ISSN (Print) 2307-6968, ISSN (Online) 2663-2209
Вчені записки Університету «КРОК» №4 (80), 2025

УДК 004.4:005.8:004.93
*DOI: 10.31732/2663-2209-2025-80-199-211*

*Дата надходження: 21.11.2025*
*Дата прийняття до друку: 15.12.2025*
*Дата публікації: 30.12.2025*

# АРХІТЕКТУРНІ ПІДХОДИ ДО ПОБУДОВИ МАСШТАБОВАНИХ СИСТЕМ ГОЛОСОВОЇ ІДЕНТИФІКАЦІЇ ТА ЇХ ВПЛИВ НА ЕКОНОМІКУ ІТ-ПРОЄКТУ

***Яна Бєлозьорова*[1], *Олег Лукутін*[2]**
[1]*Канд. техн. наук, доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій, Державного університету «Київський авіаційний інститут», Київ, Україна, e-mail: yana.bielozorova@npp.kai.edu.ua, ORCID: https://orcid.org/0000-0002-0688-3436*
[2]*Старший викладач, Університет «КРОК», Київ, Україна, e-mail: oleglv@krok.edu.ua; ORCID: https://orcid.org/0000-0001-6142-2264*

# ARCHITECTURAL APPROACHES TO BUILDING SCALABLE VOICE IDENTIFICATION SYSTEMS AND THEIR IMPACT ON IT PROJECT ECONOMISC

***Yana Bielozorova*[1], *Oleh Lukutin*[2]**
[1]*Candidate of Technical Sciences, Associate Professor of the Software Engineering Department, Faculty of Computer Science and Technologies, State University "Kyiv Aviation University", Kyiv, Ukraine, e-mail: yana.bielozorova@npp.kai.edu.ua, ORCID: https://orcid.org/0000-0002-0688-3436*
[2]*Senior Lecturer, University KROK, Kyiv, Ukraine, e-mail: oleglv@krok.edu.ua; ORCID: https://orcid.org/0000-0001-6142-2264*

***Анотація.*** *У статті досліджуються архітектурні підходи до побудови масштабованих систем голосової ідентифікації та їхній вплив на економіку ІТ-проєктів у сфері менеджменту та розробки програмного забезпечення. Актуальність теми зумовлена стрімким зростанням ринку голосових сервісів і потребою бізнесу одночасно забезпечувати високу точність біометричної аутентифікації, дотримання SLA щодо затримок та доступності, а також контроль сукупної вартості володіння (TCO). Теоретичною основою дослідження є сучасні роботи з побудови систем голосової біометрії, мікросервісних та хмарно-нативних архітектур, а також підходи до управління витратами CAPEX/OPEX для хмарних ІТ-рішень. Метою статті є розроблення концептуальної моделі зіставлення архітектурних рішень (монолітна vs мікросервісна архітектура, on-premise vs хмарне розгортання, використання сучасних ML-фреймворків) з ключовими економічними показниками ІТ-проєкту (CAPEX, OPEX, TCO, економія від масштабування, ризики недотримання SLA). Методологія дослідження ґрунтується на системному аналізі наукових джерел, порівняльному аналізі архітектурних альтернатив, формалізації економічних залежностей та побудові узагальненої моделі витрат для різних сценаріїв розгортання систем голосової ідентифікації. Отримані результати включають: структурну модель типової системи голосової ідентифікації; аналітичне порівняння монолітної та мікросервісної архітектур з погляду впливу на CAPEX/OPEX; інтегровану економічну модель оцінки TCO для чотирьох варіантів розгортання: моноліт on-premise, мікросервіси on-premise, моноліт у хмарі (IaaS) та хмарно-нативна/serverless-архітектура. Було показано, що перехід до мікросервісної хмарно-нативної архітектури зменшує частку CAPEX і переводить витрати в більш керований OPEX.*

***Ключові слова:*** *голосова ідентифікація, мікросервісна архітектура, хмарні обчислення, CAPEX, OPEX, SLA, TCO, ML-фреймворки*
***Формул: 3; рис.: 4; табл.: 1; бібл.: 14***

***Abstract.*** *The article examines architectural approaches to building scalable voice identification systems and their impact on the economics of IT projects in the field of software engineering and project management. The relevance of the topic is driven by the rapid growth of the voice services market and the need for businesses to simultaneously ensure high accuracy of biometric authentication, compliance with SLA requirements for latency and availability, as well as control of the total cost of ownership (TCO). The theoretical foundation of the study includes modern research on voice biometrics systems, microservice and cloud-native architectures, as well as cost management approaches for CAPEX/OPEX in cloud IT solutions. The aim of the article is to develop a conceptual model for correlating architectural decisions (monolithic vs. microservice architecture, on-premise vs. cloud deployment, use of modern ML frameworks)* *with key economic indicators of an IT project (CAPEX, OPEX, TCO, scaling-related savings, SLA non-compliance risks). The research methodology is based on a systematic analysis of scientific sources, comparative architectural analysis, formalization of economic dependencies, and constructing a generalized cost model for various deployment scenarios of voice identification systems. The obtained results include: a structural model of a typical voice identification system; an analytical comparison of monolithic and microservice architectures in terms of their impact on CAPEX/OPEX; an integrated economic model for evaluating TCO across four deployment options: monolithic on-premise, microservices on-premise, monolithic in the cloud (IaaS), and cloud-native/serverless architecture. It has been shown that transitioning to a microservice cloud-native architecture reduces the share of CAPEX and shifts costs toward a more controllable OPEX structure.*

***Keywords:*** *voice identification, microservice architecture, cloud computing, CAPEX, OPEX, SLA, TCO, ML frameworks*

***Equations: 3; fig.: 4; tab.: 1; bibl.: 14***

**Introduction.** Voice identification (speaker/voice identification) is gradually transforming from a field of scientific experimentation into large-scale industrial solutions, covering bank contact centers, voice bots, mobile applications, and government services. Such high-critical systems operate in near real-time mode, require support for tens or hundreds of thousands of parallel sessions, and must ensure high recognition accuracy together with the protection of personal data.

