Exploring Requirements for Software that Learns: A Research Preview

Marie Farrell[1], Anastasia Mavridou[2], and Johann Schumann[2]

[1] Department of Computer Science, The University of Manchester, UK
[2] KBR Inc. / NASA Ames Research Center, USA

Abstract. Context & motivation: The development of software that learns has revolutionized how many systems perform. For the most part, these systems are neither safety- nor mission-critical. However, as technology and aspirations advance, there is an increased desire and need for Machine Learning (ML) software in safety- and mission-critical systems, e.g., driverless cars or autonomous space robotics. **Problem:** In these domains, reliability is crucial and systems have to undergo much scrutiny in terms of both the developed artefacts and the adopted development process. Central to the development of such systems is the elicitation and definition of software requirements that are used to guide the design and verification process. The addition of software components that learn, and the associated capability for unforeseen behavior, makes defining detailed software requirements especially difficult. **Principal ideas/results:** In this paper, we identify unique characteristics of software requirements that are specific to ML components. To this end, we collect and examine requirements from both academic and industrial sources. **Contribution:** To the best of our knowledge, this is the first work that presents real-life, industrial patterns of requirements for ML components. Furthermore, this paper identifies key characteristics and provides a foundation for developing a taxonomy of requirements for software that learns.

1 Introduction

The design of critical systems begins with the definition of natural-language objectives and high-level requirements. Once defined, high-level objectives and requirements are subsequently decomposed into detailed system- and module-level requirements. Any subsequent verification and validation effort must support traceability of requirements and provide evidence that they are upheld. In fact, requirements traceability is prescribed by a number of international standards including DO-178C for the aerospace domain [21].

This process is well-understood for traditional systems since these systems typically operate in constrained, well-defined environments and thus usually exhibit predictable behavior. As a result, requirement analysis for such systems returns absolute answers, i.e., absolute success (requirements are satisfied by the system) or absolute failure (requirements are not satisfied by the system).

Non-traditional systems, on the other hand, such as ones that rely on Machine Learning (ML) components, bring uncertainty that impacts requirements elicitation and analysis. Requirements for ML components often use probabilities to quantify uncertainties about system behavior and the environment. Furthermore, their analysis may return either absolute or probabilistic answers.

Similar challenges are encountered and must be addressed for autonomous systems that often rely on AI components, such as ML for key behaviors. In general, systems consist of both hardware and software components and requirements must be defined for both of these aspects. In this paper, we are primarily concerned with software requirements, although we recognize that sometimes software and hardware requirements cannot be completely separated.

Since accurate requirements provide the underlying properties against which a system should be verified, this research preview paper provides a starting point for those interested in verifying ML systems. Understanding the nature of requirements for systems that incorporate ML components is important, particularly if these systems are to operate in critical domains e.g., aerospace.

Our ultimate goal is to extend NASA's Formal Requirements Elicitation Tool (FRET)[3] [12,19] for capturing, formalizing, and analyzing requirements for software that learns. To achieve this, we must first develop an understanding of the nature of such requirements and what distinguishes them from requirements for classical systems. Thus, our research plan involves: (1) examine existing requirements, (2) determine the commonalities and differences between requirements for classical systems and those that learn, (3) identify the specific unique characteristics of requirements for systems that learn, (4) develop a taxonomy of requirements, (5) classify and examine those identified in (1), and (6) make appropriate extensions to FRET to support requirements for software that learns. This research preview paper describes our initial findings from exploring a corpus of such requirements, focusing on steps (1)–(3) above.

We observed an increasing number of research papers discussing aspects and challenges of the requirements engineering discipline in the development of autonomous, AI-based systems [5,15,23,16]. The aspects identified in these papers are usually discussed at a high level without showcasing real-life requirements. We differ from the above related work by providing specific requirements and requirement patterns from industrial case studies and missions. The requirements that we have gathered are primarily from the aerospace domain with some autonomous driving examples. We distill the common features amongst these requirements and provide a basis for developing a taxonomy of ML requirements.

2 Requirements for Autonomous Systems

According to a recent survey [17], requirements engineering is one of the most challenging activities for ML-related system development. This is primarily due

[3] `https://github.com/NASA-SW-VnV/fret`

Req ID	Requirement	Source
	Literature Studies	
[**LR-001**]	The aircraft location does not exceed a specified lateral offset from the runway centerline during taxiing.	[4,2]
[**LR-002**]	The aircraft does not veer off the sides of the runway during taxiing.	[4,2]
[**LR-003**]	All bounding boxes produced shall be no more than 10% larger in any dimension than the minimum sized box capable of including the entirety of the pedestrian	[14]
[**LR-004**]	The ML component shall determine the position of the specified feature in each input frame within 5 pixels of actual position.	[14]
[**LR-005**]	The ML component shall identify the presence of any person present in the defined area with an accuracy of at least 0.93	[14]
[**LR-006**]	The ML component shall perform as required in the defined range of lighting conditions experienced during operation of the system.	[14]
[**LR-007**]	The ML component shall identify a person irrespective of their pose with respect to the camera.	[14]
[**LR-008**]	When Ego is 50 metres from the crossing, the object detection component shall identify pedestrians that are on or close to the crossing in their correct position	[11]
[**LR-009**]	In a sequence of images from a video feed any object to be detected should not be missed more than 1 in 5 frames.	[11]
[**LR-010**]	Position of pedestrians shall be determined within 50cm of actual position.	[11]
[**LR-011**]	The object detection component shall perform as required in all situations Ego may encounter within the defined ODD.	[11]
[**LR-012**]	The object detection component shall perform as required in the face of defined component failures arising within the system.	[11]
	R-RAV Project	
[**RRAV-001**]	The neural network shall output the cross track distance error (perpendicular distance from the rover to the centerline.) Error to truth must not exceed X.	[R-RAV]
[**RRAV-002**]	Neural network shall output cross track heading error (the angle between the rover heading and the centerline.) Error to truth must not exceed X.	[R-RAV]
[**RRAV-003**]	Upon receiving an image, the Neural Network shall output the distance and the angle within X seconds (latency).	[R-RAV]
[**RRAV-004**]	Neural network shall output a sensible distance: the value must be between 0 and half the width of the taxiway plus X (buffer X so that it can still report if it is off the taxiway).	[R-RAV]
[**RRAV-005**]	Neural network shall output a sensible angle: the value must be between -90 and 90 degrees.	[R-RAV]
[**RRAV-006**]	The neural network shall achieve a minimum of X% accuracy on training and Y% accuracy on testing.	[R-RAV]
[**RRAV-007**]	(Local robustness) The neural network shall be robust to small perturbations in the image (pixels).	[R-RAV]
[**RRAV-008**]	(Semantic variations) The neural network shall be robust to irrelevant variations in the scene.	[R-RAV]
[**RRAV-009**]	The neural network shall safely navigate intersections.	[R-RAV]
[**RRAV-010**]	The magnitude of the cross track distance error shall drop below X m within T seconds and remain there.	[R-RAV]
[**RRAV-011**]	The magnitude of the cross track heading error shall drop below X degrees within T seconds and remain there.	[R-RAV]

Table 1. Requirement examples from literature review and the R-RAV project.

to uncertainties around the exact input/output behavior of ML components. To better understand such uncertainties, we present requirements that we have gathered from different sources, as shown in Tables 1 and 2.

Req ID	Requirement Pattern (source: NASA)
[**IC-001**]	The sw shall achieve an average PARAMETER value of X.
[**IC-002**]	The sw shall estimate PARAMETER to within $+-X$ with a $Y\%$ confidence.
[**IC-003**]	The sw shall estimate the confidence of the PARAMETER estimate.
[**IC-004**]	The requirement shall be verified by measuring the average of the parameter over N repetitions.
[**IC-005**]	The sw shall estimate PARAMETER with an $X\%$ confidence interval of no more than $+-Y$.
[**IC-006**]	The sw shall calculate the PARAMETER confidence interval at an $X\%$ confidence level.
[**IC-007**]	The sw shall calculate the PARAMETER as a probability distribution.
[**IC-008**]	The sw shall determine PARAMETER with a high level of confidence.
[**IC-009**]	The sw shall detect $X\%$ of occurrences of EVENT.
[**IC-010**]	The risk-ratio requirements shall be verified using a statistically significant set of SCENARIOS.
[**IC-011**]	The sw shall cause EVENT at a rate less than X times per Y DURATION.
[**IC-012**]	The sw shall detect CONDITION that implies EVENT is probable.
[**IC-013**]	The sw shall take action so that the risk ratio thresholds are satisfied.

Table 2. Requirement patterns extracted from missions and industrial case studies.

Studying the Literature: We examined literature from the assurance case domain [2,4,9,3,11,14]. Requirements [**LR-001**] and [**LR-002**] in Table 1 refer to the TaxiNet system [10], which uses a vision-based neural network to predict an aircraft's position on the runway relative to the center-line to enable autonomous runway taxiing. We recognise that [**LR-001**] and [**LR-002**] are quite high-level and might also apply to non-autonomous systems with similar goals. These are system-level requirements and likely have implications for both hardware and software. We consider them as specific to the ML component because TaxiNet is driven by a neural network whose output is directly related to the preservation of these requirements. The autonomous driving requirement [**LR-005**] is a functional requirement about the ML component.

R-RAV Project: Table 1, Part 2 shows requirements that we elicited in conjunction with developers as part of the R-RAV project at NASA Ames, which takes a similar approach to TaxiNet and is driven by a neural network. These requirements are likely more detailed than those in [2] because the focus was on the elicitation, formalization, and verification of detailed requirements rather than on assurance case development. [**RRAV-005**], [**RRAV-006**] and [**RRAV-007**] specifically refer to the neural network. Conversely, [**RRAV-011**] describes the behavior of the neural network based control system. Such requirements about control system behavior can also be found in traditional systems.

Missions/Industrial Case Studies: Table 2, contains 13 sanitized requirement patterns, which were obtained after manually analyzing 770 requirements from missions and industrial case studies that use AI. We call these *patterns* as they do not contain specific system details. Upper case variables must be instantiated by actual names and values to yield a multitude of similar requirements. Many of the requirements shown in Table 2 specify constraints on the computed parameters (e.g., [**IC-001**]) and have a notion of confidence (e.g., [**IC-002**]) which we discuss in §3. Other requirements explicitly mention probabilities, e.g., that a particular event is probable once a specific condition is met ([**IC-012**]).

3 ML Requirement Attributes and Characteristics

By looking at the requirements in Tables 1 and 2, we observe that notions of *confidence*, *accuracy*, and *average value* are often used to describe probabilistic requirements. However, the semantics of these terms is not always clear; we elaborate below. We also consider other characteristics including robustness, data-driven learning and quality aspects.

3.1 Confidence, Criticality, and Risk Levels

Requirements [**LR-001**] and [**LR-002**] are associated with different *confidence levels*. Although these confidence levels are not part of the requirement text, they have been added in [4,2] through separate requirement attributes. For example, [**LR-001**] must hold 95% of the time (lower confidence level), while [**LR-002**] must hold 100% of the time (higher confidence level), i.e., the TaxiNet system must avoid any runway excursion.

Confidence levels are tightly related to *risk levels*. For each hazard (e.g., lateral offset violation) a risk is calculated based on the likelihood and severity of the event [22]. Risk factors are usually defined in relevant standards, e.g., see risk ratios in [1]. The level of risk of an event determines the level of risk associated with the corresponding requirement and implies the required confidence level that must be achieved. E.g., [**LR-002**] is associated with a high risk level since it should not be violated under any circumstance. On the contrary, [**LR-001**] is lower on the risk scale, since it might be violated without unwanted consequences.

Confidence levels may consist of *quantitative* and *qualitative* components. A confidence level may indicate the amount of testing necessary for the result to be accepted, the accepted success rate, the type of analysis that must be performed to verify the property, etc. All of these characterize *the level of confidence that must be achieved in order to assure that the corresponding requirement is met.* Consider, e.g., [**LR-001**], where the quantitative component of the confidence level requires: (1) a 95% success rate for TaxiNet to be within a specified lateral offset and (2) a way of ensuring (during V&V) that this measure is achieved. Qualitative components do not contain explicit numbers. For a requirement with a high criticality level the qualitative component might recommend specific (combinations of) verification techniques to ensure greater coverage and rigor.

Although confidence levels for [**LR-001**] and [**LR-002**] are not part of the requirement text, confidence levels do appear in others. For example, in [**IC-002**] the parameter estimation must hold with "Y%" confidence. Confidence levels are frequently used to inform the choice and/or combination of verification methods. For example, requirements with low confidence levels may be only tested (up to a desired level of coverage) whereas requirements with high confidence levels may require testing alongside formal verification methods and runtime monitoring [18]. For critical systems, assurance cases typically contain an argument that the confidence level is met by sufficiently rigorous verification and implemented mitigations.

3.2 Accuracy as a Measure of Functional Correctness

The notion of accuracy is frequently encountered in ML system requirements [6]. Accuracy is quantifiable and evaluates the functional correctness of an ML model (accuracy during training) or the correctness of the output of an ML component (accuracy during testing/execution). Recent work defines and provides a way of calculating the accuracy of an ML system based on the results obtained while in training and during testing/deployment [20]. Essentially, accuracy defines the rate at which the ML must answer correctly. In this respect, accuracy might be viewed as a quantitative aspect of confidence, e.g., the required success rate of an ML system.

Specifically, [**RRAV-006**] defines the required levels of accuracy during both training and testing of a neural network. [**LR-005**] requires an accuracy level of at least '0.93' during execution — this is the required rate that the ML component correctly identifies the presence of a person. [**IC-002**] incorporates both accuracy and confidence. It requires an accuracy interval, a bound within which a specific parameter should be estimated, but it also includes a percentage for the confidence level that needs to be achieved, i.e., an acceptable success rate.

We recognise that accuracy is also important in classical control systems that often rely on (potentially noisy) sensors (e.g. aircraft engine controllers). For ML components, accuracy also refers to the decision-making process. For example, an inaccurate control system produces a value outside of a correct range, it is not capable of misidentifying the presence of a human in a roadway. Further, it is possible to enumerate the outputs (even erroneous ones) of a control system but the output behaviours of a system driven by ML can often not be completely predetermined (as new behaviours may be learned). The control system inaccuracy is also more straightforwardly reproduced than ML inaccuracies since the system continues to learn.

3.3 Achievement of Average Value

We also frequently encountered the word *average* in our requirement examples (e.g., [**IC-004**]). The *achievement of an average value* measure evaluates the *consistency* of the output data of an ML system. The requirement provides the target average value (X) of the ML output data and testing/analysis techniques

need to assure that, on average, the ML component meets the target value (X). This notion of an average value is frequently encountered in requirements for ML systems but usually less often in requirements for traditional systems, which typically deal with mode logic or embody implicit state machines rather than numerical algorithms.

3.4 Robustness

ML components, such as deep neural networks, are vulnerable to adversarial perturbations in the input. This means that small changes to the input may cause the network to produce erroneous output and has implications for both safety and security [13]. A related property is *local robustness* which specifies that all points within a particular distance from one another be given the same label. E.g., **[RRAV-007]** requires that each input similar to one encountered during training produces a similar output. This kind of requirement is unique to ML systems since it refers to the training and the test set as well as previously unseen data.

3.5 Data-Driven Learning

Most ML systems are data-driven. In particular, neural networks are *trained* using *training data* (e.g. set of images) which can often contain synthetic data. After training, ML systems are tested on *testing data* that should be separate from the training data and is expected to be representative of the physical deployment environment. Once deployed, the system may continue to learn using previously unseen data that were not present in either the training or testing data. Although not discussed in this paper, requirements are necessary to specify data coverage, specific conditions (e.g., weather conditions for TaxiNet), data accuracy and validation, as well as data acquisition and processing. Care should be taken to ensure that all of the data in the training set have been accurately labelled during a pre-processing phase [14]. Further, the data used should be relevant, complete, accurate and balanced [11]. Details on requirements for ML data can be found in the proposed EASA guidelines [8].

3.6 Quality Aspects

The presented requirements are formulated as detailed and often low level. They tend to refer to the specific way that the system or component should achieve its goal and the allowable bounds for particular variables/computations. In addition to a new flavour of requirements, developers and users of ML systems are interested in quality aspects such as *explainability*, *transparency*, *fairness*, and *ethical* requirements. We have not encountered such examples but we understand that, as this field progresses, they will become necessary, in particular, for ML systems that are deployed in close proximity to humans [7].

