



DHBW

Duale Hochschule
Baden-Württemberg
Ravensburg

Multi-Agent Architecture Design

Workflow Automation in Compliance
with Operational Excellence

Bachelor's Thesis

Wirtschaftsinformatik—Business Engineering
Duale Hochschule Baden-Württemberg
Ravensburg

Francisco Rodriguez Müller, on September 17, 2025
Mat. Nr.: 2775857, Course: RV-WWIBE122
Supervisor: Prof. Dr. Paul Kirchberg

Declaration of Authenticity

I hereby declare that I have written the present bachelor's thesis independently and that I have not used any sources or aids other than those indicated. All passages that were taken from published or unpublished works are clearly marked as such. This thesis has not been submitted to any other examination authority in the same or a similar form.

Thesis Title:

Multi-Agent Architecture Design

Workflow Automation in Compliance with Operational Excellence

Ravensburg, September 17, 2025

(Place and Date)

(Signature)

Acknowledgments

I would like to thank the *All for One Group SE* for providing the opportunity and resources to write this thesis. I also extend my gratitude to my academic supervisor, *Prof. Dr. Paul Kirchberg* for his guidance throughout the process; my professional advisor, *Emily Celen* for helping me with the scope and realistic expectations for this work; and my direct supervisor and friend *Ralph Thomaßen* for his wise advice and relentless support throughout my academic journey.

Abstract

[Placeholder, still a work in progress] The literature suggests three design-relevant insights: (1) OpEx increasingly hinges on adaptability, decision quality, and compliance—not only efficiency; (2) automation is moving from deterministic scripts (RPA) to adaptive, tool-using agents; and (3) MAS provide a natural organizing structure for such agents in complex enterprises but necessitate robust governance, monitoring, and integration. These insights inform the requirements (e.g., traceability, policy constraints, observability) and the architecture choices (agent roles, interaction patterns, orchestration) developed later in this thesis.

Contents

Acknowledgments	ii
Abstract	iii
Abbreviations	v
1 Introduction	1
2 Methodology	2
2.1 Qualitative Content Analysis	3
2.2 Requirements Engineering	4
2.3 Information Systems Design	4
3 Literature Review	5
3.1 Operational Excellence	6
3.2 Workflow Automation	8
3.3 Agentic Artificial Intelligence	12
4 Requirements Modeling	17
4.1 Clustering and Consolidation	17
4.2 Reformulation and Classification	19
4.3 Model-Based Representation	20
5 Architecture Modeling	20
5.1 [placeholder]Modeling the Agents	20
5.2 [placeholder]Modeling the Architecture	20
5.3 [placeholder]Modeling the Interactions	20
6 Discussion: Applicability Criteria	20
7 Conclusion	20
References	21
Appendix A: Full SysML Models	22

Abbreviations

DSR	Design Science Research
GaaS	Governance as a Service
GenAI	Generative Artificial Intelligence
IPA	Intelligent Process Automation
ISD	Information Science Design
LLM	Large Language Model
MAS	Multi-Agent System
MBSE	Model-Based Systems Engineering
OpEx	Operational Excellence
QCA	Qualitative Content Analysis
RE	Requirements Engineering
ReAct	Reason & Action
RPA	Robotic Process Automation
SLA	Service Level Agreement
SysML	Systems Modeling Language
WfMs	Workflow Management Systems

1 Introduction

Organizations across industries continue to face persistent challenges in achieving operational excellence (OpEx). Fragmented processes, manual interventions, and inconsistent data quality undermine efficiency and decision-making. Legacy workflows and siloed systems exacerbate these inefficiencies, while traditional automation approaches often lack the adaptability needed in dynamic business environments. For companies, this translates into slower response times, higher compliance risks, and limited scalability—issues that directly threaten competitiveness.

Agentic AI, building on the advances of generative artificial intelligence (GenAI), opens new possibilities to extend automation beyond deterministic scripts. While GenAI provides the cognitive and generative capabilities, agentic AI leverages these to create adaptive, tool-using agents that can plan, act, and coordinate—thereby supporting governance, decision quality, and organizational agility. Despite this potential, both practice and academic literature lack structured strategies and conceptual frameworks for embedding such agentic capabilities into operational workflows in a scalable and value-driven way. This gap motivates the present research.

In this context, multi-agent systems (MAS) can serve as a reference architecture for integrating GenAI-enabled agentic AI into enterprise workflow automation. The central research question is:

How can a MAS architecture be designed to integrate GenAI capabilities into workflow automation, in order to enhance agility, compliance, and decision quality to achieve OpEx?

To answer this question, the study addresses the following sub-questions:

- *Which design requirements are necessary to align a multi-agent architecture with the goals of OpEx?*
- *How should a MAS be architected to fulfill these requirements?*
- *Under which conditions is deploying a generative multi-agent architecture justified over traditional automation approaches?*

Methodologically, the thesis applies Design Science Research (DSR) to develop a conceptual reference architecture. The approach synthesizes requirements from academic literature and OpEx principles, models agent roles and interactions, and derives applicability conditions for real-world deployment. Although the architecture is designed to remain industry-agnostic, a use case from the financial services

sector is introduced to illustrate how the conceptual model can be instantiated in a regulated, legacy-intensive environment.

The core contribution of this work is a conceptual design of a MAS that leverages GenAI to support OpEx in enterprise workflows. Specifically, it delivers:

- *A structured synthesis of system requirements derived from academic literature and OpEx principles.*
- *A conceptual architecture detailing agent roles, interactions, and integration points.*
- *A set of applicability conditions and design considerations to guide future deployment and evaluation of generative multi-agent architectures in practice.*

The scope is limited to conceptual design; formal evaluation and technical implementation are proposed as future work. Although the architecture is designed to remain industry-agnostic, a use case from the financial services sector is introduced to illustrate how the conceptual model can be instantiated in a regulated, legacy-intensive environment.

