Obstacle Analysis in Requirements Engineering: Retrospective and Emerging Challenges

Emmanuel Letier and Axel van Lamsweerde

***Abstract*—With the growing adoption of AI-based systems, effective risk management is more important than ever. Obstacle analysis is a requirements engineering technique introduced three decades ago for designing dependable software systems despite failures, exceptions, and unforeseen behaviors in both the software and its environment. An obstacle is an undesirable situation that violates a stakeholder goal, an environment assumption, or a software requirement. Obstacles include safety hazards, security threats, user errors, and other adverse situations. Obstacle analysis provides a structured, systematic approach for identifying, analyzing, and resolving obstacles at the requirements level. In this retrospective paper, we summarize the original technique and discuss its impacts on research and practice. We also propose a research agenda to extend obstacle analysis to address emerging challenges in AI systems engineering.**

***Index Terms*—Obstacle analysis, risk analysis, fault-tolerance, exception handling, goal-oriented requirements engineering, formal specification, AI engineering.**

I. Introduction

As software engineers, we aim to build systems that serve meaningful purposes in the world [1], [2]. Our systems must remain dependable, even in the face of failures, exceptions, and unforeseen circumstances—whether within the software itself or its operating environment. Adverse situations are inevitable: hardware components such as sensors and actuators may fail; external software systems may malfunction; our own software may occasionally produce incorrect results; communication systems may prove unreliable; people may make mistakes or act unpredictably; and the deployment environment may present exceptional situations. Despite these challenges, our systems must remain safe, secure, and useful. Achieving this level of dependability requires careful engineering, starting at the requirements level and continuing through architectural design, quality assurance, operation, and system evolution [3].

This paper provides a retrospective account of our earlier work, *Handling Obstacles in Goal-Oriented Requirements Engineering*, published in this journal's October 2000 *Special Issue on Exception Handling* [4]. The paper was an expanded version of an ICSE'98 paper [5]. It introduced a structured approach for designing dependable systems by managing exceptions *at the requirements level*. The paper's objective was to tackle the challenges of incomplete and overly optimistic requirements [1]—challenges that are still critical today in the engineering of AI-based systems [6], [7]. The paper defined *obstacles* as undesirable situations that may obstruct the satisfaction of stakeholder goals, software requirements, and environment assumptions. It presented systematic techniques for identifying obstacles and resolving them by revising the initial, idealized goals, requirements, and assumptions.

To describe the paper's novel ideas, we first briefly recall the goal-oriented requirements engineering method it builds upon. We then summarize the TSE'2000 paper, highlight its contributions over other hazard and risk analysis methods, and present subsequent research and industrial applications. We conclude by outlining a research agenda for obstacle analysis in the context of modern AI systems.

II. Goal-Oriented Requirements Engineering

Goal-oriented requirements engineering methods focus on stakeholder goals to guide the elicitation, specification, and analysis of software requirements [1]. *Stakeholder goals* are desired properties of the world in which the software operates. For example, in an Automated Driving System (ADS), stakeholder goals include adhering to traffic laws, avoiding collisions, and ensuring overall safety. A *goal model* relates stakeholder goals to subgoals through refinement links, ultimately connecting them to software requirements and environment assumptions. Goal models can be represented as AND/OR refinement graphs, where the leaves are software requirements and environment assumptions. Conceptually, such models establish an argument that if the software satisfies its requirements (R) and the environment satisfies its assumptions (A), then the stakeholder goals (G) are satisfied: $R, A \vdash G$ [2].

Consider the stakeholder goal of avoiding collisions (*G1*) in an ADS. The leaf refinements of this goal might include software requirements such as: (*R1*) *"The ADS must alert the driver when it determines that their attention is needed to prevent a potential collision"* and (*R2*) *"The ADS must activate emergency braking when it predicts an imminent collision."* The leaf refinements will also include environment assumptions on which the ADS relies to satisfy the goal *G1* of avoiding collisions: (*A1*)*"When the ADS alerts the driver, the driver will resume control of the car"*; (*A2*) *"When the driver resumes control of the car, they will avoid a collision"*; (*A3*) *"When the ADS activates emergency braking, the car will stop in a short distance"*; and so on. One can then establish that if the ADS satisfies its requirements and the environment satisfies its assumptions, the goal *G1* will be met.

