From Hazard Identification to Controller Design: Proactive and LLM-Supported Safety Engineering for ML-Powered Systems

Yining Hong
Carnegie Mellon University

Christopher S. Timperley
Carnegie Mellon University

Christian Kästner
Carnegie Mellon University

***Abstract*—Machine learning (ML) components are increasingly integrated into software products, yet their complexity and inherent uncertainty often lead to unintended and hazardous consequences, both for individuals and society at large. Despite these risks, practitioners seldom adopt proactive approaches to anticipate and mitigate hazards before they occur. Traditional safety engineering approaches, such as Failure Mode and Effects Analysis (FMEA) and System Theoretic Process Analysis (STPA), offer systematic frameworks for early risk identification but are rarely adopted. This position paper advocates for integrating hazard analysis into the development of any ML-powered software product and calls for greater support to make this process accessible to developers. By using large language models (LLMs) to partially automate a modified STPA process with human oversight at critical steps, we expect to address two key challenges: the heavy dependency on highly experienced safety engineering experts, and the time-consuming, labor-intensive nature of traditional hazard analysis, which often impedes its integration into real-world development workflows. We illustrate our approach with a running example, demonstrating that many seemingly unanticipated issues can, in fact, be anticipated.**

***Index Terms*—safety engineering, hazard analysis, software engineering for machine learning**

I. Introduction

Phrases like *"this was an unintended consequence"* or *"nobody could have anticipated this new problem"* often arise when software products face issues, such as mistakes or biases in ML models within the software. For example, applications may unintentionally amplify biases [1] or introduce privacy risks [2]. However, *unintended consequences* merely implies that these issues were not anticipated, not that they were inherently unpredictable; had they been anticipated, developers could, in many cases, have mitigated them before they occurred. This position paper argues that (1) systematic *hazard analysis* methods from safety engineering are well suited to anticipate a wide range of harms in ML-powered applications beyond traditional safety risks, and (2) LLM-supported automation can make hazard analysis more accessible and manageable in terms of effort. We believe that hazard analysis is particularly effective in the early development stages, helping identify and design controllers to mitigate harm.

Recent research has explored more or less structured strategies to anticipate potential harms, primarily bias and fairness issues, in the context of ML impact assessment [3]–[7]. However, these approaches are generally model-centric and applied ad-hoc [8], overlooking controllers beyond the model such as safeguards, trend monitoring, human oversight, and user interface modifications – areas where software engineers can contribute to responsible engineering of ML-powered systems, extending beyond model-focused considerations.

More recent research has adopted traditional safety engineering, especially hazard analysis methods such as *Failure Mode and Effects Analysis (FMEA)* and *System Theoretic Process Analysis (STPA)*, to proactively identify a broad range of ethical and social risks in ML models and ML-powered software systems [8]–[15]. However, outside traditional safety-critical domains such as autonomous vehicles [16], [17], existing studies largely discuss only the *potential* application of hazard analysis, with several emerging challenges: First, traditional hazard analysis methods are limited by predefined system boundaries, restricting the consideration of a broader scope of risks and solutions. Second, these methods require substantial time and effort, making them costly and challenging to integrate into fast-paced, continuous development workflows outside traditional safety-critical systems [11], [12]. Third, the responsibility for identifying and mitigating risks often falls to software developers or ML model creators, who may lack expertise in safety engineering, potentially reducing the effectiveness and thoroughness of these analyses [11].

In this paper, we provide a concrete illustration of how hazard analysis, with minor modifications, can proactively anticipate harms and, more importantly, guide the design of controllers to mitigate those harms before they occur at both the model and system levels. Furthermore, we show how LLMs can support this process by offering guidance to software engineers and data scientists with limited safety-engineering expertise, requiring only moderate efforts that align with the fast-paced development practices of ML-powered applications.

