PLOVDIV UNIVERSITY
"PAISII HILENDARSKI"
FACULTY OF MATHEMATICS AND INFORMATICS
DEPARTMENT OF COMPUTER TECHNOLOGIES

PAVEL ALEKSANDROV KYURKCHIEV

METHODS AND ARCHITECTURAL APPROACHES FOR SEMANTIC INDEXING AND SIMULATION OF DYNAMIC PROCESSES IN A CLOUD ENVIRONMENT

ABSTRACT

of a dissertation

for awarding the educational and scientific degree of **"Doctor" (PhD)**

in the field of higher education: 4. Natural sciences, mathematics, and informatics;

professional field: 4.6. Informatics and computer science;

PhD program: "Informatics"

Scientific supervisor: Prof. Anton Iliev, PhD

Plovdiv

2026

The dissertation was discussed and recommended for defense at an extended meeting of the Department of Computer Technologies at the Faculty of Mathematics and Informatics of Plovdiv University "Paisii Hilendarski".

The dissertation consists of 140 pages, which include 18 figures and 12 tables. The bibliography contains 146 references.

The list of the author's publications consists of 3 titles.

The defense of the dissertation will take place on **.....** in ....., Plovdiv.

The defense materials are available to those interested at the FMI secretariat, New Building of Plovdiv University, Room 330, every working day from 8:30 to 17:00.

Scientific Jury:

1. .....
2. .....
3. .....
4. .....
5. .....

Author: Pavel Aleksandrov Kyurkchiev
Title: Methods and architectural approaches for semantic indexing and simulation of dynamic processes in a cloud environment

Plovdiv, 2026

Contents

General Characteristics of the Dissertation

Relevance of the Problem

The development of computer science and the transition from isolated desktop applications to distributed cloud systems (Cloud Computing) [1] have fundamentally transformed the methods for modeling and simulation. Despite the emergence of the "Simulation-as-a-Service" concept, modern platforms face the "Paradox of Plenty" [2] – the availability of massive computational power, but inefficient knowledge management.

Traditional systems function as "black boxes," storing logic in closed or binary formats. Search within them relies on classical lexical methods (Keyword Search), which are inefficient in discovering dynamic models due to the "Vocabulary Mismatch" problem [3].

The integration of artificial intelligence methods, particularly Large Language Models (LLMs) and vector search, provides a unique opportunity to "unlock" this hidden knowledge. Existing approaches currently focus primarily on the syntactic analysis of source code [4,5], neglecting the deep mathematical and causal semantics that underlie system dynamics models. It is precisely this scientific gap that motivates the search for new architectural paradigms for translating graphs into a machine-readable context.

Object and Subject of the Research

The **object of the research** is software systems for modeling and simulation of dynamic processes, examined and implemented through the established formalism of System Dynamics (SD) [6,7], operating in a modern distributed cloud environment.

The **subject of the research** comprises the methods for architectural design, algorithms for numerical simulation of systems of differential equations, and approaches for semantic analysis, linearization, and vectorization of mathematical models using Transformer-based language architectures.

Aim of the Dissertation

The main objective of the dissertation is to investigate and apply the methods of the "System Dynamics" formalism with modern technologies for semantic analysis and vector search in a cloud environment, and through the implementation of a prototype, to validate a methodology for visual modeling, numerical simulation, and intelligent retrieval of dynamic processes.

Working Hypothesis

The dissertation posits the hypothesis that integrating a microservices cloud architecture with methods for vector semantic analysis will enable overcoming the "vocabulary mismatch" problem in dynamic models. This will increase the Precision and Recall of search by transitioning from syntactic comparison (based on variable labels) to semantic matching (based on topological and functional structure), transforming the mathematical essence of

the models into a universal, vectorized language that algorithms can effectively analyze and cluster.

Research Tasks

To achieve the set objective, the following main tasks have been defined and solved:

1. **State-of-the-art analysis and definition of an architectural model:** Comparative analysis of modeling approaches, investigation of methods for automatic equation extraction, and definition of a scalable microservices architecture integrating a simulation module with a vector database.

2. **Development of a methodology for transformation and search:** Definition of an algorithm for semantic linearization of graph models (DFS traversal and explicit cycle detection) and development of a strategy for serialization into a rich text context optimized for NLP models.

3. **Design and software implementation of an experimental environment:** Implementation of a modular cloud architecture (*Alpha Cloud Data Hub*) using Domain-Driven Design (DDD), including an interactive user interface and isolated backend modules for numerical simulation and real-time semantic indexing.

4. **Experimental validation and analysis:** Verification of the simulation module by comparison with reference models (ecological Predator-Prey model, financial model for bank liquidity) and quantitative evaluation of the semantic search efficiency against traditional lexical approaches.

Research Methodology

To achieve the objectives, a complex interdisciplinary approach was applied:

– **Constructive and deductive approach:** Building a conceptual architectural model and materializing it through an innovative software prototype (*Alpha Cloud Data Hub*), which serves as an experimental testing ground.

– **Algorithmic translation:** Development of a specialized methodology for semantic linearization (Graph Traversal and Cycle Detection), providing the connection between graph mathematical logic and natural language processing algorithms.

– **Comparative analysis and experimental validation:** Quantitative evaluation using standard metrics (Precision, Recall, F1-Score, Latency) on a specially formed isolated test corpus of 63 models and comparing the results with established lexical search methods.

– **Polyglot Persistence:** Combining document-oriented (NoSQL/MongoDB) and vector databases (Qdrant) with NLP technologies (BERT/ONNX), ensuring high performance and fault tolerance.

Volume and Structure of the Dissertation

The dissertation is written in Bulgarian and is structured in a volume of 140 pages. It is divided into an introduction, five main chapters, a conclusion, a bibliography (146 references), and three appendices. The work is richly illustrated with 18 figures, 12 tables, and numerous source code listings, which visually support the scientific conclusions drawn.

In terms of logic and content, the dissertation follows a strict deductive sequence:

- **The Introduction** motivates the relevance of the problem with knowledge management in simulation environments and defines the research hypothesis, objectives, and tasks of the study.
- **Chapter 1** provides a critical review of modeling approaches and software platforms, defining the technological vacuum and justifying the niche for developing a cloud ecosystem.
- **Chapter 2** builds the theoretical foundation, connecting the mathematical apparatus of System Dynamics with modern Transformer architectures and vector spaces.
- **Chapter 3** presents the main scientific and applied contribution – the author's methodology and algorithms for semantic linearization of cyclic graphs, optimized for artificial intelligence analysis.
- **Chapter 4** describes the practical software design of the microservices platform *Alpha Cloud Data Hub* and the integrated strategy for polyglot data persistence.
- **Chapter 5** contains the empirical validation of the system through stress tests on a specialized corpus, quantitatively proving the superiority of the semantic approach in search.
- **The Conclusion and Appendices** summarize the scientific contributions, outline perspectives for future development, and supplement the work with technical specifications, API documentation, and open data.

Summary of the Dissertation

Introduction

The first chapter introduces the field of modern computer modeling, where a clear trend of migration from local, desktop-based solutions to distributed cloud architectures and the provision of "Simulation-as-a-Service" is observed. The main problem is defined: the availability of massive computational power for executing models, combined with the inability for their intelligent discovery due to the "Vocabulary Mismatch" phenomenon. The modern engineer is in the position of a librarian who has millions of books but no catalog and must open each one individually to understand its content. Traditional search engines index only the metadata, while the internal logic of the models remains hidden in closed formats ("black boxes"). The aim, tasks, object, subject, and working hypothesis of the research are formulated, based on the integration between the mathematical formalism of System Dynamics (SD) and modern technologies for Natural Language Processing (NLP) and vector search.

Chapter 1. Domain Analysis and Existing Solutions

The introduction presents a critical review of the evolution in modeling and a comparative analysis of the main approaches:

- **Discrete-Event Simulation (DES) [8]:** Suitable for the operational management of queues and resources, but with a low level of abstraction.
- **Agent-Based Modeling (ABM):** A bottom-up approach, effective for emergent behavior, but requiring heavy procedural code (if-then-else logic).
- **System Dynamics (SD):** A top-down macro-perspective, based on integral equations and continuous time [9]. Its mathematical purity makes it ideal for translation and semantic analysis.

Critical analysis of existing platforms and formats:

- **Traditional systems (Stella, AnyLogic [10], GoldSim):** They offer massive computational power but suffer from architectural obsolescence. They function as monolithic desktop applications and use closed formats (Java Bytecode, Binary), which turns them into impenetrable "black boxes" for external search engines.
- **Exchange formats (XMILE, XML, FMI, Modelica):** Although they provide portability and precise simulation, they are "semantically blind" and generate significant "syntactic noise" compared to lighter modern formats like JSON [11]. Their structure is extremely verbose, which quickly exhausts the limited Context Window of modern Transformer models.