Ensuring these high requirements for accuracy, availability, and large-scale real-time data processing is inseparable from economic efficiency and the need to control the total cost of ownership (TCO). The choice between upfront capital expenditures (CAPEX) for on-premise infrastructure and flexible operational expenditures (OPEX) in the cloud becomes a key strategic dilemma for IT project leaders. This factor is especially relevant against the backdrop of the rapid evolution of computing architectures: there is a clear shift from monolithic applications deployed in private data centers to microservice-based, cloud-native, and serverless solutions.

At the same time, computing architectures are evolving rapidly: there is a shift from monolithic applications deployed in private data centers toward microservice-based, cloud-native, and serverless solutions. These modern architectures are widely used to build scalable systems for processing speech signals.

International evaluation programs for speaker recognition technologies demonstrate rapid improvements in accuracy due to the use of deep neural networks and distributed data-processing architectures (Greenberg et al., 2020). However, as practice in IT project management shows, architectural choices are often made intuitively or based solely on technical arguments, without a sufficiently formalized analysis of their impact on CAPEX/OPEX, TCO, SLA, and risk indicators.

For software development managers and IT economists, a key question arises: how do architectural decisions transform into cash flows, risks, and the long-term value of a project?

Thus, we face a scientific and practical problem: the lack of an integrated model that connects architectural approaches to designing scalable voice identification systems with the economic aspects of an IT project throughout its entire life cycle. The present study addresses this gap.

**Literature Review.** Modern research in the field of voice biometrics demonstrates a shift toward neural network – based models (x-vector, ResNet, transformers) and complex speech-processing pipelines that require scalable computational platforms (Liu et al., 2019; Sadjadi et al., 2020; Greenberg et al., 2020). These works focus primarily on algorithmic accuracy, yet they indirectly confirm the need for flexible infrastructure solutions capable of handling large volumes of audio data.

A number of publications and industry reports describe architectures of scalable voice and multimodal systems using microservices, message queues, container orchestration, and cloud services. Practical guides for building voice-AI systems emphasize the advantages of decomposing the solution into services for audio acquisition, pre-processing, feature extraction, model inference, logging, and analytics. This approach enables independent scaling of system bottlenecks, and more efficient management of latency and SLA compliance. Microservice architecture is generally viewed as a key pattern for building flexible and fault-tolerant systems.

Studies on the transformation of enterprise systems (Lee et al., 2024) show that the transition to MSA (microservice architecture) is closely linked to the adoption of cloud-based consumption models and a shift in the structure of CAPEX/OPEX expenditures. Analytical and practical materials from leading cloud providers (AWS, Microsoft, Google) focus on comparing CAPEX and OPEX models, cost-optimization strategies, and the design of economically efficient architectures, including serverless approaches.

However, ensuring these high requirements for accuracy, availability, and real-time processing of large volumes of data is inseparably linked to economic efficiency and the control of total cost of ownership (TCO). The choice between upfront capital expenditures (CAPEX) for owned infrastructure and flexible operational expenses (OPEX) in the cloud becomes a key strategic dilemma for IT project leaders. This factor is especially relevant against the backdrop of the rapid evolution of computing architectures: we are witnessing a shift from monolithic applications deployed in private data centers to microservice-based, cloud-native, and serverless solutions.

Several works examine the choice of ML frameworks (TensorFlow, PyTorch) as a fundamental factor in building production-grade ML systems, highlighting their evolution into end-to-end platforms oriented not only toward research but also toward industrial deployment.

**Research Objectives and Hypotheses.** To develop a conceptual model that enables aligning architectural approaches to designing scalable voice identification systems with the economics of an IT project throughout its lifecycle.

To achieve this aim, the following objectives were formulated:

1. Describe the typical architecture of a voice identification system, highlighting key services and scalability points.
2. Compare monolithic and microservice approaches for such systems, taking into account both on-premise and cloud deployment options.
3. Formalize the relationship between architectural decisions and the project’s economic indicators: CAPEX, OPEX, TCO, SLA, and associated risks.
4. Propose recommendations for IT project managers regarding the choice of architecture depending on the lifecycle stage of the voice identification system and specific business requirements.

**Methods.** The study is based on an integrated application of several scientific methods:

1. Systematic and comparative literature analysis – applied to critically evaluate existing achievements in the fields of voice biometrics and architectural patterns (microservices, cloud-native).
2. Scenario method – used to compare four key architectural deployment options (S1-S4) and assess their economic implications.
3. Formalization of economic dependencies – implemented in the form of analytical relationships, including formula (1) for TCO and formula (2) for system availability.
4. Expert risk structuring – applied to qualitatively assess typical risks associated with each architectural scenario.

**Results.**

***Structural Model of a Voice Identification System***

A generalized logical architecture of a production-ready voice identification system has been designed as part of this study. Such a system typically consists of several functionally distinct logical components:

- Audio Ingestion Service – responsible for receiving audio streams or files from client applications or communication channels (Technologies: REST API, WebSocket, VoIP protocols (SIP, RTP). Typical requirements: support for 5,000-20,000 concurrent connections for enterprise systems).

- Pre-processing Service – performs normalization, noise reduction, and voice activity detection (VAD) (Technologies: WebRTC Audio Processing, librosa, scipy. Typical characteristics: 50-100 ms latency, support for multithreaded processing).

- Feature Extraction Service – generates spectrograms, speaker embeddings or other acoustic feature representations (Technologies: TensorFlow, PyTorch, ONNX Runtime; models such as x-vector, ECAPA-TDNN, ResNet. Typical characteristics: 100-300 ms latency per audio sample (3-10 seconds), requires a GPU for real-time operation. Computational intensity: 2-5 GFLOPS per embedding).

- ML-based Identification Service – compares extracted embeddings with a reference database and produces similarity scores or identification decisions (Technologies: Cosine similarity, PLDA, vector databases (Milvus, Pinecone, FAISS). Typical characteristics: 10-50 ms latency for search across 1M profiles, horizontally scalable).

- User Profile Management Service – stores, updates and handles biometric profiles of users, including enrollment and verification metadata (Technologies: PostgreSQL, MongoDB for metadata; S3-compatible storage for embeddings. Typical characteristics: CRUD operations with latency <100 ms).

- Storage and Analytics Service – provides persistent storage of audio artifacts, processing logs and analytical data for monitoring, auditing and model improvement (Technologies: Prometheus, Grafana, ELK Stack, CloudWatch. Typical characteristics: near real-time aggregation, 30-90 days retention policy).

For the practical implementation of the proposed model, a rational technological stack has been identified:

- Audio Preprocessing: WebRTC Audio Processing, librosa (for normalization and VAD).

- Feature Extraction: ONNX Runtime, x-vector or ECAPA-TDNN models (computational intensity 2-5 GFLOPS per embedding).

- Identification Service: Vector databases (Milvus, FAISS) for search with latency < 50 ms.

- Integration Layer: API Gateway for request routing and load balancing. Message Queue (Kafka, RabbitMQ, AWS SQS) for asynchronous processing. Service Mesh (Istio, Linkerd) for managing inter-service communication in Kubernetes.

General logical diagram of a microservice-based voice identification system are illustrated in detail in Fig. 1.

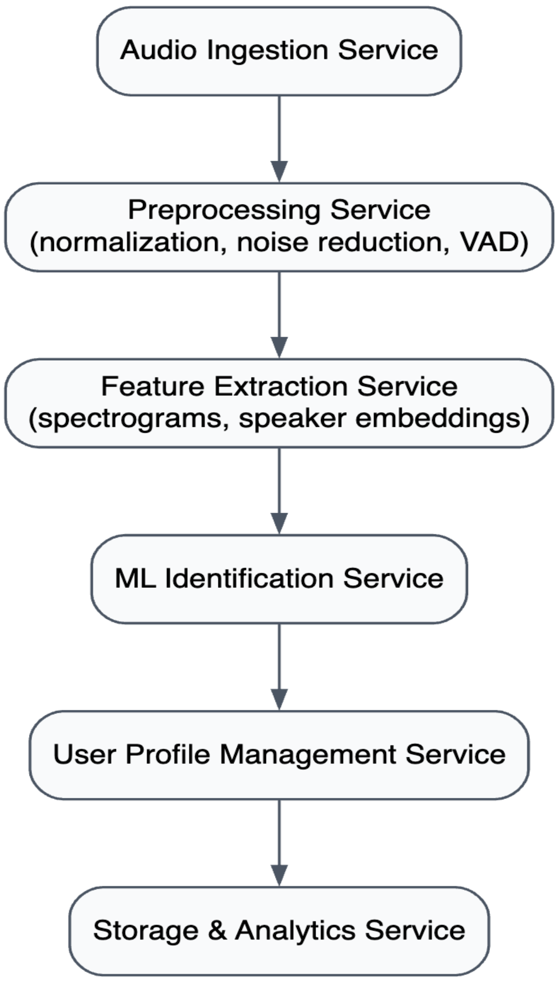
Audio Ingestion Service
Preprocessing Service (normalization, noise reduction, VAD)
Feature Extraction Service (spectrograms, speaker embeddings)
ML Identification Service
User Profile Management Service
Storage & Analytics Service

**Fig. 1. Generalized logical diagram of a microservice-based voice identification system**
*Source: developed by the Author*

***Analysis of Architectural Solutions and the Economic Perspective.*** Based on the results of the comparative analysis, it was established that the microservice approach enables independent scaling of the most computationally intensive stages, such as feature extraction and model inference. This makes it possible to use different technology stacks and update individual services without causing downtime of the entire system. To gain a more complete understanding of the nature of these differences, it is useful to examine the technical characteristics of each approach individually.

A monolithic architecture is characterized by a unified structure of the software system, in which all components operate within a single process and are deployed as one artifact. At the early stages of product development, this model provides relative simplicity in development and deployment, since there is no need to coordinate interactions between numerous services. Testing is also less labor-intensive because all parts of the system reside within a shared execution environment, which simplifies scenario reproduction and error localization. Operational complexity remains minimal, as engineering teams work with only one executable module. An additional technical advantage is the low latency of inter-component calls: interactions occur at the level of internal functions, without network overhead.

Despite these advantages, this architecture has significant limitations that become increasingly noticeable as the system grows. Scaling typically involves duplicating the entire system regardless of which specific components are under the most load, which leads to inefficient resource consumption. As the product grows in size and the number of interdependencies increases, making changes becomes more difficult: tight coupling between components raises the risk of regression errors and slows down development. Technological flexibility is also limited, as the entire project is usually tied to a single technology stack. In addition, any failure within the monolith may trigger a cascading effect and halt the operation of the entire system.

In response to these challenges, an alternative approach has emerged – microservice architecture, which offers a different way of organizing a system.

