Adapting Software Architectures to Machine Learning Challenges

Alex Serban
iCIS, Radboud University
The Netherlands
a.serban@cs.ru.nl

Joost Visser
LIACS, Leiden University
The Netherlands
j.m.w.visser@liacs.leidenuniv.nl

Abstract—Unique developmental and operational characteristics of machine learning (ML) components as well as their inherent uncertainty demand robust engineering principles are used to ensure their quality. We aim to determine how software systems can be (re-)architected to enable robust integration of ML components. Towards this goal, we conducted a mixed-methods empirical study consisting of (i) a systematic literature review to identify the challenges and their solutions in software architecture for ML, (ii) semi-structured interviews with practitioners to qualitatively complement the initial findings and (iii) a survey to quantitatively validate the challenges and their solutions. We compiled and validated twenty challenges and solutions for (re-)architecting systems with ML components. Our results indicate, for example, that traditional software architecture challenges (e.g., component coupling) also play an important role when using ML components; along with new ML specific challenges (e.g., the need for continuous retraining). Moreover, the results indicate that ML heightened decision drivers, such as privacy, play a marginal role compared to traditional decision drivers, such as scalability. Using the survey we were able to establish a link between architectural solutions and software quality attributes, which enabled us to provide twenty architectural tactics used to satisfy individual quality requirements of systems with ML components. Altogether, the results of the study can be interpreted as an empirical framework that supports the process of (re-)architecting software systems with ML components.

Index Terms—software engineering, software architecture, machine learning

I. INTRODUCTION

Software architecture (SA) plays an important role in data intensive systems, such as big data and analytics platforms. However, until recently, the focus has been on the architectural decisions related to handling and storing large amounts of data and on decisions that mitigate performance demands of analytics platforms [11], [2].

The interest to develop software with machine learning (ML) components shifts the focus to decisions regarding the operational requirements of serving, monitoring, retraining and redeploying models [3]. These decisions align with proposals to emphasise the operational aspect of SA [4]. Moreover, the inherent uncertainty of ML components demands a stronger emphasis on the uncertainty aspect of SA; where the focus is on assessing the impact of uncertainty and on the decisions made for its mitigation [5], [6].

Although a significant body of literature studied the relevance of SA for big data and analytics platforms [2], [7], there is little empirical research on the role of SA in systems with ML components [8], [9]. Our aim is to determine how software systems can be (re-)architected to enable robust integration of ML components.

Towards this goal, we conducted a mixed-methods empirical study consisting of three stages. First, we performed a systematic literature review (SLR) to identify the challenges faced in (re-)architecting systems with ML components and the solutions proposed to meet them. We analysed 42 relevant articles, from which we compiled an initial set of 18 challenges and solutions. Second, we performed 10 semi-structured interviews with practitioners from 10 organisations – ranging from start-ups to large companies. The interviews were used to complement the initial set of challenges (and solutions) and to assess the impact of each challenge on SA. In total, 2 new challenges were discovered in the interviews, as well as multiple new solutions. Third, we ran a survey with 48 software architects in order to quantitatively validate and complement the challenges and solutions. The survey also established a link between challenges, solutions and software quality attributes, allowing the solutions to be restated as architectural tactics.

Overall, our main contributions are as follows. First, we summarised academic and grey literature on the topic of SA for ML in a catalogue of SA challenges and related solutions. This information can guide practitioners to (re-)architect software with ML components, or as a gateway to relevant literature. Second, we validated and complemented the initial findings by engaging with practitioners. We found out that, although the initial challenges had solutions in the literature, the solution were considered incomplete by practitioners. Third, we linked the architectural solutions to software quality attributes from the ISO/IEC 25010 standard [10], which allowed to restate them as architectural tactics. Last, we assessed the impact of each challenge on SA, which allowed us to contrast traditional SA concerns with emergent ML concerns.

The paper is organised as follows: first we discuss related work (Section II), followed by the study design (Section III), results (Section IV), discussions (Section V) and conclusions (Section VI). All supplementary materials are public [11].

II. BACKGROUND AND RELATED WORK

Software engineering (SE) for ML is receiving increasing attention [12]. The related literature covers a broad range of
topics; from SE challenges raised by the adoption of ML components \[8\], to practices \[13\], guidelines \[14\], or design patterns \[15\]. Moreover, we consider the related field of SA for big data and analytics platforms \[2\]. Therefore, we structure the presentation in three steps: first we introduce SE challenges for ML (with a focus on SA), followed by solutions that meet the challenges and by a discussion on SA for big data and analytics, in the context of ML.

Arpteg et al. \[16\] introduced twelve SE challenges for ML, classified in three categories: development, deployment and organisational. From these, the challenges related to ML platforms, to monitoring and logging ML components and to effort estimation for development and maintenance, were also identified in our SLR. Since Arpteg et al. \[16\] do not introduce solutions, the second and third stages of our study can be used to complement theirs.

Similarly, Ishikawa and Yoshioka \[17\], as well as Wan et al. \[9\], studied how ML impacts the traditional software development life-cycle. Both studies are based on surveys and have the bulk of responses from Asia. Notwithstanding this regional bias, they concluded that testing and evaluating the quality of ML components is particularly difficult. Distinct conclusions are drawn with respect to SA. While Wan et al. \[9\] acknowledged SA for ML as difficult, Ishikawa and Yoshioka \[17\] concluded that existing SA methods apply equally to software with ML components, although the tool support is immature. We analysed the SA challenges raised by ML with finer granularity and found out that while some challenges apply equally to software with or without ML components, ML specific challenges (and solutions) also arise.

To classify the SE challenges for ML, Lwakatare et al. \[8\] introduced a taxonomy, from which the challenges related to scalability and serving were also identified in our study.

An early publication that outlined SA challenges and solutions for ML was the work of Sculley et al. \[3\]. The authors used the framework of technical debt to explore risk factors for ML components. Particularly, they argued that ML components are subject to all maintenance issues specific to software components, as well as to new issues specific only to ML. Moreover, they introduced a set of anti-patterns and practices used to avoid technical debt. Compared to Sculley et al. \[3\], the challenges (and solutions) introduced in this paper are broader and consider more quality attributes.

Breck et al. \[18\] and Zhang et al. \[19\] studied the topic of testing for ML components and introduced testing and monitoring practices for different stages of the ML development life-cycle. While these practices are relevant to SE for ML, we are interested in the architectural decisions made for testing ML components. Therefore, we focus on higher-level decisions, such as using automating tests or designing the testing pipelines.