In summary, this position paper offers a fresh and critical perspective on using hazard analysis to broadly anticipate harms in ML-powered applications and to design system-level controllers, illustrated with a concrete example. We further contribute a discussion and demonstration of the potential of how LLMs can support developers in the hazard analysis process. We envision that a lightweight, LLM-supported hazard analysis process will become a routine step in the responsible engineering of ML-powered applications, enabling the mitigation of many harms well before they occur.

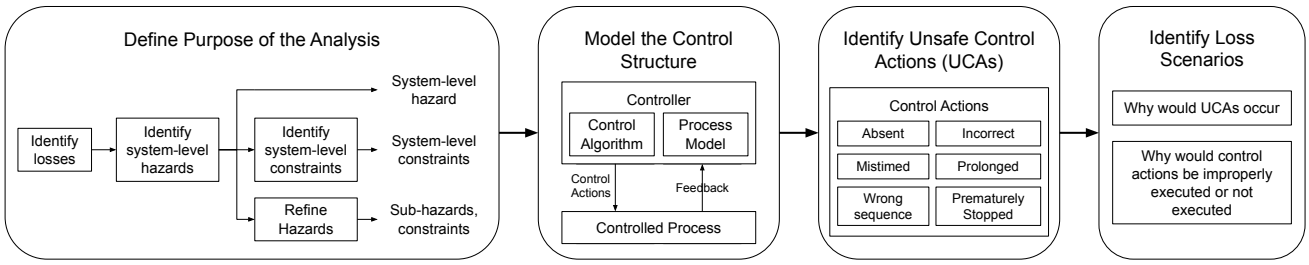

Fig. 1. An overview of the STPA method, based on the textual description and figures in the STPA handbook [18].

II. Scope and Related Work

Though there are various definitions for AI and ML systems, in this paper, we use the term "ML-powered system" to denote any software system that incorporates ML models as a component, aiming to emphasize the system as a whole with ML model as an inherently 'unreliable' module within it.

Practitioners and researchers have acknowledged the potential social risks posed by ML-powered systems [19], [20], such as bias and insufficient oversight. Technical approaches typically focus on measuring and addressing model-level issues, focusing on aspects such as fairness, shortcut reasoning, or privacy [1], [2], [21], [22], often neglecting broader system or environmental considerations. Recent research has introduced structured approaches and tools to help practitioners anticipate potential harms of ML-powered systems, ranging from impact assessment templates [23] to tools that facilitate more or less structured brainstroming [5], [6], [24], [25]. While these methods effectively identify relatively explicit harms or hazards linked to the system's overarching purpose, they fall short in analyzing the detailed components or control structures within these systems and are often used in an ad-hoc manner [11].

Safety engineering has a long history [26], with traditional methods having been applied in various safety-critical domains, such as aviation systems [27], [28] and autonomous vehicles [16], [17]. Recent research has applied these safety engineering methods, particularly the hazard analysis frameworks FMEA [13], [15] and STPA [8], [12]–[14], to identify potential hazards and assess control structures in ML-powered systems. Findings suggest that such systematic approaches are not only effective in identifying a wider spectrum of hazards but are also potentially applicable throughout all development stages. However, integrating safety engineering frameworks into the development cycles of ML systems is often impractical due to the extensive time and paperwork required [11], [12]. Many practitioners lack in-depth safety engineering expertise, which can reduce the effectiveness of these frameworks [11]. Furthermore, most existing studies still focus on the model and its development process, with limited attention to non-ML components and the social environment. This paper demonstrates that hazard analysis is effective in anticipating a wide range of hazards and aids in proactive controller design. Furthermore, it shows that LLMs are promising in supporting humans conducting STPA by reducing effort and encouraging broader, more comprehensive thinking beyond the model.

III. A Brief Introduction to STPA

While safety engineering techniques date back to the 1950s [29], [30], our approach builds on System-Theoretic Process Analysis (STPA), a state-of-the-art hazard analysis framework grounded in the System-Theoretic Accident Model and Processes (STAMP) [31]. We use STPA because it is designed for analyzing highly complex systems, can be applied from the early development stages, and considers both technical and human factors [31] – all crucial for ML-powered software.

