Architecting Agentic Communities using Design Patterns

A Framework Grounded in ODP Enterprise Language Formalism

Zoran Milosevic[1,2][0000-0002-1364-7423] and Fethi Rabhi[1][0000-0001-8934-6259]

[1] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
(z.milosevic,f.rabhi)@unsw.edu.au
[2] Deontik, Australia
zoran@deontik.com

Abstract. The rapid evolution of Large Language Models (LLM) and subsequent Agentic AI technologies requires systematic architectural guidance for building sophisticated, production-grade systems. This paper presents an approach for architecting such systems using design patterns derived from enterprise distributed systems standards, formal methods, and industry practice. We classify these patterns into three tiers: LLM Agents (task-specific automation), Agentic AI (adaptive goal-seekers), and Agentic Communities (organizational frameworks where AI agents and human participants coordinate through formal roles, protocols, and governance structures). We focus on Agentic Communitiesâ€" coordination frameworks encompassing LLM Agents, Agentic AI entities, and humansâ€"most relevant for enterprise and industrial applications. Drawing on established coordination principles from distributed systems, we ground these patterns in a formal framework that specifies collaboration agreements where AI agents and humans fill roles within governed ecosystems. This approach provides both practical guidance and formal verification capabilities, enabling expression of organizational, legal, and ethical rules through accountability mechanisms that ensure operational and verifiable governance of inter-agent communication, negotiation, and intent modeling. We validate this framework through a clinical trial matching case study. Our goal is to provide actionable guidance to practitioners while maintaining the formal rigor essential for enterprise deployment in dynamic, multi-agent ecosystems.

Keywords: Agentic AI · Design Patterns · Multi-Agent Systems · ODP Enterprise Language · AI Governance · Enterprise Architecture · LLM Agents · Autonomous Systems · Human-AI Collaboration

1 Introduction

The emergence of Large Language Models (LLMs) has significantly transformed software development, enabling creation of *LLM Agents*â€"task-specific entities leveraging LLMs for narrow automation tasks. Subsequently, advances in reasoning capabilities, tool integration, and prompting techniques led to *Agentic AI*â€"software entities with genuine agency combining autonomy with goal-directed reasoning, able to perceive environments, formulate plans, and adapt strategies dynamically [33]. Building upon these capabilities gives rise to, what we refer to as, *Agentic Communities*â€" collaboration frameworks where multiple LLM Agents, Agentic AI entities, and human actors collaborate through structured protocols to achieve common objectives beyond individual agent capabilities. Unlike traditional software agents with pre-deterministic behaviors, or LLM Agents executing assigned tasks without independent goal formulation, Agentic AI represents a fundamental departure through autonomous goal pursuit, while Agentic Communities enable coordinated multi-participant systems with explicit and operational governance architecture essential for enterprise and industrial deployment.

However, this shift from deterministic to agentic behavior introduces significant architectural challenges. Practitioners face questions about when to employ single autonomous agents versus coordinated multi-agent systems, how to ensure accountability in multi-agent autonomous decision-making, how to integrate human oversight and expertise, and how to compose patterns effectively while maintaining governance and compliance. These challenges require systematic architectural frameworks grounded in well-proven and rigorous methods. The practical need for such frameworks is evidenced by emerging industrial implementations in asset-intensive industries. For example, production deployments of Multi-Agent Generative Systems incorporating formal governance and accountability mechanisms have demonstrated measurable operational improvements in mining, oil & gas, and manufacturing sectors [27], validating the practical viability of formal accountability in safety-critical autonomous systems.

While design patterns have proven invaluable in traditional software engineering [9] and distributed systems [12,22], their application to agentic AI systems is limited to simple ones such as ReAct [39], Tool Use [28], Planning [13], and Reflection [29]. Current pattern catalogues either focus narrowly on specific LLM prompting techniques or lack the formal rigor necessary for enterprise-grade systems with stringent accountability requirements. Moreover, they rarely address the reality that enterprise systems must coordinate both AI agents and human participants within governed frameworks.

