УДК 004.4’2:004.421.2:004.89
DOI https://doi.org/10.32782/2663-5941/2025.4.2/23

Комлева Н.О.
Національний університет «Одеська політехніка»

РОЗШИРЕННЯ БАЗИСУ UML ДЛЯ ПІДТРИМКИ НЕДОСКОНАЛИХ ТА ЗМІНЮВАНИХ ВИМОГ У ПРОЄКТУВАННІ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ

У статті розглянуто проблему формалізації недосконалих, змінюваних та ймовірнісних вимог у контексті проєктування інтелектуальних програмних систем. У реальних умовах такі системи функціонують у динамічному середовищі з високим ступенем невизначеності, що ускладнює застосування традиційних підходів до інженерії вимог. Стандартний базис мови UML (Unified Modeling Language), попри її широке застосування у сфері програмної інженерії, не забезпечує належної підтримки нечіткості, неповноти, конфліктності чи варіативності вимог. Це створює інформаційний розрив між фазою збирання вимог і етапом моделювання, оскільки значущі дані втрачаються або фіксуються поза межами моделей, що обмежує можливості трасування, аналізу та автоматизованої обробки.

У роботі запропоновано розширення стандартного UML-профілю, яке дозволяє зберігати семантику недосконалих вимог. Для цього вводяться спеціальні стереотипи, теговані значення (tagged values) та обмеження (constraints), які дозволяють маркувати елементи моделей із вказанням рівня довіри, джерела інформації, ймовірності реалізації та контекстуальних умов. Показано, як ці розширення можуть бути інтегровані до різних типів UML-діаграм (use case, class, activity, state, sequence) без зміни ядра UML, зберігаючи при цьому узгодженість із метамоделлю.

Практичну реалізацію підходу описано у вигляді спеціалізованого UML-профілю, який може бути збережений як у стандартному форматі XMI для CASE-засобів типу Enterprise Architect або MagicDraw, так і у вигляді JSON-структури для StarUML. У роботі представлено приклади типових стереотипів (<<fuzzyRequirement>>, <<incomplete>>, <<conflicting>>, <<uncertainState>> тощо), а також тегованих атрибутів, які підтримують розширену семантику вимог.

Для ілюстрації загального підходу побудовано узагальнювальну діаграму етапів узгодження та інтерпретації вимог для UML-проєктування, яка відображає логіку переходу від недосконалих вимог до формалізованих моделей. Запропонований підхід є гнучким і сумісним з UML 2.x, що дає змогу легко масштабувати та адаптувати його до потреб різних інструментальних платформ. Він також сприяє поліпшенню комунікації між аналітиками, розробниками та замовниками, зменшуючи ризики втрати інформації у процесі проєктування інтелектуальних систем. Отримані результати мають потенціал стати основою для розширення CASE-засобів і розробки нових методів підтримки нечітких і динамічних вимог у середовищах із підвищеною складністю.

Ключові слова: *інженерія вимог, UML-профіль, нечіткі вимоги, неповнота, ймовірнісні моделі, CASE-засоби, стереотипи UML, формалізація вимог, автоматизоване проєктування.*

Постановка проблеми. Сучасні інтелектуальні програмні системи функціонують у середовищах, що характеризуються високим рівнем динаміки, складності та невизначеності. Для таких систем ключовим етапом життєвого циклу є проєктування на основі вимог, що часто є неповними, суперечливими або змінюваними. Традиційні методи інженерії вимог, а також стандартні засоби моделювання, зокрема UML, передбачають роботу з формально узгодженими, чітко визначеними специфікаціями. Однак на практиці це припущення є обмежувальним, оскільки в реальних умовах розробники змушені працювати з «недосконалими» вимогами.

Стандартний UML-базис, що широко застосовується в проєктуванні програмного забезпечення, не передбачає явної підтримки недосконалих вимог. На практиці такі вимоги можуть бути нечіткими, неповними, суперечливими, ймовірнісними або змінюваними, що ускладнює їх формалізацію та інтеграцію в моделі. Проєктувальники змушені або штучно «доочищувати» вимоги перед моделюванням (що призводить до втрати важливої інформації), або зберігати допоміжні

коментарі поза межами моделей, що ускладнює автоматизовану обробку, трасування та подальшу верифікацію. Для забезпечення цілісного проєктування необхідне розширення UML-базису таким чином, щоб у ньому можна було безпосередньо відображати ступінь визначеності, джерело, довіру, контекстуальні умови та варіативність кожного елемента.

Нижче наведено класифікацію основних проблем, що притаманні недосконалим вимогам:

– неповнота – вимога не містить усіх необхідних деталей, таких як умови активації, результати виконання або взаємозв'язки з іншими вимогами; це унеможливлює побудову повного проєктного артефакту без додаткових припущень;

– конфліктність – різні вимоги можуть суперечити одна одній логічно, ресурсно або функціонально; такі суперечності виявляються під час аналізу сценаріїв, залежностей або пріоритетів;

– нечіткість та багатозначність – вимога містить формулювання, які допускають кілька тлумачень, наприклад: «високий рівень доступності», «швидке реагування» або «важливий користувач»;

– невизначеність – вимога містить умовну або ймовірнісну інформацію (наприклад, «у 80% випадків очікується…»), що ускладнює жорстку трансляцію в UML-моделі;

– змінюваність – вимоги можуть змінюватись із часом або в залежності від контексту, що накладає вимоги до їх гнучкого відображення та трасування;

– невимірюваність / неперевірюваність – вимога сформульована так, що її складно або неможливо перевірити у процесі тестування або валідації.

UML-нотація, зокрема її основні діаграми (use case, class, activity, state, sequence), передбачає повністю визначену, логічно узгоджену структуру. Сутності, зв'язки та атрибути мають бути чітко визначені, без неоднозначностей. Як результат, поточний UML-базис не дозволяє: фіксувати альтернативні варіанти інтерпретації вимог, позначати ступінь довіри чи джерело вимоги, підтримувати вірогідніснісні або умовні залежності між сценаріями, маркувати частково заповнені елементи моделі (наприклад, клас без типу атрибуту або з невизначеною поведінкою). Це створює бар'єр між фазами інженерії вимог і проєктування, оскільки важлива інформація про невизначеність або контекст губиться при трансляції в UML.

Аналіз останніх досліджень і публікацій. Сучасні інтелектуальні системи повинні враховувати неповноту, конфліктність, невизначеність та динамічність вимог, що ставить нові виклики перед методами проєктування. Значну увагу дослідники приділяють питанням формалізації таких вимог, їх узгодження, а також відображення у моделях програмного забезпечення.

Зокрема, у роботі [1, с. 134–152] представлено підхід до верифікації UML-моделей на основі нечіткої логіки та описової логіки, що дозволяє враховувати нечіткість у специфікаціях. Аналогічні ідеї розвиваються у [2, с. 1–8], де розглянуто підтримку інженерії вимог за допомогою механізмів нечіткої логіки. Комплементарний підхід запропоновано у [3, с. 317–322], де формалізовано нечітке подання вимог у процесах прийняття рішень. Автори [4, с. 207–224] акцентують увагу на усуненні абстрактного розриву між моделями та кодом за допомогою нечіткої логіки в тестуванні.

У публікаціях [5, с. 1–12; 6, с. 1–15] обґрунтовано доцільність використання графових моделей для відображення залежностей між вимогами та використання цілочислового програмування для їх відбору. Системний огляд підходів до представлення невизначеності в програмних моделях наведено у [7, с. 1183–1213], де розглянуто також класифікацію типів невизначеності. Проблема роботи з ймовірнісними характеристиками подій також висвітлюється у [8, с. 88026–88048].

Дослідження [9, с. 2313–2360] містить систематичний огляд виконання UML-моделей, що є важливим для створення інструментів з підтримкою нечітких або конфліктних вимог. У [10, с. 1–29] автори аналізують роль «впевненості у знаннях» у доменних моделях. Серед останніх публікацій слід також відзначити [11, с. 1–20], де представлено уніфіковану модель невизначеності у кіберфізичних системах.