Traditionally, STPA follows four steps, outlined in Figure 1:

1) *Define the Purpose of the Analysis:* We list *stakeholders* and their values, then identify important **losses** (e.g., loss of life). Based on the losses, we determine system-level **hazards**, which are conditions that may lead to a loss (e.g., aircraft too close to other objects). Hazards are then inverted into **constraints**, which are system conditions that need to be satisfied to prevent hazards. This way, STPA anticipates harms and establishes safety requirements to mitigate them.

2) *Model the Control Structure:* We outline the control structure designed to ensure the safety constraints are met. Each previously identified constraint targeted for resolution should have at least one associated **controller** (e.g., an automated safeguard or a human supervisor). New controllers can be envisioned for constraints without sufficient controllers. STPA particularly focuses on feedback-control loops between controllers and the controlled processes.

3) *Identify **Unsafe Control Actions**:* We analyze each controller's potential failure modes by considering whether a hazard may arise if each control action is absent, incorrect, mistimed, executed in the wrong sequence, prolonged, or stopped prematurely. Control action errors that can lead to unsafe outcomes are then analyzed in the final step.

4) *Identify Loss Scenarios:* We examine why unsafe control actions may occur, potentially leading to revising existing controllers or introducing additional controllers (e.g., pilots may be unreliable in distance checking, hence we may introduce an automated warning system). The STPA process is then repeated with the modified control structure.

In a nutshell, STPA is a structured approach that works backward from harms (losses) to safety requirements (constraints) and then to mitigation strategies (controllers) and their potential failures. It encourages broad consideration of the control structure, including non-technical controllers like training, human oversight, and government regulations.

TABLE I
TERMINOLOGY OF SAFETY ENGINEERING AND MAPPING TO CONCEPTS IN ML-POWERED APPLICATION DEVELOPMENT

Concept	Explanation and mapping to ML-powered application
Loss	An event undesirable to a stakeholder, including bodily harm, allocation harms (e.g., discriminatory service), representation harms, and societal harms (e.g., polarization)
Hazard	A state or a set of conditions that may lead to a loss in certain scenarios, such as when an ML-powered software system produces faulty information
Constraint	A (safety) requirement that, when satisfied, prevents the hazard and thus the loss (e.g., an ML-powered system does not produce faulty information or warns the user if it does)
Controller	A technical or non-technical components that enforce constraints, including technical safeguards (e.g., exception handling, redundant sensors, I/O validation) and human oversight (e.g., processes, training, monitoring, alerting)
Unsafe Control Action	A control action, or lack thereof, provided by a controller can, under certain conditions, lead to a hazard (e.g., output validation suppressing correct answers)

Throughout the paper, we use safety engineering terminology for illustrations and discussions. Since this may be unfamiliar to most software engineers and data scientists, a summary and mapping are provided in Table I.

STPA is a labor-intensive process needing expert involvement and is mainly used for safety-critical systems like aviation, focusing on severe losses such as loss of life. Researchers found STPA potentially useful, but its time-consuming nature is unsuitable for rapid development, and developers often lack the necessary expertise [11]–[13].

IV. LLM-SUPPORTED HAZARD ANALYSIS TO ANTICIPATE HARMS IN ML-POWERED APPLICATIONS

We argue that hazard analysis is effective in anticipating a wide range of harms in ML-powered applications, such as fairness, usability issues, and societal concerns like deskilling and polarization, extending beyond the traditional focus on severe harms such as loss of life in safety-critical systems. Specifically, hazard analysis aids in *anticipating* problems and *designing* mitigation strategies in systems that might otherwise be released without any. In this section, we walk through the four STPA steps using a running example of a simple web system designed to recommend outdoor trails.

Evidence suggests that routinely applying hazard analysis to everyday software applications as part of responsible engineering practices is challenging due to high costs and skill requirements. Therefore, we also present how LLMs, specifically GPT-4o by OpenAI [32], can assist software engineers and data scientists in conducting such analysis without requiring extensive training and excessive paperwork.

