Value-Preserving Architectures for Agentic AI Systems

Alessandro Pesare[✉*]
TU Wien
Vienna, Austria
alessandro.pesare@tuwien.ac.at

Tommaso Dolci[*]
TU Wien
Vienna, Austria
tommaso.dolci@tuwien.ac.at

Katja Hose
TU Wien
Vienna, Austria
katja.hose@tuwien.ac.at

Emanuel Sallinger
TU Wien
Vienna, Austria
emanuel.sallinger@tuwien.ac.at

Abstract

The emergence of agentic AI and LLM-based multi-agent systems (MAS) presents unprecedented opportunities for automating complex tasks, while simultaneously raising critical concerns about the preservation of fundamental human-centered values, such as privacy, fairness, and safety. Although software engineering has traditionally focused on functional correctness, the adoption of LLMs and AI agents into complex socio-technical systems has intensified the need for responsible software engineering and robust value alignment. In MAS, architectural design decisions, such as coordination mechanisms, communication protocols, and system topologies, play a central role in shaping system behavior and the outcomes they produce. This paper argues that architectural choices influence not only the functionality and performance of MAS but can also promote value-oriented system behavior. Therefore, we investigate how different architectural designs support different human-centered values, discussing the following value-preserving architectural patterns: (i) a privacy-aware architecture with a federated topology, (ii) a distributed architecture to promote pluralism and diversity, and (iii) a guard-agent architecture to detect and mitigate unfairness. Finally, we introduce representative use cases to illustrate the proposed architectures in real-world scenarios. By linking architectural design with human-centered values, this work lays the foundation for a unified set of architectural patterns and guidelines towards the design of trustworthy MAS.

CCS Concepts

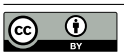
• **Software and its engineering** → **Extra-functional properties**; **Software design engineering**; • **Computing methodologies** → **Multi-agent systems**; **Intelligent agents**.

Keywords

Agentic AI, Multi-Agent System, Responsible Software Engineering

ACM Reference Format:
Alessandro Pesare, Tommaso Dolci, Katja Hose, and Emanuel Sallinger. 2026. Value-Preserving Architectures for Agentic AI Systems. In *Proceedings of the 1st International Workshop on Agentic AI for Next-Generation Software Development (AgenticDev '26), October 12–16, 2026, Munich, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3843282.3844436

[*]Both authors contributed equally to the paper.



This work is licensed under a Creative Commons Attribution 4.0 International License.
AgenticDev '26, Munich, Germany
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2985-0/2026/10
https://doi.org/10.1145/3843282.3844436

1 Introduction

Agentic AI is rapidly reshaping how software is conceived, built, and operated. AI agents re-frame the use of large language models (LLMs) from passive text generation to specialized autonomous entities capable of planning, using external tool, and adapting according to feedback at runtime [37]. For instance, a travel-planning AI agent can autonomously search online, take actions such as booking flights and hotels, and adapt its plan when circumstances change, such as a flight cancellation. The same principles of autonomy, planning, and tool use can be extended beyond a single agent. Multi-agent systems (MAS) orchestrate multiple agents that collaborate through structured communication to achieve more complex objectives by distributing a common goal across several specialized agents, leveraging problem decomposition and parallel execution that exceed single agent capability [1, 36]. In particular, agentic AI and MAS have attracted significant attention in software engineering, where coordinated agents are employed for automating requirements elicitation, code synthesis, test generation, and decision support across the software lifecycle [19, 20].

However, as the number of autonomous agents and the depth of their interactions increase, MAS gain more capability at the cost of reduced human oversight, diminished transparency, and weaker guarantees about the resulting software artifacts [12]. In the absence of both human oversight and adequate system-level design guarantees, a single failure at a coordination step can silently propagate across the entire system [18]. For instance, when personal data traverses agent boundaries to be processed outside their original domain, the decision-making process can violate the principle of data minimization [22]. In architectures governed by a central orchestrator agent, the system's implicit preferences and decision policies can influence the final outcome by filtering out minority or dissenting perspectives [3, 14]. At the same time, when agents are conditioned by historically biased data, a number of social biases can be amplified and propagated through successive stages of processing, ultimately producing outputs that embed discrimination while appearing fluent and authoritative [27, 40].

