UDK 004.78 **DOI: 10.18372/2073-4751.83.20545**

[1]**Artamonov Y.B.,**
orcid.org/0000-0002-9875-7372,
[2]**Kukhar Y.I.,**
orcid.org/0009-0000-7150-6672,
[3]**Plotytsia S.V.,**
orcid.org/0009-0007-3323-9316,
[4]**Skochynskyi B.D.,**
orcid.org/0009-0004-0603-8883,
[5]**Yanytska L.P.,**
orcid.org/0009-0007-1728-8599,

INTELLIGENT APPROACHES TO DESIGNING CLOUD-ORIENTED DECISION-SUPPORT SYSTEMS IN EDUCATION

[1,2]State University "Kyiv Aviation Institute",[3]Grid Dynamics Holdings, [4]TarasShevchenkoNationalUniversityofKyiv, [5]Luxoft USA Inc.

e-mail: eart@kai.edu.ua,
e-mail: yehor.kukhar@gmail.com,
e-mail: stepan@plotytsia.com,
e-mail: bogdanskochynskyi@gmail.com,
e-mail: lesiayanytska@gmail.com

Introduction

The digital transformation of education is accompanied by the active adoption of cloud technologies, analytical systems, and intelligent services that support managerial, pedagogical, and learning-related decision-making processes [1, 2, 3, 4]. Within the context of the evolution toward Industry 5.0 and human-centric Society 5.0, educational platforms must provide not only access to knowledge but also dynamic personalization, adaptation to individual learner characteristics, and continuous self-improvement of their internal algorithms [1, 5]. Achieving these capabilities requires intelligent architectures capable of integrating principles of cloud computing, machine learning, and cognitive modelling.

Traditional monolithic learning management systems (LMS) demonstrate limited scalability, weak support for integrating external analytical modules, and the need for manual updates of internal components, which collectively reduce system reliability and performance. As demonstrated in recent studies [6, 7, 8], a microservice architecture represents an effective approach to building contemporary distributed educational platforms, as it ensures service independence, flexibility of updates, and high fault tolerance. This architectural paradigm enables the integration of ML/AI modules as separate services, reducing the complexity of model lifecycle management and improving quality control mechanisms.

At the same time, advancements in MLOps technologies have created new opportunities for automating model training, testing, and deployment in production environments. A systematic review [9] confirms that combining MLOps practices with microservice-based architecture enables continuous model improvement (continuous training), detection of data drift, and seamless integration of AutoML tools [10]. These capabilities are particularly important for educational systems, where learning data streams are dynamic and context-dependent, and the performance of predictive models relies on the continual updating of their parameters.

The relevance of this study lies in the need to develop an intelligent cloud-oriented decision-support system (DSS) based on microservice architecture, equipped with integrated MLOps mechanisms and self-optimizing machine-learning pipelines.

The research problem spans three interconnected dimensions: architectural optimization (transition from monolithic to microservice solutions), intellectualization of analytics (integration of ML and AutoML) and cognitive support for decision-making (embedding explainable and adaptive models into learning processes).

Analysis of Research

In contemporary educational practice, decision-support systems (DSS) are increasingly implemented, typically relying on analytical dashboards, statistical modules, and models for predicting learning outcomes. However, most existing systems remain fragmented, primarily oriented toward descriptive analytics and lacking a full cognitive decision-making cycle.

An examination of current solutions [2, 3, 4, 11] shows that the majority of educational platforms do not incorporate unified methods for cognitive decision support at either the instructor level or the course-management level. Existing adaptation algorithms are generally limited to content filtering or statistical recommendations, without taking behavioural or cognitive factors into account. To transition toward intelligent DSS, it is necessary to integrate neuro-symbolic models, fuzzy logic, Bayesian inference, and multi-agent systems capable of explaining their decisions and interacting with pedagogical parameters of the learning process [3, 4, 12].

A further challenge is ensuring efficient management of computational resources and containerized components in cloud environments. As highlighted in [13, 14], automated scaling of Kubernetes clusters and load balancing across services significantly affect the performance of educational platforms and help reduce operational costs under high concurrency. Nevertheless, even under these conditions, the problem of optimizing inter-service communication persists: communication delays may degrade system responsiveness unless well-designed asynchronous interaction mechanisms are implemented [15-19].

