https://doi.org/10.31891/2219-9365-2025-81-46
УДК 004.05:004.94:004.41

РИХАЛЬСЬКИЙ Олексій
ПВНЗ «Європейський університет»
https://orcid.org/0009-0000-6892-9420
email: alexey.rykhalskiy@e-u.edu.ua

ЗВУЖЕННЯ ОБЛАСТЕЙ ВИЗНАЧЕННЯ, ЯК ШЛЯХ ДО ПІДВИЩЕННЯ КОРЕКТНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті досліджено проблему виникнення часткових функцій у процесі моделювання та розробки програмного забезпечення, що зумовлено недостатнім або неформалізованим описом області визначення моделей. У складних предметно-орієнтованих системах, де моделі часто інтегруються з різних джерел або розробляються міждисциплінарними командами, звуження області допустимих вхідних значень є критичним інструментом для забезпечення узгодженості, коректності та передбачуваності роботи програмних компонентів.

Показано, що звуження областей визначення функцій у програмуванні виникає як наслідок математичних, технічних та предметно-орієнтованих (контекстуальних) обмежень.

Увагу зосереджено на випадках, коли обмеження не є явно визначеними або реалізуються лише на рівні інженерної інтуїції. У таких ситуаціях вони не фіксуються ані в типах, ані в коді, що унеможливлює їх автоматичну перевірку, повторне використання моделей або їх перенесення в інші домени без втрати достовірності. Відсутність формального опису області визначення може призводити до порушення логіки функціонування системи, генерації некоректних або суперечливих результатів, що особливо критично в контексті відповідальних застосувань — медицини, безпілотних систем, енергетики тощо.

Особливий акцент зроблено на міждисциплінарному аспекті проблеми, коли моделі, створені в одній предметній площині, переносяться до іншої без належного уточнення області визначення. Такий перенос часто супроводжується збереженням неявних обмежень, які були релевантні в первинному контексті, але є непридатними або навіть хибними в новому середовищі. У результаті система, що виглядала стабільною, демонструє часткову або непередбачувану поведінку, що може вивести з ладу критично важливі процеси.

Запропоновано методологічні підходи до формалізації звуження області визначення як засобу підвищення якості програмного забезпечення. Підхід охоплює формалізацію допустимих значень на рівні типів і специфікацій. Стаття формує міждисциплінарний погляд на проблему коректності програмних моделей, що працюють із частковими функціями, та окреслює практичні шляхи її вирішення шляхом формального звуження областей визначення.

Ключові слова: інформаційні технології, моделювання, інтеграція, відчуження знань, контекст, неявний контекст, узгодження, несуперечливість, область визначень, область значень, обробка даних, повторне використання моделей.

RYKHALSKYY Oleksiy
Private Higher Education Institution «European University»

NARROWING THE DOMAINS OF DEFINITION AS A PATH TO IMPROVING SOFTWARE CORRECTNESS

The article explores the problem of partial functions that arise in the process of modeling and software development due to insufficiently defined or non-formalized domains of model inputs. In complex domain-oriented systems—where models are often integrated from diverse sources or developed by interdisciplinary teams—narrowing the range of admissible input values is a critical tool for ensuring consistency, correctness, and predictability of software components.

It is shown that the narrowing of function domains in programming arises as a result of mathematical, technical, and domain-specific (contextual) constraints.

The focus is placed on cases where such constraints are not explicitly defined or exist only as part of an engineer's implicit intuition. In these cases, constraints are neither reflected in type systems nor encoded in the logic, making automatic validation, model reuse, or transfer to other domains impossible, unreliable or even hazardous. The lack of formal domain specifications can lead to violations of system logic and the generation of incorrect or contradictory results—risks that are especially critical in high-stakes applications such as healthcare, autonomous systems, and energy infrastructures.

A particular emphasis is placed on the interdisciplinary nature of the problem, where models created in one application domain are transferred to another without adequately revisiting the original domain assumptions. Such transfer often preserves implicit constraints that are no longer valid or even harmful in the new context. As a result, systems that appeared stable may exhibit partial or unpredictable behavior, potentially disrupting critical processes.

The article proposes methodological approaches to the formal narrowing of input domains as a means of improving software reliability. This includes the specification of admissible values, domain boundaries, and edge-case handling within the programming and modeling infrastructure. The work contributes an interdisciplinary perspective on the correctness of software models operating with partial functions and outlines practical strategies for mitigating related issues through domain constraint formalization.

Keywords: information technologies, modeling, integration, knowledge transfer, context, implicit context, consistency, non-contradiction, domain of definition, codomain, data processing, model reusing.

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

У сучасній теорії математичного моделювання будь-який процес або явище, що піддається формалізації, зрештою описується функціональною залежністю вигляду:

$$y = f(x),$$

де:

x – вхідний параметр або набір параметрів (вектор входів),

y – вихідний параметр, тобто результат функціонування моделі або результат обчислень, здійснених у межах моделювання. Кожна така функція визначена лише на певній множині допустимих вхідних значень — області визначення $Dom(x)$, яка задається виходячи з припущень, обмежень і контексту, у якому модель створювалася. Саме ця область визначає, для яких значень вхідних параметрів модель є релевантною, коректною та інтерпретованою з точки зору предметної області.

При моделюванні складних або багатокомпонентних систем часто виникає потреба використовувати не одну, а декілька моделей, кожна з яких описує окремий аспект або підсистему загального процесу. Формально це означає наявність декількох функцій, наприклад:

$$y_1 = f_1(x)$$
$$y_2 = f_2(x)$$

Оскільки кожна модель побудована на основі власного набору припущень і орієнтована на опис окремого аспекту поведінки системи, області визначення цих функцій зазвичай не збігаються. Тобто:

$Dom(f_1) = Dom_1$

$Dom(f_2) = Dom_2$

і в загальному випадку: $Dom_1 \neq Dom_2$.

У цьому контексті постає принципово важливе питання: чи можливо та доцільно використовувати ці моделі сумісно для моделювання єдиної складної системи? В залежності від взаємозв'язку між Dom_1 і Dom_2 можливі два базові варіанти:

$Dom_1 \cap Dom_2 \neq \emptyset$ — області визначення моделей мають спільну частину. У цьому випадку виникає необхідність проаналізувати, чи достатньо інформації в цій спільній підмножині для моделювання всієї системи. Інакше кажучи, чи дозволяє звужена область $Dom_1 \cap Dom_2 \neq \emptyset$ виконати необхідні обчислення без втрати значущих аспектів моделі.

У випадку, коли $Dom_1 \cap Dom_2 = \emptyset$, маємо справу з несумісними областями визначення. Це означає, що відповідні моделі не мають жодної спільної множини вхідних параметрів, що унеможливлює їхнє безпосереднє об’єднання в межах однієї узагальненої моделі. Така ситуація є особливо критичною у випадку складних систем, де необхідна комплексна оцінка на основі кількох аспектів, що охоплюються різними моделями.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Аналіз джерел дозволив здійснити попередню класифікацію існуючих підходів до підвищення якості та коректності програмного забезпечення.

Базове представлення даних. У роботах [8, 11] центральним об'єктом дослідження є факт обмеження значень які можуть бути репрезентовані базовими типами мов програмування. Наводяться приклади формально визначених доменів для базових типів, зокрема *Dom(byte) = [0, 255], Dom(short) = [0, 65535], Dom(signed byte) = [-128, 127], Dom(signed short) = [-32768, 32767]* тощо. Таке жорстке визначення діапазонів значень з однієї сторони відіграє ключову роль у створенні більш передбачуваних моделей даних, де кожна змінна або структура об'єктивно прив'язана до допустимої множини значень. Але з іншої сторони ці деталі можуть бути втрачені під час передачі моделі з фази розробки до фази реалізації конкретним інженером.

Окрему увагу приділено підходам, де обмеження неявно закладені в дизайн мови — наприклад, у вигляді модифікованих типів, спеціальних анотацій або вбудованих механізмів перевірки значень. Це дозволяє значно знизити ймовірність того, що програміст ненавмисно може передати або обробить значення, яке виходить за межі допустимого контексту. Встановлення таких обмежень розглядається не лише як засіб захисту, а й як потужний інструмент для більш точного моделювання реальних об’єктів, процесів та станів, з якими працює програмна система.

Таким чином, результати [8, 11] підкреслюють важливість звуження області допустимих значень під час визначення типів у мові програмування. Це дозволяє фіксувати припущення щодо предметної області безпосередньо у коді, уникати зайвої неоднозначності, спрощувати логіку та полегшувати подальший аналіз і модифікацію моделей.

Проблематика використання чисел з плаваючою комою у сучасному програмуванні.

У дослідженнях [1, 8, 15] докладно розглядається низка фундаментальних обмежень, притаманних числовим типам *float* та *double*, що широко використовуються для представлення дійсних чисел у більшості мов програмування. Одним із ключових аспектів є обмеженість діапазону, в межах якого можуть існувати значення цих типів: *Dom(float) = (−3.4×10³⁸, 3.4×10³⁸), Dom(double) = (−1.7×10³⁰⁸, 1.79×10³⁰⁸)*. Хоча ці діапазони виглядають досить широкими, вони не дозволяють точно подати всі значення дійсних чисел у цих межах — і саме це обмеження точності стає джерелом численних труднощів.

Найбільш відомою проблемою є те, що операції з числами з плаваючою комою не гарантують очікуваного арифметичного результату через обмежену кількість бітів, що виділяються на мантису. Наприклад, у більшості середовищ, у форматі *float:* 0.1 + 0.2 ≠ 0.3, та 16777216 + 1 ≠ 16777217. Це пов'язано з тим, що точність представлення чисел у типах з плаваючою комою є обмеженою, і значення округлюються до найближчого, що може бути представлене у відповідному форматі IEEE 754. Формалізовано це описується як невизначеність результатів обчислень, зумовлена тим, що операції з плаваючою комою не є асоціативними. Це означає, що порядок виконання операцій може впливати на кінцевий результат: (a + b) + c ≠ a + (b + c). Така властивість особливо критична в паралельних обчисленнях, де порядок операцій може варіюватися між запусками програми, навіть при однакових вхідних даних. Це призводить до серйозних проблем з оптимізацією та інтерпретацією результатів. Також унеможливлює відтворюваність результатів, що є критично важливим у наукових обчисленнях, симуляціях або фінансових розрахунках.

Математичне звуження області визначення функцій. У дослідженнях [9, 12] акцент зроблено на випадках, коли функції мають внутрішньо притаманні математичні обмеження, що не дозволяють застосовувати їх до всіх можливих числових значень. На відміну від технічних обмежень, пов'язаних із розрядністю або типами даних у програмуванні, у цьому випадку йдеться про функціональні обмеження, які зумовлені самою природою математичних операцій.

Так, для функції $\sqrt{x}$ область визначення обмежується $x \geq 0$, якщо розглядати лише дійсні числа, функція $1/x$ не є визначеною для $x = 0$, що очевидно звужує область визначення. Подібні обмеження стосуються широкого класу елементарних та спеціальних функцій:

- *ln(x)* визначена лише для x > 0;
- *arccos(x)* — лише на інтервалі [−1, 1];

Окрім обмеження області визначення, багато функцій є небезпечними з обчислювальної точки зору у певних межах. Наприклад, функція *exp(x)* швидко росте при великих *x*, що може спричинити переповнення, тоді як *log(x)* поблизу нуля може дати надзвичайно великі від'ємні значення або взагалі викликати виняткову ситуацію. У прикладному програмуванні ці властивості часто ігноруються, що призводить до непередбачуваної поведінки.

Таким чином, математичне звуження є не лише теоретичною вимогою, а й необхідною практикою для забезпечення коректної роботи програмних компонентів, особливо у випадках, коли працюють із символьними обчисленнями, науковими моделями або інженерними симуляціями.

Контекстуальне звуження області визначення у моделюванні реальних систем. У дослідженнях [6, 14] наголошується, що при побудові моделей для реальної предметної площини надзвичайно важливо враховувати контекстуальні обмеження, які задають допустимий діапазон значень параметрів. На відміну від формального або математичного звуження, у прикладному моделюванні маємо справу з предметно зумовленим звуженням, що відповідає фізичному, технологічному або нормативному сенсу параметрів.