After this introduction in Section 1, the thesis is structured as follows: Section 2 outlines the research methodology, including the use of Design Science Research (DSR) and supporting methods. Section 3 presents a literature review on operational excellence, workflow automation, and agentic AI. Section 4 details the synthesis and modeling of requirements. Section 5 develops the conceptual multi-agent architecture. Section 6 discusses the applicability of MAS in workflow automation use cases, and Section 7 concludes with reflections and directions for future research.

2 Methodology

This thesis applies DSR methodology to create a conceptual artifact—a multi-agent architecture for workflow automation. Practically, the approach unfolded in three steps: (1) *reviewing the literature* on OpEx, workflow automation, and agentic AI; (2) *deriving and structuring requirements* from literature and case material into a requirements model; and (3) *designing a conceptual system architecture* using Systems Modeling Language (SysML).

Supporting methods included Mayring-style qualitative content analysis (QCA) for the review, requirements engineering (RE) and systems analysis for the requirements model, and information systems design (ISD) to structure the architecture and

ensure requirement-to-design traceability, supported by SysML modeling practices from Model-Based Systems Engineering (MBSE). Within DSR, the work focuses on problem identification, objective definition, and conceptual design, while instantiation/demonstration and formal evaluation are out of scope given the bachelor-thesis format and resource constraints. This scoping maintains methodological rigor while keeping the contribution focused: a well-argued reference architecture ready for subsequent implementation and empirical evaluation.

2.1 Qualitative Content Analysis

To ensure a systematic and structured literature review, this thesis employed QCA following the principles of Mayring and Fenzl (2022). As a rule-based method for synthesizing insights from textual sources, QCA was used within the DSR framework to support the problem identification and objective definition phases (Hevner et al. 2004; Peffers et al. 2007). In this thesis, it was applied in a literature-focused manner to structure the review and provide a traceable basis for subsequent RE.

The analysis was scoped along three dimensions. The *analysis unit* was defined as the overall body of literature addressing operational excellence, workflow automation, and agentic AI. The *context unit* consisted of individual publications (books, peer-reviewed articles, industry reports, standards). The *coding unit* was defined as discrete statements or conceptual claims relevant to the intersection of OpEx, automation paradigms, and AI-based multi-agent systems.

A mixed deductive-inductive approach was used. Deductive categories were derived from established theory, including OpEx dimensions such as adaptability, compliance, and decision quality, as well as prior automation frameworks (e.g., Robotic Process Automation, RPA; Intelligent Process Automation, IPA). Inductive categories emerged from the material itself, capturing issues highlighted repeatedly in the sources, such as observability, traceability, and governance in agentic AI. This balance ensured that both established and novel concerns were systematically reflected.

The outcome of this categorization was not a formal codebook, but a set of thematic clusters that guided the narrative structure of Section 3. Each subsection of the literature review is organized around these categories, which in turn serve as the input for the elicitation lists presented at the beginning of Section 4. In this way, QCA provides both a conceptual ordering of the literature and a direct bridge

into the requirements engineering process.

2.2 Requirements Engineering

Following the IEEE (1990) definition, a requirement is a *condition or capability needed by a user to solve a problem or achieve an objective*. RE provides the systematic means to derive such objectives. In this thesis, RE was applied in the early phases to ensure that the conceptual architecture rests on precise, validated needs rather than general aspirations.

The synthesis of requirements followed a structured but literature-driven process. Recurring design concerns were identified from the results of the QCA, documented in *elicitation lists*, and then consolidated into a unified set of requirement candidates. Consistent with Glinz et al. (2020), each candidate was reformulated into an atomic, unambiguous, and verifiable “shall” statement and classified into *functional requirements* (system behaviors), *quality requirements* (non-functional attributes such as performance or compliance), or *constraints* (technological or regulatory limits).

While this procedure draws on the phases described by Herrmann (2022) (elicitation, documentation, analysis, management), it was adapted to the scope of this thesis: instead of stakeholder workshops, the primary elicitation source was the systematically coded literature. To situate the requirements, a complementary systems analysis defined the system boundary, identified stakeholders and external actors, and clarified interface obligations—helping prevent scope creep and omissions.

Requirements were then represented in SysML requirement diagrams. Trace links connect each documented requirement to the respective architecture elements, enabling full requirement-to-design traceability via *«satisfy»* and *«verify»* relationships. This model-based approach ensures that design decisions can always be traced back to validated needs and that no requirement was overlooked.

In summary, the integration of RE and systems analysis provided a structured, traceable, and quality-assured requirement set. This foundation anchors the subsequent conceptual architecture in rigorously defined objectives, ensuring consistency with both DSR methodology and operational excellence goals.

2.3 Information Systems Design

In line with DSR, this thesis applies principles of information systems design to structure the artifact. The architecture was modeled in SysML, making use of

MBSE practices to ensure requirement-to-design traceability. While originating in MBSE originates in systems engineering, its modeling discipline is transferable to information systems contexts and supports the systematic development of conceptual architectures.

A conceptual architecture was systematically developed based on the previously synthesized requirements using a MBSE approach. MBSE provides a formalized way to transform requirements into a rigorous system model and to validate the design at the conceptual level. In fact, MBSE is defined as the formal application of modeling to support system requirements, design, and analysis, along with verification and validation, beginning in the conceptual design phase. Adopting MBSE thus ensured that even at this early stage, the architecture could be checked against stakeholder needs and constraints. The methodology enabled a unified representation of the entire system that made it possible to study interactions among components and agents before implementation. Each requirement was traced to corresponding elements in the model (e.g. agent roles, interactions, or policies), guaranteeing requirements-to-design traceability and consistency throughout the design process.