While software engineering has traditionally focused on functional correctness and technical reliability, the emergence of complex AI-based socio-technical systems demands increasing attention towards ethical awareness, safety, and alignment to values

such as privacy, fairness, and security [24, 34]. In response to this concern, researchers have emphasized the importance of responsible software engineering [7, 29] and the need to operationalize human-centered values throughout the software development lifecycle, from requirements specification to software testing [26, 30]. Recently, the literature has highlighted the need for architectural pattern and design techniques to operationalize the principles of responsible AI and align software to human-centered values [24].

A common approach to align agentic AI systems is to treat value guarantees as post-hoc constraints by introducing system-level guardrails, by filtering the final output or enforcing policies through human-in-the-loop intervention applied to an otherwise unconstrained architecture [21, 31]. However, these mechanisms depend on safeguards defined *a priori* and inspect outputs only after upstream transformations, when biased or harmful information have already been introduced, aggregated, and propagated. This limitation suggests that value preservation cannot be delegated to post-hoc validation only: architectural decisions such as coordination mechanisms, communication protocols, and system topologies play a central role in determining how values are ultimately realized, complementing requirements specification and system testing. Therefore, as system behavior emerges from the interactions between autonomous agents, the design of MAS architectures becomes central, and the following research question emerges: *how can architectural design be leveraged to support human-centered values in the era of agentic AI?*

This vision paper highlights the importance of MAS architectures to satisfy privacy, fairness, and pluralism in a structured way, by addressing recurring human-centered requirements in the form of reusable value-preserving architectural patterns. While previous works have proposed general agentic AI pattern catalogs [23] and studied the need for security [28], privacy [22, 39], fairness [2, 10], and pluralism [3, 14] in the context of MAS, this paper investigates the interplay between agentic architectural choices and human-centered values, representing a first step towards responsible software engineering for the design of MAS. We see this as a concrete step for engineers to retain control over human values in the systems they build. This paper makes the following contributions:

1. We frame the **principal challenges of responsible software engineering** in the context of MAS, highlighting the interplay between human-centered values in agentic AI systems and architectural design.
2. We discuss three **value-preserving architectural patterns** – Federated Silos Coordination, Peer-to-Peer Deliberation, and Plan-Triggered Guard Agents, preserving privacy, pluralism, and fairness respectively – presenting their design, agent responsibilities, and coordination flow.
3. We illustrate the architectural patterns with **concrete use-case scenarios**, showing how their design structurally supports the target value.

The rest of the paper is organized as follows. Section 2 provides an overview of agentic AI and related work in the context of responsible software engineering. Section 3 describes the three value-preserving architectural patterns for privacy, pluralism, and fairness. Section 4 illustrates the proposed design on real-world scenarios. Finally, Section 5 concludes the paper by outlining research directions towards the creation of a unified catalog of value-preserving architectural patterns for MAS.

2 Background

Agentic AI and MAS. An agentic AI system is an LLM-powered entity that (i) acts autonomously without step-by-step human instruction, (ii) plans by decomposing a high-level goal into ordered sub-tasks, and (iii) uses tools – APIs, databases, code interpreters – to execute them. Cognitive-architecture treatments map the memory, action, and decision modules any language agent requires, clarifying why planning and tool use are constitutive features of agency rather than add-ons [35]; complementary work shows that iterative self-reflection over execution traces, rather than a single forward pass, is what let agents recover from failures on long-horizon tasks [32]. Already at the single-agent level this autonomy trades transparency for capability: reasoning unfolds across many internal steps and tool calls that do not surface to the human, thus reducing the interpretability of the rationale behind a given action.

MAS amplify this opacity along three axes. First, *behavioral* opacity: coordinating several specialized agents under a shared objective yields decomposition, per-domain skill, and resilience to localized failures [20], but the resulting behavior is an emergent property of inter-agent interaction rather than of any inspectable component, so it cannot be read off from any single agent. Second, *attributional* opacity: when an outcome is produced jointly by many autonomous agents exchanging intermediate results, responsibility for it – an exposed record, a discarded viewpoint, a biased inference – is diluted across the network and hard to trace back to a single locus [12, 22]. Third, *control* opacity: in the canonical orchestrator topology every sub-task is mediated by natural-language instructions flowing from one node, so a semantic failure there propagates silently to the whole network whenever no human is observing [18], and the depth of the interaction graph leaves few points at which a human could meaningfully intervene. Opacity is thus the structural liability that makes human values preservation so difficult to guarantee by external means, a gap only partially closed by human-in-the-loop [25]. This motivates moving the guarantee into the architecture itself.

