ПЛОВДИВСКИ УНИВЕРСИТЕТ
„ПАИСИЙ ХИЛЕНДАРСКИ“
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
КАТЕДРА „КОМПЮТЪРНИ ТЕХНОЛОГИИ“

ПАВЕЛ АЛЕКСАНДРОВ КЮРКЧИЕВ

МЕТОДИ И АРХИТЕКТУРНИ ПОДХОДИ ЗА СЕМАНТИЧНО ИНДЕКСИРАНЕ И СИМУЛАЦИЯ НА ДИНАМИЧНИ ПРОЦЕСИ В ОБЛАЧНА СРЕДА

АВТОРЕФЕРАТ

на дисертационен труд
за присъждане на образователната и научна степен **„Доктор“**
в област на висше образование: 4. Природни науки, математика и информатика;
професионално направление: 4.6. Информатика и компютърни науки;
докторска програма: „Информатика“

Научен ръководител: проф. д-р Антон Илиев

гр. Пловдив
2026 г.

Дисертационният труд е обсъден и насочен за защита на разширено заседание на катедра „Компютърни технологии“ при Факултета по математика и информатика на ПУ „Паисий Хилендарски“.

Дисертационният труд съдържа 140 страници, в които са представени 18 фигури и 12 таблици. Използваната литература включва 146 източника.

Списъкът на авторските публикации се състои от 3 заглавия.

Защитата на дисертационния труд ще се състои на **.....** в ....., гр. Пловдив.

Материалите по защитата са на разположение на интересуващите се в секретариата на ФМИ, Нова сграда на ПУ, каб. 330, всеки работен ден от 8:30 до 17:00 часа.

Научно жури:

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

Автор: Павел Александров Кюркчиев
Заглавие: Методи и архитектурни подходи за семантично индексиране и симулация на динамични процеси в облачна среда

Пловдив, 2026 г.

Съдържание

Обща характеристика на дисертационния труд

Актуалност на проблема

Развитието на компютърните науки и преходът от изолирани настолни приложения към разпределени облачни системи (Cloud Computing) [1] трансформираха фундаментално методите за моделиране и симулация. Въпреки навлизането на концепцията „Симулация като услуга“ (Simulation-as-a-Service), съвременните платформи са изправени пред „Парадокса на изобилието“ [2] наличие на огромна изчислителна мощ, но неефективно управление на знанието.

Традиционните системи функционират като „черни кутии“, съхраняващи логиката в затворени или бинарни формати. Търсенето в тях разчита на класически лексикални методи (Keyword Search), които са неефективни при откриване на динамични модели поради проблема с „лексикалното несъответствие“ (Vocabulary Mismatch) [3].

Интеграцията на методи от изкуствения интелект, в частност големите езикови модели (LLMs) и векторното търсене, предоставя уникална възможност за „отключване“ на това скрито знание. Съществуващите подходи към момента се фокусират предимно върху синтактичния анализ на програмен код [4,5], пренебрегвайки дълбоката математическа и каузална семантика, която стои в основата на системнодинамичните модели. Именно тази научна празнота мотивира търсенето на нови архитектурни парадигми за транслация на графи в машинно-разбираем контекст.

Обект и предмет на изследването

Обект на изследването са софтуерните системи за моделиране и симулация на динамични процеси, разгледани и реализирани чрез утвърдения формализъм на Системна динамика (СД) [6,7], функциониращи в съвременна разпределена облачна среда.

Предмет на изследването са методите за архитектурно проектиране, алгоритмите за числена симулация на системи от диференциални уравнения и подходите за семантичен анализ, линеаризация и векторизация на математически модели чрез Трансформър-базирани езикови архитектури.

Цел на дисертационния труд

Основната цел на дисертационния труд е да се изследват и приложат методите на формализма „Системна динамика“ със съвременни технологии за семантичен анализ и векторно търсене в облачна среда, и чрез реализация на прототип да се валидира методология за визуално моделиране, числена симулация и интелигентно извличане на динамични процеси.

Работна хипотеза

В дисертационния труд се издига хипотезата, че интегрирането на микросървисна облачна архитектура с методи за векторен семантичен анализ ще позволи преодоляване

на проблема с „лексикалното несъответствие“ (vocabulary mismatch) при динамичните модели. Това ще повиши прецизността (Precision) и пълнотата (Recall) на търсене чрез преход от синтактично сравнение (по етикети на променливи) към семантично съпоставяне (по топологична и функционална структура), превръщайки математическата същност на моделите в универсален, векторизиран език, който алгоритмите могат ефективно да анализират и клъстеризират.

Задачи на изследването

За постигането на поставената цел са дефинирани и решени следните основни задачи:

1. **Анализ на състоянието и дефиниране на архитектурен модел:** Сравнителен анализ на подходите за моделиране, изследване на методите за автоматично извличане на уравнения и дефиниране на мащабируема микросървисна архитектура, интегрираща симулационен модул с векторна база данни.

2. **Разработване на методология за трансформация и търсене:** Дефиниране на алгоритъм за семантична линеаризация на графови модели (DFS обхождане и експлицитно откриване на цикли) и разработване на стратегия за сериализация към богат текстов контекст, оптимизиран за NLP модели.

3. **Проектиране и софтуерна реализация на експериментална среда:** Реализация на модулна облачна архитектура (*Alpha Cloud Data Hub*) чрез Domain-Driven Design (DDD), включваща интерактивен потребителски интерфейс и изолирани бекенд модули за числена симулация и семантично индексиране в реално време.

4. **Експериментална валидация и анализ:** Верификация на симулационния модул чрез сравнение с еталонни модели (екологичен модел Хищник-Жертва, финансов модел за банкова ликвидност) и количествена оценка на ефективността на семантичното търсене спрямо традиционните лексикални подходи.

Методология на изследването

За постигане на целите е приложен комплексен интердисциплинарен подход:

– **Конструктивен и дедуктивен подход:** Изграждане на концептуален архитектурен модел и материализирането му чрез иновативен софтуерен прототип (*Alpha Cloud Data Hub*), който служи като експериментален полигон.

– **Алгоритмична транслация:** Разработване на специализирана методология за семантична линеаризация (Graph Traversal и Cycle Detection), осигуряваща връзката между графовата математическа логика и алгоритмите за обработка на естествен език.

– **Сравнителен анализ и експериментална валидация:** Количествена оценка чрез стандартни метрики (Precision, Recall, F1-Score, Latency) върху специално формиран изолиран тестов корпус от 63 модела и съпоставяне на резултатите с утвърдени лексикални методи за търсене.

– **Полиглотно съхранение на данни (Polyglot Persistence):** Комбиниране на документно-ориентирани (NoSQL/MongoDB) и векторни бази данни (Qdrant) с NLP технологии (BERT/ONNX), гарантиращо висока производителност и отказоустойчивост.

Обем и структура на дисертацията

Дисертационният труд е написан на български език и е структуриран в обем от 140 страници. Разпределен е в увод, пет същински глави, заключение, списък с използваната литература (146 източника) и три приложения. Трудът е богато илюстриран с 18 фигури, 12 таблици и множество листинги с програмен код, които нагледно подкрепят направените научни изводи.

В логическо и съдържателно отношение дисертацията следва строга дедуктивна последователност:

– **Уводът** мотивира актуалността на проблема с управлението на знания в симулационните среди и дефинира изследователската хипотеза, целите и задачите на изследването.

– **Глава 1** прави критичен обзор на подходите за моделиране и софтуерните платформи, като дефинира технологичния вакуум и обосновава нишата за разработка на облачна екосистема.

– **Глава 2** изгражда теоретичния фундамент, свързвайки математическия апарат на Системната динамика със съвременните Трансформър архитектури и векторни пространства.

– **Глава 3** представя основния научно-приложен принос – авторската методология и алгоритмите за семантична линеаризация на циклични графи, оптимизирани за анализ от изкуствен интелект.

– **Глава 4** описва практическото софтуерно проектиране на микросървисната платформа *Alpha Cloud Data Hub* и интегрираната стратегия за полиглотно съхранение на данни.

– **Глава 5** съдържа емпиричната валидация на системата чрез стрес-тестове върху специализиран корпус, доказвайки количествено превъзходството на семантичния подход при търсене.

– **Заключението и Приложенията** обобщават научните приноси, очертават перспективи за развитие и допълват труда с технически спецификации, API документация и отворени данни.

