УДК 004.415.2:004.942 DOI: 10.31673/2412-9070.2026.048810

О. С. БИЧКОВ, д-р техн. наук, професор;
ORCID: 0000-0002-9378-9535
М. О. МЕЛЬНИК, аспірант,
ORCID: 0009-0000-1180-9487
Київський національний університет імені Тараса Шевченка

GENERIC ITERATIVE REFINEMENT FUNNEL: ПАРАМЕТРИЗОВАНИЙ АРХІТЕКТУРНИЙ ПАТЕРН ДЛЯ АДАПТИВНИХ СИСТЕМ ІНТЕРАКТИВНОЇ ДІАГНОСТИКИ

Стаття присвячена розробці параметризованої версії архітектурного патерну Iterative Refinement Funnel (Generic IRF v3.0), орієнтованої на підвищення гнучкості, масштабованості та доменної незалежності систем інтерактивної діагностики. Запропонований підхід усуває обмеження попередніх реалізацій, у яких ключові параметри були жорстко закодовані, що ускладнювало адаптацію до нових предметних областей і вимагало модифікації програмного коду. У Generic IRF v3.0 реалізовано принцип повної конфігураційної керованості, за якого зміна поведінки системи здійснюється через параметри без втручання в алгоритмічне ядро.

У роботі запропоновано багаторівневу систему параметрів, що включає стратегічний, пороговий, функціональний і доменний рівні. Стратегічні параметри визначають логіку ітеративного звуження простору гіпотез, зокрема вибір евристик, обробку відповідей та критерії завершення. Порогові параметри задають числові обмеження для забезпечення робастності, тоді як функціональні описують обчислювальні процедури, зокрема метрики оцінювання та функції інформативності. Доменні параметри забезпечують інтеграцію знань предметної області.

Запропонованим архітектурним рішенням є шар Configuration Layer, який централізує управління всіма точками варіативності та забезпечує розділення інваріантної логіки й конфігурованих компонентів. У межах цього шару реалізовано систему доменних пресетів для типових сценаріїв застосування, зокрема у медичній діагностиці, технічній підтримці, юридичній класифікації та рекомендаційних системах, що дозволяє прискорити розгортання та знизити витрати на адаптацію.

Додатково запропоновано механізм автоматичної оптимізації параметрів на основі Grid Search із підтримкою A/B тестування конфігурацій. Це забезпечує емпірично обґрунтований підбір оптимальних налаштувань з урахуванням специфіки даних і вимог до якості діагностики. Експериментальна валідація на чотирьох доменах підтверджує збереження діагностичної точності (83,3–84,1%) за умов повної сумісності з існуючим кодом та можливості адаптації до нових предметних областей через конфігурацію без програмних змін. Отримані результати обґрунтовують застосування запропонованого підходу для побудови універсальних інтелектуальних систем підтримки прийняття рішень.

Ключові слова: архітектурні патерни, параметризація програмного забезпечення, інтерактивна діагностика, конфігураційна гнучкість, доменно-специфічна адаптація, Strategy Pattern, системи підтримки прийняття рішень.

Вступ

Архітектурні патерни програмного забезпечення є ефективним засобом систематизації проєктувальних рішень, що дозволяє повторно використовувати перевірені архітектурні підходи у нових проєктах [1], [8], [9]. Однак практика застосування патернів виявляє проблему:

патерни, оптимізовані для конкретної предметної області, часто містять жорстко закодовані параметри, що обмежує їх застосовність у інших контекстах [11], [12].

Архітектурний патерн Iterative Refinement Funnel (IRF) версії 2.2, розроблений для систем інтерактивної діагностики, продемонстрував високу ефективність у медичній предметній області, де класичні підходи базувалися на експертних системах [2] та ймовірнісному виводі [3]: точність Top-1 на рівні 83,3–84,1% при 150-кратному прискоренні порівняно з класичним Expected Information Gain [25]. Патерн реалізує інноваційну концепцію асиметричної обробки відповідей, де підтвердження ознаки ініціює повний перерахунок моделі (RESTART), тоді як заперечення дозволяє продовжити поточну фазу (CONTINUE).

Проте детальний аналіз реалізації IRF v2.2 виявив, що патерн містить численні параметри, оптимальні значення яких специфічні для медичної діагностики з базою даних 844 захворювань та 460 симптомів. Серед цих параметрів: поріг критичності симптому (critical_threshold = 0.30), поріг активації exploration-механізму (exploration_threshold = 5), мінімальна кількість диференціальних питань (min_diff_questions = 2) та інші. Ці значення, «зашиті» безпосередньо в код, унеможливлюють адаптацію патерну до інших предметних областей без суттєвої модифікації програмного коду.