Responsible AI and Software Engineering. The growing societal impact of software systems, and AI-based software in particular, has shifted software engineering research and practice towards explicitly incorporating ethical considerations throughout the development lifecycle. *Ethics-aware software engineering* [4] proposes to systematically capture and analyze stakeholder values during software specification and development, including trustworthy principles such as fairness, interpretability, and safety. Building on this direction, a bridge between software engineering and responsible AI has been proposed [24, 41] to guide the design, development, and deployment of AI-based software while accounting for their broader societal impact and ensuring alignment with human-centered values. *Responsible software engineering* elevates the centrality of human-centered values such as privacy, fairness, and explainability to first-class requirements and design goals, treating them with the same rigorous methodology traditionally reserved for functional requirements [7, 29]. This perspective resonates with

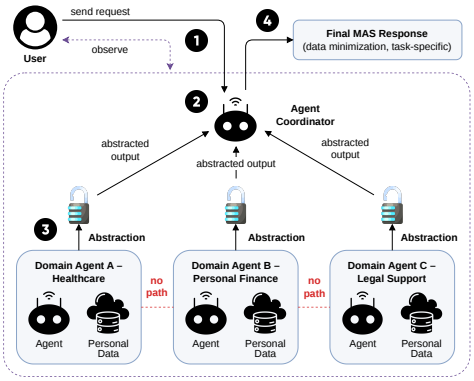


Figure 1: Federated Silos Coordination Pattern

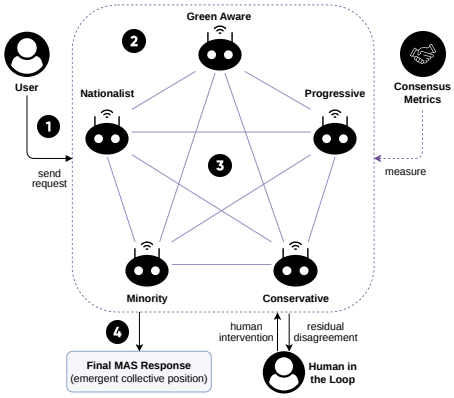


Figure 2: Peer-to-Peer Deliberation Pattern

other initiatives such as value sensitive design [16, 42], which advocate for a principled and systematic account of stakeholder values throughout the design process. Operationalizing human values in software systems denotes the process of identifying relevant user values and translating them into accessible, concrete concepts that can be implemented, validated, verified, and measured within software systems [15, 26, 30, 38]. More recently, the emergence of LLM-based software agents has given rise to the notion of *socio-critical systems*, a new category of software systems whose behavior can have significant ethical and societal consequences, underscoring the need to embed human-centered values not only at runtime, but already during their development stages [5].

3 Value-Preserving Architectures

We present three architectural patterns, each targeting a specific human-centered value: Federated Silos Coordination for privacy (Section 3.1), Peer-to-Peer Deliberation for pluralism (Section 3.2), and Plan-Triggered Guard Agents for fairness (Section 3.3). For each pattern, we first define the target value it aims to preserve and give an overview of the corresponding architecture. We then specify the pattern in a context–problem–solution form [6, 11]. The *context* describes a situation that gives rise to a problem, the *problem* outlines the issues and key constraints that arise in that context, and the *solution* describes the architecture that address it.

3.1 Federated Silos Coordination Pattern

Privacy guarantees the user's control over the access and use of their personal data. In the context of MAS, we extend this traditional notion to include fine-grained control over what is shared with each autonomous agent: personal data must be governed not only at the boundary between the system and the outside world, but also within the system. Figure 1 presents the *Federated Silos Coordination* architectural pattern, in which a coordinator agent orchestrates the overall workflow by collecting and combining only the minimal output required from each domain agent, without accessing or storing the underlying data itself. Domain agents never exchange information among themselves: the only information leaving a silo consists of minimal, task-specific abstractions, in accordance with the principle of *data minimization*, which requires using only the information strictly necessary for the task at hand.

Federated Silos Coordination Pattern

Context. A MAS operating over disjoint domains of sensitive, subject-specific information, and a user-facing task that may require combining cross-domain information.

Problem. Requests must be routed only to the minimal subset of domain agents required for completion. When cross-domain combination is needed, any direct exchange between domain agents risks leaking sensitive data beyond the domain to which it belongs. Filtering only the final answer is insufficient: once a detailed value crosses a domain boundary, it can influence subsequent intermediate reasoning, making its effects impossible for a final-output filter to reliably detect or reverse. The architecture must therefore guarantee that (i) no domain agent ever observes another domain's records, (ii) any value leaving a domain carries no more precision than the task requires, and (iii) data minimization is promoted while retaining the information necessary to complete the task.

