Requirements Engineering
https://doi.org/10.1007/s00766-024-00415-4

ORIGINAL ARTICLE

An empirical investigation of challenges of specifying training data and runtime monitors for critical software with machine learning and their relation to architectural decisions

Hans-Martin Heyn[1] · **Eric Knauss**[1] · **Iswarya Malleswaran**[1] · **Shruthi Dinakaran**[1]

Received: 3 July 2023 / Accepted: 12 February 2024

Abstract
The development and operation of critical software that contains machine learning (ML) models requires diligence and established processes. Especially the training data used during the development of ML models have major influences on the later behaviour of the system. Runtime monitors are used to provide guarantees for that behaviour. Runtime monitors for example check that the data at runtime is compatible with the data used to train the model. In a first step towards identifying challenges when specifying requirements for training data and runtime monitors, we conducted and thematically analysed ten interviews with practitioners who develop ML models for critical applications in the automotive industry. We identified 17 themes describing the challenges and classified them in six challenge groups. In a second step, we found interconnection between the challenge themes through an additional semantic analysis of the interviews. We explored how the identified challenge themes and their interconnections can be mapped to different architecture views. This step involved identifying relevant architecture views such as data, context, hardware, AI model, and functional safety views that can address the identified challenges. The article presents a list of the identified underlying challenges, identified relations between the challenges and a mapping to architecture views. The intention of this work is to highlight once more that requirement specifications and system architecture are interlinked, even for AI-specific specification challenges such as specifying requirements for training data and runtime monitoring.

Keywords Architecture framework · Artificial intelligence · Data requirements · Requirement engineering · Requirements specification · Runtime monitoring

1 Introduction

With constant regularity, unexpected and undesirable behaviour of machine learning (ML) models are reported in academia [12, 35, 38, 86, 87], the press, and by NGOs.[1] These problems become especially apparent, and reported upon, when ML models violate ethical principles through failures or biases in the ML component. Racial, religious, or gender biases are introduced through a lack of insight into the (sometimes immensely large set of) training data and missing runtime checks for example in large language models such as GPT-3 [1], or facial recognition software based on deep learning [57]. This lack of insight might be related to a lack of precise requirements. For industrial practitioners these problems become especially apparent in critical systems, such as for example in automatic driving systems, for which the company must provide due diligence, i.e., it can be made responsible for malfunctions. Improving the performance of ML models, however, often requires an exponential growth in training data [5]. Data requirements can help in preventing unnecessarily large and biased datasets because they provide guidance to the necessary sets of data for a ML project [83]. Data requirements can entail details regarding the quality of the information content represented by the data, e.g., about

✉ Hans-Martin Heyn
hans-martin.heyn@gu.se

✉ Eric Knauss
eric.knauss@cse.gu.se

[1] Computer Science and Engineering, University of Gothenburg and Chalmers, 40530 Göteborg, Västra Götaland, Sweden

[1] Non-governmental organisations, e.g., https://algorithmwatch.org/en/stories/.

Published online: 21 March 2024

the entailed features in the data, accuracy, completeness, consistency, or diversity of the information. Standards such as IEC 25012 can provide guidance towards defining requirements on data [39]. Data requirements can also cover physical aspects of data, e.g., the necessary time resolution, image resolution, or frequency spectrum coverage. Due to changes in the environment, ML models can become "stale", i.e., the context changes over time so significantly that the performance of the model decreases below acceptable levels [7]. Changes in the environment can cause a *distribution shift* in the data available to the model at runtime compared to the data that was used when training the model, and as a consequence, the training data no longer represents the data at runtime. Some examples of such distribution shifts due to changes in the environment can be found for example in [66, page 7]. Runtime monitors collect performance data and indicate the need for re-training of the model with updated training data, see [3] for an example framework for monitoring distribution shifts in safety-critical ML applications. However, these monitors need to be specified at design time.

Data requirements can support the specification of runtime monitors [9]. For example, a data requirement can be a minimum brightness of the images in order to be able to identify the object. A runtime monitor can be constructed which continuously checks the average brightness of the images and raises an alarm if the brightness drops under a critical threshold defined in the original data requirement. The lack of data requirements is particularly evident in ML models that are part of *critical* software. We define critical software as software that is safety, privacy, ethically, and/or mission critical, i.e., a failure in the software can cause significant injury or the loss of life, invasion of personal privacy, violation of human rights, and/or significant economic or environmental consequences [51]. Without clear specifications, it will not be possible to establish traceability from system requirements (e.g., functional safety requirements) to requirements set on the training data and the runtime monitoring [56].

Early on in software engineering, the close relation between software requirements specification and software architecture have been highlighted [16]. Requirements and architecture typically evolve together, and the choice of requirements can influence the architecture and vice versa, making simultaneous consideration necessary [64]. This is commonly known as the "twin peaks" of requirements and architecture [19]. Especially in nowadays common Agile Development Environments, the guidance of architecture views on different perspectives of the system is necessary to discover and align the necessary requirements iteratively [63, 65]. Therefore, we also explore in this study how architecture principles, i.e., an architecture framework as proposed for example in [32], could potentially mitigate the challenges of finding requirements and consequently specifications for software with ML components.

1.1 Scope and research questions

This article is an extension of an earlier conference article [31]. The purpose of the original study was to identify current challenges faced by practitioners in specifying requirements for training data and runtime monitoring for ML in safety-critical software. The study also explores possible relationships between these challenges. This study extends these findings by mapping the identified challenges to design decisions in relation to the architecture of the overall system.

The original article reported on the qualitative data collected in the interview study, specifically on the challenges reported by practitioners. It contributed a practitioner's point of view on the challenges reported in the academic literature on data specification and runtime monitoring specification for critical software with ML models. This paper extends the analysis and discussion of the reported challenges by identifying relationships between the challenges and relating them to architecture decisions in the design of critical ML-enabled systems. The aim is to provide starting points for a future engineering research into the use of runtime monitors for critical ML systems and their relationship to the architecture of critical ML-enabled software systems. Specifically, the original paper was extended by

- a new research question (RQ3) on the relation between challenges when specifying requirements for training data and runtime monitors and architecture decision of ML-enabled software systems,
- as an answer to RQ3 a list of identified relationships between the challenges identified in the previous study and architecture decisions in the design of critical ML-enabled software systems,
- a deeper exploration of the rationale behind the relationships between the identified challenges of specifying requirements for training data and runtime monitors for ML-enabled software systems,
- the result of a survey to validate the recommendations and implications for RE derived from the qualitative results of both studies.

The following research questions guided the previous study (RQ1, RQ2) and the extension reported in this article (RQ3):

RQ1: What are challenges encountered by practitioners when specifying requirements for training data of ML models in safety critical software?

RQ2: What are challenges encountered by practitioners when specifying requirements for runtime monitors especially in relation to fulfilling safety requirements?

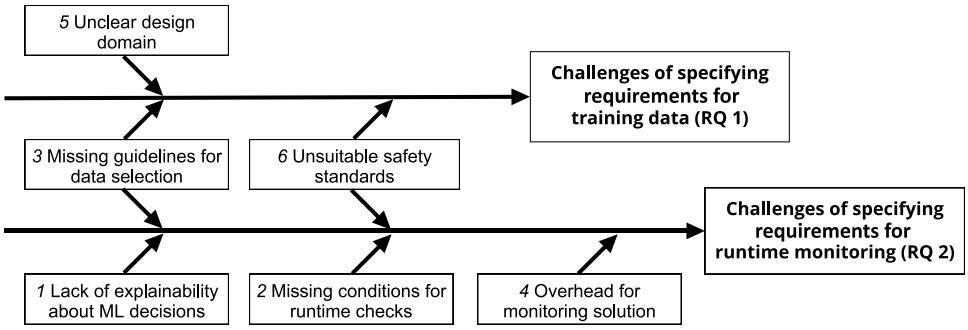

Fig. 1 Cause-effect diagram providing an overview of identified challenge groups

RQ3: How can an architecture framework provide guidance to practitioners with respect to the challenges identified in RQ1 and RQ2?

Figure 1 shows the main themes we found in answering the research questions. Concerning RQ1, the interviewees reported on several problems: the data selection process is nontransparent and guidelines especially towards defining suitable measures for data variety are missing. There are no clear context definitions that help in defining data needs, and current safety standards provide little guidance. Concerning RQ2, we found that the problem of defining suitable metrics and the lack of guidance from safety standards also inhibits the ability to specify runtime monitors. Practitioners also reported on challenges regarding explainability of ML decisions. Additionally, they reported that the processing and memory of runtime monitors in safety critical embedded systems can pose an additional challenge when specifying requirements for these monitors. Regarding RQ3, we found parallels between the identified challenges in this study and challenges in architecting ML-enabled systems identified in a previous study. This paper discusses how the relationships between the challenges translate into correspondences between architectural views that can provide guidance for overcoming the challenges.

The remaining sections of this paper are structured as follows: Sect. 2 outlines and argues for the research methods of this study; Sect. 3 presents the results and answers to the research questions; Sect. 4 discusses the findings, provides recommendations to practitioners and for further research, identifies related literature, elaborates on threats to validity, and provides a conclusion.

2 Research method

We used a qualitative interview-based survey with open-ended semi-structured interviews to collect data, which we then extended with constructive work on finding relationships between the challenges and mapping architecture concerns to the challenges. Following the suggestions of Creswell and Creswell [21] the qualitative study was conducted in four steps: Preparation of interviews, data collection through interviews, data analysis, and result validation.

2.1 Preparations of interviews

Based on the a-priori formulated research questions, two of the researchers of this study created an interview guide[2] which was validated and improved by the remaining two researchers. The interview guide contains four sections of questions: The first section includes questions about the interviewees' current role, background and previous experiences. The second section focuses on questions that try to understand challenges when specifying and selecting training data for ML models and how training data affect the performance of these models. The third section investigates challenges when ML models are incorporated in critical systems and how they affect the ability to specify training data. The fourth section concentrates on the run time monitoring aspect of the ML model and contains questions on challenges when specifying runtime monitors.

Sampling strategy We chose the participants for this study purposefully using a maximum variation strategy [22]. We were able to recruit interviewees from five different companies, ranging from a local start-up to a multinational world leading communication company. An overview is given in Table 1.

Table 1 Companies participating in the study

Company	Area of operations	Employees	Countries
1	Telecommunication networks	> 10.000	World
2	Automotive OEM	> 10.000	World
3	Automatic Driving	> 1.000	Europe
4	Industrial camera systems	> 1000	USA
5	Deep Learning optimisation for IoT	> 100	Sweden

[2] The interview guide is available in the replication package for this paper. The link is provided at the end of the paper.

