Quality trade-offs in ML-enabled systems: a multiple-case study

Vladislav Indykov
indykov@chalmers.se
Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden

Rebekka Wohlrab
wohlrab@chalmers.se
Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden
Carnegie Mellon University
Pittsburgh, USA

Daniel Strüber
danstru@chalmers.se
Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden
Radboud University
Nijmegen, Netherlands

Abstract

When building a machine-learning-enabled system, quality objectives are achieved through architectural and non-architectural tactics, including general ones as well as specific ones that address machine learning specifics, such as the focus on data. However, implementing these tactics typically compromises other quality attributes that are not the primary focus of the tactic at hand. Previous research has investigated quality aspects and tactics for machine-learning-enabled systems, but there is a lack of detailed insights on quality trade-offs observed in industrial practice, and how companies address them. A study in this direction could especially help start-ups and SMEs to benefit from the insights of other companies, and academics to develop improved tactics addressing these trade-offs in alternative, potentially more effective ways.

In this paper, to fill this gap, we present a multiple-case study of four companies in the AI sphere. As AI solution providers, all companies are faced with a variety of quality priorities, tactics, and trade-offs in their addressed application domains. We find that our subject companies consistently address a common set of core quality priorities, encompassing *reliability*, *functional suitability*, and *resource efficiency*, which they address with recurring architectural tactics such as the use of *cloud-based components* for resource efficiency, and non-architectural ones such as *Scrum practices* for functionality suitability. Finally, we find a variety of trade-offs appearing in different companies with several recurring ones, two of them–*efficiency vs. reliability*, and *system accuracy vs. explainability*–manifesting themselves in three out of the four companies.

CCS Concepts

• **Software and its engineering → Software architectures**; **Software design tradeoffs**; • **Computing methodologies → Artificial intelligence**.

Keywords

Software Quality, Machine Learning, Software Architectures

ACM Reference Format:
Vladislav Indykov, Rebekka Wohlrab, and Daniel Strüber. 2025. Quality trade-offs in ML-enabled systems: a multiple-case study. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3672608.3707754

SAC '25, March 31-April 4, 2025, Catania, Italy

ACM ISBN 979-8-4007-0629-5/25/03
https://doi.org/10.1145/3672608.3707754

1 Introduction

As machine learning (ML) becomes increasingly integrated into software systems across various industries, the demand for high-quality ML-based solutions continues to grow [25], [26]. However, the concept of quality is not universal; it varies significantly depending on the specific business goals. It depends on the domain of companies that own and use AI-based systems and on the strategic objectives of companies that deliver such systems. While the use of ML-enabled software in different domains is usually associated with a spectrum of business-oriented tasks, the development, integration, and maintenance of such systems create several complex challenges and dilemmas from a technical perspective.

Different quality priorities give rise to different approaches to achieving them by implementing architectural and non-architectural tactics [4], [10], [21]. Architectural tactics may involve decisions around system structure and component interaction, while non-architectural tactics can include workflow organization and process optimizations. Despite the efforts of companies to implement them most effectively, they usually encounter unavoidable trade-offs. Improving one quality attribute may necessitate compromises in another. For instance, developing complex models (i.e. deep learning models) usually results in high system accuracy, but also excessive computational resource consumption.

These trade-offs generate significant dilemmas for companies aiming to balance multiple, sometimes conflicting, quality objectives. As a result, there is no universal approach to developing high-quality ML-enabled systems and the consideration of context is required. This complexity sets a high entry threshold for ML and software engineering startups and small and medium-sized enterprises (SMEs) in the early stages, that are in dire need of best practices from larger successful players [20]. However, there is a lack of generally reported practical experience in quality-driven development of ML-based software, in particular, considering the inseparable connection between priorities (quality attributes), decisions to achieve them (tactics), and consequences of their implementation (trade-offs).

In this paper, to fill this gap, we present the results of a multiple-case study with four companies that deliver AI-based solutions, in short, *AI solution providers*. AI solution providers are particularly interesting for studying quality trade-offs: On the one hand, they cater to the needs of their clients, whose particular application domains usually load to a specific view on quality priorities. For example, for ML-based healthcare systems, the focus is usually on

privacy and system accuracy [14], whereas for autonomous vehicles, major attention is paid to reliability and safety [22]. On the other hand, as we observe in our study, AI solution providers align the priorities of their customers with their own strategic goals that can vary significantly (e.g., depending on their available resources [15], internal standards on security and privacy [27], ethical guidelines [18]). We report on the overall quality priorities, how they achieve them using tactics, what quality trade-offs they encounter, and how they balance these trade-offs.

Specifically, we address and answer three research questions:

RQ1: Which system quality attributes are relevant for companies that provide AI solutions? By answering this question, we identify the main quality priorities for certain companies.

RQ2: What tactics are used to achieve those quality attributes? To address this question, we share the main practical insights of how to achieve previously identified priorities.

RQ3: What major quality trade-offs are encountered by practitioners? By answering this question, we share quality dilemmas encountered in practice by our subject companies.

The synthesis of answers to all the RQs provides a focused collection of concentrated practical experiences, which can be applied by early-stage startups as best practices and used by academics to develop new solutions that address the observed trade-offs in alternative, possibly more effective ways.

2 Background

A *quality attribute (QA)* is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders [3]. In the context of this research, we use the term "quality priorities" which are the most important quality attributes for specific companies, conditioned by strategic goals and external requirements.

An *architectural tactic (AT)* is a "*technique an architect can use to achieve the required quality attributes*" [3]. Architectural tactics affect the overall system architecture in one way or another [16]. We consider the tactics that do not directly affect system architectures as "non-architectural" in the context of the current study. This division allows us to establish a line between organizational and technical solutions, which can be especially useful in assessing resources for their adoption. By definition, the connection between tactics and certain *quality attributes* is implied [16].

A *quality trade-off* is a compromise between several quality attributes [1]. *Balancing trade-offs* ensures that appropriate levels of satisficement are found for all involved quality attributes.

3 Related Work

Quality Priorities. The study of software quality for ML-enabled systems is an in-demand topic among practitioners and researchers [25], [26]. Different common quality models were developed by independent researchers [28], [17]. In 2023, the International Organization for Standardization issued a standard quality model for AI systems [9] that considered the specific nature of machine learning. While such studies are crucial to understanding the overall quality picture, in practice, it is barely possible to fully achieve all of them simultaneously [5]. As a result, AI development companies introduce quality priorities depending on their goals, external requirements (provided by clients), and internal requirements (dictated by strategic goals) [11]. These are the most important attributes that primarily reflect the overall quality of their solutions, while other characteristics may be subject to partial sacrifice. The decisions in this regard vary among companies, therefore, it is necessary to primarily identify the main priorities of the considered companies to understand their context.

Architectural and Non-Architectural Tactics. There are several review papers on architectural issues in the context of AI-based systems [4], [10], [21]. These papers explore a collection of existing architectural design decisions to address certain challenges without a clear reference to system qualities or with a focus on individual quality attributes and their metrics in isolation from other characteristics. As a result, possible trade-offs often remain unnoticed. The same situation with non-architectural tactics that are based on certain MLOps principles [19], agile-implementations for the development of ML-based software [23] and other relevant organizational decisions [24]. Using such knowledge is certainly important in the early stages of startups, however, without consideration of possible trade-offs it can lead to undesirable consequences. The research on industrial experience in this area can be beneficial and alert aspiring start-ups and SMEs about complex dilemmas that may arise.

Quality Trade-offs. Existing research actively explores quality trade-offs arising from the implementation of selected tactics in ML-enabled systems. For instance, between energy efficiency and system accuracy [6], performance and interpretability [2], performance and privacy [30]. While such studies provide deep investigations of specific cases, they leave the identification of the most crucial trade-offs for industries aside and focus on one particular trade-off. Moreover, related work on trade-offs is often performed from a fundamental academic perspective [13], without a clear rooting in practice. When it comes to strategic prospects of startups and early-stage SMEs, the need for high-level studies with insights from the industry arises, which is the focus of this work.

4 Research Method

To investigate quality trade-offs in ML-enabled systems in industry, we performed a multiple-case study. This method allowed us to gain a deep understanding of how companies handle quality objectives, implement tactics, and navigate trade-offs in practice, by focusing on several cases of interest. We selected our cases by focusing on a particular business segment of companies that provide AI-based solutions for other companies, in short, *AI solution providers*, as this would allow us to benefit from particular rich insights over different application domains. The data collection and analysis processes are described below.

Case Selection. The companies were selected based on the following criteria:

(1) *AI Solution Providers:* Each company has to be involved in the development, deployment, integration, or maintenance of ML-enabled systems or ML components.
(2) *Diverse Application Domains:* Each company has clients from different domains.

Table 1: Subject companies

Company ID	Domain	ML Team Size
C1	Custom Development of ML-based Software	approx. 30
C2	Integration of ML Components	approx. 10
C3	AI-based Content Creation & Machine Translation	approx. 20
C4	ML-based Software Support and Maintenance	approx. 30

(3) *Focus on High-Quality Solutions:* Each company has well-established processes for handling quality attributes like reliability, resource efficiency, maintainability, accuracy, and explainability in their systems.

An overview of our subject companies is provided in Table 1. All companies have their headquarters located in Sweden.

C1: This company develops customized ML-based systems for clients from different domains, including the healthcare sector, the financial sector, and the retail sector. The company provides solutions of different types, for instance, computer vision systems, natural language processing (NLP) systems, and ML-based business intelligence systems.

C2: This company integrates ML functionality in existing systems of clients from different domains, including the manufacturing sector, the real estate sector, and the governmental and public sectors. The company focuses on the integration of deep learning and large language models in workflows and data processing.

C3: This company develops AI-based content creation systems and shares them in the form of services. It delivers services to clients from different domains, including the financial sector, the manufacturing sector, and the retail sector. The company focuses on natural language processing and machine translation.