Microservice architecture involves decomposing the system into separate services, each of which performs a limited set of functions and can be deployed independently. This approach makes it possible to scale only those components that require additional resources: for example, the feature-processing module can be deployed on GPU instances, while classification modules may run on CPU-optimized infrastructure.

The use of different programming languages and technology stacks allows each service to be optimized according to its functional and performance requirements. Independent development and deployment cycles increase team productivity, as work on changes occurs in parallel and without the need to coordinate global releases.

Service isolation improves the system’s fault tolerance: a local failure is less likely to affect the system’s overall operability. In addition, a microservice architecture simplifies meeting required SLAs, as engineers can focus on optimizing precisely those components that generate the highest load.

Despite these advantages, this architecture is not without significant limitations that become increasingly apparent as the system grows.

Scaling is typically performed by replicating the entire system, regardless of which specific components are under heavy load, which leads to inefficient resource usage. As the product and the number of interdependencies grow, making changes becomes increasingly difficult: tight coupling between components raises the risk of regression errors and slows development. Technological flexibility is also limited, as the entire project is usually tied to a single technology stack.

In addition, any failure within the monolith can trigger a cascading effect and bring down the entire system.

Thus, the technical analysis shows that both approaches have their strengths and specific limitations, and therefore the choice of architecture must take into account the context of the project. A monolithic architecture is suitable for small projects or early stages of development, while microservices demonstrate their advantages in complex, dynamic, and scalable systems – but at the cost of significantly higher design and operational complexity.

Alongside the technical characteristics, the economic consequences of adopting one architecture or another also play a decisive role and require separate analysis.

The economic comparison between monolithic and microservice architectures reveals significant differences in both capital expenditures (CAPEX) and operational expenditures (OPEX). Moreover, the chosen infrastructure approach – on-premise deployment or the use of cloud platforms – plays a key role in this dichotomy.

First of all, it is necessary to analyze the capital expenditures, which heavily depend on the deployment model of the system. In the case of a monolithic architecture deployed in an on-premise environment, the initial capital expenses remain high due to the need to invest in server and network equipment, storage and backup systems, as well as in physical infrastructure provisioning. For enterprise-level systems, these costs may range from $150-300 thousand.

At the same time, adopting a microservice model in the same on-premise environment requires even higher investments. This is caused by the need to create an orchestration cluster, such as Kubernetes, as well as distributed processing, redundancy, and high availability mechanisms, which raises CAPEX to the level of $200-400 thousand.

Transitioning to cloud models significantly changes the structure of capital expenditures. A monolithic architecture deployed in the cloud under the IaaS model substantially reduces initial investments, since costs are focused mainly on system development and migration, and typically do not exceed $20-50 thousand.

A microservice architecture adapted to the cloud environment – especially in serverless or cloud-native form – demonstrates an even lower CAPEX level. Thanks to the absence of the need to maintain in-house infrastructure, the initial costs may amount to only $10-30 thousand, focusing primarily on developing service logic and configuring automatic scaling mechanisms.

Regarding operational costs, a monolithic system in a traditional on-premises deployment creates a moderate load on the budget.

The main expenses include electricity consumption, hardware maintenance, salaries for system administrators and DevOps specialists, as well as the purchase of licensed software.

The total annual costs may range from $40,000 to $70,000.

Microservices in an on-premises environment require a higher level of operational management and more highly qualified personnel, which increases OPEX to $60,000-$100,000 per year.

A similar dependency is observed in cloud models, where the nature of the workload becomes the key factor shaping the costs. A monolithic system in a cloud environment requires paying for virtual machine usage, data storage, and network traffic, which makes operating expenses sensitive to the workload level. Under average load conditions, annual costs may reach $50-90 thousand.

A microservices-based cloud architecture demonstrates the highest operating expenses under stable load, which is related to the continuous activation of a large number of services and infrastructure components. However, due to the ability to autoscale, this model becomes the most efficient in cases of variable or peak workloads. In such scenarios, annual costs range from $60-120 thousand, depending on usage patterns.

Overall, the key conclusion is that while a microservice architecture increases operational management complexity and raises OPEX, it simultaneously provides flexibility and fine-grained cost control aligned with real workload patterns. In a cloud environment, this approach minimizes capital expenditures, but it requires mature FinOps practices to effectively monitor and optimize operational costs.

The choice between local (on-premise) and cloud deployment in voice identification systems is determined by a combination of technical, regulatory, and economic factors that significantly influence both the architecture of the solution and its operational model. Systems of this class often operate in strictly regulated environments, are characterized by increased performance requirements, and are highly sensitive to latency, which makes the process of selecting the infrastructure complex and multifactorial.

The analysis of the criteria that shape the decision in favor of on-premise or cloud deployment shows that this choice largely depends on usage conditions, the nature of the workload, the availability of infrastructure, and regulatory requirements.

In cases where strict regulatory compliance, minimal latency, or full control over the computing infrastructure are top priorities, the on-premise model is generally preferred. For example, in industries with strict data residency requirements – such as the financial sector under PCI DSS Level 1 standards – local infrastructure is practically mandatory. In real-time scenarios, such as integration with telephony platforms, even a small additional network delay caused by routing traffic to the cloud can be critical.

This approach is also economically justified for systems with a high and stable workload, where infrastructure utilization exceeding 80% throughout the day makes on-premise deployment more cost-effective in the long run. Finally, the presence of an in-house data center with reserved capacity lowers the entry barrier and makes the on-premise solution organizationally convenient.

Cloud deployment is favored in a number of scenarios. Cloud platforms significantly reduce the product's time-to-market, since there is no need for lengthy procurement, installation, and configuration cycles for hardware. For systems with variable or peak load patterns – typical for many voice identification services (e.g., increased activity during daytime or at the end of the month) – cloud infrastructure provides automatic scaling without the need to maintain excessive on-premise capacity.

Additionally, for applications with global reach, the ability to deploy in multiple geographic regions helps reduce latency and improve the quality of user interaction. Cloud platforms also offer a wide range of managed services, such as AWS Transcribe or Azure Speaker Recognition, the use of which substantially reduces development costs and accelerates integration of complex components.

Thus, the choice between on-premise and cloud approaches in the context of voice identification depends on a combination of technical requirements, regulatory constraints, and economic priorities.

On-premise deployments remain appropriate in highly regulated or low-latency environments, while cloud models demonstrate advantages under dynamic workloads, the need for rapid scaling, and the use of specialized platform services.

After examining the technical, organizational, and infrastructural factors, it becomes necessary to move from a qualitative comparison to a formalized economic assessment that allows for an objective evaluation of the alternatives.

Integrated Economic Model for TCO Evaluation

To assess the Total Cost of Ownership (TCO) over a planning horizon of $H$ years, a discounted financial model is used:

$$TCO = CAPEX + \sum_{t=1..H}\left[\frac{OPEX_t}{(1+r)^t}\right] \quad (1),$$

where $r$ is discount rate (typically 8-12% for IT projects), and $OPEX_t$ represents the operating costs in year $t$, CAPEX – capital expenditures at the start of the project (equipment, licenses, initial development), $H$ – planning horizon (typically 3-5 years).

The critical importance of using discounting $(1+r)^t$ lies in the fact that it makes it possible to compare current investments (high CAPEX in on-premise scenarios S1 and S2) with future, time-distributed operational expenses (dominant OPEX in cloud scenarios S3 and S4). This ensures that the comparison of architectures (for example, between S1 and S4) is based on the present net value, which reflects the time value of money and the long-term financial impact of architectural decisions.

However, for practical decision-making regarding the choice of a deployment environment, an integral TCO assessment alone is not sufficient. In a number of scenarios, the critical factor is determining the moment when one operating model becomes more economically advantageous than another. For this purpose, an additional tool is used – break-even analysis.

$$\text{Break} - \text{even point: CAPEX\_on\_prem} + N \times \text{OPEX\_monthly\_on\_prem} = \text{CAPEX\_cloud} + N \times \text{OPEX\_monthly\_cloud} \quad (2),$$

where $\boldsymbol{N}$ is number of months of operation.

Solving equation (2) with respect to $N$ yields the break-even point, meaning the moment in time when the total costs of the on-premise deployment become lower (or higher) than the costs of the cloud model.

Break-even analysis makes it possible to determine over what period of operation investments in one's own infrastructure are justified, and in which cases the cloud infrastructure – with its variable cost structure – becomes more economically viable.

In on-premise scenarios (S1, S2), CAPEX is a significant upfront cost at the launch stage, whereas in cloud scenarios (S3, S4), CAPEX is minimal, and the majority of expenses are represented by the discounted $OPEX_t$.

To assess the availability of a multi-component microservice system (S3, S4), the following approximation is applied:

$$A_{system} \approx \Pi_{i=1..n} A_i \qquad (3),$$

where $A_i$ is the availability of an individual critical service, and $n$ is the number of critical services in the processing chain.

This formula reflects the cumulative failure effect for a sequential architecture, as shown in Fig. 1, where the failure of any one of the $n$ critical components leads to the failure of the entire system.

Thus, even if the availability of each individual service $A_i$ is high (for example, 99.9%), the overall system availability $A_{system}$ will decrease as $n$ increases.

Let us consider four basic scenarios:

-S1: Monolithic on-premise.

-S2: Microservices on-premise.

-S3: Monolithic in the cloud (IaaS, VM-based).

-S4: Cloud-native microservices/serverless.

This is a key argument in favor of a microservice cloud-native architecture (S4), which can leverage patterns of independent fault tolerance, fault isolation, and automatic self-healing (health checks) to minimize the impact of n on overall system availability $A_{system}$ and ensure high SLA requirements.

Validation of the model was carried out using a computational example of a system with the following parameters:

- Number of active users: 100,000.
- Number of identification requests: 5,000,000/month (average).
- Peak load: 300 requests/second (2× the average).
- Average audio duration: 5 seconds.
- SLA requirements: 99.9% availability, <500 ms end-to-end latency.
- Planning horizon: 5 years.
- Discount rate: 10% annually.

Detailed comparison of TCO for the four scenarios is presented in Fig. 2. The chart in this figure illustrates the dynamics of TCO growth for each scenario, allowing the identification of crossover points and the assessment of the long-term financial impact of high CAPEX (Scenarios S1 and S2) compared with the flexible OPEX-driven growth model (Scenarios S3 and S4).

