CLEI Journal, Volume 29, Issue 1, Paper 2, February 2026

# Systematic mapping of non-functional requirements and their impacts in architectures for artificial intelligence

**Yefry Astaiza, Oscar Santiago López Erazo, Luis Freddy Muñoz**
Logiciel, Fundación Universitaria de Popayán
Popayán, Colombia
yefry.astaiza@estudiante.fup.edu.co,
{santiago.lopez, freddy.munoz}@docente.fup.edu.co

**Juliana Delle Ville, Giuliana Maltempo, Leandro Antonelli***
LIFIA, Facultad de Informática, Universidad Nacional de La Plata
*CAETI, Facultad de Tecnología Informática, Universidad Abierta Interamericana
La Plata, Argentina
*Buenos Aires, Argentina
{jdelleville, gmaltempo, lanto}@lifia.info.unlp.edu.ar

**Julio Ariel Hurtado, Cesar Collazos, Pablo Ruiz, Vanessa Agredo**
IDIS, Universidad del Cauca
Popayán, Colombia
{ahurtado, ccollazo, pablohernando, vanessaagredo}@unicauca.edu.co

**Abstract**

Artificial intelligence (AI) is a branch of computer science that seeks algorithms capable of learning, reasoning, adapting, and performing tasks in a human-like manner. Software is determined by a set of functional requirements (FRs) and non-functional requirements (NFRs). NFRs, such as scalability, security, performance, and usability, define how a system should function beyond its basic functionalities. The architecture must be designed to meet these NFRs, ensuring system efficiency and reliability while maintaining a flexible and modular structure. Project-specific requirements, technological advancements, and the domain greatly influence the variety of NFRs and architectures that will be essential for the development of AI-based systems. Selecting an appropriate architecture is a process intrinsically guided by the prioritized NFRs. This systematic mapping seeks to provide the NFRs, architectures, and how these impact the selection of architectures for AI-based systems and a taxonomy. The evidence gathered shows that NFRs tend to prioritize security, scalability, and accuracy. At the architectural level, there is a similar trend toward patterns such as microservices and lambda. Finally, in terms of impact, microservices is highly influenced by reliability and scalability, workflow orchestration by transparency and explainability, and federated learning by sensitive data security.

**Keywords:** Artificial Intelligence, Non-functional requirements, Software Architecture, Systematic Mapping, Impact

## 1 Introduction

Artificial intelligence (AI) is a branch of computer science that seeks to create algorithms capable of learning, working, adapting, and performing tasks in a human-like manner. The term AI is used to refer to any machine that uses any type of algorithm or statistical model to perform perceptual, cognitive, and conversational functions typical of the human mind, such as speech, visual recognition, problem solving, and reasoning [1][2] . AI algorithms make part of software to satisfy the needs of users [2].

Software products are determined by a number of functional and non-functional requirements. Functional requirements (FRs) specify the behavior of the software product, while non-functional requirements relate to quality attributes and standards the product must satisfy. Non-functional requirements (NFRs) play a fundamental role in the design of any system, according to ISO/IEC 25010 standard for the assessment of quality attributes in traditional software products [3], [4]. Another fundamental aspect of a software system is its architecture [5], as it affects the abstraction of heterogeneous hardware components, ensures smooth transactions between network protocols, and contributes to the overall quality of the system. A system's architecture has an impact on functional requirements, outlining tools and components that can be used, and this in turn has an impact on non-functional requirements [6]. The design of a system's architecture must aim to satisfy its NFRs, ensuring efficiency and reliability while keeping the structure flexible and modular [7].

Non-functional requirements specify how the system should perform beyond its basic functionalities [4]. Some NFRs relate to human-computer interaction (HCI), in so far as they aim to ensure that the system is intuitive, accessible, and user-centred [8]. The attention here is directed towards NFRs relating to user experience, including usability, accessibility and response time. A well-designed architecture must adopt HCI principles from the beginning, allowing for the smooth integration of optimised user interfaces and ensuring a positive experience in terms of both technical compliance and user expectations [9]. Designing AI-based systems requires a balance between the choice of attributes and architectures. Project-specific requirements, technological advancement, and domain greatly affect the variety of NFRs and architectures that will be essential to building AI-based systems, and this problem can also be observed in the Internet of Things (IoT). These NFRs critically impact the software architecture, making it a challenging task to satisfy these NFRs in order to have a flexible and modular structure.
These quality attributes directly impact the software's inherent architectural decisions, even more so than functional ones. They also serve as criteria for selecting architectural patterns or styles, as well as technologies, for example, a layered architecture and its maintainability. Within software architecture, elements such as goals, constraints, and the quality attributes themselves are considered, making it impossible to develop a system that meets stakeholder expectations without addressing NFRs [10] .

Among the challenges mentioned for NFRs and architectures are the following: (i) AI tends to encompass a wide range of ideas and practices, ranging from the details of the technology used to even how it is defined, (ii) Traditional mechanisms such as replication and multi-version programming might not work in intelligent systems, and so this field is in its nascent stage, (iii) All programs, including AI systems, can be hacked by malicious users, (iv) A human agent is replaced by an AI-based agent to perform a critical task in healthcare, military, or other mission-critical situations. In these situations, any type of decision must be explainable, (v) If the size of data collected by AI-based systems increases, efficient algorithms are essential for data analysis and decision making [11].

As mentioned earlier, project-specific requirements, technological advancement, and domain greatly affect the variety of NFRs and architectures that will be essential for building AI-based systems. Thus, this systematic mapping focuses on providing what NFRs, architectures and how these impact the selection of architectures for AI-based systems. have been used in AI-based systems. Because they are vital for: (I) identifying patterns, (ii) replicability of solutions in the context of AI, (iii) establishing quality standards, (iv) making informed architectural decisions and (v) a taxonomy organizes the NFRs and the software architectures that satisfy them in AI-based systems. A preliminary version of this article was presented at the *"Jornadas Iberoamericanas de Interacción Humano-Computadora"* 2025 in Zacatecas, Mexico; the current paper contains an extension of the version presented at the event. The paper is organized as follows: Section 2 presents related works; Section 3 presents the systematic mapping methodology adopted. Section 4 describes the state of the art in the domain of artificial intelligence. Section 5 presents our findings and the answer to the research questions posed in Section 2. This is followed by a discussion of our findings in Section 6, and conclusions in Section 7.

## 2 Related Works

Few works have been found that relate NFRs and architectures, which enhances the importance of our work, these studies are listed below. This study analyses functional and non-functional requirements for artificial intelligence (AI) systems. The paper highlights that traditional approaches do not fully address the unique challenges of AI-intensive applications. Through a systematic mapping, Ahmad et al. [12], identified primary studies in order to explore methodologies, tools and techniques for specifying requirements in AI systems. One of the most important findings was that although some frameworks such as UML and domain-specific modeling languages are useful, current approaches are not adaptive enough for the needs of AI-based systems. This is due to unique characteristics such as uncertainty, data-driven decisions, and the need to incorporate ethical principles, explainability, and trust. Among the challenges identified are the difficulty in defining clear requirements, handling trade-offs between conflicting objectives (such as accuracy vs. privacy). This work indicates that there is an urgent need to develop approaches adapted to the changing requirements of AI-driven software engineering. Abdullahi et al. [13], conducted a systematic mapping study on quality models for artificial intelligence (AI) systems, software, and components. The aim of the present study was to critically analyze, classify, and evaluate existing models. It was found that, although there are proposals for AI-intensive systems and software, there are no specific models for software components, such as those based on machine learning. The identified gap lies in the fact that there are few complete and specific quality models for software components in AI, as well as the need for metrics that allow quantitative and objective evaluations. Habiba et al. [14], address requirements engineering (RE) for artificial intelligence (AI)-based systems, where they highlight the complexities added by the adaptive and probabilistic nature of these systems. A systematic mapping is performed in order to identify existing practices, challenges, and research gaps. It is observed that most works focus on requirements analysis and specification, with challenges such as explainability. The gap lies in the lack of mature methodologies and standard tools to address the particularities of AI, indicating an urgent need for adaptation in traditional RE practices. Kaur and Kaur [15], perform a systematic mapping between the years 2012 and 2023, applying machine, deep and transfer learning techniques in requirements classification. The analysis includes methods such as SVM, CNN, and evaluation metrics such as accuracy and completeness. Further investigation of some of the research directions is planned with an emphasis on the integration between application reviews and software architecture by using AI techniques. Sofian et al. [16], perform a systematic mapping on the use of artificial intelligence (AI) techniques in software engineering (SE). It examines studies published between 2015 and 2021, identifying phases of the software lifecycle where these techniques, such as machine and deep learning, have been applied to improve activities such as testing, development and maintenance. The analysis highlights the increasing use of AI, especially in automation and prediction, but identifies a gap in the homogeneous adoption of these techniques in all phases of SE, underlining the need for future research in real-world contexts and less explored phases, such as design and deployment. Regarding the identified gaps, it can be identified that a generalized architectural framework that covers multiple domains and quality requirements has not been achieved, which limits adaptability, an essential attribute of AI. De Martino and Palomba [17], conduct a systematic literature review on non-functional requirements (NFRs) in machine learning systems. It identifies 31 NFRs grouped into six main categories and summarizes 26 key challenges in their management. It is based on the analysis of 130 studies and highlights the lack of a previous comprehensive synthesis in this field. Its objective is to guide future research and improve practical support for the development of ML systems, taking into account quality, reliability, and sustainability. Ijaz and Allah [18], present a systematic review of approaches for prioritizing non-functional requirements (NFRs) in software engineering. It identifies multiple methods, including AHP-based approaches and architecture, but does not delve into the relationship of NFRs to architecture, historical projects, and artificial intelligence techniques. It also analyzes which methods have been validated and how. Its objective is to guide researchers and practitioners in selecting effective strategies for managing NFR prioritization in different development contexts. Ahmad et al. [19], analyzes how requirements are specified in AI-based systems, identifying 43 primary studies. It shows that traditional requirements engineering (RE) approaches are insufficient for AI systems, particularly in aspects such as ethics, explainability, and data management. Three conceptual frameworks and various tools and notations used are highlighted. The study concludes that new or adapted methods are needed to address the unique challenges of requirements engineering in AI-enabled systems.