It is also proven that classical deterministic algorithms for graph search (such as Subgraph Isomorphism [12] and Graph Edit Distance [13]) suffer from exponential complexity $\mathcal{O}(N!)$

and are inapplicable for real-time search in a scalable cloud environment. To overcome these deficits, the *Alpha Cloud Data Hub* system introduces a lightweight, semantically dense **JSON format**, specifically optimized for NLP analysis. This transition from procedural code to declarative data is crucial, as it allows algorithms to read the mathematical topology without the need for complex syntactic parsing. All this defines the research niche: creating an intelligent cloud system that extracts and manages knowledge, rather than merely executing simulations.

Conclusions for Chapter 1

- **Platform encapsulation:** Traditional environments (Stella, AnyLogic) treat models as "black boxes," limiting their semantic transparency and reusability.
- **Syntactic noise:** Open formats (XML, XMILE) possess a syntactic verbosity that quickly consumes the context window of Transformer models.
- **Graph limitations:** Classical algorithms for graph comparison have an exponential complexity $\mathcal{O}(N!)$ and are non-scalable for real-time operation.
- **Lexical failure:** Search engines (BM25) generate noise and fail in conceptual queries due to the *Vocabulary Mismatch* problem.
- **Research niche:** There is a technological vacuum for a *cloud-native* ecosystem connecting System Dynamics with latent vector spaces.

Chapter 2. Mathematical Formalism and Theoretical Foundations

This section presents the mathematical apparatus on which the system is built. System Dynamics is formalized through Stock and Flow Diagrams. The relationship between a stock and its net flow is described by the integral equation:

$$Stock(t) = \int_0^t (Inflow - Outflow)dt + Stock(t_0)$$

For the numerical solution of these equations in the cloud architecture, classical methods are analyzed. The 4th-order Runge-Kutta method (RK4) is considered as an industry standard for precision:

$$S_{t+dt} = S_t + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

However, for the needs of *Alpha Cloud Data Hub* and semantic translation, the choice of the Forward Euler method [14] is justified as the baseline integrator due to its computational simplicity, potential for parallelization and hardware optimization [15], and the ability to easily trace causal relationships:

$$S_{t+dt} = S_t + F(S_t, t) \cdot dt$$

For the purposes of intelligent search, the concept of Vector Embeddings [16] is introduced. Objects are transformed into continuous vectors in a multidimensional space $\mathbb{R}^n$ through a mapping function $f : Object \to \mathbb{R}^n$. Semantic similarity is calculated via Cosine Similarity:

$$similarity(A, B) = cos(\theta) = \frac{A \cdot B}{||A||||B||}$$

The use of Transformer architectures (BERT) [17,18] and the ONNX (Open Neural Network Exchange) format is justified for hardware agnosticism and local execution. The architecture of vector databases and the HNSW (Hierarchical Navigable Small World) [19] algorithm are discussed, which reduce search complexity from linear $\mathcal{O}(N)$ to logarithmic $\mathcal{O}(\log N)$.

Conclusions for Chapter 2

- **System Dynamics topology:** The distinction between stocks and flows provides a declarative structure ideal for automated translational analysis.
- **Numerical Solver:** Euler's method combines computational simplicity with semantic transparency of causal relationships in a cloud environment.
- **Vectorization via BERT:** The transformation into embeddings overcomes the manual creation of ontologies and provides resilience to terminological variations.
- **Optimization via ONNX:** The standard ensures hardware agnosticism and quantization for efficient execution in resource-constrained microservices.
- **Logarithmic search:** Vector databases (Qdrant) with HNSW reduce search complexity to logarithmic $\mathcal{O}(\log N)$, ensuring scalability.

Chapter 3. Methodology for Transformation, Simulation, and Intelligent Search

This chapter defines the algorithmic framework of the *Alpha Cloud Data Hub* system. The process of translation into executable code begins with a topological sorting of the graph to determine the Evaluation Order, treating stocks as breaking points for algebraic loops. The main scientific contribution in this chapter is the developed methodology for **"Semantic Linearization"**. Its goal is to overcome the fundamental difference between the graph topology of the models and the linear structure required by Large Language Models (LLMs). A mathematical linearization function $\mathcal{L} : G \to S$ is defined, which constructs the sequence of tokens S by concatenating local semantic paths:

$$S = Context(G) \oplus \bigoplus_{v \in V_{start}} DFS(v, \emptyset)$$

A theoretical analysis of traversal strategies was conducted. Breadth-First Search (BFS) was rejected because it causes *contextual fragmentation*. Depth-First Search (DFS) [20] was chosen because it preserves the causal chain ($A \to B \to C$), ensuring a constant distance $\Delta t = \mathcal{O}(1)$ between concepts. This optimizes the calculation of the Self-Attention mechanism in Transformer models.

Explicit Tokenization of Cycles and Algorithmic Implementation

In the presence of Feedback Loops, standard algorithms silently terminate the traversal, which leads to the loss of critical knowledge. The introduced methodology generates a special semantic marker `[CYCLE_DETECTED]`, which acts as an indicator for the NLP model. To ensure predictable execution time in the cloud microservice environment and prevent Out-Of-Memory errors, the algorithm introduces a recursion depth limiter ($D_{max} = 10$). The logic of the process is implemented through the following recursive approach:

```
// Input:  Graph G = (V, E)
// Output: Execution stack S

function BuildExecutionOrder(Graph G):
    Stack executionStack = new Stack()
    Set visited = new Set()
    Set recursionStack = new Set()

    // Step 1: Mark Stocks as base (known)
    foreach node in G.Nodes:
        if node.Type == "Stock":
            visited.Add(node)

    // Step 2: Traverse remaining nodes (DFS)
    foreach node in G.Nodes:
        if node.Type != "Stock" AND node not in visited:
            if not Visit(node,
                         executionStack,
```

```
                            visited,
                            recursionStack):
                    return Error("Algebraic Loop Detected")

    return executionStack
```

Листинг 1: Pseudocode of the topological sorting algorithm

This software implementation ensures that the space complexity remains strictly limited to $\mathcal{O}(D_{max})$, and the generated semantic context stays within the allowable window of the NLP model. The comparison between the naive and the semantic approach clearly illustrates the advantage of linearization:

> *Naive enumeration:* `Market_Demand, Revenue_Inflow, Cash_Reserves...`
> *Semantic linearization:* `? Market_Demand via influence` $\rightarrow$
> `? Revenue_Inflow via flow_to` $\rightarrow$ `? Cash_Reserves [cycle detected]`

Extraction of Global Context and Aggregation

While semantic linearization focuses on the micro-structure of causal relationships, qualitative NLP analysis also requires defining a macro-framework. Without it, the vector model might lose its orientation in the multidimensional space (e.g., failing to distinguish whether the term "flow" refers to a hydrological process or a financial transfer). For this purpose, the algorithm generates a "Semantic Anchor" based on the model's metadata, including model type, scientific domain, and structural statistics (number of nodes and links). In the final phase, the results are synthesized into a single document ready for vectorization. The aggregation process is not just a concatenation but a structured arrangement into three layers: (1) Contextual framework; (2) Structural narrative (the paths from the DFS traversal); and (3) Categorical index (lexical backup). This multi-layered structuring follows the reading logic of Transformer models. **The entire architectural processing pipeline from the structured JSON model to the final vector output is visually illustrated in Figure 1.**

To optimize the search, algorithmic optimization (HNSW), Single Vector Record (SVR) strategy for comprehensive comparison of conceptual structures, and Domain Prompt Injection are applied, systematically adding a prefix to the user query: `"Represent this sentence for searching System Dynamics models: "`.