Майже в усіх галузях моделі мають сенс лише в межах вузького операційного діапазону, за межами якого вони можуть давати некоректні або навіть небезпечні рекомендації. Наприклад:

- модель росту популяції застосовується лише в умовах сталого середовища;
- тепловий розрахунок електронного компонента вірний у межах температури (−40, +85) °C;
- економетричні моделі дійсні лише за припущенням помірної інфляції або стабільного ринку;
- модель, що описує швидкість руху транспортного засобу на автомагістралі, матиме сенс лише в межах певного діапазону: $V \in (10, 100)$ км/год.

Поза цими межами модель або перестає відповідати реальності, або її результати втрачають практичну корисність. Таке звуження області визначення на основі доменної специфіки є фундаментальним етапом моделювання, який формує припущення щодо коректності, стабільності та застосовності отриманих результатів.

Такі обмеження є не просто технічними деталями, а необхідною частиною верифікації моделей, які інакше можуть бути хибно застосовані в умовах, де початкові припущення не виконуються. Ігнорування реальної області визначення призводить до неможливості інтерпретації результатів, зниження достовірності моделі та навіть до непридатності для прийняття рішень.

Неявні обмеження як частина інженерної практики. Неявні обмеження мають різну природу і детально розібрані в [4, 5, 7, 10]. Монографія [16] приділяє особливу увагу неявному контексту. Тобто маємо справу з обмеженням області визначення в прикладному моделюванні, оскільки вона вважаються «очевидною» для фахівців, які працюють у конкретній предметній області. Це явище характерне передусім для інженерного мислення, де частина знань існує в неформалізованій, інтуїтивно засвоєній формі й не завжди документується або відображається у самій моделі.

Цей підхід ефективний у колі експертів, які працюють з однаковими припущеннями [2], однак ускладнює масштабування моделей, їх повторне використання, автоматизовану верифікацію та передачу між командами або інституціями. Відсутність явно зафіксованих обмежень може призвести до некоректного

використання моделей у нових умовах або некваліфікованими користувачами, особливо при зміні технологічного контексту, параметрів середовища або масштабу обчислень.

Особливо це критично в міждисциплінарних проєктах, де інженерні припущення повинні бути перенесені в інші домени (економіка, IT, логістика), де «зрозуміле з досвіду» вже не є зрозумілим для всіх. Тут формалізація обмежень є логічним на шляху до підвищення якості та можливості повторного використання моделей.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою статті є здійснення ґрунтовного аналізу типових та нетривіальних ситуацій, що виникають у процесі моделювання та розробки програмного забезпечення, коли використовуються моделі з обмеженою областю визначення, які, у свою чергу, призводять до появи часткових функцій. Особливу увагу приділено ситуаціям, у яких обмеження на вхідні параметри моделей зумовлені як прикладною специфікою задач, так і особливостями мов програмування та обчислювальних середовищ.

Аналіз проводиться з фокусом на мови програмування, які, з одного боку, мають власні нюанси реалізації — зокрема синтаксичні та семантичні обмеження, нюанси репрезентації даних, що додатково звужують простір рішень для інженера, — а з іншого боку, надають низку вбудованих засобів для контролю коректності.

Окремим напрямом розгляду є обґрунтування важливості ретельного аналізу та узгодження областей визначення моделей, особливо у випадках перенесення або адаптації моделей із суміжних або «схожих» предметних областей. Такий перенос часто супроводжується несвідомим збереженням початкових обмежень, які можуть виявитися неадекватними новому контексту застосування. Унаслідок цього модель, що була коректною в первинному середовищі, стає частковою або навіть хибною при інтеграції до нової системи. Саме тому системна оцінка області визначення як формальної, так і фактичної, є критично важливою умовою забезпечення коректності та цілісності моделювання, особливо в складних або міждисциплінарних застосуваннях.

Стаття спрямована на формування методологічної основи для інтеграції обмежень області визначення у процес формального моделювання, з метою підвищення достовірності, повторюваності та міждисциплінарної узгодженості програмних систем, що базуються на часткових функціях.

Методи досліджень. У цьому дослідженні використано системний і структурно-аналітичний підхід, орієнтований на всебічне вивчення чинників, що впливають на роль області визначення функцій у побудові надійних програмних моделей. Центральним напрямом дослідження є аналіз формальних і неформальних підходів до обмеження областей визначення у процесі предметно-орієнтованого моделювання.

Дослідження базується на систематизації сучасних теоретичних джерел і практичних реалізацій, що стосуються гарантування визначеності поведінки програм при роботі з обмеженими або умовно обмеженими множинами вхідних даних. Ключовими критеріями для аналізу інструментів були: наявність явного або неявного обмеження області визначення, здатність інструмента виявляти виконання поза межами допустимих входів, тип застосованого механізму контролю (системи типів, перевірки передумов, верифікація), а також відповідність розглянутих підходів вимогам сучасної програмної інженерії.

Важливою складовою дослідження виступає практичний досвід автора, який охоплює понад три десятиліття у сфері промислового програмування та понад десятиріччя педагогічної діяльності, зосередженої на викладанні дисциплін, пов'язаних із аналізом, моделюванням і побудовою надійних програмних систем. Це дозволило поєднати глибину теоретичного аналізу з практичною доцільністю, забезпечуючи обґрунтованість та релевантність сформульованих висновків.

Аспекти звуження області визначення. Будь яке моделювання в кінцевому вигляді описується функцією:

$$y = f(x),$$

де:

f — назва функції,

x — вхідній параметр,

y — вихідній параметр (результат функції, результат моделювання).

Важливо підкреслити, що x та y також можуть бути кортежами, тобто:

$x = \langle x_1, x_2, \dots x_n \rangle$,

що фактично означає $y = f(x_1, x_2, \dots x_n)$, тобто функція може приймати декілька параметрів,

$y = \langle y_1, y_2, \dots y_m \rangle$,

що фактично означає $\langle y_1, y_2, \dots y_m \rangle = f(x_1, x_2, \dots x_n)$, або $\langle y_1, y_2, \dots y_m \rangle = f(x)$, тобто функція може повертати декілька результатів.

В моделюванні реального світу, в більшості випадків $x_i \in \mathbb{R}$, $y_j \in \mathbb{R}$. Тривіальними прикладами таких функцій можуть бути: $y = 3 \cdot x$, $y = x^2$, $y = x_1^2 + 3x_2 + 1$, та ін.

В класичній математиці областю визначення параметрів x, x_1, x_2 є діапазон $(-\infty, +\infty)$.