The conceptual architecture was captured as a SysML v2 model (the latest OMG Systems Modeling Language standard for systems modeling) leveraging a plain-text modeling workflow in Eclipse Systems Mode. This setup allowed the use of SysMLv2’s textual notation to define the system’s structure and behavior in a tool-agnostic, version-controlled manner. The choice of SysMLv2 (over SysMLv1 or informal diagrams) is justified by its improved expressiveness and alignment with current MBSE best practices; as an OMG-developed language it provides robust semantics for specifying complex interactions in MAS while remaining an emerging industry standard. In summary, the information system’s design was conducted in a model-driven fashion: the SysML v2 conceptual model served as an executable blueprint of the architecture, facilitating early validation of design decisions against the requirements and providing a solid foundation for subsequent development steps (cf. Madni 2023; OMG 2023).

3 Literature Review

The literature review is organized into three subsections that together frame the problem context of this thesis. It begins with operational excellence, which defines the strategic objectives—adaptability, compliance, decision quality—that guide

enterprise transformation efforts. The second subsection addresses workflow automation, as the established technological approach for operationalizing these objectives in practice. The third subsection turns to agentic AI, a rapidly emerging paradigm that extends automation beyond deterministic scripts toward adaptive, tool-using agents. This sequence—objectives, established solutions, emerging solutions—provides a logical progression from strategic goals to current practice and then to prospective innovations. It ensures that the requirements synthesized in Section 4 are grounded in both enduring management principles and the latest technological developments relevant to workflow design.

3.1 Operational Excellence

OpEx originated as a management philosophy in the manufacturing sector, particularly in the automotive industry to optimize quality and efficiency. In this classical context, OpEx focused on minimizing defects, eliminating waste, and embedding continuous improvement practices into organizational routines (Juran and Godfrey 1999; Womack and Jones 2013). While these roots remain important, they provide only a partial foundation for understanding OpEx in today’s IT-driven enterprises, which operate in volatile environments shaped by rapid technological change, regulatory complexity, and global competition.

One central dimension of OpEx is ADAPTABILITY AND AGILITY. In IT-driven firms, where OpEx is defined less by physical production flows and more by the ability to execute strategies effectively while maintaining innovation, adaptability refers to the capacity of processes to be reconfigured in response to volatility, ensuring resilience in dynamic environments. Agility emphasizes change-readiness and rapid adaptation, qualities increasingly recognized as indicators of organizational excellence in globalized and unpredictable markets. While adaptability enhances resilience, it can reduce efficiency if frequent changes disrupt standardization; conversely, a strong focus on efficiency can make processes rigid and less responsive to unexpected events. In practice, organizations must reconcile continuous improvement with agility, balancing stability and flexibility in their operational routines (Carvalho et al. 2023).

Another recurring theme in the literature is COMPLIANCE AND RISK MANAGEMENT. OpEx in regulated industries demands that workflows embed mechanisms for ensuring transparency, auditability, and regulatory adherence (Owoade et al.

2024). Compliance safeguards minimize operational risk but can introduce bureaucratic overhead and slow decision-making. The key challenge is balancing strict rule enforcement with the flexibility needed to respond to novel business conditions, a tension especially visible in digital service environments with evolving legal frameworks (Juran and Godfrey 1999).

A further category is **DECISION QUALITY**. A core aim of OpEx is not only to accelerate decision-making but to improve its reliability and evidential grounding (Owoade et al. 2024). Automation can help by aggregating relevant data and reducing errors, but it also risks opacity and overconfidence when human oversight is limited. The central trade-off is speed versus quality: excessive automation may produce faster but less accountable outcomes, while excessive oversight slows operations. For sustainable excellence, systems must therefore support evidence-based choices and make decision pathways transparent.

Another key dimension is **EFFICIENCY AND CONTINUOUS IMPROVEMENT**. Rooted in Lean and TQM traditions, efficiency emphasizes minimizing waste and reducing manual effort, while continuous improvement institutionalizes iterative refinements in processes (Juran and Godfrey 1999; Womack and Jones 2013). Together, these principles enhance operational reliability and cost-effectiveness, yet they can conflict with the need for flexibility in volatile environments. The challenge is to design workflows that are optimized for today while remaining adaptable for tomorrow.

Customer-facing outcomes are captured by **CUSTOMER-CENTRICITY**. OpEx emphasizes that processes must be aligned with user needs and service-level commitments (Womack and Jones 2013; Juran and Godfrey 1999), ensuring reliability, responsiveness, and satisfaction. A strong customer focus can create pressure to customize and accelerate processes, which may undermine efficiency or compliance. The literature stresses that sustainable excellence requires balancing external demands with internal consistency, often formalized through service-level agreements (SLAs).

The category of **USER EMPOWERMENT AND CULTURE** recognizes that operational excellence is not solely technical but also organizational. Effective improvement requires systems that support transparency, collaboration, and employee engagement (Womack and Jones 2013). Empowerment fosters ownership and participation, but decentralizing authority can introduce inconsistency and conflict with standardization goals. Culture therefore functions as both an enabler and a constraint for process excellence, shaping how automation is accepted and leveraged by

human actors (Juran and Godfrey 1999).

Finally, TECHNOLOGY INTEGRATION AND SCALABILITY reflects the increasing role of digital platforms in enabling OpEx. Modern enterprises rely on architectures that integrate automation, AI, and cloud services to achieve scalable and resilient operations (Owoade et al. 2024). Integration enables end-to-end process coverage and agility, but also increases dependency on heterogeneous systems and external vendors. Scalability promises growth and innovation, yet without careful governance it can amplify complexity and risk.