## 3 Methodology

In this work, a systematic mapping was carried out following two approaches: (1) the guide for systematic reviews in software engineering proposed by Kitchenham [20] and (2) the guide for conducting systematic mapping studies in software engineering proposed by Petersen *et al* [21].

### 3.1 Research questions

Quantum computing research currently offers countless areas for exploration. In this paper we address two specific aspects, non-functional requirements and architecture, based on the following questions, respectively:

- What are the non-functional requirements used in AI-based systems?
- What architectures are used in in AI-based systems?
- How do non-functional requirements impact the selection of architectures for AI-based systems from the above questions?

### 3.2 Abstract and Keywords

To create the search string, the main terms and possible synonyms were identified in order to obtain as many relevant primary studies as possible. The following main search strings were identified: "Non-Functional Requirements", "Software Engineering", "Quality Characteristics", "Software Architecture", "Relationship", "Artificial Intelligence" and "AI-based systems". The alternative terms "NFRS", "Architecture" and Relationship were also identified. Table 1 summarizes the identified search strings.

**Table 1.** Search string

| Main Terms | Alternatives Terms |
|---|---|
| Non Functional Requirements<br>Software Engineering<br>Quality Characteristics<br>Software Architecture<br>Artificial Intelligence<br>Relationship | (Software Architecture) AND (<br>(non-functional requirements)<br>OR (NFRs)<br>OR (Quality Characteristics)<br>AND (Artificial Intelligence)<br>OR (AI-based systems))<br>AND (non-functional requirement OR (NFRS) ) AND<br>(Software Architecture) AND (Relationship) |

### 3.3 Search strategy

The following digital repositories were consulted: Scopus, Science Direct, IEEE Xplore and Springer. Search strings were applied to the fields Title, Abstract and Keywords. The documents considered in this study include research articles, conference proceedings, books, technical reports and graduate papers in English and Spanish published between 2019 and 2025. Table 2 summarizes the elements of the search strategy

**Table 2.** Search strategy

| Ítem | Description |
|---|---|
| Repositories: | Scopus, Science Direct, IEEE Xplore y Springer |
| Publication type: | Research papers, Conference papers, Books, Technical reports, Technical reports and Degree works |
| Search fields: | Title, Abstract y Keywords |
| Language: | Studies written in English or Spanish |
| Publication period: | 2019 - 2025 |

### 3.4 Inclusion and Exclusion Criteria

The search only considers academic and professional studies carried out in conferences, journals, technical reports or graduate works. This work excludes personal blogs, web pages or works written in Spanish, with the exception of some works in Spanish that were included due to their contribution to the research, as shown in Table 3. Works that do not focus on software engineering, non-functional requirements and architectures for artificial intelligence are not considered. Workshop abstracts and duplicate papers from the same search in different repositories were excluded.

**Table 3.** Inclusion and exclusion criteria summary

| Criteria | Rules |
|---|---|
| Inclusion Criteria | Terms included in the search string.<br>Studies reported in articles, conferences, books, technical reports or degree works.<br>Studies written in English and Spanish.<br>Publications from 2019 to 2025.<br>Studies that establish software engineering, non-functional requirements and architectures for AI-based systems. |
| Title and abstract Inclusion and exclusion criteria | Studies that do not address software engineering, non-functional requirements and architectures for AI-based systems.<br>Studies that only present abstracts or slide content.<br>Content from web pages, personal blogs or brochures. |
| Exclusion criterion for full text | Studies presenting workshop abstracts |

According to Petersen *et al.* [21], quality assessment is most essential in systematic reviews to determine the rigor and relevance of the primary studies. In systematic mapping it is not necessary, although it is recommended, to perform a quality assessment. The use of the classification of research types according to the category of proposed solutions mentioned in Wieringa *et al.* [22] would contain studies without empirical or scientific evidence. Although the latter are not usually included in systematic reviews, they are important in systematic mappings, as they help to detect trends in the topics worked on. For this reason, findings with and without empirical evidence were included in the present study. To refine the selected studies, we used the checklist proposed by Zarour *et al*. [23], reproduced in Table 4. For each item there are three possible answers: Yes = 1 point, No = 0 points and Partially = 0.5 points. Using the first quartile (6/3 = 2) as the cut-off point, if a publication obtains a score lower than 2 it is excluded from the list of final studies to avoid low-quality papers.

**Table 4.** Quality criteria checklist

| # | Question |
|---|---|
| QA1 | Is the objective of the study sufficiently well explained? |
| QA2 | Is the idea, approach and limitations presented clearly explained? |
| QA3 | Are threats to validity taken into account? |
| QA4 | Is there an adequate description of the context in which the study was conducted? |
| QA5 | Are the results of the research clearly established? |
| QA6 | The study was designed to achieve these objectives? |

## 4 Results of the systematic mapping

By applying the search strings mentioned above to the selected repositories, a total of 2,391 results were obtained: Scopus (704), Science Direct (424), IEEE Xplore (488), and Springer (775). These results were filtered by title, abstract, and conclusions, resulting in 200 articles (Scopus: 59, Science Direct: 35, IEEE Xplore: 41, Springer: 65) as shown in Table 2. Subsequently, the inclusion and exclusion criteria defined in Table 3 determined which articles were excluded. Additionally, a quality-based criterion, defined in Table 4, was applied. After applying both sets of criteria, 38 articles were selected (Table 5). Sections 3.1 and 3.2 describe the articles selected after this filtering process on studies that establish software engineering, non-functional requirements and architectures for AI-based systems.

**Table 5.** Summary of results after applying the search strategy

| Search String | Repositories | Results | Title, abstract and conclusions | Selected with Inclusion, Exclusion, quality criteria |
|---|---|---|---|---|
| 1 | Scopus | 704 | 59 | 29 |
| | Science Direct | 424 | 35 | 7 |
| | Ieee Xplore | 488 | 41 | 1 |
| | Springer | 775 | 65 | 1 |
| | **Total** | **2391** | **200** | **38** |

Although the initial search yielded 2,391 papers, applying filters by title, abstract, conclusions and inclusion, exclusion, and quality criteria reduced the sample to 38 primary studies. In a systematic mapping, validity does not depend on the absolute number of articles included, but rather on whether the selected set is sufficiently diverse and representative to cover the domain studied [20] [21]. The 38 studies analyzed meet this criterion, as they cover the main non-functional requirements identified in the literature and the architectural patterns most frequently reported in the reviewed works.