Our running example is inspired by the existing customized GPT assistant *AllTrails* [33]. The application operates within the ChatGPT chat interface and is guided by system prompts from developers. It accesses the predefined alltrails.com API to fetch trail information. Upon receiving user prompts (e.g., "Recommend trails near [district name] that are dog-friendly."), it calls the API for information and provides a response. To our knowledge, it lacks additional controllers beyond its system prompt instructions and ChatGPT's built-in safeguards. As is often the case, the LLM should be considered an unreliable component, as it may fail to follow instructions, hallucinate, or produce factual errors. We assume this application will be extended beyond its current form and expect it to gain significant popularity. Note that we explicitly chose a running example of an everyday ML-powered application, rather than a traditional safety-critical system, to illustrate how hazard analysis can proactively anticipate issues worth mitigating, even for such seemingly harmless applications.

TABLE II
VALUES AND LOSSES AMONG DIFFERENT STAKEHOLDERS (SELECTED)

Stakeholder	Value	Loss
End User	Personalization Accuracy of information	Lack of personalization Inaccuracy of information
App Developers	Maintainability Cost efficiency	High maintenance burden Increased costs
Local Businesses	Community engagement	Lack of community engagement

A. Anticipating Losses and Hazards

Stakeholders: Following traditional STPA, we begin with identifying stakeholders and their values to assess system losses. To go beyond identifying only the most severe losses, we encourage a comprehensive exploration of stakeholders, including indirect ones, which extends the scope typically practiced in STPA. In our running example, we identified a list of 20 potential stakeholders. This includes directly involved entities such as end users, app developers, and API providers, as well as indirectly affected ones like trail management organizations and local businesses.[1] Our experiments show that LLMs can generate a list of stakeholders based on an application description. We have designed prompts to encourage LLMs to consider indirect and less common stakeholders, thereby fostering a broad exploration. Developers can then use this list as inspiration for selecting stakeholders to analyze.

Losses: For each stakeholder, we consider their values and convert them into corresponding losses. In our running example, we identified 145 losses, averaging 6-8 losses per stakeholder; a selection of these is shown in Table II. Again, we found LLMs helpful for suggesting values and losses, given the broad scope and the large number of losses they identified.

Hazards and Constraints: For each loss, we identify hazards – states or conditions that can lead to the loss under certain circumstances. We consider hazards beyond traditional STPA system boundaries. For example, the hazard "system does not support user customization" may lead to the loss "lack of personalization". Given a large number of losses and to avoid prematurely focusing on only a few, we find that LLMs

[1]For the complete list of prompts and results in all steps, please refer to the supplementary material: https://docs.google.com/spreadsheets/d/1GkdN9TBscGhqfyFNdLMsLRtFKbr3DqqNzkaTDvW-XAQ/edit?usp=sharing

TABLE III
SYSTEM-LEVEL HAZARDS (SELECTED)

HID	Hazard
H4	System does not personalize recommendations effectively, support user customization, or adapt to user preferences and trends.
H24	System recommends trails in ecologically sensitive areas or during hazardous conditions.
H39	System lacks a centralized dashboard or knowledge repository for monitoring performance and user feedback.
H48	System lacks a mechanism for users to withdraw consent or access their data records.

HID: Hazard ID

TABLE IV
CONTROLLER DESIGNS (SELECTED)

HID	Controller Design
H4	Integrate a feedback loop where users can provide feedback on trail recommendations, allowing the system to improve personalization. Enable users to connect their social media or fitness tracking accounts for personalized recommendations.
H39	Develop a centralized dashboard that aggregates data from the LLM and external APIs to monitor system performance in real time. Introduce automated alerts in the dashboard to notify administrators of any performance issues or anomalies detected in the system.
H48	Implement a user-friendly interface that allows users to easily withdraw consent for data collection and processing at any time. Design the system to automatically notify users of any changes to data handling practices and obtain renewed consent if necessary.