Table 2 Participants of the study

Interview	Role	Experience
A	Researcher (Academic)	Functional Safety for ADAS (5 years)
B	Function developer	Sensor and perception systems (20 years)
C	Principal engineer	ML model integration (10 years)
D	ML model developer	Distributed and edge systems (3 years)
E	Function owner	ADAS perception functions (8 years)
F	Function developer and test engineer	Automatic driving systems (25 years)
G	Data Scientist	Distributed systems (12 years)
H	Requirement Engineer	Perception systems (8 years)
I	Researcher (Academic)	Neural Network development (8 years)
J	Functional Safety Manager	Sensor systems (20 years)

ADAS: Advanced Driver Assistance Systems

A selection criteria for the company was that they must work with safety-critical systems and ML. Within the companies we tried to find interview candidates with different roles and work experiences to obtain a view beyond the developers' perspective. Besides function developers, who are mostly responsible for the code development of conventional software functions, and ML model developers, we were interested in interviewing requirement engineers and product / function owners, who take a mediator role between the customer and the function developers, because they represent key roles in deriving system or function specifications. We provided the companies with a list of roles that we identified beforehand as interesting for interviewing. The list included functional safety experts, requirement engineers, product owners or function owners, function or model developers, and data engineers. Additionally, we interviewed two researchers from academia who participate in a joint industry EU Horizon 2020 project called VEDLIoT.[3] The aim of VEDLIoT is to develop a toolchain to enable efficient deep learning in distributed systems [47]. Both researchers worked also with ML models in industry before. Therefore, they could provide insights into both the academic and the industry perspective. A list of the ten interviewees for this study is provided in Table 2.

2.2 Data collection through interviews

All interviews were conducted remotely using either the conference software Zoom or Microsoft Teams and took between 60 - 90 min. The a-priori defined interview guide was only available to the interviewers and was not distributed to the participants beforehand. Each participant was interviewed by two interviewers who alternated in asking questions and observing. At the start of each interview, the interviewers provided some background information about the study's purpose. Then, the interview guide was followed. However, as we encouraged discussions with the interviewees, we allowed deviations from the interview guide by asking additional questions, or changing the order of the questions when it was appropriate [50]. All interviews were recorded and semi-automatically transcribed. The interviewers manually checked and anonymised the results.

2.3 Data analysis of interviews

The data analysis followed suggestions by Saldana [71] and consisted of two cycles of coding and validation of the themes through focus groups and member checking.

First coding cycle Attribute coding was used to extract information about the participants' role and previous experiences. Attribute coding is also known as *descriptive coding* in which meaningful and reoccurring attributes through the interviews are identified [69]. Attributes can for example be "role of interviewee in company", "experience", etc. Afterwards, the two interviewers independently applied structural coding to collect phrases in the interviews that represent topics relevant to answering the research questions. Structural coding can be used to label and index data relevant to a particular analysis (in our case research question) [60]. The researchers compared the individually assigned codes and applied descriptive coding with the aim of identifying phrases that describe common themes across the interviews.

Theme validation In a focus group, the identified themes were presented and discussed. Thirteen researchers from both industry and academia in the VEDLIoT project participated. Three of the participants from the workshop were also interviewed for this study. We asked the participants in two initial questions about their professional background and years of relevant experience in their role. Five participants answered they have an industrial background, six an academic background and two mentioned they have both an academic and industry background. The average years

[3] Very Efficient Deep Learning in the Internet-of-Things https://www.vedliot.eu.

of experience were 12.5 years, with a standard deviation of 10.4 years. Three participants work in the area of research and development (R &D) for ML systems in a company, six in academic research, two described their role as safety engineers, one as software developer, and one as engineering manager. The aim of the focus group was to reduce bias in the selection of themes and to identify any additional themes that the researchers might have missed. Each theme was presented to the group and openly discussed. The participants used their work experience in their respective companies to relate to the themes. Then, the participants were asked to openly discuss if any further themes should be added to the data analysis.

Second coding cycle After the themes were identified and validated, the second coding cycle was used to map the statements of the interviewees to the themes, and consequently identify the answers to the research questions. The second cycle was conducted by the two researchers who did not conduct the first cycle coding in order to reduce confirmation bias. The mapping was then confirmed and agreed upon by all involved researchers.

Validation of challenge themes Member checking, as described in [22] was used to validate the identified themes that answer RQ1 and RQ2. While performing the second coding cycle, we returned to some of the interviewees and shared the preliminary mapping of quotes to themes. This served two purposes: It created a feedback loop in the research process allowing us to involve practitioners with their perspective and experience in the mapping of the statements to the themes. Member checking also allowed us to validate that the we interpreted the interviewees' statements correctly by obtaining additional opinions. Finally, we also presented the results of the interview study in a 60 min focus group to an industry partner of VEDLIoT and allowed for feedback and comments on the conclusions we drew from the data through an open discussion.

Validation of RE implications and recommendations to practitioners We validated identified implications for Requirement Engineering (RE) and recommendation for practitioners based on the identified themes through a survey of six RE experts. The participants were selected randomly by visiting a project meeting of another research project on RE for automotive perception systems called *"Facilitating Multi-Party Engineering of Requirements"* (FAMER) [82]. We chose FAMER because the research project is strongly related to RE challenges encountered by industrial practitioners in the design of ML-enabled safety-critical perception systems for the automotive industry. Therefore, we assumed that FAMER participants are well qualified to judge both the implications for RE and the recommendations derived from the results of this study. Table 3 provides an overview of the participants' roles and experience. One of six participants answered to work in academia. Two participants stated they both work in academia and industry, and three participants work in industry. We visited a project meeting of FAMER on 24th November 2023 and asked all participants of that meeting to fill out the survey in Microsoft Forms. Besides asking for the participant's role and experience, the survey listed all implications for RE and recommendations. The participants were asked to express, based on their experience, their agreement on a Likart-scale from 1 (Strongly Disagree), to 5 (Strongly Agree). The survey itself and its results are included in the replication package.

Table 3 Participants of validation survey

ID	Role	Experience
I	Systems engineer	>10 years
II	Researcher (Academic)	>10 years
III	Researcher ML	0–1 year
IV	Safety researcher	5–10 years
V	Requirement engineer	>10 years
VI	Requirement engineer	>10 years

2.4 Data collection for constructive research

The data collection for the constructive elements of this study, i.e., the mapping of architecture views to the identified challenges entailed two components. First, we revisited data collected for a previous study conducted on applying architecture frameworks to distributed AI systems [32]. The data include recordings from two workshops on challenges of developing distributed AI-enabled systems in an industrial case study which was also part of VEDLIoT. The two workshops aimed at identifying and validating through literature and standards relevant challenges in relation to defining architectures for ML systems that are part of the IoT. In each of the workshops, eleven experts from VEDLIoT participated, whereof four identified themselves as ML expert working in industry, one as IoT expert in industry, and six as researchers from academia [32, Appendix A]. The outcome of both workshops is a list of ten themes describing the identified 28 challenges of developing distributed ML systems within VEDLIoT. For this study, we reused the themes "#1 Additional views needed for describing the AI model", "#2 Data requirements", "#3 Context and design domain", and "#8 Run Time Monitoring" as they correspond to the challenges identified in this study. Second, we collected data on architecture views suitable for designing AI systems from the architecture framework for VEDLIoT described in [32]. The architecture framework for VEDLIoT consists of 52 architecture views which relate to existing architecture standards and literature. We chose to extract architecture views from the VEDLIoT framework because it constitutes

a comprehensive and peer-reviewed collection of architecture views for AI system development. The framework was built upon relevant architecture standards and literature including, among others, the IEEE 2413 standard for architectural frameworks for the IoT [36], the IEEE 42010 standard on architecture descriptions [40], the collected work on documenting software architecture by Clements et al. [20], and the exploratory work by Muccini and Vaidhyanathan on software architecture for ML-based systems [59]. We included the data from the previous study on architecture frameworks for ML-based distributed systems because some of the identified challenges in the previous study correspond to the challenges identified from a different set of practitioners in this study. However, the scope of the study in [32] was to offer a system-holistic view on ML-enabled systems, including for example hardware and communication aspects. Instead, this study looked more closely at the challenges of specifying requirements for training data and runtime monitoring. The intention here is to investigate, if an architecture framework can also help to overcome the extended set of challenges for the two areas of training data and runtime monitors identified in this study.

2.5 Data analysis for constructive work

The aim of this constructive work is to propose the use of architecture thinking as part of a design thinking approach to provide guidance in how to solve the identified challenges with specifying requirements for the training data and runtime monitoring for critical ML enabled systems. Design thinking has been proposed as solution for RE and software development challenges for example in [79] and [46]. The relation between design thinking and architecture frameworks for software development has been investigated in [25] which describes that architecture framework support design thinking by constraining the complexity of the design tasks and by providing design links between different aspects of the system.

Design Thinking typically involves five phases: *Empathise*, in which one seek understanding of a problem from a stakeholder's perspective, *Define*, in which the problem is more clearly defined by synthesising information gathered in the Empathise phase, *Ideate*, in which a range of ideas are developed to address the defined problems, *Prototype*, in which first ideas are implemented, and *Test*, in which the prototypes are given to the original stakeholders to gather feedback.

By interviewing practitioners and identifying challenge themes and groups, the original study fulfilled the *Empathise* and *Define* phases. In the extension of the original study, we aim to address the *Ideate* phase by exploring the idea that architecture frameworks can potentially be used to mitigate the problem of specifying AI-specific requirements such as training data and runtime monitors. The *Prototype* and *Test* phase is outside the scope of this study.

Mapping of challenges to previously identified challenges First in the *Ideate* phase, we reviewed the data that led to the creation of what is referred to as "clusters of concern" based on data collected from [32], with a focus on clusters of concerns that relate to the challenges identified in this study as described earlier. In line with the IEEE 2413-2019 standard [36], a concern refers to an interest, problem, or requirement that is relevant to one or more stakeholders of the system. We used the IEEE 2413-2019 standard for architecture frameworks in the IoT as starting point because similar to the development of ML-based systems, systems for the IoT depend strongly on data availability and data quality aspects. Other similarities are data privacy concerns, security, safety and other qualitative system concerns, as well as the problem of interoperability and integration because ML components, similar to elements of an IoT system, need to be integrated with existing software and hardware systems [76]. A cluster of concern extends this definition by grouping concerns into clusters that represent a specific concern of the system at different levels of details. For example, the cluster of concern *data strategy* includes concerns about *data ingestion*, *data selection*, and *data preparation and preprocessing* [32, Appendix F]. By applying ideas from category theory, the architecture framework proposed in [32] consists of a grid of architecture views sorted vertically by their level of abstraction. Views on the same level of abstraction should have an equivalent level of detail about the system-of-interest. If the product over all architectural views on a level of abstraction is valid, then the architectural views consistently describe the system-of-interest. Valid means that no matter which architectural view one starts at, it is always possible to transit via the correspondence rules to another architectural view and still see the same system from a different perspective. Horizontally, the architecture views address the aforementioned clusters-of-concerns. This horizontal and vertical arrangement of architecture views is referred to as *compositional architecture framework*. For the constructive work, we mapped the identified challenges to architecture views within the compositional architecture framework. An example of such an cluster-of-concerns over level of abstraction arrangement of architecture views is provided in Fig. 2. The figure will be discussed in more detail in the result section of this article.