3.2 Workflow Automation

Building on the need to balance stability and agility in operational processes, workflow automation emerged as a means to systematically coordinate and streamline business workflows. It refers to the use of software systems (workflow management systems, WfMS) to orchestrate tasks, information flows, and decisions along a predefined business process model. According to industry standards, workflow automation entails routing documents, information, or tasks between participants according to procedural rules, with the goal of reducing manual effort and variability in execution (Basu and Kumar 2002). Early WfMS in the late 20th century were designed to make work more efficient, to integrate heterogeneous applications, and to support end-to-end processes even across organizational boundaries (Stohr and Zhao 2001).

One foundational aspect of workflow automation is PROCESS ORCHESTRATION. By encoding business procedures into formal process models that are executed by a central *workflow engine*, organizations could enforce consistent process flows, improve speed and accuracy, and embed compliance checks into routine operations. In essence, pre-AI workflow automation provided a structured, deterministic way to implement business processes in software, directly addressing chronic issues like fragmented manual tasks and data silos in pursuit of OpEx (Basu and Kumar 2002). Orchestration denotes the centralized coordination of tasks and activities according to a defined process logic. In a typical WfMS, a workflow engine enacts the process model, dispatching tasks to the right resources (human or machine) in the correct sequence and enforcing the business rules at each step (Basu and Kumar 2002). This engine-driven coordination brings predictability and repeatability to workflows: tasks are executed in a fixed, optimized order with minimal ad-hoc variation. By

systematically controlling task flow, early workflow systems could eliminate many manual hand-offs and delays, thereby boosting efficiency and consistency in outcomes (Stohr and Zhao 2001). The orchestration approach essentially translated managerial routines into software: for example, an order processing workflow would automatically route an order through credit check, inventory allocation, shipping, and billing steps without needing human coordination at each transition. Such deterministic sequencing was crucial for achieving the quality and reliability targets of OpEx in an era before adaptive AI capabilities.

A closely related design concern is INTEGRATION. Workflow automation inherently requires linking together diverse people, departments, and IT systems into an end-to-end process. Literature emphasizes that WfMS must integrate heterogeneous application systems and data sources to allow seamless information flow across functions (Stohr and Zhao 2001). For instance, a procurement workflow might connect an ERP inventory module, a supplier's database, and a financial system so that each step can automatically consume and produce the necessary data. This integration extends beyond technical connectivity; it also encompasses coordinating work across organizational boundaries. As e-business initiatives grew in the 1990s and early 2000s, workflows increasingly spanned multiple organizations (suppliers, partners, customers), demanding inter-organizational process integration (Basu and Kumar 2002). Research in this period identified the need for distributed workflow architectures that could bridge independent systems and companies. Georgakopoulos et al. (1995) noted that existing workflow tools had limitations in complex environments, calling for infrastructure to handle heterogeneous, autonomous, and distributed information systems. In practice, this led to the development of interoperability standards (e.g., XML-based process definitions, web service interfaces) and process choreography protocols to ensure that a workflow could progress smoothly even when multiple organizations or platforms were involved. Effective integration was thus an essential condition for workflow automation, enabling the end-to-end automation of processes that formerly stopped at organizational or system boundaries.

To manage complexity and change, MODULARITY in workflow design became another important principle. Rather than hard-coding monolithic process flows, architects sought to break workflows into modular components or sub-processes that could be reused and reconfigured as needed. This component-based approach was accelerated by the rise of service-oriented architectures and e-business “workflow

of services” concepts. For example, composite e-services and e-hubs allow organizations to construct complex workflows by composing smaller service modules. A modular workflow architecture improves maintainability: if a business rule changes or a new subprocess is required, one can update or insert a module without re-designing the entire workflow from scratch. Modularity also underpins adaptability. Ideally, a workflow systematizes routine functions but can be adjusted to accommodate new requirements or variations in the process (Basu and Kumar 2002). In other words, the literature suggests that well-designed workflow automation should combine standardization with flexibility: processes are structured into clear modules for the “happy path” of routine operations, yet those modules can be reorchestrated or overridden in exceptional cases. This design philosophy reflects an early recognition that no single process model can anticipate all future conditions, so a degree of configurability must be built in.

Despite efforts to introduce flexibility, traditional workflow automation faced notable challenges with EXCEPTION HANDLING. Exception handling refers to the ability of a system to cope with deviations, errors, or unforeseen scenarios that fall outside the predefined process flow. Basu and Kumar (2002) candidly observe that existing WfMS “tend to fall short whenever workflows have to accommodate exceptions to normal conditions”—i.e., when something unexpected occurs that was not explicitly modeled, the system often cannot resolve it autonomously, forcing human intervention. Typically, designers might anticipate a limited number of exception scenarios and build alternate paths for those (e.g., an approval escalation if a manager is absent). However, if a novel exception arises (say, a new regulatory requirement or an unplanned system outage affecting a step), the rigid workflow cannot handle it, and manual workarounds are needed. This brittleness of early workflows under dynamic conditions was widely acknowledged. Research proposed various approaches to improve exception handling, such as more advanced process metamodels and integration of AI or rule-based decision support to catch and respond to anomalies (Basu and Kumar 2002). Yet, in the pre-AI era, most workflow automation remained predominantly rule-driven and inflexible outside of predefined contingencies. Exception handling thus stood out as a critical limitation of classical automation approaches, highlighting a gap between the desire for end-to-end automation and the reality of complex, ever-changing business environments.