Recent studies emphasize that effective DSS operation requires the integration of heterogeneous data sources and the use of flexible, scalable architectures [11, 13]. For example, [11] demonstrates the application of microservice architecture in e-learning systems to support personalized access to educational materials and rapid adaptation to changes in the learning process. However, these approaches still fall short of fully addressing predictive analytics and recommendation mechanisms, which are essential for next-generation DSS.

The problem of intellectualizing decision-making processes in education is further explored in [2, 4], which show that combining data analytics with cognitive models improves the accuracy and adaptability of decision-support systems.

Practical implementation of such models within scalable cloud environments faces several technical barriers: the need for continuous model monitoring, retraining, fault tolerance, and integration with real-time data streams. To address these challenges, [14] proposes mechanisms for dynamic resource scheduling and load distribution across microservices, ensuring stable operation of ML components within cloud-based DSS.

A growing body of research focuses on cloud computing as a foundational technology for building scalable DSS. Studies [13, 14] examine the performance of container management approaches in Kubernetes and other cloud-native platforms, demonstrating that optimizing scaling policies and resource orchestration directly affects the efficiency of educational analytics systems.

Another relevant line of research concerns ensuring the reliability and resilience of critical information systems, which is crucial for educational platforms with stringent requirements for security and continuous operation. In [20], a multi-level unified data model for cybersecurity in aviation information systems is proposed, based on hierarchical structuring and controlled subsystem interaction. Similarly, [22] explores methods for assessing the criticality of failures in complex information infrastructures, focusing on risk models and analyses of catastrophic scenarios. Applying these principles to the educational domain enables the development of DSS capable of self-diagnosis, failure prediction, and strategic response planning in cases of system faults or data loss.

In addition, [21] presents a systematic methodology for assessing the transition from monolithic architectures to microservices, offering criteria for evaluating performance, reliability, and operational resilience. Their framework facilitates formalization of decomposition processes, identification of critical dependency points, and minimization of migration-related risks. Given that modern educational platforms increasingly integrate ML modules, analytical services, and personalization components, structured migration strategies become crucial for ensuring controllability and predictability of system behaviour.

Within this context, microservice architecture enables the separation of decision-making logic from operational infrastructure, allowing independent component updates and seamless integration with MLOps processes [15, 19, 23]. In particular, [23] demonstrates that combining microservices with automated ML pipelines reduces system response time by 40% and increases model accuracy by 20% compared to monolithic solutions.

Modern DSS require the integration of structured and unstructured educational data into a unified analytical environment. To achieve this, Data Warehouse and Data Lake models are employed together with intelligent agents responsible for automatic data collection, cleansing, and contextual interpretation. Studies [15, 18] indicate that such integration not only enhances the accuracy of predictive models but also improves transparency by enabling traceability of data sources and inference logic.

Recent advances in cognitive modelling propose transferring principles of human reasoning into algorithmic frameworks for educational DSS. The use of Bayesian networks, fuzzy systems, and hybrid (symbolic and subsymbolic) AI enables the development of explainable artificial intelligence (XAI), which is essential in educational contexts.

As noted in [16, 18, 23], combining cognitive models with cloud infrastructure and MLOps tools enables adaptive decision support, where the system not only predicts outcomes but also explains its conclusions to users – teachers, administrators, or students.

Постановка проблеми

The purpose of this study is to identify an architectural solution for synthesizing methods of intelligent data analysis in the development of an intelligent cloud-oriented decision-support system (DSS) for education. Such a system must be capable of self-optimizing ML pipelines, scaling dynamically, and maintaining SLO (service level objective) thresholds under variable learning workloads [13, 14, 23].