To address this gap, this paper provides a systematic approach to classifying different types of LLM-powered entitiesâ€"LLM Agents, Agentic AI, and Agentic Communitiesâ€"and identifying patterns that are relevant for each of these types. The paper uses ISO Open Distributed Processing (ODP) Enterprise Language (EL) community formalism [1] as a foundation for describing coordination challenges in Agentic Communities, including the relationship between intent and obligations, critical for reasoning about accountability when autonomous agents and humans collaborate within governed frameworks.

The remainder of this paper is organized as follows. Section 2 provides background about LLM powered agents and Agentic AI, and introduces ODP EL community formalism. Section 3 presents our comprehensive pattern catalogue with categories, classification methodology, and key insights. Section 4 presents a design pattern methodology adapted to agentic AI systems. Section 5 validates the framework through a clinical trial matching case study. Section 6 discusses the formal aspects of our architecture approach enabling rigorous operational and governance verification and implementations. Section 7 discusses related work, and Section 8 provides concluding remarks and outlines future research directions.

2 Background

2.1 LLM Agent, Agentic AI and Agentic Community

We introduce a three-tier classification framework that builds upon and refines the conceptual taxonomy proposed by Sapkota et al. [25]. While their work distinguishes AI Agents (task-specific automation) from Agentic AI (systems with multi-agent collaboration and coordinated autonomy), we decompose their "Agentic AI" category into two distinct levels to address different architectural requirements: individual agents with genuine agency, and multi-participant coordination frameworks, including both Agentic AI and human participants. This refinement enables precise pattern selection based on whether systems require single autonomous agents or coordinated communities with formal governance.

Key Terminology—Autonomy vs. Agency: Before introducing our classification, we clarify two foundational concepts. *Autonomy* denotes the ability to operate independently without constant external supervision—making decisions and taking actions without requiring continuous human control. *Agency*, however, represents a richer concept encompassing autonomy plus intentionality, goal-directedness, and adaptive behavior [36]. An autonomous system operates independently; an agentic entity independently pursues goals through contextual reasoning, strategic planning, and adaptive action-taking. This distinction underpins our classification: LLM Agents exhibit task-scoped autonomy (independent operation within defined parameters) while Agentic AI exhibits genuine agency (autonomous goal pursuit with adaptive reasoning).

LLM Agent[3]: LLM-powered software entities executing specific tasks autonomously within controlled environments. While capable of independent operation, LLM Agents lack agency—they execute assigned tasks without independent goal formulation or adaptive strategy adjustment. LLM Agents function as building blocks and utility components within larger systems. Examples include data validators, extractors, format converters, and filtering agents. An LLM Agent capability can be considered a simplified abstraction of an Agentic AI capability, focusing on task execution without the cognitive sophistication required for genuine agency.

[3] Sapkota et al. [25] use the term "AI Agents" for this category; we use "LLM Agent" throughout to emphasize the LLM foundation that distinguishes these from traditional rule-based agents.

Table 1. Three-Tier Classification: Key Characteristics

Characteristic	LLM Agent	Agentic AI	Agentic Community
Autonomy Level	Limited	High	Distributed
Decision-Making	Prompt-driven	Context-aware	Collective
Scope	Specific tasks	Complex tasks	Multi-participant
Learning	Static	Continuous	Community-level
Goal Pursuit	Executes tasks	Formulates goals	Collaborative
Participants	Single component	Single agent	Multiple (AI + human)
Formal Model	N/A	N/A	ODP-EL Community

Agentic AI: LLM-powered and other entities possessing genuine agency—the capability for autonomous goal pursuit, contextual reasoning, strategic planning, and adaptive behavior. Unlike LLM Agents, Agentic AI independently determines both what goals to pursue and how to achieve them, adapting strategies based on environmental context and past experience. Agentic AI represents the cognitive foundation enabling genuine autonomous behavior. Examples include ReAct [39] agents with reasoning traces, hierarchical planning agents [13], reflexive learners that improve through self-critique [29], and metacognitive systems [33].