### 4.1 Non-functional requirements in AI-based systems

Iván Haro [24] analyzes the use of Large Language Models (LLMs), such as ChatGPT and Gemini, developing a specialized model to recommend and apply design patterns, in order not to depend on human validation, for example, Rubén Alonso, Rodolfo E. Haber et al. [25], analyze how interoperable software platforms can integrate AI in the industry. While Daole et al. [26], propose OpenFL-XAI, a software for federated learning (FL) of explainable artificial intelligence (XAI) models, using a system based on fuzzy rules. In contrast Ranit Chatterjee et al. [27], propose to identify and classify ASFRs to avoid expensive refactoring processes using a model based on deep neural networks. Habibullah et al. [28], analyze the challenges that ML systems face in managing NFRs, using interviews and surveys to identify key differences in how NFRs are defined and measured for ML systems. However, Bajraktari et al. [7], propose a framework for documenting ML-specific NFRs to improve accuracy and consistency in documenting these NFRs. Muccini and Karthik [29], present ML-based systems and aim to highlight current architectural practices and propose improvements for the development of these systems using ML, IoT, and statistical optimization techniques. Martin Heyn et al. [30], propose a compositional architectural framework to address the challenges of (AI) systems embedded in the (IoT), their objective is to create a framework that enables collaborative and integrated design, addressing quality aspects such as security, robustness, and efficiency. Nascimento et al. [31], study the performance of software engineers in specific tasks with AI, and also evaluate performance, efficiency, and reuse in ChatGPT and IoT in complex contexts. Martínez García et al. [32] , propose a solution based on convolutional neural networks (CNN), to improve the classification of non-functional requirements (NFR), using natural language processing techniques and pre-trained matrices such as Word2Vec and FastText. Lu, et al. [33] aim to understand how AI developers perceive and implement ethical principles. Through interviews with 21 experts, they identify common problems, such as the lack of mechanisms for continuous monitoring of AI systems. Rocha de Sousa [34], proposes a software architecture for NFRs in intelligent systems that process large amounts of data generated in distributed environments, implementing NFR monitoring tools that manage resources and detect quality of service violations. Haldorai, and Ramu [35], examine how AI can predict software reuse. They make use of neural networks and the Flexible Random Fit algorithm to optimize the reuse prediction of software components in service mesh systems. Kaur and Kaur [15], performed a systematic analysis

for NFRs. Their study reviews current applications to optimize requirements identification in big data and application reviews, highlighting improvements in accuracy and promising results. Liubchenko [36], assesses the risks associated with NFRs, particularly those influenced by AI/ML components, uses a requirements tree-based approach, mapping dependencies between requirements to perform more accurate risk assessments. Seungkyu Park et al. [37], present a methodology for designing architectures, based on (AI), the objective of their study is to provide a clear structure, called IMO (Input-AI Model-Output), that facilitates the identification of the NFRs necessary to develop AI models.

### 4.2 Adopted architectures in AI-based systems

Vajpayee et al. [38], establish scalable data architectures for AI applications. The research faces the limitation of balancing between scalability and architectural simplicity. Haindl et al. [39], seek to develop a reference architecture to optimize collaboration between humans and AI systems in smart manufacturing environments. Graef and Georgievski [5], implemented a service-oriented software architecture to improve AI-based planning systems using a Service-Oriented Architecture (SOA). Lo et al. [40], conduct a study on the architectural design of federated learning systems, employing an approach based on systematic literature review and real-world applications, classifying categories and aggregating models. Aksakalli, et al. [41], identified deployment and communication patterns in microservices architectures. Through a review, they made a multi-phase selection process to ensure relevance and quality. Alves et al. [42], specified a framework for the orchestration of (ML) workflows applied to (IoT) data, using a microservices architecture. Borelli et al. [43], propose a classification of architectural patterns for the development of IoT applications, their objective is to address the complexities of designing IoT architectures that integrate sensors, actuators, and data processing components. Martín et al. [44], offer an end-to-end solution for managing ML pipelines in continuous data environments by integrating technologies such as Apache Kafka and Docker containers. Runpakprakun and Washizaki [45], explore architectures that can optimize both the response speed and consistency of ML models by deploying Lambda and Kappa in applications with high accuracy requirements facing challenges in result consistency between batch and real-time processing layers. Altintas et al. [46], present a dynamic composition approach for heterogeneous systems in remote sensing, employing federation of Kubernetes clusters and microservices. Nazir [6], investigated challenges, best practices, and key architectural design decisions in ML systems, on the main architectural decisions that could guide the design in large-scale projects. Elhabbash et al. [47], address a microservices architecture designed for predictive maintenance in Industry 4.0 environments, using incremental learning to dynamically adapt to changes in machine operating data. Jayesh Rane et al. [48], investigate scalable and adaptive deep learning algorithms for ML systems. The study focuses on optimizing the use of deep learning algorithms in applications that handle large volumes of real-time data. Laurens Martin [49]. represent workflow independence consistently and technologically, simplifying the integration and deployment of ML solutions in serverless environments. Mills [50], focuses on the design of hyper-automated AI platforms for complex digital time-spaces, that maintain their effectiveness under changing conditions without human intervention. Shah Zeb et al. [51], analyze how next-generation (NextG) wireless networks and computational intelligence (CI) use multi-tenant service architectures and patterns based on microservices and service-oriented architecture (SOA), to facilitate the provision of customized services for various industrial applications.

### 4.3 Relationship between software architectures and non-functional requirements for artificial intelligence

Non-Functional Requirements (NFRs) reflect the desired qualities of a system, such as availability, performance, portability, and maintainability. These quality attributes impact the software's inherent architectural decisions, even more than functional ones. They can also be used as criteria for selecting architectural patterns or styles, as well as technologies, for example, a layered architecture and its maintainability. In software architecture, there are considerations of elements such as goals, constraints, and quality attributes. These considerations make it impossible to develop a system that meets stakeholder expectations without addressing NFRs. One of the most common problems found in specifications is that they are not formally documented, or they are expressed informally. Architects also tend to define them based on their experience rather than on attributes proposed by users. NFR collection occurs iteratively throughout the software lifecycle. In conclusion, a key challenge is managing NFRs from their collection to their validation and their changes over time [52]. It is considerable to note that the main drawback of all of the above-mentioned AI-based approaches is that their scalability has not been evaluated. The

validation has been performed with a very small number of requirements which raises questions about their strengths [18]. Although NFRs are very important for ML systems, it is complex and difficult to define, allocate, specify, and measure NFRs for ML systems. Currently the industry and research does not have specific well-organized solutions for managing NFRs for ML systems. Because of unintended bias, the non-deterministic behavior of ML, and expensive and time-consuming exhaustive testing [53]. Some challenges are defining requirements, explaining predictions, addressing ethical issues, and issues with data requirements. Another major issue presented in the literature was the lack of integration between software engineers and data scientists [19], [54]. This study mentions that there are some empirical experimentations and approaches that can deal with the non-functional aspects that previous research did not consider. The classification outlined in the study can be used as input to increase understanding of the problems that arise when developing machine learning. [17].

# 5 Findings

## 5.1 What are the non-functional requirements used in AI-based systems? (Q1)

As can be seen in table 6, the Non-Functional Requirements (NFR) indicates that there is a notable trend in scientific texts that address the topic of artificial intelligence, such as scalability (Sca) and security (Sec), with 8 appearances in the consulted resources. Also frequent are the requirements of precision (Acc), with 6 appearances, and explainability (Ex), with 5 appearances. Likewise, the requirements addressed in the bibliography were those of adaptability (Ad), reliability (Re), and interoperability (Intop), with 3 appearances each. Another requirement addressed was performance efficiency (Perf).

**Table 6.** Frequency of non-functional requirements in the analyzed publications

| NFR | Sca | Sec | Acc | Ex | Ad | Re | Intop | Perf |
|---|---|---|---|---|---|---|---|---|
| Frec. | 8 | 8 | 6 | 5 | 3 | 3 | 3 | 1 |
| Ref. | [7], [24]–[30] | [7], [25], [28], [30]–[34] | [26]–[28], [31], [35], [36] | [7], [26], [28], [30], [32] | [7], [28], [31] | [26], [28], [30] | [7], [26], [27] | [32] |

## 5.2 What architectures are used in AI-based systems? (Q2)

From the articles consulted, it is observed that the number of architectures discussed in the context of artificial intelligence is extensive, since 7 different patterns could be found, in which the microservices architecture stands out with 9 mentions, followed by the Lambda architecture with 5 appearances in the articles consulted. The infrequent architectures are the SOA pattern and the Federated Learning pattern with 2 mentions each. Other architectures mentioned are Data Pipeline, Workflow orchestration and Layared pattern.

**Table 7.** Frequency of different architectures in the analyzed publications

| Pattern | Microservices Pattern | Lambda Pattern | SOA Pattern | Federated learning Pattern | Layered Pattern | Data Pipeline Pattern | Workflow orchestration Pattern |
|---|---|---|---|---|---|---|---|
| Frec. | 9 | 5 | 2 | 2 | 1 | 1 | 1 |
| Ref. | [6], [38], [41], [42], [45]–[47] | [38], [39], [44], [45], [50] | [5], [51] | [40], [48] | [43] | [45] | [49] |

### 5.3 How do non-functional requirements impact the selection of architectures for AI-based systems from the above questions? (Q3)

NFRs directly and more strongly impact architectural decisions than functional ones, acting as key criteria for selecting patterns and technologies. Their omission makes it impossible to meet stakeholder expectations, so their identification and strategic alignment with their needs is a critical success factor in AI systems. Among the findings (See Table 8): Microservices and Data Pipeline excel in scalability, adaptability, and reliability thanks to their modular design and parallel processing. For sensitive data, Federated Learning is optimal in terms of security by avoiding the movement of raw data. Layered facilitates explainability through its sequential structure, which isolates and audits decisions by layers. Workflow Orchestration guarantees transparency with end-to-end logging of inputs/outputs at each step. SOA and Lambda face performance and interoperability risks: SOA due to bus bottlenecks; Lambda due to the complex integration of batch/stream technologies that increase latency and degrade efficiency.

**Table 8.** NFRs impact on Architectures for AI-based System

| Pattern | Nfr Impact |
|---|---|
| Microservices Pattern | In terms of scalability, the evidence shows it is positive due to microservices that can scale IA components independently. In the other hand, security is neutral or mixed. Because if the services are isolated it reduces the attacks but communication between services increases the risk of surface. Accuracy is not affected (neutral), because it depends on the model implementation, not the architecture. But explainability is negatively affected, due to the complexity in tracing each distributed decision across multiple services. Adaptability is positively affected, on behalf of agile updates of individual services without affecting the entire system. Also, Reliability is positively affected with its inherent resilience (the isolated failures do not bring down the entire systems). Interoperability is also affected positively on behalf of Standardized APIs to integrate with other systems. Finally, performance efficiency is mixed due to the low latency of parallelism and communication overhead. |
| Lambda Pattern | Scalability demonstrates significant strength (positive), as the decoupled batch and speed layers can scale independently in response to varying load demands. Performance Efficiency is also robust (positive), benefiting from optimizations tailored for both massive-scale processing and low-latency requirements. System Reliability is enhanced by the inherent redundancy (positive), where the batch layer serves as a critical fallback mechanism during streaming failures. But there are several attributes that present inherent challenges (negatively affected): Accuracy carries inherent risk due to potential inconsistencies between batch and real-time computation results; Explainability becomes complex owing to the difficulty in generating unified explanations across distinct processing layers; and Adaptability is constrained, as structural modifications necessitate intricate synchronization between both layers. Security and Interoperability receive neutral assessments – the former necessitates implementing distinct protection policies for each layer, while the latter is contingent upon the compatibility between the underlying batch and streaming technologies. |

| Pattern | Nfr Impact |
|---|---|
| SOA Pattern | The evidence shows that this pattern has several virtues: In terms of security, centralization (ESB) facilitates policy management. Also, with adaptability, due to reusable services that accelerate changes. And finally, these kinds of architectures are highly interoperable because of the inherent Standarization (SOAP/REST). In other hand, the evidence shows some inherent challenges: Scalability is limited, is more scalable than monoliths but less than microservices due to coupling. The bus service dependency makes it difficult to map decision flows, making this kind of architecture hard to explain. The ESB is not reliable because it is a single point of failure, risking a system failure. As there is one bus, the overhead of it reduces performance. Finally, the evidence shows that this kind of architecture has no direct impact on accuracy. |
| Federated Learning Pattern | The evidence shows that these kinds of architectures have virtue: it has decentralized distribution that allows to scale millions of devices. The data does not leave the source minimizing privacy risk resulting in a more secure system. Also, the model updates without collecting centralized data making it more adaptable. In the other hand, the evidence show that this kind of architectures has some challenges: Local bias or poor data quality on device affects the overall model reducing the accuracy of it. Interoperability is challenging because it requires standardization in device-server communication. Also, explaining aggregated models from heterogeneous sources is difficult. However, the evidence show that, reliability is mixed due its tolerance of local faults but is still dependent on the aggregation server. Performance is not affected, due the bandwidth savings and computational cost on devices. |
| Layered Pattern | The layered architecture exhibits distinct trade-offs across critical quality attributes. Security is strengthened through enforced isolation between presentation, logic, and data tiers, facilitating targeted security controls. Explainability benefits positively from the linear processing flow, enhancing traceability of decisions. Conversely, Scalability is significantly constrained by architectural rigidity, impeding independent scaling of components. Adaptability is adversely affected by high inter-layer coupling; modifications to one layer necessitate extensive adjustments to adjacent layers. Performance Efficiency is negatively impacted as sequential processing introduces inherent latency across tiers. Reliability carries substantial risk due to tight dependencies, where failures in foundational layers (e.g., data) can propagate catastrophically through the system. Accuracy maintains a neutral stance, with no explicit architectural advantage or detriment identified. Interoperability demonstrates moderate capability: while well-defined internal interfaces support component interaction, complexity persists in external system integration. Collectively, this profile highlights a design prioritizing security and auditability at the expense of flexibility, performance, and fault tolerance—a characteristic trade-off in rigidly layered systems. |
| Data Pipeline Pattern | This pipeline architecture demonstrates considerable strengths across multiple quality attributes, particularly in processing efficiency and data management. Scalability is robustly supported through horizontal scaling capabilities at individual stages (e.g., ETL). Performance Efficiency benefits significantly from optimized parallel processing architectures. Reliability is enhanced through fault-tolerant mechanisms including retry logic and checkpointing. The design exhibits notable flexibility: Adaptability allows agile modifications to discrete stages, while Interoperability is facilitated by extensive connectors supporting diverse data sources and destinations. Regarding data integrity, Accuracy is strengthened by integrated validation procedures across processing stages, directly enhancing AI readiness. Explainability is significantly advanced through detailed auditing of the end-to-end data flow. However, Security presents a critical concern; the architecture necessitates end-to-end encryption as a non-negotiable requirement to mitigate inherent risks associated with data in motion. Collectively, the profile reveals a highly capable processing framework whose performance and flexibility advantages are counterbalanced by substantial data protection imperatives. |
| Workflow Orchestration Pattern | Orchestrator-driven architectures, such as those leveraging tools like Airflow, demonstrate significant advantages in workflow automation and system coordination. Scalability is robustly facilitated through the orchestrator's capability to automatically distribute and scale discrete tasks. Explainability is markedly enhanced by comprehensive visual monitoring of workflow execution, providing transparent process traceability. Adaptability benefits from dynamic flow reconfiguration capabilities, enabling responsive architectural adjustments. Reliability is strengthened through built-in mechanisms for automatic error recovery and dependency management. Interoperability is effectively supported by the orchestrator's ability to integrate heterogeneous systems into unified pipelines. Notwithstanding these strengths, Security presents inherent complexity: the orchestrator constitutes a critical central point of failure requiring stringent access controls, while secure credential management becomes paramount. Accuracy maintains a neutral assessment, with no inherent architectural bias toward enhancement or degradation of data/output correctness. Performance Efficiency reveals a nuanced trade-off: resource optimization benefits from centralized orchestration are counterbalanced by non-trivial scheduling and coordination overhead. Collectively, this profile highlights an architecture optimized for operational resilience and workflow transparency, albeit with critical centralization risks requiring rigorous security governance. |

### 5.4 Taxonomy of the positive impacts of non-functional requirements on architecture selection for AI-based systems

As a result of the research questions mentioned above, a taxonomy of the positive impacts of non-functional requirements on the selection of architectures for AI-based systems is proposed. For this purpose, [55] carried out by Usman et al. is taken into account, which focuses on a revised method for building taxonomies and is organized into four phases: (i) Planning: This phase is responsible for defining domain objectives and the classification structure, (ii) Identification and Extraction which aims to collect and refine the elements to classify, (iii) Design and Construction which focuses on establishing dimensions, categories and relationships, finally (iv) Validation which is responsible for ensuring consistency, usefulness and clear separation between classes. This taxonomy organizes the NFRs and the software architectures that satisfy them in AI-based systems and the present organization is important because it organizes knowledge, supports decision making, identifies gaps, use of common language and improves traceability with specific architectural decisions. The phases are explained below.

1. **Planning**

    a. **Domain**: NFRs, Architectures and AI.
    b. **Objective**: To relate the most relevant NFRs in AI to the architectures that best support them, based on the questions answered in the previous mapping.
    c. **Structure:** Hierarchical Tree

2. **Identification and Extraction**

    a. **NFRs:** Scalability, Security, Accuracy, Explainability, Adaptability, Reliability, Interoperability, Performance Efficiency.
    b. **Architectures:** Microservices, Lambda, Data Pipeline, Workflow Orchestration, SOA, Layered, Federated Learning.

3. **Design and Construction**

    Relationship between each NFRs and architecture:

    - **Scalability**: Microservices, Lambda, Data Pipeline, Federated Learning, Workflow Orchestration (allow for growth in data volume and users).
    - **Security** (sensitive data): Federated Learning, Layered, SOA (distribute or isolate data to reduce risk).
    - **Accuracy**: Data Pipeline (guarantees data quality and consistency).
    - **Explainability**/**Transparency**: Layered, Workflow Orchestration, SOA (separate responsibilities and make the flow traceable).
    - **Adaptability**: Microservices, SOA, Federated Learning, Data Pipeline, Workflow Orchestration (facilitate system changes and evolution).
    - **Reliability**/**Fault Tolerance**: Microservices, Lambda, Data Pipeline, Workflow Orchestration (manage redundancy and recovery).
    - **Interoperability**: Microservices, SOA, Lambda, Workflow Orchestration (allow the integration of multiple services and technologies).
    - **Performance efficiency**: Lambda, Data Pipeline (process data quickly and in an optimized manner).