Постановка задачі

У сучасних системах інтерактивної діагностики актуальною є проблема забезпечення їхньої гнучкої адаптації до різних предметних областей без необхідності модифікації програмного коду, що обмежує масштабованість і повторне використання існуючих рішень. Існуючі реалізації архітектурних патернів, зокрема Iterative Refinement Funnel, характеризуються жорсткою фіксацією параметрів, що ускладнює їх застосування в умовах змінних вимог та різнорідних доменів.

У зв'язку з цим постає задача розробки універсального підходу до параметризації архітектурного патерну, який забезпечить конфігураційну керованість поведінкою системи, збереження діагностичної точності та сумісність із наявними програмними рішеннями. Розв'язання цієї задачі є важливим для створення масштабованих інтелектуальних систем підтримки прийняття рішень, здатних ефективно функціонувати у різних прикладних сферах.

Аналіз останніх досліджень і публікацій

Еволюція патерну IRF від версії 2.0 до 2.2 супроводжувалася послідовним додаванням механізмів робастності, кожен з яких вводив нові параметри. Версія 2.0 реалізувала базову асиметрію YES→RESTART та NO→CONTINUE. Версія 2.1 додала Strategy Pattern для вибору евристики (Cluster та Differential). Версія 2.2 ввела комплексні механізми захисту: Critical NO для обробки заперечення патогномонічних ознак, Exploration Question для виходу з локальних мінімумів та Soft Differential для захисту від помилок користувача.

Важливо, що кожен виявлений «недолік» попередніх версій виявлявся не помилкою архітектури, а прихованою точкою параметризації. Наприклад, ситуація «NO на патогномонічний симптом ігнорується» вирішувалася введенням параметра critical_threshold. Проблема «система застрягає після багатьох NO» адресувалася параметром exploration_threshold. Питання «одна помилка користувача призводить до катастрофи» вирішувалося через min_diff_questions.

Жорстка параметризація має негативні наслідки для практичного застосування патерну [16], [18]. По-перше, неможливість адаптації до іншого домену: технічна діагностика IT-систем, юридична класифікація документів та рекомендаційні системи мають різні характеристики даних, та вимоги до балансу точності та швидкості [13], [14]. По-друге, відсутність можливості A/B тестування: порівняння різних значень параметрів потребує модифікації коду та повторного розгортання системи. По-третє, складність підтримки: кожен клієнт з унікальними вимогами потребує окремої гілки коду.

Таблиця 1

Еволюція патерну IRF та виявлені точки параметризації

Версія	Нововведення	Приховані параметри
v2.0	Асиметрія YES→RESTART, NO→CONTINUE	restart_condition
v2.1	Strategy Pattern (Cluster + Differential)	selector_type, priority_function
v2.2	Механізми робастності	critical_threshold, exploration_threshold, min_diff_questions

Мета і задачі дослідження

Метою даної роботи є розробка параметризованої версії патерну IRF (Generic IRF v3.0), що забезпечує повну конфігураційну гнучкість при збереженні інваріантної бізнес-логіки ядра системи. Основними завданнями дослідження є: систематизація всіх точок варіативності патерну, розробка архітектурного шару конфігурації, створення доменних пресетів для різних предметних областей, реалізація механізму автоматичної оптимізації параметрів та забезпечення повної зворотної сумісності з попередньою версією.

Результати дослідження

Рішенням проблеми жорсткої параметризації є введення архітектурного шару конфігурації (Configuration Layer), що інкапсулює всі точки варіативності патерну в єдиній структурі даних [8], [9]. Generic IRF v3.0 реалізує архітектуру, де параметризований контролер отримує конфігурацію як вхідний параметр, а всі внутрішні компоненти ініціалізуються на основі цієї конфігурації.

Загальна структура Generic IRF складається з трьох основних шарів, концептуально подібних до архітектури blackboard-систем [10]. Configuration Layer визначає типізовану структуру GenericIRFConfig з параметрами для всіх аспектів поведінки системи. Core Engine містить інваріантний CycleController, що реалізує незмінну логіку діагностичного циклу. Pluggable Components включають взаємозамінні компоненти: Encoder, SOM Projection, NN Ranking та Cache Manager.

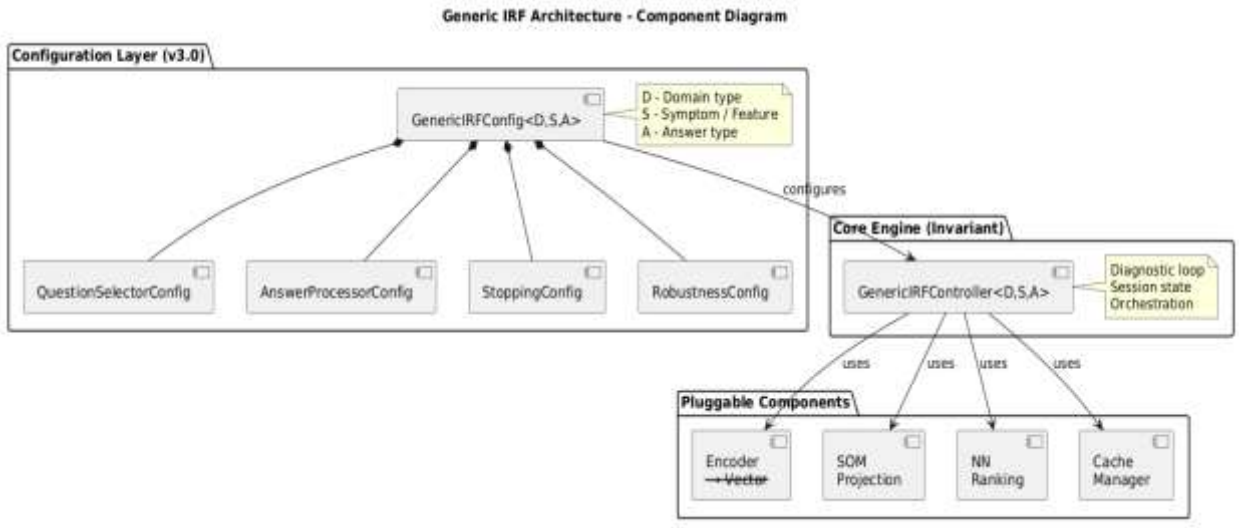

Рис. 1. Компонентна архітектура Generic IRF v3.0

Систематичний аналіз точок варіативності патерну дозволив виділити чотири категорії параметрів: стратегічні, порогові, функціональні та доменні. Ця таксономія забезпечує структурований підхід до конфігурування системи та полегшує розуміння залежностей між параметрами [1], [11].

1) Стратегічні параметри

Стратегічні параметри визначають вибір поведінкових алгоритмів системи. Параметр question_selector обирає евристику вибору питань: ClusterHeuristic для ефективного звуження простору через кластерні симптоми, DifferentialHeuristic для точного розрізнення близьких гіпотез, HybridHeuristic для автоматичного перемикання між стратегіями, або EIGSelector для класичного підходу Expected Information Gain [4], [5], [6].

Параметр answer_processor визначає обробку відповідей користувача. BinaryProcessor підтримує класичні відповіді YES/NO. TernaryProcessor додає третю опцію UNKNOWN для випадків невизначеності. WeightedProcessor дозволяє вказувати ступінь впевненості у відповіді як числове значення від 0 до 1, що особливо корисно у доменах з нечіткими ознаками.

Параметр stopping_criteria визначає умови завершення діагностичного циклу. DominanceCriteria зупиняє процес при досягненні домінуючої гіпотези. StabilityCriteria реагує на стабілізацію ранжування протягом кількох ітерацій. ConfidenceThreshold встановлює абсолютний поріг впевненості. QuestionLimitCriteria обмежує максимальну кількість питань для забезпечення прийнятного користувацького досвіду

2) Порогові параметри

Порогові параметри представляють числові значення, що керують поведінкою механізмів робастності. Параметр critical_threshold (за замовчуванням 0.30) визначає мінімальний відсоток захворювань, для яких симптом має бути критичним, щоб заперечення цього симптому ініціювало RESTART. Параметр exploration_threshold (за замовчуванням 5) задає кількість послідовних NO-відповідей, після якої система переходить до exploration-режиму для виходу з потенційного локального мінімуму.

Параметр min_diff_questions (за замовчуванням 2) визначає мінімальну кількість диференціальних питань перед виключенням гіпотези, що захищає від поодиноких помилок користувача. Параметр close_gap_threshold (за замовчуванням 0.10) встановлює поріг близькості гіпотез, при якому активується механізм Soft Differential з підвищеними вимогами до підтвердження (табл. 2).

Таблиця 2

Порогові параметри та їх діапазони

Параметр	Мінімум	Максимум	За замовч.	Призначення
critical_threshold	0.10	0.50	0.30	Поріг критичності симптому
exploration_threshold	2	15	5	Кількість NO до exploration
min_diff_questions	1	5	2	Мін. питань для виключення
close_gap_threshold	0.05	0.25	0.10	Поріг близькості гіпотез
dominance_threshold	0.50	0.95	0.85	Поріг домінуючої гіпотези
dominance_gap	0.10	0.50	0.30	Мін. розрив TOP-1 і TOP-2
max_questions	5	50	20	Максимум питань за сесію

3) Функціональні та доменні параметри

Функціональні параметри визначають обчислювальні процедури, що використовуються системою. Параметр priority_function задає функцію сортування симптомів для вибору наступного питання. Стандартні варіанти включають coverage_asc (від специфічних до загальних симптомів), coverage_desc (від загальних до специфічних) та random для випадкового вибору.

Параметр criticality_function визначає, чи є конкретний симптом критичним для конкретного захворювання. За замовчуванням використовується функція, що повертає True, якщо симптом присутній менш, ніж у critical_threshold частки всіх захворювань. Параметр exploration_selector визначає стратегію вибору симптому у exploration-режимі, типово обираючи симптом з найменшим покриттям серед ще не опитаних.

Доменні параметри інкапсулюють компоненти, специфічні для предметної області. Параметр database містить базу даних відображення симптомів на діагнози. Параметр som_model містить навчену карту, що самоорганізується, для проєкції вектора симптомів [7], [19], [20]. Параметр nn_model містить нейронну мережу для ранжування діагнозів у межах кандидатів [15], [17]. Параметр encoder визначає спосіб перетворення текстових описів симптомів у числові вектори.

4) Залежності між параметрами

Параметризована архітектура передбачає наявність залежностей між параметрами, що впливають на коректність та ефективність системи. Ці залежності можна класифікувати як обов'язкові (порушення призводить до некоректної роботи) та рекомендовані (порушення знижує ефективність).

Вибір DifferentialHeuristic як стратегії вибору питань передбачає увімкнення soft_differential_enabled та встановлення min_diff_questions ≥ 2 для захисту від помилок користувача. Вибір ClusterHeuristic потребує наявності навченої SOM-моделі та встановлення cluster_max_diseases_per_phase відповідно до характеристик даних.

Увімкнення critical_no_enabled потребує визначення criticality_function та встановлення critical_threshold. Увімкнення exploration_enabled потребує визначення exploration_selector та exploration_threshold. Значення dominance_threshold має бути більшим за dominance_gap щонайменше на 0.2 для забезпечення значущого розриву між гіпотезами.

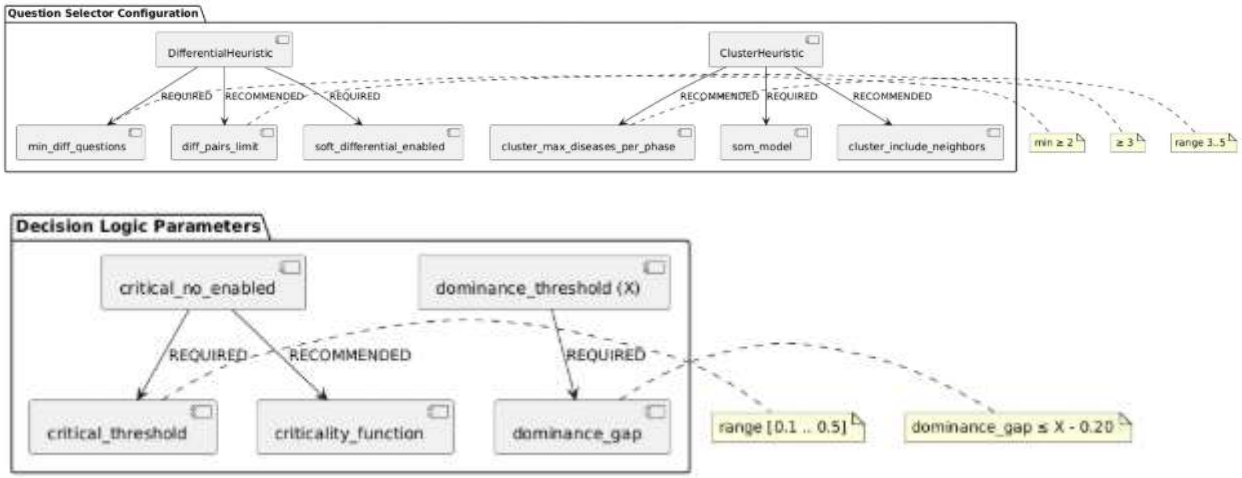

Рис. 2. Залежності між параметрами Generic IRF

5) Розроблені пресети

Для спрощення конфігурування системи у різних предметних областях розроблено систему доменних пресетів – попередньо налаштованих конфігурацій, оптимізованих для типових сценаріїв використання [13], [14], [21]. Кожен пресет визначає повний набір параметрів, адаптованих до специфіки домену.