В українському сегменті важливими є праці, де досліджується моделювання інтелектуальних діагностичних систем в умовах обмежених ресурсів і часткової невизначеності. Зокрема, в роботах [12, с. 152–156; 13, с. 99–108; 14, с. 97–100] висвітлено питання бізнес-орієнтованого інжинірингу вимог, оптимізації вибору функціоналу та обліку факторів невизначеності в процесі розробки.

Отже, попри наявність значної кількості робіт, що торкаються окремих аспектів нечіткості чи адаптації вимог, спостерігається брак комплексних підходів до розширення стандартних засобів моделювання, таких як UML, для відображення недосконалих і змінюваних вимог у динамічних середовищах. Саме це і визначає актуальність подальших досліджень у цьому напрямі.

Постановка завдання. Метою роботи є розширення базису UML з урахуванням специфіки недосконалих та змінюваних вимог, що виникають у процесі проєктування інтелектуальних систем, для забезпечення їх повноцінної формалізації, відображення невизначеності та підтримки подальшого автоматизованого моделювання.

Виклад основного матеріалу. Єдина структура, яку задає стандарт UML, передбачає однозначність, завершеність та внутрішню узгодженість усіх елементів моделі. Проте в реальних умовах інженерії вимог до інтелектуальних систем часто мають справу з неповними, багатозначними або конфліктними твердженнями. Такі вимоги не можуть бути прямо трансформовані у типові UML-діаграми, не втрачаючи при цьому критичну інформацію.

Недосконаліть вимог, зокрема їх нечіткість, неповнота, протиріччя та змінюваність, не можуть бути адекватно відображені стандартними засобами UML. Щоб зберігати цю критично важливу інформацію під час моделювання, доцільно впровадити розширення до UML-профілю. Такі розширення повинні дозволяти фіксувати джерело вимоги, ступінь довіри, ймовірність виконання, а також контекстуальні залежності. Найбільш зручним механізмом для цього є введення відповідних анотацій, що можуть бути прикріплені до елементів UML-діаграм, зокрема: власні властивості (tagged values) для збереження довіри, ймовірності, джерела; стереотипи для позначення специфічних типів вимог або їх статусу (наприклад, «<<fuzzyRequirement>>»); обмеження (constraints) для опису умов або альтернативних варіантів поведінки.

Запропонований підхід орієнтований не на зміну сутності UML як стандарту, а на розширення його семантики засобами, які вже передбачено в архітектурі UML 2.x – через створення спеціалізованого UML-профілю, який підтримуватиме ключові характеристики недосконалих вимог. Практичне застосування розширень можна проілюструвати на прикладі ключових типів діаграм, які найчастіше використовуються в процесі моделювання вимог.

Діаграма варіантів використання. Кожен варіант використання може бути розширений властивостями confidenceLevel, source, contextConditions, які дозволяють вказати ймовірність виконання, джерело вимоги (користувач, регламент, інженер тощо) та умови, за яких сценарій активується. Для відображення альтернативних сценаріїв можна застосовувати стереотипи <<variantScenario>> або <<exceptionalCase>>.

Діаграма класів. Атрибути класу можуть мати невизначений тип або значення, які позначаються спеціальним маркером (?, unknown, ToBeDetermined) і супроводжуються властивістю completeness = «partial». Стереотипи <<fuzzyClass>>, <<incompleteClass>> дають змогу явно вказати, що сутність моделі є недоопрацьованою. Відносини між класами можуть супроводжуватись умовними залежностями, позначеними через condition або probability.

Діаграма активностей. Дії з невизначеною логікою або умовами виконання позначаються стереотипами <<ambiguousAction>>, з додатковим описом у вигляді коментарів. Альтернативні шляхи, що залежать від ймовірнісних подій, можуть бути анотовані властивістю likelihood.

Діаграма станів. Стан або перехід, що активується лише за невизначених або умовних подій, позначається як <<probabilisticTransition>>. Можна ввести додаткову категорію «невизначений стан» (<<uncertainState>>), який є тимчасовим або залежить від зовнішнього контексту.