Amershi et al. \[20\] conducted an internal study at Microsoft, aimed at collecting SE challenges and practices for ML. They reported on a broad range of challenges and practices used at different stages of the ML development life cycle. In particular, modularity and component reuse in software with ML components are challenges closely related to SA, which are also discussed in this study.

Serban et al. \[13\] and Zhang et al. \[14\] introduced two sets of SE practices for ML and deep learning, respectively. While some practices are considered in SA – e.g., the adoption of continuous integration – the broad selection of practices does not allow a focus on SA (as in our study). Therefore, the findings introduced can be used to complement theirs.

Nascimento et al. \[12\] introduced a SLR on the topic of SE for ML that analysed all articles up to 2019. The authors observed that SA is not yet a popular topic. However, their taxonomy classifies software quality and infrastructure concerns separately from architecture. We argue that such concerns are discussed extensively during SA and should be considered together when evaluating the popularity.

Washizaki et al. \[21\] studied SA patterns and anti-patterns for ML, extracted from white and grey literature. Their proposal was followed by a larger study, where the initial set of patterns and anti-patterns was extended \[15\]. Their work is close to the first stage of our study (SLR), where we identified a set of challenges and solutions in SA for ML. We build upon it by enlarging the number of challenges and solutions and by extensively validating our findings in the second and third stages of the study. Moreover, although the challenges presented by Washizaki et al. \[21\] are recurrent, we found out that the solutions are not. Therefore, we are cautious in using the taxonomy of design patterns. Instead, we focus on smaller building blocks called tactics; which bridge architecture decisions with quality attributes and form the basis of design patterns \[22\], \[23\].

The challenges raised by big data systems regarding continuous expansion of data volumes and the adoption of new technologies have been well studied and several reference architectures have been proposed – e.g., \[7\], \[2\], \[24\]. However, the proposals emphasise the data aspect of SA, i.e., how to collect and manage various sources of data and satisfy performance demands of analytics platforms. Therefore, although data visualisation and ML components are present in the reference architectures, these do not record decisions taken for development, integration or serving ML components. Here, we focus on the latter, where the data aspect plays an important role but it is not the sole decision driver.

III. STUDY DESIGN

Our study was organised in 3 stages and consisted of a mixed-methods approach with a sequential exploratory strategy \[25\]. In the first stage, we ran a SLR to identify the challenges faced when (re-) architecting systems with ML components and the solutions proposed to meet them. The second stage of the study consisted of semi-structured interviews, meant to complement and partially validate the data extracted in the first stage. In the third stage, we ran a survey to gather quantitative data, augment and generalise the findings from the first two stages. Data triangulation from multiple sources is known to increase the reliability of the results \[25\]. The design of the three stages is described below.
**Systematic Literature Review.** SLRs are widely used in empirical SE research and provide a structured process to identify, evaluate and interpret the information available regarding a research topic [26], [25], [27]. SLRs consist of three parts, namely defining a research protocol, conducting a review and reporting the results. We followed the guidelines from Kitchenham and Charters [26] and defined a research protocol as follows.

**Research questions.** We aimed to gather evidence about the challenges faced when (re-) architecting software systems with ML components. Moreover, we looked for solutions that meet the challenges and synthesise practices, tactics or patterns. Towards this goal, we formulated a set of research questions, summarised with their motivation in Table I. The questions facilitated the identification of challenges in the area of SA with ML components and enabled the creation of an initial body of knowledge with solutions.

**Search strategy.** To get a broad set of studies, we used multiple information sources. First, we used automatic queries to retrieve studies from several digital libraries, namely IEEE Xplore, ACM Digital Library (ACM DL), Scopus and ScienceDirect. Shahin et al. [28] observed that SpringerLink uses a different query mechanism than the others and that Scopus indexes most articles from SpringerLink. Therefore, in order to avoid inconsistencies in data retrieval, we relied on Scopus. Second, motivated by the findings of Serban et al. [13] – which noticed that most literature on the topic of SE for ML consists of so called grey literature – we performed manual search in Google and Google Scholar, where the first 5 pages of results were inspected. Last, we complemented the data set through a snowball strategy following references of relevant articles [29].

To define the search query, we followed the guidelines from [26] and composed a string with synonyms of the words “software architecture”, “machine learning”, “challenges” and “solutions”. After piloting several queries to validate the inclusion of previously known articles, we decided to use two distinct queries. The first query retrieved challenges in SA for ML and the second query retrieved solutions. The search string for the first query was: “((“software architecture” OR “software engineering” OR “systems engineering”) AND (“machine learning” OR “deep learning” OR “artificial intelligence” OR “AI”) AND (“challenge” OR “problem” OR “issue”))”, where the emphasised string was replaced in the second query with: “AND (“solution” OR “practice” OR “guideline” OR “tactic” OR “pattern” OR “architecture pattern” OR “design pattern”). Using the word “software” next to architecture or engineering helped to avoid articles from the general field of engineering (e.g., electrical engineering) or architecture. Moreover, we observed that including both “pattern” and “design pattern” makes the query more effective.

**Exclusion and inclusion criteria.** Since the initial queries returned over 10 000 results, we limited the answers to the first 500 articles for each data source and query. This reduced the number of articles to 1 000 per source, which corresponds to recommendations and previous studies [30], [28]. Washizaki et al. [21] and Nascimento et al. [12] showed that the majority of articles on the topic of SE for ML were published after 2016. Therefore, we also restricted our search to articles published after 2016. Next, we automatically filtered for duplicates and for records that contained the words “proceedings” or “workshop” in the title. Moreover, we manually excluded all opinionated articles; coming from companies or authors which could be traced back to companies that provide tools or services for SA/SE for ML. Thus, some bias regarding solutions driven by tools was avoided. Since ML for SE receives increasing interest, we curated and removed the articles on this topic because our study focuses on SE for ML. Also, we removed tool demonstration articles and those not written in English. In the final selection, we included all studies or grey literature articles that presented challenges or solutions based either on empirical studies or on experience (e.g., studies with empirical validation or organisation blogs describing their processes).

