Facilitating Trustworthy Human-Agent Collaboration in LLM-based Multi-Agent System oriented Software Engineering

Krishna Ronanki
krishna.ronanki@gu.se
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden

Abstract

Multi-agent autonomous systems (MAS) are better at addressing challenges that spans across multiple domains than singular autonomous agents. This holds true within the field of software engineering (SE) as well. The state-of-the-art research on MAS within SE focuses on integrating LLMs at the core of autonomous agents to create LLM-based multi-agent autonomous (LMA) systems. However, the introduction of LMA systems into SE brings a plethora of challenges. One of the major challenges is the strategic allocation of tasks between humans and the LMA system in a trustworthy manner. To address this challenge, a RACI-based framework is proposed in this work in progress article, along with implementation guidelines and an example implementation of the framework. The proposed framework can facilitate efficient collaboration, ensure accountability, and mitigate potential risks associated with LLM-driven automation while aligning with the Trustworthy AI guidelines. The future steps for this work delineating the planned empirical validation method are also presented.

Keywords

Large Language Models, LLM-Based Multi-Agent Systems, Software Engineering, Trustworthy AI, Human-Agent Collaboration, DevOps

ACM Reference Format:
Krishna Ronanki. 2025. Facilitating Trustworthy Human-Agent Collaboration in LLM-based Multi-Agent System oriented Software Engineering. In *Companion Proceedings of the 33rd ACM Symposium on the Foundations of Software Engineering (FSE '25), June 23–27, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3696630.3728717

1 Introduction

Software Engineering (SE) is an inherently collaborative field [18] and SE collaboration is often artefact-based [23]. The advantage of SE collaboration being an artefact-based process is that the artefact evaluation against the ground truth can be automated [3]. This, alongside the observed fact that software engineers are increasingly becoming more reliant on the support of automation with increasing complexity in software systems [24], gave rise to the field of automated software engineering (ASE).

ASE focuses on developing methods and tools to assist humans and improve productivity in various tasks such as requirements traceability management, design specification, test data generation, defect tracking, and cost estimation with a focus on generation, storage and management of task specific artefacts [20]. The field of ASE has evolved from the application of traditional rule based tools such as static code analysers (SonarQube and ESLint), test automation tools (JUnit and pytest), dependency management tools (Maven and Gradle), and CI/CD tools (Jenkins and GitHub Actions) that automate a specific SE task to the development of frameworks that employ large language model(LLM)-based multi-agent autonomous (LMA) systems to automate multiple tasks within the software development lifecyle (SDLC) (ChatDev [17], MetaGPT [6], and CAMEL [9]).

However, the introduction of general purpose foundation models such as LLMs into ASE brings a plethora of challenges. One of the major challenge is ensuring the implementation of LMA oriented SE aligns with the trustworthy AI guidelines and principles prescribed for the users of these technologies as per the EU AI Act (AIA) [13]. Trustworthiness has been a issue for ASE even before the advent of LMA oriented SE. Multiple studies have shown that trust plays a critical and influential role within ASE [2, 11, 12].

Nonetheless, anthropomorphic features like natural language interaction and human-like reasoning enhance user trust in autonomous systems [21]. LLMs, which are capable of maintain context throughout conversations, excel in these areas [1]. Structured methods like chain-of-thought (CoT) prompting can further enhance their reasoning abilities [22].

Even as the introduction of LMA systems within ASE raises the issue of trust, the unique characteristics of LLMs, if leveraged effectively, can be used as a means to increase the users' trust within the system. This however, does not solve all the pertinent issues of LMA oriented SE.

One of the most pressing challenges in realising LMA oriented SE is facilitating trustworthy human-agent collaboration. Specifically, the strategic allocation of tasks between humans and the LMA system to leverage their unique individual strengths in an efficient and trustworthy manner [5].

