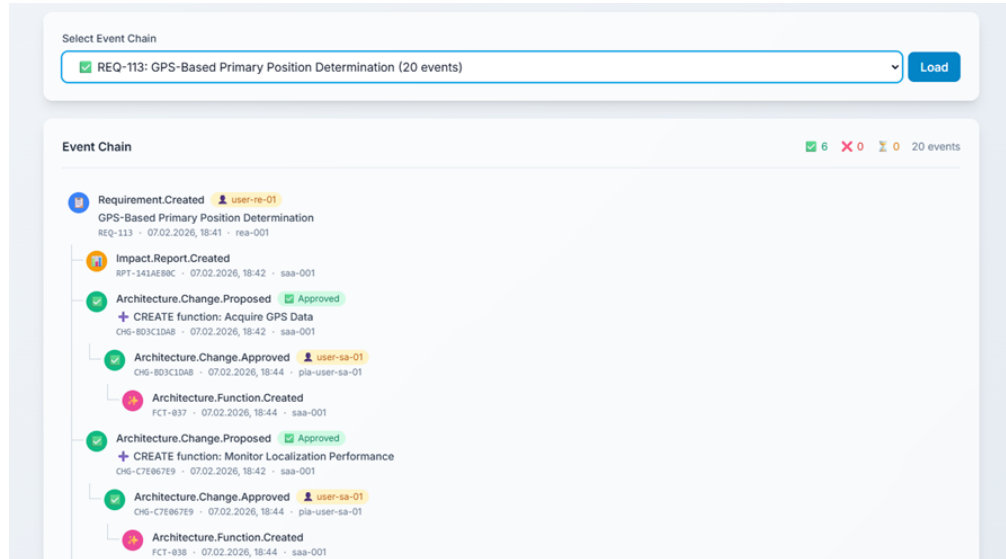


Figure 4: Trace View of the proof-of-concept prototype showing the event chain for REQ-113 (excerpt). The hierarchical visualization reconstructs all 20 causally related events using a single correlation `id` and arranges them in causal order, making it possible to trace every event back to its origin.

6.2 Limitations and Threats to Validity

Several limitations must be acknowledged. First, validation has so far been limited to the proof-of-concept demonstration and the judgement of the MBSE experts involved in the architecture’s design and development. No independent evaluation with uninvolved practitioners or domain experts has been conducted to date. Systematic validation through structured interviews, workshops, or controlled experiments with independent systems engineers is the planned next step and is essential to assess the architecture’s practical utility. Second, the proof-of-concept uses simulated tool adapters with JSON-based stores rather than real engineering tools; integration with production MBSE toolchains (e.g., Cameo, DOORS) may surface additional challenges related to API constraints, data volume, and latency. Third, the demonstration covers a single scenario of moderate complexity in one domain (autonomous transport) and therefore does not proof applicability to other engineering domains or to industrial-scale MBSE settings with thousands of artifacts, large agent populations, and concurrent decision processes. Although the architecture is designed to scale along the three dimensions outlined in Section 6.1, empirical validation across domains and at production scale remains a task for future work. Fourth, the LLM-generated impact analyses were not evaluated for correctness or completeness against a ground truth. This was a deliberate scoping decision: the proof-of-concept was designed to demonstrate the feasibility of the reference architecture’s coordination, governance, and traceability mechanisms, not to evaluate the quality of individual agent outputs, which constitutes an orthogonal research question. Fifth, inherent trade-offs of event-centric systems (including schema evolution, event ordering, and consistency reasoning) remain non-trivial engineering challenges that require further investigation at scale (Laigner et al., 2024; Overeem et al., 2021).

7 Conclusion and Future Work

This paper presented an event-driven, participant-centric reference architecture for Human-AI Collaboration in MBSE. Through an Event-Driven Information Backbone, Domain-Specific Agents, and Personalized Interaction Agents coordinated by participant contracts and a Decision Inbox pattern, the architecture addresses agent sprawl, loss of human control, and opaque decision processes. A proof-of-concept prototype demonstrated technical feasibility against seven solution objectives operationalized through six design principles.

Future work will pursue four directions. First, systematic evaluation with independent MBSE experts through structured interviews and workshops, complemented by technical validation under realistic conditions, including integration with production tools such as Cameo or DOORS, comparison of LLM-generated impact analyses against expert baselines, and

extension to multi-user scenarios with concurrent and conflicting decisions. Second, organizational guardrails for progressive autonomy, including role-based authorization, domain-specific risk taxonomies, and audit requirements aligned with regulatory and governance constraints. Third, multi-agent negotiation protocols for cases where multiple DSAs generate conflicting proposals. Fourth, schema evolution and event versioning strategies for long-lived engineering projects.