**Study selection.** After retrieving the initial set of 4 200 documents, we applied the selection criteria as follows. In the first phase we applied the automatic filters, which reduced the set to 2 613 articles. For these articles, we manually inspected the titles and the keywords and selected 66 articles to be completely assessed. These were read completely and critically analysed, which reduced their number to 29 relevant articles. From their references, 16 new articles were read, from which 13 were used in the final selection. The distribution

| ID | Research Question | Motivation |
|----|------------------|------------|
| RQ1 | Which are the challenges reported in (re-) architecting software systems with ML components? | Understand the technical and organisational challenges, but also the requirements posed by adoption of ML components. |
| RQ2 | What solutions, tactics or patterns have been reported to successfully meet these challenges? | Understand and identify solutions, tactics, or patterns for SA with ML components. |

**TABLE I: Research questions for the SLR.**

| Source | Retrieved | Automatic Filtering | Manual Inspection | Used |
|--------|-----------|---------------------|--------------------|------|
| ACM DL | 1000      | 647                 | 21                 | 7    |
| IEEE Xplore | 1000      | 521                 | 22                 | 9    |
| ScienceDirect | 1000      | 513                 | 2                  | 0    |
| Scopus | 1000      | 732                 | 2                  | 0    |
| Google Scholar | 100       | 100                 | 7                  | 3    |
| Google | 100       | 100                 | 12                 | 10   |
| Snowball | -         | -                   | 16                 | 13   |
| **Total** | **4 200** | **2 613**          | **82**             | **42** |

**TABLE II: Article selection for each information source.**
of articles and their sources for each stage of the review is presented in Table II. We observe that although ACM DL and IEEE Xplore retrieved the bulk of articles for complete assessment, the grey literature search and snowballing strategies were more effective for the final selection. Moreover, we observe that the distribution of articles per date resembles the one from [21], [12] – i.e., the number of articles is increasing year by year. Similarly, by analysing the distribution of articles based on the venue type we note that the majority of academic articles were published in conferences or workshops and not in journals. We conjecture that: (i) SA for ML is an emerging field and the publications did not reach the maturity needed for journal publication and (ii) journals have a longer publication cycle, therefore many publications may be in review. The complete list of articles, their sources and a demographic characterisation is attached in the supplementary materials (Appendix A and B).

Data extraction and synthesis. From all articles, we extracted and classified them into: (i) demographics and context, (ii) SA for ML challenges (RQ1), (iii) SA for ML solutions and tactics (RQ2), (iv) data types used. To analyse the demographics data we used descriptive statistics. To extract the data for (ii) and (iii), we used qualitative analysis methods. In particular, we used thematic analysis [31], which defines a process based on the following 5 steps: (1) familiarity with data – the articles were examined to form initial ideas, (2) initial code generation – the initial list of challenges and solutions was extracted, (3) theme search – common elements between the challenges and the solutions, respectively, were identified, (4) theme review – challenges and solutions were compared and common items were merged or dropped, (5) definition and naming – each challenge and solution was defined and named.

Interviews. To complement the results from the SLR, we conducted 10 interviews with practitioners.

Protocol. The interview protocol was designed following the guidelines from Hove and Anda [32] and consisted of 31 questions designed to support a natural conversation between the participants. All interviews were conducted online, through video calls (8 interviews) or e-mail (2 interviews). To enable participants to become familiar with the interview objectives [32], we shared a shorter version of the interview plan at least three days before the meeting.

The interviews were structured in 5 sections. First, we described the research goals and background. Second, we asked participants to share information about their background and demographics. Third, we asked to participants describe the constraints and challenges faced in SA for ML either for the last project they worked on or for a specific project. This part enabled a discussion about the challenges faced (and the solutions adopted) and was meant to complement the data obtained previously. Next, we asked participants to comment on each of the challenges from the SLR, evaluate their impact on SA and propose solutions. Last, we asked participants to provide open-ended comments. We continuously refined the questions and after the first three interviews the questions remained stable. Two questions were merged due to redundancy and one was modified to be more descriptive.

Participants. The interview participants were recruited using purposeful sampling [33]. We contacted participants with experience in (re-)architecting systems with ML components, or involved in architectural decisions (e.g., had the role of architect, or a leading position in engineering) and who are working (or worked) for companies using ML. To identify the participants, we used our personal network of contacts. Moreover, we compiled a list of organisations that use ML from outlets such as Forbes or MIT Technology Review. Later, we traced back candidates from the organisations (holding positions linked to SA) through LinkedIn and contacted them. The list of interview participants, together with the demographic characterisation is presented in Table III We observe that the participants’ background is diverse, ranging from software and system architects to engineering leaders and CTOs. Moreover, the participants’ and organisations’ experience is diverse – ranging from start-ups to large organisations with vast experience in ML. Since the organisations had different profiles, we classified them into (i) Technology – focus on developing technology products and (ii) Non-Technology – do not focus on technology products, but use ML for their processes. We also note that many participants had previous experience in research, being directly involved or in close collaborations with research groups. We hypothesise that the research driven process for ML is a contributing factor to this result.

Data analysis. The interviews were processed using thematic analysis, a technique which consists of the five steps recom-

| ID  | Position                     | Experience | Research | Org. Profile        | Org. Size | Org. Experience | ML | Team Size | Deployment Interval |
|-----|------------------------------|------------|----------|---------------------|-----------|-----------------|----|-----------|---------------------|
| P1  | Solutions Architect          | 3-5 years  | No       | Technology (Internet) | 10 000+  | 6-9 years       | 6-9 | 0-1 week  |
| P2  | System Architect             | 3-5 years  | Yes      | Non-Technology (Transport) | 10 000+  | 3-5 years       | 6-9 | 1-2 weeks |
| P3  | Software Architect           | 6-9 years  | Yes      | Technology (Automation) | 10 000+  | 3-5 years       | 10-15 | 3-4 weeks |
| P4  | Technology Lead              | 3-5 years  | Yes      | Technology (AI/ML)   | 0-50      | 0-2 years       | 10-15 | 0-1 week  |
| P5  | Software Architect           | 6-9 years  | Yes      | Non-Technology (Medical) | 10 000+  | 3-5 years       | 6-9 | 3-4 weeks |
| P6  | Software Architect           | 3-5 years  | No       | Technology (Automation) | 1000-5000 | 3-5 years       | 10-15 | 1-2 weeks |
| P7  | Director of Engineering      | 6-9 years  | No       | Technology (AI/ML)   | 51-200    | 3-5 years       | 6-9  | 3-4 weeks |
| P8  | Senior Solutions Architect   | 3-5 years  | No       | Technology (Space)   | 51-200    | 0-2 years       | 10-15 | 1-2 weeks |
| P9  | Head of Engineering          | 3-5 years  | No       | Technology (Robotics) | 0-50      | 1-2 years       | 16-20 | 3-4 weeks |
mended by Cruzes and Dyba [34]: (i) data extraction – the interviews were transcribed, read and key points were extracted, (ii) data coding – the initial SA challenges and tactics, as well as the impact of each challenge on SA (e.g., low or high impact) were defined, (iii) code to themes translation – for each transcript the initial codes were combined into potential themes (e.g., automated testing), (iv) high-order theme modelling – the themes were compared and merged, or dropped if the evidence was not sufficient (e.g., automated testing was merged in CI), (v) synthesis assessment – arguments for the extracted data were established, for example in terms of credibility (if the core themes were supported by the evidence) or confirmability (if there was consensus among the authors on the coded data).