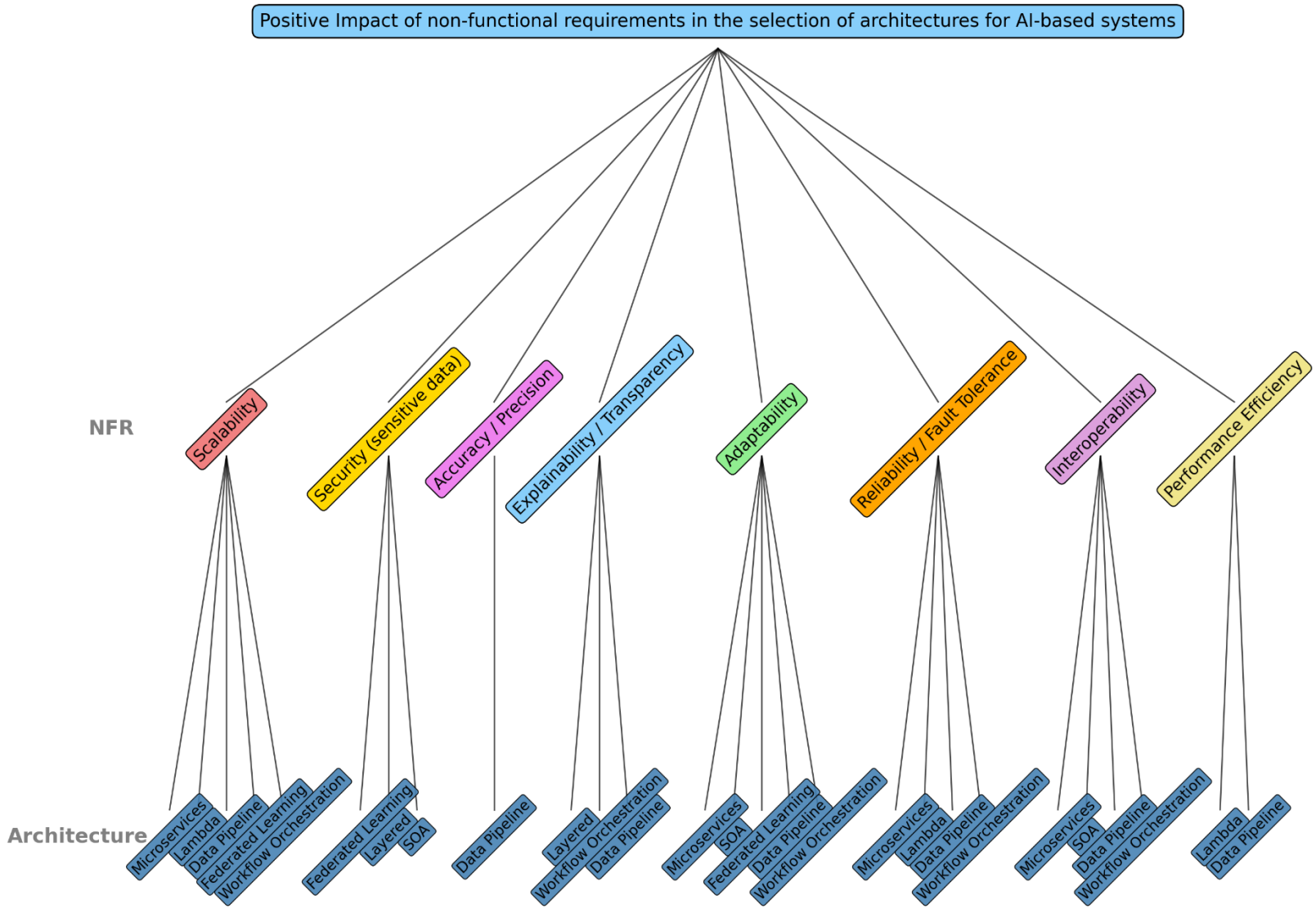



**Figure 1.** Taxonomy of the positive impacts of non-functional requirements on architecture selection for AI-based systems

## 4. Validation

The classification is primarily validated through the input of AI architecture experts, who confirm that the associations between NFRs and architectures reflect real-world industry practices. At the same time, they point out strengths and areas for improvement, given the evolving nature of the field. Therefore, the taxonomy must be continually updated, incorporating new NFRs (such as ethics or sustainability) and emerging architectures.

### Validation of the taxonomy with a questionnaire

The Likert scale is one of the most widely used tools in academia and research because it allows for the quantification of perceptions, expert judgments, and user opinions regarding a subject of study. According to Josho et al. [56], its relevance lies in the fact that the values obtained can be transformed into qualitative ranges such as Strongly Disagree, Disagree, Neutral, Agree, and Strongly Agree, which facilitates the conceptual and statistical interpretation of subjective data. In software engineering, it allows for the validation of proposals or models such as taxonomies, architectural frameworks, or development methodologies through expert opinion.

**Scale:**

1 = Strongly disagree
2 = Disagree
3 = Neutral
4 = Agree
5 = Strongly agree

**Table 9.** Likert scale values

| Average range | Interpretation | Acceptance level |
|---|---|---|
| 1.0 – 1.8 | Very Low | Strongly Disagree / Rejection |
| 1.9 – 2.6 | Low | Disagree / Poor |
| 2.7 – 3.4 | Medium | Neutral / Minimum Acceptable |
| 3.5 – 4.2 | High | Agree / Good |
| 4.3 – 5.0 | Very High | Strongly Agree / Excellent |

**Questions**

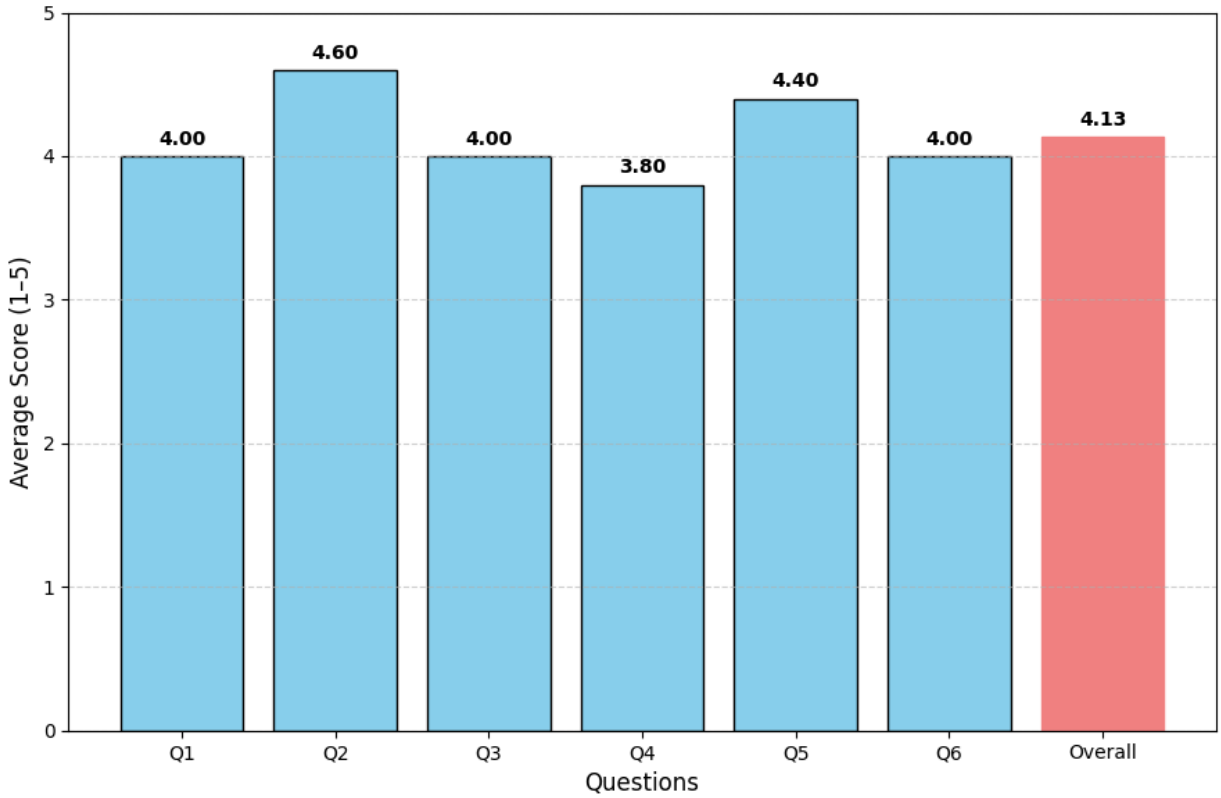
- Is the taxonomy clear and easy to understand?
- Are the included NFRs relevant and sufficient?
- Are the listed architectures relevant and sufficient?
- Is the NFR/Architectures relationship well justified?
- Is the taxonomy useful for supporting architectural decisions in AI-enabled systems?
- Overall, do you consider the taxonomy to be well constructed?
- (Optional) Brief comment: _____