We compared the challenges identified in RQ1 and RQ2 of this study with the challenges outlined in the architecture framework described in [32]. We looked for similar concerns in the statements of the interviewees of both studies and mapped them accordingly to each other. Similar here means that the concerns address the same problem in relation to specifying data and runtime monitors. For example, if an interviewee mentioned that explainability of decisions

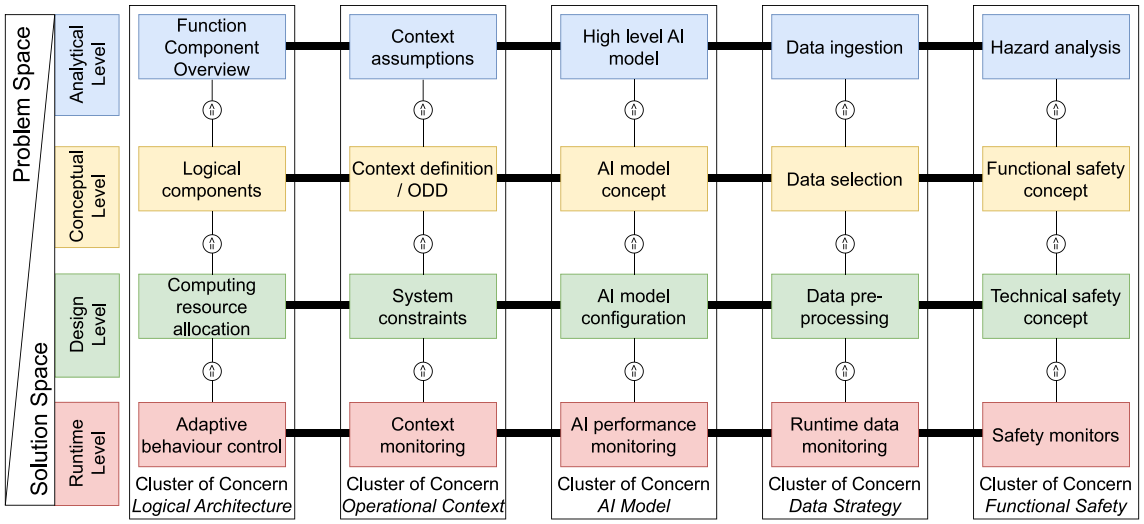

Fig. 2 Architecture co-evolution of logical architecture, AI model architecture, data strategiy, and safety concerns of the system

at runtime is a challenge, it is similar to the challenge of including new quality aspects such as explainability and fairness as architecture views for ML systems. Then, any concerns not mentioned in [32] were mapped to appropriate architecture views or clusters-of-concerns based on the researchers' expert knowledge. The researchers provided justification based on logical reasoning and the researchers' experience for why the chosen mappings were appropriate.

Identifying relations between challenges and architecture views In a second step of the constructive work, we manually performed a semantic analyses of the interview data we collected for this study. The aim was to find connections, or more specifically correspondences between the challenges. For example, the challenge of ensuring explainability of ML models might be an effect of the challenge of IP protection, which causes opacity in the development process and therefore an inability to create explainable ML models. After we identified these relations between the challenges, we created correspondences between the architecture concerns extracted from the compositional architecture framework. The hypothesis behind these correspondences is that if the challenges addressed by the architecture views are related, then these architecture views should also be related by correspondence rules.

3 Results

During the first coding cycle, structural coding resulted in 117 statements for RQ1 and 77 statements for RQ2. Through descriptive coding 14 preliminary themes were found. In descriptive coding, the researcher summarises in a short phrase the theme of a statement from the interviewee [71, page 88]. The statements found through structural coding and preliminary themes identified through descriptive coding were then discussed during a focus group. As a result of the focus group discussion, three additional themes were created (IDs 3.3, 4.3, and 6.3). Based on the feedback from practitioners, the 117 statements for RQ1 were categorised first into eight final challenge themes (IDs 3.1−3.4, 5.1−5.2, 6.1−6.2) and those themes were further categorised into three challenge groups (IDs 3, 5, 6) relating to the challenge of specifying requirements for training data. Similarly, the 77 original statements for RQ2 were grouped into twelve final challenge themes (IDs 1.1−1.3, 2.1−2.2, 3.4, 4.1−4.3, 6.1−6.3) and further into five challenge groups (IDs 1, 2, 3, 4, 6) relating to the challenge of specifying runtime monitoring. One challenge group relates exclusively to RQ1 (ID 5), three groups relate to RQ2 (IDs 1, 2, 4), and two groups relate to both RQs (IDs 3, 6). The categories and final challenge themes are listed in Table 4. Additionally, for each challenge theme, we indicate the implication of the findings for RE.

3.1 Answer to RQ1: Challenges practitioners experience when specifying requirements for training data

The interviewees were asked to share their experiences in selecting training data, the influence of the selection of training data on the system's performance and safety, and any experiences and thoughts on defining specifications for training data for ML. Based on the interview data, we identified three challenge groups related to specifying requirements for training data: missing guidelines for data selection, unclear design domain, and unsuitable safety standards, which are elaborated below.

Table 4 Challenge groups (bold) and themes found in the interview data. Data.: Challenges related to specifying training data (RQ1). Monitor.: Challenges related to specifying runtime monitoring (RQ2). The groups and themes are alphabetically ordered

ID	Challenge theme	Relates to Data	Relates to Monitor.	Related Literature
1	**Lack of explainability about ML decisions**		✓	
1.1	No access to inner states of ML models		✓	[27]
1.2	No failure models for ML models		✓	[86]
1.3	Protection of IP		✓	
2	**Missing conditions for runtime checks**		✓	
2.1	Unclear metrics and/or boundary conditions		✓	[15, 30, 73]
2.2	Unclear measure of confidence		✓	[26, 55]
3	**Missing guidelines for data selection**	✓	✓	
3.1	Disconnection from requirements	✓		[24, 72]
3.2	Grown data selection habits	✓		[5, 29]
3.3	Unclear completeness criteria	✓		[84]
3.4	Unclear measure of variety	✓	✓	[77, 85]
4	**Overhead for monitoring solution**		✓	
4.1	Limited resources in embedded systems		✓	[67]
4.2	Meeting timing requirements		✓	
4.3	Reduction of true positive rate		✓	
5	**Unclear design domain**	✓		
5.1	Design domain depends on available data	✓		[9]
5.2	Uncertainty in context	✓		[33]
6	**Unsuitable safety standards**	✓	✓	
6.1	Focus on processes instead of technical solution	✓	✓	[13]
6.2	No guidelines for probabilistic effects in software	✓	✓	[48, 73]
6.3	Safety case only through monitoring solution		✓	[51, 78]

ID3: Missing guidelines for data selection Four interviewees reported on a lack of guidelines and processes related to the selection of training data. A reason can be that data selection bases on “grown habits" that are not properly documented. Unlike conventional software development, the training of ML is an iterative process of discovering the necessary training data based on experience and experimentation. Requirements set on the data are described as disconnected and unclear for the data selection process. For example, one interviewee stated that if a requirements is set that images shall contain a road, it remains unclear what specific properties this road should have. Six interviewees described missing requirements on the data variety and missing completeness criteria as a reason for the disconnection of requirements from data selection.

> *For example, we said that we shall collect data under varying weather conditions. What does that mean?* - Interview B, Q1

> *How much of it (the data) should be in darkness? How much in rainy conditions, and how much should be in snowy situations?* - Interview F, Q2

Another interviewee stated that it is not clear how to measure variety, which could be a reason why it is difficult to define requirements on data variety.

> *What [is] include[d] in variety of data? Is there a good measure of variety?* - Interview A, Q3

RE Implication 1: **RE research should uncover new ways to specify variety and completeness criteria for data collection.** *Four validation survey participants strongly agreed, and two agreed with RE Implication 1.*

ID5: Unclear design domain Three interviewees describe uncertainty in the design domain as a reason for why it is difficult to specify training data. If the design domain is unclear, it will be challenging to specify the necessary training data.

> *We need to understand for what context the training data can be used.* - Interview J, Q4

> *ODD [(Operational Design Domain)]? Yes, of course it translates into data requirements*. - Interview F, Q5

RE Implication 2: **RE research must provide better ways to specify the context, since data selection and**

completeness criteria depend on it. *Five validation survey participants strongly agreed, and one agreed with RE Implication 2.*

ID6: Unsuitable safety standards Because we were specifically investigating ML in safety critical applications, we asked the participants if they find guidance in safety standards towards specifying requirements for training data. Five interviewees stated that current safety standards used in their companies do not provide suitable guidance for the development of ML models. While for example ISO 26262 provides guidance on how to handle probabilistic effects in hardware, no such guidance is provided for software related probabilistic faults.

> *The ISO 26262 gives guidance on the hardware design; [...] how many faults per hour [are acceptable] and how you achieve that. For the software side, it doesn't give any failure rates or anything like that. It takes a completely process oriented approach [...].* - Interview C, Q6

One interviewee mentioned that safety standards should emphasise more the data selection to prevent faults in the ML model due to insufficient training.

> *To understand that you have the right data and that the data is representative, ISO 26262 is not covering that right now which is a challenge.* - Interview H, Q7

RE Implication 3: **RE methods and practices are needed to operationalise safety standards for the selection of training data.** *Three validation survey participants strongly agreed, and three agreed with RE Implication 3.*

Besides Implication 3 towards RE, an implication towards standardisation committees responsible for the formulation of safety standards could be to revise these standards or create new standards that are more compatible with the developing processes of ML-enabled systems. Example of emerging safety standards for AI are ISO/IEC 5469: Functional Safety and AI Systems [44], and ISO/PAS 8800: Road Vehicles Safety and Artificial Intelligence [42]. An overview of emerging safety standards for AI-enabled systems is given in [4].

3.2 Answer to RQ2: Challenges practitioners experience when specifying runtime monitors

We asked the interviewees on the role of runtime monitoring for the systems they develop, their experience with specifying runtime monitoring, and the relation of runtime monitoring to fulfilling safety requirements on the system. We identified five challenge groups related to runtime monitoring: lack of explainability about ML decisions, missing conditions for runtime checks, missing guidelines for data selection, overhead for monitoring solution, and unsuitable safety standards, as explained below.