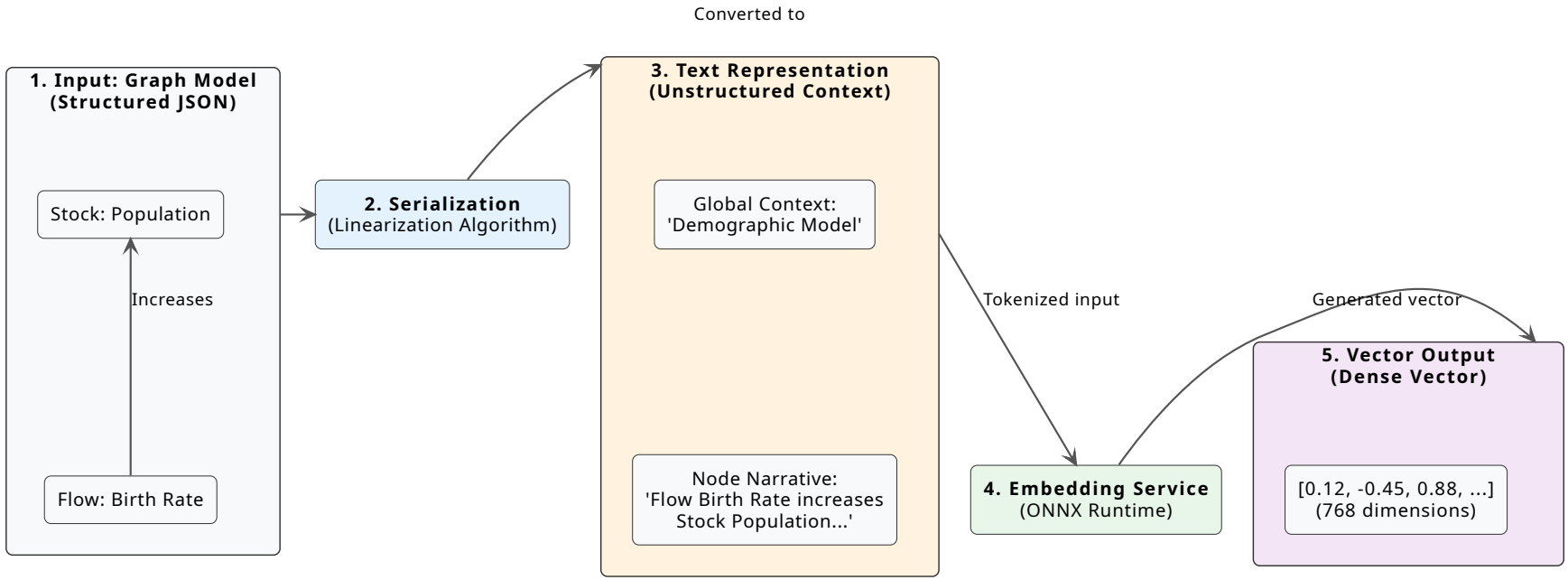

Figure 1: Architectural processing pipeline for the semantic transformation and indexing process.

Note: The diagram illustrates the data sequence from JSON format to vector index.

The developed algorithmic complex, covering the sequential phases of topological sorting, semantic linearization via modified depth-first search (DFS), and explicit tokenization of feedback loops, defines a complete and autonomous methodological framework for extracting causal knowledge from complex cyclic graphs. The integration of the global contextual framework (semantic anchor) and the three-layer aggregation of the extracted text narrative optimizes the energy of dot products in the latent space, ensuring maximally strong Attention Weights during subsequent vector encoding. By conceptually combining hierarchical HNSW graph indexing, the Single Vector Record (SVR) strategy, and asymmetric domain context injection, the proposed method successfully bridges the conceptual gap between the short, ambiguous user input and the lengthy mathematical structure of the models. Thus, the defined computational framework solves the fundamental *Vocabulary Mismatch* problem, transitioning from superficial syntactic filtering to deep structural and functional pattern recognition. This complete theoretical methodology for transformation, simulation, and intelligent search serves as a direct logical bridge to the next stage of the research, focused on software design, multi-layered decentralization, and the actual microservice implementation of the platform in a cloud environment.

Conclusions for Chapter 3

- **Semantic linearization of graphs:** The developed original methodology and mathematical transformation function successfully translate the non-linear cyclic topology of System Dynamics models into a structured one-dimensional text narrative, suitable for processing by modern latent models.
- **Efficiency of DFS traversal:** Comparative analysis proves that, unlike BFS, the depth-first search (DFS) algorithm preserves the causal context with a constant distance between concepts, which optimizes the dot products in the self-attention mechanism.
- **Explicit tokenization of cycles:** The introduction of the specialized semantic marker `[CYCLE_DETECTED]` in combination with a strict recursion depth limiter prevents the loss of structural knowledge about feedback loops and eliminates the risk of combinatorial token explosion.
- **Multi-layer aggregation and context:** Synthesizing the generated semantic context into three strictly differentiated layers (contextual framework, structural narrative, and categorical index) ensures correct orientation of the Transformer model and minimizes overall semantic entropy.

Chapter 4. Architecture of the Alpha Cloud Data Hub Platform

The software implementation of *Alpha Cloud Data Hub* is based on the principles of Domain-Driven Design (DDD) [21] and microservice architecture [22,23]. To prevent blocking the user interface during computationally heavy AI operations, the system applies an event-driven model in designing intensive data streams [24] through the *RabbitMQ* message broker [25]. The user interface is implemented as a Single Page Application (SPA) [26], allowing interactive diagram building and synchronous visualization of simulation results. The interface provides a palette with the basic elements of System Dynamics for building visual topology (Figure 2), which is converted in real-time into a JSON structure. The transition to an executable quantitative model is achieved by defining the mathematical parameters of the elements (Figure 3).

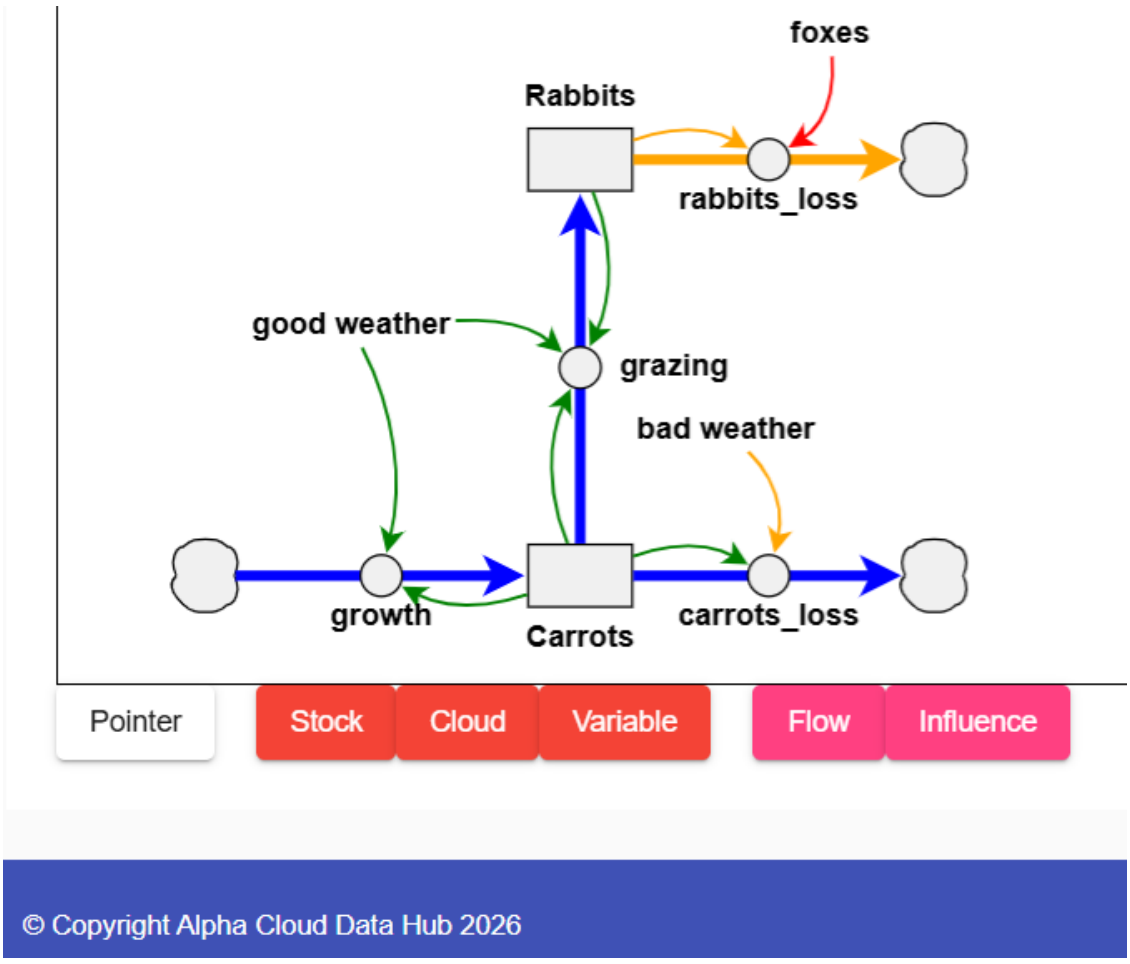

Figure 2: Canvas of the visual editor in Alpha Cloud Data Hub, demonstrating interactive construction of an ecological Stock & Flow model through direct manipulation components.

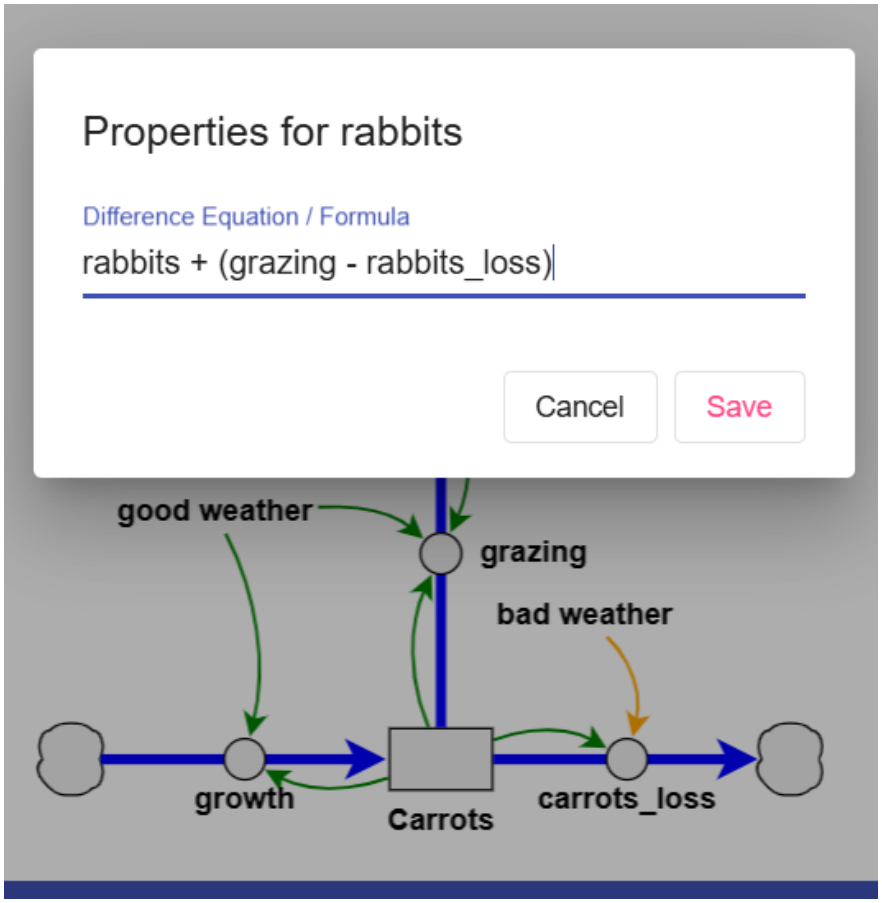

Figure 3: User interface for defining properties and mathematical formulas of a selected node (demonstration of a difference equation for a stock).

After visually constructing the model and defining its mathematical properties (illustrated in Figure 3), the platform converts the entered difference equations into declarative JSON structures, ready for simulation and semantic analysis. To ensure high responsiveness, the client Angular application performs asynchronous network requests through its built-in *HttpClient* module [27], providing immediate reactivity of the editor. While the actual calculation of time series is handled asynchronously by the *System Dynamic Solver* module, intelligent knowledge extraction is orchestrated by the synchronous phase of the semantic engine. When a user inputs a conceptual natural language query, the system vectorizes it in real-time via the local ONNX model and matches it against the indexes in the Qdrant vector database. The implementation of this process and the final result in the user interface are visually illustrated in Figure 4. The example shows the execution of **Scenario B** (Ecological domain), in which the semantic engine successfully identifies the relevant predator-prey model with a calculated cosine similarity of 73% and extracts its linearized text narrative directly from Qdrant's memory.

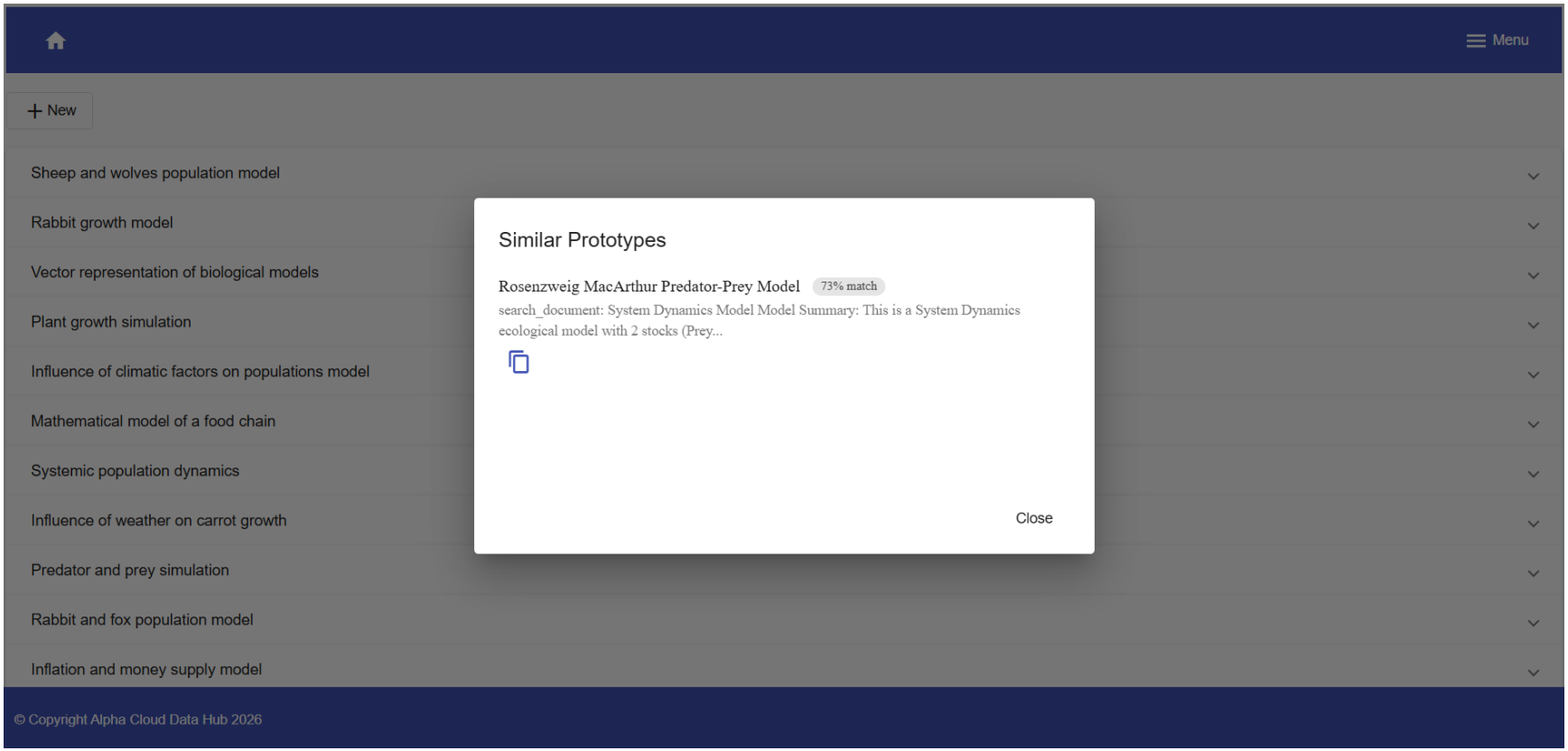

Figure 4: User interface of Alpha Cloud Data Hub, demonstrating the result of a semantic search (visualization of Scenario B).

Note: The client SPA provides an intuitive interface for accessing the semantic engine. This specific screen visualizes the execution of ***Scenario B*** *(Ecological domain, detailed in Chapter 5), where the system successfully identified the Rosenzweig-MacArthur Predator-Prey Model. The percentage expression of the cosine similarity (73% match) is clearly visible, exceeding the system-defined threshold of 0.72, along with a portion of the linearized text context (`search_document`) extracted directly from the Qdrant vector database.*

The architecture uses the Polyglot Persistence model [28], combining a document-oriented MongoDB database as the primary source of truth for the JSON graphs, and a specialized vector database Qdrant [29] for the semantic indexes. The process is strictly divided into two logical phases:

- **Asynchronous phase (Indexing):** Upon creating or updating a prototype, the web server saves the metadata in MongoDB and publishes an event in RabbitMQ. The isolated microservice *Hermes Processor* consumes the event, performs the semantic linearization, and calls the local *OnnxEmbeddingService* (`nomic-embed-text-v1.5` model) to generate a 768-dimensional vector. The vector is written to Qdrant via a high-performance gRPC protocol.
- **Synchronous phase (Search):** Triggered directly by the user. The query is converted into a vector in real-time, matched with the indexes in Qdrant (searching for the nearest neighbor via HNSW), and the obtained identifiers are then used to retrieve the full JSON documents from MongoDB. Using the MongoDB GUID as a Point ID in Qdrant provides natural Upsert semantics and prevents desynchronization.

