IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 August 2025, accepted 12 October 2025, date of publication 15 October 2025, date of current version 21 October 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3621887

RESEARCH ARTICLE

Implementing Agentic AI Into ERP Software

SIAR SARFERAZ, (Member, IEEE)
Research and Development, SAP SE, 69190 Walldorf, Germany
e-mail: siar.sarferaz@sap.com

ABSTRACT Enterprise Resource Planning (ERP) systems serve as the backbone of modern business operations, digitalizing and integrating processes across organizational departments. These systems encompass a wide array of functions, including sales, marketing, finance, supply chain management, manufacturing, services, procurement, and human resources, acting as a centralized repository of organizational data and processes. The sheer scale and complexity of ERP solutions, which typically manage tens of thousands of business processes and store data in thousands of tables, present a significant opportunity for the integration of agentic artificial intelligence (AI). However, the implementation of agentic AI within ERP systems poses considerable challenges due to the intricate nature of these platforms. ERP solutions often comprise hundreds of millions of lines of code and are designed to accommodate a variety of industry-specific and regional requirements. This complexity necessitates a systematic approach to the development and integration of agentic AI capabilities. This paper addresses the critical research question: How can agentic AI business applications be systematically developed within ERP systems? To answer this, we employ a multi-faceted methodology encompassing the extraction of business requirements from real-world use cases, the design and development of a framework for agentic AI implementation, the evaluation of the proposed framework using actual ERP scenarios. Our research aims to bridge the gap between theoretical AI concepts and practical ERP implementation, providing a structured approach for organizations to leverage agentic AI within their existing ERP infrastructure. By doing so, we seek to enhance the capabilities of ERP systems, enabling more intelligent, adaptive, and efficient business processes across various industries and geographical regions.

INDEX TERMS Enterprise resource planning, ERP, artificial intelligence, AI, agentic AI, AI agents, enterprise AI, business AI, business applications, software integration, AI development.

I. INTRODUCTION

Enterprise Resource Planning (ERP) systems have become the cornerstone of modern business operations, offering integrated management of core business processes in real-time [1]. These systems have evolved significantly since their inception, from material requirements planning (MRP) systems in the 1960s to today's cloud-based, AI-enhanced solutions [2]. As businesses face increasing complexity and competition in the global marketplace, there is a growing need for ERP systems to become more intelligent, adaptive, and proactive in supporting decision-making processes [3]. Concurrently, the field of artificial intelligence has made remarkable strides, with agentic AI emerging as a promising paradigm for creating autonomous, goal-oriented systems capable of perceiving their environment and taking actions to achieve specific objectives [4]. The potential synergy between ERP systems and agentic AI presents an opportunity to revolutionize business process management and optimization [5]. However, the integration of agentic AI into ERP systems is not without challenges. ERP solutions are inherently complex, often comprising millions of lines of code and supporting a vast array of industry-specific and regional requirements [6]. This complexity, coupled with the need for reliability, scalability, and security in enterprise systems, necessitates a systematic approach to developing agentic AI within the ERP context [7].

AI agents, also known as intelligent agents, are autonomous entities that can perceive their environment, reason about it, and take actions to achieve specific goals

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Huei Cheng.

by utilizing artificial intelligence techniques. Russell and Norvig [4] define an agent as ''anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators''. AI agents and the traditional programming paradigm differ fundamentally in how they are designed, operate, and solve problems. AI agents use reasoning, learning, and decision-making to solve problems in dynamic and uncertain environments [8]. They are dedicated and can handle complex, ambiguous, or unstructured problems. Traditional programming relies on predefined rules and logic explicitly coded by developers, working well for structured, deterministic problems with clear requirements. AI agents can learn from data, experience, or interactions to improve over time without requiring explicit rules [9]. They adapt to changes in the environment without requiring explicit reprogramming. Traditional programming is based on static behavior; any changes or improvements require manual intervention and updates to the code. AI agents can operate autonomously, make decisions without direct human control, handling uncertainty and incomplete information effectively [10]. However, in the context of ERP systems typically humans are in the loop for ensuring legal compliance. Traditional programming executes deterministic decisions based on hardcoded logic and conditions, struggling with uncertainty or scenarios not explicitly defined. AI agents perceive and interact with dynamic environments using data feeds, chats, or APIs, responding to changes in real-time and adjusting their behavior accordingly [4]. Traditional programming operates in controlled environments with predictable inputs and cannot dynamically adapt to real-world changes without additional programming. The nature of ERP business processes varies significantly, ranging from highly structured, rule-based operations to more dynamic, data-driven decision-making scenarios. This diversity in task types makes ERP an ideal candidate for a hybrid approach that leverages both traditional programming and agentic AI. Thus, the hybrid approach combining agentic AI and traditional programming represents the optimal solution for next-generation ERP systems. It allows us to leverage the strengths of both paradigms: the reliability, accuracy, and explainability of traditional programming, and adaptability, intelligence, and data-driven insights of agentic AI. This synergy results in ERP systems that are not only robust and dependable but also capable of handling the complex, dynamic challenges of modern business environments.

Incorporating agentic AI into ERP systems presents significant challenges in implementation. This integration requires not only extensive expertise in agentic AI methodologies but also a comprehensive understanding of ERP system technologies, programming models, and intrinsic business processes. To tackle these challenges, our research will first conduct a thorough review of existing literature on agentic AI applications within ERP contexts, followed by a detailed explanation of our research methodology. We will then develop a solution architecture that addresses the business requirements which we identified by analyzing ERP AI use cases. The final step involves evaluating the effectiveness of our approach by applying it to real-world scenarios.

II. LITERATURE REVIEW

We conducted a systematic review of existing literature [11] with the aim of identifying, assessing, and interpreting the available scientific findings that are relevant to our research question: How can agentic AI business applications be systematically developed within ERP systems? To address this question, we devised the search term ''(Development OR Implementation) AND (Agentic AI OR AI Agents) AND (ERP OR Enterprise Resource Planning)''. We selected literature databases based on criteria such as relevant content and accessible licenses, including ACM Digital Library, AIS Electronic Library, DBLP Computer Science Bibliography, IEEE Xplore/IEEE-IET Library, SpringerLink, Google Scholar, Wiley Online Library, and ScienceDirect. This pre-defined search term was applied to the aforementioned databases. However, due to frequently obtaining an insufficient number of results, we relaxed the search string by removing some of the AND conditions. Furthermore, we expanded the search query to include keywords such as ''agent-based systems'', ''multi-agent systems'', ''intelligent agents'', ''cognitive agents'', ''integrated management information systems'' as synonym to ERP. In total we identified 609 scientific papers. After screening the titles, abstracts, and full texts of the identified publications, we shortlisted the papers focusing on business applications for discussion in this section.

The paper [12] explores the implementation of agentic AI systems in financial services, particularly through the formation of modeling and model risk management crews, demonstrating their ability to perform complex tasks such as credit card fraud detection, approval, and portfolio risk modeling. Strengths of the paper include a detailed breakdown of how agentic AI systems can automate various roles in financial services, showcasing the structured workflow of multi-agent collaboration and providing practical use cases with performance results. However, weaknesses lie in its narrow focus, which limits its potential applications outside the specific domain of financial services and its lack of exploration of generalization to other industries.

The paper [13] proposes a hybrid Internet of Things (IoT) and agent-based business process orchestration architecture that leverages open standards to balance structure and flexibility, offering a four-layered architecture for IoT-enhanced business process management, illustrated with a supply chain management example. The paper presents a novel approach to integrating IoT and agent-based systems with business process management, providing a comprehensive framework that combines structure with autonomy. The research lacks industry adoption and does not address certain practical integration challenges, particularly concerning agent management and the application of open standards.

The paper [14] proposes a novel framework using large language model (LLM) agents to enhance recommender systems by identifying and evaluating these agents and their relationships, thereby introducing a paradigm shift in how recommendations are generated. The strength of the paper lies in its systematic identification of unique LLM agents and their relationships, offering a comprehensive framework for future recommender system studies; however, its weakness is the lack of empirical evaluation data and practical implementation examples to validate the proposed framework.

The paper [15] explores the integration of vision-language models (VLMs) into radiology workflows, focusing on designing clinically relevant AI-supported applications such as draft report generation, augmented report reviews, visual queries, and summarizing patient imaging history, and identifying radiologists' and clinicians' perceptions and design considerations for these applications. Strengths and weaknesses of the paper include its comprehensive exploration of VLM capabilities in a specific healthcare context and its use of a collaborative and iterative design process involving diverse clinical stakeholders, which provides valuable insights and practical considerations for AI integration. However, the paper's findings, derived from non-functional prototypes in speculative user feedback sessions, necessitate further empirical validation and practical testing in real clinical settings to substantiate its applicability.

The paper [16] introduces an agentic AI system called LLexus that utilizes (LLMs) to automate the execution of troubleshooting guides for managing incidents in cloud-based services by converting these guides into executable plans that can mitigate incidents with minimal human intervention. The strengths of the paper lie in its systematic approach to leveraging LLMs for optimizing and automating incident management processes and presenting case studies that showcase the feasibility and cost-effectiveness of the LLexus system. However, its weaknesses include limited insight into the scalability of LLexus across diverse environments and the current dependencies for optimum functionality.