ID1: Lack of explainability about ML A reason why it is difficult to specify runtime monitors for ML models is the inability to produce failure models for ML. In normal software development, causal cascades describe how a fault in a software components propagates trough the systems and eventually leads to a failure. This requires the ability to break down the ML model into smaller components and analyse their potential failure behaviour. Four interviewees however reported that they can only see the ML model as a "black-box" with no access to the inner states of the ML model. As a consequence, there is no insight into the failure behaviour of the ML model.

> *[Our insight is] limited because it's a black box. We can only see what goes in and then what comes out to the other side. And so if there is some error in the behavior, then we don't know if it's because [of a] classification error, planning error, execution error?* - Interview F, Q8

The reason for opacity of ML models is not necessarily due to technology limitations, but also due to constraints from protection of intellectual property (IP).

> *Why is it a black box? That's not our choice. That's because we have a supplier and they don't want to tell us [all details].* - Interview F, Q9

RE Implication 4: **RE can play a crucial role in navigating the trade-off between protecting IP of suppliers and sharing enough information to allow for safety argumentation.** *Two validation survey participants strongly agreed, two agreed, one answered neutrally, and one disagreed with RE Implication 4.*

ID2: Missing conditions for runtime checks A problem of specifying runtime monitors is the need for finding suitable monitoring conditions. This requires the definition of metrics, goals and boundary conditions. Five interviewees report that they face challenges when defining these metrics for ML models.

> *What is like a confidence score, accuracy score, something like that? Which score do you need to ensure [that you] classified [correctly]?* - Interview F, Q10

Especially the relation between correct behaviour of the ML model and measure of confidence is unclear, and therefore impede runtime monitoring specification.

> *We say confidence, that's really important. But what can actually go wrong here?* - Interview J, Q11

RE Implication 5: **RE is called to provide methods for identifying conditions for runtime checks.** *Four*

validation survey participants strongly agreed, and two agreed with RE Implication 5.

ID3: Missing guidelines for data selection The inability to specify the meaning of data variety also relates to missing conditions for runtime checks. For example, runtime monitors can be used to collect additional training data, but without a measure of data variety it is difficult to find the required data points.

> *You mean [finding sufficient data] by using a measure of variety? Yeah, that is extremely difficult. I don't really have an answer.* - Interview A, Q12

ID4: Overhead for monitoring solution An often overlooked problem seems to be the induced (processing) overhead from a monitoring solution. Especially in the automotive sector, many software components run on embedded computer devices with limited resources.

> *You don't have that much compute power in the car, so the monitoring needs to be very light in its memory and compute footprint on the device, maybe even a separate device that sits next to the device.* - Interview F, Q13

And due to the limited resources in embedded systems, monitoring solutions can compromise timing requirements of the system. Additionally, one interviewee reported concerns regarding the reduction of the ML model's performance.

> *[...] the true positive rate is actually decreasing when you have to pass it through this second opinion goal. It's good from a coverage and safety point of view, but it reduces the overall system performance.* - Interview F, Q14

RE Implication 6: **RE methods are needed to help finding suitable runtime checks that do not negatively impact the performance of the system.** *Three validation survey participants strongly agreed, one agreed, and two answered neutrally with RE Implication 6.*

ID6: Unsuitable safety standards Safety standards are mostly not suitable for being applied to ML model development. Therefore, safety is often ensured through non-ML monitoring solutions. Interviewees reported that it is not a good solution to rely only on the monitors for safety criticality:

> *[...] so the safety is now moved from the model to the monitor instead, and it shouldn't be. It should be the combination of the two that makes up safety.* - Interview B, Q15

One reason is that freedom of inference between a non-safety critical component (the ML model), and a safety critical component (the monitor) must be ensured which can complicate the system design.

> *And then especially if you have mixed critical systems [it] means you have ASIL [(Automotive Safety Integrity Level)] and QM [(Quality Management)] components in your design [and] you want to achieve freedom from interference in your system. You have to think about safe communication and memory protection.* - Interview J, Q16

RE Implication 7: ***RE is called to provide traceability and requirements information models that allow a complete description of the system, its monitors, and their relationship to high-level requirements (such as safety).*** *Four validation survey participants strongly agreed, one agreed, and one disagreed with RE Implication 7.*

3.3 Answer to RQ3: How can an architecture framework provide guidance to practitioners with respect to the challenges identified in RQ1 and RQ2?

This section identifies architecture principle that can potentially alleviate challenges identified in RQ1 and RQ2. In order to reflect the relationships identified between the challenges of specifying requirements for training data and runtime monitoring, we outline the specific correspondences that should exist within the architecture. For example, relation R6, outlined in Table 6, describes how missing guidelines for data selection are related to the inability of defining runtime checks for the ML model. If guidelines for data selection are represented in the system architecture, for example through a data pipeline view, and runtime checks for example through an architecture view describing the software components for these runtime monitors, then those two views must be related through correspondence rules.

Mapping of challenges to architecture clusters of concerns and views As answers on RQ1, which asked about challenges encountered when specifying requirements for training data, and RQ2, which asked about challenges encountered when specifying runtime monitors in relation to especially safety requirements, we identified 17 challenges in six challenge groups regarding the specification of training data and runtime monitors for safety critical software that contains ML components. In a series of workshops in 2020 and 2021 challenges that hinder developers in the development of systems with ML-components were identified and

Table 5 Mapping of challenges identified in this study to challenges and mitigation strategies identified in [32]

Identified challenge group	Related challenge in [32]	Related architecture principle [32]
1: Lack of explainability about ML decisions	#7: New quality aspects (e.g., explainability, fairness)	Explicitly integrate explainability in the architecture framework as cluster of concern. Thereby, explainability is treated like any other architecture design decision and allows for an“explainable-by-design" approach
2: Missing conditions for runtime checks	#8: Runtime monitoring	Runtime concerns such as monitoring can be made explicit with an own level of abstraction containing runtime specific architecture views
3: Missing guidelines for data selection	#2: Data requirements for ensuring the desired AI's behaviour must be considered	Data concerns that have direct impact on the system's behaviour, such as training data for the AI model, can be considered an own cluster-of-concern with correspondences to views describing the operational context, the AI model, and the functional behaviour of the system
5: Unclear design domain	#3: Description of context and design domain	The context and operational design domain should be considered own clusters-of-concern as they form many correspondences e.g., to views describing the AI model, the data need, and non-functional aspects such as safety

an architecture framework was proposed that could mitigate these challenges for such software [32]. As a first step in answering RQ3, we reviewed the data collected from these workshops[4] and compared challenges identified with those identified in this study. For all matches we found, we identified the proposed mechanism in the architecture framework that was suggested as mitigation strategy. Table 5 shows the result of the mapping of the identified challenge groups to cluster of concern, and subsequently architecture views.

Two challenge groups identified in RQ1 and RQ2, *4: Overhead for monitoring solution*, and *6: Unsuitable safety standards*, have no matching cluster of concern in [32]. This meant, that we could not directly identify correspondences between related architecture views for these challenges. However, the problem of handling the (resource) overhead needed for monitoring solution can possibly be solved by establishing clear correspondence between monitoring needs and the remaining software architecture. If monitoring is not added “on-top" of the system, but instead considered throughout the design phase, or even becomes an explicit part of the system architecting effort (see for example [14]), the problem of surprising overhead resource requirements should not occur.

In addressing the issue of an inadequate safety standard, an architectural approach alone does not offer a direct solution. However, it is possible to explicitly recognising safety as a crucial aspect of the system by considering it as own cluster-of-concern in an architecture framework. This can be accomplished through the inclusion of dedicated architecture views within the proposed framework. These views facilitate for example safety decomposition and fault tree analyses. Block diagrams allocating functional safety requirements to elements of the systems are an example of architecture views describing the functional safety concept of the system [62]. Compliance with various safety standards, such as ISO 26262 [41] or ISO/CD TS 5083 [43], often requires the establishment of traceability between safety goals and design decisions [75]. To meet this requirement, correspondence rules can be established between the safety-related views of the architecture and other relevant architecture views that might be development by other teams.

Interrelation between challenges The results relating to RQ1 and RQ2 reveal connections between the challenges. For example, the themes in the challenge groups *unsuitable safety standards* and *missing guidelines for data selection* relate to both challenges of specifying requirements for training data and runtime monitoring. Furthermore, we identified relations between different themes and across different group of themes. For example **R1a:** *IP protection* relates to the *inability of accessing the inner states* and **R1b:** for *creating failure models for ML model*. We based this assessment on a manual semantic analyses of the words used in the statements relating to these themes. For example, Interviewee F stated that:

[4] A replication package containing data from these workshops is available. The link is provided at the end of the paper.

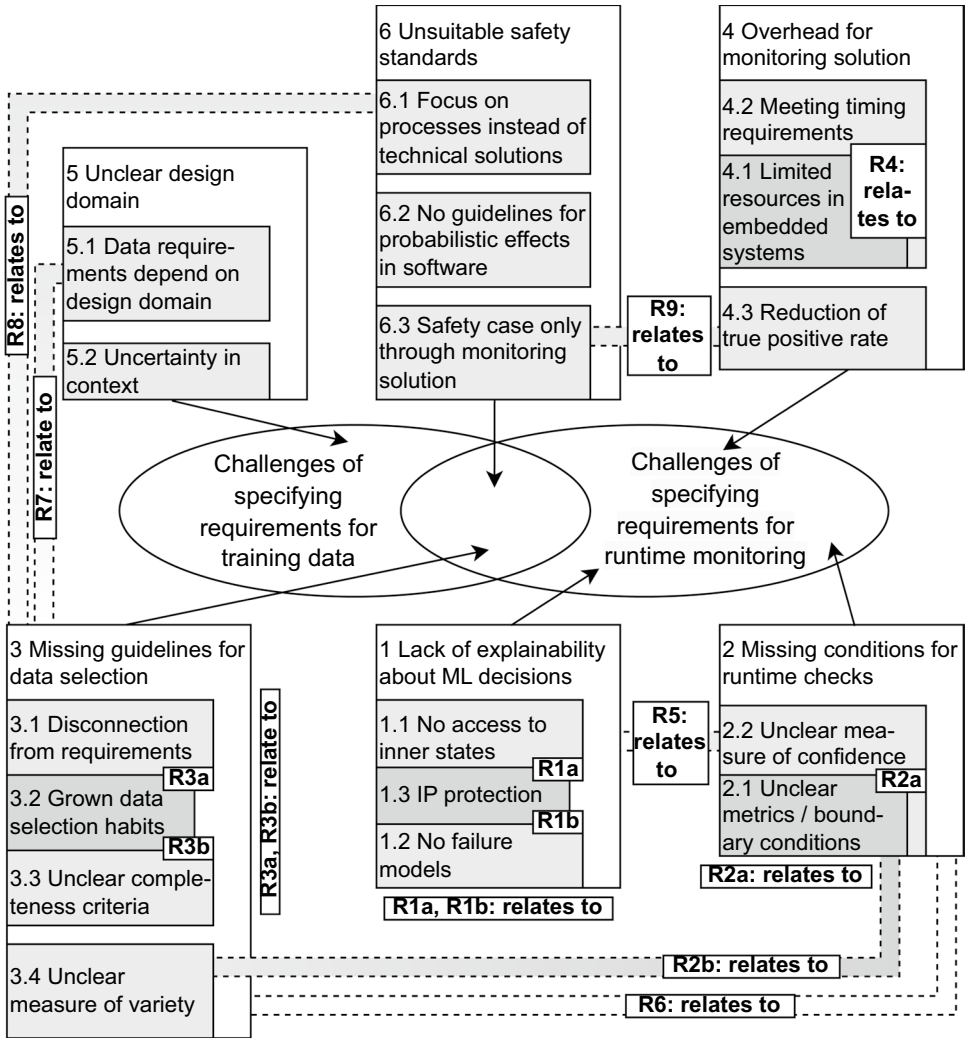

Fig. 3 Relations between the identified challenge themes and groups. Enclosed themes have been identified as causes for the surrounding themes. Furthermore, dotted lines indicate relations between different challenge themes and groups

That neural network is something [of a] black box in itself. You don't know why it do[es] things. Well, you cannot say anything about its inner behavior - Interview F

Later in the interview, the same participants states:

Why is it a black box? That's not our choice. That's because we have a supplier and they don't want to tell us [all details]. - Interview F

Fig. 3 illustrates the identified cause-effect relations, relations between the themes, and how the different themes relate to the challenges.