Goals, requirements, and assumptions are expressed in natural language and can optionally be formulated in Linear Temporal Logic (LTL) for formal reasoning and analysis. The

Emmanuel Letier is an Associate Professor in the Department of Computer Science at University College London (e.letier@ucl.ac.uk).

Axel van Lamsweerde is an Emeritus Professor in the Department of Computer Science at Université catholique de Louvain, Belgium. (axel.vanlamsweerde@uclouvain.be).

LTL formulae are used ”behind the scenes” to enable automated techniques for model checking, requirements animation, and test generation, while remaining hidden from stakeholders.

III. Obstacle Analysis

Prior to the introduction of obstacle analysis, requirements engineering methods largely overlooked the problem of idealized goals, requirements, and assumptions. Although system failures were often attributed to invalid assumptions [2], [8], no methods existed to identify, describe, or analyze such flaws. Likewise, methods were lacking to identify and analyze potential violations of idealized goals, requirements and assumptions.

Our work built on earlier contributions by Potts and Anton, who first introduced the idea of considering obstacles—characterized as “anything that could thwart a goal”—to generate exceptional scenarios during requirements elicitation [9], [10]. We extended this idea by formalizing the concept of obstacles and making them declarative instead of operational. Additionally, we integrated obstacles into a goal-oriented requirements engineering process and developed systematic techniques for their identification and resolution. The TSE’2000 paper extends our work presented at ICSE’98 [5] and features an extensive application of the approach to a large real-world example [11], based on reports of the 1993 failure of the London Ambulance Service [12].

An ***obstacle*** *O* to some goal, assumption or requirement *P* is a property satisfying two conditions:

1) **Obstruction**: if the obstacle holds in a world that satisfies the domain properties *Dom*, then *P* will not hold: $O, Dom \models \neg P$.
2) **Domain consistency**: the obstacle is logically consistent with the domain properties *Dom*, that is, $O \wedge Dom$ is satisfiable.

For example, consider the earlier environment assumption *A1*: *“When the ADS alerts the driver, the driver will resume control of the car”*. One obstacle to this assumption is *O1.1*: *“The driver is asleep when the ADS alerts them”*. The obstacle O1.1 obstructs A1 based on a domain property, *Dom1*: *”A driver cannot resume control of the car if they are asleep”*. *O1.1* is an obstacle to *A1* because *O1.1* implies the negation of *A1* when *Dom1* holds, and *O1.1* is consistent with the domain properties. That same environment assumption has many other obstacles, such as *O1.2*: *“The driver is distracted by their mobile phone when the ADS alerts them”*, and *O1.3*: *“The driver is incapacitated by a medical emergency”*.

Obstacles can be AND/OR refined into sub-obstacles, similar to fault trees [13]. This process produces **obstacle refinement trees**, providing a structured approach to exploring and analyzing situations that could lead to the violation of a goal, requirement, or assumption. Each refinement tree is rooted in the obstructed goal, requirement, or assumption, with the top-level obstacle representing its negation. Additionally, the formalization of obstacle refinement trees in LTL enables automated reasoning and analysis.

The obstacle analysis process consists of three main steps:

1) **Goal Model Elaboration:** Build a goal model that captures stakeholder goals and their refinements into subgoals down to software requirements and environment assumptions.
2) **Obstacle Identification:** Identify obstacles that obstruct the *leaf* goals in the goal model, that is, those requirements and assumptions.
3) **Obstacle Resolution:** Generate new goals and requirements, or revise existing ones, in order to prevent, reduce, or mitigate the identified obstacles.

This process is iterative and incremental. Obstacle identification can begin before the goal model is fully elaborated. Obstacle identification and resolution may be intertwined. Resolving obstacles often generates new goals and requirements, which may, in turn, trigger further obstacle identification and resolution. The process ends when the risks associated with the remaining unresolved obstacles are deemed acceptable—a challenging judgement, which is partly addressed in subsequent work (see Section V).

Obstacle Identification Techniques. To support obstacle identification, our TSE’2000 paper introduced two formal, logic-based techniques: a regression procedure inspired by AI planning and obstacle refinement patterns in the style of formal goal refinement patterns [14]. The paper also presented informal heuristics derived from these patterns and past experiences.

Obstacle Resolution Techniques. Our paper also introduced a catalog of resolution strategies, classified into three broad categories.

1. Obstacle elimination strategies aim to remove the obstacle entirely. These strategies include: (1) selecting an alternative design where the obstructed assumption or requirement is no longer needed; (2) introducing a new goal that requires avoiding the obstacle; or (3) weakening the obstructed requirement or assumption.