Діаграма послідовностей. Для повідомлень, які можуть бути умовними, передбачено властивість expectedProbability або optional = true. Стереотип <<uncertainMessage>> використовується для ідентифікації комунікацій, що є спірними або не завжди ініціюються.

Ці зміни дозволяють будувати моделі, які зберігають зв'язок із реальними вимогами без необхідності штучного уточнення або спрощення. Відповідно, така модель буде краще придатною для автоматизованої обробки, перевірки варіантів і подальшої еволюції вимог.

У таблиці 1 наведено основні стереотипи UML-профілю, призначені для позначення властивостей вимог у статичних моделях. Для динамічних діаграм відповідні стереотипи подано в таблиці 2. Таблиця 3 ілюструє приклади тегованих значень, які дозволяють деталізувати рівень довіри, джерело вимоги, контекстні обмеження, варіативність або неповноту. На рис. 1 представлено етапи узгодження та інтерпретації вимог для UML-проєктування, які ілюструють перехід від вимог до формалізованої UML-моделі, придатної для подальшого автоматизованого проєктування.

Процес починається з виявлення недосконалих вимог, після чого здійснюється їх класифікація за типами проблем. На основі цієї класифікації формується відповідне розширення базових елементів UML, яке в подальшому інтегрується у метамо-

Таблиця 1

Стереотипи UML-профілю для класів та акторів

Стереотип	Значення	Приклад використання
<<fuzzyRequirement>>	Вимога з нечіткою формалізацією	«Система повинна бути високопродуктивною»
<<incomplete>>	Вимога не містить повного опису	Відсутній тип атрибуту або метод реалізації
<<conflicting>>	Вимога суперечить іншій	«Повідомлення має бути одночасно публічним і приватним»
<<alternative>>	Альтернативний сценарій або залежність	Альтернативна версія бізнес-процесу
<<contextDependent>>	Реалізація залежить від середовища	Поведінка змінюється залежно від місця/часу

Таблиця 2

Стереотипи UML-профілю для динамічних діаграм

Стереотип	Значення	Приклад
<<uncertainState>>	Стан, який активується не завжди	«Очікування підтвердження»
<<probabilisticMessage>>	Повідомлення з ймовірністю доставки	«Дані отримані з імовірністю 90%»
<<vagueTransition>>	Переходи з умовами, які нечітко формалізовані	«Якщо користувач задоволений»

Таблиця 3

Основні теговані значення, що доповнюють стереотипи

Теговане значення	Тип	Опис
confidence	float [0.1]	Рівень довіри до вимоги або її частини
source	string	Джерело формулювання або надходження вимоги
likelihood	float [0.1]	Ймовірність виконання або реалізації вимоги
context	string	Контекстні умови, які впливають на інтерпретацію
timestamp	datetime	Дата останньої модифікації або валідації вимоги
conflictGroup	string	Ідентифікатор групи суперечливих вимог
resolutionHint	string	Пропозиція щодо розв'язання або уточнення вимоги

дель, що описує структуру доповнених елементів. Розроблена метамодель впроваджується в інструментальне середовище моделювання (наприклад, StarUML), після чого ці розширення можуть бути використані під час проєктування інтелектуальних систем.

На метамодельному рівні кожен стереотип є підкласом відповідного UML-елемента (Class, UseCase, Message, State) і має зв'язок з відповідними тегованими значеннями, що реалізуються як атрибути з визначеним типом (float, string, datetime). Таким чином, модель збагачується додатковими семантичними шарами без зміни основної структури UML.

Для забезпечення обміну UML-моделями з розширеними властивостями застосовуються різні формати збереження, залежно від обраного CASE-засобу. Такі інструменти, як Enterprise Architect, MagicDraw або Papyrus, використовують стандарт XMI (XML Metadata Interchange) для серіалізації моделей, що дозволяє зберігати додаткові анотації (стереотипи, теговані значення, коментарі) згідно зі специфікацією UML 2.5. Натомість інші інструменти, такі як StarUML, працюють з власним JSON-поданням моделі, у якому метамодель та профілі описуються також у форматі JSON. Завдяки цьому користувачі можуть застосовувати власні розширення для позначення невизначеностей або неповноти без втрати зручності використання інструменту.