Але…

Звуження області визначення, обумовлене математичними обмеженнями. Одним із фундаментальних джерел звуження області визначення функціональних залежностей є математична природа самих функцій. При формальному описі виникають об'єктивні обмеження, які унеможливлюють застосування функції до деяких вхідних значень. Це звуження є незалежним від технічної реалізації або мови програмування і зумовлене виключно властивостями математичних операцій, які використовуються у виразі функції.

Розглянемо декілька прикладів математичних функцій, кожна з яких має чітко визначені області допустимих значень аргументів:

$$y = \sqrt{x}$$

Ця функція визначена лише тоді, коли підкореневий вираз невід'ємний, тобто $x \geq 0$. Таким чином, область визначення становить $[0, +\infty)$.

$$y = {}^{1}/_{x}$$

Поділ на нуль є забороненою операцією в математиці, тому функція не визначена при $x=0$. Відповідно, область визначення: $(-\infty, 0) \cup (0, +\infty)$.

$$y = \sqrt{x - 5}$$

Тут підкореневий вираз $x-5$ повинен бути невід'ємним, що означає $x \geq 5$. Відповідно, область визначення: $[5, +\infty)$.

$$y = \sqrt{7 - x}$$

У цьому випадку, навпаки, вираз $7-x$, який під коренем має бути не від'ємним: $7-x \geq 0$, тобто $x \leq 7$. Отже, область визначення: $(-\infty, 7]$.

$$y = {}^{1}/_{\sqrt{2 - x}}$$

Тут підкореневий вираз повинен бути додатним, адже корінь квадратний з нуля дає нуль, а поділ на нуль заборонений. Тобто: $2-x>0$, що еквівалентно $x<2$. Область визначення: $(-\infty, 2)$.

Ці приклади ілюструють, що навіть при побудові суто аналітичної (математичної) моделі обмеження на допустимі значення аргументів виникають безпосередньо з структури самої функції.

Особливо критичним математичне звуження стає у випадках, коли функції є складеними або використовуються як компоненти в більших системах. Наприклад, при складанні моделей у вигляді композицій функцій, область визначення всієї системи є перетином областей визначення кожної з функцій-компонентів. Так, якщо одна функція вимагає $x>0$, а інша — $x<2$, то обчислення буде коректним лише на перетині цих множин, тобто в діапазоні $(0, 2)$.

Таким чином, математичне звуження області визначення не лише неминуче, але й критично важливе для забезпечення коректності моделі. Воно вимагає від розробника або аналітика не лише знання загальних правил обробки функціональних виразів, але й уважного формального аналізу кожної математичної залежності, яка використовується в системі.

Звуження області визначення, обумовлене типами даних мови програмування. Рано чи пізно будь-яка модель, після етапів побудови, апробації та перевірки, переходить у фазу практичного використання. На цьому етапі вона реалізується в обчислювальній системі засобами конкретної мови програмування. Це автоматично вводить додаткові обмеження на область допустимих вхідних значень та результати функціонування, які не завжди були враховані на етапі моделювання. Звуження виникає через специфіку представлених у мові типів даних та їхніх технічних характеристик.

Більшість сучасних мов програмування використовують стандартні числові типи, такі як *byte, short, int, long* для представлення цілих чисел $\mathbb{Z}$. Але вже ці типи накладають певні обмеження:

$$Dom(unsigned\ byte) = [0, 255],\ Dom(signed\ byte) = [-128, 127],$$
$$Dom(unsigned\ short) = [0, 65535],\ Dom(signed\ short) = [-32768, 32767],$$
$$Dom(unsigned\ int) = [0, 2^{32}-1],\ Dom(signed\ int) = [-2^{31}, 2^{31}-1],$$
$$Dom(unsigned\ long) = [0, 2^{64}-1],\ Dom(signed\ long) = [-2^{63}, 2^{63}-1].$$

Для представлення дійсні числових даних $\mathbb{R}$ використовуються типи *float, double* які також накладають певні обмеження:

$$Dom(float) = (-3.4\times10^{38}, 3.4\times10^{38}),\ Dom(double) = (-1.7\times10^{308}, 1.79\times10^{308})$$

Хоча ці діапазони виглядають досить широкими, вони не дозволяють точно подати всі значення дійсних чисел у цих межах завдяки обмеженню представлення у двійковому форматі. Більш серйозні проблеми детально описані в [1].

Таким чином, типи даних, використовувані в мові програмування, виступають технічними обмежувачами області визначення. Це не лише накладає відповідальність на розробника щодо перевірки припустимих значень, але й вимагає переосмислення ймовірних граничних випадків, які раніше могли здаватися теоретично безпечними.

Контекстуальне звуження області визначення у моделюванні реальних систем. Одним із ключових аспектів забезпечення коректності моделей у прикладному моделюванні є контекстуальне звуження області визначення [3, 7, 10]. На відміну від математичних або технічних обмежень, які випливають із властивостей функцій або реалізації на конкретній мові програмування, контекстуальні обмеження виникають безпосередньо з особливостей предметної галузі, до якої належить об'єкт моделювання.

Під контекстом [16] у даному випадку розуміється предметно обумовлене оточення, у межах якого модель має сенс і може давати коректні результати. Це включає фізичні, технологічні, економічні, нормативні та інші межі, які визначають допустимі значення вхідних параметрів та обґрунтованість самих функціональних залежностей.

У реальних системах діапазон значень більшості параметрів є обмеженим не лише фізичними властивостями, а й нормативними вимогами, конструктивними характеристиками, або експлуатаційними умовами. Наприклад, маса легкового автомобіля зазвичай варіюється в межах від 1000 до 3000 кг. Побудова моделі з припущенням, що маса може набувати значень у довільному діапазоні, включаючи від'ємні або надзвичайно великі значення, не просто некоректна — вона безпосередньо суперечить реальності.

Контекстуальне звуження є обов'язковою умовою семантичної валідності моделі і є характерними для різноманітних галузей.

Модель швидкості руху автомобіля на автомагістралі матиме зміст лише в діапазоні $V \in [40,130]$ км/год. Швидкості менші за 40 км/год не відповідають характеру автомагістралі, а значення понад 130 км/год, зазвичай, заборонені законодавством і можуть спричинити помилкову оцінку ризиків або часу прибуття.

Розрахунок теплових режимів для електронних компонентів має сенс у межах робочої температури, наприклад від −40°C до +85°C. За межами цього діапазону фізичні властивості матеріалів змінюються, і модель стає непридатною для оцінки теплових навантажень.

Економетричні моделі, що базуються на припущенні про стабільність ринку, перестають бути актуальними в умовах гіперінфляції або війни. Наприклад, модель прогнозування споживчого попиту на основі історичних даних втратить актуальність при різкій зміні купівельної спроможності населення.

Моделі росту популяцій часто побудовані на припущенні про стабільне середовище. Зміни у кліматичних або харчових умовах можуть повністю зруйнувати передбачення моделі, якщо не було враховано контекстуальних меж.

Фармакокінетичні моделі базуються на діапазонах дозувань, допустимих для певних груп пацієнтів. Вихід за межі допустимих доз може зробити модель не лише некоректною, а й небезпечною.

Ігнорування контекстуальних меж не лише знижує точність обчислень, але й підриває довіру до моделі як інструменту прийняття рішень. У практиці системного моделювання, особливо в критичних галузях — таких як авіація, атомна енергетика, оборонна промисловість — звуження області визначення за контекстними ознаками є обов'язковою складовою стадії перевірки моделі.

Крім того, контекстуальне звуження необхідне для забезпечення можливості тестування та повторного використання моделей. Наприклад, модель, що передбачає використання лише в умовах середніх навантажень, не повинна бути застосована для розрахунків у пікових режимах без відповідної модифікації або масштабування.

Контекстуальне звуження області визначення є не менш важливим, ніж формальне або математичне. Воно дозволяє точно визначити діапазон застосовності моделі, забезпечує можливість коректної інтерпретації достовірності результатів і, найголовніше, — узгодженість із реальним світом. У системах, що працюють із реальними даними, врахування контексту є запорукою безпеки, стабільності та ефективності.

Неявний контекст як джерело звуження області визначення в моделюванні. У процесі побудови [12] та використання [13] моделей для опису складних систем надзвичайно важливим є не лише формальне визначення структури моделі та математичних залежностей, а й врахування контексту, в межах якого ця модель має сенс. У цьому контексті особливої уваги заслуговує явище неявного контексту [2] — такого, що не описаний формально, але критично впливає на область визначення функції, модель або алгоритм.

Проблема втрати контексту при передачі моделей. Однією з найскладніших проблем у розробці складних прикладних моделей є збереження та передача контекстної інформації між учасниками життєвого циклу моделі: від аналітика або інженера-моделювальника до розробника програмного забезпечення, тестувальника або кінцевого користувача. Часто контекст — включаючи діапазони значень параметрів, умовні припущення та обмеження застосовності — існує лише в уявленні інженера або розробника, або ж зберігається в неструктурованій формі, наприклад, у технічній документації, коментарях чи усних домовленостях. У результаті інженер або програміст, який пізніше використовує модель, не отримує критично важливої інформації, що веде до використання моделі поза її реальною областю визначення, що гарантовано

призводить до отримання хибних результатів. Отримання хибних результатів призводить для «відсіювання» моделі і початку проєктування нової, що значно підвищує вартість розробки та її строки.

Класичний приклад: маса в класичній та релятивістській фізиці. Один із показових прикладів неявного контексту — це використання параметра маси в двох формально подібних, але принципово відмінних фізичних моделях:

Класичний другий закон Ньютона: $F = m \cdot a$, застосовується у випадках, коли швидкість руху об'єкта значно менша за швидкість світла у вакуумі $v \ll c$.

Формула Ейнштейна в межах спеціальної теорії відносності: $E = m \cdot c^2$, використовується, коли об'єкти рухаються зі швидкостями, наближеними до швидкості світла у вакуумі, тобто $v \approx c$.

Хоча обидві формули містять параметр m, це не одна і та ж маса, а концептуально різні фізичні поняття. У першому випадку — це маса у класичному сенсі, у другому — релятивістська маса, яка залежить від швидкості. Якщо цю відмінність не зафіксувати або не передати через інтерфейс моделі, користувач може застосувати її в некоректному контексті, отримавши хибні результати.

Припущення, що не були передані. Ще одним проявом неявного контексту є припущення, які залишаються на рівні інженерного мислення, але не документуються або не описуються формально. У таких випадках розробник моделі, добре обізнаний із предметною галуззю, може не вважати за потрібне фіксувати граничні значення параметрів, технологічні обмеження чи експлуатаційні припущення, вважаючи їх «очевидними» або «зрозумілими для всіх». Цей підхід може бути прийнятним у межах однієї команди або проєкту, але призводить до катастрофічних наслідків у випадку передачі моделі іншому фахівцю, який не володіє тією ж глибиною знань. Як наслідок — модель застосовується до значень, які ніколи не тестувалися і не аналізувалися, що призводить до суттєвих похибок або аварійного завершення обчислювального процесу.

Приклад із промислової практики. Розглянемо приклад моделі керування температурою в промисловій печі, яка була розроблена для роботи в діапазоні від +600°C до +1000°C. Цей діапазон не був явно зафіксований у коді, оскільки інженери вважали його очевидним. Проте при розширенні системи та застосуванні тієї ж моделі для іншого типу печі, яка працює при температурах до +300°C, програмне забезпечення функціонувало, але регулятори поводилися неправильно. Причиною стала втрата неявного контексту — модель була некоректно використана поза її фактичною областю застосування, хоча формально вона працювала.

Особливо небезпечною є ситуація, коли модель є «чорним ящиком» — наприклад, нейронна мережа, — і розробник або користувач не має доступу до її внутрішньої логіки та реалізації. Якщо при цьому відсутня або втрачена інформація щодо припустимих діапазонів вхідних даних, умов навчання або природи вихідних інтерпретацій, така модель стає вразливою до некоректного застосування. Навіть якщо у поточних умовах вона демонструє прийнятну точність, з часом, внаслідок змін у зовнішньому середовищі або типах вхідних даних, майже гарантовано відбудеться помилка в інтерпретації або використанні результатів, що призведе до серйозних проблем. У таких випадках відсутність явного або задокументованого контексту є критичним недоліком, що ставить під сумнів надійність та валідність всієї системи, яка використовує цю модель.