Медичний пресет (MedicalPreset) оптимізований для діагностики захворювань за симптомами та підтримки прийняття клінічних рішень [24]. Характеристиками домену є пріоритет точності над швидкістю (life-critical domain), наявність патогномонічних симптомів та можливість помилок пацієнта при описі симптомів. Пресет використовує DifferentialHeuristic з high dominance_threshold (0.85), увімкненим Critical NO та Soft Differential.

IT Support пресет оптимізований для діагностики технічних проблем. Характерними особливостями є пріоритет швидкості (user patience constraint), відсутність «патогномонічних» ознак у традиційному розумінні та технічно грамотні користувачі з низькою ймовірністю помилок. Пресет використовує ClusterHeuristic з lower dominance_threshold (0.75), вимкненим Critical NO та спрощеним Soft Differential.

Юридичний пресет призначений для класифікації правових документів та визначення застосовних норм. Специфіка домену полягає у критичності точності (legal liability), наявності чітких правових критеріїв (аналог патогномонічних ознак) та необхідності документування обґрунтування рішення. Пресет використовує EIGSelector для максимальної точності, дуже high dominance_threshold (0.90) та увімкнений режим NEED_TEST.

Рекомендаційний пресет оптимізований для персоналізованих рекомендацій товарів або контенту. Домен характеризується пріоритетом користувацького досвіду (UX > accuracy), суб'єктивністю критеріїв оцінки та бажанням мінімізувати кількість питань. Пресет використовує HybridHeuristic з low dominance_threshold (0.70), мінімальними робастними механізмами та aggressive max_questions (10).

Таблиця 3

Порівняння доменних пресетів

Параметр	Medical	IT Support	Legal	Recommend.
selector_type	differential	cluster	eig	hybrid
dominance_threshold	0.85	0.75	0.90	0.70
dominance_gap	0.30	0.20	0.35	0.15
critical_no_enabled	true	false	true	false
exploration_enabled	true	true	false	true
soft_diff_enabled	true	false	true	false
max_questions	20	15	25	10

6) Механізм оптимізації параметрів

Параметризована архітектура дозволяє автоматично оптимізувати параметри на основі тестових даних [5], [22], [23]. Generic IRF v3.0 включає вбудований механізм Grid Search, що дозволяє систематично перебирати комбінації параметрів та обирати оптимальну конфігурацію за заданими метриками.