Survey. To generalise the findings with a large sample size and augment the solutions, we ran an online survey. The survey was developed using the guidelines from Kitchenham and Pfleeger [35] and Ciolkowski et al. [36]. We designed a cross-sectional observational study asking participants at the moment of taking the survey which solutions they adopt, for each challenge. Moreover, we asked participants about their background in order to assign them to groups; making the study a concurrent control study in which participants are not randomly assigned to groups.

Questionnaire. The questionnaire consisted of five sections. In (i) the preliminaries we asked participants about their background (5 questions), to select a recent project where they played a role in SA for ML and to provide information regarding the challenges faced, the project constraints and the data types used (3 questions). Next, we asked participants to (ii) select or propose new solutions for the challenges identified previously (20 questions). Since multiple solutions involved instrumentation, monitoring or alerts, we added a question regarding the architectural decisions for designing these modules (1 question). Afterwards, we asked participants to (iii) select the architectural style (if any) adopted in their project (1 question) and to (iv) link the solutions to software quality attributes (1 question). The questionnaire ended with a section where (v) participants could provide open ended feedback (1 question).

The answers allowed multiple choices, with the solutions extracted from the SLR and from the interviews. Besides, we provided an open answer called ‘Others’, where participants could propose new solutions. The quality attributes used in the fourth section were extracted from the ISO/IEC 25010 standard [10], which is widely regarded as mature. However, we found the “Installability” and “Replaceability” attributes out-dated and replaced them with “Deployability”, which better reflects deployment and roll-back.

Survey Pilot. Before distributing the survey, we invited four candidates to assess the survey in our presence and suggest improvements. The participants did not consider any question redundant. Using their feedback, we added three new answers possible to the questions where the answers were considered incomplete and rephrased other answers and questions.

Distribution. To distribute the survey, we used a snowballing strategy. At first, we reached out to our network of contacts, asked them to fill in the survey and forward it to potential candidates. Second, we expanded the list of contacts from interview recruitment. In total, we sent 286 e-mails or private messages to potential participants. Third, we advertised the survey through open channels used by practitioners, i.e., Reddit and LinkedIn.

Data Analysis. We processed the standard answers using descriptive statistics and the open-ended answers using thematic analysis. Moreover, we analysed the association between the adoption of solutions using the Phi coefficient.

IV. RESULTS

We present the results from the three stages of the study as follows: (i) the SLR results in Section IV-A, (ii) the interview results in Section IV-B and (iii) the survey results in Section IV-C.

A. Results from the SLR

From the SLR we identified an initial set of 18 challenges, presented in Table IV. We note that the SLR data have numerical references. To classify the practices, we used a custom taxonomy because the ML taxonomy is different than the traditional SE taxonomy [20]. Moreover, ML taxonomies are divergent [13]. For example, Amershi et al. [20] present a nine-stage taxonomy for the ML process, while Sato et al. [61] use only six stages. These taxonomies have roots in the CRISP-DM model [67]. However, recent studies show these models are not fit for all contexts [68]. Since existing taxonomies are divergent, we constructed a broad taxonomy compatible with previous work and focusing on SA.

The taxonomy was used to classify the challenges (and solutions) in: (i) Requirements (Reqs.) – requirements elicitation for ML components, mapped to model requirements and business understanding [20], [67], (ii) Data – data collection, preparation and validation, mapped to data taxonomies [20], [13], [67], (iii) Design – the system’s structure, SA decisions and trade-offs, mapped to training and coding taxonomies [20], [13], (iv) Testing – testing and validation of software with ML components, mapped to the evaluation taxonomies [20], [67] and (v) Operational (Ops.) – deployment, monitoring and evolution, mapped to deployment taxonomies [20], [67].

RQ1. Answering RQ1 from Table I, we identified 18 challenges through the SLR, classified in five categories. The Reqs. challenges focus on the inability to understand a project and estimate the effort upfront. Moreover, the opaque nature of ML components – for which functional requirements are difficult to define and which may be subject to regulatory restrictions, emerged as challenging.

The Data challenges relate to data preparation and data quality assessment. This result contrasts previous concerns from big data and analytics platforms [2], where the focus was on data storage and accessibility. Nonetheless, this result corresponds with the expectation that ML components are evolved from big data platforms and therefore extend and overcome the challenges met there.
Table IV: List of SA challenges for ML, and related solutions.