Ensuring Fault Tolerance and Computational Isolation

A key advantage of this decentralized architecture is its built-in Fault Tolerance. Using RabbitMQ not only solves the UI blocking problem but also ensures data reliability through Message Acknowledgments and Durable Queues. In the event of a temporary crash in the computationally heavy *Hermes Processor* microservice (responsible for AI vectorization), the generated events are not lost but remain in the queue until successfully processed. In parallel, the simulation engine (*System Dynamic Solver*) is externalized as a fully isolated component in the application layer. It parses the mathematical expressions into Abstract Syntax Trees (AST) and executes numerical integration completely independently from the semantic search modules. This strict isolation guarantees that potential overloads in the Transformer model inference processes will not compromise the core functionality of building and simulating mathematical models. **The interaction between these components and the logical separation of the flows are visually illustrated in Figure 5.**

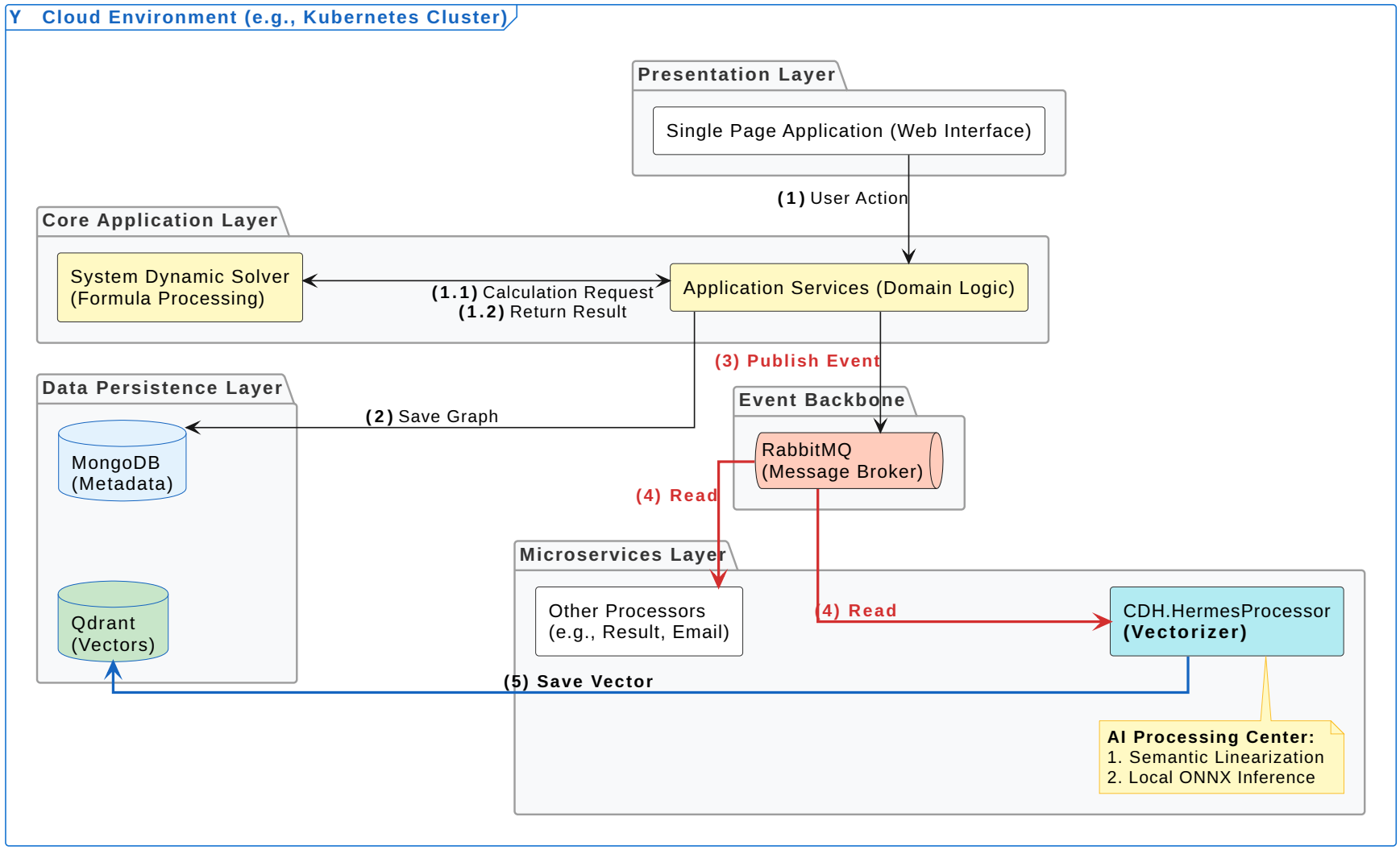

Figure 5: High-level system architecture in CDH: Synchronous search flows and asynchronous data semantic indexing flows.

Note: The diagram illustrates the strict separation between synchronous query processing and heavy asynchronous vectorization via RabbitMQ. The fully isolated computational logic is also shown.

Conclusions for Chapter 4

- **Distributed microservice architecture:** Designing the platform based on DDD principles and the event-driven model with RabbitMQ ensures full computational decoupling, eliminating the risk of UI blocking during heavy AI operations.
- **Polyglot Persistence:** Combining the document-oriented database MongoDB (as the primary source of truth for JSON graphs) with the specialized vector database Qdrant achieves an optimal balance between computational speed and avoiding excessive memory overload.
- **Local inference and security:** The strategic integration of a local ONNX Runtime session for generating embeddings eliminates network latency, protects the system from hidden technical debt, and guarantees full protection of industrial intellectual property.
- **Consistency and reactivity:** Using the MongoDB GUID as a direct Point ID in the vector index provides natural Upsert semantics during editing, and the browser-based Angular SPA guarantees high UX responsiveness in real-time.

Chapter 5. Experimental Validation and Analysis of Results

The experimental validation was conducted in an isolated containerized environment (*Docker*) to guarantee the reproducibility of the results. For this purpose, a specialized test corpus ("Gold Standard") of 63 system dynamics models with a high degree of structural and semantic heterogeneity was constructed. Table 1 presents the quantitative and thematic distribution of the models in the corpus.

Table 1: Thematic distribution of models in the test corpus

Subject Area (Domain)	**Number of Models**	**Share (%)**	**Avg. Stocks**
Ecology and Agriculture	12	19.0%	1.8
Demography and Social Systems	11	17.5%	1.5
Economics and Finance	10	15.9%	1.2
Biology and Medicine	9	14.3%	2.0
Ecology and Environment	7	11.1%	1.8
Infrastructure and Production	6	9.5%	2.3
Energy and Resources	4	6.3%	1.0
Abstract and Mathematical	4	6.4%	1.5
Total	**63**	**100%**	**1.6**

Quantitative and structural analysis of the test corpus (Table 1):
The thematic distribution in the corpus demonstrates balanced heterogeneity with a dominant share of ecological (19.0%) and demographic models (17.5%). A critical indicator for algorithmic verification is the topological complexity of the models (detailed in Appendix C through the parameters "Average number of nodes" and "Average number of links"), as well as the column "Avg. Stocks", as they define the breakpoints of the algebraic loops

during topological sorting. Their variation from 1.0 to 2.3 provides a broad base of non-linear equations with varying feedback density, guaranteeing objective testing of the simulation module (*System Dynamic Solver*) and the semantic linearization processes.

Validation of the Simulation Module

To verify the computational accuracy and cross-domain invariance of the *System Dynamic Solver* module, testing was conducted on a specialized "Gold Standard" of reference models. The aim is to prove that the platform correctly solves the specified systems of ordinary differential equations (ODEs) using the Forward Euler method. The validation covers various scientific domains and confirms the stability of the integrator under competing feedback loops, successfully generating the expected non-linear and oscillating behavior:

– **Financial model for bank liquidity management:** The system successfully simulates the dynamics of a central stock (Bank Liquidity), which is fed by a sinusoidal inflow and depleted by two balancing outflows (Lending and Reserve Allocation), proving the stability of the numerical integrator under economic scenarios.

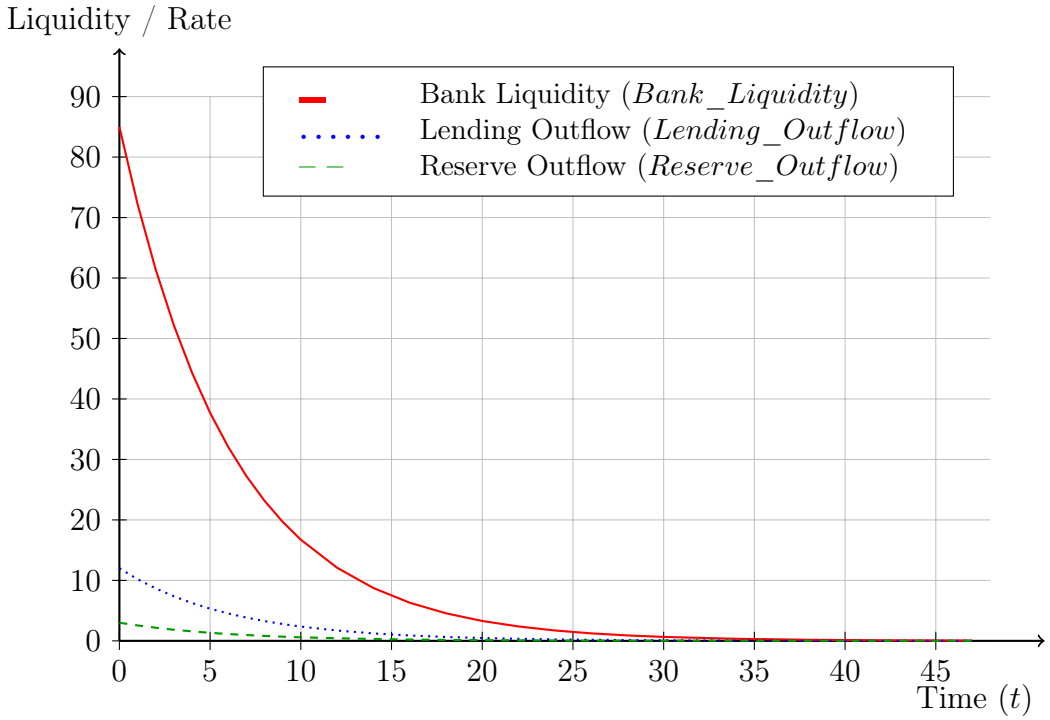

Figure 6: Simulation results from a reference financial model for bank liquidity management.