The paper [17] comprehensively explores the development and application of agentic AI, emphasizing its autonomy, flexibility, and ethical considerations across various sectors such as healthcare, finance, and adaptive software systems. The strengths of the paper lie in its detailed analysis of the technical, ethical, and societal aspects of agentic AI, providing a well-rounded overview that is valuable for researchers, developers, and policymakers. However, its weakness is the lack of focus on specific implementation strategies or detailed case studies that integrate agentic AI into existing systems, which may limit its practical applicability.

The paper [18] proposes a multi-agent system approach to enhance adaptive design in enterprise architecture, demonstrating flexibility and responsiveness to changing organizational needs through AI-driven predictive analytics and decision-making. The strengths of the paper include a novel approach integrating multi-agent systems for dynamic enterprise architecture design, presenting robust case studies, and demonstrating improved adaptability and efficiency in enterprise architecture design. However, its weaknesses lie in its heavy reliance on data quality, potential technological limitations, and the necessity for scalability refinements in complex systems.

The paper [28] comprehensively examines AI agents based on large models, their core capabilities (multimodal perception, retrieval-augmented generation, reasoning, and interaction), and implementation frameworks, culminating in a practical electromagnetic target analysis and recognition (ETAR) agent architecture. A strength of the paper is its thorough analysis of the technical components required for effective AI agents, including detailed exploration of frameworks like LangChain and AutoGen, and the practical implementation considerations for domain-specific applications. However, a notable weakness is the limited discussion of potential challenges and limitations when deploying large model-based agents in real-world electromagnetic target analysis scenarios, such as computational resource requirements, latency issues, and evaluation metrics for determining successful implementations.

The paper [29] advocates for a revolutionary approach to software development where AI serves as a universal translator between human intent and computer operations, enabling dynamic, context-aware systems that can adapt to changing user needs without requiring extensive reprogramming. A notable strength of the paper is its compelling articulation of the translation problems inherent in current software development processes and how generative AI, particularly through agentic systems, can overcome these limitations by allowing users to express goals in natural language rather than adapting to rigid interfaces. However, the paper falls short in addressing the practical challenges of implementing living software systems at scale, including considerations of computational requirements, privacy implications, and governance structures needed to effectively deploy and maintain such systems in enterprise environments.

In conclusion, while the papers resulted from the systematic literature review methodology deal with agentic AI regarding business applications, they do not cover the challenges regarding development of agentic AI in ERP systems. Key research gaps include limited exploration of practical implementation methodologies for integrating agentic AI within existing complex ERP architectures, insufficient attention to the challenges of scaling agent-based systems to enterprise-level operations, lack of standardized approaches for agent development and operations in ERP contexts. The existing literature provides valuable theoretical foundations and isolated use cases but fails to address the systematic development of agentic AI business applications specifically within ERP systems. This gap underscores the importance of

research focused on developing comprehensive frameworks and methodologies for agentic AI implementation in ERP.

III. RESEARCH METHODOLOGY

Since our literature review didn't uncover any significant work regarding agentic AI implementation in ERP software, we turned our attention to examining use cases of agentic AI within the ERP domain [19], [20] for analyzing the requirements and deducing corresponding solution. In our analysis of these 26 use cases, we concentrated on crucial questions, such as whether agentic AI was truly the optimal approach for addressing the underlying issue, or if rule-based methods could be equally effective. Additionally, we investigated the technical features necessary for the successful implementation of agentic AI use cases. To ensure thorough exploration of the topic, we scrutinized use cases across the entire spectrum of ERP modules. From this comprehensive analysis, we derived business requirements, which we then validated through feedback from customers and domain experts. Our investigation revealed that the primary gap was the lack of a framework to guide the development and operations of agentic AI use cases within the ERP context. To address this deficiency, we put forward a framework designed to standardize and streamline the implementation and operations of these use cases to reduce overall costs. To validate the effectiveness of our proposed framework, we provided a concrete implementation as proof, utilizing SAP's ERP platform, a widely recognized solution in the market. Finally, we applied our framework to implement real-world agentic AI use cases, demonstrating that our proposed solution is both practical and effective. Figure 1 shows a representation of our methodology, with the outer circle depicting our iterative process model and the inner circle showing the methods we employed.

In order to address the identified gaps, we facilitated established software engineering methodologies [21]. We utilized well-known requirement analysis techniques to identify requirements [22]. To transform these requirements into viable solution concepts, we implemented common software design practices [23]. Subsequently, we adhered to standard software development procedures to implement these concepts [23]. To guarantee the verifiability of our findings and conclusions, we adopted the design science research methodology [24]. This methodology provides a robust framework for validating both our processes and results. Furthermore, our work is grounded in the design and action domain, aligning with the taxonomy of information systems theory [25]. This domain serves as the theoretical foundation for our paper. The combination of these methodologies significantly enhances the rigor and reliability of our research and findings.

Our approach is further informed by fundamental artificial intelligence theories that provide the conceptual foundation for agentic systems. Drawing on the belief-desire-intention agent architecture proposed by Bratman [30] and formalized by Rao and Georgeff [31], we conceptualize agents as cognitive entities with beliefs about their environment, desires representing their goals, and intentions reflecting their committed plans. This theoretical framework allowed us to structure our agent implementations with clear reasoning patterns that align with business process requirements. Additionally, we incorporated insights from multi-agent systems theory [10], particularly regarding agent communication, coordination, and organizational structures, which helped address the complex interactions between different components in the ERP ecosystem. For the generative AI aspects of our implementation, we applied theoretical constructs from the transformer architecture [32], which has revolutionized natural language processing and serves as the foundation for large language models. To ensure practical viability, we also incorporated the theoretical framework of responsible AI [33], which emphasizes accountability, explainability, and transparency in AI systems. This guided our implementation of critical features like content filtering, legal compliance, and explainability mechanisms.

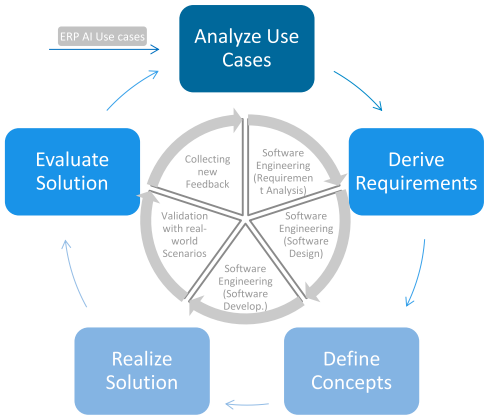

FIGURE 1. Process Model (outer circle) with Applied Methods (inner circle).

IV. IDENTIFIED REQUIREMENTS

The agentic AI use cases we examined were derived from a wide range of ERP core business processes. These processes encompassed the entire spectrum from idea to market, source to pay, plan to fulfill, lead to cash, recruit to retire, and acquire to decommission, as well as governance and finance. This comprehensive approach ensured that all segments of ERP modules were taken into account in our analysis. As previously outlined in our methodology section, we employed requirement engineering techniques [22] to extract both functional and non-functional requirements from the agentic AI use cases we investigated:

A. [REQ-01] VENDOR DIVERSITY

This requirement emphasizes the importance of maintaining an open ecosystem for generative AI models for realizing agents. It ensures that the ERP software is not locked into a single AI vendor or model, allowing for flexibility and choice.

This approach enables organizations to select the most suitable AI models for their specific needs, adapt to evolving technologies, and potentially leverage multiple models for different tasks. Vendor diversity also promotes competition and innovation in the AI space, potentially leading to better performance and cost-effectiveness for ERP users.

B. [REQ-02] BUILT-IN AGENTIC AI

This requirement calls for the systematic integration of agentic AI capabilities into business processes within the ERP system. The goal is to seamlessly embed AI agents into the workflow, ensuring that their features are accessible to the right users, in the appropriate contexts, and at optimal times. This integration should feel natural and intuitive, enhancing user productivity without disrupting established processes. It may involve context-aware AI assistants, automated decision support systems, or intelligent process automation, all tailored to specific roles and tasks within the ERP environment.

C. [REQ-03] STANDARDIZED DEVELOPMENT

This requirement aims to create a uniform programming model for developers working with AI agents within the ERP system, regardless of the underlying agentic technology. This standardization simplifies the development process, reduces the learning curve for developers, and promotes code reusability across different AI implementations. It involves creating abstraction layers, standardized APIs, or development frameworks that provide a consistent interface for interacting with various AI agents and models.

D. [REQ-04] STANDARDIZED OPERATIONS

Similar to the standardized development requirement, this focuses on providing a uniform configuration and operation experience for customers, irrespective of the agentic technology used. This standardization ensures that end-users and system administrators can manage and interact with AI agents consistently across different modules or applications within the ERP system. It includes standardized user interfaces for AI configuration, consistent monitoring and reporting tools, and unified processes for managing AI-related tasks.