Solution. Organize the system around a single *coordinator* and a set of *domain agents*, providing an architectural enforcement point for privacy. First, at each domain agent, interpose a local *abstraction step*: every value the agent is about to externalize is generalized before emission (e.g., a scalar is bucketed into a range, a detailed attribute is mapped onto a coarser category). Second, restrict the communication topology to a hub-and-spoke form: abstracted representations are emitted only to the coordinator and never to sibling domain agents, so no direct inter-domain channel exists. The coordinator routes each request only to the minimal subset of specialized agents the task requires and integrates their intermediate outputs into a coherent response; this integration is *selective*, governed by data minimization: only the signals strictly necessary for task completion are combined, and subject-specific details not required by the final output are never exposed.

3.2 Peer-to-Peer Deliberation Pattern

Pluralism is the belief that diverse groups are represented equally and fairly within the system, promoting diversity and active cooperation. Figure 2 presents an overview of the *Peer-to-Peer Deliberation* architectural pattern where pluralism each agent communicates

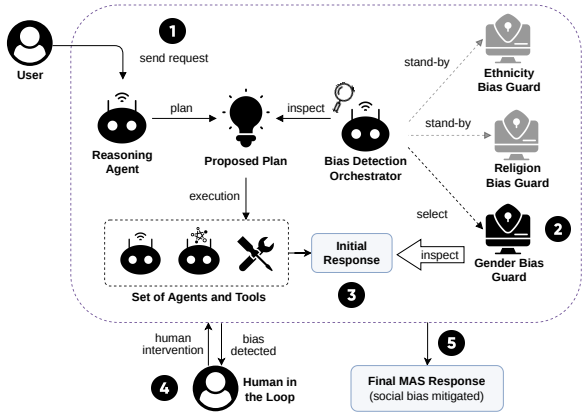


Figure 3: Plan-Triggered Guard Agents Pattern

directly with each other, no central orchestrator entity controls the deliberation, and the collective position emerges from iterative exchange rather than from aggregation by an authoritative node. Each agent impersonates a distinct deliberative position and no node holds final authority over the output.

Peer-to-Peer Deliberation Pattern

Context. A MAS in which a user-facing task requires considering multiple legitimate perspectives, values, or viewpoints, and no single agent's opinion should be given overriding authority.
Problem. Any centralized coordination point risks privileging one perspective over the others. At the same time, a fully decentralized deliberation among opinionated agents risks never converging, with each agent defending its own position indefinitely. The architecture must therefore guarantee that (i) no single agent can determine the outcome, (ii) diverse deliberative positions are structurally represented, and (iii) the deliberation terminates, either in a shared position or in an explicit account of the residual disagreement.
Solution. Remove any privileged coordinator and arrange role-bearing agents in a fully decentralized, complete communication graph: every agent communicates directly with every other. Assign each agent a distinct deliberative position, so that the space of perspectives is structurally covered. To resolve the tension between pluralism and convergence, define a *consensus metric* that quantifies how far the agents currently are from a shared position. The metric serves two purposes: it provides a measurable stopping condition, turning an open-ended exchange into a process with readable progress; and it enables intervention in the agents' feedback loops: when the metric signals stalling or polarization, the system nudges agents towards compromise without imposing any single opinion. The objective is not to manufacture agreement at any cost, which would itself erode pluralism, but to reach well-founded compromise whenever possible. When no such compromise is reached, the system does not force convergence: the residual disagreement, together with the positions held by each agent, is surfaced to the human user, who retains final decision authority.

3.3 Plan-Triggered Guard Agents Pattern

Fairness requires that no group or individual subject is systematically disadvantaged or discriminated by the system. Figure 3 illustrates the *Plan-Triggered Guard Agents* pattern where fairness is promoted by interposing specialized guard agents into the processing flow, each agent targeting a specific bias type (e.g., gender, ethnicity, religion). Rather than inspecting the final output where bias is hard to detect, guards intercept and correct bias at runtime, before it contaminates the downstream steps. Guards are modular nodes, attached to but separable from the core pipeline.