For example, resolving the obstacle *O1.1* (“*Driver Asleep*”) could involve: (1) designing an alternative ADS that does not depend on the driver to avoid collisions, thus eliminating the need for assumption *A1* and, consequently, the existence of *O1.1*; (2) introducing a new goal, “*Keep the Driver Awake*”, which could then be refined into software requirements for monitoring and ensuring driver wakefulness; or (3) modifying *A1* to a weaker assumption *A1’*: “*When the ADS alerts the driver and the driver is awake, the driver will resume control of the car*”. Such a change would then be propagated along refinement links in the goal model, affecting related goals and requirements. The paper introduced formal patterns for deidealizing assertions and systematically propagating the resulting changes.

2. Obstacle reduction strategies aim to decrease the likelihood or frequency of an obstacle occurring rather than eliminating it entirely. These strategies often involve measures designed to influence human behavior. The original paper did not explore such strategies in detail; examples include education and training (e.g., informing drivers about the importance of staying alert), behavioral nudges (e.g., periodic cues to maintain driver attentiveness), or incentive structures (e.g., rewards for the safe and attentive use of the ADS).

3. Obstacle tolerance strategies aim to mitigate the consequences of an obstacle that cannot be entirely eliminated or is too costly to address fully. These strategies produce new goals to ensure that critical parent goals of the obstructed requirement or assumption remain satisfied despite the obstacle. There are three steps: (1) identify all parent goals impacted by the obstacle by following the refinement links in the goal model; (2) determine which of these parent goals are critical and must be preserved despite the obstacle; and (3) produce new goals to ensure the critical parent goals are maintained in the presence of the obstacle. For example, consider the obstacle *O1: "The ADS alerts the driver, and the driver does not resume control of the car*", which cannot be entirely eliminated. A critical parent goal affected by *O1* is the initial goal *G1*: "*Avoid Collisions*". Applying this strategy leads to generating a new goal: "*Collisions must be avoided even if the driver does not resume control of the car when alerted*". Refining this new goal results in requirements such as: "*If the driver does not resume control of the car when alerted, the ADS should slow the car to a gradual stop*", and "*The ADS must activate emergency braking when it predicts an imminent collision*".

A single obstacle can be resolved in multiple ways. The obstacle resolution strategies provide a structured approach for exploring alternatives. By systematically considering all strategies, the process encourages the creative exploration of alternative system designs that might otherwise be missed.

Once potential resolutions are identified, they must be evaluated, and a subset selected based on factors such as cost, impact on goal satisfaction, and associated risks. The TSE'2000 paper did not address the evaluation and selection of resolution strategies—a challenging problem that we explored in subsequent work discussed in Section V.

IV. Contributions

Our TSE'2000 paper introduced several contributions to requirements engineering and risk analysis, particularly in comparison to other hazard and risk analysis methods such as Fault Tree Analysis (FTA), Failure Mode and Effect Analysis (FMEA), and Hazard and Operability Study (HAZOP) [13].

Goal-Anchored Risk Identification. The primary advantage of obstacle analysis lies in its integration with a goal model. Anchoring risk analysis on a goal model, elaborated through systematic techniques [1], exposes the numerous environment assumptions upon which the system depends. These assumptions are made explicit and serve as a starting point for risk identification. The paper introduced heuristics and formal techniques to support this identification.

Goal-Driven Risk Reduction. The integration with a goal model also facilitates the generation and evaluation of multiple risk reduction alternatives. The goal model helps identify the potential impacts of an obstacle on stakeholder goals by tracing refinement links upward through obstacle trees and goal refinement trees. It also helps pinpoint critical goals that should be preserved despite the obstacles. In this way, risk reduction is explicitly driven by the stakeholders' critical goals. The paper introduced a catalog of obstacle resolution strategies to support this process.

Systematic Goal-Oriented Process. Analyzing the numerous failures and exceptions in a typical system is unmanageable without a systematic process. Obstacle analysis offers a structured approach to manage this complexity, ensuring that all activities remain focused on achieving stakeholder goals.

Structured Documentation of Exceptions. Obstacle analysis also supports organizing requirements documentation. It allows the formulation of failure-handling and exception-handling goals, requirements, and assumptions to be distinct from—but clearly linked to—the goal model for ideal scenarios. This ensures clarity and traceability in addressing both normal and exceptional cases.