**Results**

**Table 10.** Questionnaire results

| Question | Expert 1 | Expert 2 | Expert 3 | Expert 4 | Expert 5 | Average |
|---|---|---|---|---|---|---|
| 1 | 4 | 5 | 3 | 4 | 4 | **4.0** |
| 2 | 5 | 4 | 5 | 5 | 4 | **4.6** |
| 3 | 4 | 4 | 3 | 5 | 4 | **4.0** |
| 4 | 4 | 4 | 3 | 4 | 4 | **3.8** |
| 5 | 4 | 5 | 4 | 4 | 5 | **4.4** |
| 6 | 4 | 4 | 3 | 5 | 4 | **4.0** |
| **Average** | **4.17** | **4.33** | **3.5** | **4.5** | **4.17** | **4.13** |

**Figure 2.** Average Evaluation per Question of the Taxonomy




Finally, the experts' analysis shows that this taxonomy is, overall, a good proposal that is well-constituted and constructed, obtaining an overall rating of 4.13 out of 5. The taxonomy is especially strong in the relevance and sufficiency of NFRs. It is also considered clear and useful for supporting architectural decisions in AI-based systems. The aspect that requires greater attention is the justification of the relationship between NFRs and architectures. In conclusion, the taxonomy is well evaluated; however, it has clear suggestions from the experts to improve the justification of the relationships between NFRs and architectures, which makes it a good future work because the taxonomy is being updated in iterations due to the change in this technology.

**Suggestions**

- The overall framework is well-structured, but the visualization could be simplified to make it more intuitive.

- The taxonomy is clear, but it would be helpful to add practical examples illustrating how each NFR relates to a specific architecture.

- The NFRs included are relevant, although I notice very general explanations regarding architectures. I recommend adding real-life use cases that support these associations.

## 6 Discussion

The advancement of emerging technologies based on artificial intelligence (AI) poses key challenges, among which is the correct choice of NFRs and software architectures that enable efficient performance aligned with the requirements of project leaders. Key conditions such as project focus; user requirements and current technology market conditions directly influence the decisions that will be made beforehand. When analyzing current approaches, there is a notable preference for certain architectures and attributes that are widely used in modern AI-focused projects in the technology sector.

Among the most visible Nfrs are scalability, security and accuracy, which are critical for programs that operate in real time, such as the Internet of Things (IoT) or machine learning (ML) systems. For example, in high data traffic applications, scalability is a crucial attribute to handle high system demands and implement updates without affecting the user experience, while, in commercial or financial applications, reliability and security are vital to protect sensitive data. However, attributes such as interoperability and efficiency are important, they tend to be less visible in certain scenarios, particularly in those projects where resources do not limit the process or where the system does not integrate with multiple platforms.

Current trends in the technology market have shown that reliability and adaptability are also of greater importance, especially in ML systems, where applications must adapt quickly to large data sets and system demand. On the other hand, each attribute is determined by customer requirements, the development environment and the methodologies used. In continuous development environments, for example, attributes such as performance efficiency may be less of a priority in the face of the need to deliver fast, scalable products.

Regarding the most widely used software architectures in AI-based systems, the most dominant ones are those that facilitate scalability and flexibility. Microservices architecture has established itself as one of the most widely used alternatives thanks to its modularity, which facilitates the implementation and maintenance of complex systems. This architecture allows updating and scaling individual modules without altering the overall operation of the system, which is especially useful in AI and ML-based projects. On the other hand, Lambda Architecture is widely implemented in Big Data and real-time processing projects, as it combines batch and streaming processing, efficiently handling large volumes of data. More classical architectures, such as monolithic systems or SOA, have lost importance in modern applications due to their reduced ability to adapt to new technologies, and their maintenance cost makes them less of a priority in an IA environment. Even so, SOA is still functional in corporate systems that require high interoperability, although its adoption has been largely replaced by architectures that allow handling large databases and modularity in modern applications. The implementation of an architecture is also influenced by aspects such as project complexity, team experience and community support. Although architectures such as Lambda are ideal for real-time analysis, their development can be demanding for teams with limited experience in distributed environments. Similarly, less commonly used patterns, such as Workflow Orchestration and Data Pipeline, are essential in systems where accuracy and traceability are required, such as in the automation of specific processes. The relationship between attributes and architectures is clear once you identify the most widely adopted patterns, such as microservices and Lambda, as they are closely related to attributes such as scalability, security and reliability, which are crucial for modern systems. As technology continues to evolve, these NFRs and architectures are likely to remain dominant, but tailored to the specific demands of growing industries. For example, federated learning systems, which prioritize decentralization and privacy, are already beginning to influence how architectures must adapt to complex scenarios. The design of AI and ML-based systems requires a balance between the choice of attributes and architectures. The most visible attributes, such as scalability and security, along with flexible architectures such as microservices and lambda, reflect current market needs. However, project-specific requirements and technological advancement greatly impact the variety of NFRs and architectures that will be essential for building AI/ML-based systems. NFRs are vital aspects to consider when implementing systems that interact with users. In the HCI domain, these NFRs not only impact system performance, but also significantly

impact the user experience. For example, scalability and interoperability are essential to ensure that the system can be kept up to date and can integrate with other modules. Reliability and security ensure that users have confidence in the system. Explainability and accuracy are crucial in interfaces where system decisions must be easy for users to understand. Likewise, adaptability facilitates experiences aligned to user expectations, while performance efficiency ensures 24/7 operation, which improves the level of user satisfaction. On the other hand, architectures, such as those based on microservices, play a fundamental role in HCI by enabling adaptive and scalable systems that optimize user interaction in applications such as IoT, offering fast responses. Architectural patterns such as Lambda and Data Pipeline help manage data by presenting information efficiently, so that the user can understand it in a better way, facilitating decision making and reducing mental fatigue. The relationship between HCI and architectural patterns focuses on how software architectures improve the experience through user-centered design. Patterns such as Model-View-Controller allow separating the user interface from control logic and data management, simplifying quick adjustments based on user interaction. Architectures such as Federated Learning and Workflow Orchestration in ML systems optimize workflows and protect privacy. Thus, the integration of NFRs and software architectures with HCI will contribute to the creation of robust, easy and user-friendly systems. Since NFRs directly impact architectural decisions more than functional requirements, they also serve as criteria for selecting architectural patterns or styles, as well as technologies. As mentioned above, architectures consider objectives, constraints, and quality attributes. The omission of NFRs makes it impossible to meet stakeholder expectations, so identifying and strategically aligning them with stakeholder needs is a success factor for AI-based systems. Among the most notable results is that architectures such as Microservices and Data Pipeline stand out for maintaining their functionality and quickly recovering from failures, offering high scalability, adaptability, and reliability due to their modular design and parallel processing. Sensitive data is very important in security, therefore, Federated Learning is optimal when dealing with data that must be handled with extreme care. Regarding explainability, the Layered architecture, due to its sequential structure, makes it easy to isolate and audit where decisions are made in the workflow, facilitating explanations by stage. The Workflow Orchestration pattern allows you to record the end-to-end process, logging the inputs and outputs of each step, making the journey and data transparent. Finally, both SOA and Lambda face some performance efficiency and interoperability issues. Regarding SOA, the service bus can cause bottlenecks, and since it handles heterogeneous patterns, the transformations degrade efficiency. Lambda requires different technologies, increasing latency, complicating integration, and also forcing costly transformations. According to the rating given by the panel of experts, the taxonomy proposed has been classified as well-built and structured. Hence, achieving an exceptional rating of 4.13 out of 5. Among all the many factors of this taxonomy, most of the strengths come of the NFRs. Apart from them, many reviewers also noted the taxonomy's simplicity and practicality relevance to architectural articulation on AI-driven systems. Also, the reviewers pointed out an important enhancement: the explanation of NFRs and architectures relationships. This is an important part of the work, and it allows the taxonomy to be periodically modified, which enhances the taxonomy and keeps it responsive to the advancements of AI.

## 7 Conclusions

This systematic mapping allowed us to identify the NFRs and architectures most commonly used in AI- and ML-based projects. As for architectures, we observed a tendency favouring the microservices and Lambda architectures, mainly as a function of their scalability, modularity, and adaptability. These architectures have prevailed due to their capability to manipulate large amounts of data and their compatibility with developing technologies such as machine learning. Despite being less prevalent, other approaches such as the monolithic and SOA architectures remain important in specific contexts.

Regarding the analysis of the impact of NFRs, we observed that scalability, security and accuracy are essential in the majority of consulted works. In ML-based systems, each of these attributes relate to the need to process large volumes of data in real time, ensure data protection, and provide accurate results, respectively. Other quality attributes such as adaptability and reliability also play an important role, especially in systems that seek flexibility towards a changing environment.

For HCI, adaptability is a vital issue because it enables AI-intensive systems to dynamically adjust their behavior, interface, functionality, or context variables and preferences based on user needs, improving usability, satisfaction, and accessibility through more personalized and intuitive interaction.