– **Lotka-Volterra Predator-Prey Ecological Model [30]:** The simulation successfully generates the coupled phase-shifted oscillations characteristic of the model. This confirms the computational module's ability to precisely handle competing feedback loops in non-linear population dynamics systems [31].

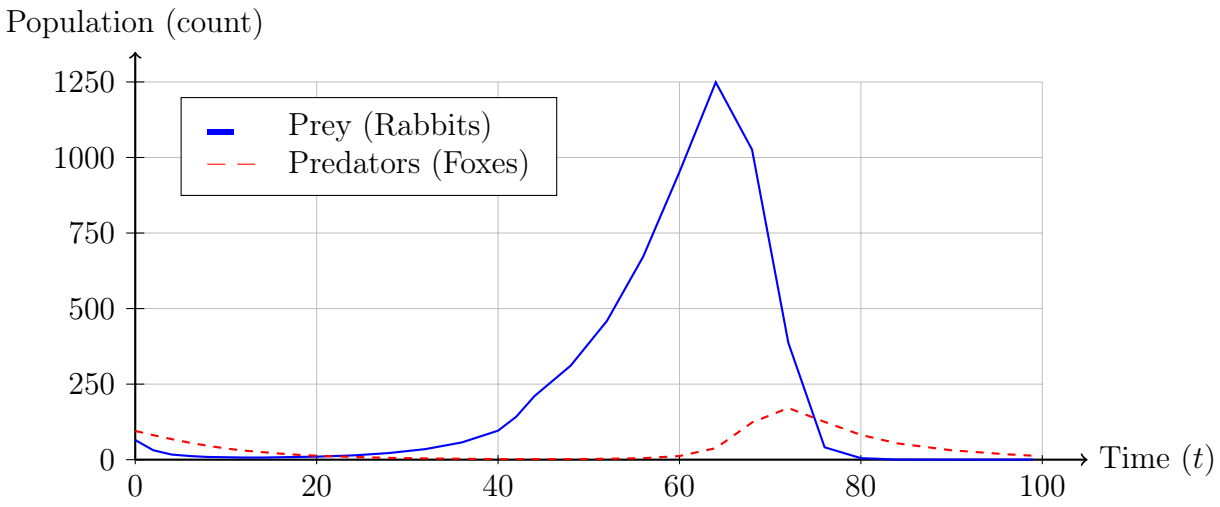

Figure 7: Simulation results from a Predator-Prey ecological model (Scenario B). The graph clearly illustrates the coupled phase-lagged oscillations, where the peak in the prey population leads to a subsequent exponential growth in predators.

Evaluation of Semantic Search

To realize an objective comparison of the proposed semantic technology against traditional lexical algorithms (PostgreSQL Keyword/Full-Text Search, Apache Lucene BM25 [32]), the application architecture was temporarily extended with control mechanisms (*Baseline Checks*). Figure 8 presents the architectural topology of the experiment. It illustrates how the same user query is routed in parallel: on one hand, to the proposed polyglot architecture (vector queries to *Qdrant* and metadata from *MongoDB*), and on the other, to the reference relational and full-text systems (*PostgreSQL* and *Apache Lucene*), which serve purely for benchmark analysis and an isolated control environment.

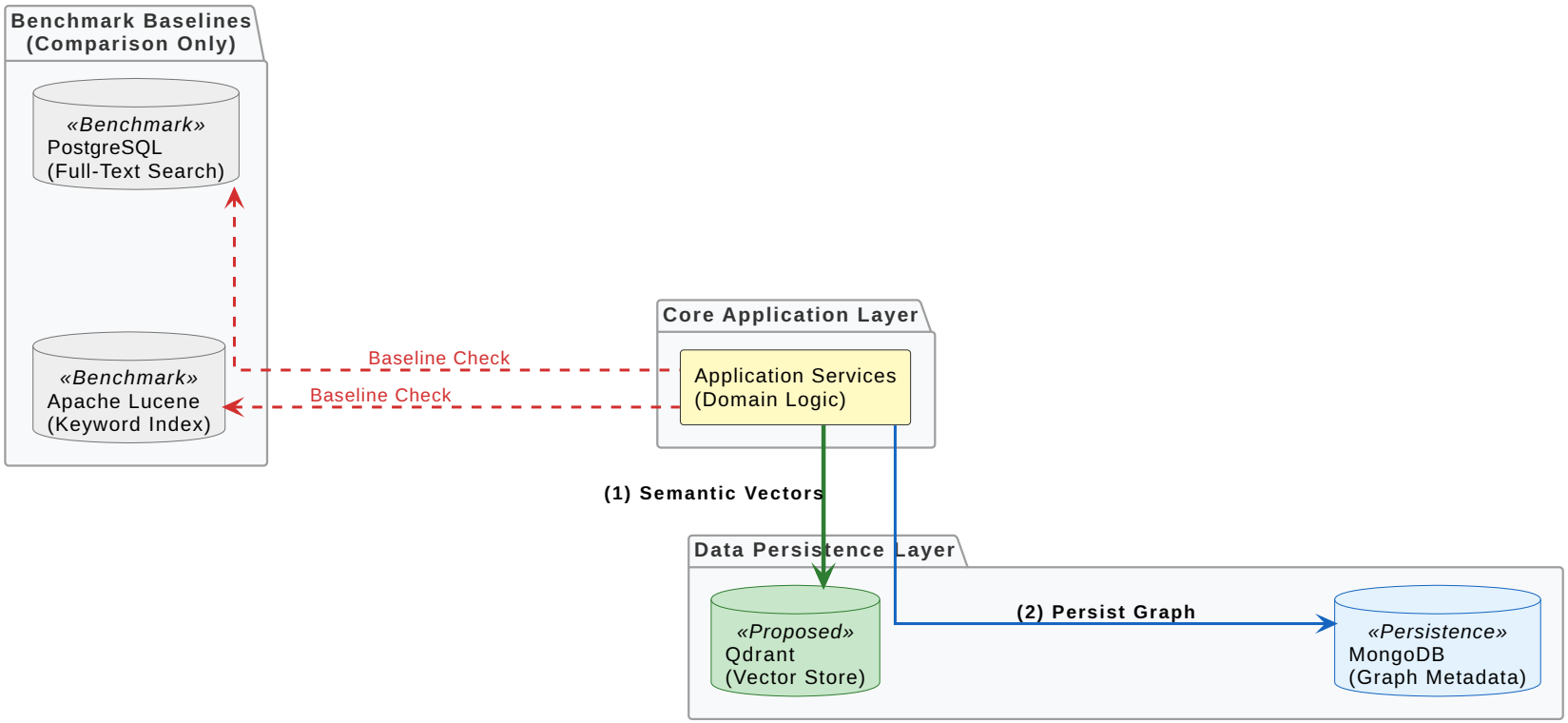

Figure 8: Architecture of the experimental setup: Proposed semantic module vs. baseline lexical systems for comparison.

Note: The diagram illustrates the parallel integration of PostgreSQL and Apache Lucene. In the context of the system, they function entirely as an isolated reference model (Baselines) for performance evaluation and have no relation to the actual production data flow in Alpha Cloud Data Hub.

To simulate realistic user searches, five control scenarios were defined, targeting the main archetypes of system dynamics:

- **Scenario A (Economics):** Query *"Dynamics of cash flows and resources in economic systems"*.
- **Scenario B (Ecology):** Query *"Rosenzweig-MacArthur model stability"*.
- **Scenario C (Epidemiology):** Query *"Climate influence on viral transmission simulation"*.
- **Scenario D (Genetics):** Query *"Weighted gene regulatory network modeling"*.
- **Scenario E (Demography):** Query *"Population dynamics with seasonal migration patterns"*.

The quantitative performance evaluation was conducted using standard information retrieval metrics (Precision, Recall, F1-Score, and Latency), whose methodological justification is detailed in the first chapter of the dissertation. To eliminate subjective bias, the relevance of the retrieved models to the conceptual queries was pre-established through a double-blind expert review. In this extended abstract, Scenario A (Economic domain) is detailed, as it most vividly illustrates overcoming the "vocabulary mismatch" problem, while the results from the other scenarios are summarized quantitatively to prove cross-domain consistency.