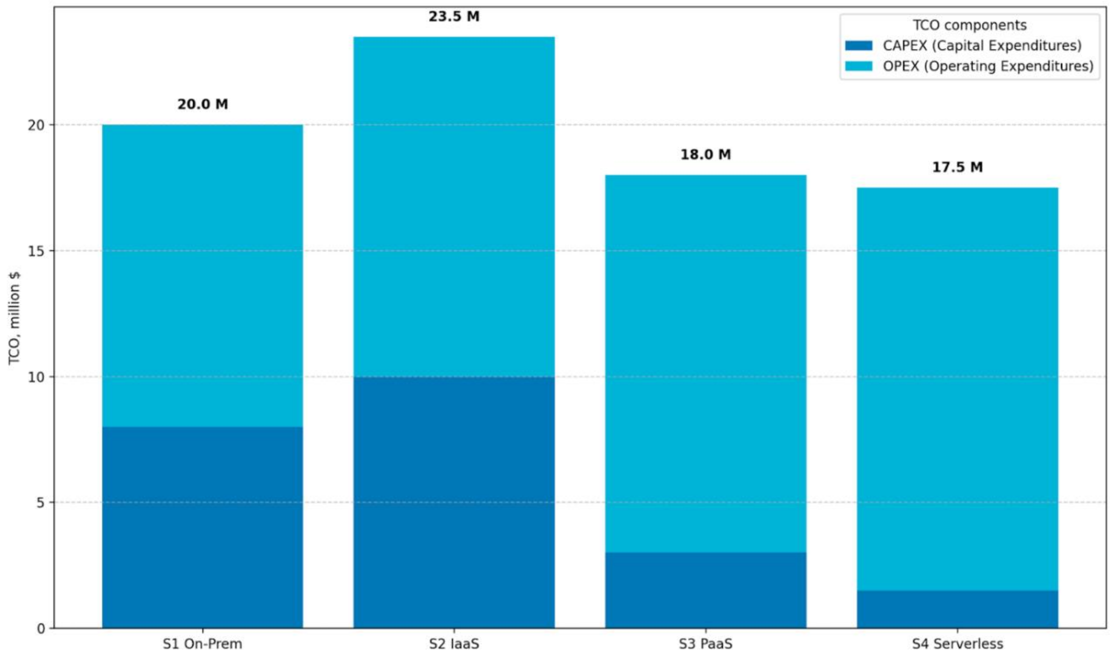

**Figure 2. Cumulative TCO Dynamics for S1–S4 Over a 5-Year Planning Horizon**

*Source: developed by the Author*

Figure 3 presents the internal structure of TCO for each architectural solution, detailing the cost components: capital expenditures (CAPEX), operational expenditures for system operation, and personnel-related expenses. The analysis allows the identification of the main financial drivers of each scenario and correlates them with the qualitative characteristics summarized in Table 1.

In addition to the integrated economic model for TCO evaluation, Table 1 provides a detailed comparative description of the four key architectural scenarios (S1-S4) for speaker-recognition systems. The table summarizes and compares critical factors such as the structure of capital (CAPEX) and operational (OPEX) costs, scalability levels, management complexity, and typical risks inherent to each architectural approach within the lifecycle of an IT project.

From an economic perspective, each of the four architectural alternatives (S1-S4) has a clearly defined cost structure:

-Monolithic on-premise (S1): Characterized by substantial initial capital expenditures (CAPEX), followed by relatively stable but high ongoing operational expenditures (OPEX).

-Cloud-native microservices/serverless (S4): Imply minimization of CAPEX by shifting most costs into a flexible OPEX model that is directly tied to the actual system load.

*Table 1*

**Comparison of architectural scenarios for voice identification systems**

| Scenario | Cost Structure | Scalability | Management Complexity | Typical Risks |
|---|---|---|---|---|
| **S1: Monolithic on-premise** | High CAPEX, moderate OPEX | Limited | Relatively low at the initial stage | Technological obsolescence |
| **S2: Microservices on-premise** | High CAPEX + orchestration costs | High | High | Operational complexity |
| **S3: Monolithic in the cloud** | Low CAPEX, increasing OPEX | Better than S1 | Medium | “Lift-and-shift” inefficiencies (migrating an inefficient architecture to the cloud) |
| **S4: Cloud-native microservices/serverless** | Minimal CAPEX | Very high | High at the design stage | Vendor lock-in (dependence on cloud provider) |

*Source: developed by the Author*

It is important to note that the high Operational complexity in Scenario S2 (On-premise Microservices) is a key factor influencing OPEX. It includes additional expenditures for maintaining and managing the container orchestration infrastructure (e.g., Kubernetes), complex network configuration, monitoring of distributed transactions, and the need to employ highly qualified DevOps personnel. All of this leads to a significant increase in operational costs compared to the monolithic solution (S1).

While Figure 2 visualizes the quantitative evolution of total cost of ownership (TCO) over time, Table 1 complements this analysis with a qualitative assessment and structured comparison of these scenarios based on several essential non-financial and operational parameters.

The table is structured to cover the following aspects necessary for making an informed architectural decision:

1.Cost Structure (CAPEX and OPEX): A detailed breakdown of capital (CAPEX) and operational (OPEX) components is provided for each scenario, with emphasis on the distribution of investments (e.g., hardware, licenses, cloud services, personnel).

2.Operational Efficiency and Scalability: An evaluation of management complexity and system flexibility, as well as the architecture's capability for horizontal and vertical scaling in response to changing workload and business requirements.

3.Typical Risks: A summary of the main technical, organizational, and financial risks associated with each architectural approach (e.g., vendor lock-in, integration challenges, infrastructure dependencies).

Thus, Table 1 makes it possible not only to assess which scenario is most economically advantageous in the long term, but also to understand its operational limitations and potential risks, providing a comprehensive basis for choosing the optimal deployment strategy for the system.