4 Uncertainty in ML Requirements

Clearly, many of the characteristics identified in §3 relate to uncertainty and probability plays a distinguishing role in ML requirements. As such, below we examine the probabilistic nature of requirements for ML systems.

Not Always Probabilistic: ML systems contain uncertainty. However, some of the requirements and the way that the system functions follow deterministic behavior as is usual for traditional systems. E.g., requirements [**LR-002**] (confidence level 100%) and [**RRAV-005**] describe specific behaviors that do not, at least at a high-level, create or rely on uncertain behavior. To this end, even though the addition of ML increases uncertainty, there are still more traditional requirements that the system should uphold. These requirements can be captured with existing requirement engineering (RE) tools that do not necessarily support probabilities and be verified using existing methods.

Probabilities Within Requirements: The majority of the requirements that we collected *contain* probabilities. This feature is unique to ML systems since more traditional systems usually exhibit deterministic behavior and so probabilities are not often present in their requirements. Examples include: [**RRAV-006**] and [**IC-012**], both are formulated as logical statements with explicit probabilities. To capture such requirements, RE tools must support probabilities.

Confidence Levels or Probabilities about Requirements: In several cases, there was explicit mention of *confidence levels* or of probabilities *about* how often a specific requirement should hold. Such requirements were prevalent in the mission/industrial case studies requirement patterns. Confidence levels may appear inside (e.g., [**IC-002**]) or outside of the requirement text (e.g., [**LR-001**]) through an associated attribute/tag. To support the specification of confidence levels, tags can be used in RE tools to account for both quantitative and qualitative confidence components. Such tags allow us to separate, in some cases, formalization from analysis concerns and may inform the choice of verification method, as outlined in the previous section.

5 Conclusion, Limitations, and Future Work

Threats to Validity: In this paper, we provide a collection of requirements for software that learns. The selection of specific requirements poses a threat to *conclusion validity*, since our efforts were primarily focused on the aerospace domain and it is possible that other domains would reveal additional characteristics. We intend to explore this as future work. It is also possible that our derived patterns are incomplete and examining other use cases will help to identify gaps. We examined a significant number of requirements, both from the literature and other projects, but given that applications of ML software are fast-evolving, it is possible that this sample is not large enough to be representative. This poses a threat to *external validity* of the generalizability of our results. Finally, we do

not explicitly distinguish requirements of the ML component from requirements of systems that may be put in place (e.g. monitors) that maintain oversight and invoke mitigations if the ML system produces erroneous or dangerous results.

Conclusion: As ML becomes more prevalent, it is necessary to create new/modify existing methods to verify systems incorporating ML. This paper focuses on providing and analysing multiple, detailed, real-life requirements for ML components. We observed that such requirements fall into recurring patterns. Table 2 shows 13 reusable patterns, which were derived from 770 mission/industrial requirements. We studied their common characteristics, and provided a classification in terms of their probabilistic nature. We focused primarily on requirement specification, rather than verification, but in practice these requirements would be verified. Exploring the available methods that are capable of expressing probabilistic properties for verification will be necessary as this work progresses.

Our ultimate goal is to extend FRET, but we believe that the presented work is relevant in general for RE tools and approaches. The collection of identified requirement patterns can be integrated as requirement templates in RE tools to guide developers in writing ML requirements, which can be challenging. These patterns, coupled with the key characteristics that we identify, provide a foundation for developing a taxonomy of requirements for software that learns.

Acknowledgements: The authors thank Thomas Pressburger and Irfan Sljivo for requirement examples, insightful feedback and discussions. Thanks also to the anonymous reviewers who provided detailed improvement suggestions. Marie Farrell was supported by a Royal Academy of Engineering Research Fellowship. Anastasia Mavridou and Johann Schumann were supported by NASA contract 80ARC020D0010.