C4: This company develops and maintains customized ML-based systems for clients from different domains, including the construction sector, the manufacturing sector, and the public sector. The company provides different services for the development of ML-based control systems and embedded systems, however, the main focus is on the maintenance and support of existing ML-based systems, including AI model updates, proactive monitoring, and automated testing.

Our selected companies illustrate the wide-ranging application of AI and ML across various sectors, highlighting the importance of both development and maintenance for quality-driven solutions. Each company brings specific expertise and capabilities, enabling them to represent diverse needs and priorities.

Data Collection. Data collection was conducted through semi-structured interviews with key employees of each company. Participants were selected based on their role in the development, design, and maintenance of ML-based systems, ensuring that they had direct experience with the decision-making processes related to system quality. Our interviewees were software architects, ML engineers, project managers, and team leaders. Interviews with 6 participants were conducted (1-2 interviewees per company).

The semi-structured format maintains consistency in the core objectives by being structured around a set of predefined questions, while still providing flexibility in exploring aspects in detail using *ad hoc* follow-up questions.

This format is particularly suitable for discussions of artifacts that may be subject to different interpretations. For instance, architectural and non-architectural tactics can have varied levels of abstraction among different interviewees, as well as the same system qualities can be interpreted differently. Hence, during the interviews, we paid not so much attention to the terminology used by the participants, but to the underlying phenomenons they described.

For RQ1, interviewees were initially provided with full freedom to define their quality priorities, however, further, their responses were aligned in real-time with a predefined list of quality attributes to provide consistent terminology. For instance, we agreed with an interviewee to introduce the term "resource efficiency", when they had previously reported concerns associated with computational and labor resources. According to all respondents, the introduced terminology accurately characterized their priorities. For RQ2, interviewees reported architectural tactics to address previously identified priorities in a free form. For RQ3, interviewees provided the list of the most crucial quality trade-offs they encountered in practice and some experience of how to balance them. We note that during their answers on RQ3, they also provided certain architectural and non-architectural tactics to balance trade-offs that were not mentioned previously for RQ2. We decided not to update the answer to RQ2, but to retain these experiences as part of the answer to RQ3 to maintain the consistency of the interviews.

During the planning and conducting of the interviews, Hove and Anda's recommendations for semi-structured interviews in empirical software engineering research were followed [12]. Each interview lasted approximately 45 minutes. All interviews were recorded and transcribed for subsequent analysis. We make the interview questions publicly available in our Supplementary Artifact [1], furthering reproducibility and providing a basis for further investigations. Due to privacy concerns, we omit to publish the full interview recordings and transcripts.

Data Extraction. The recordings of the interviews were transcribed with the help of the Otter.ai platform [7]. Manual verification of the transcribed interviews confirmed the high quality of transcription and full factual correspondence with everything said in the video recordings. The next step was a deep analysis of the phenomena discussed. As mentioned earlier, interviewees could operate with different terminologies but describe the same things. Clarifying questions and refining our hypotheses with participants during the interviews allowed us to identify which quality attributes, tactics, and trade-offs they described. Further, all of the extracted data was structured in the resulting tables associated with RQs. For RQ1, quality priorities were grouped by companies reported them; for RQ2, tactics were grouped by quality priorities they addressed and the companies reported them; for RQ3 companies were grouped by trade-offs they reported and augmented with brief descriptions of cases.

5 Results

Quality Priorities. We now report the quality priorities provided by our subject companies, showing an overview in Table 2. All four

[1] https://doi.org/10.6084/m9.figshare.27074683.v1

Table 2: Quality Priorities per Company

Company ID	Quality Priorities
C1	Functional Suitability, Usability, Reliability, Resource Efficiency, Maintainability.
C2	Functional Suitability, Reliability, Data Quality, Resource Efficiency, Fairness.
C3	Functional Suitability, Reliability, Resource Efficiency, Maintainability.
C4	Functional Suitability, Maintainability, Resource Efficiency, System Accuracy, Explainability, Reliability.

companies reported functional suitability, resource efficiency, and reliability as their key quality priorities.

According to C1, their main quality priority is to follow the goals of their clients strictly. They conduct deep investigations of customer needs that further evolve into functional requirements. Compliance of the system with these requirements is called *functional suitability*. While those requirements can be dictated by the specifics of their clients, the company follows four more quality priorities that are not usually provided by them, which are universal for all their projects. The first one is *resource efficiency*. The company operates with limited labor and computational resources, which often creates a need to optimize certain processes and algorithms, re-balance existing powers, and offer feasible alternatives to unachievable solutions in terms of budget. They conduct constant workload monitoring and functioning capacity analysis. The second one is system *usability*. In addition to high client orientation, the company strives to deliver systems that are also efficient to the end users in terms of ergonomics and user interface optimization. The third priority is *maintainability*. Since the company also provides the services of post-production maintenance and technical support, it follows the strategy of "highly maintainable" solution development to avoid technical debts. It usually requires extra resources at the stage of design and development, however, it also brings serious resource savings strategically. Finally, *reliability* is one of the key priorities. Machine learning can never be completely stable due to evolving data, model updates, and changing environments, however, mitigation of this instability is one of their main objectives.

According to C2, the quality of the system is characterized primarily by the data it operates with. Crucial attention is usually paid to *data quality*. Improvement of data quality is especially resource-intensive when projects use dynamically updated data. This also highlights a pronounced role of *resource efficiency* monitoring. Another important factor is *system reliability*. In the understanding of C2, reliability consists of two main sub-priorities: the availability of the system in changing conditions and the safety of model usage. Finally, C2 reports that the model and system must always take into account ethical considerations. This is a *"fairness"* priority.

C3 considered system *reliability* as the main quality priority. The main current focus of C3 is on AI-based content generation, perspectives of which are quite promising, however, clients stay wary and conservative. The low reliability of developed components may call into question the current possibilities of AI to generate high-quality insightful content and motivate potential clients to ignore this option and continue to create textual content manually. It in turn can affect the overall success of the company. Further, *resource efficiency* is also a key priority. Each project requires individual assessment of the applicability and expediency of ML integration. Finally, *maintainability* of the developed ML functionality is required to be on a high level since the company also takes on the responsibility for supporting delivered solutions.

The main business strategy of C4 is to support and maintain existing systems built by their company as well as by third-party developers. From this follows the key priority of high system *maintainability*. The main goal is to improve this attribute for all systems within their scope to save resources strategically. Another important attribute is *system accuracy*. The company has a lot of requests from clients to enhance the accuracy of system outputs to obtain certain objectives. Another wide range of requests is related to the improvement in predictability that is achieved through high *explainability* of the system's behavior. Finally, *resource efficiency* and *system reliability* are also priorities that are constantly monitored and seriously influence the decisions made.

Table 3: Used Architectural Tactics

Quality Priorities	Companies Reported	Architectural Tactics
Resource Efficiency	C1, C2, C3, C4	Cloud-based Components
	C4	Model Pruning
	C4	Data Caching
Maintainability	C1, C4	Microservices
	C1, C4	Containerization
	C1	Experiment Tracking
	C1, C4	Dependency Tracking
Reliability	C1, C4	Model Verification Module
	C1, C4	Code Versioning
	C1, C4	Data Versioning
	C1	Pipeline Management
	C2, C3	Human-in-the-Loop
	C3	Rule-based Models
	C4	User Feedback Module
Data Quality	C2	Data Source Evaluation
Fairness	C2, C4	Bias Mitigation Module
	C4	Feature Engineering
Explainability	C4	SHAP Explanations
	C4	Local Interpretable Models
System Accuracy	C1	Data Preprocessing
	C1	Feature Engineering
	C1, C4	Hyperparameter Tuning
	C4	Data Postprocessing
Security	C1	Data Encryption

Tactics to Achieve Quality Priorities. To address their quality goals, our considered companies employ both architectural and non-architectural tactics. We now first present architectural tactics, before moving to non-architectural ones. Our considered companies employ the following architectural tactics, as presented in Table 3.

In company C1, teams implement diverse architectural tactics depending on the project specifics, however, some tactics are used in almost all projects to follow the internal quality priorities. To improve resource efficiency, the company often resorts to *cloud-based tools and components*. It saves significant resources, however, it is only possible with the consent of the clients. For instance, these are third-party cloud storage services, ETL (Extract, Transform, Load) tools, and serverless computing tools. To improve maintainability, C1 usually uses tactics of *microservice architecture* (which means the separation of the whole system into smaller independent parts, with each part having its realm of responsibility) and *containerization* (packaging an application and its dependencies into lightweight, isolated units, allowing for consistent and efficient deployment across different environments). *Experiment tracking* (managing and monitoring of model configurations, hyperparameters, metrics, and outputs) and *dependency tracking* (managing and monitoring of the libraries, frameworks, and external components) are also widespread tactics to ensure maintainability.

To achieve high reliability of developed solutions, C1 usually uses *code versioning* (managing and monitoring of changes in both ML and non-ML specific code), *data versioning* (managing and monitoring of changes in datasets used by ML models) and *pipeline management* (managing and monitoring of the dataflows and processes in a series of interconnected stages). Those tactics are usually implemented through experiment management tools. Finally, C1 developed its own model verification module (component designed to assess whether a model meets specific reliability requirements) that is usually integrated into safety-critical systems.

The basic set of architectural tactics such as *feature engineering* (selecting, modifying, or creating variables (features) from raw data), *data preprocessing* (cleaning, transforming, and organizing raw data into a suitable format) and *hyperparameter tuning* (searching for the optimal set of hyperparameters) are always applied to increase metrics of system accuracy.

Finally, *data encryption* algorithms are always introduced for security-critical and privacy-preserving systems. While security was not mentioned as a quality priority of the C1, it is often introduced for certain cases (e.g., for the healthcare domain).

C2 reported 3 architectural tactics. To address the "data quality" priority, the company usually conducts deep analysis and evaluation of data sources' reliability. According to C2, if the sources are trustworthy, it solves fundamental issues associated with data used by models. Further, to improve system reliability the company introduces internal or external experts to verify the most safety-critical deliverables provided by ML. Finally, C2 developed *automated bias mitigation module* to exclude a set of sensitive parameters used by the model at different stages.