To addressed this challenge that hinders human-agent collaboration in LMA oriented SE, a RACI-based framework along with implementation guidelines were developed to help define the roles and responsibilities of the LMA system and humans within a SDLC. An example implementation of the proposed framework is also provided by specifying a RACI matrix for a hypothetical implementation of LMA oriented SE within a DevOps environment. We chose the DevOps framework for the example implementation as it offers a structured SDLC with standardised processes and roles, which is observed to maximise the competences of the resources involved by decreasing manual work and offer more innovation capabilities [4], all of which are crucial for the realisation of LMA oriented SE.

Section 2 provides relevant background and related works that contributed in the problem formulation and developing the solution space. Sections 3 present the framework and its implementation guidelines, and the example implementation of the framework in a DevOps environment. The article concludes within Section 4 by outlining how this work in progress will be continued further.

2 Background

In this section, a brief overview of the concepts and state-of-the-art literature relevant to this article along with with nature of mentioned literature's relevance to this study are presented.

2.1 LLM-based Multi-Agent (LMA) Systems

When adopting an agent-oriented view of the world, it soon becomes apparent that most problems require or involve multiple agents [8]. This hold true for LLM-based agents as well. LMA systems are better capable of addressing real world challenges that are often spread across multiple domains and require expertise from different areas compared to singular LLM-based agents [5], including within SE [6]. The LMA system leverages the capabilities of multiple specialised agents, each with distinct skills and roles. These agents collaborate seamlessly, working together to plan and execute tasks to achieve a shared objective [5].

2.2 Trustworthy AI Requirements

The Ethics Guidelines developed by European Commission's (EC) High-level expert group on artificial intelligence (AI HLEG) have identified four ethical principles that form the basis for trustworthy AI which include *Respect for Human Autonomy*, *Prevention of Harm*, *Fairness*, and *Explicability* [14, 15]. But in order to achieve trustworthty AI in practice, they believe that there are seven key requirements that must be continuously assessed and managed throughout the entire lifecycle of an AI system: *Human Agency and Oversight*, *Technical Robustness and Safety*, *Privacy and Data Governance*, *Transparency*, *Diversity, Non-discrimination, and Fairness*, *Societal and Environmental Well-being*, and *Accountability*. The Trustworthy AI requirements are used as a reference within the proposed framework to ensure the human-LMA system collaboration is trustworthy.

2.3 RACI Matrix

The RACI matrix (Responsible, Accountable, Consulted, Informed) is a framework for defining roles and responsibilities within processes. It clarifies who is responsible for performing tasks (R), who is accountable for the outcome (A), who needs to be consulted to provide input (C), and who needs to be informed (I) [7]. In a LMA system, the RACI matrix can help in assigning clear roles between humans and the LLM-agents.

2.4 Related Work

He et al. [5] conducted a systematic review of recent primary studies to map the current landscape of LMA applications across various stages of the software development lifecycle (SDLC). They identify critical research gaps and propose a comprehensive research agenda focusing on, among other things, optimising agent synergy and trustworthiness within an LMA system.

Lin et al. [10] introduce LCG, a LLM-based code generation framework inspired by established SE practices like waterfall, scrum, and test driven development(TDD). LCG uses multiple LLM agents to emulate software process models with roles such as requirement engineer, developer, and tester to collaboratively enhance code quality. Their results underscore the importance of adopting software process models to bolster the quality and consistency of LLM-generated code. This work inspired us to select the DevOps framework within the example implementation of the framework.

Zhang et al. [25] developed an Autonomous LLM-based Agent System (ALAS) and evaluated its impact on user story quality. As part of their experimental study, they created two LLM-agent profiles: agent product owner (PO), which is responsible for managing product backlog and prioritising user stories based on business value and customer needs and agent requirements engineer (RE), which concentrates on the quality of user stories. The agent profiles were designed to reflect the actual functions of POs and REs in agile teams. We based the characteristics of the predefined heterogenous LLM-agents [5, 6] within the example implementation of the framework based on agent PO and agent RE.