HID: Hazard ID

effectively support analyzing all losses in a few minutes for under USD 2. In our running example, we identified 1,159 hazards for 145 losses across 20 stakeholders, averaging 5-10 hazards per loss. While some hazards are obvious, such as incorrect model outputs, we discovered many others we would not have anticipated without such systematic analysis, such as *"System recommends trails in ecologically sensitive areas or during hazardous conditions."*

At this stage, LLMs also effectively aid in merging similar hazards identified from different losses. In our running example, with human supervision for granularity, we reduced the hazards to 50 distinct ones, as examples shown in Table III.

B. Proactive Design of Control Structure

Since the existing AllTrails application had few controllers, as is common in many ML-powered applications, there was not much to analyze regarding the existing control structures in STPA steps 2–4. However, the STPA process is also effective for *proactively envisioning new controllers* to mitigate the identified hazards, aiding practitioners in *modeling the control structure*, which is the second step in STPA.

We consider potential controllers or design structures that (a) prevent the hazard, (b) reduce its likelihood, (c) decrease the probability of it causing a loss, and (d) minimize the severity of the loss if it occurs. In our running example, aided by LLMs, we identified 10 to 15 possible designs per hazard. While some controllers may be impractical, costly, or insufficiently effective, our systematic approach revealed some easy-to-implement and potentially effective ideas we might not have otherwise considered, such as connecting social media or fitness tracking accounts for personalized recommendations. Among the identified hazards, we chose to explore controllers for three hazards: H4, H39, and H48. These pertain to the lack of personalization, absence of system monitors, and inability for users to withdraw consent, respectively. Some potential controllers are listed in Table IV. Having assessed the risks and costs, we adopted, modified, and merged several controller designs suggested by the LLMs. The resulting control structure incorporates human feedback and hierarchical monitoring mechanisms, which would be challenging to develop without this process. We then performed the second step of STPA and visually modeled the control structure of the system, as illustrated in Figure 2.

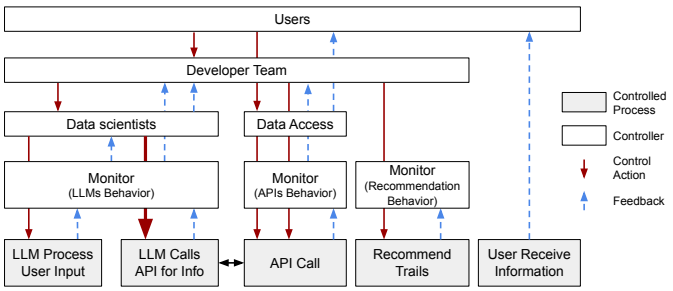

Fig. 2. The control structure of the system: The bold arrow represents the control action analyzed in Section IV-C, involving data scientists upgrading or downgrading the LLM version or altering the prompts.

Again, we found that LLMs can help consider potential designs, especially when given instructions to explore different designs. However, developers remain crucial in making judgments about relevance and importance.

C. Analyzing and Revising Controllers

After designing and modeling the control structure, we now return to steps 3 and 4 of the STPA process to analyze whether the controllers are sufficient or if they introduce new problems.

We identified *unsafe control actions* by examining each control action against a checklist for potential constraint violations if the action was (a) absent, (b) incorrect, (c) mistimed, (d) executed in the wrong sequence, (e) prolonged, or (f) stopped early. We then identify potential *loss scenarios*, which are states or events that may lead to the unsafe control action.

In our running example, we analyze the control action where data scientists modify the LLMs that call APIs, as depicted by the bold arrow in Figure 2. Some identified unsafe control actions and corresponding loss scenarios are listed in Table V. We found LLMs effective in systematically exploring different problems by following the checklist.