Formally, for a set of learning events {E}, users {U}, and decisions {D}, the system must maximize the expected utility of decisions subject to performance and reliability constraints:

$$\max_{\pi\in\Pi} E_{(u,e)\sim D}\left[U(d=\pi(u,e),u,e)\right]$$
$$\begin{cases} latency(\pi)\le L^* \\ availability(\pi)\ge A^* \\ cost(\pi)\le C^* \end{cases} \tag{1}$$

whereπ – denotes the decision-making policy (service orchestration and model inference);
U – represents domain-specific utility (e.g., improved academic performance or timely pedagogical interventions);
L^*, A^*, C^* – SLO thresholds (for example, for API Latency SLO: 95% of requests must complete within ≤200 ms; for Availability

SLO: the platform must operate ≥99.9% of the time; for ML Model Accuracy SLO: recommendation accuracy ≥0.82).

Under this formulation, the intelligent DSS, at the Observation stage, collects data from various sources: LMS activity logs, user feedback, assessment results, sensor inputs, and behavioural signals. These form the incoming information stream o_t.

The Orientation stage involves constructing a cognitive representation of the context, i.e., the knowledge set K_t that describes the current state of the learning environment:

$$K_t=f_{orient}(K_{t-1},o_t), \qquad (2)$$

where $f_{orient}()$ – function responsible for aggregating, filtering, and normalizing incoming signals.

Following this, the system proceeds to the Decision stage, in which an action d_t is selected from the set of available alternatives D.

The final stage, Action, implements the chosen decision through the corresponding service of the platform (e.g., recommending educational content or initiating an instructor intervention), generating new observations and closing the cognitive adaptation loop [12].

To analyze and categorize educational data within the DSS, classical machine-learning algorithms are applied to ensure both interpretability and predictive accuracy.

A Decision Tree constructs a sequence of attribute-based questions that minimize information entropy; Random Forest aggregates results from multiple trees, increasing robustness to noise and overfitting; and Gradient Boosting minimizes the error of the previous ensemble by sequentially constructing weak learners $h_m(x)$:

$$F_m(x)=F_{m-1}(x)+v\, h_m(x), \qquad (3)$$

where v –a learning rate;
$h_m(x)$ –a regression function selected based on the loss gradient.

Within the educational context, these algorithms enable automatic identification of at-risk student groups, prediction of potential underperformance, assessment of the impact of student activity on outcomes, and generation of personalized content recommendations.

Learning-outcome forecasting is a key component of cognitive DSS. Regression and time-series models are commonly employed.

ARIMA(p,d,q) and LSTM models are used to analyze temporal patterns of student activity, allowing the system to forecast engagement fluctuations, detect declines in motivation, and automatically propose corrective measures [14].

Integrating predictive models into the MLOps lifecycle of the cloud architecture ensures their automatic retraining under data drift, a critical requirement in highly dynamic learning environments.

However, the formal optimization model alone does not define how such functionality can be implemented in a real high-load educational system. This necessitates intelligent approaches that combine artificial-intelligence methods (ML, AutoML, NAS), cognitive decision-support models, and MLOps principles.

For instance, the orientation model (2) requires aggregation, normalization, and storage services capable of constructing the cognitive state K_t. The gradient boosting scheme (3) requires an execution environment supporting repeated retraining, version control, and performance monitoring. The optimization problem (1) can only be solved when the policy π is materialized as a set of services responsible for observation, decision-making, and user interaction while maintaining SLO constraints.

Thus, the mathematical formulation directly imposes architectural requirements for the system: asynchronous event exchange, component isolation, scalable deployment, automated ML-pipeline management, and mechanisms for monitoring model quality. This justification motivates the next stage of research – developing a microservice-based cloud-oriented architecture capable of implementing models (1)–(3) and supporting a complete cognitive cycle in a dynamic educational environment.