C3 also shared the 3 most frequently used architectural tactics. Similarly to C1, the company sometimes uses different *cloud-based* components in their systems and development processes, e.g., cloud databases, pre-built ML models, and AWS AI service [8]. The company sometimes integrates domain *experts in the loop*, who are introduced to guide the model with more appropriate decisions for the most complex cases to ensure system reliability. With the same motivation, supporting *rule-based models* (to handle categorical data) are introduced.

Table 4: Used Non-architectural Tactics

Quality Priorities	Companies Reported	Non-architectural Tactics
Functional Suitability	C1	Regular Meetings with Clients
	C1	Knowledge Transferring
	C1, C3, C4	Scrum Practices
Usability	C1	Early Involvement of End Users
Maintainability	C2, C3	Supporting Documentation
	C4	Knowledge Base
Security	C2, C4	Data Governance
Resource Efficiency	C3, C4	Cross-Functional Teams

C4 shared 5 more tactics in addition to some of the ones mentioned previously. To save computational resources, the company introduces *model pruning* (removing unnecessary parameters from a machine learning model to reduce its size and complexity) and *data caching* (temporarily storing frequently accessed data in an operative storage layer, e.g., model outputs). To ensure reliability, the company integrates *user feedback module*, where it is possible, to receive operative information from end users on system faults and errors. Unlike C1, C4 primarily uses *feature engineering* to control the fairness of the system (e.g., absence of bias based on sensitive parameters, excluding outputs that violate privacy considerations). The unique priority of *explainability* is achieved by the integration of *Local Interpretable Model-Agnostic Explanations* (approximating behavior of the complex model with a simpler, interpretable model) and Shapley Additive Explanations (assigning each feature an importance value based on cooperative game theory) [29].

Our subject companies reported the following non-architectural tactics presented in Table 4.

To ensure the full functional suitability of the system, C1 holds regular meetings with stakeholders. These meetings help to regularly adjust the budget, coordinate tasks, negotiate functional requirements, and allocate resources. To successfully fulfill functional requirements, knowledge transferring among teammates is conducted regularly in an operative manner. Finally, the overall development process is usually organized in accordance with main Scrum principles [31] which provide the most possible flexibility in conditions of constantly changing client's priorities. A remarkable fact about C1 is its tactic to introduce end users in the early stages of system design and development to ensure the high usability of their solutions. According to C1, end users often differ from the formal clients. The company strives to always consider their needs as well.

To achieve high maintainability of the components integrated by C2, sufficient resources are invested in the development of supporting documentation explaining technical details and non-obvious dependencies. To address security and privacy issues C2 provides a data governance framework of policies, procedures, and standards together along with an integrated component to make data flows transparent to the clients.

C3, similarly to C1, emphasized the effectiveness of Scrum in successfully achieving project goals. Similarly to C2, the company

provides detailed support documentation for its systems and components. C3 is considered a cross-functional team (team members have diverse expertise and skills from different functional areas) the most labor-resource-efficient way of organization in their domain.

C4 developed a knowledge base (centralized repository of information, resources, and expertise) for internal employees with a collection of best practices based on previous experience in development and maintenance. Ready-made insights are applicable to standard cases and sufficiently simplify maintenance. Like other companies, C4 follows Scrum practices, data governance principles, and cross-functional team organization.

The answer to RQ2 presented different perspectives on the use of quality-driven tactics in ML-enabled systems. The most widespread architectural tactic was the use of *cloud-based components* to save computational and labor resources, while the most widespread non-architectural tactic was the use of *Scrum practices* to improve the functional suitability of final solutions.

Reported Quality Trade-offs. Based on the quality criteria and tactics identified in the previous research questions, our subject companies reported the quality trade-offs that affect their decisions most significantly. They are presented in Table 5.

C1 reported three quality trade-offs it deals with most frequently. According to C1, the first step in balancing any trade-off is to analyze the client's objectives and project scope. When it comes to the trade-off between resource efficiency and model accuracy, brainstorming by the team and the client is organized to prioritize the tasks and set performance thresholds (minimally acceptable levels of accuracy). Lack of labor resources can be compensated by the involvement of outsourcing powers requiring project budget extension. Computational resources can be compensated by the introduction of extra cloud-based components. The trade-off between resource efficiency and security appears rarely, but when it does, it seriously affects the whole project planning. First of all, security requirements from the stakeholders are deeply investigated by the team. When it is not possible to just use lightweight security protocols and outsource encryption algorithms, encryption specialists are introduced, which can lead to a significant increase of the project budget. The company strives to deliver highly reliable solutions, however, when the client requires extra robust functioning, the team spends extra resources on testing and simulations. Sometimes it introduces the ML-based algorithms of predictive maintenance to predict hardware failures and prevent downtime.