Шляхи фіксації та передачі неявного контексту. Вирішення цієї проблеми потребує спеціальних механізмів формалізації контекстної інформації, зокрема:

- введення метаданих до моделей, що фіксують діапазони допустимих значень, одиниці виміру, припущення щодо середовища використання;
- використання контрактів або пре- та пост-умов у реалізації функцій, які перевіряють входи перед виконанням;
- документування не лише алгоритмів, а й умов застосовності моделей, включаючи граничні сценарії, для яких модель була протестована;
- застосування систем з обмеженнями на рівні типів, що дозволяють зафіксувати допустимі діапазони.

Неявний контекст є невід’ємною, хоча й не завжди очевидною складовою області визначення моделі. Саме тому завдання формалізації, фіксації та передачі неявного контексту слід вважати ключовим етапом у забезпеченні надійності та достовірності моделювання.

Перенесення моделей між предметними галузями. Одним із найцікавіших і водночас найскладніших напрямів у сучасному моделюванні є проблема перенесення моделей з однієї предметної площини до іншої. Такий перенос стає дедалі актуальнішим у сучасному науковому і прикладному середовищі, де повторне використання моделей дозволяє суттєво скоротити витрати часу, ресурсів та зменшити складність побудови нових рішень. В ідеалі, адаптація вже існуючих моделей мала б відбуватись швидко та ефективно, забезпечуючи можливість повторного застосування перевірених підходів у нових умовах.

Проте на практиці перенесення моделей часто стикається з неочевидними труднощами, головним джерелом яких є прив’язаність моделей до предметної області, що моделюється. Модель, яка була коректно

побудована, перевірена та ефективно функціонувала в одній галузі, при перенесенні в іншу часто втрачає адекватність, що зумовлено неврахуванням або втратою контексту — як явного, так і неявного.

Ще більш складним є виявлення неявного контексту, що включає приховані припущення розробника моделі, особливості середовища її створення, методи попередньої валідації, або специфіку вихідних даних, які використовувались під час побудови. Результуюча модель може містити в собі ряд припущень щодо лінійності процесів, відсутності шуму, стабільності зовнішніх факторів або навіть політичного контексту, який ніяк не позначений у формальних елементах моделі.

Наприклад, економетрична модель, побудована для ринку зі стабільною інфляцією та зрозумілою податковою політикою, не може бути без адаптації застосована до ринку, що перебуває у стані війни або глибокої кризи. Формально функція працюватиме, але її результати будуть абсолютно хибними через невідповідність неявного контексту.

Для забезпечення безпечного та коректного перенесення моделей доцільно використовувати систему перевірочних критеріїв, що охоплює обидва рівні контексту:

- Чи відповідають вхідні параметри нового середовища структурі та діапазонам вихідної моделі?
- Чи враховано всі одиниці виміру та типи шкал (лінійна, логарифмічна тощо)?
- Чи є припущення про сталість або ізольованість системи релевантними в новому контексті?
- Чи зберігається природа зв’язків між параметрами при переході до нової предметної площини?
- Які критичні припущення залишились непозначеними у вихідній моделі, але мають вплив на результати?

Найефективнішим інструментом для підвищення придатності моделей до перенесення є формалізація контекстної інформації. Мова йде не лише про супровідну документацію, а про інтеграцію метаданих у саму структуру моделі: фіксацію одиниць виміру, обмежень, структур даних, а також інтерпретацій вхідних та вихідних параметрів. Формальні описи, які містять явні посилання на застосовність і межі валідності, дозволяють використовувати моделі повторно з мінімальними ризиками.

Одиниці вимірювання. Одним із критично важливих, але часто недооцінених аспектів моделювання складних систем є врахування одиниць вимірювання вхідних та вихідних параметрів моделі. Навіть за умови коректно визначеної структури моделі, чітко зафіксованої області визначення та формальної верифікації її математичних залежностей, ігнорування або неправильне тлумачення одиниць вимірювання може повністю зруйнувати її коректність. У багатьох випадках такі помилки залишаються непоміченими аж до моменту отримання хибних або катастрофічних результатів.

Класичним прикладом є втрата міжпланетного апарата Mars Climate Orbiter у 1999 році. Модель управління тягою працювала згідно з американською системою вимірювань, а команда, яка готувала телеметричні дані, працювала в метричній системі. Формально модель залишалась функціональною, але через невідповідність одиниць вимірювання, значення імпульсу були інтерпретовані неправильно, що призвело до критичної похибки в розрахунку траєкторії. Космічний апарат увійшов у атмосферу Марса під занадто низьким кутом і був знищений. Цей приклад яскраво демонструє, що навіть високоточна модель втрачає свою корисність, якщо її контекст — у тому числі одиниці вимірювання — не є явно зафіксованим, узгодженим і контрольованим.

Помилки, пов’язані з одиницями вимірювання [10], трапляються в усіх сферах. Їхня природа полягає у тому, що одиниці вимірювання не є частиною математичного виразу, а, як правило, належать до описового рівня (коментарів, документації, інтерфейсів). Якщо система введення даних не виконує автоматичну перевірку або перетворення одиниць, модель залишається вразливою до людських помилок.

У галузі розробки програмного забезпечення одиниці вимірювання часто ігноруються на рівні типів, і програмісти маніпулюють «голими числами», не перевіряючи, чи вони представляють, наприклад, метри чи міліметри, секунди чи хвилини. Це призводить до семантичних помилок, які не завжди виявляються на етапі тестування.

Сучасна інженерна практика вимагає формалізації одиниць вимірювання як складової моделі. Це означає, що у всіх параметрах мають бути чітко зафіксовані одиниці, автоматичне приведення до стандартної системи має бути частиною обчислювального процесу, порушення очікуваних одиниць повинно викликати помилку на рівні компіляції або виконання.

Деякі конвертації одиниць вимірювання можуть виконуватись автоматично, однак це можливо лише за умови, що одиниці чітко задані та представлені у формалізованому вигляді. Для цього необхідно використовувати спеціальні інструменти та підходи, які дозволяють прив’язати одиниці виміру до скалярних значень у програмному коді, спеціальні бібліотеки або системи статичної перевірки одиниць на етапі компіляції.