Practitioners can now review these unsafe control actions and loss scenarios to decide whether and how to revise certain controllers. For example, our initial monitoring approach was

TABLE V
UNSAFE CONTROL ACTIONS AND LOSS SCENARIOS (SELECTED)

Unsafe Control Actions	Loss Scenario
Premature control actions, like upgrading the model without thorough analysis or confirmation, can destabilize operations or reduce user satisfaction.	Data scientists might overly rely on new model version improvements without properly assessing their impact on system interactions with APIs.
Delayed control actions can cause inefficiency, poor performance, user frustration, and a negative experience.	Data scientists may face data overload, insufficient tools, or cognitive strain, hindering decision-making and delaying issues without clear prioritization.

found to be naive, lacking an explicit structure to ensure that an operator regularly checks the dashboard, effectively detects issues, and does not cease monitoring due to notification fatigue. This encourages establishing explicit procedures for (a) setting up automated alerts, (b) implementing an engineering process for tracking and eliminating false positive alerts, and (c) designating someone responsible for monthly check-ins with the operator to ensure accountability. This approach goes far beyond simple engineering interventions such as exception handling, redundancy, or using an LLM to review the output of another LLM [34], instead considering the entire sociotechnical system and its environment.

V. DISCUSSIONS AND CONCLUSION

While others have advocated adopting safety engineering to anticipate harms in ML-powered applications [8]–[15], this paper reinforces this idea and provides a concrete example of the process, demonstrating its potential to anticipate harms beyond traditional safety concerns like bodily harm and mission loss. Although our running example appears simple – merely combining a system prompt with an API on OpenAI's customized GPTs platform – the analysis identified potential harms ranging from minor and far-fetched to worth addressing. This encouraged comprehensive early system design to mitigate these harms. We advocate that developers of novel ML-powered applications should undertake this practice, even if the application appears harmless. There is no justification for not attempting to anticipate the "unintended consequences," given the uncertainty, potential bias, and possible shortcuts associated with ML components in software systems.

However, we also experienced firsthand how quickly the analysis can become unmanageable, especially when expanding beyond severe losses in traditional safety-critical systems. We analyzed dozens of stakeholders, hundreds of potential losses, and thousands of hazards. This scope can easily grow to include even more potential controllers and associated issues. Without technical support, this approach is overwhelming and can feel like a bureaucratic paper-heavy compliance activity, raising questions about how to encourage its routine adoption as part of everyday responsible engineering practices.

Assisting Developers vs. Automating Compliance Activity: LLMs effectively assist in navigating STPA steps and extensively exploring stakeholders, losses, hazards, and controllers, helping to scale the process beyond just the severe losses. However, there is a risk of over-reliance on LLMs for fully automating hazard analysis. We envision LLMs as support tools, not replacements for developers' critical judgment. While LLMs can produce ideas and explore broadly by following checklists, developers are essential for thinking beyond suggestions, assessing severity, and setting the analysis focus. Balancing LLM support while preserving human creativity and judgment is a crucial question for future research.

Expertise Requirements: Automated assistance in hazard analysis can enhance accessibility by offering step-by-step guidance and examples to developers lacking safety engineering training. However, there is a risk that these developers may perform a more superficial analysis. Ideally, increased accessibility enables selective engagement of safety experts, allowing their expertise to scale across more projects. Balancing this requires further research, potentially drawing insights from similar discussions in software security [35], [36] and democratizing data science [37], [38].

Cataloging Common Controllers: ML research and much software engineering research on ML primarily focus on the model – to improve accuracy, measure fairness, explain model internals, or rectify shortcuts. In contrast, system-level mitigations such as safeguards, user interface design, and human oversight receive less attention, despite their effectiveness as controllers [39]. Hazard analysis promotes broader system thinking, and we believe a pattern catalog of common system-level interventions could foster discussions and be embedded into the process of identifying and designing controllers.

Fostering Adoption: Interview studies indicate practitioners hesitate to integrate hazard analysis into their agile and exploratory workflows, citing the need for organizational changes such as increased incentives, managerial confidence, support, understanding, and resource investment [11], [13]. Further research is required to embed hazard analysis into routine engineering, even in non-safety-critical contexts where success is a lack of issues. Lowering effort and demonstrating value, while appealing to developers' responsibility and mastery, may effectively change practices and culture. While much work remains, we can learn from past efforts such as DevOps [40], establishing a quality or security culture [41], adopting fairness audits [42], promoting social responsibility through education [43], and broadly altering organizational culture [44]. Technological innovations, education, and activism can each promote more responsible engineering practices.