| Nr. | Category Challenges                                                                 | Solutions                                                                 | References |
|-----|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------|------------|
| 1   | Reqs.                                                                               | At design time the information available is insufficient to understand the customers or the projects. |            |
|     |                                                                                     | Run simulations to gather data. Use past experience. Measure and document uncertainty sources. | [17], [37] |
|     |                                                                                     | Use metrics as functional requirements. Include understandability and explainability of the outputs. | [18], [39] |
| 2   | Reqs.                                                                               | ML components lack functional requirements.                                  |            |
|     |                                                                                     | Analyse regulatory constraints up-front. Adopt an AI code of conduct. Design audit trails. | [41], [42] |
| 3   | Reqs.                                                                               | ML projects have regulatory restrictions and may be subject to audits.        |            |
|     |                                                                                     | Design separate modules/services for data quality assessment. Integrate external tools. | [14], [44], [49] |
| 4   | Data                                                                                | Data preparation may result in a jungle of scrapes, joins, and sampling steps, often with intermediate outputs. |            |
|     |                                                                                     | Design separate modules/services for data collection and data preparation. Integrate external tools. | [31], [44], [10] |
| 5   | Data                                                                                | Data quality is hard to test, and may have unexpected consequences.          |            |
|     |                                                                                     | Standardise model interfaces. Use one middleware. Reuse virtualisation, infrastructure and test scripts. | [20], [47], [48] |
| 6   | Design                                                                              | Separate concerns between training, testing, and serving, but reuse code between them. |            |
|     |                                                                                     | Separate business logic from ML components. Standardise interfaces and use one middleware between them. | [49], [50] |
| 7   | Design                                                                              | ML components are highly coupled, and errors can have cascading effects.     |            |
|     |                                                                                     | Design independent modules/services for ML and data. Standardise interfaces and use one middleware. Relax coupling heuristics between ML and data. | [9], [51], [52] |
| 8   | Design                                                                              | ML components bring inherent uncertainty to a system.                        |            |
|     |                                                                                     | Use n-versioning. Design and monitor uncertainty metrics. Employ interpretable models/human intervention. | [53], [54], [55], [56] |
| 9   | Design                                                                              | ML components can fail silently. These failures can be hard to detect, isolate and solve. |            |
|     |                                                                                     | Use metric monitoring and alerts to detect failures. Use n-versioning. Employ interpretable models. | [57], [58], [59] |
| 10  | Design                                                                              | ML components are intrinsically opaque, and deductive reasoning from the architecture artefacts, code or metadata is not effective. |            |
|     |                                                                                     | Instrument the system to the fullest extent. Use n-versioning. Employ interpretable models. Design log modules to aggregate/visualise metrics. | [60], [61] |
| 11  | Design                                                                              | Avoid unstructured components which link frameworks or APIs (e.g., glue code). |            |
|     |                                                                                     | Wrap components in APIs/modules/services. Use standard interfaces and one middleware. Use virtualisation. | [41] |
| 12  | Design                                                                              | Automation and understanding of ML tasks is difficult (AutoML).             |            |
|     |                                                                                     | Version configuration files. Design the log and versioning systems to support AutoML data retrieval. | [9], [44], [48], [13], [58] |
| 13  | Design                                                                              | ML testing goes beyond programming bugs to issues that arise from model, data errors, or uncertainty. |            |
|     |                                                                                     | Design model and data tests. Use CI/CD. Use integration and unit tests. Use data ownership for test modules. | [16], [60], [20], [49] |
| 14  | Testing                                                                             | Validation of ML components for production is difficult.                     |            |
|     |                                                                                     | Use metrics and CI/CD for validation. Use alerts, visualisations, human intervention. Design release processes. | [17], [59], [61] |
| 15  | Testing                                                                             | ML testing goes beyond programming bugs to issues that arise from model, data errors, or uncertainty. |            |
|     |                                                                                     | Use metrics and CI/CD for validation. Use alerts, visualisations, human intervention. Design release processes. | [17], [59], [61] |
| 16  | Ops.                                                                                | ML components require continuous maintenance, retraining and evolution.      |            |
|     |                                                                                     | Design for automatic continuous retraining. Use CI/CD. Use automatic rollback. Use infrastructure-as-code. Adopt standard release processes. | [61], [62], [41], [49] |
| 17  | Ops.                                                                                | Manage the dependencies and consumers of ML applications.                   |            |
|     |                                                                                     | Encapsulate ML components in identifiable modules/services. Use authentication and access control. Log consumers of ML components. | [61], [63], [49] |
| 18  | Ops.                                                                                | Balance latency, throughput, and fault-tolerance, needed for training and serving. |            |
|     |                                                                                     | Design for batch processing (training) and stream processing (serving), i.e., lambda architecture. Physically isolate the workloads. Use virtualisation. | [44], [65], [15], [48] |
| 19  | Ops.                                                                                | Trace back decisions to models, data and reproduce past results.            |            |
|     |                                                                                     | Design for traceability and reproducibility; log pointers to versioned artefacts, version configurations, models and data. | [66], P1 |
| 20  | Org.                                                                                | ML applications use heterogeneous technology stacks which require diverse backgrounds and skills. |            |
|     |                                                                                     | Form multi-disciplinary teams. Adopt an AI code of conduct. Define processes for decision-making. Raise awareness about ML risks within the team. | [P1] |

The largest category of challenges, Design, includes both traditional SA challenges, such as managing component coupling and new ML specific challenges, such as managing inherent uncertainty or designing for development automation (AutoML). We also notice a challenge regarding the integration of ML components with traditional software components and business logic (7), which finds it difficult to distinguish failures between the two.

In contrast to Design, the Testing challenges are ML specific. Here, the focus is on model testing – which goes beyond programming bugs – and on validation for production – which does not rely on new features or bug fixes, but on measurements that must meet multiple criteria. For example, accuracy, robustness or bias.

In the Ops. category, the challenges relate to deployment, maintenance and resource usage between training and testing. We observe that maintenance of ML components is based on retraining and deploying models trained with new data, which erodes the boundaries between maintenance and evolution.

RQ2. Answering RQ2 from Table I through the SLR we found distinct solutions to each challenge in Table IV. A detailed list of solutions extracted from the SLR is presented in Appendix D while Table IV presents the solutions from all stages of the study. We note that 52% of solutions came from the SLR, while the rest came from later stages of the study.
could also identify the main decision drivers for specific data types. Here, we note that participants using Images & Videos or Time Series found “Scalability” and “Hardware” constraints as the main decision driver. Moreover, participants using Simulations were also driven by “Hardware” constraints. We also observed a new decision driver – called “Generalisation” – which describes the ability of a ML component to maintain training performance in production. This driver is related to challenge (10) and the solution suggested by participants was to use n-versioning; i.e., multiple versions of the ML models (some of which may be more trustworthy).

While evaluating the challenges extracted from the SLR, we asked participants to assess their impact on SA. The results are illustrated in Fig. 1 and use an ordinal scale with three possible options: low, medium or high impact. Challenge (3) could not be evaluated because the participants did not report regulatory restrictions. We believe this result is due to the fact that ML regulations are still in draft phase or not yet enforceable [71].

Within the challenges with the highest impact, we observe one traditional challenges that is strengthened by ML (component coupling (8)) and multiple ML specific challenges. For example, opaqueness of ML components (11) or training-serving resource management (18). The highest impact on SA comes from the need to continuously retrain ML components (16), while the lowest impact comes from ML task automation (13).