Інтеграція моделей для моделювання ширшої, більш складної предметної площини. У сучасному науково-технічному середовищі дедалі частіше виникає потреба у побудові комплексних моделей [6, 16], які охоплюють ширші й більш складні предметні області, ніж ті, для яких первинно створювалися окремі

компоненти. Цей підхід, відомий як інтеграція моделей, передбачає поєднання кількох існуючих моделей в одну узгоджену систему з метою опису більш складних, міждисциплінарних процесів.

Інтеграція моделей є основою сучасних підходів у таких галузях, як індустрія 4.0, системна біологія, цифрові двійники, інтелектуальні транспортні системи, де жодна окрема модель не здатна повністю описати всі аспекти цільової системи. Проте такий підхід несе у собі низку складнощів — зокрема, виявлення та узгодження області визначення кожної з моделей, які залучаються до інтеграції.

Ключовим питанням при інтеграції декількох моделей є визначення перетину їх областей визначення. Формально, якщо маємо дві моделі M_1 та M_2 з відповідними областями визначення Dom_1 та Dom_2, то для узгодженого використання їх у рамках єдиної моделі необхідно, щоб було визначено й проаналізовано перетин $Dom_1 \cap Dom_2$.

Якщо цей перетин порожній або практично незначний, спроба поєднати такі моделі є методологічно хибною, оскільки результати моделі не матимуть значення в жодному реально можливому сценарії. Якщо ж перетин існує, але не задокументований або нечітко описаний, можуть виникати складнощі у застосуванні інтегрованої моделі, її тестуванні, та інтерпретації результатів.

Залежно від характеру моделей та їхнього доменного покриття, можна виділити кілька типів співвідношень між Dom_1 та Dom_2:

Повне включення: $Dom_1 \subseteq Dom_2$ або навпаки. Такий варіант дозволяє використовувати модель з вужчою областю як частковий опис у межах ширшої моделі. Наприклад, модель аеродинаміки автомобіля може бути включена у загальну модель транспортного потоку.

Частковий перетин: $Dom_1 \cap Dom_2 \neq \emptyset$, але $Dom_1 \not\subseteq Dom_2$ та $Dom_2 \not\subseteq Dom_1$. Це найпоширеніший і водночас найпроблемніший випадок, який вимагає детального аналізу, адаптації моделей і, можливо, введення граничних умов, проміжних перетворень, або контакту з автором моделі.

Немає перетину: $Dom_1 \cap Dom_2 = \emptyset$. Спроба об’єднати моделі є беззмістовною або потребує кардинальної перебудови як мінімум однієї з них.

На практиці визначення $Dom_1 \cap Dom_2$ може включати:

- аналіз типів параметрів (чи є параметри ідентичними або конвертація можлива);
- співставлення одиниць вимірювання;
- перевірку граничних значень і допустимих діапазонів;
- оцінку сумісності припущень, закладених у моделі;
- нормалізацію даних перед об'єднанням.

Особливу роль відіграють семантичні перетворення: іноді параметри з однаковими назвами в різних моделях мають різні смислові навантаження, що унеможливлює їхнє пряме поєднання без уточнення семантики.

Після визначення й технічної перевірки $Dom_1 \cap Dom_2$ важливим етапом є перевірка інтегрованої моделі на цьому перетині. Необхідно впевнитися, що об’єднана модель не генерує конфліктних або взаємовиключних результатів, результати залишаються фізично та логічно інтерпретованими, чутливість моделі до зміни параметрів зберігається в межах допустимих похибок.

Чим більше моделей залучено до інтеграції, тим складніше забезпечити узгодженість їх областей визначень. У багатьох сучасних проєктах, таких як цифрові двійники або предиктивне моделювання складних технічних систем, потрібно інтегрувати десятки й сотні моделей. У таких випадках узгодження доменів перетворюється на окреме завдання, що вимагає автоматизації, застосування формальних методів перевірки та використання онтологій для семантичного зіставлення параметрів.

Інтеграція моделей для моделювання ширших предметних областей відкриває нові можливості для системного аналізу, але водночас висуває жорсткі вимоги до узгодженості доменів окремих компонентів. Перетин областей визначення $Dom_1 \cap Dom_2$ має бути чітко визначеним, перевіреним і формалізованим як одна з ключових умов валідності інтегрованої моделі. Ігнорування цієї вимоги призводить до некоректних результатів, підвищеного ризику помилок і як результат – втрати можливості інтерпретації. Тому контроль перетину доменів має розглядатися не як другорядне технічне питання, а як центральний елемент процесу інтеграції та міждисциплінарної координації.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Проведене дослідження дозволяє стверджувати, що область визначення, одиниця виміру та шкала параметра є невід’ємними складовими його формального опису. Ігнорування або недостатнє опрацювання цих аспектів призводить до помилок при побудові, повторному використанні та інтеграції моделей у ширшу предметну площину. Пропонується уніфікований підхід до представлення параметрів у вигляді кортежу

$$P = \langle N, U, D \rangle,$$

де N — назва параметру, U — одиниця виміру, D — область визначення параметру. Така формалізація дозволяє уникнути двозначностей і сприяє кращій інтеграції моделей у складних міждисциплінарних системах..

На основі структури параметра, пропонується уніфікований підхід до представлення моделі:

$$M = \langle f, \{P_i\}, IC, D, A \rangle,$$

де: f — функціональний опис моделі, $\{P_i\}$ — множина формалізованих параметрів відповідно попереднього визначення, IC — неявний контекст, D — детальний опис предметної площини, A — контакти автора моделі.

Такий підхід дозволяє забезпечити як контроль семантичної коректності моделі, так і відстежування її джерел та умов використання.

Результати дослідження підкреслюють важливість не лише математичних, але й контекстуальних, семантичних і технічних аспектів звуження області визначення, зокрема при інтеграції моделей, врахуванні шкал виміру та формалізації неявного контексту. Запропонована структура відкриває перспективи для подальшої автоматизації аналізу сумісності моделей, а також побудови систем на базі концептуально цілісного представлення моделі та її параметрів.

Аналіз областей визначення параметрів дозволяє швидко прийняти рішення, чи взагалі можливо використовувати обрані моделі разом. Чи є звужена область визначення релевантною предметній площини, що моделюється.

Проміжні результати даного дослідження було апробовано на профільних науково-практичних конференціях [18, 19], де основні положення щодо формалізації параметрів моделі, структурування області визначення, а також роль контексту в інтеграції моделей викликали зацікавлення серед фахівців у галузі математичного моделювання, програмної інженерії та інформаційних технологій. Отримані відгуки та дискусії в межах конференцій стали підґрунтям для подальшого вдосконалення формального опису моделі та підтвердили практичну значимість обраного підходу.