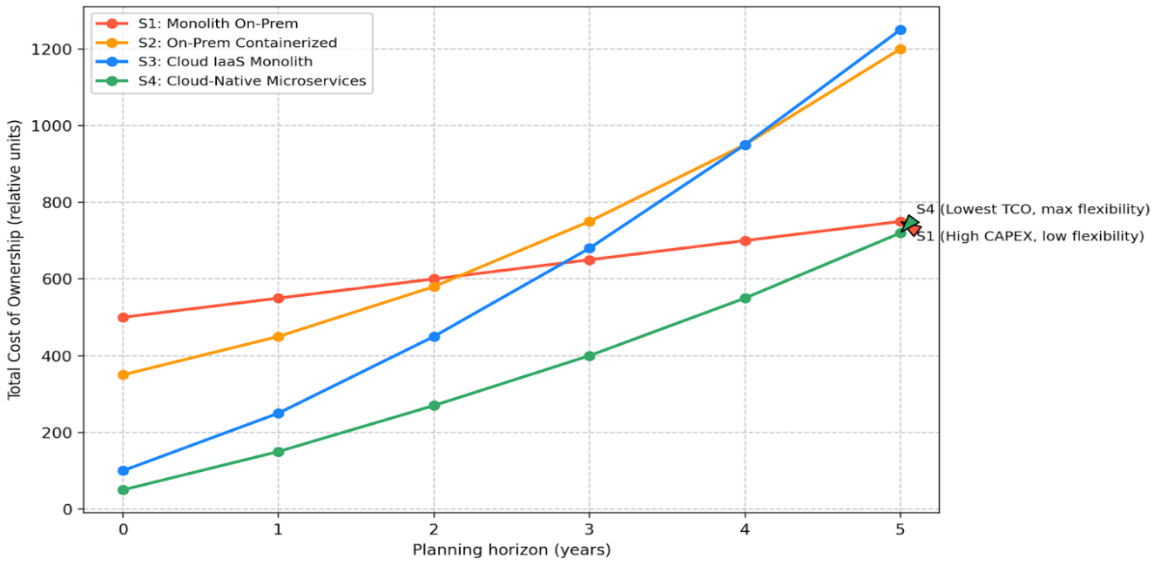

**Figure 3. TCO Cost Structure for Scenarios S1-S4 Over a 5-Year Planning Horizon**

*Source: developed by the Author*

**Discussion.** Architecture of a voice identification system directly determines the economics of an IT project.

The conducted research confirms that monolithic on-premise solutions (S1) are characterized by high CAPEX and limited scalability, which makes them risky in cases of rapidly increasing workloads.

In contrast, cloud-native microservices (S4) shift expenditures into flexible OPEX, allowing costs to be aligned much more accurately with actual system load.

It was found that the microservice approach (S2, S4) is naturally aligned with the high scalability and SLA requirements typical for voice identification systems.

The break-even analysis shows that cloud deployment (S4) remains more cost-effective than on-premise infrastructure (S1) at workloads up to 6 million requests per month. Once this threshold is exceeded (>6M/month), the fixed costs of on-premise solutions become more advantageous than the linearly increasing OPEX of cloud services. This indicates the need to reassess the architectural strategy at the product maturity stage.

It enables isolation and independent scaling of the ML component, simplifying the use of specialized instances (GPU/TPU).

The choice between on-premise and cloud deployment should be considered not as a technical decision, but as a strategic management decision based on the analysis of TCO over a multi-year horizon.

Despite the high complexity at the design stage, cloud-native architecture provides the highest economic flexibility and the highest level of system availability $A_{system}$, which is critical for enterprise applications.

It is important to consider the exponential increase in the cost of meeting SLA requirements. The calculations show the impact of availability requirements on OPEX:

- 99.5%: Baseline level (cost multiplier 1.0x).
- 99.9%: Requires Multi-AZ (cost multiplier 1.3x).
- 99.99%: Requires multi-region active-active architecture and 24/7 support, which increases OPEX by 2.5 × compared to the baseline level.

In addition, the selection of ML frameworks (TensorFlow, PyTorch) and the ecosystem of cloud services shapes a new cost landscape, influencing time-to-market and infrastructure expenses-factors also incorporated into the integrated model.

The stages of the architectural evolution of a voice identification system from MVP to enterprise-grade are shown in Fig. 4.