C2 usually deals with three main trade-offs. The first one is between resource efficiency and reliability. Since the team is quite small, it does not always have an opportunity to build complex models from scratch. In such cases, they agree with clients on the use of ready-made cloud-based models. The client and the team decide to be dependent on the third-party solution availability, however, the team provides at least one support alternative, which is a model with lower accuracy and performance that replaces the main one in case of third-party system failure. The second trade-off is between reliability and system accuracy. The team noticed the phenomenon of model overfitting when the formal metrics of accuracy are unrealistically high, but the ability of the model to generalize inputs is quite low. In such cases, the team introduces additional cross-validation, model pruning, or expanding the model training dataset. Finally, more complex models usually provide higher accuracy, but less explainability of their decision-making process. This case is totally dependent on the client's priorities. Customers are rarely interested in the explainability of the models, they accept models as black boxes as long as they efficiently solve their tasks. However, in cases when they are concerned, the team can replace complex models with simpler ones or introduce a technique of model stacking (by combining highly interpretable models (e.g., decision trees) with more complex models (e.g., deep learning).

C3 reported the six most common trade-offs in their experience. Regarding the trade-off between system accuracy and resource efficiency, the company selects appropriate models that provide acceptable accuracy but fit computational limits. To find a balance between system reliability and resource efficiency, the company involves automated testing, parallel testing, and test optimizations. C3 also reported cases when teams were focused on increasing system accuracy and optimizing resources, however, main functional requirements built on the needs of stakeholders were associated with other quality attributes. For instance, there was a case when the speech recognition module primarily required high speed of processing, but due to blurred functional requirements team spent a lot of effort on increasing accuracy but lost the focus on the speed of response. The only solution there according to C3 is to monitor and clarify client's needs constantly. The tactic of balancing explainability and system accuracy is the same as for C2. Finally, the integration of the expert to maintain a system is essential, however, when comes to the maintenance of complex models, humans can provide biases based on their subjective decisions. In the area of machine translation, it is especially relevant. Data and model versioning are used for systematic tracking of changes, allowing for identifying and correcting bias introduced in specific versions.

For company C4, the majority of trade-offs emerge between maintainability and other priorities. That is because the main business process for C4 is associated with the maintenance and support of ML-based solutions. When addressing these trade-offs, the companies usually do not aim for a balance between the different dimensions. Instead, explainability, security, and usability are typically prioritized over maintainability, based on requirements by specific customers. To meet these requirements, the company allocates extra resources for solution development and maintenance, which leads to higher costs and increased budget needs.

The answer to RQ3 presents the most common quality trade-offs occurring across different companies. The most frequently reported trade-off is between *resource efficiency and reliability*.

6 Discussion

Our findings shed light on the challenges faced by companies in balancing various quality attributes when developing ML-enabled systems. We now discuss their contribution to the theory of software quality of ML-enabled systems, their implications for practitioners and academics, and threats to validity.

Theory Building. For RQ1, by identifying quality priorities that exist across multiple organizations, such as reliability, resource efficiency, and functional suitability, we provide a high-level understanding of the strategic goals of real industry. This observation proves the fact that companies can operate in different ways and

Table 5: Identified Trade-offs

Trade-off	Companies	Description
Resource Efficiency vs. System Accuracy	C1, C3	Significant labor resources are spent on manual data preprocessing, feature engineering and hyperparameter tuning. Significant amounts of computational resources (GPU) can be used to train own models instead of using existing ones to improve system accuracy.
Resource Efficiency vs. Security	C1	Development of complex cryptographic algorithms requires significant labor resources, while encryption of large datasets requires significant computational powers.
Resource Efficiency vs. Reliability	C1, C3, C4	Sufficient resources (both computational and labor) are usually invested in running simulations, stress tests, and monitoring system behavior under various conditions.
	C2	Relying on third-party models often makes sense in terms, of resources (financial, computational, & labor), but if a third-party system fails or the server gets down there is no way to restore the process manually.
Resource Efficiency vs. Functional Suitability	C3	The obvious strategy to conserve resources can detract from the main needs of stakeholders and real strategic goals.
Resource Efficiency vs. Maintainability	C4	Prioritizing immediate resource savings usually leads to shortcuts in development processes, such as inadequate testing or documentation. It significantly complicates further maintenance.
System Accuracy vs. Functional Suitability	C3	The obvious strategy to increase system accuracy metrics can detract from the main needs of stakeholders and real strategic goals.
Reliability vs. System Accuracy	C2	High accuracy of the model sometimes connected to overfitting, when the model cannot sufficiently generalize input data.
System Accuracy vs. Explainability	C2, C3, C4	Integration of complex deep learning models (e.g., Large Language Models) significantly improves the accuracy of system outputs, however, this makes it barely possible to explain how the model comes to certain conclusions.
Fairness vs. Maintainability	C3	If the model is maintained by humans, they can introduce potential bias by decisions they personally consider the most rational.
	C4	Ethical considerations differ from one client to another. It makes it complicated for maintainers to properly set up bias mitigation modules and conduct data processing.
Usability vs. Maintainability	C4	Providing users with customization options improves overall system usability but introduces too many unnecessary dependencies that are complicated to monitor and maintain.
Security vs. Maintainability	C4	Strict security and privacy policies of security-critical clients introduce significant limitations on possible decisions for proper maintenance.
Explainability vs. Maintainability	C4	Highly explainable models (and designed systems) limits flexibility when adapting to new requirements. It leads to significant rework in case of changed priorities (for example, in favor of system accuracy).

serve different clients, however, they often address similar priorities when it comes to building ML-enabled systems. For instance, reliability is essential for the safe use of machine learning in industrial conditions, while resource efficiency is important due to the high computational demands of ML models.