Кратко съдържание на дисертационния труд

Увод

Първата глава въвежда в проблематиката на съвременното компютърно моделиране, където се наблюдава ясна тенденция на миграция от локални, настолно-базирани решения към разпределени облачни архитектури и предоставяне на „Симулация като услуга“ (Simulation-as-a-Service). Дефиниран е основният проблем, наличието на огромна изчислителна мощ за изпълнение на модели, съчетано с невъзможност за тяхното интелигентно откриване поради феномена „лексикално несъответствие“ (Vocabulary Mismatch). Съвременният инженер е в позицията на библиотекар, който разполага с милиони книги, но няма каталог и трябва да отваря всяка една поотделно, за да разбере нейното съдържание. Традиционните търсещи машини индексират единствено метаданните, докато вътрешната логика на моделите остава скрита в затворени формати („черни кутии“). Формулирани са целта, задачите, обектът, предметът и работната хипотеза на изследването, базирани на интеграцията между математическия формализъм на Системната динамика (СД) и съвременните технологии за Обработка на естествен език (NLP) и векторно търсене.

Глава 1. Анализ на предметната област и съществуващи решения

Увода представя критичен преглед на еволюцията в моделирането и сравнителен анализ на основните подходи:

– **Дискретно-събитийно моделиране (DES) [8]:** Подходящо за оперативно управление на опашки и ресурси, но с ниско ниво на абстракция.

– **Агентно-ориентирано моделиране (ABM):** Подход „отдолу-нагоре“ (bottom-up), ефективен за възникващо поведение, но изискващ тежък процедурен код (if-then-else логика).

– **Системна динамика (SD):** Макро-перспектива (top-down), базирана на интегрални уравнения и непрекъснато време [9]. Математическата ѝ чистота я прави идеална за транслация и семантичен анализ.

Критичен анализ на съществуващите платформи и формати:

– **Традиционни системи (Stella, AnyLogic [10], GoldSim):** Те предлагат огромна изчислителна мощ, но страдат от архитектурна остарялост. Функционират като монолитни настолни приложения и използват затворени формати (Java Bytecode, Binary), което ги превръща в непроницаеми „черни кутии“ за външни търсачки.

– **Формати за обмен (XMILE, XML, FMI, Modelica):** Макар да осигуряват преносимост и прецизна симулация, те са „семантично слепи“ и генерират значителен „синтактичен шум“ спрямо по-леки съвременни формати като JSON [11]. Структурата им е изключително разтеглена (verbose), което бързо изчерпва ограничения прозорец на внимание (Context Window) на съвременните Трансформър модели.

Доказва се също, че класическите детерминистични алгоритми за търсене в графи (като Изоморфизъм на подграфи [12] и Graph Edit Distance [13]) страдат от експоненциална сложност $\mathcal{O}(N!)$ и са неприложими за търсене в реално време в мащабируема облачна среда.

За преодоляване на тези дефицити, системата *Alpha Cloud Data Hub* въвежда лек, семантично плътен **JSON формат**, оптимизиран специфично за NLP анализ. Този преход от процедурен код към декларативни данни е ключов, тъй като позволява на алгоритмите да четат математическата топология без нужда от сложен синтактичен разбор (parsing). Всичко това дефинира изследователската ниша: създаване на интелигентна облачна система, която извлича и управлява знание, а не просто изпълнява симулации.

Изводи по Глава 1

- **Капсулираност на платформите:** Традиционните среди (Stella, AnyLogic) третират моделите като „черни кутии“, ограничавайки семантичната прозрачност и повторното им използване.
- **Синтактичен шум:** Отворените формати (XML, XMILE) притежават синтактична многословност, която бързо изразходва контекстния прозорец на Трансформър моделите.
- **Ограничения при графите:** Класическите алгоритми за сравнение на графи имат експоненциална сложност $\mathcal{O}(N!)$ и са немащабируеми за работа в реално време.
- **Лексикален провал:** Търсещите машини (BM25) генерират шум и се провалят при концептуални заявки заради проблема с лексикалното несъответствие (*Vocabulary Mismatch*).
- **Изследователска ниша:** Налице е технологичен вакуум за *cloud-native* екосистема, свързваща Системната динамика с латентните векторни пространства.

Глава 2. Математически формализъм и теоретични основи

В тази част е изложен математическият апарат, върху който стъпва системата. Системната динамика се формализира чрез диаграми на запаси и потоци (Stock and Flow Diagrams). Връзката между запаса и нетния поток се описва чрез интегралното уравнение:

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

За числено решаване на тези уравнения в облачната архитектура са анализирани класическите методи. Разгледан е методът на Рунге-Кута от 4-ти ред (RK4) като индустриален стандарт за прецизност:

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

За нуждите на *Alpha Cloud Data Hub* и семантичната транслация обаче е обоснован изборът на Явния метод на Ойлер (Forward Euler) [14] като базов интегратор, поради неговата изчислителна простота, възможност за паралелизация и хардуерна оптимизация [15], и възможност за лесно проследяване на причинно-следствените връзки:

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

За целите на интелигентното търсене се въвежда концепцията за векторни представяния (Vector Embeddings) [16]. Обектите се преобразуват в непрекъснати вектори в многомерно пространство $\mathbb{R}^n$ чрез функция за изобразяване $f : Object \to \mathbb{R}^n$. Семантичната близост се изчислява чрез Косинусова прилика (Cosine Similarity):

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

Обосновава се използването на Трансформър архитектури (BERT) [17,18] и формата ONNX (Open Neural Network Exchange) за хардуерна агностичност и локално изпълнение. Разгледана е архитектурата на векторните бази данни и алгоритъма HNSW (Hierarchical Navigable Small World) [19], който редуцира сложността на търсене от линейна $\mathcal{O}(N)$ до логаритмична $\mathcal{O}(\log N)$.

Изводи по Глава 2

- **Топология на Системната динамика:** Разграничението между запаси и потоци осигурява декларативна структура, идеална за автоматизиран транслационен анализ.
- **Числен Solver:** Методът на Ойлер съчетава изчислителна простота и семантична прозрачност на каузалните връзки в облачна среда.
- **Векторизация чрез BERT:** Преобразуването в ембединги преодолява ръчното създаване на онтологии и дава устойчивост към терминологични вариации.
- **Оптимизация чрез ONNX:** Стандартът гарантира хардуерна агностичност и квантуване за ефективно изпълнение в микросървиси с ограничени ресурси.
- **Логаритмично търсене:** Векторните бази данни (Qdrant) с HNSW редуцират сложността на търсене до логаритмична $\mathcal{O}(\log N)$, осигурявайки мащабируемост.

Глава 3. Методология за трансформация, симулация и интелигентно търсене

Главата дефинира алгоритмичната рамка на системата *Alpha Cloud Data Hub*. Процесът на транслация към изпълним код започва с топологично сортиране на графа за определяне на реда на изпълнение (Evaluation Order), третирайки запасите като точки на прекъсване на алгебричните цикли.

Основният научен принос в тази глава е разработената методология за **„Семантична линеаризация“**. Целта ѝ е да преодолее фундаменталната разлика между графовата топология на моделите и линейната структура, изисквана от езиковите модели (LLMs). Дефинира се математическа функция за линеаризация $\mathcal{L} : G \to S$, която конструира редицата от токени S чрез конкатенация на локални семантични пътища:

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

Извършен е теоретичен анализ между стратегиите за обхождане. Обхождането в ширина (BFS) е отхвърлено, тъй като причинява *контекстуална фрагментация*. Избрано е Обхождане в дълбочина (DFS) [20], тъй като то запазва каузалната верига ($A \to B \to C$), осигурявайки константно разстояние $\Delta t = \mathcal{O}(1)$ между концепциите. Това оптимизира изчисляването на механизма за само-внимание (Self-Attention) в Трансформър моделите.

Експлицитна токенизация на циклите и алгоритмична реализация

При наличие на затворени контури (Feedback Loops), стандартните алгоритми тихо прекратяват обхождането, което води до загуба на критично знание. Въведената методология генерира специален семантичен маркер `[CYCLE_DETECTED]`, който действа като указател за NLP модела.