3 Responsibility Assignment Framework

In this section, the proposed framework is presented for assigning responsibilities for the humans and the LLM-based agents of the LMA system. The definitions of each of the responsibility assignments are adapted to LMA oriented SE based on the original definitions of each of the four responsibility assignments of the standard RACI matrix.

- **R-Responsible**: The actor(s) assigned this responsibility do the work to complete the task.
- **A-Accountable**: The actor(s) assigned this responsibility delegate the task to the *responsible* actors and are responsible for the validation of the outcomes before the task is marked as complete.
- **C-Consulted**: The actor(s) assigned this responsibility provides inputs to guide the *responsible* actor. The *responsible* actor and the *accountable* actor, if human, exercise their best judgement to decide how to use the *consulted* actors' inputs. If the *responsible* actor is an LLM-agent, then it should take the *consulted* actor's input into account.
- **I-Informed**: The actor(s) assigned this responsibility need to be kept in the loop of the task execution without needing any explicit actions from the actor's end for the execution of the task.

3.1 Framework Implementation Guidelines

- **Step-1**: List all the tasks involved in the process where the LMA system will be implemented. Ensure all the tasks are artefact-based.
- **Step-2**: List all the human actor(s) and LLM-agent(s) required for the completion of the task.
- **Step-3**: List all the applicable regulatory and compliance constraints on the actors involved.

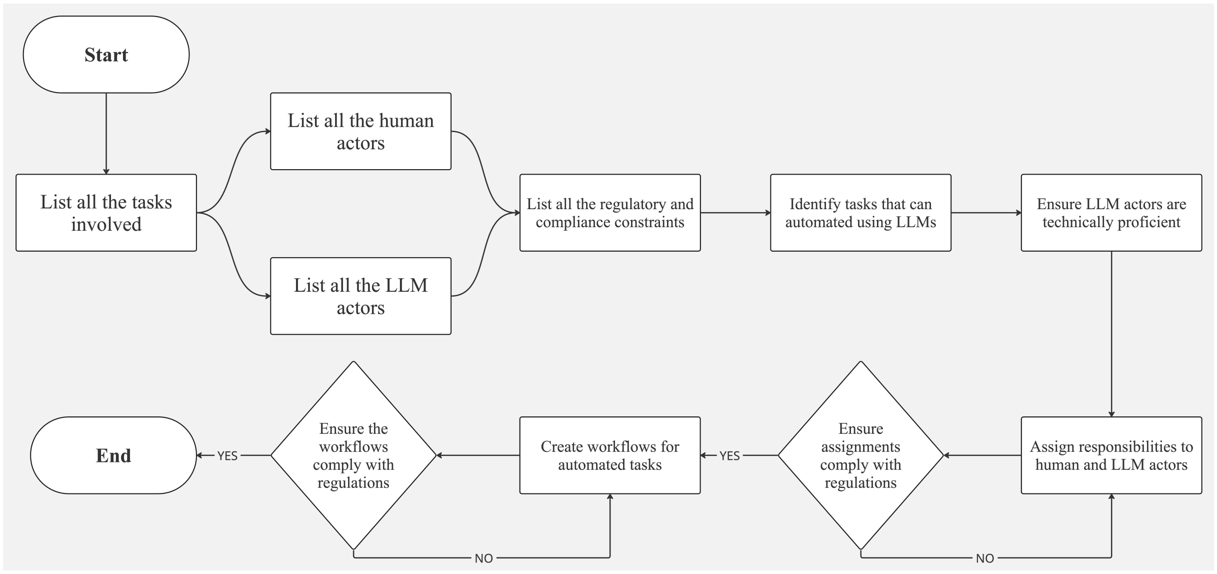
Start
List all the tasks involved
List all the human actors
List all the LLM actors
List all the regulatory and compliance constraints
Identify tasks that can automated using LLMs
Ensure LLM actors are technically proficient
Assign responsibilities to human and LLM actors
Ensure assignments comply with regulations
YES
NO
Create workflows for automated tasks
Ensure the workflows comply with regulations
YES
NO
End