For RQ2, we found that integration of cloud-based components is frequently used to architecturally achieve those priorities. However, while cloud-based architectures can enhance resource efficiency and scalability, they may decrease reliability, by introducing dependency on third-party services. Surprisingly, no company reported concerns associated with privacy and security, despite the fact that ML training might require user data, and its transferring to an external cloud provider might lead to privacy concerns. A possible explanation is the confidence of providers in the tooling used and transferred responsibility for making decisions about the possibility of using cloud tools and components to the clients. Another finding is the adoption of Scrum as a non-architectural tactic to manage resources, ensure functional suitability, and provide teams and clients with the necessary flexibility.

Finally, for RQ3, our study of trade-offs reported by companies leads to an understanding of what dilemmas are usually met by AI developers, how the decisions are made, and what techniques of balancing are used.

Implications. This research provides actionable insights for both startups and SMEs, which often lack the resources to experiment with various tactics and learn from trade-offs in the same way that larger organizations might. First, the startup should select key quality priorities to follow from a wide spectrum of possible attributes. Secondly, it should implement appropriate tactics to achieve those priorities consistently, such as those reported by our subject companies. Thirdly, they should be mindful of trade-offs that are likely to appear in the process of design and development, which can potentially save significant resources—as the proverb says, "forewarned is forearmed".

Researchers can use our findings in several ways. First, our list of identified trade-offs (Table 5) can serve as inspiration to develop new and improved solutions for important challenges in practice. For example, three of our four considered companies report on

system accuracy vs. *explainability* trade-offs—with the recent trend on large language models (LLMs), there is a dire need for LLM solutions that are able to provide explanations. Second, they can develop automated recommender systems for practitioners to identify quality priorities, tactics, and trade-offs. Third, they can further their knowledge of quality trade-offs by focusing on particular domains and scopes of quality goals.

Threats to Validity. The main threat to external validity is determined by the specifics of certain companies (i.e. set of clients they serve, their resources, and their budgets). While we see the potential of applying our findings to the startups and SMEs, the specifics mentioned previously may have a more severe impact than we expect. Furthermore, the fact that all the selected companies are headquartered exclusively in Sweden may reflect the priorities of AI developers in a particular country. To mitigate this threat, it is possible to conduct more interviews with full competitors of studied companies (i.e. that have clients from the same domains and perform the same set of activities) or deeply analyze the specifics of startups and SMEs when applying those findings.

Internal validity is associated with the selected methodology of multiple case studies. Despite the fact that interviewees are directly involved in the decision-making process and design of ML-enabled solutions, they still can represent their subjective understanding of the described processes. Other employees within the same development team may provide other insights on the same issues. To mitigate this threat we involved more than one interviewee per company where it was possible.

The main threat to construct validity is associated with our strategy to rely entirely on the responses of interviewees. The questions were formulated in a way to explicitly highlights our intention to study the most important and frequently addressed quality priorities, tactics, and trade-offs. However, there is still a risk that perspective on the importance of those is determined by the responsibilities of a specific interviewee. To mitigate this threat, we involved only interviewees with the roles of CEO, project manager, or team leader. From our perspective, such roles can provide the most strategic insights.

7 Conclusion

The findings emphasize the need for a more nuanced approach to addressing trade-offs between key quality attributes, particularly in resource-constrained environments such as startups and SMEs. This study investigated 9 quality priorities followed by 4 independent ML developing companies, 24 architectural and 8 non-architectural tactics to achieve those priorities, and 12 quality trade-offs that appear in the context of different projects per company.

Significant future work directions are to develop improved solutions addressing the identified trade-offs, recommender systems that support companies in identifying quality goals, tactics, and ways to balance trade-offs, and compare the results from related work with the results of this study.