Formal Foundations and Reasoning Obstacle analysis provides precise definitions for key concepts such as obstacles, goal obstructions, and obstacle refinements; these concepts are grounded in Linear Temporal Logic. Such formal foundation enables automated techniques for obstacle identification and resolution. Additionally, many of the informal heuristics for obstacle identification and resolution are derived from and informed by this formal foundation.

These benefits of obstacle analysis became increasingly clearer over time through subsequent research and practical experience, discussed in the next section.

V. Subsequent Research and Applications

Our TSE'2000 obstacle analysis framework paved the way to further research and industrial applications.

Research Developments. Obstacle analysis highlights that goals may occasionally be violated, leading to the need to reason about levels of goal satisfaction. In addition to considering a goal as a Boolean property (e.g., *"no collision"*), it is necessary to define metrics that quantify the level of goal satisfaction (e.g., *"number of collisions per 100,000 vehicle-miles travelled"*). We therefore extended our goal modelling framework with a quantitative layer for specifying such metrics, linking stakeholder goal metrics to software requirements metrics, and evaluating the impacts of requirements-level design decisions on these metrics [15]. The resulting quantitative goal models can be analyzed using stochastic simulation and multi-objective optimization techniques [16]. These techniques enable the evaluation of obstacle severity and the identification of Pareto-optimal obstacle resolutions. The approach has since been extended to address parameter uncertainty [17] and implemented in a lightweight goal modelling tool [18]. In parallel, probabilistic goals were introduced to assess how likely and critical the identified obstacles are, so as to support an informed resolution step [19]. This probabilistic framework was later extended to cope with uncertainty margins about estimates of likelihoods in such obstacle assessment [20].

Further support was also provided for the resolution step to determine where and how exception handling goals should be integrated in a goal model to make it more complete [21].

As an alternative to handling obstacles at design time, techniques were also developed for runtime monitoring of obstacle satisfaction rates together with on-the-fly system adaptation to more effective countermeasures [22].

As mentioned before, the LTL formalization of goals and obstacles enables formal reasoning. Various efforts were undertaken in this direction. For the obstacle identification step, a tool-supported technique was developed for developing a complete and consistent set of obstacles. The technique combines model checking, for generating counterexample traces and witness traces from goal specifications, and inductive obstacle learning from these negative/positive traces [23]. The combined use of model checking and inductive learning further allowed the obstacle-driven generation of resolutions [24] and the automated adaptation of goal/obstacle models to varying environment conditions [25].

Obstacle analysis techniques have also been extended for specific concerns.

For security, one must address malicious obstacles, referred to as *anti-goals*, set up by attackers to threaten security goals [26]. Threat trees are built systematically through anti-goal refinement until leaf nodes are derived that are either software vulnerabilities observable by the attacker or anti-requirements implementable by this attacker. Security requirements are then obtained as countermeasures by application of threat resolution operators.

For scalability, one needs to consider situations where a goal cannot be satisfied by its responsible agent because the load imposed by the goal exceeds the agent capacity [27]. Patterns and heuristics are available to support the identification and resolution of such scalability-related obstacles.

Our obstacle analysis framework has also been applied in various research areas, for example, in managing uncertainty in adaptive systems [28]–[30]; defining requirements for a product family of DNA devices [32]; assessing the likelihood of goal conflicts [33]; migrating legacy systems to cloud platforms [34]; exploring trade-offs in resolving obstacles [35]; analysing cybersecurity risks in automotive systems [36]; and ensuring cybersecurity in operating system kernels [37]. These examples illustrate the breadth of work that referenced and used our obstacle analysis framework over time.

Industrial applications. Obstacle analysis has also been applied in industrial projects [38]. Noteworthy examples include: the specification of contingency requirements for an unpiloted aerial vehicle at NASA [39]; the definition of security goals and requirements for a threat assessment and response management system for civil aviation, developed in response to the 9/11 terrorist attacks [40]; the analysis of scalability requirements for the redesign of a complex, large-scale financial fraud detection system used by many of the world's largest banks [27], [41]; and an industrial pilot study where goal-obstacle modeling was used for the certification of new technology in safety-critical infrastructure for the offshore energy industry [31]. Many applications are unreported. For example, a major global provider of telecommunication technologies and mobile devices used goal modelling and obstacle analysis in the development of a fingerprint authentication feature for mobile phones.