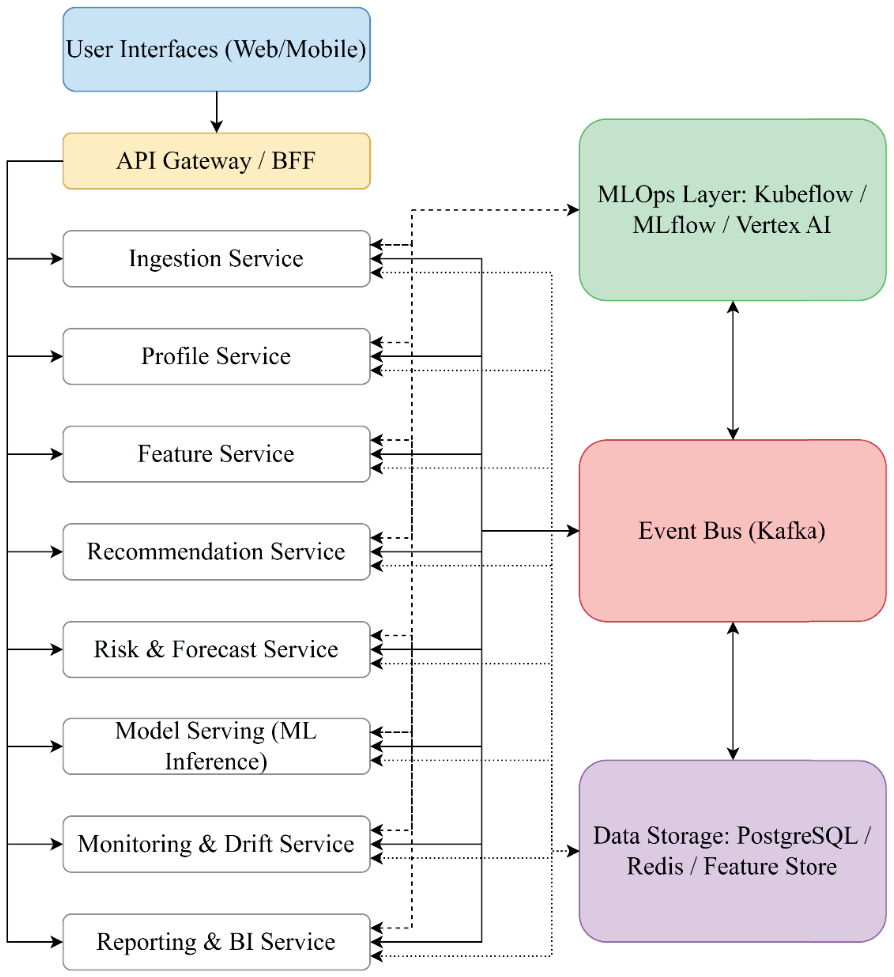
User Interfaces (Web/Mobile)
API Gateway / BFF
Ingestion Service
Profile Service
Feature Service
Recommendation Service
Risk & Forecast Service
Model Serving (ML Inference)
Monitoring & Drift Service
Reporting & BI Service
MLOps Layer: Kubeflow / MLflow / Vertex AI
Event Bus (Kafka)
Data Storage: PostgreSQL / Redis / Feature Store

Figure 1. The proposed architecture of a cloud-oriented decision-support system in education

Design of a Microservice-Based Cloud-Oriented Architecture

The architecture of the proposed cloud-oriented decision-support system for education (Fig. 1) is constructed according to principles of microservice decomposition, where each component performs a narrowly focused, specialized function while remaining integrated with others through standardized protocols and an event-driven messaging infrastructure.

At the upper level resides the user-interface layer, which includes web and mobile applications and provides interaction for students, instructors, and administrative users. All user requests are routed through an API Gateway / Backend-for-Frontend (BFF), responsible for authentication, authorization, request routing, and enforcing centralized access policies. The API Gateway serves as a controlled entry point, reducing load on internal services and unifying external integrations.

Requests are then forwarded to a set of microservices implementing the system's domain logic. The Ingestion Service is responsible for receiving and normalizing educational, behavioural, and operational data. These are subsequently processed by the Profile Service, which maintains user profiles and individual learning trajectories. The Feature Service constructs machine-oriented features for predictive and recommendation models, interacting with data sources and the MLOps subsystem. The Recommendation Service handles content-personalization requests and learning-path generation, leveraging outputs of machine-learning models. The Risk & Forecast Service evaluates academic risks (such as probability of dropout or low performance) and produces forecasts based on historical data.

Model Serving provides real-time deployment and execution of AI models and operates as an independent component responsible for low inference latency, scalable behaviour, and hot model updates. The Monitoring & Drift Service observes service stability, analyses data and model drift, detects anomalies, and triggers retraining procedures in case of quality degradation. The Reporting & BI Service provides analytical visualization, management dashboards, and supports decision-making at the instructor, faculty, or administrative levels.

A key integration element of the architecture is the Event Bus based on Kafka, which ensures asynchronous message exchange among microservices, traffic isolation, and scalable interaction within an event-driven model. Through Kafka, services publish events, react to them, or initiate model updates, feature generation, and analytical workflows. This avoids tight coupling and guarantees resilience to partial failures.

The MLOps Layer, comprising Kubeflow, MLflow, and Vertex AI, orchestrates the construction, versioning, preparation, and optimization of ML pipelines; automatic model retraining; performance evaluation; and experiment tracking. This layer enables continuous model evolution without user intervention and interacts with the Feature Service, Recommendation Service, Monitoring & Drift Service, and Model Serving through automated pipelines.

The data-storage subsystem includes PostgreSQL for transactional operations, Redis for caching and high-speed event processing, and a centralized Feature Store as a unified source of consistent features for models. Interaction with these components is bidirectional: services read required data and update it based on events arriving from Kafka and UI modules.

From a logical standpoint, the DSS is divided into four interrelated levels that form a continuous cyclical loop: data – analytics – decisions – feedback.

Data Ingestion Layerincorporates connectors to LMS/LTI systems, mobile and web clients, and external data sources (testing platforms, video services, etc.). The event stream is standardized using the schema: event_type, actor, context, payload, timestamp. Asynchronous delivery through Kafka minimizes coupling; synchronous calls are limited to API Gateway interactions.

Operational transactions of Data Storage Layerare stored in PostgreSQL (OLTP); telemetry time series are stored in a metrics repository; hot data is cached in Redis. Analytical snapshots and long event histories are stored in a columnar database or S3-compatible object storage for offline computation. Features for ML are maintained in a Feature Store to ensure consistency between online and offline pipelines.

Analytics & Decisions Layer includes microservices responsible for feature processing, model evaluation, online recommendations, risk forecasting, explainability (XAI), and data/model-drift monitoring. Orchestration of ML pipelines is also located here.

Presentation Layercomprises student and instructor interfaces, administrative dashboards, and reporting components. Data is accessed through the BFF layer and analytical APIs.

Microservice architecture adheres to principles of loose coupling and independent scalability:

– API Gateway: single entry point, routing, rate-limiting, OAuth2/OIDC, auditing;

– BFF (Backend-for-Frontend): separate BFFs for student, instructor, and admin interfaces, aggregating data from domain services and reducing backend chatter;

– Ingestion Service: receives events from clients/external sources, normalizes them, and publishes to Kafka;

– Profile Service: user profiles, learning trajectories, competence statuses; CRUD operations in PostgreSQL with Redis-accelerated reads;

– Feature Service: builds and serves online features; synchronizes with the Feature Store;

– Recommendation Service: online content recommendations; supports A/B experiments and canary model releases;

– Risk & Forecast Service: risk scoring, performance forecasting, and intervention triggers;

– Model Serving: isolated instances for model inference (REST/gRPC), versioning, and traffic routing by model version;

– Monitoring & Drift Service: collects performance metrics, detects data/concept drift, initiates retraining;

– Reporting & BI Service: analytical data marts, reporting APIs, export mechanisms.

Critical domain events are transported through Kafka topics such as:edu.activity. logged (atomic learner actions), ml.features.updated (feature updates), ml.model.requested / ml.model.scored (model-inference lifecycle), etc.

A typical user-action event contract in JSON format:

```
{
  "event_type": "edu.activity.logged",
  "actor_id": "user-123",
  "context": {"course_id": "c-42", "lesson_id": "l-7"},
  "payload": {"action": "quiz.submit", "score": 0.82, "duration_sec": 315},
  "timestamp": "2025-11-05T14:21:05Z"
}
```

Observability is ensured by:

– OpenTelemetry for unified logs/traces/metrics;

– Prometheus metrics with Grafana visualization and logging through a centralized logging stack;

– SLO/SLI parameters for critical paths (latency_p95, recommendation_hit@K, error_rate, etc.).

Data-processing flows within the decision lifecycle can be summarized as follows:

1. A user event arrives at the Ingestion Service and is published to Kafka.

2. The Feature Service updates online features and synchronizes them with the Feature Store.

3. The Recommendation Service calls Model Serving, obtains recommendations, and returns them through the BFF and UI.

4. The Monitoring & Drift Service tracks performance metrics and publishes an ml.drift.detected event in case of drift.

5. Kubeflow executes a training pipeline, and MLflow registers a new model version.

6. Canary deployment evaluates online performance; promotion to production occurs only if SLOs are met; otherwise, automatic rollback is triggered.

7. Reporting & BI aggregates events for management dashboards and official reporting.

The overall analysis demonstrates that architectural decisions, algorithmic components, and data-processing flows reinforce one another, establishing a robust foundation for a DSS capable of adaptation, self-optimization, and scalable operation in real educational environments.

Conclusions

Contemporary educational platforms have reached a point where traditional monolithic LMS solutions can no longer adequately respond to the dynamic nature of learning data, the growing number of users, or the increasing demand for personalization. The transition toward an intelligent cloud-oriented decision-support system in education requires a simultaneous rethinking of theoretical decision-making models, architectural principles of platform design, and the lifecycle of machine-learning models. It is at the intersection of these three dimensions—cognitive, architectural, and MLOps-related—that the logic of the proposed approach is formed.

The proposed DSS conceptualizes the learning process as a cognitive cycle of observation–orientation–decision–action, where each stage has a clearly defined informational role. Learning events, behavioural signals, and assessment results constitute an observational data stream, from which a dynamic representation of contextual knowledge is constructed. This cognitive model underlies the formal problem formulation, where the system does not merely accumulate data but optimizes the utility of decisions under constraints defined by service-level objectives—response latency, availability, and model accuracy. Pedagogical goals (timely interventions, reducing the risk of underperformance, supporting individualized learning trajectories) are thus directly linked to the technical characteristics of the cloud infrastructure, representing an important step toward a holistic understanding of the educational platform.

To realize the described cognitive cycle in a real high-load environment, the system must support asynchronous event flows, isolated domain services, independent scaling, and a strictly organized lifecycle of machine-learning models. This requirement motivated the development of a microservice-based cloud-oriented architecture in which mathematical dependencies are mapped to concrete technical implementations—services, data stores, communication channels, and scaling policies.

A key result of the study is that MLOps is not an auxiliary component of the architecture but constitutes its core. The integration of Kubeflow, MLflow, and Vertex AI provides automatic model retraining under data drift, experiment management, canary deployments, and rollback procedures in case of quality degradation. These capabilities enable continuous self-adjustment of models and restoration of target SLO indicators based on ongoing evaluation of system state.

The proposed approach demonstrates how intelligent methods acquire practical meaning within the structural components of microservice implementation and MLOps-based lifecycle organization. The practical potential of such a DSS spans university platforms, corporate training systems, and national educational infrastructures requiring scalability, personalization, reliability, and transparency in AI-model operation.

Future research directions include the integration of multimodal data and the development of dynamic knowledge graphs, which would enable a transition from isolated recommendations to deep adaptation of learning trajectories. Overall, the presented cloud-oriented microservice DSS can serve as a foundation for a new class of intelligent educational ecosystems that combine engineering rigor, cognitive coherence, and a human-centric perspective as the core of the learning process.

References

1. Adel A, Alani N.H.S. Human-centric collaboration and Industry 5.0 framework in smart cities and communities: fostering sustainable development goals 3, 4, 9, and 11 in Society 5.0. *Smart Cities.* 2024. Vol. 7, 4. P. 1723–1775. DOI:10.3390/smartcities7040068

2. Chatzopoulou D.I., Economides A.A. Adaptive assessment of student's knowledge in programming courses. *Journal of Computer Assisted Learning.* 2010. Vol. 26, 4, P. 258-269. DOI: 10.1111/j.1365-2729.2010.00363.x.

3. Melesko J, Kurilovas E. Personalised intelligent multi-agent learning system for engineering courses. *2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE).* 2016. DOI: 10.1109/AIEEE.2016.7821821

4. Kristensen T., Dyngeland M. Design and Development of a Multi-Agent E-Learning System. 2015. *International Journal of Agent Technologies and Systems.* Vol. 7, 2, P. 19-74. DOI: 10.4018/IJATS.2015040102

5. Yanytska L. The rise of human-centric manufacturing in the industry 5.0 era. *Int J Adv Manuf Technol.* 2025. Vol. 139. P. 5067–5077. DOI: 10.1007/s00170-025-16192-5.

6. Di Francesco P., Lago P., Malavolta I. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software.* 2019. P. 77-97. DOI: 10.1016/j.jss.2019.01.001.

7. Marieska M.D., Yunanta A., Auliam H., Utami A.S. Rizqie M.Q. Performance Comparison of Monolithic and Microservices Architectures in Handling High-Volume Transactions. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi).*

2025. Vol. 9, 3. P. 594-600. DOI: 10.29207/resti.v9i3.6183

8. Dragoni N., Lanese I., Larsen S.T., Mazzara M., Mustafin R., Safina L. Microservices: How to make your application scale. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* 2017. 10742 LNCS, 95-104. DOI: 10.1007/978-3-319-74313-4_8.

9. Zarour M., Alzabut H., Al-Sarayreh K.T. MLOps best practices, challenges and maturity models: A systematic literature review. *Information and Software Technology.* 2025. Vol. 183. 107733. DOI: 10.1016/j.infsof.2025.107733.

10. Barbudo R., Ventura S., Romero J.R. Eight years of AutoML: categorisation, review and trends. *Knowl Inf Syst.* 2023. 65. 5097–5149. DOI: 10.1007/s10115-023-01935-1

11. Artamonov Y., Golovach I., Zymovchenko V. Use analysis of microserves in e-learning system with multi-variant access to educational materials. *Technology Audit and Production Reserves.* 2021. 4 (2 (60)), 45–50. DOI: 10.15587/2706-5448.2021.237760.

12. Artamonov E.B, Zholdakov O.O. Concept of creating a software environment for automated text manipulation. *Scientific journal "Proceedings of the National Aviation University".*2010 . Vol. 3 (44), 111-115.

13. Fu Y., Gu S., Cheng L., Liu L. Performance evaluation of resource management schemes for cloud-native platforms with computing containers. *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC).* 2022. DOI: 10.1109/ipccc55026.2022.9894300.

14. Mustyala A. Dynamic resource allocation in Kubernetes: Optimizing cost and performance. *EPH – International Journal of Science and Engineering.* 2021. 7, 3. DOI: 10.53555/ephijse.v7i3.237.

15. GonzálezS. Modular software design in distributed systems: Strategic approaches for building scalable, maintainable, and fault-tolerant architectures in modern microservice environments. *Eigenpub Review of Science and Technology.* 2023. 7, 1. DOI: 10.1007/s10916-020-1195-x.

16. Hang Y., Xiulei W., Changyou X., Bo X. A Microservice Resilience Deployment Mechanism Based on Diversity. *Security and Communication Networks.* 2022. 7146716. DOI: 10.1155/2022/7146716.

17. Kazanavičius J.,& Mažeika D. The Evaluation of Microservice Communication While Decomposing Monoliths. *Computing and Informatics.* 2023. 42(1), 1–36. DOI: 10.31577/cai_2023_1_1

18. MejíaP. Best practices for microservice framework design. *Advances in Intelligent Information Systems.* 2022. 7, 1. URL: https://questsquare.org/index.php/JOURNALAIIS/article/view/70.

19. Bravetti M., Giallorenzo S., Mauro J., Talevi I., Zavattaro G. Optimal and automated deployment for microservices. *Fundamental Approaches to Software Engineering.*2019 .351–368. DOI: 10.1007/978-3-030-16722-6_21.

20. Gnatyuk S. Multilevel Unified Data Model for Critical Aviation Information Systems Cybersecurity. *2019 IEEE 5th International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD).* 2019. 242-247. DOI: 10.1109/APUAVD47061.2019.8943833.

21. Auer F., Lenarduzzi V., Felderer M., Taibi D. From monolithic systems to Microservices: An assessment framework. *Information and Software Technology.* 2021. Vol. 137.

22. Gnatyuk S., Sydorenko V., Polihenko O., Sotnichenko Y., Nechyporuk O. Studies on the disasters criticality assessment in aviation information infrastructure. *CEUR Workshop Proceedings.* 2020. 282–296. ISSN: 16130073.

23. Artamonov Y.B., Plotytsia S.V., Radchenko K.M., Kotsiur A.B. Microservice Architecture of Intelligent Educational Platforms with ML Pipeline Self-Optimization. Science and technology today. 2025. 10, 51 P. 1059-1073. DOI: 10.52058/2786-6025-2025-10(51)-1059-1073.

ArtamonovY.B., KukharY.I., PlotytsiaS.V., SkochynskyiB.D., YanytskaL.P.

INTELLIGENT APPROACHES TO DESIGNING CLOUD-ORIENTED DECISION-SUPPORT SYSTEMS IN EDUCATION

The article presents a comprehensive approach to designing an intelligent cloud-oriented decision-support system (DSS) for the educational domain, combining the principles of microservice architecture, cognitive modelling, and machine-learning operations. The relevance of the study is justified by the growing volume of educational data, the need for personalized learning trajectories, and the necessity to maintain stable SLO indicators under dynamically changing learning conditions. The analysis of current research highlights the limitations of traditional monolithic LMS platforms, which lack sufficient flexibility, scalability, and capabilities for integrating intelligent models.

The proposed system architecture is based on event-driven microservice interaction via Kafka and incorporates modules for collecting and normalizing learning events, a feature-engineering subsystem, recommendation and risk-prediction services, and a dedicated pipeline for modelling and monitoring ML components. The results of the study demonstrate that combining microservice decomposition with intelligent data-analysis methods improves recommendation accuracy, enhances performance indicators, and ensures the resilience of the educational platform under high load. The presented architecture can serve as a foundation for building scalable and adaptive next-generation educational ecosystems.

***Keywords**: decision-support system, microservice architecture, educational data analytics, MLOps lifecycle, adaptive learning systems.*

Артамонов Є.Б., Кухар Є.І., Плотиція С.В., Скочинський Б.Д., Яницька Л.П.

ІНТЕЛЕКТУАЛЬНІ ПІДХОДИ ДО ПРОЄКТУВАННЯ ХМАРНО-ОРІЄНТОВАНИХ СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В ОСВІТІ

У статті представлено комплексний підхід до проєктування інтелектуальної хмарно-орієнтованої системи підтримки прийняття рішень (СППР) у сфері освіти, в якій поєднано принципи мікросервісної архітектури, когнітивного моделювання та операції машинного навчання. Актуальність роботи обґрунтована зростанням масштабів освітніх даних, потребою в персоналізованих навчальних траєкторіях та необхідністю забезпечення стабільних SLO-показників у динамічних умовах навчального процесу. Аналіз сучасних досліджень виявив обмеження традиційних монолітних LMS, які не забезпечують достатньої гнучкості, масштабованості та можливостей інтеграції інтелектуальних моделей.

Запропонована архітектура система ґрунтується на подієво-орієнтованій взаємодії мікросервісів через Kafka та включає модулі збору й нормалізації навчальних подій, підсистему формування ознак, сервіси рекомендацій та прогнозування ризиків, а також окремий контур моделювання та моніторингу ML-компонентів. Результати дослідження демонструють, що поєднання мікросервісної декомпозиції з інтелектуальними методами аналізу даних підвищує точність рекомендацій, покращує показники продуктивності та забезпечує стійкість освітньої платформи за умов високого навантаження. Представлена архітектура може бути використана як основа для створення масштабованих та адаптивних освітніх екосистем нового покоління.

***Ключові слова**: система підтримки прийняття рішень, мікросервісна архітектура, аналітика освітніх даних, життєвий цикл MLOps, адаптивні навчальні системи.*