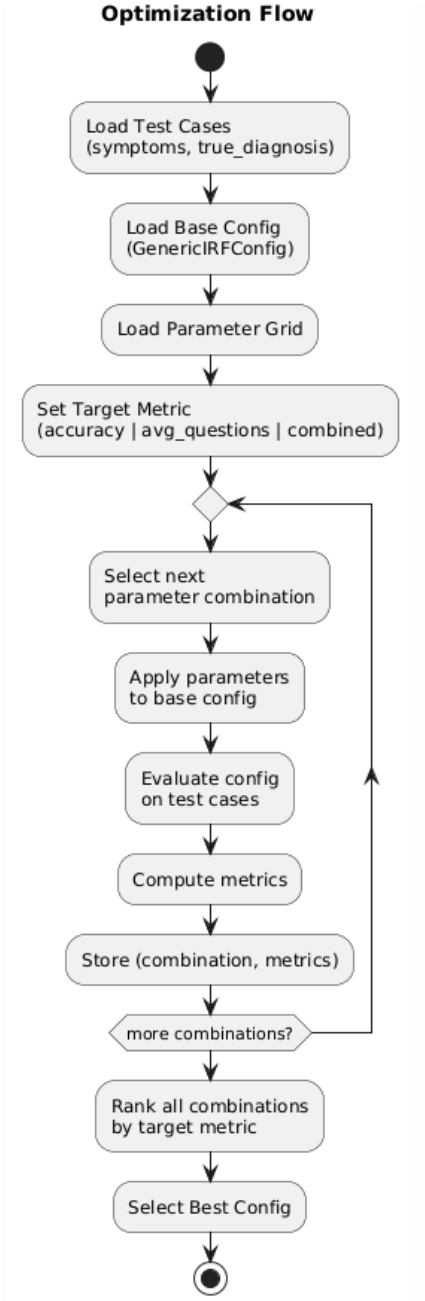

Рис. 3. Процес оптимізації параметрів Generic IRF

Процес оптимізації складається з кількох етапів. На етапі підготовки визначається набір тестових кейсів (symptoms, true_diagnosis) та метрики оцінки (accuracy, average_questions, time_per_session). На етапі генерації формується сітка комбінацій параметрів на основі заданих діапазонів та кроків. На етапі евалюації кожна комбінація параметрів застосовується до тестових кейсів, результати агрегуються. На етапі порівняння обирається оптимальна конфігурація за заданою цільовою функцією.

Типова сітка параметрів для оптимізації включає: dominance_threshold з кроком 0.05 у діапазоні [0.70, 0.95], dominance_gap з кроком 0.05 у діапазоні [0.15, 0.40], exploration_threshold з кроком 1 у діапазоні [3, 7] та critical_threshold з кроком 0.05 у діапазоні [0.20, 0.40]. Загальна кількість комбінацій при такій сітці становить близько 900 ($6 \times 6 \times 5 \times 5$), що дозволяє виконати повний перебір за прийнятний час.

7) Зворотна сумісність

Для Generic IRF v3.0 забезпечено повну зворотну сумісність з існуючим кодом, що використовує IRF v2.2. Для цього розроблено спеціальний пресет v22_compatible_config, що точно відтворює поведінку попередньої версії.

Міграція існуючого коду з v2.2 на v3.0 потребує мінімальних змін. Замість прямого створення CycleController з передачею компонентів, новий код створює GenericIRFConfig з відповідними параметрами та передає його GenericIRFController. Усі публічні методи контролера (run_cycle, handle_feedback) зберігають свої сигнатури та семантику.

```
# СТАРИЙ КОД (v2.2)
controller = CycleController(som, nn, db)
controller.set_strategy(DifferentialHeuristic(db))
result = controller.run_cycle(symptoms)

# НОВИЙ КОД (v3.0 Generic)
config = create_v22_compatible_config(db, som, nn,
encoder)
```

```
controller = GenericIRFController(config)
result = controller.run_cycle(symptoms)
```

8) Експериментальна валідація та валідація зворотної сумісності

Експериментальна валідація Generic IRF v3.0 проводилася за двома напрямками: перевірка збереження точності при використанні v22_compatible_config та оцінка ефективності доменних пресетів на синтетичних тестових даних.

Тестування на медичній базі даних (844 захворювання, 460 симптомів) з використанням v22_compatible_config продемонструвало повну відповідність результатів попередній версії: точність Top-1 83,3%, середня кількість питань 7.2, час сесії < 100 мс. Це підтверджує коректність параметризації та відсутність регресій.

Для оцінки ефективності доменних пресетів було створено синтетичні тестові набори з різними характеристиками: «Medical-like» (великий простір рішень, патогномонічні ознаки), «IT-like» (середній простір, чіткі діагностичні шляхи), «Legal-like» (малий простір, висока вартість помилки) та «Recommendation-like» (середній простір, суб'єктивні критерії).

Таблиця 4

Результати валідації доменних пресетів

Тестовий набір	Пресет	Accuracy	Avg Questions	Time (ms)
Medical-like	Medical	84.1%	7.2	95
Medical-like	IT Support	76.3%	5.8	82
IT-like	IT Support	89.2%	4.5	78
IT-like	Medical	91.0%	6.8	91
Legal-like	Legal	95.5%	8.1	245
Recommendation	Recommend.	72.8%	3.2	65

Результати демонструють, що кожен пресет забезпечує оптимальний баланс точності та ефективності на відповідному типі даних, підтверджуючи коректність вибору параметрів. Medical пресет на IT-like даних показує вищу точність (91.0% vs 89.2%), але за рахунок більшої кількості питань (6.8 vs 4.5), тоді як IT Support пресет оптимізований саме для мінімізації кількості питань, що підтверджує trade-off між точністю та швидкістю.

Висновки та перспективи подальших досліджень

У роботі представлено Generic IRF v3.0 – параметризовану версію архітектурного патерну Iterative Refinement Funnel, що забезпечує повну конфігураційну гнучкість при збереженні інваріантної бізнес-логіки ядра системи. Основні наукові та практичні результати роботи включають: по-перше, розроблено таксономію параметрів патерну, що систематизує всі точки варіативності у чотири категорії: стратегічні, порогові, функціональні та доменні. Ця класифікація забезпечує структурований підхід до конфігурування системи та полегшує розуміння залежностей між параметрами.