Another salient theme in the literature is WORKFLOW GOVERNANCE—the structures and mechanisms for overseeing automated workflows and aligning them with

organizational policies. As companies entrusted core business processes to software, ensuring the correct and intended execution of those processes became vital. Key governance considerations include monitoring, auditing, and controlling workflows. A WfMS typically provides monitoring dashboards and logs so that managers can track the state and performance of process instances (e.g., to identify bottlenecks or errors). It also enforces role-based access control, ensuring that only authorized personnel perform certain tasks or approvals, which is essential for compliance in regulated industries. Basu and Kumar (2002) highlight the importance of organizational “metamodels” and control mechanisms that tie workflows to an enterprise’s structure—for example, defining which organizational roles are responsible for each task and how escalation or overrides should happen under specific conditions. Additionally, governance extends to establishing standards and best practices for workflow design and deployment. Industry coalitions and standards bodies (such as the WFMC in the 1990s) issued reference models and interface standards to promote consistency and interoperability in workflow implementations. In the context of inter-organizational workflows, governance also means agreeing on protocols and service-level commitments between partners so that automated interactions remain trustworthy and transparent. Overall, robust governance in workflow automation ensures not only efficiency but also accountability, security, and compliance. It addresses the managerial and oversight challenges that arise once processes are no longer directly handled by individuals but by software agents following prescribed logic. From the structured analysis of these sources, pre-AI workflow automation can be summarized by a set of key architectural categories. These represent the dominant design objectives and constraints that had to be addressed in classical workflow systems, and they mirror the strengths as well as the limitations of those systems. The pre-AI workflow automation literature established a foundation of structured, rule-driven process management focused on efficiency, integration, and control. The categories above encapsulate both the core capabilities that made traditional workflow systems valuable and the pain points (like inflexibility in the face of change) that constrained their applicability. These insights provide a structured basis for the requirements derivation in Section 4 and guide the architectural considerations in later chapters. By grounding the design in these well-understood aspects of workflow automation, the thesis ensures that the proposed multi-agent architecture builds on proven practices while also targeting the gaps. Indeed, many of the limitations noted here—especially around exception handling and adaptive-

ness—motivate the incorporation of agentic AI elements in the next section, which explores how intelligent agents can augment and transform workflow automation to better achieve OpEx.

3.3 Agentic Artificial Intelligence

Agentic AI refers to systems composed of multiple interacting generative agents that autonomously collaborate to achieve complex goals. It represents a shift beyond single AI agents toward orchestrated multi-agent ecosystems, enabled largely by recent advances in the field. Whereas traditional automation (e.g. rule-based RPA) executes predefined steps, an agentic architecture features adaptive, goal-directed agents that can perceive context, make decisions, and act with minimal hard-coded instructions. This idea builds on classic MAS principles of autonomy and social action (Castelfranchi 1998; Ferber 1999), but now agents are augmented with learning and reasoning capabilities from large language models (LLMs). Sapkota et al. (2025) highlight this paradigm shift by highlighting the difference between AI agents and agentic AI, framing the latter as “multi-agent collaboration, dynamic task decomposition, persistent memory, and orchestrated autonomy” in pursuit of flexible problem-solving.

A defining trait of agentic AI is a higher degree of AUTONOMY. Agents can operate without constant human or central control, making and executing decisions in real time. Early MAS research already emphasized agent autonomy—e.g. agents as entities with independent control over their actions and state. Modern generative agents greatly amplify this autonomy by leveraging LLM-based reasoning to plan multi-step actions toward goals. For instance, frameworks like AutoGPT (Yang et al. 2023) demonstrated that a single LLM-based agent can iteratively break down objectives, choose actions, and adjust based on feedback without human intervention. This autonomy promises agility and decision quality (agents can respond to situational changes or large search spaces beyond rigid scripts), but it also introduces risks. As Russell et al. (2015) caution, each additional decision delegated to an opaque AI agent shifts “ethical control” away from human operators. In enterprise settings, uncontrolled autonomous decisions might lead to policy violations or unsafe actions (Gaurav et al. 2025). Thus, an architectural challenge is balancing agent freedom with mechanisms to supervise or constrain critical decisions. In practice, this means designing agents with clearly scoped authorities, fail-safes, or escalation

paths (e.g. requiring human confirmation for high-impact actions) to align autonomy with organizational policies.

tool-use capability is another hallmark of agentic AI architectures. Simply put, agents are not limited to their built-in knowledge; they can invoke external tools, APIs, or other services as part of their reasoning loop. This extends an agent’s functionality—for example, an AI agent might call a database, run a code snippet, or query web services to gather real-time information. Research shows that augmenting LLM agents with tool integration significantly improves their problem-solving scope and accuracy. Notably, the ReAct paradigm interleaves an agent’s chain-of-thought with tool calls, allowing it to perceive (via queries) and act (via external operations) iteratively (Yao et al. 2023). Such designs transform static LLMs into dynamic cognitive agents that can perceive, plan, and adapt, a critical capability for complex, multi-step workflows. For workflow automation, this means an agent can not only parse instructions but also execute parts of a workflow (e.g. trigger an RPA bot or send an alert) and then reason over the results. Architecturally, enabling tool use requires adding interface layers for the agent to safely interact with enterprise systems (APIs, databases, RPA scripts), along with policies on allowed tools. It’s worth noting that tool integration adds orchestration complexity and potential error propagation paths (Sapkota et al. 2025). Therefore, designs often include an orchestration layer or planner agent that manages when and how tools are invoked, checks tool outputs, and handles exceptions (e.g. what if a tool fails or returns unexpected data). In summary, tool-use greatly enhances agent capabilities, but it demands careful architectural planning to manage the added complexity and ensure robust tool-agent interaction.