Таким чином, запропоноване розширення UML може бути реалізоване як UML-профіль у форматі XMI для інструментів типу Enterprise Architect, MagicDraw, Papyrus тощо чи JSON-розширення для StarUML та подібних середовищ. Це забезпечує підтримку збереження семантики нечітких, змінюваних та неповних вимог незалежно від обраного інструменту моделювання.

Висновки. У статті було обґрунтовано необхідність розширення стандартного базису UML для підтримки недосконалих, нечітких, суперечливих та змінюваних вимог, які характерні для інтелектуальних систем, що проєктуються в умовах високої динаміки та невизначеності. Аналіз

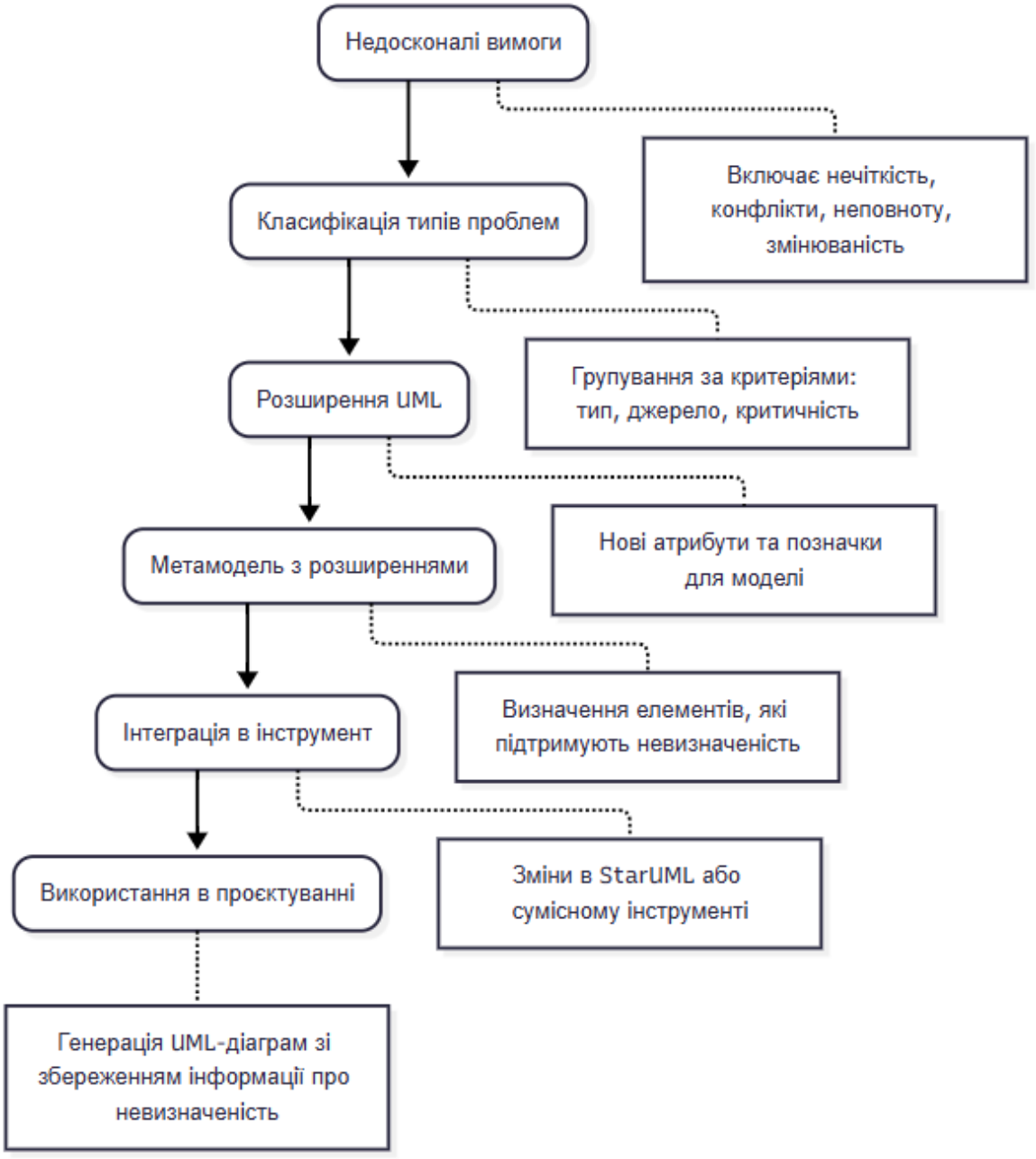

Рис. 1. Етапи узгодження та інтерпретації вимог для UML-проєктування

існуючих підходів продемонстрував обмеженість традиційних засобів моделювання у фіксації таких властивостей вимог, як ступінь довіри, ймовірність реалізації, контекстуальна залежність або неповнота опису.

Запропоновано формальний підхід до розширення UML за допомогою спеціалізованих стереотипів і тегованих значень, що дозволяє явно відображати властивості невизначеності в моделях. Побудовано загальну структуру процесу розширення моделі – від класифікації вимог до інтеграції з CASE-засобами та практичного застосування у проєктуванні.

Описані рішення можуть бути реалізовані у вигляді UML-профілю для популярних інструментів, таких як StarUML або Enterprise Architect, без втрати сумісності з існуючими моделями. Таке розширення забезпечує більш тісну інтеграцію між фазами інженерії вимог і проєктування, підвищуючи точність, адаптивність і пояснюваність розроблюваних систем.

Подальші дослідження будуть присвячені формалізації переходу від вимог із зазначеними характеристиками до відповідних UML-артефактів, а також реалізації прототипу плагіна для CASE-середовищ.