Acknowledgment. This work was partially funded by Vetenskapsrådet, project *SEMLA: Software Engineering for Machine Learning - Integrated Approach*, and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Sebastian Barney, Kai Petersen, Mikael Svahnberg, Aybüke Aurum, and Hamish Barney. 2012. Software quality trade-offs: A systematic map. *IST* (2012).
[2] George Baryannis, Samir Dani, and Grigoris Antoniou. 2019. Predicting supply chain risks using machine learning: The trade-off between performance and interpretability. *Future Generation Computer Systems* (2019).
[3] Len Bass, Paul Clements, and Rick Kazman. 2003. *Software architecture in practice.*
[4] Manoj Bhat, Klym Shumaiev, Uwe Hohenstein, Andreas Biesdorf, and Florian Matthes. 2020. The evolution of architectural decision making as a key focus area of software architecture research: A semi-systematic literature study. In *ICSA.*
[5] Jan Bosch. 2000. *Design and use of software architectures: adopting and evolving a product-line approach.* Pearson Education.
[6] Alexander EI Brownlee, Jason Adair, Saemundur O Haraldsson, and John Jabbo. 2021. Exploring the accuracy–energy trade-off in machine learning. In *IEEE GI.*
[7] Melissa Corrente and Ivy Bourgeault. 2022. *Innovation in transcribing data: Meet otter. ai.* SAGE Publications, Ltd.
[8] Peter Elger and Eóin Shanaghy. 2020. *AI as a Service: Serverless machine learning with AWS.* Manning Publications.
[9] International Organization for Standardization. 2023. *ISO/IEC 25059:2023 Software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality model for AI systems.* Technical Report. ISO.
[10] Xavier Franch, Silverio Martínez-Fernández, Claudia P Ayala, and Cristina Gómez. 2022. Architectural decisions in AI-based systems: An ontological view. In *QUATIC.*
[11] Christian Gilbertson, Miranda Mundt, Joshua Teves, Simone Toribio, and Reed Milewicz. 2024. Towards Evidence-Based Software Quality Practices for Reproducibility: Practices and Aligned Software Qualities. In *ACM-REP.*
[12] S.E. Hove and B. Anda. 2005. Experiences from conducting semi-structured interviews in empirical software engineering research. In *METRICS.*
[13] Vladislav Indykov, Daniel Strüber, and Rebekka Wohlrab. 2024. Architectural Tactics to Achieve Quality Attributes of Machine-Learning-Enabled Systems: A Systematic Literature Review. *Available at SSRN 4909520* (2024).
[14] Nazish Khalid, Adnan Qayyum, Muhammad Bilal, Ala Al-Fuqaha, and Junaid Qadir. 2023. Privacy-preserving artificial intelligence in healthcare: Techniques and applications. *Computers in Biology and Medicine* (2023).
[15] Mansi Khemka and Brian Houck. 2024. Toward Effective AI Support for Developers: A survey of desires and concerns. *Queue* (2024).
[16] Suntae Kim, Dae-Kyoo Kim, Lunjin Lu, and Sooyong Park. 2009. Quality-driven architecture development using architectural tactics. *JSS* (2009).
[17] Hiroshi Kuwajima, Hirotoshi Yasuoka, and Toshihiro Nakae. 2020. Engineering problems in machine learning systems. *ML* (2020).
[18] Stefan Larsson. 2021. AI in the EU: Ethical Guidelines as a Governance Tool. *The European Union and the technology shift* (2021).
[19] Sasu Mäkinen, Henrik Skogström, Eero Laaksonen, and Tommi Mikkonen. 2021. Who needs MLOps: What data scientists seek to accomplish and how can MLOps help?. In *WAIN.*
[20] Jorge Melegati and Fabio Kon. 2020. Early-stage software startups: main challenges and possible answers. In *Fundamentals of Software Startups: Essential Engineering and Business Aspects.* Springer, 129–143.
[21] Henry Muccini and Karthik Vaidhyanathan. 2021. Software architecture for ML-based systems: What exists and what lies ahead. In *WAIN.*
[22] Huihui Pan, Mao Luo, Jue Wang, Tenglong Huang, and Weichao Sun. 2024. A safe motion planning and reliable control framework for autonomous vehicles. *T-IV* (2024).
[23] Romesh Ranawana and Asoka S Karunananda. 2021. An agile software development life cycle model for machine learning application development. In *SLAAI-ICAI.*
[24] Maria Saenz, Elena Revilla, and Cristina Simón. 2020. *Designing AI systems with human-machine teams.*
[25] Peter Santhanam. 2020. Quality management of machine learning systems. In *EDSMLS.* 1–13.
[26] Alex Serban and Joost Visser. 2022. Adapting software architectures to machine learning challenges. In *SANER.* 152–163.
[27] Lijun Shan, Behrooz Sangchoolie, and Peter et al. Folkesson. 2019. A survey on the applicability of safety, security and privacy standards in developing dependable systems. In *SAFECOMP.*
[28] Julien Siebert, Lisa Joeckel, Jens Heidrich, Adam Trendowicz, Koji Nakamichi, Kyoko Ohashi, Isao Namba, Rieko Yamamoto, and Mikio Aoyama. 2022. Construction of a quality model for machine learning systems. *SQ* (2022).
[29] Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. 2022. On the tractability of SHAP explanations. *JAIR* 74 (2022), 851–886.
[30] Max van Haastrecht, Matthieu Brinkhuis, and Marco Spruit. 2024. Federated Learning Analytics: Investigating the Privacy-Performance Trade-Off in Machine Learning for Educational Analytics. In *AIED.*
[31] Kevin Vlaanderen, Slinger Jansen, Sjaak Brinkkemper, and Erik Jaspers. 2011. The agile requirements refinery: Applying SCRUM principles to software product management. *IST* (2011).