Plan-Triggered Guard Agents Pattern

Context. A MAS supporting decision-making tasks that affect individuals or groups (e.g., hiring, credit, healthcare), where the system output is susceptible to one or more classes of bias.
Problem. Bias can emerge at any phase of the decision-making process, and even a single biased step may compromise the fairness of the entire outcome: a biased intermediate action propagates through subsequent agentic steps and shapes the result in ways that a post-hoc check might not be able to reverse. At the same time, not every task carries the same risk: applying every possible bias check to every step is impractical. The architecture must therefore guarantee that (i) bias detection is targeted to the specific risks of the decision-making process rather than applied indiscriminately, (ii) each detected bias is handled by a component specialized for that bias class to enhance modularity, (iii) mitigation occurs before performing any action (e.g. tool invocation, answer generation, skill selection).
Solution. Separate the pipeline into three phases: *plan*, *review*, and *execute*. In the plan phase, the agent reasons over the task and emits a proposed action plan without carrying out any action. In the review phase, a *bias-detection orchestrator* inspects the plan and selectively activates specialized *guard agents*, each designed to detect and mitigate a specific class of bias; only the guards warranted by the structure and risk profile of the plan are employed. Each activated guard verifies the presence or absence of its target bias and intervenes before any action; when a detected bias cannot be automatically mitigated, the plan is flagged and escalated to a human reviewer, who retains final decision authority. Because the guard agents are decoupled from the core pipeline, they can be configured per domain, updated, or replaced independently.

4 Patterns in Practice: Application Examples

In this section, we assess the architectural patterns from the previous section on different use-case scenarios, showing how their design structurally sustain the target values.

4.1 Federated Silos Coordination for Privacy

Consider a general-purpose MAS designed to support personalized decision-making across multiple domains, e.g., healthcare and financial planning, by processing user personal data. The primary stakeholder is the end user, entitled to receive decision-support and personalized suggestions with privacy preservation and guarantees of data minimization. In conventional MAS architectures, agents

may share or access unnecessary cross-domain personal data, maximizing the use of data. However, in a personalized healthcare scenario, a medical agent should not access legal or financial personal data: previous legal disputes related to hospital bills, outstanding debts, or low income should not influence medical recommendations, which must rely exclusively on health-related information.

Scenario Workflow. The user requests personalized decision support for a suitable treatment for hypertension (step ❶ in Figure 1). Upon receiving the request, the agent coordinator identifies the medical domain and dispatches the task to the medical agent (step ❷), which accesses only the user's clinically relevant information (e.g., medical history, current medications, and diagnostic results) in accordance to the principle of data minimization. The agent invokes external tools to generate treatment recommendations, e.g., machine-learning models for precision medicine (step ❸), without accessing personal information about unrelated domains, e.g., financial status or legal records. The final recommendation is returned to the coordinator, which outputs the result to the user (step ❹). Suppose the user explicitly requests a treatment recommendation considering financial affordability: the coordinator would independently dispatch the request to both the medical and financial agents. Each agent processes only the data required for its domain and produces independent, abstracted responses on available treatments and personal finance. The coordinator presents the outcomes to the user without exchanging or merging personal data across domains, preventing information leakage by design.

4.2 Peer-to-Peer Deliberation for Pluralism

Consider a MAS designed to retrieve, aggregate, and summarize online news on current events. The main stakeholders include the end users, who consume the generated summaries, and the news providers, whose diverse perspectives should be equally represented. Diversity is the key system requirement to ensure that generated summaries reflect a plurality of viewpoints rather than only dominant or polarized narratives. In conventional agentic architectures, information retrieval and summarization are typically centralized. As a result, the system may over-represent majority viewpoints, either because they are more prevalent in the retrieved sources or due to the central LLM tendency to produce summaries aligned with its internal preferences, potentially overlooking minority or dissenting perspectives.

Scenario Workflow. A user requests a summary of the latest international climate summit (step ❶ in Figure 2). The user request is forwarded to multiple peer news-retrieval agents. Each agent independently gathers information from a distinct set of news sources, producing a preliminary summary that reflects their alignment: the nationalist agent emphasizes the need for energy autonomy at the expense of climate impact, the green-aware agent stresses the need for climate action, and the minority agent highlights concerns from developing countries (step ❷). The agents exchange and discuss their findings through a deliberation phase, allowing alternative, and minority viewpoints to be surfaced rather than suppressed by a single centralized summarization process (step ❸). The resulting perspectives represent both majority opinions and minority opinions promoting representation of diverse viewpoints and reduces the risk of over-emphasizing dominant narratives (step ❹).