ACKNOWLEDGMENT

We thank Nadia Nahar, Laurie Williams, William Enck, and Alexandros Kapravelos for their feedback on this work.

References

[1] H. Suresh and J. Guttag, "A framework for understanding sources of harm throughout the machine learning life cycle," in *Proceedings of the 1st ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, 2021, pp. 1–9.

[2] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "When machine unlearning jeopardizes privacy," in *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, 2021, pp. 896–911.

[3] T. P. Pagano, R. B. Loureiro, F. V. N. Lisboa, G. O. R. Cruz, R. M. Peixoto, G. A. d. S. Guimarães, L. L. d. Santos, M. M. Araujo, M. Cruz, E. L. S. de Oliveira *et al.*, "Bias and unfairness in machine learning models: a systematic literature review," *arXiv preprint arXiv:2202.08176*, 2022.

[4] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM computing surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[5] Z. Buçinca, C. M. Pham, M. Jakesch, M. T. Ribeiro, A. Olteanu, and S. Amershi, "Aha!: Facilitating ai impact assessment by generating examples of harms," *arXiv preprint arXiv:2306.03280*, 2023.

[6] Z. J. Wang, C. Kulkarni, L. Wilcox, M. Terry, and M. Madaio, "Farsight: Fostering responsible ai awareness during ai application prototyping," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–40.

[7] Microsoft, "Responsible AI impact assessment template," 2022, accessed: 2024-11-07. [Online]. Available: https://blogs.microsoft.com/wp-content/uploads/prod/sites/5/2022/06/Microsoft-RAI-Impact-Assessment-Template.pdf

[8] E. W. Jatho III, L. O. Mailloux, S. Rismani, E. D. Williams, and J. A. Kroll, "System safety engineering for social and ethical ml risks: A case study," *arXiv preprint arXiv:2211.04602*, 2022.

[9] H. Khlaaf, P. Mishkin, J. Achiam, G. Krueger, and M. Brundage, "A hazard analysis framework for code synthesis large language models," *arXiv preprint arXiv:2207.14157*, 2022.

[10] R. Dobbe, "System safety and artificial intelligence," in *2022 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '22. ACM, Jun. 2022. [Online]. Available: http://dx.doi.org/10.1145/3531146.3533215

[11] N. Martelaro, C. J. Smith, and T. Zilovic, "Exploring opportunities in usable hazard analysis processes for ai engineering," 2022.

[12] S. Rismani, R. Shelby, A. Smart, R. Delos Santos, A. Moon, and N. Rostamzadeh, "Beyond the ml model: Applying safety engineering frameworks to text-to-image development," in *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, 2023, pp. 70–83.

[13] S. Rismani, R. Shelby, A. Smart, E. Jatho, J. Kroll, A. Moon, and N. Rostamzadeh, "From plane crashes to algorithmic harm: applicability of safety engineering frameworks for responsible ml," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–18.

[14] S. Rismani, R. Dobbe, and A. Moon, "From silos to systems: Process-oriented hazard analysis for ai systems," *arXiv preprint arXiv:2410.22526*, 2024.

[15] S. Rismani and A. Moon, "How do ai systems fail socially?: an engineering risk analysis approach," in *2021 IEEE International Symposium on Ethics in Engineering, Science and Technology (ETHICS)*. IEEE, 2021, pp. 1–8.

[16] R. Adler, P. Feth, and D. Schneider, "Safety engineering for autonomous vehicles," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*. IEEE, 2016, pp. 200–205.

[17] A. Abdulkhaleq, S. Wagner, D. Lammering, H. Boehmert, and P. Blueher, "Using stpa in compliance with iso 26262 for developing a safe architecture for fully automated vehicles," *arXiv preprint arXiv:1703.03657*, 2017.

[18] Leveson, Nancy G. and Thomas, John P., *STPA handbook*, MIT Partnership for Systems Approaches to Safety and Security (PSASS), Cambridge, Massachusetts, U.S., 2018.