Finally, our mapping also highlights the implementation of designs for high-interactivity, user-centred systems, where the adoption of HCI principles is crucial. These systems pose additional challenges. Quality attributes such as usability, accessibility and response time are critical, especially in the case of applications for non-technical users or in contexts where user experience affects competitivity. As for future work, it would be interesting to research the

benefits and challenges of the IA powered System and a generalized architectural framework that covers multiple domains and quality requirements that have not been achieved.

NFRs play a fundamental and critical role in the creation of software architectures for artificial intelligence, as they are equally or more decisive than functional ones when choosing or designing the architecture. Their omission leads to architectures that do not meet expectations and also fail to satisfy the real needs of end users, clients, and stakeholders in general. Therefore, their identification and strategic alignment with system objectives and stakeholder needs is crucial, as architectural patterns are specifically chosen or designed to address these NFRs. For example, Microservices is highly influenced by Reliability and Scalability, Workflow Orchestration by Transparency and Explainability, and Federated Learning by Sensitive Data Security. In conclusion, they are selected or adapted specifically as a direct response to priority NFRs.

NFRs define how the system should behave in terms of (performance, security, usability, etc.). As stated above, they are essential to meet the realistic needs and expectations of all stakeholders. Selecting an appropriate architecture is a process intrinsically guided by prioritized NFRs. Different architectural styles and patterns offer distinct advantages and disadvantages in terms of quality, performance, security, or flexibility. Therefore, identifying, prioritizing, and strategically aligning NFRs with system objectives and stakeholder demands is not only a technical step, but a crucial success factor in any complex software project, such as AI-based systems.

In conclusion, the evaluation and expert reinforced the ascribed taxonomy as standing on a solid and well- structured pillars with a commendable score of 4.13 out of 5. Its most notable merit lies in the relevance and sufficiency of the identified Non-Functional Requirements (NFRs) as well as its utility and clarity in assisting architectural decision-making in AI systems. Such a favorable evaluation strengthens the notion that the taxonomy's primary aim is achieved. This, however, is a double- edged sword as the evaluation also outlined what could enhance the work namely, justifying the NFR-architectures more deeply. The taxonomy could certainly benefit, as the experts suggested, from, improving basic visual organization, the provision of interrelated practical examples, and the addition of diverse contextual narratives alongside the core principles. Such focus on deepening the rationale will most greatly advance the taxonomy's development and utilize the feedback's greatest potential. The feedback received readily describes next steps in the taxonomy. The work will concentrate the next iteration on deepening the aspects that have been selected, taking into consideration that the elucidated aspects of the framework will evolve in parallel with technology.

## 8 Acknowledgement and Funding

This paper has been partially supported by the research project 6131: Collaborative Group Decision for knowledge acquisition in agriculture using AI techniques - Contrato de financiamiento de recuperación contingente No. 112721-182- 2023 celebrado entre Fiduciaria Colombiana de Comercio Exterior S.A. Fiducoldex actuando como vocera y administradora del Fondo Nacional de Financiamiento para la Ciencia, la Tecnología y la Innovación, Fondo Francisco José de Caldas y la Universidad del Cauca.

**References**

[1] C. Longoni, A. Bonezzi, and C. K. Morewedge, "Resistance to Medical Artificial Intelligence," *J. Consum. Res.*, vol. 46, no. 4, pp. 629–650, Dec. 2019, doi: 10.1093/jcr/ucz013.

[2] C. Collins, D. Dennehy, K. Conboy, and P. Mikalef, "Artificial intelligence in information systems research: A systematic literature review and research agenda," *Int. J. Inf. Manage.*, vol. 60, p. 102383, Oct. 2021, doi: 10.1016/j.ijinfomgt.2021.102383.

[3] O. S. Lopez and J. A. Hurtado, "IoTAP: Prácticas de diseño de la arquitectura y sus requisitos de calidad en el contexto de desarrollo de software basado en IoT," 2023.

[4] K. Ojo-Gonzalez and B. Bonilla-Morales, "Requerimientos no funcionales para sistemas basados en el Internet de las cosas (IoT): Una revisión," *I+D Tecnológico*, vol. 17, no. 2, 2021, doi: 10.33412/idt.v17.2.3303.

[5] S. Graef and I. Georgievski, "Software Architecture for Next-Generation AI Planning Systems," Feb. 2021, [Online]. Available: http://arxiv.org/abs/2102.10985.

[6] R. Nazir, "Studying Software Architecture Design Challenges, Best Practices and Main Decisions for Machine Learning Systems.," 2021.

[7] E. Bajraktari, T. Krause, and C. Kücherer, "Documentation of Non-Functional Requirements for Systems

with Machine Learning Components," 2024. [Online]. Available: http://ceur-ws.org.
[8] C. A. Collazos, W. J. Sarmiento, A. Solano, and Y. A. Méndez, "Interacción Humano-Computador en la Sociedad Colombiana de Computación," *Rev. Colomb. Comput.*, vol. 21, no. 2, pp. 102–104, Dec. 2020, doi: 10.29375/25392115.4040.
[9] V. Silva-Rodríguez, S. E. Nava-Muñoz, L. A. Castro, F. E. Martínez-Pérez, H. G. Pérez-González, and F. Torres-Reyes, "Classifying design-level requirements using machine learning for a recommender of interaction design patterns," *IET Softw.*, vol. 14, no. 5, pp. 544–552, Oct. 2020, doi: 10.1049/iet-sen.2019.0291.
[10] M. Lorgat, H. Paredes, and M. Gulam, "Handling Non-Functional Requirements in Software Architecture," *Architecture*, pp. 30–31, 2021, [Online]. Available: https://www.researchgate.net/publication/352128522.
[11] A. M. Saghiri, S. M. Vahidipour, M. R. Jabbarpour, M. Sookhak, and A. Forestiero, "A Survey of Artificial Intelligence Challenges: Analyzing the Definitions, Relationships, and Evolutions," *Appl. Sci.*, vol. 12, no. 8, p. 4054, Apr. 2022, doi: 10.3390/app12084054.
[12] K. Ahmad, M. Abdelrazek, C. Arora, M. Bano, and J. Grundy, "Requirements engineering for artificial intelligence systems: A systematic mapping study," *Inf. Softw. Technol.*, vol. 158, no. January, p. 107176, 2023, doi: 10.1016/j.infsof.2023.107176.
[13] M. A. Ali, N. K. Yap, A. A. A. Ghani, H. Zulzalil, N. I. Admodisastro, and A. A. Najafabadi, "A Systematic Mapping of Quality Models for AI Systems, Software and Components," *Appl. Sci.*, vol. 12, no. 17, 2022, doi: 10.3390/app12178700.
[14] U. E. Habiba, M. Haug, J. Bogner, and S. Wagner, *How mature is requirements engineering for AI-based systems? A systematic mapping study on practices, challenges, and future research directions*, vol. 29, no. 4. Springer London, 2024.
[15] K. Kaur and P. Kaur, "The application of AI techniques in requirements classification: a systematic mapping," *Artif. Intell. Rev.*, vol. 57, no. 3, p. 57, Feb. 2024, doi: 10.1007/s10462-023-10667-1.
[16] H. Sofian, N. A. M. Yunus, and R. Ahmad, "Systematic Mapping: Artificial Intelligence Techniques in Software Engineering," *IEEE Access*, vol. 10, pp. 51021–51040, 2022, doi: 10.1109/ACCESS.2022.3174115.
[17] V. De Martino and F. Palomba, "Classification and challenges of non-functional requirements in ML-enabled systems: A systematic literature review," *Inf. Softw. Technol.*, vol. 181, p. 107678, May 2025, doi: 10.1016/j.infsof.2025.107678.
[18] K. B. Ijaz, I. Inayat, and F. Allah Bukhsh, "Non-functional Requirements Prioritization: A Systematic Literature Review," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2019, pp. 379–386, doi: 10.1109/SEAA.2019.00064.
[19] K. Ahmad, M. Abdelrazek, C. Arora, M. Bano, and J. Grundy, "Requirements engineering for artificial intelligence systems: A systematic mapping study," *Inf. Softw. Technol.*, vol. 158, p. 107176, Jun. 2023, doi: 10.1016/j.infsof.2023.107176.
[20] S. Kitchenham, B & Charters, "Guidelines for performing systematic literature reviews in software engineering," *Tech. report, Ver. 2.3 EBSE Tech. Report. EBSE*, vol. 1, no. January 2007, pp. 1–54, 2007, [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.117.471&rep=rep1&type=pdf.
[21] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," 2008, doi: 10.1142/S0218194007003112.
[22] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requir. Eng.*, vol. 11, no. 1, pp. 102–107, Mar. 2006, doi: 10.1007/s00766-005-0021-6.
[23] M. Zarour, A. Abran, J.-M. Desharnais, and A. Abdulrahman, "An investigation into the best practices for the successful design and implementation of lightweight software process assessment methods: A systematic literature review," *J. Syst. Softw.*, 2014, doi: https://doi.org/10.1016/j.jss.2014.11.041.
[24] I. Haro Limiñana, "Inteligencia Artificial Generativa aplicada al diseño de Software," pp. 114–115, 2023.
[25] R. Alonso, R. E. Haber, F. Castaño, and D. Reforgiato Recupero, "Interoperable software platforms for introducing artificial intelligence components in manufacturing: A meta-framework for security and privacy," *Heliyon*, vol. 10, no. 4, p. e26446, Feb. 2024, doi: 10.1016/j.heliyon.2024.e26446.
[26] M. Daole, A. Schiavo, J. L. Corcuera Bárcena, P. Ducange, F. Marcelloni, and A. Renda, "OpenFL-XAI: Federated learning of explainable artificial intelligence models in Python," *SoftwareX*, vol. 23, p. 101505, Jul. 2023, doi: 10.1016/j.softx.2023.101505.
[27] R. Chatterjee, A. Ahmed, and P. R. Anish, "Identification and Classification of Architecturally Significant Functional Requirements," in *Proceedings - 7th International Workshop on Artificial Intelligence and Requirements Engineering, AIRE 2020*, Sep. 2020, pp. 9–17, doi: 10.1109/AIRE51212.2020.00008.