По-друге, реалізовано Configuration Layer - архітектурний шар, що інкапсулює всі параметри у типізованій структурі GenericIRFConfig. Це дозволяє змінювати поведінку системи через конфігурацію замість модифікації коду, що спрощує підтримку та еволюцію системи.

По-третє, створено систему доменних пресетів для чотирьох типових предметних областей: медичної діагностики, технічної підтримки, юридичної класифікації та рекомендаційних систем. Пресети забезпечують швидке розгортання у нових доменах та слугують відправною точкою для подальшої оптимізації.

По-четверте, розроблено механізм автоматичної оптимізації параметрів через Grid Search з підтримкою A/B тестування різних конфігурацій. Це дозволяє систематично покращувати якість системи на основі емпіричних даних.

По-п'яте, забезпечено повну зворотну сумісність з попередньою версією патерну через v22_compatible_config, що дозволяє поступову міграцію існуючих систем без ризику регресій.

Подальші напрямки дослідження включають розробку механізмів автоматичного налаштування параметрів на основі характеристик вхідних даних (meta-learning), інтеграцію з систе-

мами моніторингу для динамічної адаптації параметрів у production-середовищі та розширення системи пресетів на основі досвіду впровадження у реальних проєктах.

Внесок авторів

Максим МЕЛЬНИК – збір і перевірка емпіричних даних, програмне забезпечення; Олексій БИЧКОВ – концептуалізація; методика.

Декларація про штучний інтелект

Штучний інтелект не використовувався.

Конфлікт інтересів

Робота виконана відповідно до принципів академічної доброчесності, етичних норм проведення наукових досліджень та вимог редакційної політики щодо запобігання конфлікту інтересів.

Список використаної літератури

1. *Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.*
2. *Shortliffe, E. H., & Buchanan, B. G. (1975). A model of inexact reasoning in medicine. Mathematical Biosciences, 23, 351–379.*
3. *Pearl, J. (1988). Probabilistic reasoning in intelligent systems. Morgan Kaufmann.*
4. *MacKay, D. J. C. (2003). Information theory, inference, and learning algorithms. Cambridge University Press.*
5. *Settles, B. (2009). Active learning literature survey (CS Technical Report 1648). University of Wisconsin–Madison.*
6. *Shannon, C. E. (1948). A mathematical theory of communication. Bell System Technical Journal, 27, 379–423.*
7. *Kohonen, T. (2001). Self-organizing maps (3rd ed.). Springer.*
8. *Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). Pattern-oriented software architecture: A system of patterns. Wiley.*
9. *Shaw, M., & Garlan, D. (1996). Software architecture: Perspectives on an emerging discipline. Prentice Hall.*
10. *Engelmore, R., & Morgan, T. (1988). Blackboard systems. Addison-Wesley.*
11. *Fowler, M. (2002). Patterns of enterprise application architecture. Addison-Wesley.*
12. *Newman, S. (2015). Building microservices: Designing fine-grained systems. O'Reilly Media.*
13. *Elhaddad, O. M., Alfares, A. A., Alshahrani, S. M., & Bahkali, S. A. (2024). AI-driven clinical decision support systems: An ongoing pursuit of potential. Cureus, 16(4), e57728. https://doi.org/10.7759/cureus.57728*
14. *Park, J., Kim, D., Lee, S., Choi, Y., & Kim, H. (2024). XAI-based clinical decision support systems: A systematic review. Applied Sciences, 14(15), 6638. https://doi.org/10.3390/app14156638*
15. *Kim, S. H., Lee, J. H., Park, Y. S., et al. (2024). Machine learning-based clinical decision support system for treatment recommendation and overall survival prediction of hepatocellular carcinoma: A multi-center study. npj Digital Medicine, 7(4). https://doi.org/10.1038/s41746-023-00976-8*
16. *Pumplun, L., Peters, F., Gawlitza, J. F., & Buxmann, P. (2023). Bringing machine learning systems into clinical practice: A design science approach to explainable machine learning-based clinical decision support systems. Journal of the Association for Information Systems, 24(4), 953–979. https://doi.org/10.17705/1jais.00820*