Besides challenges, the interviews allowed us to complement the initial set of solutions. In total, 48% new solutions came from the interviews. Participants provided new solutions for all challenges besides challenges (2) and (3). While the solutions for (2) were regarded as complete, (3) was disregarded because participants did not report regulatory constraints.

During the thematic analysis, we combined several solutions by bridging ML and SE terminology, while striving for conciseness. Here, we describe resulting themes which may be ambiguous due to name compression. Using the same type of interfaces for business logic and ML components, for all ML components or within all projects was modelled as the use of “standard interfaces”. Participants reported multiple techniques to standardise the interfaces, e.g., REST APIs, gRPC or more general contracts for service oriented architectures. While the techniques are project specific, the architectural decision to unify the interfaces is singular.

Moreover, using multiple versions of a ML model – also called ensembles of models in ML – is similar to n-versioning programming. Therefore, we grouped these solutions in the “n-versioning” theme.

The separation of concerns and encapsulation of code was modelled as one theme: “design separate modules/services”. Here, participants reported that code was either developed as separate modules or as independent services. This development included encapsulation for reuse.

Furthermore, we defined the use of the same middleware for business logic and ML components in training and serving as “use one middleware” and the development of dashboards or modules to analyse the ML components in “visualisations”. Comprehensive materials and traces for the interview analysis...
can be found in Appendices E and F, while a detailed description of the themes in J.

C. Results from the survey

In total, we received 53 answers from which we filtered out (using the preliminary questions) respondents who did not play a role in SA for ML. Moreover, we filtered out respondents who spent less than two minutes fulfilling the survey and respondents who answered less than 50% of the preliminaries or less than 50% of the technical questions. This process ensured only thoughtful answers were used in the analysis and entailed 48 complete answers.

**Demographics.** We first grouped respondents by demographics. We note that the majority of respondents (57%) work for Technology organisations and have between 3-5 (40%) or 1-2 (28%) years of experience. These results align with related work [17], [13] and are in line with expectations that Technology companies are early adopters of ML technologies. Other groups are also represented, i.e., Non-Technology (28%), Governmental Org. (9%) and Research labs (6%). Similarly, beginners which just started (13%) and very experienced respondents, with 6-9 years of experience (19%), are well represented.

We also grouped respondents by regions into Europe (53%), North America (34%) and Asia (13%). Here, we observe a slight over-representation of Europe and under-representation of Asia. The possible bias stemming from the grouping by regions will be discussed in Section V.

As with the interviews, we asked respondents about their team size, data types used and deployment intervals. This data is illustrated in Fig. 2a where the height of the bars represents the percentage of respondents and the height of the connections represents the percentage of respondent who fall in the target class. We observe that the majority of respondents belong to teams between 6-9 (43%) or 10-15 (34%) members. In particular, the majority of Technology and Non-Technology teams have between 6-9 members, while the majority of Research teams are larger, between 10-15 members.

Regarding data types, we observe that, with the exception of Audio and Graphs, the data types have similar distributions. Moreover, the majority of respondents using Tabular data deploy new versions between 1-2 weeks, while respondents using Images, Videos, Audio or Text between 3-4 weeks. We conjecture that this result relates to the ML techniques suitable to each data type, i.e., Images, Videos, Audio or Text models are based on deep learning, require longer training times and the collection of larger data sets. In contrast, Tabular data can be processed with more traditional ML techniques (e.g., Random forest). These techniques require smaller data sets and training time; making the teams more agile.

Overall, the demographics indicate that our survey data is diverse and resembles data from interviews and related work [17], [13]. More details about the demographic analysis can be found in Appendix H.

**Decision drivers.** Second, we asked respondents about the most important decision drivers in their projects. This data is illustrated in Fig. 2b. We observe that “Scalability” and “Hardware” are consistent with the interviews and occupy leading positions. To better understand the data challenges, we divided them into "Low data quality" and "Scarcity of data". Taken together, data related concerns are consistent with the interviews. Separately, they were considered equally important, but none of them ranks high.

We also note that “Performance”, “Interpretability” and “Interoperability” rank higher for survey respondents, while “Privacy” and “Security” rank very low. This result is cause of concern, since documents from policy makers and advisory bodies suggest these topics are paramount for trustworthy development of ML [72]. We conjecture that, although a large body of academic literature on security of ML exists, it is still limited in its applicability. For example, all defences against adversarial examples – a known threat for ML components – have been breached [73]. The data also indicates...
that respondents prioritise operational quality attributes, such as scalability or performance in spite of security or privacy. 

**Solutions to challenges in Table [IV]** Third, using the survey results we filtered out and ranked the solutions from previous stages of the study. In particular, we considered the solutions that were selected less than 5% of the time as not relevant and filtered them out. Moreover, we used the number of times the solutions were selected by respondents to rank them. The ranking is reflected in Table [IV] by the order in which the solutions are presented.

For all challenges, respondents could also suggest new solutions or provide comments using the “Other” field. In total, we received three suggestions and open comments. We analysed the results using thematic analysis and found out that all suggestions were variations of the solutions provided or comments suggesting some solutions do not apply. For example, one respondent mentioned that, due to tight performance constraints, it was not possible to apply n-versioning. The comment suggests that, given exceptional constraints, some solutions do not apply. This result is expected since the first two stages of the study strove for generalisation and outliers may exist. Nonetheless, the lack of novel suggestions for solutions brings evidence that the first two stages of the study entailed comprehensive solutions to all challenges.

We also note that some solutions are recurrent and can be applied to multiple challenges. For example, the use of standard interfaces for ML components and business logic or the use of interpretable models. These results are expected since architectural decisions may impact multiple aspects.

As mentioned previously, an extra question was added for decisions regarding instrumentation, monitoring and alerts. The solutions were inspired by interviews; where participants reported the development of independent logging, alert or visualisation modules. Here, we note that the majority of survey respondents reported the development of independent modules/services for instrumentation and monitoring. Moreover, respondents reported on separating logging concerns between training and serving and on the development of independent modules to aggregate and visualise logs. A small percentage of respondents used external tools for instrumentation.

An elaborate analysis of the survey answers is presented in Appendix I and a description of the solutions in Appendix J. 