Agentic AI systems are inherently multi-agent—they comprise not one but many agents, often with specialized roles, that must coordinate their efforts. This multi-agent approach stems from the insight that complex workflows can be decomposed: instead of one monolithic AI agent trying to do everything, a team of agents can each handle subtasks and then combine results. Such specialization aligns with principles of OpEx (e.g. division of labor and expertise) and has been shown to improve performance. For example, Shu et al. (2024) report that a collaborative team of LLM-based agents achieved up to 70% higher success rates on complex tasks compared to a single-agent approach. Architecturally, coordination mechanisms are crucial to harness these gains. Agents need to communicate their intentions, share data/results, and synchronize plans. The literature distinguishes coordination

structures along two dimensions: hierarchy vs. flat and centralized vs decentralized decision-making. In a centralized hierarchical design, a top-level planner/manager agent delegates tasks to subordinate agents and integrates their outputs (akin to a project manager overseeing specialists). This can simplify global coordination and ensure alignment with a single source of truth (the planner’s goal), at the cost of a single point of failure or bottleneck. Conversely, decentralized teams use peer-to-peer negotiation or voting; all agents are more equal and collectively decide on task assignments or conflict resolution (drawing on concepts from distributed AI and game theory). For instance, one recent system had developer agents jointly agree on a solution design without a central boss, mimicking consensus decision-making (Qian et al. 2024). Each approach has trade-offs: hierarchical control can be more efficient for well-structured processes, while decentralized collaboration may be more robust to single-agent failure and better for ill-structured problems. Inter-agent communication protocols (what messages agents send and when) are another design facet—simple cases use direct message passing or shared memory, whereas more complex setups might use an event-bus or blackboard architecture for asynchronous communication. Importantly, specialization means each agent can be bounded in scope (e.g. a compliance checker agent vs. a data retrieval agent), which helps with scalability: each agent’s LLM or reasoning module can operate within a focused context window, and different team members can even use different model types suited to their niche. This modularity and specialization, orchestrated through well-defined coordination logic, is a key architectural strength of agentic AI. It mirrors how human organizations structure teams for efficiency, and indeed is crucial for aligning multi-agent AI workflows with complex enterprise processes.

As agent behaviors become more autonomous and distributed, ensuring observability of the system is vital. Observability here means that the internal states, decisions, and actions of agents can be monitored and understood by humans or supervisory systems. Traditional MAS literature often dealt with observability in terms of state visibility (e.g. in partially observable environments), but in an enterprise context it translates to runtime transparency and traceability of what agents are doing and why. One challenge is that LLM-driven agents reason in natural language (or latent vectors), making their decision process somewhat opaque. Sapkota et al. (2025) highlight that AI agents “lack transparency, complicating debugging and trust”, and they advocate for robust logging and auditing pipelines to make agent operations inspectable. In practice, agentic architectures include components

to log key events: each prompt an agent generates, each tool API call and its result, each decision or plan the agent commits to, etc. Such audit logs enable post-hoc analysis, error tracing, and explanations—for example, if a workflow failed or a compliance issue occurred, developers can replay the agent interactions to pinpoint the cause. Some frameworks even expose an agent’s chain-of-thought (the intermediate reasoning steps) in a controlled way for debugging or compliance review. Beyond logging, observability can be enhanced through dashboarding and alerts: e.g. real-time monitors that track agent performance metrics or detect anomalies (like an agent taking too long on a task or generating an out-of-bounds output). The end goal is to treat an agentic AI system not as a “black box” automation, but as an observable workflow that operations teams can supervise akin to any critical IT system. This also ties into explainability: by capturing the rationale behind decisions (even if only in approximate form, such as storing the intermediate reasoning text), the system can later provide explanations for its actions, which is invaluable for trust and for continuous improvement. Overall, the literature suggests that designing for transparency—instrumenting agents with logging, and perhaps even designing agents to self-report their status—is a best practice to ensure agentic AI doesn’t become an inscrutable tangle of automations. High observability supports OpEx principles by enabling traceability, accountability, and faster incident response when something goes wrong.

Finally, a recurrent theme is the need for strong governance mechanisms in agentic AI architectures to ensure alignment with rules, ethics, and organizational policies. By their nature, autonomous agents may produce unexpected or undesired outcomes—a risk amplified in multi-agent settings where interactions are complex and no single agent has full oversight. Without proper governance, an agentic system could easily violate compliance requirements or strategic constraints, undermining OpEx goals (e.g. a well-intentioned agent might inadvertently expose sensitive data or execute an unauthorized transaction). In fact, Gaurav et al. (2025) warn that the “absence of scalable, decoupled governance remains a structural liability” in today’s agentic AI ecosystems. To address this, researchers are exploring policy-enforcement layers that sit between the agents and the outside world. One such approach is Governance-as-a-Service (GaaS), a framework that intercepts agent actions at runtime and checks them against explicit rules or constraints. Rather than trusting each agent to self-regulate, an external governance layer can block or redirect high-risk actions, log rule violations, and even adapt penalties or restrictions on agents that

exhibit misbehavior over time. This effectively creates an oversight controller for the MAS—analogous to a “compliance officer” in a human organization—that ensures no single agent can compromise the system’s integrity. Key design elements include declarative policy rules (defining allowable vs. disallowed outputs or tool uses), a mechanism to monitor all agent outputs (to flag violations), and possibly a trust score or reputation model to quantify an agent’s reliability based on past behavior. Beyond automated enforcement, governance also encompasses human oversight: for example, requiring human approval for certain agent decisions (human-in-the-loop checkpoints) or having a fallback where a human operator can intervene if the agents encounter an ambiguous ethical situation. In classical MAS research, analogous concepts existed like normative agents and electronic institutions that enforce “rules of engagement” among agents; the new twist is that with LLM-based agents we must often treat the models as black boxes, so governance can’t be injected into their internal logic easily and must surround them instead. The overarching recommendation is that any architecture for generative multi-agent workflows should bake in governance from the start—not as an afterthought—to manage risk. As one author succinctly put it, such a system “does not teach agents ethics; it enforces them”. This governance emphasis aligns tightly with OpEx goals of compliance, risk management, and trustworthiness.