E. [REQ-05] MODEL ADOPTION

This requirement addresses the need for tools and mechanisms to effectively adopt and customize generative AI models for specific agent implementations within the ERP system. This includes capabilities for prompt engineering, allowing developers to create and refine prompts that guide the agent's behavior. It also involves tools for incorporating embeddings, which can help in tailoring the agent's understanding to domain-specific knowledge and terminology relevant to the ERP context. These adoption mechanisms ensure that the AI agents can be fine-tuned to perform optimally within the specific business contexts of the ERP system.

F. [REQ-06] LEGAL COMPLIANCE

This requirement ensures that the integration of agentic AI within the ERP system adheres to various legal and regulatory standards. It encompasses several key areas: 1. Data privacy and protection: Ensuring that AI agents act according to authorization and handle sensitive business data in compliance with regulations like GDPR; 2. Consent management: Implementing systems to obtain and manage user consent for AI agent processing of personal data; 3. Automated decision-making: Providing transparency and safeguards around agent-driven decisions, especially those with significant impacts on individuals; 4. Read access logging: Maintaining detailed logs of data access by AI agents for auditing purposes; 5. Legal auditing: Implementing mechanisms to facilitate legal audits of AI agents and their decision-making processes; This requirement is crucial for maintaining trust, protecting user rights, and ensuring the legal operation of AI-enhanced ERP systems across different jurisdictions.

G. [REQ-07] CONTENT VALIDATION

This requirement focuses on implementing robust validation mechanisms for both inputs and outputs from AI agents within the ERP system. It serves multiple purposes: 1. Business correctness: Ensuring that AI-generated content aligns with business rules and processes; 2. Preventing discriminatory responses: Filtering out potentially biased or discriminatory content generated by AI; 3. Hate speech prevention: Identifying and blocking any form of hate speech or inappropriate language; 4. Data integrity: Validating input data to prevent corruption of AI agent or exploitation through adversarial inputs. These filtering mechanisms are essential for maintaining reliability, fairness, and professional standards of agent-augmented ERP operations.

H. [REQ-08] RESPONSE TIME

This requirement sets specific performance benchmarks for AI agent interactions within the ERP system. It defines three tiers of response times: 1. 150ms for instant feedback: Ensuring near-instantaneous responses for simple queries or interactions; 2. 1000ms for simple interactions: Allowing slightly more time for more complex, but still relatively straightforward tasks; 3. 3000ms for complex interactions: Providing a longer, but still reasonable, response time for more involved AI agent processing tasks; These benchmarks are crucial for maintaining user engagement and ensuring that AI agent augmentation enhances rather than hinders workflow efficiency in ERP operations.

I. [REQ-09] LIFECYCLE CUSTOMER

This requirement focuses on providing comprehensive lifecycle management tools and processes for customers utilizing AI agents within their ERP system. It encompasses: 1. Setup of agentic technology: Tools and guidance for initial deployment and configuration of AI agents; 2. Reuse tools:

Mechanisms to facilitate the reuse of AI components or configurations across different parts of the ERP system; 3. Monitoring: Robust monitoring capabilities to track AI agent performance, usage, and impact on business processes; This lifecycle management ensures that customers can effectively deploy, maintain, and optimize their use of AI agents throughout the lifespan of their ERP implementation.

J. [REQ-10] LIFECYCLE PROVIDER

This requirement addresses lifecycle management needs from the perspective of the ERP software provider. It includes: 1. Provisioning of agents: Tools and processes for deploying new AI agents; 2. Updating agent's code and content: Mechanisms for seamlessly updating the underlying code or knowledge base of AI agents; 3. Support handling: Systems for providing customer support related to AI agent functionalities; 4. Zero-downtime facilitation: Ensuring that updates and changes to AI agents can be implemented without disrupting ongoing ERP operations; These capabilities are crucial for the ERP provider to maintain and evolve the agentic AI capabilities of their system while minimizing disruption to customers.

K. [REQ-11] ERROR HANDLING

This requirement calls for robust mechanisms and tools to manage errors and exceptions in AI agent-driven processes within the ERP system. It encompasses: 1. Error handling and resolution: Systematic approaches to identify, log, and resolve errors in AI agent operations; 2. Business monitoring: Tools to monitor the impact of AI agent errors on business processes and detect anomalies; 3. Fallback techniques: Implementing backup mechanisms or processes that can take over if AI agents fail or produce unreliable results; These error handling capabilities are essential for maintaining the reliability and trustworthiness of AI agent-augmented ERP operations.

L. [REQ-12] MASS PROCESSING

This requirement addresses the need for handling large-scale, asynchronous processing using AI agents, despite the typically synchronous nature of generative AI models. It involves developing mechanisms to queue, manage, and process large batches of AI tasks efficiently. This capability is crucial for scenarios like end-of-day processing, large-scale data analysis, or bulk operations in ERP systems, ensuring that AI agents can be leveraged for high-volume tasks without compromising system performance.

M. [REQ-13] SCALABILITY

This requirement emphasizes the need for the agentic infrastructure within the ERP system to scale effectively. It should be able to handle increases in both the number of AI agent calls (volume scalability) and the number of customers using the system (multi-tenancy scalability). This scalability is crucial for maintaining performance and responsiveness as the usage of AI agents features grows within the ERP ecosystem.

N. [REQ-14] CONFIGURATION

This requirement focuses on providing flexible and powerful configuration options for agentic technologies within the ERP system. It includes: 1. Configurability of AI agent parameters: Ability to adjust settings like maximum thinking steps or temperature of generative AI models; 2. Hardware limit configurations: Options to set resource allocation limits for AI agent operations; 3. Persistence of customer configurations: Ensuring that custom configurations are not overwritten during system updates or upgrades; These configuration capabilities allow for parameterizing AI agent and ensuring that AI agent behaviors align with specific customer needs and constraints.

O. [REQ-15] EXTENSIBILITY

This requirement calls for mechanisms and tools that allow for the extension of AI agent capabilities within the ERP system. It should provide ways for customers or third-party developers to enhance the functionality of existing AI agents. Crucially, it stipulates that these customer extensions should be preserved during system updates and upgrades, ensuring continuity and protecting customer investments in customizations.

P. [REQ-16] LOCALIZATION

This requirement addresses the need for AI agents to support multiple languages and adapt to different regional contexts. It involves not just translating outputs but also ensuring that AI agents understand and respond appropriately to regional specific queries or contexts. This localization is crucial for global ERP deployments, ensuring that AI agent features are equally effective and appropriate across different regions and languages.

Q. [REQ-17] METERING

This requirement focuses on implementing usage tracking and measurement for AI agent consumption within the ERP system. It involves accurately measuring and reporting on the usage of AI agent resources, which is important for billing purposes, resource allocation, or performance optimization. This metering capability provides transparency and accountability in AI agent usage within the ERP environment.

R. [REQ-18] SECURITY

This requirement aims to implement a granular permission system that defines clear roles for AI agents, ensuring they can only access and manipulate data within their authorized scope. This approach minimizes the risk of data breaches and unauthorized operations. Deploying a comprehensive security strategy that includes strong encryption for data at rest and in transit, as well as adopting zero-trust policies. This multi-faceted approach requires continuous

authentication and authorization. Implementing safeguards to prevent unauthorized modifications of AI-generated outputs and defend against adversarial attacks.

S. [REQ-19] EXPLAINABILITY

This requirement is about developing user-friendly tools that translate complex AI-driven decisions into easily understandable explanations for business users. Implementing mechanisms to track and document the logic and data sources behind AI agent recommendations. This traceability ensures accountability and allows users to understand the rationale behind AI-generated suggestions, fostering confidence in the system's output. Designing the AI system to produce outputs that can be thoroughly audited and justified, particularly considering regulatory requirements. This includes maintaining detailed logs of AI processes, decisions, and the data used, enabling comprehensive reviews and demonstrating compliance with relevant regulations.

T. [REQ-20] AI ETHICS

This final requirement underscores the importance of ethical considerations in the deployment of AI agents within ERP systems. It mandates that all agentic applications must align with AI Ethics principles. This includes considerations such as fairness, transparency, privacy, and accountability in AI agent decision-making. Adhering to these ethical principles is crucial for building trust with users and ensuring that AI agent augmentation in ERP systems benefits businesses and individuals without compromising ethical standards.

The requirement validation process in our study was designed to ensure that the identified requirements for agentic AI integration within ERP systems were both comprehensive and aligned with real-world needs. Initially, requirements were extracted from a systematic analysis of diverse agentic AI use cases [19], [20] spanning all major ERP modules, including finance, supply chain, human resources, and customer management. To validate these requirements, a two-pronged approach was undertaken. First, iterative workshops were conducted with domain experts - including ERP architects, business process analysts, and AI developers - who reviewed and refined each requirement based on their practical experiences and anticipated operational challenges. This collaborative feedback loop enabled clarification of ambiguous requirements and identification of potential gaps. Second, the requirements were further validated through direct engagement with ERP customers across different industries. Structured interviews were held to understand end-user priorities, regulatory contexts, and integration pain points specific to their business environments. The combination of expert workshops and customer interviews ensured that the requirements were not only technically viable but also contextually relevant and prioritized according to business impact. This rigorous validation process established a robust foundation upon which the proposed solution architecture was constructed, significantly enhancing the practical applicability and acceptance of agentic AI in ERP settings.