By shifting the integration paradigm from workflow-centric control flows to participant contracts and event-based coordination, the proposed architecture offers a scalable foundation for embedding AI agents into MBSE processes while keeping human engineers firmly in control. At the same time, the architecture ensures that decision-making processes are traceably captured and preserved, opening new perspectives for engineering transparency, organizational learning, and evidence-based process improvement.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used generative AI tools (Gemini, ChatGPT, Claude) in order to improve the readability and language of the text. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

- Albayrak, S., et al., 2005. Agent technology for personalized information filtering: The PIA-system, in: Proceedings of the 2005 ACM symposium on Applied Computing. ACM, pp. 1603-1609.
- Amershi, S., et al., 2019. Guidelines for Human-AI Interaction. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19.
- Dellermann, D., Ebel, P., Söllner, M., Leimeister, J.M., 2019. Hybrid Intelligence. Business & Information Systems Engineering 61(5), 637–643.
- European Union, 2024. Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union.
- Feng, X., et al., 2024. Large language model-based human-agent collaboration for complex task solving. In: Findings of the Association for Computational Linguistics: EMNLP 2024.
- Hagemann, V., Rieth, M., Suresh, A., Kirchner, F., 2023. Human-AI teams - Challenges for a team-centered AI at work. Frontiers in Artificial Intelligence 6, 1252897.
- Hong, S., et al., 2024. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In: Proceedings of the International Conference on Learning Representations (ICLR 2024).
- Horvitz, E., 1999. Principles of mixed-initiative user interfaces. Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99
- Kambhampati, S., 2020. Challenges of Human-Aware AI Systems. AI Magazine 41(3), 3–17.
- Klein, G., et al., 2004. Ten Challenges for Making Automation a 'Team Player' in Joint Human-Agent Activity. IEEE Intelligent Systems 19(06), 91–95.
- Kreps, J., Narkhede, N., Rao, J., 2011. Kafka: A distributed messaging system for log processing. Proceedings of the NetDB Workshop.
- Laigner, R., et al., 2024. An empirical study on challenges of event management in microservice architectures. arXiv preprint (arXiv:2408.00440).
- Malyi, S., Serdyuk, P., 2024. Developing a performance evaluation benchmark for event sourcing databases. Selected Issues of Scientific Networking 15, 159–168
- NIST (National Institute of Standards and Technology), 2024. AI Risk Management Framework: Generative AI Profile (NIST AI 600-1). National Institute of Standards and Technology.
- Overeem, M., Spoor, M., Jansen, S., Brinkkemper, S., 2021. An empirical characterization of event sourced systems and their schema evolution-lessons from industry. Journal of Systems and Software 178, 110970.
- Peffers, K., et al., 2007. A design science research methodology for information systems research. Journal of Management Information Systems 24, 45-77.
- Qian, C., et al., 2024. ChatDev: Communicative Agents for Software Development. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 1722–1737.
- Roche, J., Natu, S., Tolle, D., 2025. The current reality and future trends in model-based systems engineering. CIMdata, Inc., Position Paper (Roche_Final_Distribution_NA_2025.pdf). Available at: https://www.cimdata.com/images/position-papers/Roche_Final_Distribution_NA_2025.pdf.
- Schmutz, J.B., et al., 2024. AI-teaming: Redefining collaboration in the digital era. Current Opinion in Psychology 58, 101837.
- Shen, Y., et al., 2023. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. Advances in Neural Information Processing Systems (NeurIPS).
- Vats, V., et al., 2024. A survey on human-AI collaboration with large foundation models. arXiv preprint (arXiv:2403.04931).
- Wang, L., et al., 2024a. A survey on large language model based autonomous agents. Frontiers of Computer Science 18(6), 186345.
- Wang, J., et al., 2024b. Mixture-of-agents: Enhances large language model capabilities. arXiv preprint (arXiv:2406.04692).
- Wu, Q., et al., 2023. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. arXiv preprint (arXiv:2308.08155).
- Wu, S., et al., 2025. Digital thread in engineering: Concept, state of art, and enabling framework. Advanced Engineering Informatics 65, 103258.
- Zhang, L., et al., 2025. MBSE 2.0: Toward more integrated, comprehensive, and intelligent MBSE. Systems 13(7), 584.
- Zou, H.P., et al., 2025. LLM-based human-agent collaboration and interaction systems: A survey. arXiv preprint (arXiv:2505.00753).