Further relations are: **R2a:** *Unclear metrics and/or boundary conditions* is related to *unclear measures of confidence* and **R2b:** *unclear measures of variety*. **R3a:** *Grown data selection habits* relates to a *disconnection from requirements* and **R3b:** *unclear completeness criteria*. And, **R4:** *limited resources in embedded systems* relates to the challenge of *meeting timing requirements* when using runtime monitors. We also found several statements that indicate relations between themes of different challenge groups. The **R5:** *lack of explainability about ML decisions* relates to the *inability to understand and trust the measure of confidence*. **R6:** *Missing guidelines for data selection* relate to the *ability to define runtime checks*. **R7:** *Unclear design domain, and the dependency of data requirements on the design domain* relate to *missing guidelines for data selection*. **R8:** *The focus on processes instead of technical solutions* relate to *missing guidelines for data selection*. **R9:** *Achieving a safety case only through monitoring solution* relate to *reduction of true positive rate*. Table 6 lists all links found between the challenges, their IDs, and it provides rationale for the relationships.

Relations between architecture views based on relations between requirement specification challenges Six of the relations listed in Table 6 involve challenges from different challenge groups. For example, **R2b: Unclear metrics and/or boundary conditions are related to unclear measures of variety** relates to challenge **2.2: Unclear measure**

Table 6 Identified relations between challenges and rationales for these relations

ID	Relation between challenges
R1a	**IP protection is related to the inability of accessing the inner states. (1.1,1.3)**
Rationale: Deep neural networks require extensive development and often also financial investments. Supplier companies can therefore consider their ML models as valuable intellectual property. Access to the inner workings of the models can be restricted by companies to protect their competitive advantage. This is typically done through non-disclosure agreements or limited sharing with external parties. Related quote: Q9	
R1b	**IP protection is related to the inability of creating failure models for ML model. (1.2, 1.3)**
Rationale: IP protection can restrict access to proprietary data that was used to train the models. The inability to access and inspect the training data can limit the ability to understand and analyse failure cases. Furthermore, IP protection may prevent access to the model architecture, the training procedures, and hyper-parameter settings. Without this knowledge, it can be difficult to impossible to identify and understand the causes of failures. Related quotes: Q8, Q9	
R2a	**Unclear metrics / boundary conditions are related to unclear measures of confidence. (2.1, 2.2)**
Rationale: Confidence estimates provide information about the level of certainty of a ML model's predictions. The absence of well-defined metrics can lead to ambiguity in assessing the correctness and confidence of the model's output. Possible metrics are for example precision, F1-score, recall, and intersection over unit (IoU). In image classification and object detection tasks, accuracy is used as a typical metric to measure the model's performance. It is challenging to relate confidence estimates to the actual performance of the model without clearly defined metrics. Related quote: Q10	
R2b	**Unclear metrics / boundary conditions are related to unclear measures of variety. (2.2, 3.4)**
Rationale: Similarly to R2a, Assessing the diversity or representativeness of a dataset used for training of ML models will be challenging or even impossible without clearly defined metrics that determine whether the dataset adequately covers the target distribution and captures the necessary variations in the target. Related quotes: Q3, Q11, Q12	
R3a	**Grown data selection habits are related to a disconnection from requirements. (3.1, 3.2)**
Rationale: Data collection is influenced by factors such as data availability, convenience, and historical practices. If data selection habits are driven mostly by convenience or historical practices, there is a risk of disconnection from the actual requirements of the task. To mitigate the disconnection from the system's requirements, it is important to adopt data selection strategies to ensure that the data cover the desired variations and address specific challenges based on higher level stakeholders' requirements. Related quotes: Q1, Q2	
R3b	**Grown data selection habits are related to unclear completeness criteria. (3.2, 3.3)**
Rationale: If there is no clear understanding of the required variations, diversity, or representativeness of the data due to the rationale provided in R3a, defining completeness criteria for a dataset can be challenging too. Unclear completeness criteria can lead to biases in the dataset or omission of critical scenarios which can impact the performance of the trained ML model. Related quote: Q2	
R4	**Limited resources in embedded systems are related to challenges of meeting timing requirements when using runtime monitors. (4.1, 4.2)**
Rationale: Runtime monitoring requires additional processing and memory resources, that are then not available for model inference. This can reduce the speed of inference due to less available computing resources for the ML model's inference task especially in embedded automotive systems with limited computational resources. Related quotes: Q13, Q14	
R5	**Lack of explainability about ML decisions is related to the inability to understand and trust the measure of confidence. (1, 2.2)**
Rationale: If the model's decision-making process is not transparent or explainable, it becomes challenging to trust the associated measures of confidence. Therefore, when developers and/or users can understand how and why the model arrived at certain decisions, they can evaluate the reasoning and assess the reliability of the model's outputs, including the measures of confidence. A lack in explainability can compromise trust in the model's confidence estimates. Related quotes: Q8, Q11	
R6	**Missing guidelines for data selection is related to the inability to define runtime checks. (2, 3)**
Rationale: Data selection guidelines should provide guidance in fulfilling data requirements. However, without data requirements, criteria or recommendations for data selection are difficult to define. This makes it difficult to determine the specific checks or validations that should be applied to the input data during the runtime of the ML model, because already for the training data, there were no clearly defined criteria or checks. Related quotes: Q3, Q10	
R7	**Unclear design domain, and the dependency of data requirements on the design domain, are related to missing guidelines for data selection. (3, 5.1)**

Table 6 (continued)

ID	Relation between challenges
Rationale: The design domain can refer to the specific context, problem, or application in which the ML model is being developed. Data requirements include considerations such as data quality, physical properties of the data such as image resolution, diversity, representativeness, or domain-specific constraints. If the design domain is unclear it will be challenging to determine context-specific data requirements. Lack of clarity about the problem statement, application scope, or intended functionality hinders the ability to identify the relevant data sources, data quality standards, or other considerations necessary for effective data selection. Related quotes: Q1, Q2, Q4, Q5	
R8	**The focus on processes instead of technical solutions in the context of functional safety is related to missing guidelines for data selections. (3, 6.1)**
Rationale: If there is a too strong focus on processes, such as defining workflows, documenting methodologies, or establishing quality assurance procedures, it can divert attention away from addressing the technical aspects of data selection. Technical solutions can refer to the discovery of specific algorithms, architectures, or techniques used to implement and train the ML model. When the focus primarily revolves around processes rather than technical solutions, the development of explicit data selection strategies may be neglected because companies can assume the technical solutions for data selection are covered within the broader processes. Related quotes: Q2, Q7	
R9	**Achieving a safety case only through monitoring solution is related to a reduction of true positive rate. (4.3, 6.3)**
Rationale: The safety case for systems with ML components relies often on runtime monitors as primary means of identifying safety issues. Runtime monitors however can limit the operational boundary of the system which can lead to a reduction of the true positive rate of the ML component. Achieving a comprehensive safety case for systems with ML components requires not only a reliance on monitoring solutions, but must also take the ML component and its development process, including the data selection, into account. Related quotes: Q14, Q15	

of confidence and **3.4: Unclear measure of variety**. We mapped challenges to architecture concerns based on the earlier identified mapping of the identified challenges of both studies shown in Table 5 and based on the mapping of the challenges to architecture views presented in [32]. Table 7 presents the results of this mapping. Challenge theme **2.2** is mapped to architecture views concerning the AI model. Challenge theme **3.4** is mapped to architecture views concerning the data strategy. In summary relation **R2b** suggests that there should be correspondences between architecture views describing the AI model (such as the AI model concept / layer structure) and architecture views describing the data strategy (such as views describing the data pipeline). The cross-group relation **R5: Lack of explainability about ML decisions is related to the inability to understand and trust the measure of confidence** suggests architecture view correspondences between views in the cluster-of-concern "quality aspect", specifically views concerning the explainability of the system, and the AI model architecture. The relation **R6: Missing guidelines for data selection is related to the inability to define runtime checks** suggests correspondence rules between views concerning the data strategy (data pipeline) and views on software components for the runtime monitoring. Furthermore, the relation **R7: Unclear design domain, and the dependency of data requirements on the design domain, are related to missing guidelines for data selection** requires correspondences between architecture views describing the operational context (such as an ODD description for automotive systems), and views describing the data strategy. The relation **R8: The focus on processes instead of technical solutions in the context to functional safety is related to missing guidelines for data selections** suggests correspondences between architecture views describing the quality aspects of safety (e.g., in the form of hazard analyses tables and fault tree diagrams) and views describing the data strategy. And, **R9: Achieving a safety case only through monitoring solution is related to a reduction of true positive rate** relates to architecture views describing the safety quality aspect of the system and architecture views describing the AI model, i.e., correspondences should exist not only between the safety views and the monitoring solutions, but also towards the AI model itself in order to diversify the safety risk. We did not map challenge 3.2: Grown data selection habits to an architecture view because we assume that overcoming existing habits in a company requires rather a properly defined process then an architecture view on the system.