V. SOLUTION ARCHITECTURE

In this section we depict the solution architecture as shown in Figure 2 for implementing agentic applications in context of ERP to resolve the previously explained business requirements. AI agents interact with generative AI models based on prompts for reasoning and execute actions itemized as tools. We distinguish between content-based and code-based AI agents. Content-based agents are defined in a declarative way, meaning their behavior is specified in advance through a design-time file (e.g., YAML file). These agents run centrally within the Conversational AI & Agent Runtime, which manages and stores their state. In contrast, code-based agents are built using coding. They operate outside of the Conversational AI & Agent Runtime, processed on any infrastructure and agent runtime specific to the ERP platform, and follow interface rules for enabling interoperability. The development and operations (DevOps) are streamlined by the DevOps Framework for code- and content-based agents. Content-based agents are running on the AI Technology Platform and particularly target cross-system scenarios where a declarative approach is sufficient. Code-based agents are deployed on the ERP platform and aim for sophisticated logic with processing of local ERP assets (e.g., private methods, accessing business and configuration data, querying analytical or search models, facilitating frameworks) where coding is required.

The AI Technology Platform (e.g., AWS, GCP, Azure, SAP BTP) is deployed side-by-side to the ERP system to offload tasks for scalability and avoiding resource bottlenecks for the ERP business processes. The Agent Runtime is a sophisticated framework that serves as the backbone for executing and managing AI agents. This Agent Runtime provides a controlled and secure environment for processing content-based AI agents. It manages the entire lifecycle of these agents, from their initial deployment to their eventual termination, ensuring they have access to the necessary resources throughout their existence. This management extends to maintaining the AI agents' state, a critical function that allows AI agents to persist information and maintain context between executions, enabling more coherent and continuous operations. One of the primary assets of the Agent Runtime lies in its ability to handle the intricate task of scheduling and orchestrating agent activities. In a complex ERP system, multiple agents may need to operate concurrently, each performing specialized tasks. The runtime expertly juggles these various agents, managing their execution to optimize system performance and prevent conflicts. Communication is another aspect of the Agent Runtime's functionality. It serves as a bridge between the AI agents and the outside world, providing standardized interfaces for interacting with databases, external systems, and other components of the ERP environment. This standardization simplifies the development process, allowing AI agent

creators to focus on business logic rather than the intricacies of system integration. Security is paramount in any enterprise system, and the Agent Runtime plays a crucial role in this regard. It implements robust security measures to ensure that AI agents operate within clearly defined boundaries. This includes managing access controls, ensuring that AI agents only have access to the data and systems they're authorized to use. By centralizing these security measures in the runtime, we can maintain a consistent and auditable security posture across all AI agent operations. The Agent Runtime also serves as a central point for monitoring and logging AI agent activities. This capability is invaluable for debugging, performance optimization, and auditing purposes. System administrators can gain insights into AI agent behavior, track resource usage, and identify potential issues before they impact business operations. This monitoring extends to version control and update management, ensuring that the correct version of each AI agent is always in use and that updates can be rolled out smoothly across the system. Resource management is another critical function of the Agent Runtime. In a large-scale ERP system, efficient allocation of computational resources is essential. The Agent Runtime dynamically manages memory, processing power, and other resources, ensuring that all AI agents have what they need to operate effectively without overwhelming the system. This management extends to scalability support, allowing the system to gracefully handle increases in the number of AI agents or overall workload.

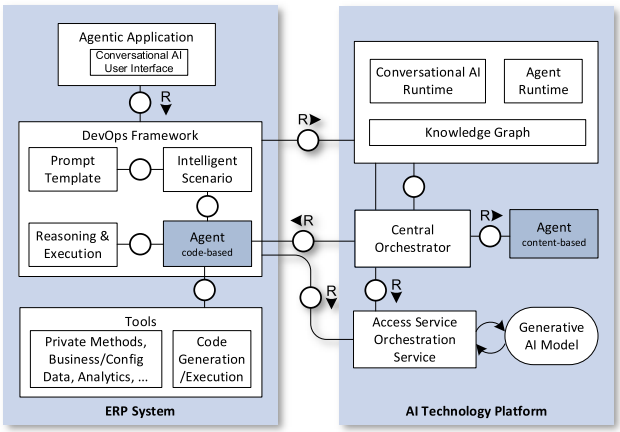

FIGURE 2. Solution Architecture.

The Conversational AI Runtime serves as the backbone for processing user inputs, managing dialogue flow, and generating appropriate responses. It includes natural language processing based on generative AI models for interpreting user input, a dialogue management system for tracking conversation state, a response generation module for creating AI outputs and an integration layer for connecting to the ERP systems. The Conversational AI Runtime ensures these components interact smoothly, allowing the AI to maintain context across multiple turns of conversation, handle various input types, and produce coherent, contextually appropriate responses. Importantly, the runtime also manages performance optimization, error handling, and includes mechanisms for continuous learning and improvement based on interactions. The Conversational AI UI as part of the Agentic Application is a specialized web-based client designed to facilitate continuous interaction between users and the ERP system. This interface presents a chat environment that can display diverse response formats, including lists, interactive buttons, and graphical charts. Its versatility allows for seamless integration into ERP systems, as it can function within an embedded frame, separate from the primary application window.

The Knowledge Graph as part of the runtime is a structured representation of information that captures entities, their attributes, and the relationships between them in a way that machines can understand and reason about. It facilitates implementing AI agents by providing a rich, interconnected data structure that AI agents can leverage for various tasks. The Knowledge Graph enables AI agents to understand the context of their environment more comprehensively. In an ERP setting, this could mean understanding the relationships between different business entities (e.g., customers, products, orders, suppliers) and their attributes. By representing information in a graph structure, the Knowledge Graph allows more sophisticated reasoning processes for AI agents. AI agents can traverse the graph to infer new information, identify patterns, and make logical deductions. In complex ERP environments, data and processes often reside in multiple, disparate systems. The Knowledge Graph can serve as a unified layer that integrates data and processes from various sources, providing AI agents with a holistic view of the organization's information landscape. The Knowledge Graph is integrated with the Conversational AI and Agent Runtime, allowing content-based agents to leverage this structured knowledge directly while code-based agents require a corresponding remote API call.

The Central Orchestrator serves as the nerve center in a multi-agent ERP system. It acts as a supervisory entity that coordinates and manages the activities of various AI agents operating within the ERP system. This component is responsible for ensuring smooth collaboration, efficient resource utilization, and coherent goal-oriented behavior across the entire agent network. The Central Orchestrator oversees task allocation, facilitates inter-agent communication, and maintains a global view of the system's state. It resolves conflicts that may arise between agents, aligns their individual actions with overarching system objectives, and dynamically adjusts the system's behavior based on changing conditions or requirements. The Access Service plays a crucial role in the solution architecture by providing a unified interface to diverse generative AI models (e.g., GPT, Mistral, Gemini). Our approach is based on generative AI models as they are multipurpose, handle reasoning and cover systematically the different ERP domains. The Access Service abstracts the complexities of connecting to those

AI models, offering a streamlined approach to interact with these. Beyond simple connectivity, the Access Service incorporates advanced features that enhance the capabilities of generative AI. One such feature is Retrieval Augmented Generation (RAG), which combines the generative power of AI models with the ability to retrieve and incorporate relevant information from external sources. This results in more informed and contextually appropriate outputs. Additionally, the Access Service includes vital safeguards in the form of content filtering and data masking. These features, collectively referred to as the Orchestration Service, ensure that the AI-generated content adheres to specified guidelines and protects sensitive information. Content filtering helps maintain the quality and appropriateness of the output, while data masking safeguards private or confidential data from unnecessary exposure. By consolidating these functionalities, the Access Service not only simplifies the use of generative AI models but also enhances their practical application in various contexts, particularly in environments where data security and content quality are paramount.