In summary, the literature portrays agentic AI as a powerful paradigm for workflow automation that, if well-designed, can dramatically enhance agility, adaptability, and decision quality in operations. By combining autonomous, tool-using agents into coordinated architectures, organizations can automate complex processes that previously required human judgment. At the same time, achieving sustainable excellence with such systems demands careful attention to transparency and control: architects must ensure agents remain observable and governable to uphold compliance and reliability standards. These insights set the stage for the next section of this thesis, which will integrate OpEx principles, workflow automation requirements, and agentic AI capabilities into a unified reference architecture. The themes of autonomy, coordination, and governance identified here directly inform the design choices and requirements elaborated in the subsequent chapters.

4 Requirements Modeling

The literature review identified recurring design concerns across operational excellence, workflow automation, and agentic AI. Synthesizing these insights yields *elicitation lists* which represent the initial outcome of requirements documentation based on systematically coded sources. A subsequent *clustering* step consolidated overlapping items into a unified set of requirement candidates.

Each candidate was then reformulated into an atomic, unambiguous, and verifiable “shall” statement, following the best-practice formulation rules of Glinz et al. (2020). The final requirements were organized into *functional requirements*, *quality requirements*, and *constraints*, in line with the Glinz taxonomy.

Requirements were then represented in SysML v2 as dedicated requirement elements, with their textual statements captured in the description field. Trace links connect each requirement to its source in the elicitation lists and to the architecture elements that *«satisfy»* it. This model-based representation ensures that design decisions remain traceable to validated needs and that requirement coverage can later be verified systematically.

4.1 Clustering and Consolidation

In order to derive actionable system requirements from the literature, a structured clustering process was applied. This eliminated redundancies, consolidated overlapping issues, and normalized vocabulary across disciplines. The process began by translating *coded statements* from each domain into elicitation items—discrete, *design-relevant units* derived from the coding units identified in the QCA process described in Section 2.1.

O — OPERATIONAL EXCELLENCE

1. ADAPTABILITY AND AGILITY — processes must remain reconfigurable in response to volatile conditions, ensuring resilience in dynamic environments.
2. COMPLIANCE AND RISK MANAGEMENT — regulatory adherence and transparency must be embedded into workflows to minimize compliance risks.
3. DECISION QUALITY — automation should enable data-driven, timely, and well-informed decisions rather than simply increasing speed.

4. EFFICIENCY AND CONTINUOUS IMPROVEMENT — workflows should reduce manual effort, eliminate waste, and institutionalize iterative refinements.
5. CUSTOMER-CENTRICITY — operations must align with user needs and service-level commitments to sustain value delivery.
6. USER EMPOWERMENT AND CULTURE — systems should support collaboration, transparency, and employee engagement in improvement processes.
7. TECHNOLOGY INTEGRATION AND SCALABILITY — architectures must accommodate automation, AI, and cloud services to enable sustainable innovation.

W — WORKFLOW AUTOMATION

1. PROCESS ORCHESTRATION — workflow engines must enforce task sequences and business rules to guarantee reliable execution.
2. INTEGRATION AND INTEROPERABILITY — automation must seamlessly connect heterogeneous applications, data sources, and organizational boundaries.
3. MODULARITY AND REUSABILITY — workflows should be composed of modular tasks or subprocesses that can be reused and reconfigured with minimal effort.
4. EXCEPTION HANDLING AND FLEXIBILITY — systems must detect, manage, and escalate deviations rather than failing in unforeseen scenarios.
5. WORKFLOW GOVERNANCE — monitoring, audit trails, and role-based controls must ensure accountability and compliance throughout automated processes.

A — AGENTIC AI

1. AUTONOMY IN DECISION-MAKING — agents should operate independently within clearly scoped authority to enhance agility while managing risks.
2. TOOL USE AND INTEGRATION — agents must invoke external tools, APIs, or services reliably, requiring robust interfaces and safeguards.
3. COORDINATION AND SPECIALIZATION — multi-agent systems should divide labor through explicit roles and structured coordination mechanisms.
4. OBSERVABILITY AND TRANSPARENCY — all agent actions and decisions must be logged and explainable to support trust, debugging, and compliance.
5. GOVERNANCE AND COMPLIANCE — oversight mechanisms, including policy enforcement layers and human-in-the-loop checkpoints, are essential to align agent behavior with organizational and ethical standards.

To reduce redundancy and ensure conceptual clarity, the elicitation items from each domain were compared and consolidated based on semantic similarity and functional overlap. Particular attention was given to cross-domain intersections—such

as governance appearing in both workflow automation and agentic AI—and to areas where multiple coding units aligned on a shared concern. The result of this consolidation step is a reduced set of requirement candidates, each traceable to one or more domain-specific clusters. These are summarized in Table 4.1, which maps the original coded clusters to the consolidated requirement areas used for formulation.