3.4 An example of relations between data concerns, runtime monitoring, and architecture decisions

Figure 2 earlier provided an example of how an architecture framework can be used to link multiple architectural concerns in ML systems development. The figure shows an extract from the architecture framework proposed in [32] for five clusters of concerns: Logical architecture, operational context, AI model, data strategy, and functional safety. For each cluster of concern, four architecture views are defined at different levels of abstraction. At the most abstract level, the *analytical level*, system development still takes place mostly in a problem space. The developers gather knowledge and analyse the problem that the system is supposed to solve. Already here, the problem to be solved can be viewed from different “perspectives",

Table 7 Mapping of the identified challenges to clusters of concern and architecture views given in [32] based on the mapping of challenges between the studies depicted in Table 5

	Context and ODD	Data strategy	AI model	Hardware	Quality aspects
Overview	Provides assumptions and detailed description about the operational context in which the desired behaviour will be executed	Provides guidance on capturing and storing, as well as selecting data for training. This includes the definition of data quality	Provides descriptions of the AI/ML model that uses the data captured and stored according to the data strategy to make predictions. It includes detailed information about the model's architecture and configuration	Entails architecture information on the required hardware components at different levels of abstraction. This includes hardware for sensors, processing units, and actuators as long as they are in relation to the system's scope[1]	Contains architecture views governing quality aspects of the system, such as safety which can stem from standards such as ISO 26262 [41] or explainability / fairness
Roles and Stakeholders	Business/use case owners, user, developers, data scientists, testers, regulators	Data scientists, testers, regulators, (test) drivers, lab personnel	Developers, testers, regulators	Business owners, developers, testers, lab personnel	Developers, testers, regulators, users
Model kinds	ODD templates, context diagram, class diagram, taxonomies, causal diagrams	Quality models, Planguage, causal diagrams	Flowcharts and block-diagrams, graph models, mathematical equations, pseudo-code, UML diagrams, Configuration specifications	Blockdiagrams, circuit diagrams, specification sheets, board-level schematics	Risk assessment models, fairness evaluations, failure mode and effect models, fault-tree diagram, formal verification
Operations on views	Identify and manage context uncertainty, provide foundation to define data requirements	Ensure that appropriate data is covered in training, data optimisation, data reduction, operationalisation of quality attributes, statistical modelling	Ensures that the AI/ML model is suitable to the planned operation and use case, aligns input and output layers with expected data, ensures compatibility with hardware	Aligns hardware needs with use case expectations, ensures sufficient performance for AI model inference	Identifies weak points of systems, ensures quality aspects such as redundancy for reliability of system or fairness and explainability
Corres-pondence rules	Data ingestion and data selection	Context and ODD, AI model, hardware, quality views (safety, explainability)	Context and ODD, Data ingestion and selection, System and component hardware architecture	AI model, operational context and ODD, data ingestion, system hardware and component architecture	System hardware architecture, component architecture, AI model concept, data selection, learning setup, context and ODD
Mapped Challenges[2]	2.1, 5.1, 5.2	3.1, 3.3, 3.4	1.1, 1.3, 2.1, 2.2, 6.2	4.1, 4.2	1.2, 4.3, 6.1, 6.2, 6.3

[1]For example, the electric circuitry that triggers a brake request for an automatic emergency breaking system is in scope of the hardware cluster of concern for such a system. However, the mechanical components of the brake system are out of scope

[2] A description of the mapping to the challenges themes is provided in Sect. 3.3

i.e., in this example from a logical architecture, an operational context, an AI model, a data and a security point of view. Defining own architecture views for the AI model and creating correspondences to other system views can answer *challenge theme 1.1: No access to inner states of ML models*. Table 4 lists all challenges themes. Early on, already in the problem space, correspondences can be created between different points of view, that is between different architecture views. For example, a *hazard analysis* is only valid under certain *context assumptions*. Furthermore, a *hazard analysis* might highlight certain critical operational scenarios that need to be considered in the *data ingestion* for the AI model's training data, thus creating more reasonable completeness criteria for data collection as mentioned in *challenge theme 3.3: Unclear completeness criteria*. At the next level of abstraction, *conceptual level*, system development moves closer to the solution space. Here, for example, we can establish correspondences between the *context definition or operational design domain (ODD)* and the *data selection*, which can address the challenges in *challenge group 5: Unclear design domain*. Measures of variety for the data, and measures of confidence for the AI model can, on this level, clearly be linked to the defined operational context. This can solve challenges in *Challenge group 3: Missing guidelines for data selection*. Other correspondences can be made between the *functional safety concept*, which would define necessary redundancies in the system, the *logical components* and the *AI model concept*, e.g. defining a maximum inference time for the AI model, which relates to *challenge themes 4.2: meeting timing requirements*. At the *design level*, the final design decisions of the system are made in the solution space. For example, based on the chosen hardware architecture, computational resources can be allocated to the AI model, creating a correspondence between the *computational resource allocation* and *AI model configuration* views, since the AI model must be configured to run most efficiently on the chosen computational resources. The least abstract level, i.e. the most concrete level of abstraction, is the *runtime level*. This defines the runtime behaviour of the system for each cluster of interest, including the design of the necessary runtime monitors and the adaptive behaviour of the system. For example, safety monitors can now be defined for safety-critical aspects that could not be adequately covered by the technical safety concept. Links can be made to the *adaptive behaviour control*, which can, for example, deactivate the AI model in the event of malfunction or unexpected results. The safety case would mostly rely on the previous design decisions, i.e. the technical safety concept that ensures appropriate training data, AI model setup and computing resources. The safety case would therefore not rely solely on safety monitors to answer *challenge theme 6.3: Safety case only through monitoring solution*.

4 Discussion

This interview study investigated challenges experienced by practitioners when specifying data needed for training and validating of critical ML-enabled system, as well as specifying necessary runtime monitoring solutions for such systems. Some of the underlying problems and potential solutions have also been investigated in other publications.

4.1 Related literature

The problem of finding the "right" data For acquiring data, data scientists have to rely on data mining with little to no quality checking and potential biases [6]. Biased datasets are a common cause for erroneous or unexpected behaviour of ML models in critical environments, such as in medical diagnostic [10], in the juridical system [28, 58], or in safety-critical applications [23, 78].

There are attempts to create "unbiased" datasets. One approach is to curate manually the dataset, such as in the FairFace dataset [49], the CASIA-SURF CeFaA dataset [53], or Fairbatch [70]. An alternative road is to use data augmentation techniques to "rebalance" the dataset [45, 77]. However, it was discovered that it is not sufficient for avoiding bias to use an assumed balanced datasets during training [29, 84, 85] because it is often unclear which features in the data need to be balanced. Approaches for curating or manipulating the dataset require information on the target domain, i.e., one needs to set requirements on the dataset depending on the desired operational context [8, 24, 33]. But deriving a data specification for ML is not common practise [37, 54, 72].

The problem of finding the "right" runtime monitor Through clever test strategies, some uncertainty can be eliminated in regards to the behaviour of the model [15]. However, ML components are often part of systems of systems and their behaviour is hard to predict and analyse at design time [81]. DevOps principles from software engineering give promising ideas on how to tackle remaining uncertainty at runtime [55, 80]. An overview of MLOps can be for example found in [52]. As part of the operation of the model, runtime models that "augment information available at design-time with information monitored at runtime" help in detecting deviations from the expected behaviour [26]. These runtime models for ML can take the form of model assertions, i.e., checking of explicitly defined attributes of the model at runtime [48]. However, the authors state that "bias in training sets are out of scope for model assertion". Another model based approach can be the creation

of neuron activation patterns for runtime monitoring [18]. Other approaches treat the ML model as “black-box", and only check for anomalies and drifts in the input data [68] the output [73], or both [27]. However, similar to the aforementioned challenges when specifying data for ML, runtime monitoring needs an understanding on how to “define, refine, and measure quality of ML solutions" [34], i.e., in relation to non-functional requirements one needs to understand which quality aspects are relevant, and how to measure them [30]. Most commonly applied safety standards emphasise processes and traceability to mitigate systematic mistakes during the development of critical systems. Therefore, if the training data and runtime monitoring cannot be specified, a traceability between safety goals and the deployed system cannot be established [13].

For many researchers and practitioners, runtime verification and monitoring is a promising road to assuring safety and robustness for ML in critical software [2, 15]. However, runtime monitoring also creates a processing and memory overhead that needs to be considered especially in resource-limited environments such as embedded devices [67]. The related work has been mapped to the challenges identified in the interview study in Table 4.

4.2 Recommendation for practitioners

The identified relations of the challenges described by the participants, as well as the solutions suggested in related work, allow us to formulate recommendations and implications towards RE practises. The implications towards RE practises are stated after each challenge theme in Sect. 3. Because these recommendations try to solve causes described by the participants of the interview study, we think they are a useful first step towards solving the challenges related to specifying requirements for training data and runtime monitoring.

Recommendation 1: Avoid restrictive IP protection. IP protection is a cause of the inability to access the inner states and workings of the ML models (black-box model). This can lead to nontransparent measures of confidence, and an inability to formulate failure models. To the best of our knowledge, no studies have yet been performed on the consequences of IP protection of ML models on the ability to monitor and reason (e.g., in a safety case) for the correctness of ML model decisions. *Three validation survey participants strongly agreed, one agreed, one answered neutrally, and one disagreed with Recommendation 1.*

Recommendation 2: Relate measures of confidence to actual performance metrics. For runtime monitoring, the measure of confidence is often used to evaluate the reliability of the ML model results. However, without first understanding and relating this measure to clearly defined performance metrics of the ML model, the measure of confidence provides little insight for runtime monitoring. In general, the definition of appropriate metrics and boundary conditions should become an integral part of RE for ML as it affects both the ability to define data requirements and the requirements for runtime monitoring. *Three validation survey participants strongly agreed and three agreed with Recommendation 2.*

Recommendation 3: Overcome grown data selection habits. Grown data selection habits were cited as a reason for the lack of clear completeness criteria and a disconnect from requirements. Based on our findings, we argue that more systematic data selection processes need to be established in organisations and companies. This would allow the data selection process to be better linked to RE, and it would create traceability between system requirements, completeness criteria and data requirements. In addition, it could also reduce the amount of data required for training and thus reduce development costs. *Three validation survey participants strongly agreed, two agreed, and one answered neutrally with Recommendation 3.*

Recommendation 4: Balance hardware limitation in embedded systems. Runtime monitors impose processing and memory overhead that can compromise timing requirements and reduce the performance of the ML model. Today, the safety of systems with ML is mostly provided by some forms of runtime monitoring. By instead decomposing the safety requirements on both the monitoring and the ML model, the monitors could become more resource efficient, faster, and less constraining on the decisions of the ML model. This however means also that safety requirements on the ML models can trigger requirements on the training data. *Two validation survey participants strongly agreed, one agreed, and three answered neutrally with Recommendation 4.*