4.3 Plan-Triggered Guard Agents for Fairness

Consider an agentic decision-support system for human resource management. The system supports multiple tasks through external tool invocation, e.g., automated resume parsing, candidate evaluation, or generating reports about employees. While fairness and equal treatment are critical for bias-sensitive tasks such as resume parsing [9, 13], it is less relevant for descriptive tasks, motivating selective activation of fairness-preserving mechanisms. In this context, the stakeholders include job applicants, whose opportunities may be affected by automated decisions, and the company, whose reputation and regulatory compliance depend on fair hiring practices. In conventional architectures, LLM-based agents may inherit biases from historical training data and reproduce discriminatory patterns through statistical associations. Furthermore, while agentic workflows typically include system-level guardrails and external detection systems, biases introduced by one agent may propagate or even be amplified across reasoning stages within the system.

Scenario Workflow. A recruiter requests to rank candidates for a software engineering position (step ❶ in Figure 3). Two candidates are considered: Candidate A is male, while Candidate B is female. Upon identifying the task as fairness-critical, the bias-detection orchestrator activates the gender guard agents, which inspect the intermediate ranking and identify that sensitive attributes may influence the recommendation (step ❷). Although Candidate B has prior professional experience that better match the job requirements, the agentic workflow initially ranks Candidate A higher, reflecting the historical bias of software engineering been predominantly associated with males (step ❸). The workflow is therefore flagged and the task requires the intervention of the human recruiter (step ❹), who reviews the candidates and ultimately selects Candidate B, preventing unfair outcomes (step ❺).

5 Conclusions and Research Directions

In this paper, we investigated how architectural choices in MAS can reinforce human-centered values by design. We introduced an initial set of MAS architectural design patterns to support privacy, pluralism, and fairness, laying the foundation for a unified catalog of patterns to develop human-centered MAS. Future research will investigate how different agentic patterns can be composed to satisfy multiple value requirements simultaneously, examining the synergies, trade-offs, and limitations that arise when coordinating architectures designed to optimize different values. Moreover, we plan to address the problem of adapting agentic systems to evolving stakeholder requirements and the potential shift of values at system runtime [8, 17]. Finally, the definition of metrics is another crucial challenge for evaluating value preservation, representing an important step towards operationalizing values in software systems [30], e.g., metrics to measure pluralistic alignment [33].

Acknowledgments

This work is supported by ARMADA, funded by the European Union's Horizon Europe Marie Skłodowska-Curie Actions (MSCA) under grant No. 101168951, by the Vienna Science and Technology Fund (WWTF) grant Nos. 10.47379/VRG18013, 10.47379/ICT25032, 10.47379/NXT22018, 10.47379/ICT2201, 10.47379/DCDH001, and by the Austrian Science Fund (FWF) grant No. 10.55776/COE12.

References

[1] Apoorva Adimulam, Rajesh Gupta, and Sumit Kumar. 2026. The Orchestration of Multi-Agent Systems: Architectures, Protocols, and Enterprise Adoption. doi:10.48550/arXiv.2601.13671

[2] Amanda Aird, Paresha Farastu, Joshua Sun, Elena Stefancová, Cassidy All, Amy Voida, Nicholas Mattei, and Robin Burke. 2024. Dynamic Fairness-aware Recommendation Through Multi-agent Social Choice. *ACM Trans. Recomm. Syst.* 3, 2 (2024), 21:1–21:35. doi:10.1145/3690653

[3] Joshua Ashkinaze, Emily Fry, Narendra Edara, Eric Gilbert, and Ceren Budak. 2025. Plurals: A System for Guiding LLMs via Simulated Social Ensembles. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. ACM, 1–21. doi:10.1145/3706598.3713675

[4] Fatma Başak Aydemir and Fabiano Dalpiaz. 2018. A Roadmap for Ethics-Aware Software Engineering. In *Proceedings of the International Workshop on Software Fairness*. ACM, 15–21. doi:10.1145/3194770.3194778

[5] Luciano Baresi, Matteo Camilli, Tommaso Dolci, and Giovanni Quattrocchi. 2024. A Conceptual Framework for Quality Assurance of LLM-based Socio-critical Systems. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2314–2318. doi:10.1145/3691620.3695306

[6] Len Bass, Paul Clements, and Rick Kazman. 2003. *Software Architecture in Practice* (2 ed.). Addison-Wesley Longman Publishing Co., Inc., USA.