Figure 1: Responsibility Assignment Framework

- **Step-4**: Identify the tasks that can be automated using LLM-agents.
- **Step-5**: Ensure the LLM-agent(s) selected are technically proficient in automating the identified task(s).
- **Step-6**: Assign each actor across the row an R, A, C, or I to indicate the role they'll play within the workflow to mark them as complete.
- **Step-7**: Ensure the assignments from **Step-6** are in do not conflict with the regulatory and compliance constraints. If yes, return to step **Step-5** and reassign the responsibilities to eliminate conflicts.
- **Step-8**: Create workflows for executing the automated tasks based on the responsibility assignments.
- **Step-9**: Ensure the created workflows do not conflict with the regulatory and compliance constraints. If yes, return to **Step-8** and redesign the workflows to eliminate conflicts.

This framework can be implemented to assign responsibilities for the human and LLM-agent actors for a single task or a suite of tasks within a SDLC phase or even an entire SDLC.

3.1.1 *Constraints of the Framework.*

(1) There should be at least one '*Responsible*' assignment for each task.
(2) There should be at least one '*Accountable*' assignment for each task. The only exception is when there's no influence of the LMA system on the output of the task, i.e, the assignment of LLM-agent(s) is either '*Informed*' or no assignment and the assignment of human(s) is '*Responsible*' and/or '*Accountable*'.
(3) There should be no '*Responsible*' assignment for the LMA system or any of its agents if there's no '*Accountable*' assignment for at least one human actor within a task.

3.2 Example Implementation

In this example implementation of the proposed framework, the **Planning** phase of the DevOps SDLC is considered. Table 1 provides the responsibility assignments for the human(s) and LLM-based agents for the collaborative execution of all the tasks within the planning phase.

- **Step-1**: *List all the tasks involved in the planning phase of the DevOps SDLC.* A typical planning phase within the DevOps SDLC has six tasks: requirements elicitation, creating product roadmap, creating features and user stories, creating product backlog, performing sprint planning and task allocation.
- **Step-2**: *List all the human actor(s) and LLM-agent(s) required for the completion of the task.* In this scenario, we consider three human actors: product owner, business analyst, & scrum master and three LLM-agents: LLM-agent A modelled after agent PO, LLM-agent B modelled after agent RE and LLM-agent C, a scrum implementation assistant, i.e, an LLM that has been fine-tuned for helping users implement scrum in practice[1].
- **Step-3**: *List all the applicable regulatory and compliance constraints on the actors involved.* Since the agents are based on LLMs, referring to the AIA, which is the most comprehensive in terms of coverage of applicable AI principles [19], is beneficial. If the LLM-agents are developed in-house from scratch, then the AIA's requirements for high-risk AI systems and GPAI requirements are applicable. If the LLM-agents include a third party developed models such OpenAI's GPT models for example, then the AIA's requirements for deployers of high-risk AI systems and GPAI models are applicable [15].

[1]LLM-agent C is hypothetical agent that does not exist as of yet to the best of authors' knowledge. The existence of such an agent is an assumption made by the authors to provide the example implementation of the proposed framework