Code-based agents are realized with classes (e.g., Java, ABAP) following predefined interfaces to allow composition by the Central Orchestrator. The DevOps Framework is used to manage the necessary Prompt Templates and communication to generative AI models via the Access Service as shown in Figure 2. The DevOps Framework standardizes the development and operations of Agentic Applications in the ERP system. It covers the entire process from development to configuration and deployment. The core entity of the DevOps Framework is the Intelligent Scenario, a design time artifact that contains all objects needed for an Agentic Application. The DevOps Framework facilitates a prompt executor, which works with pre-defined Prompt Templates. These templates provide standardized structures for generative AI model interactions, particularly useful for large language models. The Prompt Templates are utilized to specify the instruction and tasks of the AI agent. At runtime, the Agentic Application dynamically fills the templates with specific values using APIs provided by the DevOps Framework. By encapsulating all elements within the Intelligent Scenario, the DevOps Framework ensures easier management and deployment of Agentic Applications in the ERP environment, effectively bridging design-time configuration with runtime execution. An additional component for Reasoning & Execution of code-based agents is envisioned. Agent execution and reasoning are fundamental processes that define how AI agents operate and make decisions within a given environment. Reasoning is the cognitive process by which an agent processes information, draws inferences, and makes decisions. It's the *thinking* part of an AI agent's operation. In the context of ERP systems, reasoning capabilities allow agents to interpret complex business scenarios, apply rules and logic, and determine appropriate actions or recommendations. For content-based agents, reasoning is often defined in their declarative specifications. The Agent Runtime interprets these rules and executes the reasoning process accordingly. This might involve evaluating conditions, following predefined decision paths, or applying simple inference mechanisms based on the current state and input data. Code-based agents, however, can implement more sophisticated reasoning algorithms. These might include probabilistic reasoning, machine learning models, or complex heuristics tailored to specific business problems. The Reasoning & Execution component in our architecture provides support for these advanced reasoning capabilities, allowing AI agents to leverage the full power of the ERP platform and integrated AI services. A key aspect of reasoning in our solution architecture is the integration with the Access Service and the use of generative AI models. This integration allows AI agents to perform natural language processing, understand context, create execution plans and generate human-like responses or recommendations. The Code Generation & Execution, in particular, enables dynamic code generation based on generative AI and the execution on the ERP platform, adding a layer of adaptability to the AI agents' reasoning capabilities. Agent execution refers to the actual running of an agent's code or the interpretation and enactment of its declarative specifications. This process involves activating the agent, processing its inputs, performing its designated tasks, and producing outputs or actions. In our architecture, agent execution is handled differently for content-based and code-based agents. For content-based agents, execution primarily occurs within the Agent Runtime. The runtime ensures that the agent operates consistently with its defined behavior, handling aspects such as timing, event triggers, and interaction with other system components. Code-based agents, on the other hand, have their execution more directly tied to the underlying programming language and infrastructure of the ERP platform. The execution of these AI agents is managed by the code runtime environment of the ERP platform, which handles memory allocation, processing threads, and integration with the broader ERP business processes. Our architecture includes the Reasoning & Execution component specifically tailored for these code-based agents, providing specialized support for their operation within the ERP ecosystem. Reasoning and execution are deeply intertwined processes. As an AI agent executes its tasks, it continuously engages in reasoning to interpret results, make decisions, and determine next steps. This creates a dynamic, adaptive behavior where the AI agent can respond to changing conditions and new information in real-time. The architecture emphasizes transparency and explainability in agent reasoning. This is particularly important in business contexts where decisions need to be auditable and understandable. The system provides mechanisms to trace the reasoning process, allowing users to understand how and why an agent arrived at a particular decision or recommendation.

Tools refer to the resources, methods, or technologies that an AI agent uses to perform its actions or enhance its capabilities. Code-based agents can use all frameworks and artifacts of the ERP platform. This includes, for example,

application artifacts like hundreds of thousands of private methods, classes, analytical and search models, views for accessing business and configuration data. Conversely, code-based agents can resolve complex ERP use cases. As already mentioned, the code-based agents could also utilize the generative AI to create code and execute it correspondingly. The Central Orchestrator instruments content- and code-based agents. This component allows for a hybrid approach across systems where both types of agents can coexist and complement each other, providing a flexible solution that can address a wide range of business needs. The choice between content-based and code-based agents depends on several factors, including the complexity of the task, the available skills within the organization, and the need for customization. Content-based agents are ideal for scenarios requiring easy management of behaviors and handling cross-system needs, while code-based agents shine in complex, customized AI solutions that require deep integration and sophisticated logic. An Agentic Application can consist of code- and content-based agents. The interplay of the key building blocks of the solution architecture are shown in Figure 3 for a conversational content-based AI agent.

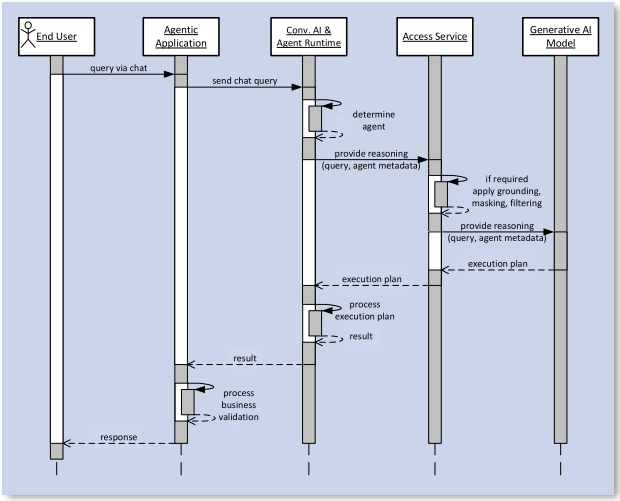

FIGURE 3. Processing conversational content-based AI Agent.

VI. EVALUATION

To demonstrate the practical feasibility of the proposed solution architecture for implementing agentic AI applications in ERP software, we depict a concrete framework. The framework consists of multiple components and is founded on the previously explained solution architecture. A practical implementation must be built on an existing ERP platform. We selected SAP ERP for this purpose as it is widespread, well known and the market leading product. Given the extensive functionality of the framework integrating agentic AI into ERP systems, we will selectively highlight key components. Explicitly, we will focus on the Intelligent Scenario, the content-based Agent, and the Orchestration Service, as depicted in Figure 2.

Once the prompt engineering phase is concluded, we can initiate the development of the agentic application. This process begins with defining the Intelligent Scenario, as shown in Figure 4. The Intelligent Scenario has a name and a description (Figure 4, section I). Furthermore, an agent class is registered for the Intelligent Scenario (Figure 4, section II). In the case of content-based agents this class contains coding for the API to interact with agent. In the case of code-based agents, the class includes additional coding for implementing the agentic logic but also realizing of interfaces for e.g. reasoning and tool access.

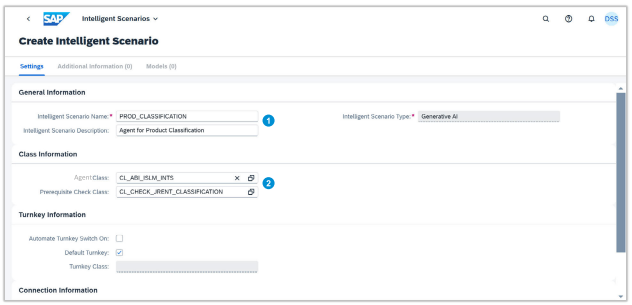

FIGURE 4. Intelligent Scenario – Creation.

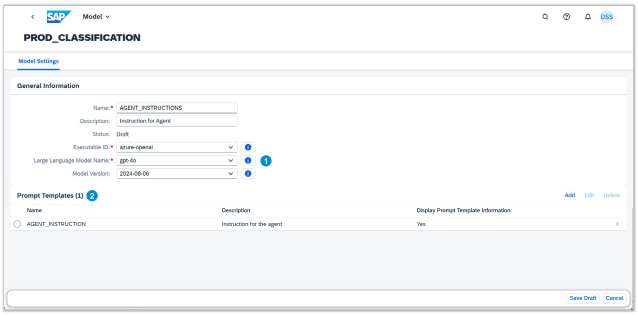

FIGURE 5. Intelligent Scenario – Prompt Template.

The Intelligent Scenario encompasses Prompt Templates, which are the result of the prompt engineering process (Figure 5, section II). Moreover, the Intelligent Scenario contains crucial metadata, including details about the specific generative AI model in use and its corresponding version number (Figure 5, section I). The Prompt Templates incorporates instructions for the agent, which are processed by generative AI models.

In context of reasoning and execution of the agent, the prompts are sent to generative AI model via the Access Service. This service manages the connection to and interaction with the generative AI model. Within the Access Service, there is a component called the Orchestration Service as shown in Figure 6. This Orchestration Service provides a set of functions commonly required for agentic applications in ERP systems like grounding, data masking and content

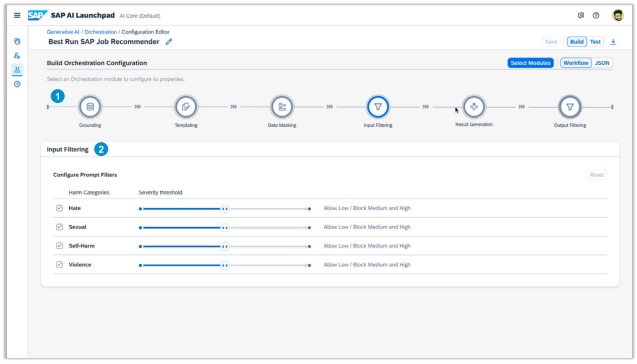

FIGURE 6. Access Service – Orchestration Service.

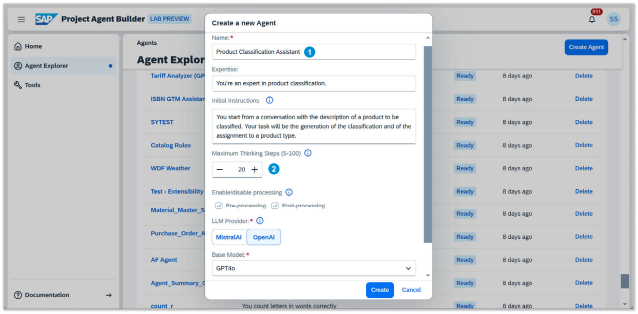

FIGURE 7. Content-Based Agent – Creation.

filtering. Those modules within the Orchestration Service can be combined to create a pipeline (Figure 6, section I). This pipeline is executed through a single API call. The output of one module in this pipeline becomes the input for the next module. The Orchestration Service centrally controls the order in which these modules are executed. However, users have the option to customize the settings for each module (Figure 6, section II) and omit optional modules by submitting an orchestration configuration. This capability allows us to tailor the orchestration process for specific requirements of the agentic application.