17. Susanto, A. P., Lyell, D., Widyantoro, B., Berkovsky, S., & Magrabi, F. (2023). Effects of machine learning-based clinical decision support systems on decision-making, care delivery, and patient outcomes: A scoping review. Journal of the American Medical Informatics Association, 30(12), 2050–2063. https://doi.org/10.1093/jamia/ocad180

18. Labkoff, S., Oladimeji, B., Kannry, J., et al. (2024). Toward a responsible future: Recommendations for AI-enabled clinical decision support. Journal of the American Medical Informatics Association, 31(11), 2730–2739. https://doi.org/10.1093/jamia/ocae29

19. Tripathi, K. (2024). The novel hierarchical clustering approach using self-organizing map with optimum dimension selection. Health Care Science, 3, 88–100. https://doi.org/10.1002/hcs2.90

20. Javed, A., Rizzo, D. M., Lee, B. S., & Gramling, R. (2024). SOMTimeS: Self organizing maps for time series clustering and its application to serious illness conversations. Data Mining and Knowledge Discovery, 38(3), 813–839. https://doi.org/10.1007/s10618-023-00979-9

21. Giebel, G. D., Raszke, P., Nowak, H., et al. (2025). Improving AI-based clinical decision support systems and their integration into care from the perspective of experts: Interview study among different stakeholders. JMIR Medical Informatics, 13(1), e69688. https://doi.org/10.2196/69688

22. Li, S. S., Balachandran, V., Feng, S., et al. (2024). MEDIQ: Question-asking LLMs and a benchmark for reliable interactive clinical reasoning. Advances in Neural Information Processing Systems, 37, 28858–28888.

23. Gao, Y., Li, R., Croxford, E., et al. (2025). DR.KNOWS: Leveraging medical knowledge graphs into large language models for diagnosis prediction: Design and application study. JMIR AI, 1, e58670. https://doi.org/10.2196/58670

24. Kell, G., Roberts, A., Ferrari, D., et al. (2024). Question answering systems for health professionals at the point of care: A systematic review. Journal of the American Medical Informatics Association, 31(4), 1009–1024. https://doi.org/10.1093/jamia/ocae01

25. Бичков, О. С., & Мельник, М. О. (2026). Iterative Refinement Funnel: Архітектурний патерн для систем інтерактивної діагностики з асиметричною обробкою відповідей. Зв'язок, (6).

O. Bychkov, M. Melnyk

GENERIC ITERATIVE REFINEMENT FUNNEL: A PARAMETERIZED ARCHITECTURAL PATTERN FOR ADAPTIVE INTERACTIVE DIAGNOSTIC SYSTEMS

*The article is devoted to the development of a parameterized version of the architectural pattern *Iterative Refinement Funnel* (Generic IRF v3.0), aimed at enhancing the flexibility, scalability, and domain independence of interactive diagnostic systems. The proposed approach eliminates the limitations of earlier implementations, in which key parameters were hard-coded, thereby complicating adaptation to new domains and necessitating modifications to the program code. In Generic IRF v3.0, the principle of full configurability is implemented, whereby system behavior is altered through parameter settings without intervention in the algorithmic core.*

The paper introduces a multi-level parameter system comprising strategic, threshold, functional, and domain levels. Strategic parameters define the logic of iterative hypothesis space reduction, including the selection of heuristics, response processing, and termination criteria. Threshold parameters establish numerical constraints to ensure robustness, while functional parameters describe computational procedures, including evaluation metrics and informativeness functions. Domain parameters facilitate the integration of domain-specific knowledge.

*The proposed architectural solution is the *Configuration Layer*, which centralizes the management of all variability points and ensures a separation between invariant logic and configurable components. Within this layer, a system of domain presets is implemented for typical application sce-*

narios, including medical diagnostics, technical support, legal classification, and recommendation systems, thereby accelerating deployment and reducing adaptation costs.

Additionally, a mechanism for automatic parameter optimization based on Grid Search with support for A/B testing of configurations is proposed. This enables empirically grounded selection of optimal settings, taking into account data characteristics and diagnostic quality requirements. Experimental validation across four domains confirms the preservation of diagnostic accuracy (83.3–84.1%) while maintaining full compatibility with existing code and enabling adaptation to new domains through configuration alone, without code modifications. The obtained results demonstrate the feasibility of using the proposed approach as a foundation for building universal intelligent decision support systems.

Keywords: architectural patterns, software parameterization, interactive diagnostics, configuration flexibility, domain-specific adaptation, Strategy Pattern, decision support systems.

Надійшла до редакції: 06.07.2026
Прийнята до друку: 20.07.2026
Опубліковано: 17.08.2026