За да се гарантира предвидимо време за изпълнение в облачната микросървисна среда и да се предотврати препълване на паметта (Out-Of-Memory), алгоритъмът въвежда ограничител на дълбочината на рекурсията ($D_{max} = 10$). Логиката на процеса е реализирана чрез следния рекурсивен подход:

```
// Вход:  Граф G = (V, E)
// Изход: Стек за изпълнение S

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

    // Стъпка 1: Маркиране на Stocks като базови известни()
    foreach node in G.Nodes:
        if node.Type == "Stock":
            visited.Add(node)

    // Стъпка 2: Обхождане на останалите възли (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: Псевдокод на алгоритъма за топологично сортиране

Тази програмна реализация гарантира, че пространствената сложност остава строго ограничена до $\mathcal{O}(D_{max})$, а генерираният семантичен контекст остава в рамките на позволения прозорец на NLP модела. Сравнението между наивния и семантичния подход илюстрира ясно предимството на линеаризацията:

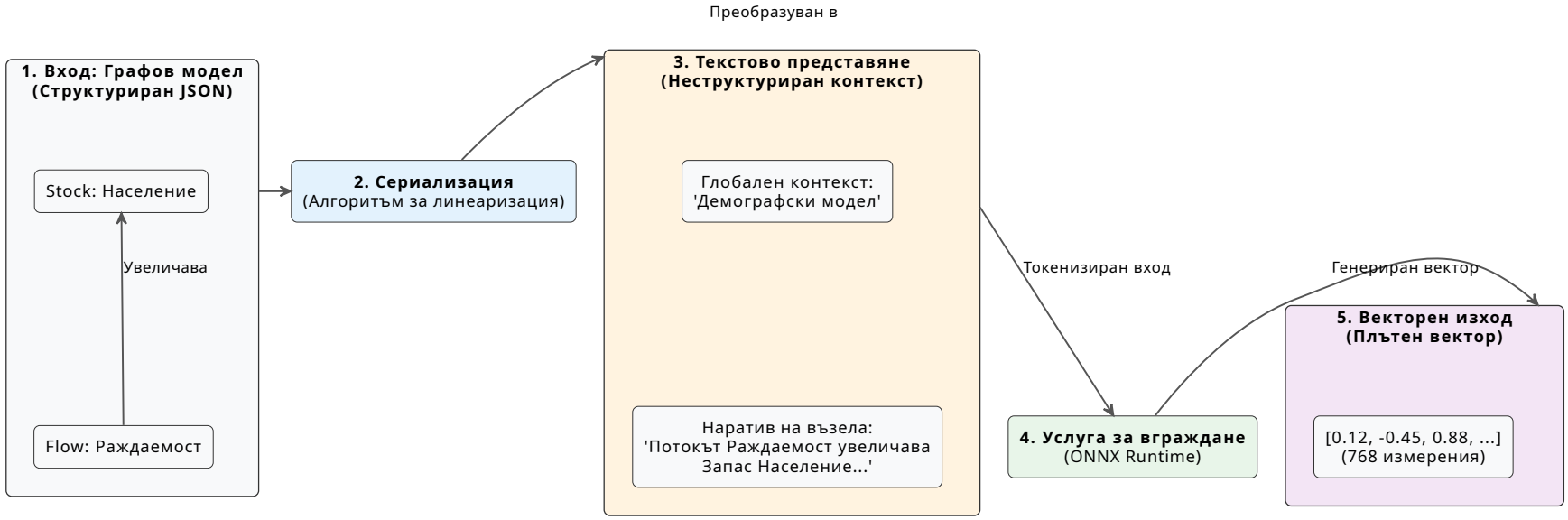
> *Наивно изброяване:* `Market_Demand, Revenue_Inflow, Cash_Reserves...`
> *Семантична линеаризация:* `? Market_Demand via influence` $\rightarrow$
> `? Revenue_Inflow via flow_to` $\rightarrow$ `? Cash_Reserves [cycle detected]`

Извличане на глобален контекст и агрегиране

Докато семантичната линеаризация се фокусира върху микро-структурата на причинно-следствените връзки, качественият NLP анализ изисква и дефиниране на макро-рамка. Без нея, векторният модел може да загуби ориентация в многомерното пространство (например, да не различи дали терминът „поток“ се отнася за хидрологичен процес или за финансов трансфер). За тази цел алгоритъмът генерира „семантична котва“ (Semantic Anchor), базирана на метаданните на модела, включваща тип на модела, научен домейн и структурна статистика (брой възли и връзки).

Във финалната фаза резултатите се синтезират в единен документ, годен за векторизация. Процесът на агрегиране не е просто конкатенация, а структурирано подреждане в три слоя: (1) Контекстуална рамка; (2) Структурен наратив (пътищата от DFS обхождането); и (3) Категориен индекс (лексикално подсигуряване). Това многослойно структуриране следва логиката на четене на Трансформър моделите. **Цялостният архитектурен поток на обработка от структурирания JSON модел до финалния векторен изход е нагледно илюстриран на Фигура 1.**

За оптимизация на търсенето се прилагат алгоритмична оптимизация (HNSW), Стратегия на единния векторен запис (SVR) за цялостно сравняване на концептуални структури и Инжектиране на домейн контекст (Prompt Injection), при което към потребителската заявка системно се добавя префикс: `"Represent this sentence for searching System Dynamics models: "`.



Фигура 1: Архитектурен поток на обработка (Pipeline) на процеса за семантична трансформация и индексиране.

Забележка: Диаграмата илюстрира последователността на данните от JSON формат до векторен индекс.

Разработеният алгоритмичен комплекс, обхващащ последователните фази на топологично сортиране, семантична линеаризация чрез модифицирано обхождане в дълбочина (DFS) и експлицитна токенизация на обратните връзки, дефинира цялостна и автономна методологическа рамка за извличане на каузално знание от комплексни циклични графи. Интегрирането на глобалната контекстуална рамка (семантична котва) и трислойното агрегиране на извлечения текстови наратив оптимизира енергията на скаларните произведения в латентното пространство, подсигурявайки максимално силни тегла на внимание (Attention Weights) при последващото векторно кодиране. Чрез концептуалното съчетаване на йерархичното HNSW графово индексиране, стратегията на единния векторен запис (SVR) и асиметричното инжектиране на домейн контекст, предложеният метод успешно преодолява концептуалната пропаст между краткия, двусмислен потребителски вход и дългата математическа структура на моделите. По този начин дефинираната изчислителна рамка решава фундаменталния проблем с лексикалното несъответствие (*Vocabulary Mismatch*), преминавайки от повърхностно синтактично филтриране към дълбоко структурно и функционално разпознаване на патерни. Тази завършена теоретична методология за трансформация, симулация и интелигентно търсене служи като директен логически мост към следващия етап на изследването, насочен към софтуерното проектиране, многослойната децентрализация и реалната микросървисна реализация на платформата в облачна среда.

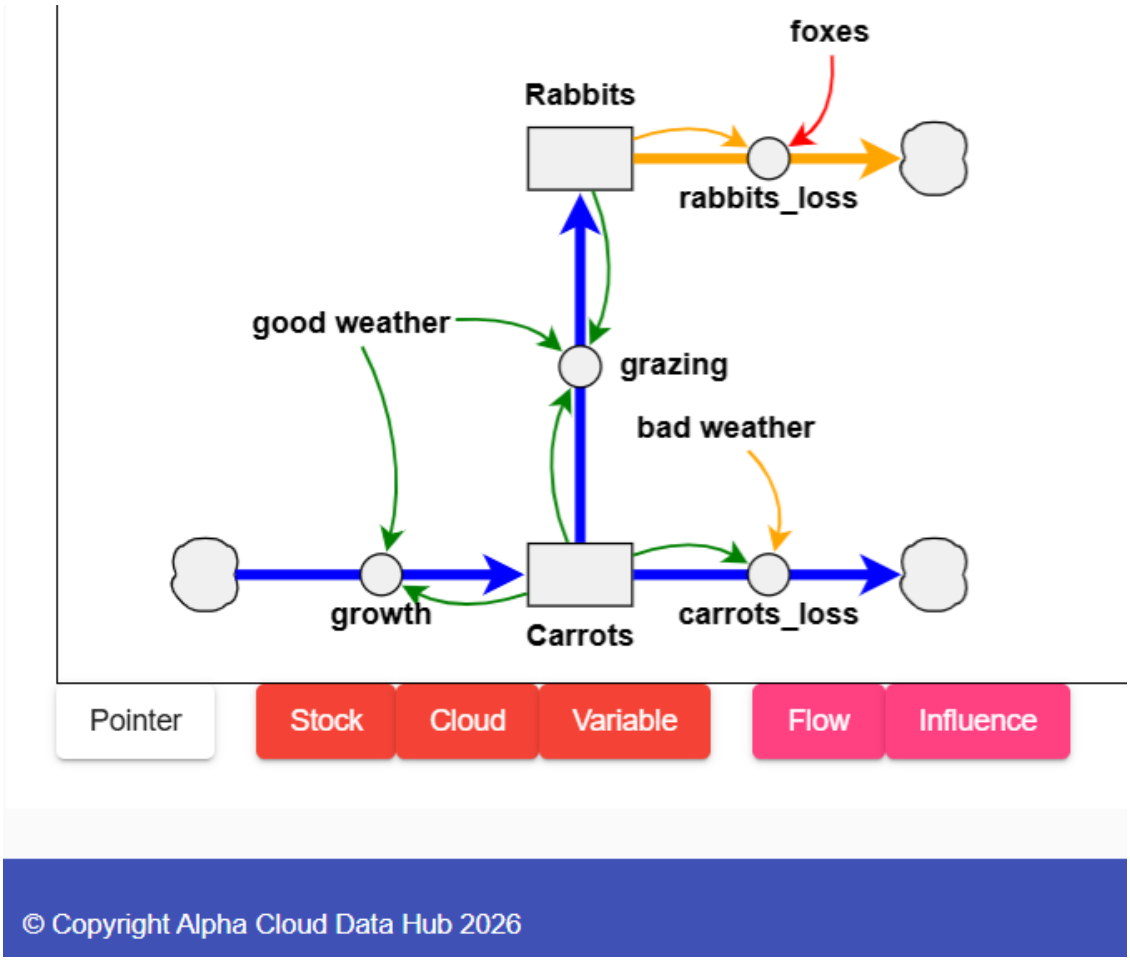
Изводи по Глава 3

- **Семантична линеаризация на графи:** Разработената оригинална методология и математическа функция за трансформация успешно превеждат нелинейната циклична топология на моделите по Системна динамика в структуриран едномерен текстов наратив, годен за обработка от съвременни латентни модели.
- **Ефективност на DFS обхождането:** Сравнителният анализ доказва, че за разлика от BFS, алгоритъмът за обхождане в дълбочина (DFS) запазва причинно-следствения контекст при константно разстояние между концепциите, което оптимизира скаларните произведения в механизма за само-внимание.
- **Експлицитна токенизация на циклите:** Въвеждането на специализирания семантичен маркер `[CYCLE_DETECTED]` в комбинация с твърд ограничител на дълбочината на рекурсията предотвратява загубата на структурно знание за обратните връзки и елиминира риска от комбинаторна експлозия на токени.
- **Многослойно агрегиране и контекст:** Синтезирането на генерирания семантичен контекст в три строго разграничени слоя (контекстуална рамка, структурен наратив и категориен индекс) осигурява правилно ориентиране на Трансформър модела и минимизира общата семантична ентропия.

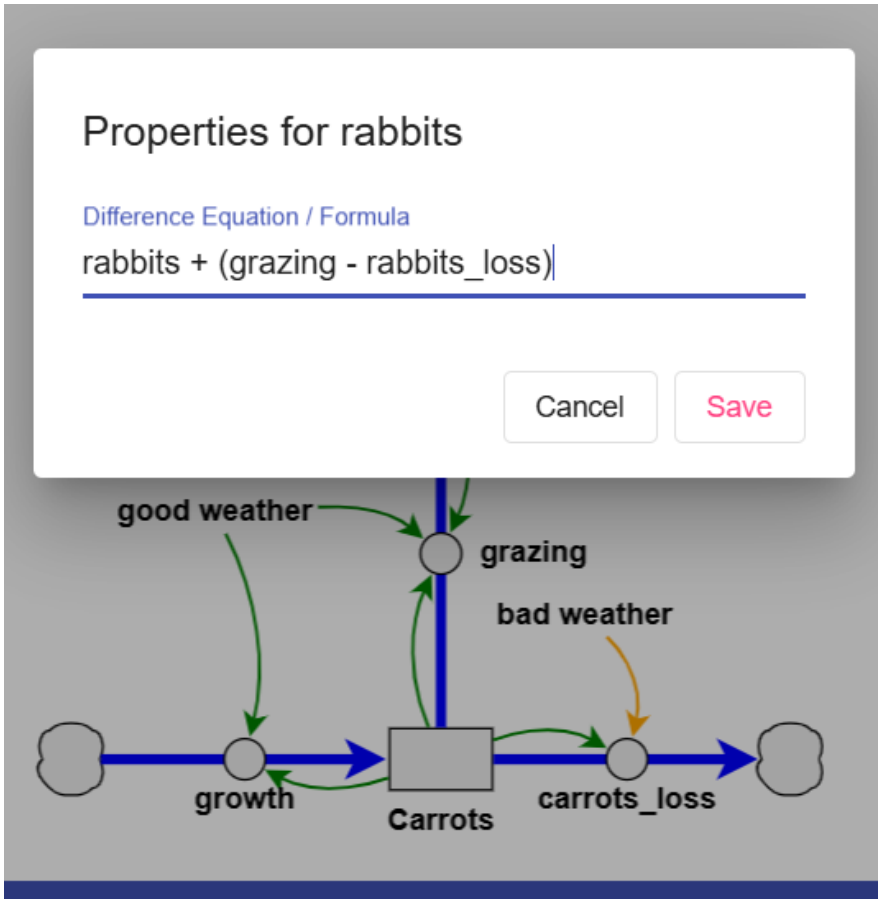
Глава 4. Архитектура на платформата Alpha Cloud Data Hub

Софтуерната реализация на *Alpha Cloud Data Hub* е базирана на принципите на Domain-Driven Design (DDD) [21] и микросървисна архитектура [22,23]. За да се предотврати блокирането на потребителския интерфейс при изчислително тежки AI операции, системата прилага събитийно-ориентиран модел при проектирането на интензивни потоци от данни [24] чрез брокера за съобщения *RabbitMQ* [25].

Потребителският интерфейс е реализиран като Single Page Application (SPA) [26], позволяващ интерактивно изграждане на диаграми и синхронна визуализация на симулационните резултати. Интерфейсът предоставя палитра с базовите елементи на Системната динамика за изграждане на визуална топология (Фигура 2), която се преобразува в реално време в JSON структура. Преходът към изпълним количествен модел се осъществява чрез дефиниране на математическите параметри на елементите (Фигура 3).



Фигура 2: Работно платно на визуалния редактор в Alpha Cloud Data Hub, демонстриращо интерактивно изграждане на екологичен Stock & Flow модел чрез компоненти за директна манипулация.

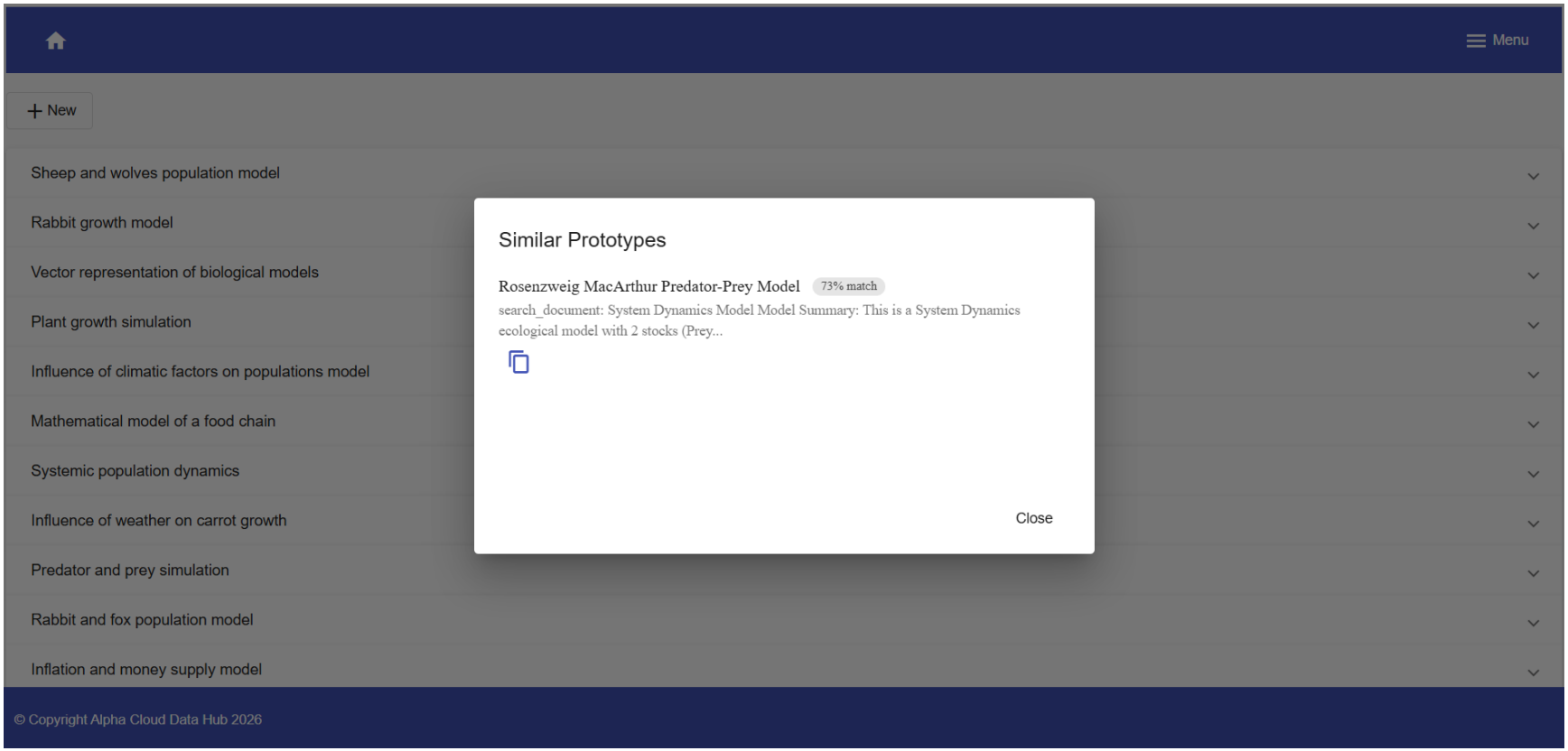


Фигура 3: Потребителски интерфейс за дефиниране на свойства и математически формули на избран възел (демонстрация на разностно уравнение за запас).

След визуалното конструиране на модела и дефинирането на неговите математически свойства (илюстрирано на Фигура 3), платформата преобразува въведените разностни уравнения в декларативни JSON структури, готови за симулация и семантично анализиране. За да се гарантира висока отзивчивост, клиентското Angular приложение извършва асинхронните мрежови заявки чрез своя вграден модул *HttpClient* [27], осигурявайки незабавна реактивност на редактора.

Докато същинското изчисляване на времевите редове се поема асинхронно от модула *System Dynamic Solver*, интелигентното извличане на знания се оркестрира от синхронната фаза на семантичния двигател. Когато потребителят въведе концептуална заявка на естествен език, системата я векторизира в реално време чрез локалния ONNX модел и я съпоставя с индексите във векторната база данни Qdrant.

Реализацията на този процес и крайният резултат в потребителския интерфейс са нагледно илюстрирани на Фигура 4. На примера е показано изпълнението на **Сценарий B** (Екологичен домейн), при който семантичният двигател успешно идентифицира релевантния модел на хищник-жертва с изчислено косинусово сходство от 73% и извлича неговия линеаризиран текстов наратив директно от паметта на Qdrant.



Фигура 4: Потребителски интерфейс на Alpha Cloud Data Hub, демонстриращ резултат от семантично търсене (визуализация на Сценарий B).

Забележка: Клиентското SPA приложение предоставя интуитивен интерфейс за достъп до семантичния двигател. Конкретният екран визуализира изпълнението на ***Сценарий B*** *(Екологичен домейн, детайлно разгледан в Глава 5), при който системата успешно е идентифицирала модела на хищник-жертва (Rosenzweig MacArthur Predator-Prey Model). Ясно се вижда процентното изражение на косинусовото сходство (73% match), което надвишава системно заложения праг от 0.72, както и част от линеаризирания текстов контекст (`search_document`), извлечен директно от векторната база данни Qdrant.*

Архитектурата използва модела на „полиглотно съхранение на данни“ (Polyglot Persistence) [28], комбинирайки документно-ориентирана база данни MongoDB като първичен източник на истината за JSON графите, и специализирана векторна база Qdrant [29] за семантичните индекси.

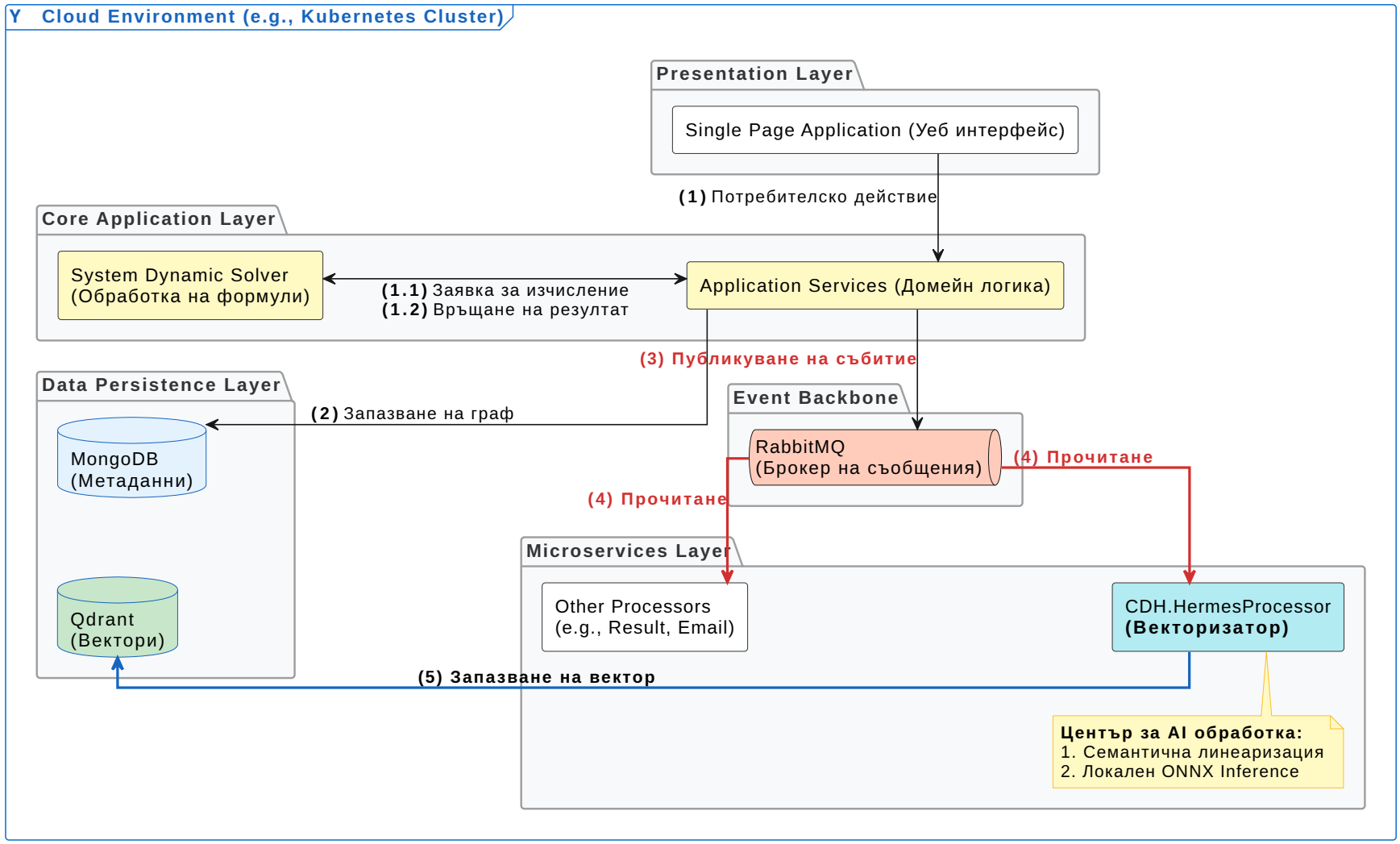
Процесът е строго разделен на две логически фази:

– **Асинхронна фаза (Индексиране):** При създаване или обновяване на прототип, уеб сървърът запазва метаданните в MongoDB и публикува събитие в RabbitMQ. Изолираният микросървис *Hermes Processor* консумира събитието, извършва семантичната линеаризация и извиква локалния *OnnxEmbeddingService* (модел `nomic-embed-text-v1.5`) за генериране на 768-мерен вектор. Векторът се записва в Qdrant чрез високопроизводителен gRPC протокол.

– **Синхронна фаза (Търсене):** Задейства се директно от потребителя. Заявката се преобразува във вектор в реално време, съпоставя се с индексите в Qdrant (търсейки най-близък съсед чрез HNSW), след което получените идентификатори се използват за извличане на пълните JSON документи от MongoDB. Използването на MongoDB GUID като Point ID в Qdrant осигурява естествена Upsert семантика и предотвратява десинхронизация.