Qualitative Analysis of Information Noise (Scenario A)

When testing the conceptual economic query *"Dynamics of cash flows and resources in economic systems"*, traditional algorithms (Keyword Search) demonstrated complete contextual blindness. They retrieved models like *"Systemic population dynamics"* (from demography) and *"Extended Model of Predator-Prey Dynamics"* (from ecology) solely due to the matching of the word "dynamics". This generated 60% information noise (False Positives). The semantic module of *Alpha Cloud Data Hub* completely eliminated this problem, as the vector space associates structural topology with the concept of dynamics without requiring a lexical match.

Table 2: Comparative analysis for Economic domain (Scenario A)

Method	Results	Relevant	Precision	Recall	F1-Score	Latency
PostgreSQL Keyword Search	5	2	0.40	0.18	0.25	∼77.32 ms
PostgreSQL Scan	0	0	0.00	0.00	0.00	∼19.38 ms
PostgreSQL Full-Text Search	0	0	0.00	0.00	0.00	∼43.02 ms
Apache Lucene Substring Scan	5	2	0.40	0.18	0.25	∼276.75 ms
Apache Lucene BM25	5	2	0.40	0.18	0.25	∼23.66 ms
Semantic Search	**11**	**11**	**1.00**	**1.00**	**1.00**	**∼1891.86 ms**

The vector search analyzed the linearized graph topology and achieved 100% precision and recall (F1-Score = 1.00) at a cosine similarity threshold of $t = 0.72$, completely eliminating information noise.

Calibration of the Semantic Similarity Threshold (Cosine Similarity Threshold)

To ensure maximum precision in knowledge extraction and to justify the choice of the 0.72 threshold, an additional experiment (Sensitivity Analysis) was conducted. Since the vector space in the `nomic-embed-text-v1.5` model is extremely dense, small variations in the rejection threshold lead to drastic changes in search quality. Various filtration steps were applied to the control corpus, and the results are documented in Table 3.

Table 3: Optimization of F1-Score relative to the cosine similarity threshold (t)

Similarity Threshold (t)	**Precision**	**Recall**	**F1-Score**
$t \geq 0.60$	0.55	1.00	0.71
$t \geq 0.65$	0.78	1.00	0.88
$t \geq 0.70$	0.91	1.00	0.95
$t \geq 0.72$ **(Optimal)**	**1.00**	**1.00**	**1.00**
$t \geq 0.75$	1.00	0.81	0.89
$t \geq 0.80$	1.00	0.45	0.62

As evident from the data, at low threshold values ($t < 0.70$), the system allows numerous False Positives because the algorithm begins to associate abstractions that are too distant. On the other hand, excessively restrictive thresholds ($t > 0.75$) cause a sharp drop in Recall because the system starts rejecting valid structural analogies that have only minor variations in their topology. The local maximum of the harmonic mean is reached exactly at $t = 0.72$, which is why this value is defined as the optimal system boundary in the core of *Alpha Cloud Data Hub*.

Performance and Scalability Analysis

A large-scale stress test was conducted to profile computational complexity across graphs with different topologies (Table 4). For the stress test, three models with fundamentally different structural profiles were selected:

- **Model A (Finance):** A linear cash flow model (4 components, 5 links) serving as a Baseline.
- **Model B (Biology):** A complex model for cellular population dynamics (10 components, 13 links) with isolated feedback loops.
- **Model C (Ecology):** A plant growth model (6 components, 8 links) with a highly intertwined topology and simultaneous, overlapping algebraic loops.

Table 4: Comparative analysis of latency for adding, processing, and vectorizing models with different topological complexities

Metric / Parameter	**Model A (Finance)**	**Model B (Biology)**	**Model C (Ecology)**
Structural profile of the model			
Total components	4	10	6
Topological links	5	13	8
Generated context (Tokens)	554	1013	721
Server Processing (Synchronous phase)			
Formula Processing	2862.04 ms	2845.49 ms	12028.81 ms
MongoDB Operations	6.02 ms	6.48 ms	6.67 ms
RabbitMQ Publish	0.09 ms	0.13 ms	0.10 ms
Total Time (Backend)	**2999.51 ms**	**2998.86 ms**	**12152.72 ms**
AI Microservice Hermes (Asynchronous phase)			
Text Extraction	0.78 ms	0.81 ms	0.86 ms
Tokenization	38.81 ms	75.25 ms	50.25 ms
ONNX Model Inference	5621.65 ms	5248.61 ms	5423.35 ms
Mean Pooling	42.36 ms	65.10 ms	57.34 ms
Qdrant Upsert	12.68 ms	86.82 ms	28.21 ms
Total Time (Hermes)	**5719.24 ms**	**5479.75 ms**	**5562.75 ms**
Total System Time	**≈ 8.72 seconds**	**≈ 8.48 seconds**	**≈ 17.72 seconds**

The results prove that the density of overlapping algebraic loops (and not the number of nodes) causes an exponential jump in topological sorting time (over 12 seconds for Model C). The additional AI vectorization adds a constant $\sim$5.5 seconds. This empirically validates the architectural decision to offload semantic analysis into an asynchronous RabbitMQ queue. Despite the heavy indexing phase during writes (up to 18 seconds), the latency for **semantic search** (reads) remains stable at around $\sim$1.89 seconds, ensuring excellent responsiveness.

Conclusions for Chapter 5

- **Correctness of the simulation core:** Experimental verification proves that the simulation module precisely calculates deterministic mathematical models from various domains, perfectly reproducing their specific non-linear and oscillating behavioral descriptions.
- **Superiority of semantic search:** The semantic approach categorically outperforms traditional lexical methods by completely eliminating information noise for abstract conceptual queries and achieving maximum precision and recall at the optimal similarity threshold of $t = 0.72$.
- **Validation of asynchronous architecture:** Profiling the computational complexity confirms that algebraic loop density strains the backend, which practically justifies the architectural decision to offload the heavy Transformer vectorization into an asynchronous queue to preserve low search latency.

Conclusion

The conclusion provides a summary of the dissertation. It evaluates the degree to which the set tasks were fulfilled and formulates the main scientific-applied and applied contributions, among which are the creation of the semantic linearization algorithm and the implementation of the *Alpha Cloud Data Hub* cloud prototype. Technical challenges and directions for future development are outlined:

- **Infrastructure optimization:** Implementing hardware GPU acceleration for microservices to reduce vectorization time to under 200 ms, multi-layer caching (Redis), and elastic horizontal scaling (Kubernetes [33]).
- **Algorithmic upgrades:** Transitioning to multi-vector indexing architectures (ColBERT-type) by decomposing the graph into logical segments to overcome the "information bottleneck" when aggregating massive industrial models into a single vector.

Scientific and Scientific-Applied Contributions

Based on the theoretical and practical research reflected in the dissertation, the following four main contributions have been defined:

Scientific-Applied Contributions

1. An innovative methodology and algorithm for the semantic linearization of cyclic graph structures have been created, specified through the System Dynamics formalism. The introduction of cyclicity markers (explicit loop tokenization) allows the preservation of causal topology in the form of a dense vector context suitable for Natural Language Processing (NLP) analysis.
2. A hybrid, polyglot-based architectural model has been developed, which integrates an Event-Driven microservice environment (RabbitMQ) with specialized vector search (Qdrant). This model solves the problem of computational blocking during heavy AI processing (Inference) by isolating the synchronous read process from the asynchronous knowledge creation pipeline.

Applied Contributions

3. The software prototype *Alpha Cloud Data Hub* has been designed, implemented, and deployed. The prototype validates the effectiveness of the proposed theoretical methods in practice, providing a web-based ecosystem for visual modeling, simulation, and intelligent model discovery through asymmetric semantic search.

Justifying (Empirical) Scientific-Applied Contributions

4. The advantages of the semantic vector approach over traditional lexical algorithms have been empirically justified in extracting knowledge from dynamic models. Through experimental analysis, it has been confirmed that the proposed architecture successfully addresses the "vocabulary mismatch" problem, achieving 100% Recall for conceptual queries and significantly reducing the information noise (False Positives) inherent in full-text search.

The relationships between the contributions, objectives, tasks, their location in the dissertation, and the associated publications are summarized in Table 5.

Table 5: Relationship between contributions, tasks, dissertation chapters, and publications

Contribution	Type	Tasks	Chapter	Publications
1	Scientific-Applied	Task 2	Chapter 3	Pub. 3
2	Scientific-Applied	Tasks 1 & 3	Chapter 4	Pubs. 1, 2 & 3
3	Applied	Tasks 3 & 4	Chapters 4 & 5	Pubs. 1, 2 & 3
4	Scientific-Applied	Task 4	Chapter 5	Pub. 3

Publications and Approbation of the Results

Publications on the Topic of the Dissertation

The main theoretical postulates, architectural solutions, and experimental results of the dissertation are reflected in 3 scientific publications, which cover all the formulated scientific-applied and applied contributions:

1. **Kyurkchiev, P.**, *Architecture of Web Based System for Symbol Programming a Real Process*, (2015), Scientific Conference "Innovative ICT: Research, Development and Application in Business and Education", Hisar, ISBN: 978-954-8852-56-7.