VI. Perspectives

Although obstacle analysis was originally conceived within the context of the KAOS goal-oriented method, its core principles of identifying and resolving obstacles to goals, requirements, and assumptions are applicable to other requirements engineering methods that incorporate explicit specifications of goals, requirements, and assumptions. This includes the Problem Frame approach [42], the REVEAL method [43], [44] and, potentially, future frameworks for requirements engineering in the age of AI. We anticipate that the core principles of obstacle analysis might play significant roles both in the engineering of AI systems (SE4AI) and in AI-supported software engineering (AI4SE).

Obstacle Analysis for AI Systems Engineering. Obstacle analysis is well-suited to the design of AI systems. Since the machine learning (ML) components in such systems cannot be expected to be correct at all times, there is a clear need for systematically identifying cases where the ML model may fail, and for designing a system to handle such cases effectively. Obstacle identification and resolution techniques might provide valuable support for this process. Typical obstacles (e.g., false positives and negatives) and resolution strategies (e.g., human-in-the-loop, guardrails) are already well-documented [45], [46]. They might be formulated as heuristics and patterns for obstacle identification and resolution to simplify and promote their application.

Significant research is also needed to address some of the most complex challenges of requirements engineering for AI systems [47], [48], such as those described below.

One important challenge in the field of autonomous vehicles is the specification of Operational Design Domains (ODD), that is, real-world contexts under which the AI-based system is expected to operate safely [49], [50]. In our goal modelling framework, an ODD specification amounts to a set of environment assumptions that the autonomous vehicle software relies on to satisfy its safety goals. An accurate ODD specification is crucial for the system's safety, as it informs the construction of training and evaluation datasets. The ODD specification is also crucial for the implementation of monitoring requirements to detect when the system leaves its ODD or moves from one ODD to another. A challenging aspect of specifying an ODD is the identification and specification of edge case scenarios, such as rare events, exceptional situations and adverse behaviors [50]. Obstacle analysis might provide a natural conceptual framework to address this challenge through novel techniques for identifying and resolving obstacles to ODD assumptions.

The challenges of specifying ODD assumptions and analysing their obstacles extend beyond autonomous vehicles. All AI-based systems are designed to operate within a specific context; assumptions about that context guide the system design as well as the ML model training, evaluation, and runtime monitoring. The ability to specify such assumptions and analyse their obstacles will be essential in future requirements engineering methods for AI-based systems.

A second significant challenge is the AI alignment problem [51]. Many AI systems, particularly those using reinforcement learning, are driven by the optimization of an objective function. The AI alignment problem arises when the objective function provided to the AI system diverges from the actual stakeholder goals that system designers, users, and society intend to be pursued. This misalignment can lead to situations where the AI system optimizes its objective function but violates important stakeholder goals. From an engineering perspective, the AI alignment problem can be analysed through the lens of obstacle analysis. Since mathematical objective functions are always proxies for actual stakeholder goals, building AI systems with guarantees that they will always pursue the correct goals in the correct way appears unfeasible. Consequently, potential failures must be anticipated and addressed by identifying and resolving "*alignment obstacles*", that is, situations where optimizing the objective function leads to the violation of critical stakeholder goals. The research challenge lies in developing automated techniques to support the identification and resolution of such obstacles. This effort would complement other research on modelling human values, which are also relevant to addressing the AI alignment problem [52].

A third research challenge is to extend obstacle analysis to address new categories of stakeholder goals, such as fairness and explainability, which have become critical concerns in the design of AI systems [53]–[55].

More broadly, we believe that a formal, quantitative goal-oriented requirements engineering framework provides a strong conceptual foundation for developing future requirements engineering methods for AI engineering, due to its ability to relate stakeholder goals to software requirements (including relating stakeholder goals metrics to ML performance metrics [18]), to systematically explore and resolve large numbers of obstacles, and to analyze trade-offs under uncertainty [16]–[18].

AI-Assisted Obstacle Analysis. Another promising area of research is the exploration of AI techniques to support obstacle analysis. This aligns with the current surge in applying AI methods, particularly Large Language Models (LLMs), to software engineering tasks, including requirements engineering tasks [56]–[58].