Література

1. Ahrens W., Demmel J., Nguyen H. D. Algorithms for Efficient Reproducible Floating Point Summation. ACM Transactions on Mathematical Software, Volume 46, Issue 3. Article No.: 22, Pages 1 – 49 https://doi.org/10.1145/3389360
2. Bandler R. Frogs into Princes: Neuro Linguistic Programming / R. Bandler, J. Grinder ; ed. by J. O. Stevens. – Real People Press, 1979. – 194 p.
3. Bazerman M., Luca M. The power of experiments: Decision-making in a data-driven world – Cambridge : MIT Press, 2020. – 232 c.
4. Bhise V. D. Decision-Making in Energy Systems. – Boca Raton : CRC Press, 2021. – 386 p.
5. Boccara N. Modeling Complex Systems. – New York : Springer, 2010. – 490 p.
6. Evans E. Domain-driven design: tackling complexity in the heart of software. – 1st ed. – Boston : Addison-Wesley Professional, 2003. – 560 p.
7. Galef J. The Scout Mindset: Why Some People See Things Clearly and Others Don't. – New York : Portfolio / Penguin, 2021. – 288 p.
8. IEEE Standard for Floating-Point Arithmetic. https://standards.ieee.org/ieee/754/6210
9. Ousterhout J. A philosophy of software design. – 1st ed. – Stanford : Yaknyam Press, 2018. – 190 p.
10. Pfanzagl J. Theorie der Messung. – Berlin : Springer-Verlag, 1971. – 248 p.
11. Pierce C. Types and programming languages. – Cambridge : MIT Press, 2002. – 648 p.
12. Sayama H. Introduction to the Modeling and Analysis of Complex Systems. – Binghamton : Open SUNY Textbooks, 2015. – 496 p.
13. Sinha S., & Lee Y. M. (2024). Challenges with developing and deploying AI models and applications in industrial systems. Discover Artificial Intelligence, 4(1), Article 55. https://doi.org/10.1007/s44163-024-00151-2
14. White R., Engelen G., Uljee I. Modeling Cities and Regions as Complex Systems: From Theory to Planning Applications. – Cambridge : The MIT Press, 2024. – 344 p.
15. Wikipedia contributors. IEEE 754. https://en.wikipedia.org/wiki/IEEE_754
16. Валькман Ю.Р., Гриценко В.И., Рыхальский А.Ю. Модельно-параметрическое пространство: теория и применение. Киев: Наук. думка, 2012 – 192 с.
17. Рихальський О.Ю. Методи підвищення коректності програмного забезпечення: сучасний функціонал компіляторів та інші підходи // VII Всеукраїнська науково-технічна конференція «Комп'ютерні технології: інновації, проблеми, рішення», 02–03 грудня 2024 р. – Житомир, Україна.

18. Рихальський О.Ю. Підвищення коректності програмного забезпечення методом звуження областей визначення // Всеукраїнська науково-технічна інтернет-конференція «Автоматизація, комп'ютерні та біомедичні технології», 26 березня 2025 р. – Дніпро, Україна.

References

1. Ahrens, W., Demmel, J., & Nguyen, H. D. (2020). Algorithms for efficient reproducible floating point summation. ACM Transactions on Mathematical Software, 46(3), Article 22, 1–49. https://doi.org/10.1145/3389360
2. Bandler, R., & Grinder, J. (1979). Frogs into princes: Neuro linguistic programming (J. O. Stevens, Ed.). Real People Press.
3. Bazerman, M., & Luca, M. (2020). The power of experiments: Decision-making in a data-driven world. MIT Press.
4. Bhise, V. D. (2021). Decision-making in energy systems. CRC Press.
5. Boccara, N. (2010). Modeling complex systems. Springer.
6. Evans, E. (2003). Domain-driven design: Tackling complexity in the heart of software (1st ed.). Addison-Wesley Professional.
7. Galef, J. (2021). The scout mindset: Why some people see things clearly and others don't. Portfolio / Penguin.
8. IEEE. (2019). IEEE Standard for Floating-Point Arithmetic. https://standards.ieee.org/ieee/754/6210
9. Ousterhout, J. (2018). A philosophy of software design (1st ed.). Yaknyam Press.
10. Pfanzagl, J. (1971). Theorie der Messung. Springer-Verlag.
11. Pierce, B. C. (2002). Types and programming languages. MIT Press.
12. Sayama, H. (2015). Introduction to the modeling and analysis of complex systems. Open SUNY Textbooks.
13. Sinha, S., & Lee, Y. M. (2024). Challenges with developing and deploying AI models and applications in industrial systems. Discover Artificial Intelligence, 4(1), Article 55. https://doi.org/10.1007/s44163-024-00151-2
14. White, R., Engelen, G., & Uljee, I. (2024). Modeling cities and regions as complex systems: From theory to planning applications. MIT Press.
15. Wikipedia contributors. (n.d.). IEEE 754. Wikipedia. https://en.wikipedia.org/wiki/IEEE_754
16. Valkman, Yu. R., Hritsenko, V.I., Rykhalskyy O. Yu. (2012). Modelno-parametrycheskoe prostranstvo: teoryia y prymenenye [Model-Parametric Space: Theory and Application]. Kyiv. Naukova Dumka.
17. Rykhalskyy, O. Y. (2024, December 2–3). Metody pidvyshchennia korektnosti prohramnoho zabezpechennia: suchasnyi funktsional kompiliatoriv ta inshi pidkhody [Methods for improving software correctness: Modern compiler func-tionality and other approaches]. Proceedings of the VII Vseukrainska naukovo-tekhnichna konferentsiia «Kompiuterni tekhnolohii: innovatsii, problemy, rishennia». Zhytomyr, Ukraine.
18. Rykhalskyy, O. Y. (2025, March 26). Pidvyshchennia korektnosti prohramnoho zabezpechennia metodom zvuzhennia oblastei vyznachennia [Improving software correctness by narrowing the domains]. Proceedings of the Vseukrainska naukovo-tekhnichna internet-konferentsiia «Avtomatyzatsiia, kompiuterni ta biomedychni tekhnolohii». Dnipro, Ukraine.