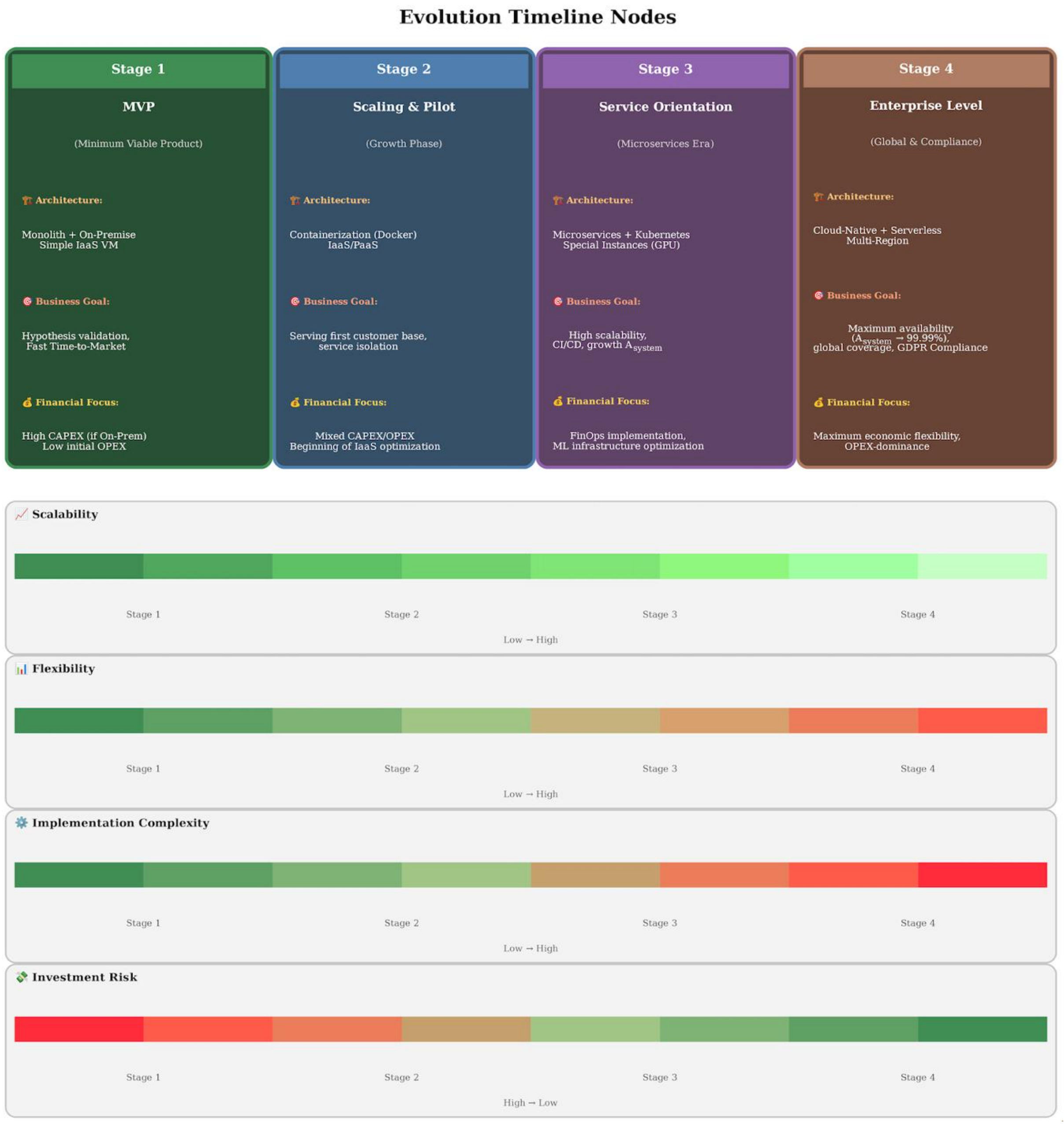

**Fig. 4. Stages of architectural evolution of a voice identification system from MVP to enterprise level**

*Source: developed by the Author*

**Conclusions.** According to the results of the study, the stated goal was achieved and all research objectives were fulfilled:

-The scientific novelty of the work lies in the fact that, for the first time, a conceptual model has been developed that integrates architectural and economic analysis for scalable voice identification systems, encompassing the relationship between CAPEX, OPEX, TCO, and SLA across four key deployment scenarios. A structural model of a typical voice identification system was designed, highlighting the key services and scaling points.

-The practical significance lies in formulating recommendations for IT project managers regarding the selection of architectures for voice identification systems. The proposed model can be used as a tool for preliminary techno-economic assessment when planning the implementation or modernization of biometric authentication systems in the financial sector, telecommunications, e-commerce, and government services.

**References:**

1. Amzur. (2024). Comparing CapEx & OpEx cloud models: Cloud cost optimization strategy. Amzur.
2. AWS. (2021). Optimizing enterprise economics with serverless architectures (AWS Whitepaper). Amazon Web Services.
3. CloudZero. (2023). CapEx vs. OpEx in the cloud: 10 key differences. CloudZero.
4. Gill, S. S., Singh, I., Chana, I., Buyya, R., et al. (2024). Modern computing: Vision and challenges. Journal of Modern Computing, 1(1), 1–35. [https://www.sciencedirect.com/science/article/pii/S2772503024000021]
5. Greenberg, C. S., Mason, L. R., Sadjadi, S. O., & Reynolds, D. A. (2020). Two decades of speaker recognition evaluation at the National Institute of Standards and Technology. Computer Speech & Language, 60, 101032. [https://colab.ws/articles/10.1016%2Fj.csl.2019.101032]
6. Guha, A. (2025, October 6). Building scalable voice AI: From MVP to enterprise. TringTring.AI. [https://tringtring.ai/blog/technical-deep-dive/building-scalable-voice-ai-from-mvp-to-enterprise]
7. Lee, C., et al. (2024). A systematic literature review on the strategic shift to cloud ERP leveraging microservice architecture and MSPs for resilience and agility. Electronics, 13(14), 2885. [https://www.mdpi.com/2079-9292/13/14/2885]
8. Liu, Y., et al. (2019). The THUEE system for NIST 2019 speaker recognition evaluation CTS challenge. arXiv preprint arXiv:1912.11585. [https://arxiv.org/pdf/1912.11585]
9. Microsoft. (2025). Cost efficiency considerations for your cloud adoption strategy. Microsoft Azure Architecture Center.[ https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/strategy/inform/cost-efficiency]
10. Someshwara Team. (2025, July 11). Building scalable voice AI systems: Best practices and challenges. Someshwara Tech Blog. [https://www.someshwara.com/blog/building-scalable-voice-ai]
11. TensorFlow. (n.d.). TensorFlow: An end-to-end open source platform for machine learning. Google Open Source.[ https://opensource.google/projects/tensorflow]
12. PyTorch Foundation. (n.d.). PyTorch: An open source deep learning framework. PyTorch / Linux Foundation AI. [https://ai.meta.com/tools/pytorch]
13. Google Cloud. (2020, May 15). Principles of cloud cost optimization. Google Cloud Blog. [https://cloud.google.com/blog/topics/cost-management/principles-of-cloud-cost-optimization]
14. Milvus. (2023). How can microservices architectures benefit audio search applications? Milvus AI Quick Reference. [https://milvus.io/ai-quick-reference/how-can-microservices-architectures-benefit-audio-search-applications]