**Associations between solutions.** Fourth, we analysed the associations between the adopted solutions. For this analysis, we modeled the adoption of solutions as dichotomous variables and used the Phi coefficient. To determine the statistical significance of the observed associations, we performed Chi-Square tests with a significance level of 0.05. We found multiple significant medium to strong associations ($\phi > 0.4$), of which we report an illustrative selection. We also analysed the associations between solutions using the Jaccard similarity, which entails analogous results.

For example, designing separate modules/services for data quality assessment (5) is associated with the design of independent modules/services in component coupling (7) ($\phi = 0.43$). Moreover, the use of one communication middleware to reduce coupling (7) is associated with standardisation and reuse of model interfaces between training and serving (6) ($\phi = 0.51$). Design of independent modules/services in component coupling (7) is also associated with CI/CD in maintenance of ML component (16) ($\phi = 0.54$). These results indicate the solutions may be complementary and suggest their joint adoption can be interdependent and incremental.

Similarly, model tests (15) are associated with data tests (15) ($\phi = 0.63$) and integration tests (15) are associated with test automation (15) ($\phi = 0.88$). However, ML tests (e.g., data tests) are not associated with traditional software tests (e.g., unit tests) or with test automation. These results indicate a separation between ML and SE concerns exists. Moreover, they indicate that mature teams jointly adopt advanced test practices, as also noticed in [13].

**Architectural styles for ML.** Next, respondents were asked to select the architectural styles employed in their projects. The results are illustrated in Fig. [23] and indicate that the majority of respondents used the event-driven style. Nonetheless, the difference between event-driven, lambda and micro-service/SOA architecture styles is not large. Although we did not find significant associations between the architectural styles, the lambda architecture can be used concomitantly with other architectural styles. We also searched in literature for evidence to strengthen these findings, but could not find any study on this topic.

**Quality attributes.** Last, respondents were asked to link the solutions to software quality attributes (characteristics) from ISO/IEC 25010 [10], which enabled to restate them as architecture tactics [23]. Tactics are architectural building blocks from which design patterns can be created and represent architectural decisions that improve individual quality attributes [22], [23]. Therefore, the results of this analysis provide direct guidance for practitioners who aim to improve

### Table V: Solutions mapping to the ISO/IEC 25010 model.

| Characteristics | Sub-characteristics | Solutions |
|-----------------|---------------------|-----------|
| Func. Suitability | Func. Completeness | 1, 2, 3 |
|                 | Compliance         | 3 |
| Performance efficiency | Capacity | 2, 11 |
| Compatibility    | Co-existence       | 7, 8 |
|                 | Interoperability   | 6, 12 |
| Usability        | User Error Protection | 19, 20 |
| Reliability      | Availability       | 10, 16, 18 |
|                 | Fault Tolerance    | 9, 10 |
| Security         | Modularity         | 4, 5, 7, 8, 12 |
|                 | Reusability        | 4, 5, 6, 8 |
|                 | Analysability      | 11, 17 |
| Maintainability  | Modifiability      | 4, 5 |
|                 | Testability        | 5, 7, 9, 10, 11, 14, 15 |
| Portability      | Adaptability       | 13, 16, 18 |
|                 | Deployability (with Instal-lability and Replaceability) | 12, 16, 17, 18 |
specific quality attributes of systems with ML components.

Since the solutions do not presume a ranked order, we considered all solutions equally important. The final results are presented in Table V. We note that “Scalability” and “Interoperability” – considered important decision drivers (Fig. 25) – are addressed by multiple solutions. Similarly, “Maintainability”, considered to have the biggest impact on SA by interview participants (Fig. 1), is addressed by the largest number of solutions. We also observe that some quality attributes from the standard (e.g., Operability or Maturity) are not addressed by any solution and note this result does not imply that missing quality attributes are not challenging. Instead, some quality attributes may not be applicable or require adaptation to accommodate ML components, as previously suggested by Kuwajima et al. [74].

We also mention that the “Compliance” sub-characteristics are not present in the quality standard because compliance is considered part of the overall system requirements. Therefore, “Compliance” spans all characteristics in Table V. To avoid confusion, we represent “Compliance” as a sub-characteristic of “Functional Suitability”.

V. DISCUSSION

We comment on several aspects of the study. First, regarding the challenges discovered from interviews, we analysed the percentage of survey respondents who did not have a strategy to tackle them. We found out that 23% of respondents had no strategy for challenge (19) and 22% of respondents had no strategy for challenge (20). These results show that more than 75% of respondents tackled these challenges and bring evidence that both challenges are relevant, in spite of the fact that they were mentioned in one interview each. Moreover, the answers for challenge (20) have similar distributions for teams consisting of 6-9 and 10-15 members, suggesting the challenge is not motivated by team size.

Second, both in the interviews and in the survey, the architectural decision drivers for trustworthy ML [72] – i.e., “Robustness”, “Security”, “Privacy” – were not considered critically important by respondents. McGraw et al. [75] argue that, from a security engineering perspective, the SA of systems with ML components is an important first step. However, the results of our study show that practitioners focus on lower level concerns, such as performance or scalability. Similar data was also observed previously [76].

Third, we noticed that some solutions have low adoption. For example, self-adaptation for managing inherent uncertainty (9) was used by less than 5% of respondents. While this result may seem surprising, according to Mahdavi-Hezavehi et al. [77] the number of self-adaptation techniques for “automated learning”, i.e., ML components, is small. We conjecture that a small number of solutions are not applied because they are not fit, or still prototypes in academia.

Last, when mapping the solution to software quality attributes, we used the mature and authoritative ISO/IEC 25010 standard [10]. At the moment, no similar model exists for ML components, although policy makers indicate such models are under development [71]. The results from our study indicate that, with the exception of “Interpretability”, ML specific quality attributes are not yet critical SA decision drivers. However, we expect this to change once mature quality models for ML are available.

Threats to validity. We identified three potential threats to validity, corresponding to the three stages of the study. First, the SLR can be affected by missing or exclusion of relevant papers. To mitigate this threat, we used multiple digital libraries for information retrieval. Additionally, we complemented the results with grey literature (manual search) and through snowballing. The researchers’ bias in the data extraction was prevented using a data extraction form (Appendix C), which allowed consistency in data analysis and through discussions between authors with different backgrounds.