- **Step-4**: *Identify the tasks that can be automated using LLM-agents.*
  1. Since the requirements for a software system should come from the customer, the requirements elicitation task should not be automated. However, they can use the LLM-agents' assistance with the requirements elicitation.
  2. Creating a product roadmap is a collaborative process involving multiple stakeholders to ensure the alignment of the product vision, establish collaboration, and aid risk management, which makes it not suitable to be completely automated using LLM-agents.
  3. Creating features and user stories is one of the tasks is suitable for automating using LLM-agent B, which is based on agent RE.
  4. Creating product backlog is next the task that is suitable for automating using LLM-agent A, which is based on agent PO.
  5. Sprint planning and task allocation, which is are typically performed by the scrum master are also suitable as they are artefact-based processes.
- **Step-5**: *Ensure the LLM-agent(s) selected are technically proficient in automating the identified task(s).* Since LLM-agent B is based on agent RE, it is capable of creating features and user stories. Similarly, as LLM-agent A is based on agent PO, it has the technical proficiency and domain knowledge required to create a product backlog. Since, LLM-agent C is fine-tuned for helping users implement scrum in practice, it has the technical proficiency and domain knowledge required to perform sprint planning and task allocation.
- **Step-6**: *Assign each actor across the row an R, A, C, or I to indicate the role they'll play within the workflow to mark them as complete.*
  1. For reasons described in **step-4**, the *responsible* actor for requirements elicitation task is the business analyst and *accountable* is the product owner. The LLM-agent B and LLM-agent C can be the *consulted* actors to leverage their domain knowledge while the scrum master and LLM-agent A can be the *informed actors* as keeping them in the loop will be helpful with providing them with the overall context of LMA system being implemented.
  2. Similarly, for creating the product roadmap task, the *responsible* actor is the product owner. The LLM-agent A and LLM-agent C are *consulted* actors while the scrum master and business analyst are the *informed* actors. There is no *accountable* assignment in this scenario as per the second framework constraint. The *responsible* actor is the *accountable* actor as well.
  3. For creating features and user stories, as described in **step-4**, the *responsible* actor will be LLM-agent B while the business analyst who will use the LLM to generate the user stories and features will be the *accountable* actor. LLM-agent A can be helpful in this scenario so it can be the *consulted* actor and rest can be either *informed* or no assignment based on the necessity.
  4. Similarly, for creating the product backlog task, the *responsible* actor will be LLM-agent A while the product owner who will use the LLM will the *accountable* actor. As it is valuable to keep the business analyst and scrum master in loop to enhance the collaboration, they can be either *consulted* or *informed*, based on the necessity while the rest of the actors can remain unassigned for this task.
  5. For sprint planning and task allocation, the scrum master who is also the *accountable* actor, can use LLM-agent C, which is the *responsible* actor, to generate the required outputs to complete the task. The product owner can be the *informed* actor to keep them in the loop.
- **Step-7**: *Ensure the assignments from **Step-6** are in do not conflict with the regulatory and compliance constraints. If yes, return to step **Step-5** and reassign the responsibilities to eliminate conflicts.* The AIA's obligations and requirements do not apply to the tasks where LLM's responsibility is just informed. Since there is a human actor who is assigned the *accountable* responsibility whenever there is either a *responsible* or an *informed* responsibility for the LLM-agent, there is no conflict with the AIA's obligations and requirements.
- **Step-8**: *Create workflows for executing the automated tasks based on the responsibility assignments.*
  1. Although the requirements elicitation and creating product roadmap tasks have been identified as not suitable for automation, LLM-agents can still be leveraged to assist the human actors in these tasks. For example, the business analysts within the requirements elicitation task can use the LLM-agent B and LLM-agent C to create the requirements elicitation questionnaire or planning the elicitation meetings. Similarly, the product owner in the creating product roadmap task can leverage LLM-agent A and LLM-agent C's assistance.
  2. For the remaining four tasks, the workflows can be designed in such a way that the final output of the user interaction with the LLM-agents will yield a verifiable artefact. For example, the features & user stories will be generated using the LLM-agent based on the requirements elicited by the business analyst from the clients. The business analyst needs to verify the outputs before the product backlog can be generated by the LLM-agent based on the list of features and user stories generated. This time, the product owner needs to verify the product backlog before the sprint planning and task allocation can be performed. The outputs of the LLM-based sprint planning and taks allocation needs to verified by the scrum master before they can implemented in practice.
- **Step-9**: *Ensure the created workflows do not conflict with the regulatory constraints. If yes, return to **Step-8** and redesign the workflows to eliminate conflicts.* Since, the first two tasks don't involve the use of LLMs in the workflow, the AIA's requirements do not apply. For the next four tasks, since the technical capabilities of the LLM-agents has been ensured within **step-5** and human oversight and validation mechanisms were ensured within **step-8**, they do not conflict with the AIA's requirements for deployers.