Список літератури:

1. Zhang F., Cheng J. Verification of fuzzy UML models with fuzzy Description Logic // *Applied Soft Computing*. 2018. Vol. 73. P. 134–152. DOI:10.1016/j.asoc.2018.08.025

2. Egesoy A., Güzel A. Fuzzy Logic Support for Requirements Engineering // *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)*. 2021. Vol. 9, No. 2. P. 1–8.

3. Larbi A., Malki M. Uncertain Decision-Making Requirements Formalizing with Complement Fuzzy UML Model // *Procedia Computer Science*. 2022. Vol. 198. P. 317–322. DOI:10.1016/j.procs.2021.12.247

4. Cazzola W., Ghosh S., Al-Refai M. Bridging the model-to-code abstraction gap with fuzzy logic in model-based regression test selection (FLiRTS 2) // *Software & Systems Modeling*. 2021. Vol. 21, Iss. 1. P. 207–224. DOI:10.1007/s10270-021-00899-6

5. Mougouei D., Powers D.M.W. Dependency-Aware Software Requirements Selection using Fuzzy Graphs and Integer Programming // *arXiv*. 2020. Mar 11. P. 1–12.

6. Mougouei D., Powers D.M.W. Modeling and Selection of Interdependent Software Requirements using Fuzzy Graphs // *arXiv*. 2020. Mar 3. P. 1–15.

7. Troya J., Moreno N., Bertoa M. F., et al. Uncertainty representation in software models: a survey // Software and Systems Modeling. 2021. Vol. 20. P. 1183–1213. DOI: 10.1007/s10270-020-00842-1.

8. Moreno N., Bertoa M. F., Burgueño L., Vallecillo A. Managing measurement and occurrence uncertainty in complex event processing systems // IEEE Access. 2019. Vol. 7. P. 88026–88048. DOI: 10.1109/ACCESS.2019.2923953.

9. Ciccozzi F., Malavolta I., Selic B. Execution of UML models: a systematic review of research and practice // *Software and Systems Modeling*. – 2019. – Vol. 18. – P. 2313–2360. – DOI: 10.1007/s10270-018-0675-4.

10. Burgueño L., Muñoz P., Clarisó R., Gérard S., Vallecillo A. Dealing with belief uncertainty in domain models // *ACM Transactions on Software Engineering and Methodology*. 2022. Vol. 31, Iss. 2. P. 1–29.

11. Mäkelburg J., Perez-Palacin D., Mirandola R., Acosta M. Surveying Uncertainty Representation: A Unified Model for Cyber-Physical Systems // *arXiv*. 2025. Mar 3. P. 1–20.

12. Комлева Н. О. Інжиніринг бізнес-вимог при розробці складних діагностичних систем // *Вчені записки ТНУ імені В. І. Вернадського. Серія: технічні науки*. Київ, 2024. Т. 35 (74), № 1. С. 152–156. DOI: 10.32782/2663-5941/2024.1.1/23.

13. Антощук С. Г., Комлева Н. О. Оптимізаційне моделювання інтелектуальних систем діагностування // *Вісник Херсонського національного технічного університету*. 2024. № 2. С. 99–108. DOI: 10.35546/kntu2078-4481.2024.2.14.

14. Комлева Н. О. Особливості математичного моделювання інтелектуальних систем діагностування і прийняття рішень в умовах невизначеності // *Розвиток наукової думки постіндустріального суспільства: сучасний дискурс* : зб. наук. пр. з матеріалами VII Міжнар. наук. конф., м. Одеса, 25 квіт. 2025 р. / Міжнар. центр наук. дослідж. Вінниця : ТОВ «УКРЛОГОСГруп», 2025. С. 97–100. DOI: 10.62731/mcnd-25.04.2025.001.

Komleva N.O. EXTENDING THE UML FOUNDATION TO SUPPORT IMPERFECT AND EVOLVING REQUIREMENTS IN THE DESIGN OF INTELLIGENT SYSTEMS

The article explores the problem of formalizing imperfect, evolving, and probabilistic requirements in the context of designing intelligent software systems. In real-world conditions, such systems operate in dynamic environments with a high degree of uncertainty, which complicates the application of traditional requirements engineering approaches. The standard UML (Unified Modeling Language) foundation, despite its widespread use in software engineering, does not provide adequate support for vagueness, incompleteness, conflicts, or variability of requirements. This creates an informational gap between the requirements gathering phase and the modeling stage, as significant data is either lost or recorded outside the models, limiting traceability, analysis, and automated processing capabilities.

The paper proposes an extension to the standard UML profile that enables the preservation of the semantics of imperfect requirements. This is achieved by introducing specific stereotypes, tagged values, and constraints, which allow model elements to be marked with indications of trust level, source of information, probability of realization, and contextual conditions. It is shown how these extensions can be integrated into various types of UML diagrams (use case, class, activity, state, sequence) without altering the UML core, while maintaining consistency with the metamodel.

The practical implementation of the approach is described in the form of a specialized UML profile, which can be stored either in the standard XMI format for CASE tools such as Enterprise Architect or MagicDraw, or as a JSON structure for StarUML. The paper presents examples of typical stereotypes (<<fuzzyRequirement>>,

<<incomplete>>, <<conflicting>>, <<uncertainState>>, etc.) as well as tagged attributes that support the extended semantics of requirements.

To illustrate the overall approach, a general diagram of the stages of requirements reconciliation and interpretation for UML design has been constructed, which reflects the logic of transitioning from imperfect requirements to formalized models. The proposed approach is flexible and compatible with UML 2.x, enabling easy scalability and adaptation to the needs of various modeling platforms. It also enhances communication between analysts, developers, and stakeholders, reducing the risk of information loss during the design of intelligent systems. The results obtained have the potential to form a basis for extending CASE tools and developing new methods to support vague and dynamic requirements in high-complexity environments.

Key words: *requirements engineering, UML profile, fuzzy requirements, incompleteness, probabilistic models, CASE tools, UML stereotypes, requirements formalization, automated design.*

Дата надходження статті: 20.07.2025
Дата прийняття статті: 31.07.2025
Опубліковано: 27.10.2025