A first direction would be to explore the use of LLMs for identifying and resolving obstacles in the context of goal-oriented requirements models outlined in this paper. Goal modelling tools such as *Objectiver* (https://www.objectiver.com) could include an AI assistant that would help requirements engineers in identifying and resolving obstacles. Incorporating our existing heuristics and strategies for obstacle identification and resolution into the LLM prompts might enhance the assistant's ability to identify a broader range of obstacles and propose more diverse resolution alternatives.

Another direction would be to enhance emerging tools for the automated generation of requirements documents from natural language problem statements by integrating obstacle analysis steps into their chain of thought. This approach might build on promising initial research in this area [59] and be incorporated into commercial tools in the style of chatPRD (https://www.chatprd.ai/). An obstacle analysis feature would enable such tools to generate more comprehensive requirements by considering potential failures and exceptions, rather than focusing solely on the normal "happy path".

A third direction is to develop AI-based data analysis and decision support techniques to assess obstacle likelihoods, predict their impacts on stakeholder goals and ultimately guide the selection of suitable obstacle resolutions among all generated candidates. These techniques would enhance, complement or replace existing techniques for requirements-level decision-making [18] and for design-time and run-time analysis of obstacles and their resolutions [20], [22].

In conclusion, we believe that obstacle analysis provides a useful conceptual framework for studying emerging challenges of AI-based systems engineering and for developing new techniques to address these challenges. Moreover, we anticipate that obstacle analysis will play a crucial role in AI-assisted software engineering methods by enabling future AI-based requirements assistants to generate more complete requirements that account for potential failures, adverse behaviors, and other exceptions in both the software and its environment.

Acknowledgments

This paper is dedicated to the memory of Dewayne Perry (1940-2024), who was Guest Editor of the TSE'2000 Special Issue on Exception Handling. Dewayne will always be remembered for his leadership in software engineering research and for his insights on the bridges between software engineering and music.

References

[1] A. van Lamsweerde, *Requirements engineering: from system goals to UML models to software specifications*. Wiley, 2009.

[2] M. Jackson, "The world and the machine," in *Proceedings of the 17th international conference on Software engineering*, 1995, pp. 283–292.

[3] L. I. Millett, M. Thomas, and D. Jackson, *Software for dependable systems: Sufficient evidence?* National Academies Press, 2007.

[4] A. Van Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering," *IEEE Transactions on software engineering*, vol. 26, no. 10, pp. 978–1005, 2000.

[5] ——, "Integrating obstacles in goal-driven requirements engineering," in *Proceedings of the 20th international conference on software engineering*. IEEE, 1998, pp. 53–62.

[6] N. Nahar, H. Zhang, G. Lewis, S. Zhou, and C. Kästner, "A meta-summary of challenges in building products with ML components–collecting experiences from 4758+ practitioners," in *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*. IEEE, 2023, pp. 171–183.

[7] M. Kalinowski, D. Mendez, G. Giray, A. P. S. Alves, K. Azevedo, T. Escovedo, H. Villamizar, H. Lopes, T. Baldassarre, S. Wagner *et al.*, "Naming the pain in machine learning-enabled systems engineering," *arXiv preprint arXiv:2406.04359*, 2024.

[8] A. Van Lamsweerde, R. Darimont, and P. Massonet, "Goal-directed elaboration of requirements for a meeting scheduler: Problems and lessons learnt," in *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*. IEEE, 1995, pp. 194–203.

[9] C. Potts, "Using schematic scenarios to understand user needs," in *Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques*, 1995, pp. 247–256.

[10] A. I. Antón and C. Potts, "The use of goals to surface requirements for evolving systems," in *Proceedings of the 20th international Conference on Software Engineering*. IEEE, 1998, pp. 157–166.

[11] E. Letier *et al.*, "Reasoning about agents in goal-oriented requirements engineering," Ph.D. dissertation, PhD thesis, Université catholique de Louvain, 2001.

[12] D. Page, P. Williams, and D. Boyd, *Report of the Inquiry into the London Ambulance Service, February 1993*. South West Thames Regional Health Authority, 1993.

[13] N. G. Leveson, *Safeware: system safety and computers*. Addison-Wesley, 1995.

[14] R. Darimont and A. Van Lamsweerde, "Formal refinement patterns for goal-driven requirements elaboration," *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 6, pp. 179–190, 1996.

[15] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *FSE*, 2004, pp. 53–62.

[16] W. Heaven and E. Letier, "Simulating and optimising design decisions in quantitative goal models," in *International Requirements Engineering Conference*, 2011, pp. 79–88.

[17] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *ICSE*, 2014, pp. 883–894.