Second, the data from interviews may be subject to bias. To limit this bias, we analysed the participants’ profiles and ensured they have relevant experience for the study. We recruited participants with diverse backgrounds and experience, working for organisations with distinct sizes and experience in ML. We also used two strategies to alleviate memory bias, i.e., we shared a short version of the interviews before the meeting and asked participants to share their experience from a recent project in the preliminaries. Moreover, we assured participants of data confidentiality and anonymsation, in order to limit participants from answering the questions in a manner that would better position them.

Third, to limit the survey bias we included additional fields besides the answers from SLR or interviews (e.g., Other fields for all challenges). We also advertised the survey to diverse groups, in order to limit selection bias. Nonetheless, as shown in Section IV-C, some groups of respondents are under-represented and may introduce selection bias. This bias can be removed by gathering more data, as we plan to do in the future. Last, to avoid researchers’ bias, we used data triangulation from multiple sources.

VI. CONCLUSIONS AND FUTURE RESEARCH

We studied how systems can be (re-)architected to enable robust adoption of ML components. We ran a mixed-methods empirical study consisting of: (i) a SLR which revealed 42 relevant articles, from which we compiled 18 SA challenges (and solutions) for ML, (ii) 10 semi-structured interviews which revealed 2 new challenges and 28 new solutions and (iii) a survey with 48 architects.

We reported on the impact of each challenge on SA and the main SA decision drivers for ML. Moreover, we established a link between solutions and quality attributes from the ISO/IEC 25010 standard, which allowed us to provide practitioners with twenty architectural tactics for systems with ML components.

For future research we plan to increase the number of respondents to the survey and enable more robust analyses. Moreover, we plan to add depth to the interpretation of the results through validation interviews and expand the quality attributes from ISO/IEC 25010 with ML specific quality attributes.
[52] A. Homeman, A. Mellinger, and I. Ozkaya, “AI engineering: 11 foundational practices,” tech. rep., Carnegie Mellon University, 2020.

[53] A. Aniculaesei, J. Grieser, A. Rausch, K. Rehfeldt, and T. Warnecke, “Towards a holistic software systems engineering approach for dependable autonomous systems,” in International Workshop on Software Engineering for AI in Autonomous Systems, pp. 23–30, IEEE, 2018.

[54] A. C. Serban, “Designing safety critical software systems to manage inherent uncertainty,” in International Conference on Software Architecture, pp. 246–249, IEEE, 2019.

[55] A. Biondi, F. Nesti, G. Cicero, D. Casini, and G. Buttazzo, “A safe, secure, and predictable software architecture for deep learning in safety-critical systems,” IEEE Embedded Systems Letters, 2019.

[56] E. Woods, “Software architecture in a changing world,” IEEE Software, vol. 33, no. 6, pp. 94–97, 2016.

[57] M. Scheerer, J. Klamroth, R. Reussner, and B. Beckert, “Towards classes of architectural dependability assurance for machine-learning-based systems,” in International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 31–37, 2020.

[58] G. Rios, “Patterns (and anti-patterns) for developing ML systems.” https://www.usenix.org/legacy/events/sysml08/tech/rios_talk.pdf, 2019. [Online; accessed 12-08-2021].

[59] Y. Nishi, S. Masuda, H. Ogawa, and K. Uetsuki, “A test architecture for machine learning product,” in International Conference on Software Testing, Verification and Validation workshops, IEEE, 2018.

[60] L. Reimann and G. Kniesel-Wünsche, “Achieving guidance in applied machine learning through software engineering techniques,” in International Conference on Art, Science, and Engineering of Programming, pp. 7–12, 2020.

[61] D. Sato, A. Wider, and C. Windheuser, “Continuous delivery for machine learning.” https://martinfowler.com/articles/cd4ml.html, 2019. [Online; accessed 12-08-2021].

[62] D. Baylor, K. Haas, K. Katsiapis, S. Leong, R. Liu, C. Menwald, et al., “Continuous training for production ML in the tensorflow extended (TFX) platform,” in USENIX Conference on Operational Machine Learning (OpML 19), pp. 51–53, 2019.

[63] J. Everett, “Daisy architecture.” https://datalanguage.com/blog/daisy-architecture, 2019. [Online; accessed 12-08-2021].

[64] A. Batyuk, V. Voityshyn, and V. Verhun, “Software architecture design of the real-time processes monitoring platform,” in International Conference on Data Stream Mining & Processing, pp. 98–101, IEEE, 2018.

[65] P. Menon, “Demystifying data lake architecture.” https://www.datasciencecentral.com/profiles/blogs/demystifying-data-lake-architecture, 2020. [Online; accessed 12-08-2021].

[66] S. Chan, “A design pattern for machine learning with scala.” https://www.youtube.com/watch?v=hhXs4AOGRpI, 2020. [Online; accessed 12-08-2021].

[67] R. Wirth and J. Hipp, “Crisp-dm: Towards a standard process model for data mining,” in International conference on the practical applications of knowledge discovery and data mining, pp. 29–39, Springer, 2000.

[68] M. Haakman, L. Cruz, H. Huijgens, and A. van Deursen, “AI lifecycle models need to be revised. an exploratory study in fintech,” arXiv:2010.02716, 2020.

[69] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Larivi`ere, A. Beygelzimer, F. d’Alch´e Buc, E. Fox, and H. Larochelle, “Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program),” arXiv:2003.12206, 2020.

[70] P. Kruchten, “What do software architects really do?,” Journal of Systems and Software, vol. 81, no. 12, pp. 2413–2416, 2008.

[71] A. Jobin, M. Ienca, and E. Vayena, “The global landscape of ai ethics guidelines,” Nature Machine Intelligence, vol. 1, no. 9, 2019.

[72] High-Level Expert Group on AI, “Ethics guidelines for trustworthy AI.” https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai, 2019. [Online; accessed 12-08-2021].

[73] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns, “A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements,” in Managing Trade-Offs in Adaptable Software Architectures, pp. 45–77, Elsevier, 2017.

[74] A. Serban, K. van der Blom, H. Hoos, and J. Visser, “Practices for engineering trustworthy machine learning applications,” IEEE 1st Workshop on AI Engineering, 2021.

[75] A. Serban, K. van der Blom, H. Hoos, and J. Visser, “Practices for engineering trustworthy machine learning applications,” IEEE 1st Workshop on AI Engineering, 2021.

[76] A. Serban, K. van der Blom, H. Hoos, and J. Visser, “Practices for engineering trustworthy machine learning applications,” IEEE 1st Workshop on AI Engineering, 2021.