Recommendation 5: Relate the challenges of specifying requirements for training data and runtime monitoring to architecture decisions of the AI-enabled systems. As emphasised in Recommendation 4, solving one challenge (overhead of runtime monitoring) requires design changes in other architecture areas of the system (ML model design, selection of training data / data pipeline). We found relationships between the challenges identified in this study and the challenges of designing AI-based systems identified in a previous study, which led to the definition of an architecture framework for AI systems described in [32]. Since the system architecture and the requirements / system specifications evolve together in what is known as the twin-peak-model [19], we propose to overcome the specification challenges by decomposing the design task early on into different architecture views that describe the relevant concerns of the system and by establishing correspondences between the architecture views. Several recent studies support the recommendation to relate challenges in the development of ML-enabled systems to architecture decisions: In a combined literature

survey and interview study, Nazir et al. provide an overview on the role of architectural design decisions for ML-enabled systems [61]. There, data is been highlighted as one major design challenge. For example, pre-processing and preparation of data requires architecture design decision regarding the data pipeline and data management for ML-enabled system development [17]. The authors also state that one interviewee (I5) highlighted that "Data accuracy and completion of data are crucial when training the models of ML systems" [61], which corresponds to the findings in this study. Bhat et al. highlight the necessity of relating RE challenges to architecture design decisions for large software engineering project such as ML system development [11]. *Three validation survey participants strongly agreed, two agreed, and one answered neutrally with Recommendation 5.*

4.3 Threats to validity

Threats related to internal validity A lack of rigour (i.e., degree of control) in the study design can cause confounding which can manifest bias in the results [74]. The following mechanisms in this study tried to reduce confounding: The interview guide was peer-reviewed by an independent researcher, and a test session of the interview was conducted. To reduce personal bias, at least two authors were present during all interviews, and the authors took turn in leading the interviews. To confirm the initial findings from the interview study and reduce the risk of researchers' bias, a workshop was organised which was also visited by participants who were not part of the interview study. This workshop, plus additional meetings with practitioners were used to conduct member checking of the preliminary results. For answering RQ3, the researchers used their expert knowledge to map the identified challenges to architecture views. However, the researchers referred to an already published mapping of similar challenges to architecture views and used that mapping as guidance for all but two challenge groups. For the remaining two challenge groups, the authors provide arguments to why (*ID 4: overhead for monitoring solution*) or why not (*ID 6: unsuitable safety standards*) such a mapping to architecture views is possible.

Threats related to construct validity Another potential bias can arise from the sampling of participants. Although we applied purposeful sampling, we still had to rely on the contact persons of the companies to provide us with suitable interview candidates. We could not directly see a list of employees and therefore not choose the candidates ourselves. We tried to support our contact persons with a list of roles that should be part of the interviews and discussed the final selection of participants with our industry partners before inviting them to the interviews.

Threats related to external validity Regarding generalisability of the findings, the limited number of companies involved in the study can pose a threat to external validity. However, two of the companies are world-leading companies in their fields, which, in our opinion, gives them a deep understanding and experience of the discussed problems. Furthermore, we included companies from a variety of different fields to establish better generalisability. Furthermore, our data includes only results valid for the development of safety-critical ML models. We assume that the findings are applicable also to other forms of criticality, such as privacy-critical, but we cannot conclude on that generalisability based on the available data.

4.4 Conclusion

This article reports on an interview-based study that identified challenges associated with specifying requirements for training data and runtime monitoring for safety-critical ML models. Through interviews conducted at five companies we identified 17 challenge themes in six groups. We also performed a manual semantic analysis to identify the underlying relationships between the challenges. We found that several underlying challenges affect both the ability to specify training data and runtime monitoring. For example, we concluded that restrictive IP protection can lead to an inability to access and understand the inner states of an ML model. Without insight into the states of the ML model, the measure of confidence cannot be related to actual performance metrics. Without clear performance metrics, it is difficult to define the necessary degree of variety in the training data. In addition, grown data selection habits prevent proper RE for training data. Finally, safety requirements should be distributed across both the ML model, which can impose requirements on the training data, and on runtime monitors, which can reduce the overhead of the monitoring solution. Based on the identified challenge groups, we formulated a set of seven implications towards RE. The study also compared the challenges found with those identified in a previous study on the design of distributed ML-based systems and the creation of an architecture frameworks for such systems. We found several parallel challenges in both studies, suggesting that an architecture framework and early decomposition of the design task into architecture concerns could potentially mitigate the challenges of specifying data and monitoring requirements for critical ML-based systems. Finally, we proposed five recommendations to practitioners based on the identified relations between the challenges described by the participants of the study and the mapping of previously identified challenges in specifying requirements for training data and monitoring to architectural views. Both the implications for RE, as well as the recommendations for practitioners were validated by an additional survey of practitioners. Both the implications as well as the recommendations will serve as starting point for further engineering research.

Acknowledgements This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 957197 and from the Sweden's innovation agency Vinnova under grant agreement No 2023-00771.

Data availability A replication package containing data from the workshops and interviews related to RQ1 and RQ2 is available at https://doi.org/10.7910/DVN/WJ8TKY. A replication package containing data from the workshops related to RQ3 is available at https://doi.org/10.7910/DVN/VXFFFU.

References

1. Abid A, Farooqi M, Zou J (2021) Persistent anti-muslim bias in large language models. In: Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society. pp 298–306
2. Ashmore R, Calinescu R, Paterson C (2021) Assuring the machine learning lifecycle: desiderata, methods, and challenges. ACM Comput Surv 54(5):1–39
3. Aslansefat K, Sorokos I, Whiting D, Tavakoli Kolagari R, Papadopoulos Y (2020) Safeml: safety monitoring of machine learning classifiers through statistical difference measures. In: International symposium on model-based safety and assessment. pp 197–211. Springer
4. Ballingall S, Sarvi M, Sweatman P (2023) Standards relevant to automated driving system safety: a systematic assessment. Trans Eng 13:100202
5. Banko M, Brill E (2001) Scaling to very very large corpora for natural language disambiguation. In: Proceedings of the 39th annual meeting of the association for computational linguistics. pp 26–33
6. Barocas S, Selbst AD (2016) Big data's disparate impact. Calif. L. Rev. 104:671
7. Bayram F, Ahmed BS, Kassler A (2022) From concept drift to model degradation: an overview on performance-aware drift detectors. Knowl Based Syst 245:108632
8. Bencomo N, Guo JL, Harrison R, Heyn HM, Menzies T (2021) The secret to better AI and better software (is requirements engineering). IEEE Softw 39(1):105–110
9. Bencomo N, Whittle J, Sawyer P, Finkelstein A, Letier E (2010) Requirements reflection: requirements as runtime entities. In: Proceedings of the 32nd ACM/IEEE international conference on software engineering vol. 2, pp 199–202
10. Bernhardt M, Jones C, Glocker B (2022) Potential sources of dataset bias complicate investigation of underdiagnosis by machine learning algorithms. Nat Med 28(6):1157–1158
11. Bhat M, Shumaiev K, Koch K, Hohenstein U, Biesdorf A, Matthes F (2018) An expert recommendation system for design decision making: who should be involved in making a design decision? In: 2018 IEEE international conference on software architecture (ICSA). pp 85–8509. IEEE
12. Blodgett SL, Barocas S, Daum'e H, Wallach HM (2020) Language (technology) is power: A critical survey of "bias" in nlp. In: ACL
13. Borg M, Englund C, Wnuk K, Duran B, Levandowski C, Gao S, Tan Y, Kaijser H, Lönn H, Törnqvist J (2018) Safely entering the deep: a review of verification and validation for machine learning and a challenge elicitation in the automotive industry. J Automot Softw Eng 1(1):1–19
14. Brand T, Giese H (2018) Towards software architecture runtime models for continuous adaptive monitoring. In: MoDELS (Workshops). pp 72–77
15. Breck E, Cai S, Nielsen E, Salib M, Sculley D (2017) The ml test score: a rubric for ml production readiness and technical debt reduction. In: 2017 IEEE international conference on big data. pp 1123–1132. IEEE
16. Brown DW, Carson CD, Montgomery WA, Zislis PM (1988) Software specification and prototyping technologies. AT &T Tech J 67(4):33–45
17. Castellanos C, Pérez B, Correal D, Varela CA (2020) A model-driven architectural design method for big data analytics applications. In: 2020 IEEE international conference on software architecture companion (ICSA-C). pp 89–94. IEEE
18. Cheng CH, Nührenberg G, Yasuoka H (2019) Runtime monitoring neuron activation patterns. In: 2019 Design, automation & test in Europe conference & exhibition. pp 300–303. IEEE
19. Cleland-Huang J, Hanmer RS, Supakkul S, Mirakhorli M (2013) The twin peaks of requirements and architecture. IEEE Softw 30(2):24–29
20. Clements P, Bachmann F, Bass L, Garlan D, Ivers J, Little R, Nord R, Stafford J (2011) Documenting software architectures: views and beyond. SEI Series in Software Engineering, second edn
21. Creswell JW, Creswell JD (2017) Research design: qualitative, quantitative, and mixed methods approaches. Sage publications, Thousand Oaks
22. Creswell John W, Poth CN (2017) Qualitative inquiry and research design: choosing among five approaches, 4th edn. Sage Publishing, Thousand Oaks
23. Fabbrizzi S, Papadopoulos S, Ntoutsi E, Kompatsiaris I (2021) A survey on bias in visual datasets. arXiv preprint arXiv:2107.07919
24. Fauri D, Dos Santos DR, Costante E, den Hartog J, Etalle S, Tonetta S (2017) From system specification to anomaly detection (and back). In: Proceedings of the 2017 workshop on cyber-physical systems security and privacy. pp 13–24
25. Gamble MT (2016) Can metamodels link development to design intent? In: Proceedings of the 1st international workshop on bringing architectural design thinking into developers' daily activities. pp 14–17
26. Giese H, Bencomo N, Pasquale L, Ramirez AJ, Inverardi P, Wätzoldt S, Clarke S (2014) Living with uncertainty in the age of runtime models. In: Models@ run. time, pp 47–100. Springer
27. Ginart T, Zhang MJ, Zou J (2022) Mldemon: Deployment monitoring for machine learning systems. In: International conference on artificial intelligence and statistics. pp 3962–3997. PMLR
28. Goodman B, Flaxman S (2017) European union regulations on algorithmic decision-making and a "right to explanation''. AI magazine 38(3):50–57
29. Gwilliam M, Hegde S, Tinubu L, Hanson A (2021) Rethinking common assumptions to mitigate racial bias in face recognition datasets. In: Proceedings of the IEEE CVF. pp 4123–4132
30. Habibullah KM, Horkoff J (2021) Non-functional requirements for machine learning: understanding current use and challenges in industry. In: 2021 IEEE 29th RE Conference. pp 13–23. IEEE
31. Heyn HM, Knauss E, Malleswaran I, Dinakaran S (2023) An investigation of challenges encountered when specifying training data and runtime monitors for safety critical ml applications. In: International working conference on requirements engineering: foundation for software quality. pp 206–222. Springer
32. Heyn HM, Knauss E, Pelliccione P (2023) A compositional approach to creating architecture frameworks with an application to distributed AI systems. J Syst Softw 198:111604
33. Heyn HM, Subbiah P, Linder J, Knauss E, Eriksson O (2022) Setting AI in context: a case study on defining the context and operational design domain for automated driving. In: International working conference on requirements engineering: foundation for software quality. pp 199–215. Springer
34. Horkoff J (2019) Non-functional requirements for machine learning: challenges and new directions. In: 2019 IEEE 27th RE conference. pp 386–391. IEEE