[18] S. A. Busari and E. Letier, "RADAR: A lightweight tool for requirements and architecture decision analysis," in *ICSE*, 2017, pp. 552–562.

[19] A. Cailliau and A. van Lamsweerde, "Assessing requirements-related risks through probabilistic goals and obstacles," *Requirements Engineering*, vol. 18, pp. 129–146, 2013.

[20] A. Cailliau and A. Van Lamsweerde, "Handling knowledge uncertainty in risk-based requirements engineering," in *2015 IEEE 23rd international requirements engineering conference (RE)*. IEEE, 2015, pp. 106–115.

[21] ——, "Integrating exception handling in goal models," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, 2014, pp. 43–52.

[22] A. Cailliau and A. V. Lamsweerde, "Runtime monitoring and resolution of probabilistic obstacles to system goals," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 14, no. 1, pp. 1–40, 2019.

[23] D. Alrajeh, J. Kramer, A. Van Lamsweerde, A. Russo, and S. Uchitel, "Generating obstacle conditions for requirements completeness," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 705–715.

[24] D. Alrajeh, A. Van Lamsweerde, J. Kramer, A. Russo, and S. Uchitel, "Risk-driven revision of requirements models," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 855–865.

[25] D. Alrajeh, A. Cailliau, and A. van Lamsweerde, "Adapting requirements models to varying environments," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 50–61.

[26] A. Van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *Proceedings. 26th International Conference on Software Engineering*. IEEE, 2004, pp. 148–157.

[27] L. Duboc, E. Letier, and D. S. Rosenblum, "Systematic elaboration of scalability requirements through goal-obstacle analysis," *IEEE Transactions on Software Engineering*, vol. 39, no. 1, pp. 119–140, 2012.

[28] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, "RELAX: a language to address uncertainty in self-adaptive systems requirement," *Requirements engineering*, vol. 15, pp. 177–196, 2010.

[29] A. J. Ramirez, A. C. Jensen, B. H. Cheng, and D. B. Knoester, "Automatically exploring how uncertainty impacts behavior of dynamically adaptive systems," in *2011 26th IEEE/ACM international conference on automated software engineering (ASE 2011)*. IEEE, 2011, pp. 568–571.

[30] M. A. Langford, K. H. Chan, J. E. Fleck, P. K. McKinley, and B. H. Cheng, "MoDALAS: Model-driven assurance for learning-enabled autonomous systems," in *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2021, pp. 182–193.

[31] M. Sabetzadeh, D. Falessi, L. Briand, S. Di Alesio, D. McGeorge, V. Åhjem, and J. Borg, "Combining goal models, expert elicitation, and probabilistic simulation for qualification of new technology," in *2011 IEEE 13th International Symposium on High-Assurance Systems Engineering*. IEEE, 2011, pp. 63–72.

[32] R. R. Lutz, J. H. Lutz, J. I. Lathrop, T. H. Klinge, D. Mathur, D. M. Stull, T. G. Bergquist, and E. R. Henderson, "Requirements analysis for a product family of dna nanodevices," in *2012 20th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2012, pp. 211–220.

[33] R. Degiovanni, P. Castro, M. Arroyo, M. Ruiz, N. Aguirre, and M. Frias, "Goal-conflict likelihood assessment based on model counting," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 1125–1135.

[34] M. Fahmideh and G. Beydoun, "Reusing empirical knowledge during cloud computing adoption," *Journal of Systems and Software*, vol. 138, pp. 124–157, 2018.

[35] C. Ponsard and R. Darimont, "Towards quantitative trade-off analysis in goal models with multiple obstacles using constraint programming." in *ICSOFT*, 2020, pp. 537–543.

[36] C. Ponsard, V. Ramon, and J.-C. Deprez, "Goal and threat modelling for driving automotive cybersecurity risk analysis conforming to iso/sae 21434." in *SECRYPT*, 2021, pp. 833–838.

[37] S. Mergendahl, S. Fickas, B. Norris, and R. Skowyra, "Manipulative interference attacks," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 4569–4583.

[38] A. Van Lamsweerde, "Goal-oriented requirements engineering: a roundtrip from research to practice [enginering read engineering]," in *Proceedings. 12th IEEE International Requirements Engineering Conference, 2004*. IEEE, 2004, pp. 4–7.