For defining content-based agents our framework provides the design time tool as depicted in Figure 7. For code-based agents just a standard Integrated Development Environment (IDE) for coding is utilized. The content-based agent has a name (Figure 7, section I) as identifier. When it comes to prompting generative AI models one of the best methods is role-playing. The Expert field specifies the role of the AI agent. When using agent-to-agent communication the AI agents consider the expertise of other AI agents for delegating the tasks. Initial Instructions are part of the Prompt Templates and play a crucial role in guiding an AI agent's behavior and decision-making process. These instructions, which are distinct from regular chat messages, are evaluated with system privileges and are prominently placed at the top of the agent's context. They provide detailed guidance on how the agent should handle various situations, utilize tools, respond to queries, and process responses.

The concept of Maximum Thinking Steps is an important constraint in AI agent operations (Figure 7, section II). It defines the upper limit of steps an agent can take to complete a conversation or task. If this limit is exceeded without achieving the desired outcome, the operation is automatically terminated, and the user is notified. To optimize agent performance, Pre- and Post-processing steps can be implemented to refine the input provided to the agent and the output delivered to the human or system. However, it's important to exercise caution when modifying well-crafted prompts, as unnecessary alterations may lead to unintended behaviors or the addition of superfluous information. The goal is to enhance the AI agent's effectiveness without compromising the integrity of its core functionalities.

We can interact with AI agents in different ways depending on their design. Conversational agents primarily communicate through chat interfaces, while non-conversational agents are typically accessed via APIs or respond to specific events. After an AI agent completes its execution, it's often valuable to examine its trace as shown in Figure 8. This feature is particularly useful to visualize the exact path the agent took to arrive at its final answer or complete a given task (Figure 8, section I). By activating this function, a comprehensive graph illustrates each discrete step the agent undertook during its process. This graph provides a visual representation of the agent's decision-making journey, offering insights into its problem-solving approach. To gain an even deeper understanding of the agent's actions, we can interact with the graph itself. By clicking on each node of the graph, we can delve into the specifics of each phase of the agent's process (Figure 8, section II). This step-by-step breakdown allows for a thorough analysis of the AI agent's methodology, helping to illuminate its reasoning and potentially identify areas for improvement or optimization in future iterations.

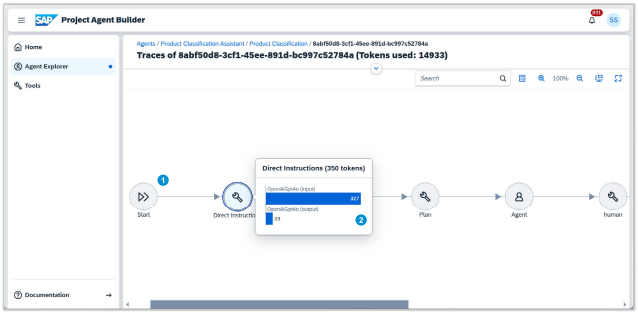

FIGURE 8. Content-Based Agent – Execution.

The proposed solution framework has been successfully utilized across numerous agentic AI use cases [19], [20] spanning different ERP domains, demonstrating its versatility and adaptability. We will apply the introduced framework on an exemplary agentic AI application related to ERP dispute management and subsequently discuss additional use cases.

The dispute management use case centers on handling billing discrepancies in subscription-based services. This scenario is ubiquitous across various subscription services, such as music or movie streaming platforms, where customers are billed recurring monthly fees. In a typical situation, customers receive their monthly invoices for subscribed services. The dispute process is initiated when a customer notices an unexpected change in their billing amount. For instance, a subscriber who has been consistently charged 200€per month might observe an increase to 211.40€in a subsequent bill. This 11.40€increase naturally prompts the customer to question the charge and potentially open a dispute. The dispute management process is designed to address several critical customer expectations. Primarily, customers desire swift resolution of their disputes, given that it involves their finances. They also expect accuracy in the resolution to avoid repeated disputes over the same issue. While cost-effectiveness is more of a concern for the service provider, efficient dispute resolution ultimately benefits both parties. The dispute typically commences with the customer sending an email to query the unexpected charge. This email serves as the entry point for the dispute management process. Subsequently, the system needs to validate the dispute, investigate the reason for the billing change, and determine an appropriate resolution. The investigation might reveal that while there was a contractually agreed 2% annual increase (raising the fee from 200€to 204€), the actual billed amount of 211.40€exceeds this agreed-upon increase. This discrepancy forms the basis of a valid dispute.

As illustrated in Figure 9, the AI agent plays a crucial role in resolving the dispute management use case by automating and streamlining the entire process from initial contact to final resolution. This intelligent solution handles disputes efficiently, accurately, and cost-effectively, addresses the key expectations of both customers and the company. When a customer sends an email regarding a billing discrepancy, the AI agent first intercepts and analyzes the content. It uses natural language processing to determine if the email constitutes a dispute. If confirmed as a dispute, the agent automatically creates a case in the system, eliminating the need for manual case creation by human staff. The AI agent then proceeds to analyze the dispute in detail. It accesses relevant data sources, including the customer's billing history, contract details, and applicable pricing policies. In the given example, the agent would compare the disputed invoice amount of 211.40€with the previous bills of 200€and the contract terms stipulating a 2% annual increase. Using its analytical capabilities, the AI agent identifies the discrepancy between the billed amount and the contractually agreed amount. It calculates that the correct charge should be 204€(reflecting the 2% increase) rather than the billed 211.40€. The agent then determines the overcharged amount of 7.40€. Based on this analysis, the AI agent recommends creating a credit note for the overcharged amount. It can even generate this credit note automatically, subject to predefined approval thresholds. For smaller amounts, the agent might be authorized to release the credit note without human intervention, further expediting the process. Throughout the resolution process, the AI agent documents each step and decision in the dispute case notes. This ensures full transparency and provides an audit trail for future reference.

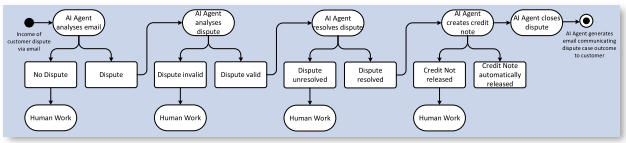

FIGURE 9. Dispute Management – AI Agent Process.

The agent also prepares a clear explanation of the resolution, which can be used in communication with the customer. Once the credit note is issued, the AI agent can automatically draft and send an email to the customer explaining the resolution, including details of the credit issued and the reason for the original discrepancy. This communication helps maintain customer satisfaction by providing a quick and thorough response. Finally, the AI agent closes the dispute case in the system, updating all relevant records. It may also flag the case for review if it identifies a potential systemic issue that could lead to similar disputes in the future. Figure 10 shows the realization of dispute management scenario in the ERP system. The dispute case with the corresponding details is displayed in the dispute management application. On the right side, a chat dialog is depicted with which the caseworker can interact with the AI agent based on natural language. The AI agent explains the identified discrepancy and recommends actions to resolve the dispute.

Our empirical evaluation of the dispute management implementation demonstrates improvements across multiple dimensions defined in our requirements framework. Regarding response time requirements (REQ-08), the AI agent consistently achieved sub-1000ms processing for standard dispute analysis, with 81% of simple billing discrepancy cases resolved within the target threshold. Complex multi-contract disputes averaged 3,100ms processing time, well within the 3000ms specification for complex interactions. To address scalability (REQ-13), we conducted load testing with simulated concurrent dispute volumes ranging from 50 to 500 simultaneous cases, demonstrating linear performance scaling with average response times increasing from 847ms to 2,340ms respectively. Error analysis revealed that 12.3% of dispute management failures occurred due to incomplete email parsing, 5.8% from invalid contract data references, and 3.2% from generative AI model timeouts during peak processing periods. Mass processing requirements (REQ-12) were validated through batch operations handling 1000 historical dispute records, completing analysis in 0.9 hours versus an estimated 128 human-hours for equivalent non-agentic approach. Accuracy metrics are acceptable but could not outperform non-agentic approach. The AI agent achieved

81.7% accuracy in dispute classification, compared to 97.3% for non-agentic approach in controlled testing scenarios. The main failure modes observed included incomplete email parsing, invalid contract data references, and occasional model timeouts. These were mitigated through stricter data validation, fallback execution paths, and continuous model retraining on error cases. Cost efficiency analysis revealed a 67% reduction in operational expenses per resolved dispute. These metrics validate our architectural approach while demonstrating practical viability for enterprise-scale deployment across diverse ERP environments. Moreover, as the AI agent handles more cases, it continually learns and improves its capabilities. It can identify patterns in disputes, helping the company proactively address recurring issues. This might involve suggesting updates to billing systems, recommending clarifications in contract language, or proposing improved communication strategies for price changes.