References

1. Standard Specification for Detect and Avoid System Performance Requirements. *ASTM International*, 2020.
2. E. Asaadi, S. Beland, A. Chen, E. Denney, D. Margineantu, M. Moser, G. Pai, J. Paunicka, D. Stuart, and H. Yu. Assured Integration of Machine Learning-based Autonomy on Aviation Platforms. In *Digital Avionics Systems*, pages 1–10. IEEE, 2020.
3. E. Asaadi, E. Denney, J. Menzies, G. J. Pai, and D. Petroff. Dynamic Assurance Cases: A Pathway to Trusted Autonomy. *Computer*, 53(12):35–46, 2020.
4. E. Asaadi, E. Denney, and G. Pai. Quantifying assurance in learning-enabled systems. In *Computer Safety, Reliability, and Security*, pages 270–286. Springer, 2020.
5. H. Belani, M. Vuković, and Željka Car. Requirements Engineering Challenges in Building AI-Based Complex Systems, 2019.
6. D. M. Berry. Requirements engineering for artificial intelligence: What is a requirements specification for an artificial intelligence? In *Requirements Engineering: Foundation for Software Quality*, pages 19–25. Springer, 2022.

7. T. Chuprina, D. Mendez, and K. Wnuk. Towards artefact-based requirements engineering for data-centric systems. In *Workshop on Requirements Engineering for Artificial Intelligence*, volume 2857. CEUR-WS, 2021.
8. European Union Aviation Safety Agency (EASA). EASA Concept Paper: First usable guidance for Level 1 machine learning applications, 2021.
9. D. J. Fremont, J. Chiu, D. D. Margineantu, D. Osipychev, and S. A. Seshia. Formal analysis and redesign of a neural network-based aircraft taxiing system with VerifAI. In *Computer Aided Verification*, pages 122–134. Springer, 2020.
10. E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta. Vision-based road-following using a small autonomous aircraft. In *IEEE Aerospace Conference*, volume 5, pages 3006–3015, 2004.
11. L. Gauerhof, R. Hawkins, C. Picardi, C. Paterson, Y. Hagiwara, and I. Habli. Assuring the safety of machine learning for pedestrian detection at crossings. In *Computer Safety, Reliability, and Security*, pages 197–212. Springer, 2020.
12. D. Giannakopoulou, A. Mavridou, J. Rhein, T. Pressburger, J. Schumann, and N. Shi. Formal requirements elicitation with FRET. 2020.
13. D. Gopinath, G. Katz, C. S. Păsăreanu, and C. Barrett. Deepsafe: A data-driven approach for assessing robustness of neural networks. In *Automated Technology for Verification and Analysis*, pages 3–19. Springer, 2018.
14. R. Hawkins, C. Paterson, C. Picardi, Y. Jia, R. Calinescu, and I. Habli. Guidance on the assurance of machine learning in autonomous systems (AMLAS). *arXiv preprint arXiv:2102.01564*, 2021.
15. H.-M. Heyn, E. Knauss, A. P. Muhammad, O. Eriksson, J. Linder, P. Subbiah, S. K. Pradhan, and S. Tungal. Requirement engineering challenges for ai-intense systems development. In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, pages 89–96. IEEE, 2021.
16. J. Horkoff. Non-functional requirements for machine learning: Challenges and new directions. In *Requirements Engineering*, pages 386–391. IEEE, 2019.
17. F. Ishikawa and N. Yoshioka. How Do Engineers Perceive Difficulties in Engineering of Machine-Learning Systems? - Questionnaire Survey. In *International Workshop on Conducting Empirical Studies in Industry and International Workshop on Software Engineering Research and Industrial Practice*, pages 2–9, 2019.
18. M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher. Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys*, 52(5):1–41, 2019.
19. A. Mavridou, H. Bourbouh, D. Giannakopoulou, T. Pressburger, M. Hejase, P.-L. Garoche, and J. Schumann. The Ten Lockheed Martin Cyber-Physical Challenges: Formalized, Analyzed, and Explained. In *Requirements Engineering*, pages 300–310, 2020.
20. K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, M. Aoyama, L. Joeckel, J. Siebert, and J. Heidrich. Requirements-driven method to determine quality characteristics and measurements for machine learning software and its evaluation. In *Requirements Engineering*, pages 260–270. IEEE, 2020.
21. L. Rierson. *Developing safety-critical software: a practical guide for aviation software and DO-178C compliance*. CRC Press, 2017.
22. R. S. Ross. Guide for Conducting Risk Assessments. Technical report, National Institute of Standards and Technology, Sept. 2012. SP 800-30 Rev. 1.
23. A. Vogelsang and M. Borg. Requirements Engineering for Machine Learning: Perspectives from Data Scientists, 2019.