[39] R. Lutz, A. Patterson-Hine, S. Nelson, C. R. Frost, D. Tal, and R. Harris, "Using obstacle analysis to identify contingency requirements on an unpiloted aerial vehicle," *Requirements Engineering*, vol. 12, pp. 41–54, 2007.

[40] R. Darimont and M. Lemoine, "Security requirements for civil aviation with UML and goal orientation," in *Requirements Engineering: Foundation for Software Quality: 13th International Working Conference, REFSQ 2007, Trondheim, Norway, June 11-12, 2007. Proceedings 13*. Springer, 2007, pp. 292–299.

[41] L. Duboc, E. Letier, D. S. Rosenblum, and T. Wicks, "A case study in eliciting scalability requirements," in *2008 16th IEEE International Requirements Engineering Conference*. IEEE, 2008, pp. 247–252.

[42] M. Jackson, *Problem Frames: Analyzing and structuring software development problems*. Addison-Wesley Longman Publishing Co., Inc., 2000.

[43] P. C. S. Limited, "REVEAL: A Keystone of Modern Systems Engineering," Praxis Critical Systems Limited, White Paper Reference S.P0544.19.1, 2001.

[44] J. Hammond, R. Rawlings, and A. Hall, "Will it work?" in *Proceedings Fifth IEEE International Symposium on Requirements Engineering*. IEEE, 2001, pp. 102–109.

[45] C. Kastner, *Machine learning in production: from models to products*. MIT Press, 2025.

[46] G. Hulten, *Building intelligent systems: a guide to machine learning engineering*. Apress, 2018.

[47] A. P. S. Alves, M. Kalinowski, G. Giray, D. Mendez, N. Lavesson, K. Azevedo, H. Villamizar, T. Escovedo, H. Lopes, S. Biffl, J. Musil, M. Felderer, S. Wagner, T. Baldassarre, and T. Gorschek, "Status quo and problems of requirements engineering for machine learning: Results from an international survey," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2023, pp. 159–174.

[48] M. Kalinowski, D. Mendez, G. Giray, A. P. S. Alves, K. Azevedo, T. Escovedo, H. Villamizar, H. Lopes, T. Baldassarre, S. Wagner, S. Biffl, J. Musil, M. Felderer, N. Lavesson, and T. Gorschek, "Naming the pain in machine learning-enabled systems engineering," *arXiv preprint arXiv:2406.04359*, 2024.

[49] H.-M. Heyn, E. Knauss, A. P. Muhammad, O. Eriksson, J. Linder, P. Subbiah, S. K. Pradhan, and S. Tungal, "Requirement engineering challenges for ai-intense systems development," in *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*. IEEE, 2021, pp. 89–96.

[50] K. M. Habibullah, H.-M. Heyn, G. Gay, J. Horkoff, E. Knauss, M. Borg, A. Knauss, H. Sivencrona, and P. J. Li, "Requirements and software engineering for automotive perception systems: an interview study," *Requirements Engineering*, pp. 1–24, 2024.

[51] I. Gabriel, "Artificial intelligence, values, and alignment," *Minds and machines*, vol. 30, no. 3, pp. 411–437, 2020.

[52] W. Hussain, H. Perera, J. Whittle, A. Nurwidyantoro, R. Hoda, R. A. Shams, and G. Oliver, "Human values in software engineering: Contrasting case studies of practice," *IEEE Transactions on Software Engineering*, vol. 48, no. 5, pp. 1818–1833, 2020.

[53] Y. Brun and A. Meliou, "Software fairness," in *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2018, pp. 754–759.

[54] L. Chazette, V. Klös, F. Herzog, and K. Schneider, "Requirements on explanations: a quality framework for explainability," in *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE, 2022, pp. 140–152.

[55] J. M. Wing, "Trustworthy AI," *Communications of the ACM*, vol. 64, no. 10, pp. 64–71, 2021.

[56] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, pp. 1–79, 2024.

[57] M. Borg, "Requirements engineering and large language models: Insights from a panel," *IEEE Software*, vol. 41, no. 2, pp. 6–10, 2024.

[58] A. Vogelsang, "From specifications to prompts: On the future of generative large language models in requirements engineering," *IEEE Software*, vol. 41, no. 5, pp. 9–13, 2024.

[59] M. Krishna, B. Gaur, A. Verma, and P. Jalote, "Using LLMs in software requirements specifications: An empirical evaluation," *arXiv preprint arXiv:2404.17842*, 2024.