[7] Amel Bennaceur, Carlo Ghezzi, Jeff Kramer, and Bashar Nuseibeh. 2024. Responsible Software Engineering: Requirements and Goals. In *Introduction to Digital Humanism: A Textbook*. Springer Nature Switzerland, 299–315. doi:10.1007/978-3-031-45304-5_20

[8] Amel Bennaceur, Diane Hassett, Bashar Nuseibeh, and Andrea Zisman. 2023. Values@Runtime: An Adaptive Framework for Operationalising Values. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. 175–179. doi:10.1109/ICSE-SEIS58686.2023.00024

[9] Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man Is to Computer Programmer as Woman Is to Homemaker? Debiasing Word Embeddings. In *Advances in Neural Information Processing Systems*, Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf

[10] Angana Borah and Rada Mihalcea. 2024. Towards Implicit Bias Detection and Mitigation in Multi-Agent LLM Interactions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. ACL, 9306–9326. doi:10.18653/v1/2024.findings-emnlp.545

[11] Matteo Camilli, Elisabetta Di Nitto, Alfonso Fuggetta, Alessandro Margara, Matteo Giovanni Rossi, Damian Andrew Tamburri, et al. 2025. *Software Engineering-A Structured Journey from Inception to Delivery*. Bocconi University Press.

[12] Alan Chan, Rebecca Salganik, Alva Markelius, Chris Pang, Nitarshan Rajkumar, Dmitrii Krasheninnikov, Lauro Langosco, Zhonghao He, et al. 2023. Harms from Increasingly Agentic Algorithmic Systems. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23)*. ACM, 651–666. doi:10.1145/3593013.3594033

[13] Tommaso Dolci, Fabio Azzalini, and Mara Tanelli. 2023. Improving Gender-Related Fairness in Sentence Encoders: A Semantics-Based Approach. *Data Science and Engineering* 8, 2 (2023), 177–195. doi:10.1007/s41019-023-00211-0

[14] Shangbin Feng, Taylor Sorensen, Yuhan Liu, Jillian Fisher, Chan Young Park, Yejin Choi, and Yulia Tsvetkov. 2024. Modular Pluralism: Pluralistic Alignment via Multi-LLM Collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. ACL, 4151–4171. doi:10.18653/v1/2024.emnlp-main.240

[15] Maria Angela Ferrario and Emily Winter. 2023. Applying Human Values Theory to Software Engineering Practice: Lessons and Implications. *IEEE Transactions on Software Engineering* 49, 3 (2023), 973–990. doi:10.1109/TSE.2022.3170087

[16] Batya Friedman, David G. Hendry, and Alan Borning. 2017. A Survey of Value Sensitive Design Methods. *Foundations and Trends in Human–Computer Interaction* 11, 2 (2017), 63–125. doi:10.1561/1100000015

[17] Carlos Gavidia-Calderon, Amel Bennaceur, Anastasia Kordoni, Mark Levine, and Bashar Nuseibeh. 2022. What Do You Want From Me? Adapting Systems to the Uncertainty of Human Preferences. In *2022 IEEE/ACM 44th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 126–130. doi:10.1109/ICSE-NIER55298.2022.9793539

[18] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23)*. ACM, 79–90. doi:10.1145/3605764.3623985

[19] Junda He, Christoph Treude, and David Lo. 2025. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 124:1–124:30. doi:10.1145/3712003

[20] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, et al. 2023. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=VtmBAGCN7o

[21] Juhee Kim, Xiaoyuan Liu, Zhun Wang, Shi Qiu, Bo Li, Wenbo Guo, and Dawn Song. 2026. The Attack and Defense Landscape of Agentic AI: A Comprehensive Survey. doi:10.48550/arXiv.2603.11088

[22] Wenkai Li, Liwen Sun, Zhenxiang Guan, Xuhui Zhou, and Maarten Sap. 2025. 1-2-3 Check: Enhancing Contextual Privacy in LLM via Multi-Agent Reasoning. In *Proceedings of the The First Workshop on LLM Security (LLMSEC)*. ACL, 115–128. https://aclanthology.org/2025.llmsec-1.9/

[23] Yue Liu, Sin Kit Lo, Qinghua Lu, Liming Zhu, Dehai Zhao, Xiwei Xu, Stefan Harrer, and Jon Whittle. 2025. Agent Design Pattern Catalogue: A Collection of Architectural Patterns for Foundation Model Based Agents. *Journal of Systems and Software* 220 (2025), 112278. doi:10.1016/j.jss.2024.112278

[24] Qinghua Lu, Liming Zhu, Xiwei Xu, Jon Whittle, and Zhenchang Xing. 2022. Towards a Roadmap on Software Engineering for Responsible AI. In *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*. ACM, 101–112. doi:10.1145/3522664.3528607