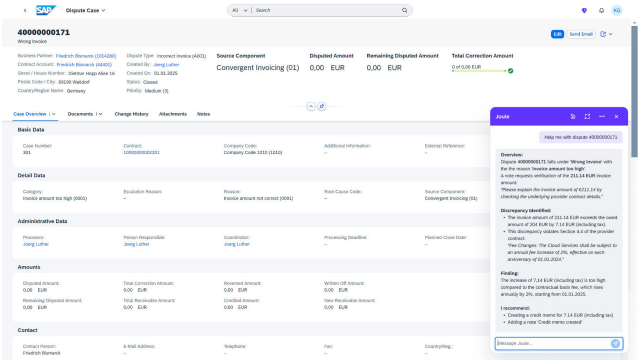

FIGURE 10. Dispute Management – AI Agent Conversation.

The implementation of legal compliance (REQ-06), security (REQ-18) and AI Ethics (REQ-20) reuses existing ERP infrastructure and concepts [36]. Our framework extends established ERP change logging mechanisms to capture AI agent decisions including input data sources, reasoning paths and generated outputs, creating an immutable audit trail for regulatory inspection. Data protection and privacy is enforced through the available ERP data access control frameworks and existing data lineage tracking that documents personal data access, processing purposes, and retention periods for each agent interaction, enabling automated subject access request responses and data deletion workflows. The development process for agents considers continuous reviews by AI Ethics experts to avoid deviation from ERP established ethical guidelines. When the agents interact with generative AI models corresponding input and output filtering is utilized via the Orchestration Service (Figure 6) to prevent hate, violence, sexual and self-harm content. Model monitoring (REQ-09) integrates with standard ERP technical and business monitoring tools. Ongoing is our research on advanced model monitoring mechanisms that shall continuously assess AI agent behavior for bias detection, performance degradation, and ethical deviation through statistical analysis of decision patterns across demographic groups, business units, and temporal periods.

While the AI agent can handle most aspects of the dispute resolution process autonomously, it's designed to involve human agents when necessary. Complex cases or those requiring judgment beyond its defined parameters are escalated to human staff, ensuring that all disputes are handled appropriately. By leveraging AI in this manner, the company can provide more efficient and satisfactory dispute resolution experience for its customers while optimizing its internal processes and resources.

In conclusion, to evaluate the proposed solution architecture for implementing agentic AI into ERP software, we presented a framework. This proved that the theoretical solution concept can be practically realized. We successfully applied the framework on the exemplary dispute management scenario, but its versatility has since been proven across numerous agentic AI use cases, all of which have been successfully incorporated into SAP's ERP suite [19], [20]. Furthermore, the framework itself has been incorporated into SAP's core ERP product [26] and has empowered customers and partners to create their own bespoke agentic AI applications, highlighting its accessibility and adaptability. The successful deployment of the framework in production environments provides empirical evidence of its efficacy under actual operational conditions. All ERP products are implemented with diverse programming languages. Nevertheless, from theoretical computer science, we know that all programming languages are turing-complete [27]. This fundamental principle of theoretical computer science ensures that they possess equivalent expressive capabilities. Consequently, our successful implementation of the framework using SAP's ABAP language strongly concludes that it can be repeated across other ERP platforms, regardless of their underlying programming languages. This language-agnostic nature of our framework significantly enhances its potential for widespread adoption and application. It offers a universal approach to implementing agentic AI applications across various ERP ecosystems, thereby extending its validity and relevance beyond a single vendor's solution. This broad applicability positions our framework as a pivotal tool in the evolving landscape of intelligent ERP systems.

While the turing-completeness principle provides theoretical assurance that our framework can be adapted across programming languages, the practical implementation requires systematic architectural mapping and careful consideration of platform-specific constraints. The foundational challenge lies in translating SAP's specific technological stack to heterogeneous ERP environments. Our SAP implementation leverages ABAP classes for code-based agents, the Business Technology Platform for AI model access, and proprietary configuration mechanisms for intelligent scenarios. When adapting to platforms like Oracle ERP Cloud, Microsoft Dynamics, or open-source alternatives like ERPNext, each component requires careful architectural translation. Beginning with the Agent

Runtime component, which in SAP operates within the ABAP application server. For Oracle environments, this translates to Oracle Application Development Framework containers, while Microsoft Dynamics would utilize the Common Data Service runtime. The key principle involves identifying each platform's equivalent execution environment that provides lifecycle management, state persistence, and security boundaries similar to our SAP implementation. The Access Service presents another critical mapping challenge. Our SAP framework integrates with SAP AI Core for generative model access, but other platforms require different integration patterns. Oracle's AI Services, Azure Cognitive Services, or AWS integration endpoints each demand platform-specific API adaptations while maintaining our standardized orchestration pipeline concept. Data integration represents perhaps the most complex adaptation requirement. SAP's extensive metadata framework, with its thousands of business objects and configuration tables, must be mapped to each target platform's data architecture. This involves creating platform-specific knowledge graphs that capture business entity relationships while preserving the reasoning capabilities our agents require. While SAP uses transport management, other platforms employ different artifact management approaches requiring careful consideration of versioning, configuration persistence, and multi-tenancy support across diverse technological foundations. To illustrate the cross-platform implementation, we provide exemplary API specifications that abstract platform-specific complexities through standardized interfaces. Our RESTful API design follows OpenAPI 3.0 standards with core endpoints including Agent Management (POST /api/v1/agents, GET /api/v1/agents/{id}, PUT /api/v1/agents/{id}/config), Execution Control (POST /api/v1/agents/{id}/execute, GET /api/v1/executions/{id}/status, POST /api/v1/executions/{id}/cancel), and Knowledge Integration (GET /api/v1/knowledge/entities, POST /api/v1/knowledge/query). The API specifications define platform-agnostic JSON schemas for agent configurations, prompt templates, and execution results, enabling seamless translation between different platforms native object models.

Quantitative assessment of our agentic AI framework reveals minimal local resource overhead since generative AI models are hosted on hyperscaler infrastructure (e.g., AWS, Azure, GCP) with pay-per-use billing models. The ERP system overhead primarily consists of Agent Runtime operations, consuming an estimated 1.2GB additional memory per 100 concurrent agents and 1-2% CPU increase for agent orchestration, state management, and API communication handling. Network bandwidth utilization is expected to 50-80KB per agent execution for inference API calls to hyperscaler endpoints, with response payload sizes typically ranging from 2-15KB depending on execution plan complexity. Local energy consumption is estimated to increase by 1-2% due to additional CPU processing for agent orchestration and network communications, while the energy-intensive generative AI inference occurs remotely on hyperscaler infrastructure optimized for AI workloads with renewable energy commitments. These measurements were obtained from controlled ERP test environments using built-in performance telemetry tools. Cost analysis shows inference expenses of $0.002-0.008 per agent execution depending on prompt complexity and chosen model tier, making operational costs predictable and scalable based on actual usage rather than fixed infrastructure investments. The pay-per-use model enables organizations to scale agentic capabilities without upfront hardware investments while maintaining cost transparency through detailed usage analytics provided by hyperscaler billing systems.

The evaluation of the proposed agentic AI framework carries several notable limitations. The evaluation has primarily considered short-term feasibility, technical correctness, and initial deployment outcomes. Long-term operational effects, including the sustainability of AI model updates, maintenance challenges, and the resilience of agentic components under evolving business requirements or heavy system loads, have not been systematically explored. Moreover, organizational and human factors - such as change management, user acceptance, AI oversight, and governance - have not been addressed, even though these are critical for widespread adoption and sustained value realization in real enterprise contexts. Beyond these evaluation-specific limitations, the framework itself introduces overhead. The addition of agent runtimes, orchestration layers, knowledge graph integration, and generative AI model access increases the overall system complexity. This added abstraction results in greater computational resource consumption, which can introduce latency and potentially impact real-time ERP performance, especially under high-throughput or batch-processing workloads. Furthermore, integrating security controls, explainability features, content filtering, and lifecycle management creates additional administrative and development burdens, requiring specialized expertise and ongoing maintenance. These operational demands may lead to increased infrastructure costs and a higher risk of technical debt if not proactively managed.

VII. CONCLUSION

This paper has presented a comprehensive framework for systematically implementing agentic AI applications within Enterprise Resource Planning (ERP) systems. Through our research methodology, we identified key business requirements for integrating AI agents into complex ERP environments and proposed a solution architecture to address these requirements. The architecture encompasses critical components such as Agent Runtime, Conversational AI Runtime, Knowledge Graph, Central Orchestrator, and DevOps Framework, providing a holistic approach to agentic AI integration.

Our evaluation demonstrated the practical feasibility of the proposed solution architecture through the development of a concrete framework based on SAP's ERP platform. The successful application of this framework to numerous real-world

scenarios validates its effectiveness in addressing complex business processes within ERP systems. The framework's ability to handle both content-based and code-based agents offers flexibility in addressing diverse business needs, from simple cross-system interactions to sophisticated, customized AI solutions.

The successful deployment of the framework in production environments and its incorporation into SAP's core ERP product provide empirical evidence of its efficacy under actual operational conditions. Furthermore, the framework's language-agnostic nature, rooted in the principle of turing-completeness, suggests its potential for implementation across various ERP platforms, regardless of their underlying programming languages. Thus, this research contributes significantly to bridge the gap between theoretical AI concepts and practical ERP implementation.

