
CHALMERS
UNIVERSITY OF TECHNOLOGY

# A Framework for Managing Quality Requirements for Machine Learning-Based Software Systems

# A Framework for Managing Quality Requirements for Machine Learning-Based Software Systems

Khan Mohammad Habibullah, Gregory Gay, and Jennifer Horkoff

Chalmers and University of Gothenburg, Sweden
khan.mohammad.habibullah@gu.se, greg@greggay.com, jennifer.horkoff@gu.se

**Abstract.** Systems containing Machine Learning (ML) are becoming common, and the tasks performed by such systems must meet certain quality thresholds, e.g., desired levels of transparency, safety, and trust. Recent research has identified challenges in defining and measuring the achievement of non-functional requirements (NFRs) for ML systems. Managing NFRs is particularly challenging due to the differing nature and definitions of NFRs for ML systems including non-deterministic behavior, the need to scope over different system components (e.g., data, models, and code), and difficulty in establishing new measurements (e.g., measuring explainability). To address these challenges, we propose a framework for identifying, prioritizing, specifying, and measuring attainment of NFRs for ML systems. We present a preliminary evaluation of the framework via an interview study with practitioners. The framework captures a first step towards enabling practitioners to systematically deliver high-quality ML systems.

**Keywords:** Quality Framework · NFR Framework · Non-Functional Requirements · Requirements Engineering · Machine Learning.

## 1 Introduction

Software systems have become increasingly dependent on Machine Learning (ML)[1] to perform complex decision-making and prediction tasks based on large quantities of data [42]. Although the inclusion of ML advances the capabilities of such systems, it also makes the systems and their development more complex—introducing challenges related to the impossibility of exhaustive testing [20], non-determinism and potentially unsafe operations [10], unfair or unexplainable results [9], and unintended bias [28].

To ensure successful development, implementation, and operation, ML systems must satisfy certain quality requirements. In the field of requirements engineering, these quality aspects are known as "non-functional requirements" (NFRs). For ML systems, NFRs are arguably the key requirements, as it is challenging to identify functional requirements over ML components [46]. NFRs for ML systems have recently become a widely investigated topic, with research investigating NFR importance, definition, and measurement [20], scoping over

[1] We will use the term "ML systems" to describe such systems.

different components (e.g., models, data, and code) [19, 20, 37, 41], and research on specific NFR types such as explainability [11], trust [40], or safety [14].

Past research has uncovered challenges related to the difficulty in defining different types of NFRs (e.g., correctness)—including prioritization, uncertainty, domain dependence, requirement interdependence, lack of documentation, regulations, and lack of awareness among practitioners and customers—as well challenges in measuring attainment of NFRs—including lack of established practices or clear baselines, complex ecosystems, testing costs, and bias [19, 20]. As ML systems contain interacting code and non-code components (e.g., models), NFRs may need to be scoped over different aspects of the system [19, 20, 37, 41]. Further, the trade-offs between different NFRs may change with their shifting definitions and are not yet well-understood [3, 19, 20, 25]. Practitioners agree that NFR attainment is a major contributor to the success of ML systems, but they lack knowledge, processes, and tools to help define and measure NFRs for ML systems [3, 20, 41, 45].

Towards addressing these challenges, we propose a *framework* to help practitioners manage NFRs for ML systems. This framework, consisting of five steps, guides NFR specification from a high level—identifying, prioritizing, defining, scoping, and balancing trade-offs between *NFR types* (e.g., accuracy)—to template-based specification of low-level individual *specific NFRs* (e.g., "The lane identification model must have an accuracy of 99.99%").

Our framework could enable practitioners (e.g., ML system developers, architects, product owners, and testers) and researchers to systematically deliver high-quality ML systems by offering step-by-step guidance for defining, prioritizing, scoping, measuring, and specifying NFR types and specific NFRs during the development process. Although such a framework could be useful for NFR management in any type of system, the challenges specific to ML systems indicate that this type of guidance and management is particularly needed, and the framework has been designed with these ML system challenges in mind.

While describing our proposed framework, we use autonomous perception systems—part of driving automation systems (DAS)—as a running example [22]. Autonomous perception systems use ML, trained on complex sensor data, to identify and analyze objects within a vehicle's environment. The framework has undergone a preliminary evaluation through four interviews with five practitioners. This evaluation demonstrates the potential of this framework, and also provides feedback for future revisions.

## 2 Running Example

Driving automation systems (DAS)—including both autonomous driving and advanced driver assistance systems—are systems designed to augment or automate aspects of vehicle operation. Autonomous perception systems (APS)—an important element of the DAS ecosystem—are ML systems that identify and analyze objects in a vehicle's environment based on sensor input. Such systems combine ML models with conventional signal processing [33].

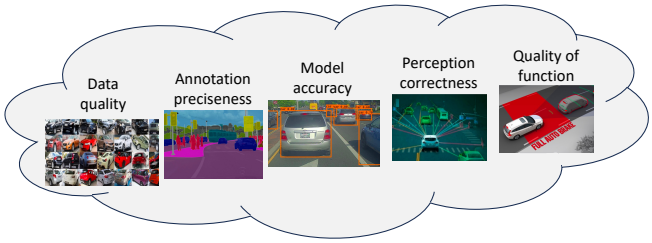


**Fig. 1.** Elements contributing to the quality of an APS, from [22].

The quality of an APS is dependent on—and must be specified at—multiple scopes within the overall system. As illustrated in Figure 1, APS quality is determined by the quality of data collected from the operational design domain, the object detection model, functional code, and other aspects of the system.

Specifying NFRs for an APS is challenging due to the complexity of perception tasks, a dynamic environment where conditions change rapidly, uncertainty and ambiguity inherent in sensor data, real-time data processing, compliance with regulatory standards and guidelines, and ethical considerations [34].

An APS must satisfy certain NFRs (e.g., related to accuracy, safety, and performance). However, it is difficult to determine what NFR types are the most important and how to measure their attainment. Additionally, the meaning of certain NFR types needs to be better understood, and that meaning must be adapted to different scopes, as suggested in Figure 1. For example, a practitioner may need to define broadly what "performance" means for an APS at the system level, as well as what "performance" means in regards to a model or a code component[2]. With this definition, it becomes easier to decide which measurements should be employed at each scope to assess attainment of performance requirements, or how performance should be balanced against other NFR types such as safety. Past research has shown that practitioners working in the APS domain currently lack a structured process for managing NFRs [22]. As such, we employ APS as a representative of ML systems that could benefit from our framework.

## 3 NFR Management Framework

Our proposed NFR management framework consists of five steps—each consisting of one or more concrete tasks—presented in Fig. 2. Although the steps are presented in an ordered sequence, practitioners can jump between steps and tasks, as it is expected that NFR identification, definition, scoping, trade-off, measurement, and specification will be iterative in nature.

We discuss NFR management at two levels. Steps 1–4 of the framework relate to broad high-level *types* of NFRs (e.g., 'performance"). Step 5 of the framework relate to *specific instances* of those types (e.g., "PE003: the object detection

[2] When discussing models, the term "performance" often is a synonym for accuracy, while for code, "performance" generally refers to execution time.

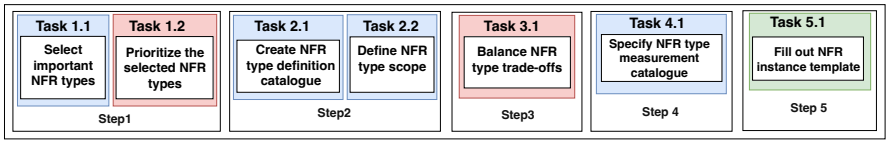


**Fig. 2.** Overview of the framework. Tasks in blue are performed on NFR types, tasks in red compare NFR types, and tasks in green are performed on specific NFR instances.

**Table 1.** Mapping between NFR challenges for ML systems and our framework.

| Challenge | Framework Aspect |
|---|---|
| New NFR types are needed [20, 24, 41] | Task 1.1 |
| NFR types change in priority for systems with or without ML [19, 20] | Task 1.2 |
| NFR type definitions need to be adjusted [20, 41] | Task 2.1 |
| NFRs type definitions need to be scoped over components [19, 20, 41] | Task 2.2 |
| Trade-offs between NFR types are important and unclear [3, 20, 25] | Step 3 |
| Difficulty measuring attainment of different types of NFRs [20] | Step 4 |
| Lack of documentation [20] | Step 5 |
| Lack of management guidance, practices, and knowledge [20, 41] | All |
| Lack of management solutions (e.g., frameworks or tools) [3, 20, 45] | All |

function should detect objects defined in the operational design domain within 0.001 seconds.").

We summarize how the elements of the framework address identified challenges in NFR management in Table 1. For example, "new" NFR types like fairness can be discovered and considered as part of Task 1.1—where important NFR types are identified—with the help of a list of potentially relevant NFRs.

### 3.1 Step 1: Select and Prioritize NFR Types

**Task 1.1: Select Important NFR Types:** Practitioners should first select the subset of NFR types that are potentially important for their system. By selecting important NFR types, practitioners can ensure research allocation (time, effort, and budget) more effectively, mitigate risks and prevent potential issues that could arise later, and make informed design decisions that prioritize aspects of system architecture, infrastructure, and implementation that contribute most significantly to meeting performance objectives. The importance of a particular NFR type may depend on the context or domain of the system under development. As a starting point, a set of important NFR types generally important for ML systems was identified in previous work [20]—a subset are listed in Table 2. This list must, however, be adapted to a specific domain or ML system (e.g., by discarding irrelevant NFR types or adding additional types).

*Running Example:* In applying Task 1.1 for an APS, practitioners may identify NFR types such as accuracy, explainability, performance, reliability, safety, security, and privacy as important (as suggested in recent research [22]). For example, as a safety-critical system, understanding how the APS reaches a decision is critical, and transparency makes it easier to understand how erroneous decisions are made. Real-time performance can also be critical in a vehicle, as this ensures the system reacts swiftly to changing environmental situations. En-

**Table 2.** Example definitions of NFR types relevant for ML systems as identified and defined by Habibullah et al. [19, 21]. Here we include only those NFR types that may be important for an APS system (12/35 identified NFR types).

| **NFRs** | **Definition** |
|---|---|
| Accuracy | The number of correctly predicted data points out of all the data points of ML systems. |
| Explainability | The extent to which the internal mechanics of ML-enabled system can be explained in human terms. |
| Fairness | The ability of a ML system to operate in a fair and unbiased manner. |
| Integrity | The ability of the ML system to ensure that data is real, accurate and safeguarded from unauthorised modification. |
| Interpretability | The extraction of relevant knowledge from a model concerning relationships in the data or learned by the model. |
| Performance | The ability of a ML system to perform actions within defined time or throughput bounds. |
| Privacy | An ML model or ML system is private if an observer examining the output is not able to determine whether a specific information was used in the computation. |
| Reliability | The probability of the ML system performing without failure for a specific number of uses or amount of time. |
| Safety | The absence of failures or conditions that render a system dangerous. |
| Security | Security measures ensure a ML system's safety against malicious use or sabotage. |
| Transparency | The extent to which a human user can infer why the ML system made a particular decision or produced a particular externally-visible behaviour. |
| Usability | How effectively users can learn and use an ML system. |
| ... | .... |

suring security and privacy ensures that the APS is guarded against adversarial attacks, data manipulation, or data leaks.

**Task 1.2: Prioritize the Selected NFR Types:** The initial list of selected NFR types may be too large to practically manage. It is very difficult to prioritize NFR types at a general level, as significance may vary from system to system. Therefore, a practitioner may want to prioritize the NFR types based on the context of the ML system to be developed. We first recommend that practitioners rank the identified NFR types (e.g., very important, important, less important), then use that ranking to determine where to prioritize efforts. Practitioners can prioritize in an ad-hoc way, discussing among themselves and coming to understand the meaning and purpose of the NFR types, or they can use one or more established RE prioritization techniques (e.g., [2]).

*Running Example:* For an APS, accuracy is likely to be given the highest priority because it is essential for the system to function correctly. Safety is also likely to get high priority, as there is a risk of harm to drivers when hazards occur. The practitioners might also prioritize performance, as the system needs to make quick decisions to avoid obstacles or accidents.

### 3.2 Step 2: Define NFR Types and Identify NFR Type Scope

**Task 2.1: Create NFR Type Definition Catalogue:** After identifying important NFR types for the ML system to be developed, we recommend that practitioners create a definition catalogue by adapting the general definitions of these NFRs (e.g., from Table. 2) to their particular system. The adjustment is needed as the definition of an NFR type may vary depending on context. The resulting catalogue helps to establish a clear and shared understanding of the NFRs among the stakeholders working on a particular system, facilitate the identification and management of potential quality issues, and enable more effective decision-making and risk management [43, 47].

*Running Example:* Example definitions for an APS are presented in Table 3.

**Table 3.** Definitions of important NFR Types for autonomous perception system.

| NFRs | Definition |
|---|---|
| Accuracy | The ability to precisely and correctly identify objects defined in the Operational Design Domain. |
| Reliability | The system consistently identify objects correctly and without sudden errors, ensuring the system can react appropriately to the environment. |
| Safety | Objects are accurately identified and located—preventing collisions, accidents, and other potential hazards—even in the presence of poor visibility, occlusions, and unexpected object behaviors. |

**Table 4.** Examples of NFR scoping for an APS.

| | System Element the NFR Type Can be Defined Over | | | | |
|---|---|---|---|---|---|
| **NFR** | **Training Data** | **ML Pipeline** | **ML Model** | **Results** | **Whole System** |
| Accuracy | ✖ | ✔ | ✔ | ✔ | ✔ |
| Explainability | ✖ | ✔ | ✔ | ✔ | ✔ |
| Integrity | ✔ | ✔ | ✔ | ✔ | ✔ |
| Performance | ✔ | ✔ | ✔ | ✔ | ✔ |
| Reliability | ✔ | ✔ | ✔ | ✔ | ✔ |
| Safety | ✔ | ✔ | ✔ | ✔ | ✔ |

**Task 2.2: Scope NFR Types over Different Elements of ML Systems:** A ML system is typically not a monolith, but different types of components working together to deliver functionality [37]. Some components relate directly to ML, while others do not. Therefore, the definitions of NFR types may differ between components or groups of components. Past research suggests one form of scoping—that NFRs could be specified over datasets, models, ML pipelines used to train models, results of executing those models, and the system code [19].

Scoping is important, as the definition, importance, and measurement of NFR types may differ depending on the components considered. For example, "usability" of a ML pipeline refers to how difficult it is to use that pipeline to train a model, "usability" of a model refers to how difficult it is to extract decisions from the model, and "usability" of results refers to how effectively users can understand and apply the model output for practical purpose.

We envision Tasks 2.1 and 2.2 to be highly iterative. Practitioners need to understand how (and which) NFRs apply to different scopes and may need to refine NFR type definitions at each scopes.

*Running Example:* An example of a possible scoping of NFR types for an APS is presented in Table 4. For example, when specifying accuracy, one might have requirements on the accuracy of a particular model (e.g., ability to detect particular types of objects). However, one might also specify NFRs on the meta-accuracy of the ML pipeline—that is, *any* model produced by the pipeline must meet a minimal level of accuracy. The system may have further requirements on the accuracy of broader functions that make use of these models.

Similarly, reliability at the system level refers to the ability of the code to operate correctly and without crashing, while at the training data level, this may be reinterpreted to limit the amount of corrupt or noisy data. In different scopes, reliability may refer to the ML pipeline's ability to produce models or the model's ability to produce a prediction on-demand without error. The other rows of Table 4 have similar explanations.

**Table 5.** Examples of NFR type trade-offs for APS Systems. The cited papers offer more discussion on the potential trade-offs.

| Conflicting NFR Types | APS Example |
|---|---|
| Accuracy, Interpretability [7] | Some model architectures, e.g., deep convolutional neural networks, provide high accuracy for image recognition, but provide limited insight into how predictions are reached. |
| Accuracy, Privacy [1] | Certain data (e.g., GPS or visual data) could enhance the accuracy of a model's predictions. However, this data is also potentially sensitive—violating the privacy of users or individuals in the APS' environment. Protecting privacy may require masking or removing sensitive information from the training data, which could negatively impact the model's accuracy. |
| Safety, Performance [31] | Ensuring APS safety may require adding additional safeguards or constraints that could negatively impact the performance. For example, an ensemble of models may collectively produce safer verdicts than a single model, but this requires asking all models for predictions and then calculating a combined verdict. |

### 3.3 Step 3: Balance NFR Type Trade-offs

Practitioners may observe trade-offs among different NFR types—e.g., performance, usability, and security often conflict. It is also important to note that the nature and criticality of these trade-offs may differ based on the scope within the system, or the context and domain of the system under development. For example, trade-offs between accuracy and interpretability at the ML model level may have a different weight and impact than trade-offs between the two at the result-level.

In cases where a trade-off exists, practitioners need to find a balance between the conflicting NFR types. Trade-offs can be considered in an ad-hoc way, or via more structured methods (e.g., [36]). The emphasis here is to make such trade-offs explicit and raise them for discussion and eventual resolution.

*Running Example:* Some examples of trade-offs among different NFR types for an APS are presented in Table 5. An appropriate balance must be found, potentially guided by standards like ISO 26262 in the automotive industry [26].

### 3.4 Step 4: Specify NFR Measurement Catalogue

At this stage, we suggest practitioners adapt a measurement catalogue for the selected NFR types. In future work, we aim to provide an example NFR measurement catalogue for the NFR types identified in our previous study [23]. A partial excerpt of such a catalogue is shown in the first three columns of Table 6. For each NFR type, the catalogue should record all measurements—including equations—that will be used to assess the satisfaction of each of the specific NFRs of each type at each scope in the system. Considering the scope of NFR measurement is important (see Task 2.2), as our prior work found that definition and measurement of NFR types varies depending on the scope [18].

**Table 6.** Example excerpt from a NFR measurement catalogue for an APS.

| NFR Type | Measurement | Equation | APS Example Application |
|---|---|---|---|
| Accuracy | F-Score | $\frac{2*(Precision*Recall)}{(Precision+Recall)}$ | The F-Score is the harmonic mean of a system's precision and recall, and is commonly used to assess the accuracy of an APS (e.g., when detecting pedestrians). |
| Reliability | Mean Time Between Failures (MTBF) | $\frac{Uptime}{Num.Failures}$ | MTBF can be used to define how often an APS is allowed to fail in practice. If MTBF rises above the defined threshold, improvements must be made. |

Having a clear and comprehensive measurement catalogue is crucial, as it provides a systematic method to identify how to assess the attainment of each NFR. It enables identification of targets to show that an NFR is attained. It also enables detection of measurement challenges early in the development process, provides a means to group related NFRs, and enables proactive management and monitoring of NFR attainment and overall system quality.

*Running Example:* Table 6 illustrates an excerpt from an NFR measurement catalogue for an APS.

### 3.5 Step 5: Specify NFRs Using Template

In the final step, practitioners fill out a template as part of the process of specifying the low-level individual NFRs of the types selected and defined in the previous steps. Each NFR type will have many specific NFRs that must be fulfilled to ensure the overall quality is upheld. It is important that NFRs have documentation that is stable, unambiguous, and complete, so that stakeholders can form a shared understanding of the NFR and ensure that the NFR is satisfied. A template can help ensure the attainment of this goal, facilitating discussion, reducing development time, cost, and effort, and enabling verification [16].

Past research has found that the practitioners need to capture a context-aware definition of the NFR, importance (and justification for the importance), system scope, measurement method, measurement target, and trade-offs with other specific NFRs. We capture these elements in a template—presented in Table 7—inspired by existing templates such as the one offered by Volere [35].

**Table 7.** Template for specifying individual NFRs.

| |
|---|
| **Requirement ID:** A unique id. |
| **Requirement Type:** NFR Type (e.g., performance) |
| **Definition:** A context-adapted definition of the specific NFR type. |
| **Description:** The primary text describing the requirement. |
| **Rationale:** A justification of the requirement. |
| **Importance Level:** The overall importance of this specific NFR for the system (e.g., very important, important, not important). |
| **System Scope:** System elements the specified NFR is relevant to. |
| **Measurement:** The measurement used to assess attainment of this NFR. |
| **Measurement Target:** The measurement threshold required to satisfy this NFR. |
| **Trade-Offs:** IDs of other specific NFRs that conflict with this specific NFR, and explanation of the conflict. |
| **Source:** Person, regulation, document, etc. that suggested the requirement. |
| **Responsible:** Person responsible for managing this requirement. |

**Table 8.** An example of a specific accuracy NFR for an APS.

| |
|---|
| **Requirement ID:** AC001 |
| **Requirement Type:** Accuracy |
| **Definition:** Accuracy for object detection in an autonomous perception system refers to ability to precisely and correctly identify objects defined in the Operational Design Domain. |
| **Description:** The lane identification model shall detect lane divisions on all road types with high accuracy. |
| **Rationale:** If the perception system is not accurate, it may make mistakes or incorrect decisions that could have serious consequences (e.g., accidents). |
| **Importance Level:** Very Important |
| **System Scope:** ML model |
| **Measurement:** F-Score |
| **Measurement Target:** 99.99999% |
| **Trade-Offs:** To protect passenger privacy, the system employ object detection models that blur or anonymize faces and license plates, sacrificing some detection accuracy (requirement PR013). |
| **Source:** Tom Robertsson (product owner) |
| **Responsible:** Tom Robertsson (product owner) |

*Running Example:* Table 8 offers an example of a specific accuracy NFR for the lane detection model that is part of the overall object detection functionality of an APS. In reality, many similar accuracy requirements would need to be captured for different aspects of the system functionality and for different scopes.

## 4 Preliminary Evaluation

### 4.1 Methodology

To gain initial feedback on the framework, we conducted a qualitative interview study. We chose interviewees who had experience with NFRs and were currently working with ML in industry.

**Sampling:** The sampling method was a mix of convenience and purposive sampling. We contacted people in the industry who have requirements engineering and ML experience, and then asked them if they knew any qualified candidates we could contact.

**Data Collection:** We interviewed five practitioners over four interviews. These practitioners have 6–25 years of experience working with RE and ML. Among the five, three are from the automotive domain, one from the telecommunications and networking domain, and another from the smart home monitoring domain. Table. 9 presents demographic information of the interviewees. Although the numbers of interviewees is small, this is a preliminary evaluation of our proposed framework. We believe that the background and experience of the interviewees will be valuable for refining the framework before conducting a more thorough evaluation.

The interviews lasted approximately 60 minutes, and were conducted between September and October 2023 via Microsoft Teams. We recorded interviews with the permission of the interviewees, then transcribed and anonymized for further analysis. During this interview, we first presented the framework steps, then asked about the utility of the overall framework, as well as each step. We asked if any elements should be removed, added, or adjusted. The interviews were semi-structured, with the ability to ask follow-up questions[3].

**Data Analysis:** The collected data was qualitative in nature. We applied a mixed form of thematic analysis, where we started with high-level codes based on our interview questions, then also extracted and refined additional inductive codes from the transcripts. Codes were then grouped into themes and subthemes. The first author did the initial coding, then all codes were checked and discussed by all authors. Based on this thematic analysis, we extracted impressions and insights about the framework, as well as suggestions for future improvements to the framework.

[3] The interview guide and presentation to the interviewees can be found at `https://doi.org/10.7910/DVN/QUPD8D`. Due to the sensitive nature of the collected practitioner data from the interviews, we can not share the full interview transcripts.

**Table 9.** Interviewees' Demographic Information

| ID | Field of Work | Role | Exp. (Years) | ML Exp. | NFR Exp. | Responsibilities |
|---|---|---|---|---|---|---|
| I1 | Smart home monitoring | Software developer | 6 | Yes | Yes | Development and data analysis |
| I2 | Automotive | Researcher and senior system safety engineer | 8 | Yes | Yes | Requirements specifications, safety functions analysis, system design |
| I3 | Automotive | Researcher and chief safety engineer | 25 | Yes | Yes | Responsible for safety compliance, Fulfill stakeholder requirements by ensuring ISO 26262 standard, SOTIF standard, and cybersecurity and quality standards |
| I4 | Automotive | Researcher and requirements engineer | 14 | Yes | Yes | Work with system level requirements, ensure safety compliance |
| I5 | Telecommunications | Software engineer and data analyst | 8 | Yes | Yes | Anomaly detection, data analysis, software development |

### 4.2 Impressions and Insights

Overall, the feedback was positive, with all interviewees suggesting the need for such a framework. The interviewees did not suggest that any steps or tasks be removed. However, they did make several observations about the individual steps and tasks, as well as the framework as a whole. We indicate the number of interviewees supporting a particular point in parentheses, e.g., (2).

Interviewees had feedback on when the framework would be useful. Two interviewees (2) stated that our framework was potentially very useful at early stages of development—particularly tasks centered around NFR types. Furthermore, interviewees (2) suggested that the framework would be very useful for a new system or when performing a large overhaul, but that it may be less useful as part of continuous evolution.

Interviewees (2) pointed out that the framework could be used to identify important NFR types for an ML system, especially having been provided a pre-existing list of NFRs that are commonly important for ML systems. In addition, interviewees noted that creating a definition catalog could facilitate understanding of the contextual meaning and purpose of an NFR type (2), and that having a definition catalogue could improve shared understanding and facilitate discussion among stakeholders (3).

In terms of prioritization, interviewees (2) pointed out that this step should be iterative and repeated at different times and with different degrees of thoroughness. Scoping was identified as helping in prioritizing as well as identifying and discussing trade-offs (2).

Interviewees (3) had comments on measurements, pointing out that some NFR types may not be critical enough or too difficult to measure. They also noted that some measurements and targets could be extracted from standards. Finally, interviewees (2) suggested that the template was useful as a means of documenting specific NFRs. However, they also noted that it would be too much work to fill out the template for all NFRs.

### 4.3 Potential Future Improvements

Interviewees also pointed out many ways in which the framework can be improved. Interviewees (3) suggested that the framework should also account for stakeholder roles, it should document stakeholder roles more explicitly and at many steps as possible to facilitate discussion and negotiation.

Other interviews made suggestions concerning support for NFR breakdown and traceability. One interviewee stated that the framework should consider requirements breakdown and layering more explicitly (e.g., logical level, system level, feature level), possibly as part of—or in addition to—scoping. Interviewees (2) added that the framework should enable explicit traceability between NFR types and specific NFRs to ensure decisions made at the type level are reflected at the instance level.

In terms of process, interviewees (2) stressed that the framework must fit into their process (e.g., it may need to be adapted to work within a specific agile process). In addition, an interviewee mentioned that the framework should keep historical records of decisions (e.g., trade-offs and prioritization).

Interviewees (3) expressed that it is difficult and time consuming to fill out the template for every requirement, and we should identify when this type of documentation is needed. We should explore ways to make filling out the template easier, like identifying optional fields, populating the framework with default values, and providing examples to guide and ease application. Interviewees (2) suggested the inclusion of a default definition catalogue (including defaults for different scopes). As some scopes may not be applied to some systems, the framework must be flexible enough to function without fully scoping all NFR types. Another interviewee said that the framework could provide examples of trade-offs among NFR types. Regarding default values and measurements, interviewees (2) suggested that the framework include an extensive measurement catalogue and guidance on the most common measurements and measurement targets. Further interviewees (3) pointed out that standards (e.g., ISO 26262 [26]) could provide template defaults and guidance for certain domains, and one interviewee suggested that earlier steps regarding NFR types could populate generic default templates for those types.

Our future work includes framework tooling. An interviewee mentioned that a tool based on our framework could offer suggestions to the users of potentially incomplete or missed aspects (e.g., trade-offs, measurements, template fields), providing a type of quality or completeness check. In addition, we will explore how our proposed framework will facilitate compliance with the EU AI Act [13].

Finally, currently, we have focused on trade-offs amongst NFR types, but one interviewee rightly points out that the framework should also be able to identify potential trade-offs at the instance level and inform responsible stakeholders. We will enhance our proposed framework by incorporating the suggested improvements from the interviewees, including a detailed breakdown of NFRs, better traceability between NFR types and specific NFRs, managing historical records of decisions, flexible scoping options, and easier implementation processes.

### 4.4 Threats to Validity

**External validity:** We used a mixture of purposive and convenience sampling as the interviews required a certain set of expertise. The number of interview participants may impact the reliability of the evaluation. In addition, three out of five interviewees were from the automotive domain. However, we selected

interviewees based on deep knowledge of both NFRs and ML, and their suggestions are applicable to other domains. Importantly, the goal of this evaluation is simply to gain preliminary feedback. We plan to conduct a comprehensive assessment in the future, after incorporating some of the recommendations into the framework. Although the focus of the framework has been on ML systems, elements of the framework could be generally useful to manage NFRs for all types of systems. However, by considering ML-specific definitions, scopes, and measurements, we focus this work on the particular challenges of NFR handling in ML systems.

**Internal validity:** In addition, while the evaluation analysis relies on thematic coding, potential validity threats due to the need for interpretation were addressed by having all authors review and discuss the codes.

## 5 Related Work

In this section, we summarize related work on NFRs, and compare the framework presented in this work to other NFR management frameworks.

**NFRs:** NFRs play a critical and important role in ensuring system quality [30]. However, consistent and clear approaches to eliciting, documenting, and validating NFRs are not fully established [15]. There is a lack of agreement among researchers on when and how NFRs should be considered and integrated into the overall RE process [4,12]. Stakeholders are sometimes unaware of NFRs, because of lacking background knowledge of the domain [17].

**Scope of NFRs over ML systems:** ML aspects of a system are part of a larger software system, which may include components for configuration, data collection, feature extraction, analysis tools, monitoring, and glue code [37]. In addition, an interview study with four data scientists noted the increasing complexity of ML-based development processes, which require the effective use of large data sets and a reliance on NFRs and suggested that requirements engineering for ML should focus on requirements over data along with requirements for the system [45]. As part of a systematic mapping study, Habibullah et al. identified clusters of NFRs for ML systems based on shared characteristics and scoped NFRs over datasets, ML pipelines, models, results, and system code [19].

**Towards Solutions for Managing NFRs for ML Systems:** Recent research has reported several aspects that are required to address to manage NFRs for ML systems. The importance of identifying and extracting relevant NFRs for ML systems is reported in a survey study [39]. With an aim at constructing a quality model for ML systems, Siebert et al. emphasized the need for NFR identification, scoping, and measurement [41]. Two different systematic mapping studies pointed out the need for further research and necessity of RE techniques for managing NFRs for ML systems [3, 44].

Towards establishing solutions for managing NFRs, standards define necessary quality properties of the software and its components. For example, ISO/IEC 25010 defines quality models for software and systems [27]. However,

**Table 10.** Comparison of select NFR frameworks to our proposed framework.

| Framework | Focus | Identification | Prioritization | Definition Catalogue | Scope | Trade-Offs | Measurement Catalogue | Template and/or Documentation |
|---|---|---|---|---|---|---|---|---|
| [12] | Non-ML | ✔ | ✔ | ✖ | ✖ | ✔ | ✖ | ✖ |
| [38] | Non-ML | ✔ | ✖ | ✖ | ✖ | ✔ | ✖ | ✔ |
| [41] | ML | ✔ | ✖ | ✖ | ✔ | ✖ | ✔ | ✔ |
| [29] | ML | ✔ | ✔ | ✔ | ✖ | ✖ | ✖ | ✖ |
| [6] | ML | ✔ | ✖ | ✖ | ✖ | ✖ | ✖ | ✔ |
| Our Framework | ML | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

these quality models cannot be applied directly to ML systems due to the challenges identified previously (e.g., definition, scoping, and measurement differences from traditional systems) [41]. While several frameworks exists that focus on different aspects of RE for ML systems (e.g., Holistic DevOps [8], Ethics-Aware SE for analyzing ethical requirements [5], and RE4ML for requirements elicitation for ML systems [32]), few comprehensive frameworks exists for managing NFRs for ML systems.

We compare our framework with other NFR-related frameworks proposed from the literature in Table 10, where ✔ indicates that the framework provides capabilities or guidelines for managing the aspect specified in that column. A ✖ indicates that the study either did not discuss the activity or mentioned it only in passing, and the framework does not provide specific guidance on that aspect.

NFR frameworks have been proposed for traditional systems, including one centered around acquiring domain knowledge, identifying and decomposing NFRs, identifying operationalizations for meeting NFRs in the system, dealing with ambiguities, trade-offs, priorities, and inter-dependencies among NFRs and operationalizations, supporting decisions with rationale, and evaluating decision impact [12]. A framework has also been proposed for managing NFRs in agile development, supporting elicitation, analysis, documentation, and validation [38].

For ML systems, a quality model construction process has been proposed that includes defining a quality meta-model, defining use case and application context, identifying relevant ML quality requirements, identifying relevant entities of an ML system, identifying reference elements of an ML quality model, and instantiating the quality model for the use cases [41]. An approach has also been proposed to build quality models that focuses on definition and ordering of "characteristics" (NFRs) for AI systems [29]. In addition, templates exist for documenting NFRs for ML systems [6].

Although these frameworks overlap with ours in certain aspects, illustrated in Table 10, to our knowledge, our framework is the first to cover a wide range of steps and tasks in the management of NFRs for ML systems, addressing many of the ML system challenges identified in past studies (e.g., [3, 20, 22, 45]).

## 6 Conclusion

We have proposed a framework for managing NFRs for ML systems. This framework can be used—starting from the early stages of a project—to define, scope, and prioritize the NFR types that are important to the system-under-development. It can be used to identify measurements to ensure specific NFRs of those types are satisfied to the necessary degree. Finally, the framework offers templates to specify and document individual NFRs of those types.

We conducted semi-structured interviews to offer feedback on our proposed framework. All of the participants stated that our proposed framework could be useful in their practice, and suggested how it could improve the quality of the systems they develop. However, the interviewees also recommended potential improvements. The current framework can be considered as the first iteration of a broader design science process. In future work, we will incorporate suggestions from the interviewees (e.g., explicit representation of stakeholders, default definitions and measurements, and clarifying when the template should be used).

**Acknowledgment:** This research is supported by a Swedish Research Council (VR) Project: Non-Functional Requirements for Machine Learning: Facilitating Continuous Quality Awareness (iNFoRM).

## References

1. Abbasi, W., Mori, P., Saracino, A., Frascolla, V.: Privacy vs accuracy trade-off in privacy aware face recognition in smart systems. In: 2022 IEEE Symposium on Computers and Communications (ISCC). pp. 1–8. IEEE (2022)
2. Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M.N.: A systematic literature review of software requirements prioritization research. Information and software technology **56**(6), 568–585 (2014)
3. Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., Grundy, J.: Requirements engineering for artificial intelligence systems: A systematic mapping study. Information and Software Technology **158**, 107176 (2023)
4. Ameller, D., Franch, X., Gómez, C., Martínez-Fernández, S., Araújo, J., Biffl, S., Cabot, J., Cortellessa, V., Méndez, D., Moreira, A., et al.: Dealing with non-functional requirements in model-driven development: A survey. IEEE Transactions on Software Engineering (2019)
5. Aydemir, F.B., Dalpiaz, F.: A roadmap for ethics-aware software engineering. In: Proceedings of the International Workshop on Software Fairness. pp. 15–21 (2018)
6. Bajraktari, E., Krause, T., Kücherer, C.: Documentation of non-functional requirements for systems with machine learning components. In: 2024 REFSQ Workshops
7. Baryannis, G., Dani, S., Antoniou, G.: Predicting supply chain risks using machine learning: The trade-off between performance and interpretability. Future Generation Computer Systems **101**, 993–1004 (2019)
8. Bosch, J., Olsson, H.H., Crnkovic, I.: It takes three to tango: Requirement, outcome/data, and ai driven development. In: SiBW 2018, Software-intensive Business: Start-ups, Ecosystems and Platforms, Espoo, Finland, December 3, 2018. pp. 177–192. CEUR-WS. org (2018)
9. Burkart, N., Huber, M.F.: A survey on the explainability of supervised machine learning. Journal of Artificial Intelligence Research **70**, 245–317 (2021)
10. Challen, R., Denny, J., Pitt, M., Gompels, L., Edwards, T., Tsaneva-Atanasova, K.: Artificial intelligence, bias and clinical safety. BMJ Quality & Safety **28**(3), 231–237 (2019)
11. Chazette, L., Brunotte, W., Speith, T.: Explainable software systems: from requirements analysis to system evaluation. Requirements Engineering pp. 1–31 (2022)
12. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering, vol. 5. Springer Science & Business Media (2012)

13. European Commission, Directorate-General for Communications Networks, Content and Technology: EUR-Lex - 52021PC0206 - EN - EUR-Lex. CNECT (2021), https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206
14. Faria, J.M.: Machine learning safety: An overview. In: Proceedings of the 26th Safety-Critical Systems Symposium, York, UK. pp. 6–8 (2018)
15. Glinz, M.: On non-functional requirements. In: 15th IEEE International requirements engineering Conference (RE 2007). pp. 21–26. IEEE (2007)
16. Gondal, M.A., Qureshi, N.A., Mukhtar, H., Ahmed, H.F.: An engineering approach to integrate non-functional requirements (nfr) to achieve high quality software process. In: ICEIS (2). pp. 377–384 (2020)
17. Gruber, K., Huemer, J., Zimmermann, A., Maschotta, R.: Integrated description of functional and non-functional requirements for automotive systems design using sysml. In: 2017 7th IEEE International Conference on System Engineering and Technology (ICSET). pp. 27–31. IEEE (2017)
18. Habibullah, K.M., Diaz, J.G., Gay, G., Horkoff, J.: Scoping of non-functional requirements for machine learning systems. In: 2024 IEEE 32nd Int. Requirements Engineering Conference (RE). IEEE (2024)
19. Habibullah, K.M., Gay, G., Horkoff, J.: Non-functional requirements for machine learning: An exploration of system scope and interest. SE4RAI workshop, 44th International Conference on Software Engineering (ICSE 2022) (2022)
20. Habibullah, K.M., Gay, G., Horkoff, J.: Non-functional requirements for machine learning: Understanding current use and challenges among practitioners. Requirements Engineering **28**(2), 283–316 (2023)
21. Habibullah, K.M., Heyn, H.M., Gay, G., Horkoff, J., Knauss, E., Borg, M., Knauss, A., Sivencrona, H., Li, J.: Requirements engineering for automotive perception systems: An interview study. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 189–205. Springer (2023)
22. Habibullah, K.M., Heyn, H.M., Gay, G., Horkoff, J., Knauss, E., Borg, M., Knauss, A., Sivencrona, H., Li, P.J.: Requirements and software engineering for automotive perception systems: an interview study. Requirements Engineering pp. 1–24 (2024)
23. Habibullah, K.M., Horkoff, J.: Non-functional requirements for machine learning: understanding current use and challenges in industry. In: 2021 IEEE 29th International Requirements Engineering Conference (RE). pp. 13–23. IEEE (2021)
24. Horkoff, J.: Non-functional requirements for machine learning: Challenges and new directions. In: 2019 IEEE 27th International Requirements Engineering Conference (RE). pp. 386–391. IEEE (2019)
25. Hort, M.: Investigating Trade-offs For Fair Machine Learning Systems. Ph.D. thesis, UCL (University College London) (2023)
26. ISO: Road vehicles – Functional safety (2011)
27. ISO/IEC25010: Systems and software engineering systems and software quality requirements and evaluation (square) (2023)
28. Kamishima, T., Akaho, S., Sakuma, J.: Fairness-aware learning through regularization approach. In: 2011 IEEE 11th International Conference on Data Mining Workshops. pp. 643–650. IEEE (2011)
29. Kharchenko, V., Fesenko, H., Illiashenko, O.: Quality models for artificial intelligence systems: characteristic-based approach, development and application. Sensors **22**(13), 4865 (2022)
30. Kopczyńska, S., Ochodek, M., Nawrocki, J.: On importance of non-functional requirements in agile software projects—a survey. In: Integrating Research and Practice in Software Engineering, pp. 145–158. Springer (2020)

31. Mairiza, D., Zowghi, D.: Constructing a catalogue of conflicts among non-functional requirements. In: Evaluation of Novel Approaches to Software Engineering: 5th International Conference, ENASE 2010, Athens, Greece, July 22-24, 2010, Revised Selected Papers 5. pp. 31–44. Springer (2011)
32. Nalchigar, S., Yu, E., Keshavjee, K.: Modeling machine learning requirements from three perspectives: a case report from the healthcare domain. Requirements Engineering **26**, 237–254 (2021)
33. Pandharipande, A., Cheng, C.H., Dauwels, J., Gurbuz, S.Z., Ibanez-Guzman, J., Li, G., Piazzoni, A., Wang, P., Santra, A.: Sensing and machine learning for automotive perception: A review. IEEE Sensors Journal **23**(11), 11097–11115 (2023)
34. Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G.P., Cho, W.: A review on autonomous vehicles: Progress, methods and challenges. Electronics **11**(14), 2162 (2022)
35. Robertson, S., Robertson, J.: Mastering the requirements process: Getting requirements right. Addison-wesley (2012)
36. Ruhe, G., Eberlein, A., Pfahl, D.: Trade-off analysis for requirements selection. International Journal of Software Engineering and Knowledge Engineering **13**(04), 345–366 (2003)
37. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F., Dennison, D.: Hidden technical debt in machine learning systems. Advances in neural information processing systems **28** (2015)
38. Sherif, E., Helmy, W., Galal-Edeen, G.H.: Proposed framework to manage non-functional requirements in agile. IEEE access (2023)
39. Shreda, Q.A., Hanani, A.A.: Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches. IEEE Access (2021)
40. Siau, K., Wang, W.: Building trust in artificial intelligence, machine learning, and robotics. Cutter business technology journal **31**(2), 47–53 (2018)
41. Siebert, J., Joeckel, L., Heidrich, J., Trendowicz, A., Nakamichi, K., Ohashi, K., Namba, I., Yamamoto, R., Aoyama, M.: Construction of a quality model for machine learning systems. Software Quality Journal **30**(2), 307–335 (2022)
42. Smola, A., Vishwanathan, S.: Introduction to machine learning. Cambridge University, UK **32**(34), 2008 (2008)
43. Supakkul, S., Hill, T., Chung, L., Tun, T.T., do Prado Leite, J.C.S.: An nfr pattern approach to dealing with nfrs. In: 2010 18th IEEE International Requirements Engineering Conference. pp. 179–188. IEEE (2010)
44. Villamizar, H., Escovedo, T., Kalinowski, M.: Requirements engineering for machine learning: A systematic mapping study. In: 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 29–36. IEEE (2021)
45. Vogelsang, A., Borg, M.: Requirements engineering for machine learning: Perspectives from data scientists. In: IEEE 27th International Requirements Engineering Conference Workshops (REW). pp. 245–251. IEEE (2019)
46. Wan, Z., Xia, X., Lo, D., Murphy, G.C.: How does machine learning change software development practices? IEEE Transactions on Software Engineering **47**(9), 1857–1871 (2019)
47. Werner, C., Li, Z.S., Lowlind, D., Elazhary, O., Ernst, N.A., Damian, D.: Continuously managing nfrs: Opportunities and challenges in practice. IEEE Transactions on Software Engineering (2021)