2. **Kyurkchiev, P.**, *Cloud Computing of Incoming Data - Analyze and Storage of Dynamic Mathematical Models*, (2017), Announcements of Union of Scientists - Sliven, ISSN: 1311-2864.

3. **Kyurkchiev, P.**, Iliev, A., Kyurkchiev, N., *Semantic Search for System Dynamics Models Using Vector Embeddings in a Cloud Microservices Environment*, (2026), Future Internet 18(2), 86, DOI: 10.3390/fi18020086. **(Web of Science, IF=4.6, Q2)**

Participation in Scientific Forums (Approbation)

The research results were presented and discussed before the scientific community in the form of two papers/reports at the following scientific forums:

– Scientific Conference "Innovative ICT: Research, Development and Application in Business and Education" (2015, Hisarya). Presented a paper on the topic *"Architecture of a Web-Based System for Symbolic Programming of Real Processes"*.

– Scientific Session of the Union of Scientists in Bulgaria – Sliven Branch (2017, Sliven). Presented a paper accompanying the publication *"Cloud Computing of Incoming Data - Analyze and Storage of Dynamic Mathematical Models"*.

Funding and Connection to Scientific Projects

The present dissertation summarizes results achieved within the framework of research activities supported by the "Scientific Research" Fund at Plovdiv University "Paisii Hilendarski". Part of the research and prototype implementations were funded by the following projects:

– Project **IT15-FMIIT-004 "Research in the Field of Innovative ICT Oriented towards Business and Education"** at the Scientific Research Fund (NPD) of Plovdiv University "Paisii Hilendarski".

– Project **MU17-FMI-007 "ICT in Support of Scientific Research in Mathematics and Informatics and Their Applications"** at the Scientific Research Fund (NPD) of Plovdiv University "Paisii Hilendarski".

ACKNOWLEDGMENTS

First and foremost, I express my sincere gratitude to my scientific supervisor, **Prof. Anton Iliev, PhD**, for the trust placed in me, the opportunities provided, his patience, and his unwavering support throughout the entire process of my work on the dissertation.

I would like to express my special appreciation for the professional support, valuable scientific guidance, and methodological advice of **Prof. Asen Rahnev, PhD** and **Prof. Nikolay Pavlov, PhD**.

I am grateful to all my colleagues at the Department of Computer Technologies at the Faculty of Mathematics and Informatics of Plovdiv University "Paisii Hilendarski" for the excellent academic environment, the fruitful discussions, and the collegiality, which in various ways contributed to the realization of this work.

Last but not least, I express my deepest gratitude to my family for their unwavering support and understanding in all my endeavors.

It has been a true joy to walk this path, driven by my love for science, making this dissertation the most beautiful and valuable result of this inspiring stage of my life.

References

[1] P. Mell and T. Grance. *The NIST definition of cloud computing*. Tech. rep. National Institute of Standards and Technology, 2011. DOI: 10.6028/NIST.SP.800-145.

[2] H. A. Simon. "Designing organizations for an information-rich world". *Computers, communications, and the public interest*, **72**, 1971, p. 37.

[3] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. "The vocabulary problem in human-system communication". *Communications of the ACM*, **30**(11), 1987, pp. 964–971. DOI: 10.1145/32206.32212.

[4] Z. Feng et al. "CodeBERT: A Pre-Trained Model for Programming and Natural Languages". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, 2020, pp. 1536–1547. DOI: 10.18653/v1/2020.findings-emnlp.139.

[5] D. Guo et al. "GraphCodeBERT: Pre-training Code Representations with Data Flow". In: *International Conference on Learning Representations (ICLR)*. 2021. URL: https://openreview.net/forum?id=jLoC4ez43PZ (visited on 02.07.2026).

[6] J. W. Forrester. "Industrial Dynamics: A Major Breakthrough for Decision Makers". *Harvard Business Review*, **36**, 1958, pp. 37–66.

[7] J. D. Sterman. *Business dynamics: systems thinking and modeling for a complex world*. Boston, MA: Irwin/McGraw-Hill, 2000. ISBN: 978-0072389159.

[8] G. Gordon. "A general purpose systems simulator". In: *Proceedings of the December 12-14, 1961, eastern joint computer conference*. ACM. 1961, pp. 87–104. DOI: 10.1145/1460690.1460701.

[9] D. H. Meadows. *Thinking in systems: A primer*. Chelsea Green Publishing, 2008. ISBN: 978-1603580557.

[10] A. Borshchev. *The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6*. AnyLogic North America, 2013. ISBN: 978-0989573177.

[11] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta. "Comparison of JSON and XML data interchange formats: a case study". In: *Caine*. Vol. 9. 2009, pp. 157–163.

[12] J. R. Ullmann. "An algorithm for subgraph isomorphism". *Journal of the ACM (JACM)*, **23**(1), 1976, pp. 31–42. DOI: 10.1145/321921.321925.

[13] K. Riesen. *Structural pattern recognition with graph edit distance*. Cham: Springer, 2015. ISBN: 978-3319227238.

[14] J. C. Butcher. *Numerical methods for ordinary differential equations*. 3rd. John Wiley & Sons, 2016. ISBN: 978-1119121503.

[15] N. Pavlov. "Efficient Matrix Multiplication Using Hardware Intrinsics and Parallelism with C#". In: *Proceedings of the International Scientific Conference REMIA*. Plovdiv University "Paisii Hilendarski". 2021. ISBN: 978-619-202-711-7.

[16] G. Salton, A. Wong, and C.-S. Yang. "A vector space model for automatic indexing". *Communications of the ACM*, **18**(11), 1975, pp. 613–620. DOI: 10.1145/361219.361220.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (visited on 02.07.2026).

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Vol. 1. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[19] Y. A. Malkov and D. A. Yashunin. "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **42**(4), 2018, pp. 824–836. DOI: 10.1109/TPAMI.2018.2889473.

[20] R. Tarjan. "Depth-first search and linear graph algorithms". *SIAM journal on computing*, **1**(2), 1972, pp. 146–160. DOI: 10.1137/0201010.

[21] E. Evans. *Domain-driven design: tackling complexity in the heart of software*. Boston, MA: Addison-Wesley Professional, 2003. ISBN: 978-0321125217.

[22] J. Lewis and M. Fowler. "Microservices: a definition of this new architectural term". *MartinFowler.com*, 2014. URL: https://martinfowler.com/articles/microservices.html (visited on 02.07.2026).

[23] S. Newman. *Building microservices: designing fine-grained systems*. O'Reilly Media, Inc., 2015. ISBN: 978-1491950357.

[24] M. Kleppmann. *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media, Inc., 2017. ISBN: 978-1449373320.

[25] P. Dobbelaere and K. S. Esmaili. "Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry paper". In: *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*. 2017, pp. 227–238. DOI: 10.1145/3093742.3093908.

[26] M. S. Mikowski and J. C. Powell. *Single page web applications: JavaScript end-to-end*. Simon and Schuster, 2013. ISBN: 978-1617290756.

[27] A. Freeman. *Pro Angular 9: Build Powerful and Dynamic Web Apps*. Berkeley, CA: Apress, 2020. ISBN: 978-1484259979. DOI: 10.1007/978-1-4842-5998-6.

[28] P. J. Sadalage and M. Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Boston, MA: Addison-Wesley Professional, 2012. ISBN: 978-0321826626.

[29] Y. Han, C. Liu, and P. Wang. "A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge". *arXiv preprint arXiv:2310.11369*, 2023. DOI: 10.48550/arXiv.2310.11369.

[30] A. J. Lotka. "Analytical note on certain rhythmic relations in organic systems". *Proceedings of the National Academy of Sciences*, **6**(7), 1920, pp. 410–415. DOI: 10.1073/pnas.6.7.410.

[31] N. Pavlov, A. Golev, A. Iliev, A. Rahnev, and N. V. Kyurkchiev. "On the Kumaraswamy-Dagum-Log-Logistic sigmoid functions with applications to population dynamics". *Biomath Communications*, **5**(2), 2018. DOI: 10.11145/bmc.2018.03.247.

[32] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. "Okapi at TREC-3". In: *Proceedings of the Third Text REtrieval Conference (TREC-3)*. Vol. 500. 225. National Institute of Standards and Technology. 1994, pp. 109–126.

[33] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes. "Borg, Omega, and Kubernetes". *Communications of the ACM*, **59**(5), 2016, pp. 50–57. DOI: 10.1145/2890784.