The RACI matrix for the example implementation of the proposed framework based on above description is provided in Table 1.

Task	Product Owner	Business Analyst	Scrum Master	LLM-agent A	LLM-agent B	LLM-agent C
Requirements elicitation	A	R	I	I	C	C
Create product roadmap	R	I	I	C	-	C
Create features & user stories	I	A	I	C	R	-
Create product backlog	A	I/C	I	R	-	-
Sprint planning	I	-	A	C	-	R
Task allocation	I	-	A	C	-	R

Table 1: RACI matrix for the example implementation of the proposed framework

This is, however, an example implementation of the proposed framework using a hypothetical example. The framework allows practitioners to customise and adapt the output of RACI matrix based on the the SDLC framework and the standard operating procedures (SOP) specific to their organisation in a flexible manner.

4 Conclusion and Future Work

The proposed framework leverages the RACI matrix to define the roles and responsibilities of humans and LLM-based agents in the LMA system. This structured role delineation enhances collaboration, ensures accountability, and mitigates potential risks associated with LLM-driven automation. By systematically distributing decision-making authority and oversight, the framework can aid in optimising the efficiency of the LMA system based SE workflows while maintaining human control over critical software development activities, aligning with the Trustworthy AI guidelines.

Despite its theoretical promise, the framework has not yet undergone empirical validation. Moving forward, a groupware walkthrough [16] based multi case study will be conducted to collect empirical validation data from **experts/potential users**.

References

[1] Malak Abdullah, Alia Madain, and Yaser Jararweh. 2022. ChatGPT: Fundamentals, Applications and Social Impacts. In *2022 Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. 1–8. doi:10.1109/SNAMS58071.2022.10062688

[2] Maria Christakis and Christian Bird. 2016. What Developers Want and Need From Program Analysis: An Empirical Study. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering* (Singapore, Singapore) *(ASE '16)*. 332–343. doi:10.1145/2970276.2970347

[3] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2023. Large Language Models for Software Engineering: Survey and Open Problems. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. 31–53. doi:10.1109/ICSE-FoSE59343.2023.00008

[4] João Faustino, Daniel Adriano, Ricardo Amaro, Rubén Pereira, and Miguel Mira da Silva. 2022. DevOps Benefits: A Systematic Literature Review. *Software: Practice and Experience* 52, 9 (2022), 1905–1926. doi:10.1002/spe.3096 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3096

[5] Junda He, Christoph Treude, and David Lo. 2025. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead. *ACM Trans. Softw. Eng. Methodol.* (2025). doi:10.1145/3712003

[6] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. arXiv:2308.00352 [cs.AI] https://arxiv.org/abs/2308.00352

[7] Project Management Institute. 2000. A Guide to the Project Management Body of Knowledge (PMBOK Guide). Project Management Institute.

[8] Nicholas R. Jennings. 1999. Agent-Oriented Software Engineering. In *Multi-Agent System Engineering*, Francisco J. Garijo and Magnus Boman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–7.