Осигуряване на отказоустойчивост и изчислителна изолация

Ключово предимство на така проектираната децентрализирана архитектура е нейната вградена отказоустойчивост (Fault Tolerance). Използването на RabbitMQ не само решава проблема с блокирането на потребителския интерфейс, но и осигурява надеждност на данните чрез механизми за потвърждаване на съобщенията (Message Acknowledgments) и устойчиви опашки (Durable Queues). В случай на временен срив в изчислително тежкия микросървис *Hermes Processor* (отговарящ за AI векторизацията), генерираните събития не се губят, а остават в опашката до успешното им обработване. Успоредно с това, симулационният двигател (*System Dynamic Solver*) е изнесен като напълно изолиран компонент в приложния слой. Той парсва математическите изрази в абстрактни синтактични дървета (AST) и изпълнява числената интеграция напълно независимо от модулите за семантично търсене. Тази стриктна изолация гарантира, че евентуално претоварване в процесите по инференция на Трансформър модела няма да компрометира основната функционалност по изграждане и симулиране на математическите модели. **Взаимодействието между тези компоненти и логическото разделение на потоците са нагледно илюстрирани на Фигура 5.**



Фигура 5: Високо ниво на системната архитектура в CDH: Синхронни потоци за търсене и асинхронни потоци за семантично индексиране на данни.

Забележка: Диаграмата илюстрира стриктното разделение между синхронната обработка на заявки и тежката асинхронна векторизация чрез RabbitMQ. Показана е и напълно изолираната изчислителна логика.

Изводи по Глава 4

- **Разпределена микросървисна архитектура:** Проектирането на платформата въз основа на DDD принципите и събитийно-ориентирания модел с RabbitMQ осигурява пълно изчислително декуплиране, елиминирайки риска от блокиране на потребителския интерфейс при тежки AI операции.
- **Полиглотно съхранение:** Комбинирането на документно-ориентираната база данни MongoDB (като първичен източник на истината за JSON графите) със специализираната векторна база Qdrant постига оптимален баланс между изчислителна скорост и избягване на прекомерно натоварване на паметта.
- **Локален инференс и сигурност:** Стратегическото интегриране на локална ONNX Runtime сесия за генериране на ембединги елиминира мрежовото забавяне, предпазва системата от скрит технически дълг и гарантира пълна защита на индустриалната интелектуална собственост.
- **Консистентност и реактивност:** Използването на MongoDB GUID като директен Point ID във векторния индекс осигурява естествена Upsert семантика при редакция, а браузърно-базираното SPA приложение на Angular гарантира висока UX отзивчивост в реално време.

Глава 5. Експериментална валидация и анализ на резултатите

Експерименталната валидация е проведена в изолирана контейнеризирана среда (*Docker*) за гарантиране на повторяемост на резултатите. За целта е конструиран специализиран тестов корпус („Златен стандарт“) от 63 системнодинамични модела с висока степен на структурна и семантична хетерогенност. В Таблица 1 е представено количественото и тематично разпределение на моделите в корпуса.

Таблица 1: Тематично разпределение на моделите в тестовия корпус

Предметна област (Домейн)	**Брой модели**	**Дял (%)**	**Ср. запаси**
Екология и земеделие	12	19.0%	1.8
Демография и социални системи	11	17.5%	1.5
Икономика и финанси	10	15.9%	1.2
Биология и медицина	9	14.3%	2.0
Екология и околна среда	7	11.1%	1.8
Инфраструктура и производство	6	9.5%	2.3
Енергетика и ресурси	4	6.3%	1.0
Абстрактни и математически	4	6.4%	1.5
Общо	**63**	**100%**	**1.6**

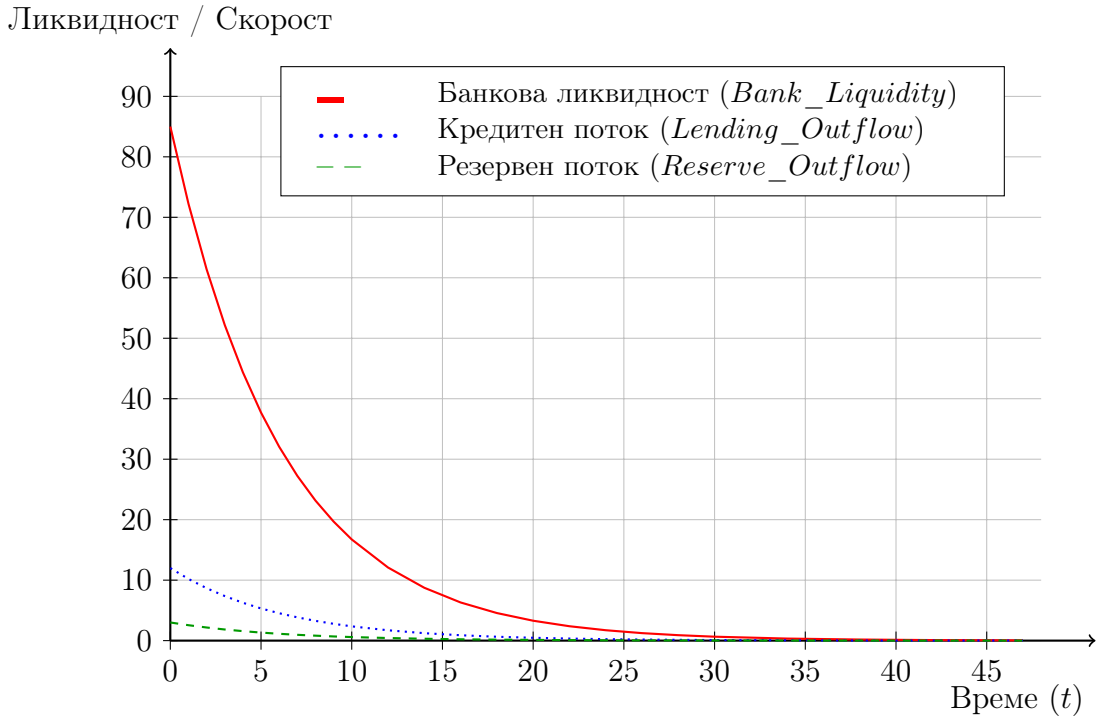
Количествен и структурен анализ на тестовия корпус (Таблица 1): Тематичното разпределение в корпуса демонстрира балансирана хетерогенност с доминиращ дял на екологичните (19.0%) и демографските модели (17.5%). Критичен показател

за алгоритмичната верификация е топологичната сложност на моделите (детайлно описана в Приложение В чрез параметрите „Среден брой възли“ и „Среден брой връзки“), както и колоната „Среден брой запаси (Stocks)“, тъй като те дефинират точките на прекъсване на алгебричните цикли при топологичното сортиране. Вариацията им от 1.0 до 2.3 осигурява широка база от нелинейни уравнения с различна плътност на обратните връзки, което гарантира обективното тестване на симулационния модул (*System Dynamic Solver*) и процесите по семантична линеаризация.

Валидация на симулационния модул

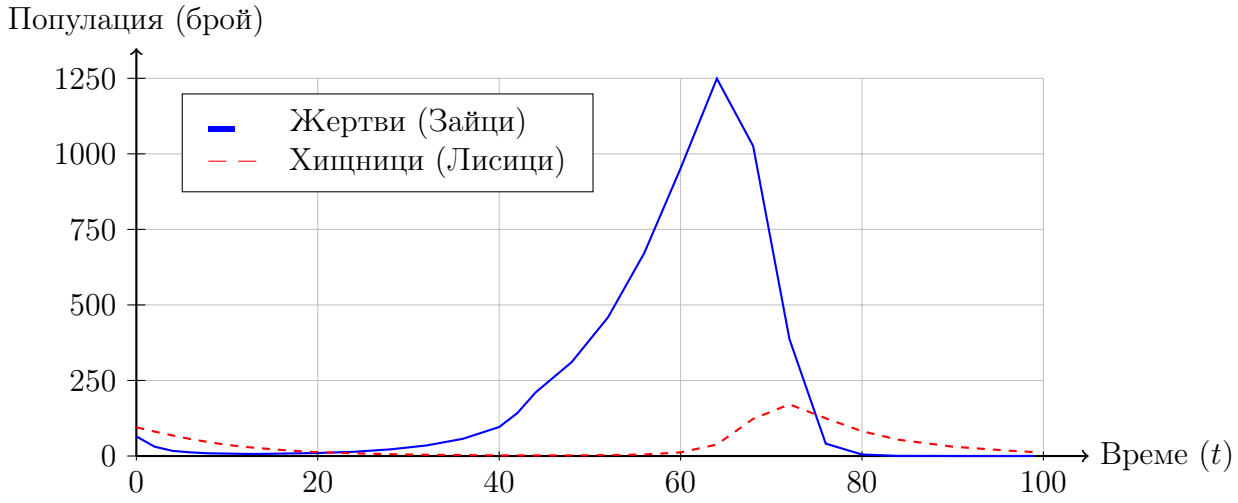
За верификация на изчислителната точност и междудомейнната инвариантност на модула *System Dynamic Solver*, е проведено тестване върху специализиран „Златен стандарт“ от еталонни модели. Целта е да се докаже, че платформата решава коректно заложените системи от обикновени диференциални уравнения (ОДУ) чрез Явния метод на Ойлер. Валидацията обхваща различни научни домейни и потвърждава стабилността на интегратора при конкуриращи се обратни връзки, генерирайки успешно очакваното нелинейно и осцилиращо поведение:

– **Финансов модел за управление на банкова ликвидност:** Системата успешно симулира динамиката на централен запас (Банкова ликвидност), който се захранва от синусоидален входящ поток и се изчерпва от два балансиращи изходящи потока (Кредитиране и Заделяне на резерви), доказвайки стабилността на числения интегратор при икономически сценарии.



Фигура 6: Симулационни резултати от еталонен финансов модел за управление на банкова ликвидност.

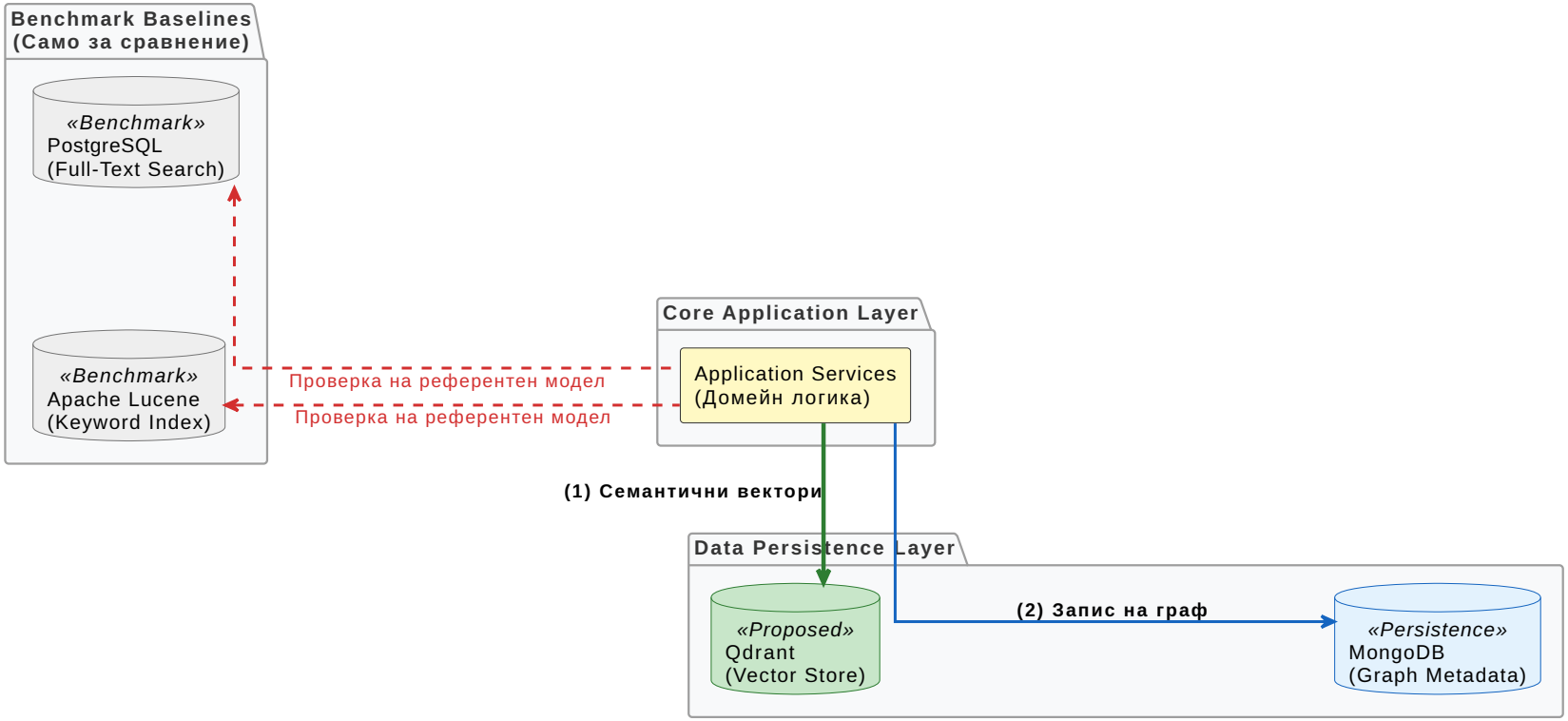
– **Екологичен модел Хищник-Жертва (Lotka-Volterra) [30]:** Симулацията успешно генерира характерните за модела свързани осцилации с фазово изместване. Това потвърждава способността на изчислителния модул да обработва прецизно конкуриращи се контури на обратна връзка при нелинейни системи за популационна динамика [31].



Фигура 7: Симулационни резултати от екологичен модел Хищник-Жертва (Сценарий B). Графиката ясно илюстрира свързаните осцилации с фазово закъснение, като пикът в популацията на жертвите води до последващ експоненциален ръст при хищниците.

Оценка на семантичното търсене

За да се реализира обективно съпоставяне на предложената семантична технология спрямо традиционните лексикални алгоритми (PostgreSQL Keyword/Full-Text Search, Apache Lucene BM25 [32]), приложната архитектура бе временно разширена с контролни механизми (*Baseline Checks*). На Фигура 8 е представена архитектурната топология на експеримента. Тя илюстрира как една и съща потребителска заявка се маршрутизира паралелно: от една страна към предложената полиглотна архитектура (векторни заявки към *Qdrant* и метаданни от *MongoDB*), а от друга – към еталонните релационни и пълнотекстови системи (*PostgreSQL* и *Apache Lucene*), които служат единствено за бенчмарк анализ и изолирана контролна среда.



Фигура 8: Архитектура на експерименталната постановка: Предложен семантичен модул спрямо базови лексикални системи за сравнение.

Забележка: Диаграмата илюстрира паралелната интеграция на PostgreSQL и Apache Lucene. В контекста на системата те функционират изцяло като изолирани референтен модел (Baselines) за оценка на ефективността и нямат отношение към същинския продукционен поток на данни в Alpha Cloud Data Hub.

За да се симулират реалистични потребителски търсения, са дефинирани пет контролни сценария, насочени към основните архетипи на системната динамика:

- **Сценарий A (Икономика):** Заявка *„Dynamics of cash flows and resources in economic systems“*.
- **Сценарий B (Екология):** Заявка *„Rosenzweig-MacArthur model stability“*.
- **Сценарий C (Епидемиология):** Заявка *„Climate influence on viral transmission simulation“*.
- **Сценарий D (Генетика):** Заявка *„Weighted gene regulatory network modeling“*.
- **Сценарий E (Демография):** Заявка *„Population dynamics with seasonal migration patterns“*.

Количествената оценка на ефективността е извършена чрез стандартните метрики за информационно извличане (Точност, Пълнота, F1-Score и Времезакъснение), чиято методологична обосновка е детайлно разгледана в първа глава на дисертационния труд. За да се елиминират субективните пристрастия, релевантността на извлечените модели спрямо концептуалните заявки бе предварително установена чрез двойно-сляп експертен преглед (double-blind expert review).

В настоящия автореферат детайлно е разгледан Сценарий А (Икономически домейн), тъй като той най-ярко илюстрира преодоляването на проблема с „лексикалното несъответствие“, докато резултатите от останалите сценарии са обобщени количествено за доказване на междудомейнната консистенция.

Качествен анализ на информационния шум (Сценарий А)

При тестване на концептуалната икономическа заявка *„Dynamics of cash flows and resources in economic systems“*, традиционните алгоритми (Keyword Search) демонстрират пълна контекстуална слепота. Те извличат модели като *„Systemic population dynamics“* (от демографията) и *„Extended Model of Predator-Prey Dynamics“* (от екологията), единствено заради съвпадение на думата „dynamics“. Това генерира 60% информационен шум (False Positives). Семантичният модул на *Alpha Cloud Data Hub* напълно елиминира този проблем, тъй като векторното пространство асоциира структурната топология с концепцията за динамика, без да изисква лексикално съвпадение.

Таблица 2: Сравнителен анализ за Икономически домейн (Сценарий А)

Метод	Резултати	Релевантни	Точност	Пълнота	F1-Score	Времезакъснение
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**

Векторното търсене анализира линеаризираната топология на графа и постига 100% точност и пълнота (F1-Score = 1.00) при праг на косинусова прилика $t = 0.72$, елиминирайки напълно информационния шум.

Калибриране на прага на семантично сходство (Cosine Similarity Threshold)

За да се гарантира максимална прецизност при извличането на знания и да се обоснове изборът на прага от 0.72, е проведен допълнителен експеримент (Sensitivity Analysis). Тъй като векторното пространство в модела `nomic-embed-text-v1.5` е изключително плътно, малки вариации в прага на отхвърляне водят до драстични промени в качеството на търсене. Приложени са различни стъпки на филтрация върху контролния корпус, като резултатите са документирани в Таблица 3.