CLUSTER IDS	SOURCE DOMAINS	CONSOLIDATED REQ. AREA
o1	OpEx	ADAPTABILITY AND AGILITY
o2, w5, a5	All three	GOVERNANCE & COMPL.
o3, a1	OpEx, Agentic AI	DECISION AUTONOMY
w1, w3	Work. Aut.	ORCHEST. & MODULARITY
w4, a3	Work. Aut., Agentic AI	EXCEP. HANDLING & COORD.
a4, o2	Agentic AI, OpEx	OBSERVABILITY & TRAC.
w2, a2	Work. Aut., Agentic AI	TOOL INTEGRATION
o4	OpEx	CONTINUOUS IMPROVEMENT
o5	OpEx	CUSTOMER-CENTRICITY
o6	OpEx	USER EMPOWERMENT
o7	OpEx	TECH. INTEGRATION & SCALE

Table 4.1: Mapping of Domain-Specific Clusters to Consolidated Requirement Areas

This structured consolidation establishes a coherent foundation for the subsequent refinement of architecture-level requirements.

4.2 Reformulation and Classification

Following Glinz et al. (2020), each requirement is formulated as a single, unambiguous “shall” statement that is necessary, atomic, verifiable, and consistent. Requirements are classified into functional, quality, and constraint types. In SysML v2, requirements are modeled as dedicated requirement elements with their textual content captured in the description field, and their relationships (e.g. «satisfy», «verify») expressed through model links. Rationale and verification considerations are documented narratively in the requirements analysis and traceability sections, while the requirements themselves remain concise textual statements embedded in the SysML v2 model.

Functional requirements specify externally visible system behaviors, separated from quality attributes and constraints. They were elicited from the categorized insights of the literature review, documented in “shall” form, and prepared for trace links in the SysML requirements model.

4.3 Model-Based Representation

5 Architecture Modeling

Listing 1: Excerpt of the Requirements model

5.1 [placeholder]Modeling the Agents

5.2 [placeholder]Modeling the Architecture

5.3 [placeholder]Modeling the Interactions

6 Discussion: Applicability Criteria

7 Conclusion

Future work should extend this conceptual design into practical evaluation and implementation. In particular, empirical validation of the architecture in industry settings, tool-supported instantiation in SysML, and comparative studies against traditional workflow automation would provide valuable evidence of its applicability and impact. Further, integrating additional agentic AI capabilities such as autonomous negotiation or explainability could enhance both usability and compliance assurance.

References

- Basu, Amit and Akhil Kumar (2002). “Research Commentary: Workflow Management Issues in e-Business”. *Information Systems Research* 13, pp. 1–14.
- Carvalho, André M. et al. (2023). “Operational excellence, organizational culture, and agility: bridging the gap between quality and adaptability”. *Total Quality Management & Business Excellence* 34, pp. 1598–1628.
- Castelfranchi, Cristiano (1998). “Modelling social action for AI agents”. *Artificial Intelligence* 103, pp. 157–182.
- Ferber, Jacques (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*. 1st ed. Harlow Bonn: Addison-Wesley, pp. 479–498.
- Gaurav, Suyash et al. (2025). *Governance-as-a-Service: A Multi-Agent Framework for AI System Compliance and Policy Enforcement*.
- Georgakopoulos, Diimitrios et al. (1995). “An overview of workflow management: From process modeling to workflow automation infrastructure”. *Distributed and Parallel Databases* 3, pp. 119–153.
- Glinz, Martin et al. (2020). *Handbook for the CPRE Foundation Level according to the IREB Standard*. International Requirements Engineering Board.
- Herrmann, Andrea (2022). *Grundlagen der Anforderungsanalyse: Standardkonformes Requirements Engineering*. Wiesbaden Heidelberg: Springer Vieweg.
- Hevner, Alan R. et al. (2004). “Design Science in Information Systems Research”. *MIS Quarterly* 28, pp. 75–105.
- IEEE (1990). *Standard Glossary of Software Engineering Terminology*. Corrected ed. New York: Institute of Electrical and Electronics Engineers.
- Juran, Joseph Moses and A. Blanton Godfrey (1999). *Juran’s quality handbook*. 5th ed. McGraw Hill.
- Mayring, Philipp and Thomas Fenzl (2022). “Qualitative Inhaltsanalyse”. *Handbuch Methoden der empirischen Sozialforschung*. Springer Fachmedien Wiesbaden, pp. 691–706.
- Owoade, Samuel et al. (2024). “Systematic Review of Strategic Business Administration Practices for Driving Operational Excellence in IT-Driven Firms”. *International Journal of Scientific Research in Science and Technology* 11, pp. 680–700.
- Peppers, Ken et al. (2007). “A design science research methodology for information systems research”. *Journal of Management Information Systems* 24, pp. 45–77.

- Qian, Chen et al. (2024). *ChatDev: Communicative Agents for Software Development*.
- Russell, Stuart et al. (2015). “Research Priorities for Robust and Beneficial Artificial Intelligence”. Version 1.
- Sapkota, Ranjan et al. (2025). “AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges”. *Information Fusion* 126, pp. 103–599.
- Shu, Raphael et al. (2024). *Towards Effective GenAI Multi-Agent Collaboration: Design and Evaluation for Enterprise Applications*.
- Stohr, Edward A. and J. Leon Zhao (2001). “Workflow Automation: Overview and Research Issues”. *Information Systems Frontiers* 3, pp. 281–296.
- Womack, James P. and Daniel T. Jones (2013). *Lean Thinking: Ballast abwerfen, Unternehmensgewinn steigern*. 3. Auflage. Weinheim: Campus Verlag.
- Yang, Hui et al. (2023). *Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions*.
- Yao, Shunyu et al. (2023). *ReAct: Synergizing Reasoning and Acting in Language Models*.

Appendix A: Full SysML Models

Requirements Model

Architecture Model

Interactions Model