[9] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 51991–52008. https://proceedings.neurips.cc/paper_files/paper/2023/file/a3621ee907def47c1b952ade25c67698-Paper-Conference.pdf

[10] Feng Lin, Dong Jae Kim, Tse-Husn, and Chen. 2024. When LLM-based Code Generation Meets the Software Development Process. arXiv:2403.15852 [cs.SE] https://arxiv.org/abs/2403.15852

[11] Emerson Murphy-Hill, Chris Parnin, and Andrew P. Black. 2012. How We Refactor, and How We Know It. *IEEE Transactions on Software Engineering* 38, 1 (2012), 5–18. doi:10.1109/TSE.2011.41

[12] David J. Niedober, Nhut T. Ho, Gina Masequesmay, Kolina Koltai, Mark Skoog, Artemio Cacanindin, Walter Johnson, and Joseph B. Lyons. 2014. Influence of Cultural, Organizational and Automation Factors on Human-Automation Trust: A Case Study of Auto-GCAS Engineers and Developmental History. In *Human-Computer Interaction. Applications and Services*, Masaaki Kurosu (Ed.). Springer International Publishing, Cham, 473–484.

[13] Future of Life Institute. 2024. High-level Summary of the AI Act. https://artificialintelligenceact.eu/high-level-summary/

[14] High-Level Expert Group on Artificial Intelligence. 2019. *Ethics Guidelines for Trustworthy Artificial Intelligence (AI)*. www.aepd.es/sites/default/files/2019-12/ai-ethics-guidelines.pdf

[15] European Parliament. 2024. Artificial Intelligence Act. https://www.europarl.europa.eu/doceo/document/TA-9-2024-0138_EN.pdf

[16] David Pinelle and Carl Gutwin. 2002. Groupware Walkthrough: Adding Context to Groupware Usability Evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, USA) *(CHI '02)*. 455–462. doi:10.1145/503376.503458

[17] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Bangkok, Thailand, 15174–15186. doi:10.18653/v1/2024.acl-long.810

[18] Gema Rodríguez-Pérez, Reza Nadri, and Meiyappan Nagappan. 2021. Perceived Diversity in Software Engineering: A Systematic Literature Review. *Empirical Software Engineering* 26 (2021), 1–38.

[19] Krishna Ronanki, Beatriz Cabrero-Daniel, Jennifer Horkoff, and Christian Berger. 2023. RE-centric Recommendations for the Development of Trustworthy(er) Autonomous Systems. In *Proceedings of the First International Symposium on Trustworthy Autonomous Systems* (Edinburgh, United Kingdom) *(TAS '23)*. Article 1, 8 pages. doi:10.1145/3597512.3599697

[20] Simin Wang, Liguo Huang, Amiao Gao, Jidong Ge, Tengfei Zhang, Haitao Feng, Ishna Satyarth, Ming Li, He Zhang, and Vincent Ng. 2023. Machine/Deep Learning for Software Engineering: A Systematic Literature Review. *IEEE Transactions on Software Engineering* 49, 3 (2023), 1188–1231. doi:10.1109/TSE.2022.3173346

[21] Adam Waytz, Joy Heafner, and Nicholas Epley. 2014. The Mind in the Machine: Anthropomorphism Increases Trust in an Autonomous Vehicle. *Journal of Experimental Social Psychology* 52 (2014), 113–117. doi:10.1016/j.jesp.2014.01.005

[22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 24824–24837. https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

[23] Jim Whitehead. 2007. Collaboration in Software Engineering: A Roadmap. In *Future of Software Engineering (FOSE '07)*. 214–225. doi:10.1109/FOSE.2007.4

[24] David Gray Widder, Laura Dabbish, James D. Herbsleb, Alexandra Holloway, and Scott Davidoff. 2021. Trust in Collaborative Automation in High Stakes Software Engineering Work: A Case Study at NASA. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Article 184, 13 pages. doi:10.1145/3411764.3445650

[25] Zheying Zhang, Maruf Rayhan, Tomas Herda, Manuel Goisauf, and Pekka Abrahamsson. 2024. LLM-Based Agents for Automating the Enhancement of User Story Quality: An Early Report. In *Agile Processes in Software Engineering and Extreme Programming*. Springer Nature Switzerland, Cham, 117–126.