Таблица 3: Оптимизация на F1-Score спрямо прага на косинусова прилика (t)

Праг на сходство (t)	**Точност (Precision)**	**Пълнота (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
$t \geq 0.72$ **(Оптимален)**	**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

Както е видно от данните, при ниски стойности на прага ($t < 0.70$), системата допуска множество фалшиво-положителни резултати (False Positives), тъй като алгоритъмът започва да асоциира твърде далечни абстракции. От друга страна, прекалено рестриктивните прагове ($t > 0.75$) предизвикват рязък спад в пълнотата (Recall), защото системата започва да отхвърля валидни структурни аналогии, които имат съвсем леки вариации в топологията си. Локалният максимум на хармоничното средно се достига точно при $t = 0.72$, поради което тази стойност е дефинирана като оптимална системна граница (boundary) в ядрото на *Alpha Cloud Data Hub*.

Анализ на производителността и мащабируемостта

Проведен е мащабен стрес-тест за профилиране на изчислителната сложност при различни по топология графи (Таблица 4). За целите на стрес-теста са подбрани три модела с коренно различен структурен профил:

- **Модел А (Финанси):** Линеен модел на парични потоци (4 компонента, 5 връзки), служещ като базова линия (Baseline).
- **Модел Б (Биология):** Комплексен модел за клетъчна популационна динамика (10 компонента, 13 връзки) с изолирани цикли на обратна връзка.
- **Модел В (Екология):** Модел за растителен растеж (6 компонента, 8 връзки) със силно преплетена топология и едновременни, припокриващи се алгебрични цикли.

Таблица 4: Сравнителен анализ на забавянето при добавяне, обработка и векторизация на модели с различна топологична сложност

Метрика / Параметър	**Модел А (Финанси)**	**Модел Б (Биология)**	**Модел В (Екология)**
Структурен профил на модела			
Общ брой компоненти	4	10	6
Брой топологични връзки	5	13	8
Генериран контекст (Токени)	554	1013	721
Сървърна обработка (Синхронна фаза)			
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
Общо време (Backend)	**2999.51 ms**	**2998.86 ms**	**12152.72 ms**
AI Микросървис Hermes (Асинхронна фаза)			
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
Общо време (Hermes)	**5719.24 ms**	**5479.75 ms**	**5562.75 ms**
Пълно системно време	**≈ 8.72 секунди**	**≈ 8.48 секунди**	**≈ 17.72 секунди**

Резултатите доказват, че плътността на припокриващите се алгебрични цикли (а не броят на възлите) предизвиква експоненциален скок във времето за топологично сортиране (над 12 секунди за Модел B). Допълнителната AI векторизация добавя константно ∼5.5 секунди. Това емпирично валидира архитектурното решение за изнасяне на семантичния анализ в асинхронна RabbitMQ опашка. Въпреки тежката фаза на индексиране при запис (до 18 секунди), времезакъснението (Latency) при **семантично търсене** (четене) се задържа стабилно около ∼1.89 секунди, гарантирайки отлична отзивчивост.

Изводи по Глава 5

- **Коректност на симулационното ядро:** Експерименталната верификация доказва, че симулационният модул изчислява прецизно детерминирани математически модели от различни домейни, възпроизвеждайки напълно коректно тяхното специфично нелинейно и осцилиращо поведенческо описание.

- **Превъзходство на семантичното търсене:** Семантичният подход категорично превъзхожда традиционните лексикални методи, като елиминира изцяло информационния шум при абстрактни концептуални заявки и постига максимална точност и пълнота при оптимален праг на сходство $t = 0.72$.

- **Валидация на асинхронната архитектура:** Профилирането на изчислителната сложност потвърждава, че плътността на алгебричните цикли натоварва бекенда, което практически обосновава архитектурното решение за изнасяне на тежката Трансформър векторизация в асинхронна опашка за запазване на ниско времезакъснение при търсене.

Заключение

В заключението е направено обобщение на дисертационния труд. Оценена е степента на изпълнение на поставените задачи и са формулирани основните научно-приложни и приложни приноси, сред които създаването на алгоритъма за семантична линеаризация и реализацията на облачния прототип *Alpha Cloud Data Hub*.

Очертани са техническите предизвикателства и насоките за бъдещо развитие:

- **Инфраструктурна оптимизация:** Внедряване на хардуерно GPU ускорение на микросървисите за редуциране на времето за векторизация до под 200 ms, многослойно кеширане (Redis) и еластично хоризонтално мащабиране (Kubernetes [33]).

- **Алгоритмични надграждания:** Преход към архитектури за мултивекторно индексиране (ColBERT-тип) чрез декомпозиция на графа на логически сегменти, за да се преодолее „информационното тясно място“ при агрегиране на масивни индустриални модели в единствен вектор.

Научни и научно-приложни приноси