35. Humbatova N, Jahangirova G, Bavota G, Riccio V, Stocco A, Tonella P (2020) Taxonomy of real faults in deep learning systems. In: 2020 IEEE/ACM 42nd international conference on software engineering. pp 1110–1121
36. IEEE SA Board of Governors/Corporate Advisory Group (BoG/CAG) (2019) IEEE Std 2413: Architectural Framework for the Internet of Things (IOT). IEEE Computer Society
37. Ishikawa F, Yoshioka N (2019) How do engineers perceive difficulties in engineering of machine-learning systems?-questionnaire survey. In: 2019 IEEE/ACM Joint 7th international workshop on conducting empirical studies in industry. pp 2–9. IEEE
38. Islam MJ, Nguyen G, Pan R, Rajan H (2019) A comprehensive study on deep learning bug characteristics. In: 2019 ACM 27th European software engineering conference. pp 510–520
39. ISO (2008) ISO/IEC 25012:2008: Software engineering–Software product quality requirements and evaluaton (SQuaRE). International organization for standardization, Geneva, www.iso.org
40. ISO (2012) ISO/IEC/IEEE 42010:2012: Systems and software engineering–Architecture description. Swedish Standards Institute, Stockholm, www.sis.se
41. ISO (2018) ISO 26262:2018: Road vehicles–Functional safety. International Organization for Standardization, Geneva, www.iso.org
42. ISO (2023) ISO/CD PAS 8800: road vehicles safety and artificial intelligence, under development. International Organization for Standardization, Geneva, www.iso.org
43. ISO (2023) ISO/CD TS 5083: safety for automated driving systems–Design, verification and validation, under development. International organization for standardization, Geneva, www.iso.org
44. ISO (2023) ISO/IEC DTR 5469: functional safety and AI systems, under development. International organization for standardization, Geneva, www.iso.org
45. Jaipuria N, Zhang X, Bhasin R, Arafa M, Chakravarty P, Shrivastava S, Manglani S, Murali VN (2020) Deflating dataset bias using synthetic data augmentation. In: Proceedings of the IEEE CVF. pp 772–773
46. Kahan E, Genero M, Oliveros A (2019) Challenges in requirement engineering: could design thinking help? In: Quality of information and communications technology: 12th international conference, QUATIC 2019, Ciudad Real, Spain, September 11–13, 2019, Proceedings 12. pp 79–86. Springer
47. Kaiser M, Griessl R, Kucza N, Haumann C, Tigges L, Mika K, Hagemeyer J, Porrmann F, Rückert U, vor dem Berge M, et al (2022) Vedliot: very efficient deep learning in IOT. In: 2022 Design, Automation & Test in Europe conference & exhibition (DATE). pp 963–968. IEEE
48. Kang D, Raghavan D, Bailis P, Zaharia M (2020) Model assertions for monitoring and improving ml models. Proc Mach Learn Syst 2:481–496
49. Karkkainen K, Joo J (2021) Fairface: face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. In: Proceedings of the IEEE CVF. pp 1548–1558
50. King N, Horrocks C, Brooks J (2018) Interviews in qualitative research. Sage publications, Thousand Oaks
51. Knight JC (2002) Safety critical systems: challenges and directions. In: 24th international conference on software engineering. pp 547–550
52. Kreuzberger D, Kühl N, Hirschl S (2022) Machine learning operations (mlops): overview, definition, and architecture. arXiv preprint arXiv:2205.02302
53. Liu A, Tan Z, Wan J, Escalera S, Guo G, Li SZ (2021) Casia-surf cefa: a benchmark for multi-modal cross-ethnicity face anti-spoofing. In: Proceedings of the IEEE CVF. pp 1179–1187
54. Liu H, Eksmo S, Risberg J, Hebig R (2020) Emerging and changing tasks in the development process for machine learning systems. In: Proceedings of the international conference on software and system processes. pp 125–134
55. Lwakatare LE, Crnkovic I, Bosch J (2020) Devops for AI–challenges in development of AI-enabled applications. In: 2020 International conference on software, telecommunications and computer networks. pp 1–6. IEEE
56. Marques J, Yelisetty S (2019) An analysis of software requirements specification characteristics in regulated environments. J Softw Eng Appl (IJSEA) 10(6):1–15
57. Mehrabi N, Morstatter F, Saxena N, Lerman K, Galstyan A (2021) A survey on bias and fairness in machine learning. ACM Comput Surv 54(6):1–35
58. Miron M, Tolan S, Gómez E, Castillo C (2021) Evaluating causes of algorithmic bias in juvenile criminal recidivism. Artif Intell Law 29(2):111–147
59. Muccini H, Vaidhyanathan K (2021) Software architecture for ml-based systems: what exists and what lies ahead. In: Proceedings of the 43rd international conference on software engineering, http://arxiv.org/abs/2103.07950
60. Namey E, Guest G, Thairu L, Johnson L (2008) Data reduction techniques for large qualitative data sets. Handbook Team-Based Qualit Res 2(1):137–161
61. Nazir R, Bucaioni A, Pelliccione P (2023) Architecting ml-enabled systems: challenges, best practices, and design decisions. J Syst Softw 207:111860
62. Nilsson J, Bergenhem C, Jacobson J, Johansson R, Vinter J (2013) Functional safety for cooperative systems. Tech. rep, SAE Technical Paper
63. Nord RL, Ozkaya I, Kruchten P (2014) Agile in distress: architecture to the rescue. In: Agile methods. Large-scale development, refactoring, testing, and estimation: XP 2014 international workshops, Rome, Italy, May 26-30, 2014, Revised Selected Papers 15. pp 43–57. Springer
64. Nuseibeh B (2001) Weaving together requirements and architectures. Computer 34(3):115–119
65. Pelliccione P, Knauss E, Heldal R, Ågren SM, Mallozzi P, Alminger A, Borgentun D (2017) Automotive architecture framework: the experience of volvo cars. J Syst Architect 77:83–100
66. Quinonero-Candela J, Sugiyama M, Schwaighofer A, Lawrence ND (2008) Dataset shift in machine learning. Mit Press, Cambridge
67. Rabiser R, Schmid K, Eichelberger H, Vierhauser M, Guinea S, Grünbacher P (2019) A domain analysis of resource and requirements monitoring: towards a comprehensive model of the software monitoring domain. Inf Softw Technol 111:86–109
68. Rahman QM, Sunderhauf N, Dayoub F (2021) Per-frame map prediction for continuous performance monitoring of object detection during deployment. In: Proceedings of the IEEE CVF. pp. 152–160
69. Richards L (2014) Handling qualitative data: a practical guide. Handling qualitative data pp 1–264
70. Roh Y, Lee K, Whang S, Suh C (2021) Sample selection for fair and robust training. Adv Neural Inf Process Syst 34:815–827
71. Saldaña J (2013) The coding manual for qualitative researchers. Sage Publishing, Thousand Oaks
72. Sambasivan N, Kapania S, Highfill H, Akrong D, Paritosh P, Aroyo LM (2021) Everyone wants to do the model work, not the data work: Data cascades in high-stakes AI. In: 2021 conference on human factors in computing systems. pp 1–15
73. Shao Z, Yang J, Ren S (2020) Increasing trustworthiness of deep neural networks via accuracy monitoring. arXiv preprint arXiv:2007.01472
74. Slack MK, Draugalis JR Jr (2001) Establishing the internal and external validity of experimental studies. Am J Health Syst Pharm 58(22):2173–2181

75. Steghöfer JP, Knauss E, Horkoff J, Wohlrab R (2019) Challenges of scaled agile for safety-critical systems. In: Product-focused software process improvement: 20th international conference, PROFES 2019, Barcelona, Spain, November 27–29, 2019, Proceedings 20. pp 350–366. Springer
76. Tripathi S, De S (2019) Data-driven optimizations in IOT: a new frontier of challenges and opportunities. CSI Trans ICT 7:35–43
77. Uchôa V, Aires K, Veras R, Paiva A, Britto L (2020) Data augmentation for face recognition with cnn transfer learning. In: 2020 international conference on systems, signals and image processing. pp 143–148. IEEE
78. Uricár M, Hurych D, Krizek P, Yogamani S (2019) Challenges in designing datasets and validation for autonomous driving. arXiv preprint arXiv:1901.09270
79. Vetterli C, Brenner W, Uebernickel F, Petrie C (2013) From palaces to yurts: why requirements engineering needs design thinking. IEEE Internet Comput 17(2):91–94
80. Vierhauser M, Rabiser R, Grünbacher P (2016) Requirements monitoring frameworks: a systematic review. Inf Softw Technol 80:89–109
81. Vierhauser M, Rabiser R, Grünbacher P, Danner C, Wallner S, Zeisel H (2014) A flexible framework for runtime monitoring of system-of-systems architectures. In: 2014 IEEE conference on software architecture. pp 57–66. IEEE
82. Vinnova (2023) Famer–Facilitating multi-party engineering of requirements, https://www.vinnova.se/en/p/-party-engineering-of-requirements/, Accessed: 2023-11-28
83. Vogelsang A, Borg M (2019) Requirements engineering for machine learning: perspectives from data scientists. In: 2019 IEEE 27th international requirements engineering conference workshops. pp 245–251. IEEE
84. Wang A, Liu A, Zhang R, Kleiman A, Kim L, Zhao D, Shirai I, Narayanan A, Russakovsky O (2022) Revise: a tool for measuring and mitigating bias in visual datasets. Int J Comput Vis 130(7):1790–1810
85. Wang T, Zhao J, Yatskar M, Chang KW, Ordonez V (2019) Balanced datasets are not enough: estimating and mitigating gender bias in deep image representations. In: Proceedings of the IEEE/CVF international conference on computer vision october
86. Wardat M, Le W, Rajan H (2021) Deeplocalize: fault localization for deep neural networks. In: 2021 IEEE/ACM 43rd international conference on software engineering. pp 251–262. IEEE
87. Zhang X, Xie X, Ma L, Du X, Hu Q, Liu Y, Zhao J, Sun M (2020) Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. 2020 IEEE/ACM 42nd international conference on software engineering pp 739–751