[28] K. M. Habibullah, G. Gay, and J. Horkoff, “Non-functional requirements for machine learning: understanding current use and challenges among practitioners,” *Requir. Eng.*, vol. 28, no. 2, pp. 283–316, Jun. 2023, doi: 10.1007/s00766-022-00395-3.

[29] H. Muccini and K. Vaidhyanathan, “Software Architecture for ML-based Systems: What Exists and What Lies Ahead,” Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.07950.

[30] H.-M. Heyn, E. Knauss, and P. Pelliccione, “A compositional approach to creating architecture frameworks with an application to distributed AI systems,” *J. Syst. Softw.*, vol. 198, p. 111604, Apr. 2023, doi: 10.1016/j.jss.2022.111604.

[31] N. Nascimento, P. Alencar, and D. Cowan, “Artificial Intelligence versus Software Engineers: An Evidence-Based Assessment Focusing on Non-Functional Requirements.” Jul. 06, 2023, doi: 10.21203/rs.3.rs-3126005/v1.

[32] S. E. M. García, C. A. Fernández-y-Fernández, and E. G. R. Pérez, “Classification of Non-functional Requirements Using Convolutional Neural Networks,” *Program. Comput. Softw.*, vol. 49, no. 8, pp. 705–711, Dec. 2023, doi: 10.1134/S0361768823080133.

[33] Q. Lu, L. Zhu, X. Xu, J. Whittle, D. Douglas, and C. Sanderson, “Software Engineering for Responsible AI: An Empirical Study and Operationalised Patterns,” Nov. 2021, doi: 10.1109/ICSE-SEIP55303.2022.9793864.

[34] RITA ROCHA DE SOUSA, “Real-time data analytics for Non-Functional Requirements satisfaction,” 2021, Accessed: Dec. 10, 2024. [Online]. Available: https://recipp.ipp.pt/bitstream/10400.22/20199/1/DM_RitaSousa_2021_MEI.pdf.

[35] R. Subha, A. Haldorai, and A. Ramu, “Artificial Intelligence Model for Software Reusability Prediction System,” *Intell. Autom. Soft Comput.*, vol. 35, no. 3, pp. 2639–2654, 2023, doi: 10.32604/iasc.2023.028153.

[36] V. Liubchenko, “The Requirements Tree Technique for Dependencies-Driven Risk Assessment of AI/ML-based Software Design,” 2022.

[37] S. Park, J. yoon Lee, and J. Lee, “AI system architecture design methodology based on IMO (Input-AI Model-Output) structure for successful AI adoption in organizations,” *Data Knowl. Eng.*, vol. 150, p. 102264, Mar. 2024, doi: 10.1016/j.datak.2023.102264.

[38] A. Vajpayee, R. Mohan, V. Vardhan, and R. Chilukoori, “Building Scalable Data Architectures for Machine Learning,” *Int. J. Comput. Eng. Technol.*, vol. 15, no. 4, pp. 308–320, 2024, doi: 10.5281/zenodo.13234031.

[39] P. Haindl, G. Buchgeher, M. Khan, and B. Moser, “Towards a reference software architecture for human-AI teaming in smart manufacturing,” in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, May 2022, pp. 96–100, doi: 10.1145/3510455.3512788.

[40] S. K. Lo, Q. Lu, L. Zhu, H. Paik, X. Xu, and C. Wang, “Architectural Patterns for the Design of Federated Learning Systems,” Jan. 2021, [Online]. Available: http://arxiv.org/abs/2101.02373.

[41] I. Karabey Aksakalli, T. Çelik, A. B. Can, and B. Tekinerdoğan, “Deployment and communication patterns in microservice architectures: A systematic literature review,” *J. Syst. Softw.*, vol. 180, p. 111014, Oct. 2021, doi: 10.1016/j.jss.2021.111014.

[42] J. M. Alves, L. M. Honorio, and M. A. M. Capretz, “ML4IoT: A Framework to Orchestrate Machine Learning Workflows on Internet of Things Data,” *IEEE Access*, vol. 7, pp. 152953–152967, 2019, doi: 10.1109/ACCESS.2019.2948160.

[43] F. Borelli, G. Biondi, F. Horita, and C. Kamienski, “Architectural Software Patterns for the Development of IoT Smart Applications,” Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.04781.

[44] C. Martín, P. Langendoerfer, P. S. Zarrin, M. Díaz, and B. Rubio, “Kafka-ML: Connecting the data stream with ML/AI frameworks,” *Futur. Gener. Comput. Syst.*, vol. 126, pp. 15–33, Jan. 2022, doi: 10.1016/j.future.2021.07.037.

[45] S. R. O. P. H. W. F. K. Y.-G. G. N. Y. Y. F. J. Romphun Runpakprakun, *Software Engineering Patterns for Machine Learning Applications (SEP4MLA) -Part 3 - Data Processing Architectures*. The Hillside Group, 2021.

[46] I. Altintas *et al.*, “Towards a Dynamic Composability Approach for using Heterogeneous Systems in Remote Sensing,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.06918.

[47] A. Elhabbash, K. Rogoda, and Y. Elkhatib, “MARTIN: An End-to-end Microservice Architecture for Predictive Maintenance in Industry 4.0,” in *2023 IEEE International Conference on Software Services Engineering (SSE)*, Jul. 2023, pp. 10–19, doi: 10.1109/SSE60056.2023.00013.

[48] J. Rane, Ö. Kaya, S. K. Mallick, and N. L. Rane, “Scalable and adaptive deep learning algorithms for large-scale machine learning systems,” in *Future Research Opportunities for Artificial Intelligence in Industry 4.0 and 5.0*, Deep Science Publishing, 2024.

[49] L. M. Tetzlaff, “BPMN4sML: A BPMN Extension for Serverless Machine Learning.,” Aug. 2022, [Online].

Available: http://arxiv.org/abs/2208.02030.
[50] Jeshurun Nishakaran Mills, “Hyper-Automated Artificial Intelligence Platforms for Complex Spatiotemporal Digital Environments,” 2024.
[51] S. Zeb *et al.*, “Industry 5.0 is Coming: A Survey on Intelligent NextG Wireless Networks as Technological Enablers,” May 2022, [Online]. Available: http://arxiv.org/abs/2205.09084.
[52] M. Lorgat, H. Paredes, and M. Gulam, “Proceedings of the 37th International Business Information Management Association (IBIMA),” in *Architecture*, 2021, pp. 30–31, [Online]. Available: https://www.researchgate.net/publication/352128522.
[53] K. M. Habibullah, G. Gay, and J. Horkoff, “Non-functional requirements for machine learning: understanding current use and challenges among practitioners,” *Requir. Eng.*, vol. 28, no. 2, pp. 283–316, Jun. 2023, doi: 10.1007/s00766-022-00395-3.
[54] N. Nascimento, P. Alencar, and D. Cowan, “Artificial Intelligence versus Software Engineers: An Evidence-Based Assessment Focusing on Non-Functional Requirements,” Jul. 2023. doi: 10.21203/rs.3.rs-3126005/v1.
[55] M. Usman, R. Britto, J. Börstler, and E. Mendes, “Taxonomies in software engineering: A Systematic mapping study and a revised taxonomy development method,” *Inf. Softw. Technol.*, vol. 85, pp. 43–59, May 2017, doi: 10.1016/j.infsof.2017.01.006.
[56] A. Joshi, S. Kale, S. Chandel, and D. Pal, “Likert Scale: Explored and Explained,” *Br. J. Appl. Sci. Technol.*, vol. 7, no. 4, pp. 396–403, Jan. 2015, doi: 10.9734/BJAST/2015/14975.