На базата на проведените теоретични и практически изследвания, отразени в дисертационния труд, са дефинирани следните четири основни приноса:

Научно-приложни приноси

1. Създадена е новаторска методология и алгоритъм за семантична линеаризация на циклични графови структури, специфицирани чрез формализма на Системната динамика. Въвеждането на маркери за цикличност (explicit loop tokenization) позволява съхраняването на каузалната топология под формата на плътен векторен контекст, годен за анализи чрез Обработка на естествен език (NLP).
2. Разработен е хибриден, полиглот-базиран архитектурен модел, който интегрира Event-Driven микросървисна среда (RabbitMQ) със специализирано векторно търсене (Qdrant). Този модел решава проблема с изчислителното блокиране при тежка AI обработка (Inference), като изолира синхронния процес на четене от асинхронния конвейер за създаване на знания.

Приложни приноси

3. Проектиран, реализиран и внедрен е софтуерен прототип *Alpha Cloud Data Hub*. Прототипът валидира ефективността на предложените теоретични методи на практика, осигурявайки уеб-базирана екосистема за визуално моделиране, симулация и интелигентно откриване на модели чрез асиметрично семантично търсене.

Обосноваващи (Емпирични) научно-приложни приноси

4. Емпирично са обосновани предимствата на семантичния векторен подход спрямо традиционните лексикални алгоритми при извличане на знания от динамични модели. Чрез проведен експериментален анализ е потвърдено, че предложената архитектура успешно адресира проблема с „лексикалното несъответствие“, като постига 100% пълнота (Recall) при концептуални заявки и значително редуцира информационния шум (False Positives), присъщ на пълнотекстовото търсене.

Връзките между приносите, целите, задачите, мястото на описание в дисертационния труд и направените публикации са обобщени в Таблица 5.

Таблица 5: Връзка между приноси, задачи, раздели в дисертацията и публикации

Принос	Вид	Задачи	Глава	Публикации
1	Научно-приложен	Зад. 2	Глава 3	Публ. 3
2	Научно-приложен	Зад. 1 & 3	Глава 4	Публ. 1, 2 & 3
3	Приложен	Зад. 3 & 4	Глави 4 & 5	Публ. 1, 2 & 3
4	Научно-приложен	Зад. 4	Глава 5	Публ. 3

Публикации и апробация на резултатите

Публикации по темата на дисертацията

Основните теоретични постановки, архитектурни решения и експериментални резултати от дисертационния труд са отразени в 3 научни публикации, които покриват всички формулирани научно-приложни и приложни приноси:

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)**

Участия в научни форуми (Апробация)

Резултатите от изследванията са представени и дискутирани пред научната общност под формата на два доклада на следните научени форуми:

– Научна конференция „Иновационни ИКТ: Изследвания, разработка и приложения в бизнеса и обучението“ (2015 г., гр. Хисар). Изнесен доклад на тема *„Архитектура на уеб базирана система за символно програмиране на реални процеси“*.

– Научна сесия на Съюза на учените в България – клон Сливен (2017 г., гр. Сливен). Изнесен доклад, съпътстващ публикацията *„Cloud Computing of Incoming Data - Analyze and Storage of Dynamic Mathematical Models“*.

Финансиране и връзка с научни проекти

Настоящият дисертационен труд обобщава резултати, постигнати в рамките на научноизследователска дейност, подпомогната от Фонд „Научни изследвания“ към Пловдивски университет „Паисий Хилендарски“. Част от изследванията и реализациите на прототипите са финансирани по проект:

– Проект **ИТ15-ФМИИТ-004 „Изследвания в областта на иновационни ИКТ с ориентация към бизнеса и обучението“** към НПД на Пловдивски университет „Паисий Хилендарски“.

– Проект **МУ17-ФМИ-007 „ИКТ в помощ на научните изследвания по математика и информатика и приложенията им“** към НПД на Пловдивски университет „Паисий Хилендарски“.

БЛАГОДАРНОСТИ

На първо място изразявам своята искрена признателност към научния си ръководител **проф. д-р Антон Илиев** за оказаното доверие, предоставените възможности, търпението и безрезервната подкрепа в процеса на цялостната ми работа по дисертационния труд.

Специална признателност изказвам за професионалната подкрепа, ценните научни напътствия и методически съвети на **проф. д-р Асен Рахнев** и **проф. д-р Николай Павлов**.

Благодаря на всички колеги от катедра „Компютърни технологии“ към Факултета по математика и информатика на ПУ „Паисий Хилендарски“ за създадената отлична академична среда, ползотворните дискусии и колегиалността, които по различни начини допринесоха за реализирането на настоящия труд.

Накрая, но не на последно място, изразявам дълбоката си благодарност към моето семейство за неизменната подкрепа и разбиране във всички мои начинания.

За мен беше истинско щастие да извървя този път, воден от обичта към науката, превръщайки настоящата дисертация в най-красивия и ценен резултат от този вдъхновяващ етап от моя живот.

Литература

[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 (посетен на 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 (посетен на 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 (посетен на 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.