[25] Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. 2023. Human-in-the-Loop Machine Learning: A State of the Art. *Artificial Intelligence Review* 56, 4 (2023), 3005–3054. doi:10.1007/s10462-022-10246-w

[26] Davoud Mougouei, Harsha Perera, Waqar Hussain, Rifat Shams, and Jon Whittle. 2018. Operationalizing human values in software: a research roadmap. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. ACM, 780–784. doi:10.1145/3236024.3264843

[27] Thi-Nhung Nguyen, Linhao Luo, Amardeep Kaur, Rollin Omari, Tamas Abraham, Junae Kim, Thuy-Trang Vu, and Dinh Phung. 2026. The Social Cost of Intelligence: Emergence, Propagation, and Amplification of Stereotypical Bias in Multi-Agent Systems. doi:10.48550/arXiv.2510.10943

[28] Shaina Raza, Ranjan Sapkota, Manoj Karkee, and Christos Emmanouilidis. 2026. TRiSM for Agentic AI: A Review of Trust, Risk, and Security Management in LLM-based Agentic Multi-Agent Systems. *AI Open* 7 (2026), 71–95. doi:10.1016/j.aiopen.2026.02.006

[29] Ina Schieferdecker. 2020. Responsible Software Engineering. In *The Future of Software Quality Assurance*. Springer International Publishing, 137–146. doi:10.1007/978-3-030-29509-7_11

[30] Mojtaba Shahin, Waqar Hussain, Arif Nurwidyantoro, Harsha Perera, Rifat Shams, John Grundy, and Jon Whittle. 2022. Operationalizing Human Values in Software Engineering: A Survey. *IEEE Access* 10 (2022), 75269–75295. doi:10.1109/ACCESS.2022.3190975

[31] Md Shamsujjoha, Qinghua Lu, Dehai Zhao, and Liming Zhu. 2025. Swiss Cheese Model for AI Safety: A Taxonomy and Reference Architecture for Multi-Layered Guardrails of Foundation Model Based Agents. In *2025 IEEE 22nd International Conference on Software Architecture (ICSA)*. 37–48. doi:10.1109/ICSA65012.2025.00014

[32] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Thirty-Seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=vAElhFcKW6

[33] Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell L. Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, et al. 2024. Position: A Roadmap to Pluralistic Alignment. In *Forty-First International Conference on Machine Learning*. https://openreview.net/forum?id=gQpBnRHwxM

[34] Bernd Carsten Stahl and Damian Eke. 2024. The Ethics of ChatGPT – Exploring the Ethical Issues of an Emerging Technology. *International Journal of Information Management* 74 (2024), 102700. doi:10.1016/j.ijinfomgt.2023.102700

[35] Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2024. Cognitive Architectures for Language Agents. doi:10.48550/arXiv.2309.02427

[36] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. 2025. Multi-Agent Collaboration Mechanisms: A Survey of LLMs. doi:10.48550/arXiv.2501.06322

[37] Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, et al. 2026. Agentic Reasoning for Large Language Models. doi:10.48550/arXiv.2601.12538

[38] Jon Whittle, Maria Angela Ferrario, Will Simm, and Waqar Hussain. 2021. A Case for Human Values in Software Engineering. *IEEE Software* 38, 1 (2021), 106–113. doi:10.1109/MS.2019.2956701

[39] Faouzi El Yagoubi, Godwin Badu-Marfo, and Ranwa Al Mallah. 2026. AgentLeak: A Full-Stack Benchmark for Privacy Leakage in Multi-Agent LLM Systems. doi:10.48550/ARXIV.2602.11510

[40] Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Kun Wang, Qingsong Wen, et al. 2025. NetSafe: Exploring the Topological Safety of Multi-agent System. In *Findings of the Association for Computational Linguistics: ACL 2025*. ACL, 2905–2938. doi:10.18653/v1/2025.findings-acl.150

[41] Liming Zhu. 2023. Software Engineering as the Linchpin of Responsible AI. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 3–4. doi:10.1109/ICSE48619.2023.00012

[42] N. Zuber, J. Gogoll, S. Kacianka, J. Nida-Rümelin, and A. Pretschner. 2024. Value-Sensitive Software Design: Ethical Deliberation in Agile Development Processes. In *Introduction to Digital Humanism: A Textbook*. Springer Nature Switzerland, 339–358. doi:10.1007/978-3-031-45304-5_22

Received 2026-07-14; accepted 2026-08-20