Our research makes contributions to theory in both information systems and artificial intelligence domains. First, we extend the literature on agent-based computing by providing a novel integration framework specifically designed for complex enterprise environments. This addresses a critical gap in existing research, which has predominantly focused on either theoretical agent architectures or isolated use cases rather than comprehensive implementation approaches. Second, our work advances the understanding of human-AI collaboration in organizational contexts by proposing concrete mechanisms for embedding agentic intelligence within established business processes. The hybrid approach combining code-based and content-based agents creates a new theoretical lens for understanding how different forms of AI agency can be effectively combined within enterprise systems. This hybrid model challenges the binary distinction often made between rule-based and learning-based approaches in AI literature.

From a practical perspective, our framework offers several immediate benefits to organizations seeking to implement agentic AI within their ERP environments. The standardized development and operations approach significantly reduces the technical barriers to entry for organizations wishing to leverage agentic AI. By providing a consistent programming model and operational interface regardless of the underlying AI technologies, our framework enables faster implementation cycles and reduced training costs. By providing a structured approach for organizations to leverage agentic AI within their existing ERP infrastructure, we aim to enhance the capabilities of ERP systems, enabling more intelligent, adaptive, and efficient business processes across various industries and geographical regions. As the field of AI continues to evolve, frameworks like the one presented in this paper will play a crucial role in shaping the future of intelligent enterprise systems.

Despite the contributions of our research, several limitations warrant acknowledgment. First, our research did not extensively evaluate the long-term maintenance aspects of agentic AI applications in ERP systems. As generative AI technologies evolve rapidly, ensuring the sustainability and backwards compatibility of deployed agents remains a significant challenge. The implications of AI model drift, changing language capabilities, and evolving security vulnerabilities require further investigation in operational contexts. Second, our research focused primarily on technical and architectural considerations rather than organizational change management aspects of AI adoption. The human and organizational factors in successful agentic AI implementations, including training requirements, role transformations, and evolving governance structures, represent important dimensions not fully addressed in our current framework.

This research opens several promising avenues for future investigation. First, extending our framework to accommodate emerging AI capabilities such as multimodal agents that can process visual, audio, and textual inputs simultaneously would significantly enhance the applicability of our approach. As generative AI models increasingly incorporate these capabilities, ERP systems will need architectural patterns to effectively leverage these advancements. Second, developing comprehensive methodologies for testing and quality assurance specific to agentic AI in ERP contexts represents a critical research direction. The non-deterministic nature of AI agents creates unique challenges for ensuring reliability and consistency in business-critical applications. New approaches combining traditional software testing with AI-specific evaluation techniques are needed to ensure robustness in production environments. Finally, investigating cross-agent collaboration patterns and coordination mechanisms within enterprise contexts would extend our current framework. As organizations deploy multiple AI agents across different business processes, understanding how these agents can effectively communicate, coordinate activities, and resolve conflicts becomes increasingly important. Theoretical and practical approaches to multi-agent orchestration within ERP constraints would significantly advance the field.

REFERENCES

[1] T. H. Davenport, "Putting the enterprise into the enterprise system," *Harvard Bus. Rev.*, vol. 76, no. 4, pp. 121–131, 1998.

[2] F. R. Jacobs and F. C. Weston, "Enterprise resource planning (ERP)—A brief history," *J. Oper. Manag.*, vol. 25, no. 2, pp. 357–363, 2007.

[3] L. D. Xu, E. L. Xu, and L. Li, "Industry 4.0: State of the art and future trends," *Int. J. Prod. Res.*, vol. 56, no. 8, pp. 2941–2962, Apr. 2018.

[4] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., London, U.K.: Pearson, 2016.

[5] Y. Duan, J. S. Edwards, and Y. K. Dwivedi, "Artificial intelligence for decision making in the era of big data—Evolution, challenges and research agenda," *Int. J. Inf. Manage.*, vol. 48, pp. 63–71, Oct. 2019.

[6] M. A. Rashid, L. Hossain, and J. D. Patrick, "The evolution of ERP systems: A historical perspective," in *Enterprise Resource Planning: Solutions and Management*. Hershey, PA, USA: IGI Global, 2002, pp. 35–50.

[7] S. Schneider and M. Leyer, "Me or information technology? Adoption of artificial intelligence in the delegation of personal strategic decisions," *Managerial Decis. Econ.*, vol. 40, no. 3, pp. 223–231, Apr. 2019.

[8] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, 2nd ed., Cambridge, U.K.: Cambridge Univ. Press, 2017.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Cambridge, MA, USA: MIT Press, 2018.

[10] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed., Hoboken, NJ, USA: Wiley, 2009.
[11] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Inf. Softw. Technol.*, vol. 55, no. 12, pp. 2049–2075, 2013.
[12] I. Okpala, A. Golgoon, and A. R. Kannan, "Agentic AI systems applied to tasks in financial services: Modeling and model risk management crews," 2025, *arXiv:2502.05439*.
[13] T. Kampik, A. Malhi, and K. Främling, "Agent-based business process orchestration for IoT," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Oct. 2019, pp. 393–397.
[14] I. D. S. Portugal, P. Alencar, and D. Cowan, "An agentic AI-based multi-agent framework for recommender systems," in *Proc. IEEE Int. Conf. Big Data (BigData)*, Dec. 2024, pp. 5375–5382.
[15] N. Yildirim et al., "Multimodal healthcare AI: Identifying and designing clinically relevant vision-language applications for radiology," in *Proc. CHI Conf. Human Factors Comput. Syst.*, May 2024, pp. 1–22.
[16] P. Las-Casas, A. G. Kumbhare, R. Fonseca, and S. Agarwal, "LLexus: An AI agent system for incident management," *ACM SIGOPS Operating Syst. Rev.*, vol. 58, no. 1, pp. 23–36, 2024.
[17] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic AI: Autonomous intelligence for complex goals—A comprehensive survey," *IEEE Access*, vol. 13, pp. 18912–18936, 2025.
[18] J. Chen and L. Zhao, "AI-driven innovation in enterprise architecture: A multi-agent system approach to adaptive design," in *Proc. 8th Int. Conf. Electron. Inf. Technol. Comput. Eng.*, 2025, pp. 768–774.
[19] SAP SE. (Apr. 1, 2025). *Product Roadmap*. [Online]. Available: https://roadmaps.sap.com/
[20] SAP SE, Discovery Center. (2025). *SAP Business AI Features*. [Online]. Available: https://discovery-center.cloud.sap/ai-catalog/
[21] M. Broy and M. Kuhrmann, *Einführung in Die Softwaretechnik*. Berlin, Germany: Springer, 2021.
[22] J. Dick, E. Hul, and K. Jackson, *Requirements Engineering*. Cham, Switzerland: Springer, 2017.
[23] J. Dooley and V. Kazakova, *Software Development, Design, and Coding*. Cham, Switzerland: Springer, 2024.
[24] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.
[25] S. Gregor, "The nature of theory in information systems," *MIS Quart.*, vol. 30, no. 3, pp. 611–642, 2006.
[26] SAP SE, Product Documentation. *Intelligent Scenario Lifecycle Management*. Accessed: Apr. 1, 2025. [Online]. Available: https://help.sap.com
[27] N. S. Yanofsky, *Theoretical Computer Science for the Working Category Theorist*. Cambridge, U.K.: Cambridge Univ. Press, 2022.
[28] W. Liu, W. Chang, C. Shi, Y. Wang, R. Hu, C. Zhang, and H. Ouyang, "Research on intelligent agent technology and applications based on large models," in *Proc. IEEE 4th Int. Conf. Inf. Technol., Big Data Artif. Intell. (ICIBA)*, Dec. 2024, pp. 466–472.
[29] J. White, "Building living software systems with generative & agentic AI," 2024, *arXiv:2408.01768*.
[30] M. Bratman, *Intention, Plans, and Practical Reason*. Cambridge, MA, USA: Harvard Univ. Press, 1987.
[31] M. Georgeff and A. Rao, "Modeling rational agents within a BDI-architecture," in *Proc. 2nd Int. Conf. Princ. Knowl. Represent. Reasoning*, 1991, pp. 473–484.
[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[33] V. Dignum, *Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way*. Cham, Switzerland: Springer, 2019.
[34] S. Sarferaz, *Compendium on Enterprise Resource Planning*. Cham, Switzerland: Springer, 2022.

SIAR SARFERAZ (Member, IEEE) received the Ph.D. degree in computer science. He had studied computer science and philosophy. He began his career as a Method Researcher at Siemens, before moving to SAP SE, where he has now worked for more than 20 years. He is currently a Chief Software Architect with the Research and Development Department, Enterprise Resource Planning (ERP) Solution, SAP SE, Walldorf, Germany. In this role, he drives the digital transformation by defining the solution architecture for the product and by providing concepts scaling for mission critical business processes. He is the Lead Architect of artificial intelligence implementations for SAP SE's ERP product and is responsible for all conceptions of how to add intelligence to business processes. In the context of ERP software, he owns more than 30 patents and has published numerous books.