[19] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 610–623.

[20] L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh *et al.*, "Ethical and social risks of harm from language models," *arXiv preprint arXiv:2112.04359*, 2021.

[21] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair representations," in *International conference on machine learning*. PMLR, 2013, pp. 325–333.

[22] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[23] Microsoft, "Responsible ai tools and practices," 2024, accessed: 2024-11-09. [Online]. Available: https://www.microsoft.com/en-us/ai/tools-practices

[24] K. Kieslich, N. Diakopoulos, and N. Helberger, "Anticipating impacts: Using large-scale scenario writing to explore diverse implications of generative ai in the news environment," 2023.

[25] E. Bogucka, M. Constantinides, S. Šćepanović, and D. Quercia, "Co-designing an ai impact assessment report template with ai practitioners and ai compliance experts," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, vol. 7, 2024, pp. 168–180.

[26] N. J. Bahr, *System Safety Engineering and Risk Assessment: A Practical Approach*, 2nd ed. Boca Raton, FL: CRC Press, 2014.

[27] N. Leveson, "A new accident model for engineering safer systems," *Safety science*, vol. 42, no. 4, pp. 237–270, 2004.

[28] S. International, "Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment," SAE International, Tech. Rep. ARP4761A, December 2023. [Online]. Available: https://www.sae.org/standards/content/arp4761a/

[29] N. G. Leveson, *Safeware: system safety and computers*. New York, NY, USA: Association for Computing Machinery, 1995.

[30] N. R. Storey, *Safety Critical Computer Systems*. USA: Addison-Wesley Longman Publishing Co., Inc., 1996.

[31] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press, 01 2012.

[32] OpenAI, "Gpt-4," 2023, accessed: 2024-11-04. [Online]. Available: https://openai.com/research/gpt-4

[33] ——, "Chatgpt assistant: Alltrails," https://chatgpt.com/g/g-KpF6lTka3-alltrails, accessed: 2024-10-25.

[34] S. Shankar, J. Zamfirescu-Pereira, B. Hartmann, A. Parameswaran, and I. Arawjo, "Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences," in *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, 2024, pp. 1–14.

[35] M. Howard and D. LeBlanc, *Writing Secure Code*, 2nd ed. Redmond, WA: Microsoft Press, 2003.

[36] M. Howard and S. Lipner, *The Security Development Lifecycle: SDL, a Process for Developing Demonstrably More Secure Software*. Redmond, WA: Microsoft Press, 2006.

[37] V. K. Singh and K. Joshi, "Automated machine learning (automl): an overview of opportunities for application and research," *Journal of Information Technology Case and Application Research*, vol. 24, no. 2, pp. 75–85, 2022.

[38] J. Drozdal, J. Weisz, D. Wang, G. Dass, B. Yao, C. Zhao, M. Muller, L. Ju, and H. Su, "Trust in automl: exploring information needs for establishing trust in automated machine learning systems," in *Proceedings of the 25th international conference on intelligent user interfaces*, 2020, pp. 297–307.

[39] C. Kästner, *Machine Learning in Production: From Models to Products*. Cambridge, MA: The MIT Press, 2025.

[40] W. P. Luz, G. Pinto, and R. Bonifácio, "Adopting devops in the real world: A theory, a model, and a case study," *Journal of Systems and Software*, vol. 157, p. 110384, 2019.

[41] K. E. Wiegers, *Creating a software engineering culture*. Pearson Education, 1996.

[42] B. Rakova, J. Yang, H. Cramer, and R. Chowdhury, "Where responsible ai meets reality: Practitioner perspectives on enablers for shifting organizational practices," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–23, 2021.

[43] R. J. Hironimus-Wendt and L. E. Wallace, "The sociological imagination and social responsibility," *Teaching Sociology*, vol. 37, no. 1, pp. 76–88, 2009.

[44] E. H. Schein, *Organizational culture and leadership*. John Wiley & Sons, 2010, vol. 2.