Agentic Community: Coordination frameworks where multiple participants—including LLM Agents, Agentic AI software entities, and human actors—collaborate through structured protocols and shared infrastructure. Agentic Communities exhibit emergent behaviors beyond individual participants through multi-agent collaboration, dynamic task decomposition, and persistent shared memory. In such a framework, human actors function as coordinated participants, supervisors, or peer collaborators rather than external observers. Examples include multi-agent workflows, orchestration platforms, negotiation frameworks, and human-AI collaborative systems. This category encompasses and extends aspects of Sapkota et al.'s "Agentic AI" by adding explicit human integration and, importantly, formal accountability governance specification leveraging ODP-EL community formalism (Section 2.2).

Note that Agentic Communities may not always involve human users directly, but there must be traceability from community actions to the parties ultimately responsible for their effects—ensuring accountability in autonomous systems. Further, communities can model organizational aspects of a single organization (roles, capabilities, rules) or capture cross-organizational contracts and workflows, for which federations (a specialized community type) can be used. Section 2.2 details the ODP-EL formal foundation for Agentic Communities.

These three types are illustrated in Figure 1, noting increasing levels of autonomy, and emergent behavior, with Agentic Communities coordinating heterogeneous participants including both artificial and human agents.

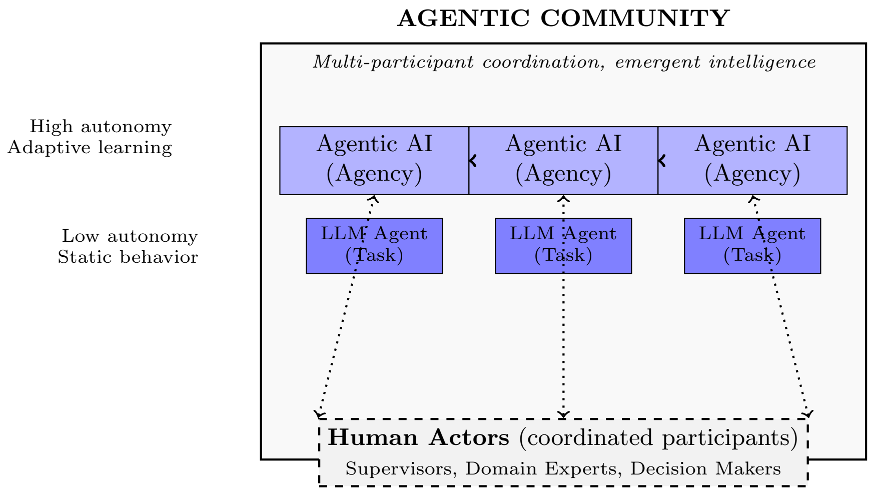

Fig. 1. Three types of LLM-powered entities and their relationships. Agentic Communities coordinate heterogeneous participants including LLM Agents, Agentic AI systems, and human actors through structured protocols, shared infrastructure and agreed and computable governance specification.

2.2 Formal Foundation: ODP-EL Communities

As introduced in Section 2.1, Agentic Communities coordinate heterogeneous participantsâ€"LLM Agents, Agentic AI entities, and human actorsâ€"through structured protocols within governed frameworks. To enable rigorous specification of such multi-participant coordination with verifiable governance properties, we ground Agentic Communities structure, behaviour and governance rules, in the ISO Open Distributed Processing Enterprise Language (ODP-EL) standard (ISO/IEC 15414) [1,17]â€"an internationally recognized framework for specifying enterprise distributed systems with rigorous, formal based semantics.

This formal grounding enables our pattern-based architectural approach by providing mechanisms for expressing design patterns as ODP-EL community templatesâ€"specifications that combine practical architectural guidance with formal semantics. Through this integration, patterns instantiate as community specifications comprising roles (behavioral placeholders fillable by LLM Agents, Agentic AI, or human actors), normative constraints (obligations, permissions, prohibitions), and contracts (normative relationships binding roles together). This dual natureâ€"patterns remaining accessible to practitioners while supporting formal verificationâ€"addresses the critical gap between proven architectural solutions and the verifiable accountability, traceable authority, and provable compliance required for enterprise deployment in regulated industries.

The ODP-EL framework, proven in complex enterprise distributed systems theory and practice, provides essential constructs that transform our design patterns into precise architecture specifications, including verifiable properties: