УДК 004.415.5 DOI https://doi.org/10.35546/kntu2078-4481.2025.3.2.30

Н. О. КОМЛЕВА
кандидат технічних наук,
завідувач кафедри інженерії програмного забезпечення
Національний університет «Одеська політехніка»
ORCID: 0000-0001-9627-8530

М. І. НІКІТЧЕНКО
аспірант кафедри інженерії програмного забезпечення
Національний університет «Одеська політехніка»
ORCID: 0009-0007-9560-7057

МЕТОД ДВОСТОРОННЬОЇ СИНХРОНІЗАЦІЇ UML-МОДЕЛІ З ДВОМА ПОДАННЯМИ НА ОСНОВІ ІНКРЕМЕНТАЛЬНИХ ОНОВЛЕНЬ

У статті представлено формальний метод двосторонньої синхронізації UML-моделі з двома поданнями, які зберігається в окремих форматах: структурне подання реалізується у вигляді об'єктної нотації JavaScript, а поведінкове – у форматі міжплатформного обміну XMI із застосуванням спеціалізованого профілю. Основною метою дослідження є забезпечення узгодженості обох подань у процесі еволюції моделі на основі валідації інваріантів та інкрементального оновлення. У роботі описано математичну формалізацію множин структурних та поведінкових елементів, а також відповідності між ними, що реалізується за допомогою трасувальних зв'язків. Розроблено механізм відстеження змін у поведінковій частині моделі з побудовою графа залежностей, який дозволяє визначити підмножину елементів, що потребують повторної перевірки інваріантів. Запропонований метод верифікації моделі ґрунтується на інваріантах консистентності та забезпечує формальні гарантії семантичної відповідності. У рамках дослідження було проведено експериментальне оцінювання запропонованого методу на наборі тестових UML-моделей, що моделюють типові бізнес-процеси у сфері електронної комерції, зокрема обробку замовлень, управління кошиком, авторизацію користувачів і виконання транзакцій. До складу моделей входили різноманітні типи поведінкових діаграм, зокрема діаграми станів для опису життєвих циклів замовлень, діаграми діяльності для моделювання логіки послідовного виконання дій, а також діаграми послідовностей для представлення сценаріїв взаємодії між об'єктами системи. Завдяки використанню моделей з реалістичними сценаріями взаємодії було перевірено здатність методу до виявлення і локалізації порушень консистентності, що виникають внаслідок часткових змін як у структурному, так і в поведінковому поданні. Результати показали істотне зменшення кількості перевірок у порівнянні з повною валідацією. Запропонований підхід є сумісним з існуючими інструментами UML-моделювання та може бути інтегрований у процеси CI/CD та еволюційного аналізу архітектури.

Ключові слова: *UML, подання, синхронізація моделей, інкрементальне оновлення, консистентність, граф залежностей.*

N. O. KOMLEVA
Candidate of Technical Sciences,
Head of the Department of Software Engineering
Odesa Polytechnic National University
ORCID: 0000-0001-9627-8530

M. I. NIKITCHENKO
Postgraduate Student at the Department of Software Engineering
Odesa Polytechnic National University
ORCID: 0009-0007-9560-7057

METHOD FOR BILATERAL SYNCHRONIZATION OF A UML MODEL WITH TWO REPRESENTATIONS BASED ON INCREMENTAL UPDATES

The article presents a formal method for bilateral synchronization of a UML model with two views stored in separate formats: the structural view is implemented as JavaScript object notation, and the behavioral view is implemented in the XMI cross-platform exchange format using a specialized profile. The main goal of the research is to ensure the consistency of both views in the process of model evolution based on invariant validation and incremental updating. The paper

describes the mathematical formalization of sets of structural and behavioral elements, as well as the correspondence between them, which is implemented using traceable links. A mechanism for tracking changes in the behavioral part of the model has been developed with the construction of a dependency graph, which allows determining a subset of elements that require re-checking of invariants. The proposed model verification method is based on consistency invariants and provides formal guarantees of semantic correspondence. As part of the study, an experimental evaluation of the proposed method was carried out on a set of test UML models that simulate typical business processes in the field of e-commerce, in particular order processing, shopping cart management, user authorization, and transaction execution. The models included various types of behavioral diagrams, including state diagrams for describing order lifecycles, activity diagrams for modeling the logic of sequential execution of actions, and sequence diagrams for representing interaction scenarios between system objects. By using models with realistic interaction scenarios, the method's ability to detect and localize consistency violations resulting from partial changes in both structural and behavioral view was tested. The results showed a significant reduction in the number of checks compared to full validation. The proposed approach is compatible with existing UML modeling tools and can be integrated into CI/CD and evolutionary architecture analysis processes.

Key words: *UML, presentation, model synchronization, incremental update, consistency, dependency graph.*

Постановка проблеми

У сучасних середовищах розроблення програмного забезпечення зростає попит на гнучке та адаптивне використання моделей, що поєднують різні види подань. Одним із прикладів такого підходу є комбіноване представлення моделей уніфікованого моделювання UML, де структурні аспекти зберігаються у форматі об'єктної нотації JavaScript (JSON), а поведінкові – у форматі обміну моделями XMI. Така подільність забезпечує сумісність з різними інструментами, підтримує кросплатформну обробку, а також спрощує інкрементальне оновлення окремих фрагментів моделі без необхідності повної реконструкції.

Разом із перевагами, подібне поєднання подань породжує низку складних викликів, зокрема щодо підтримання узгодженості між ними, забезпечення формальної верифікації їхньої відповідності та збереження цілісності моделі при її еволюції. Попередні дослідження були зосереджені на валідаційних інваріантах для окремих подань, а також на алгоритмах контролю консистентності між структурною та поведінковою частинами. Проте недостатньо розробленим залишається питання ефективної двосторонньої синхронізації, яка б дозволяла виявляти і застосовувати локальні зміни до іншого подання, не порушуючи цілісності моделі.

Метою цієї роботи є формальне обґрунтування та розроблення методу двосторонньої синхронізації UML-моделі, що базується на інкрементальному оновленні та використовує попередньо верифіковану структуру як передумову коректної роботи. Основу підходу становить теорема інкрементальної консистентності, яка дозволяє обмежити обсяг повторної перевірки при часткових змінах моделі.

Аналіз останніх досліджень і публікацій

Проблематика синхронізації моделей у контексті інженерії програмного забезпечення є предметом активних досліджень [1]. Особливої актуальності це набуває в рамках багатоподаневих підходів, де одна й та сама система описується за допомогою набору гетерогенних, але семантично пов'язаних моделей [2].

Фундаментальною основою для вирішення цього завдання є концепція двосторонніх трансформацій, яка передбачає не лише перетворення моделі з одного подання в інше, а й зворотне поширення змін [3]. Класичним підходом до реалізації є використання потрійних граматик графів, які дозволяють формально описати правила відповідності між двома моделями та автоматично генерувати механізми їхньої синхронізації [4, 5]. Проте такий підхід часто має високу обчислювальну складність і може бути неоптимальними для інкрементальних оновлень, оскільки потребує перегляду значної частини моделі навіть при локальних змінах. Для підвищення ефективності процесу валідації активно розвиваються методи інкрементальної перевірки консистентності [6, 7], що дозволяє суттєво скоротити час на валідацію. Але більшість досліджень зосереджена на узгодженості між структурними діаграмами, тоді як питання консистентності між структурними та поведінковими аспектами залишається менш дослідженим [8].

В рамках попередніх праць авторів [9–11] було сформульовано набір валідаційних правил для обох типів подання, а також розроблено механізм інкрементального контролю консистентності при зміні однієї з частин моделі. Водночас зазначені підходи не реалізовують повноцінну синхронізацію в обох напрямах та не формалізують умови коректності таких перетворень. Також у суміжних дослідженнях розглядається підтримка комбінації форматів у середовищах моделювання, однак більшість із них не надає відкритих механізмів для трасування між елементами різних подань [12], що обмежує можливості автоматизованої перевірки відповідності або адаптації одного подання до змін в іншому.

Формулювання мети дослідження

Метою роботи є створення методу двосторонньої синхронізації UML-моделі з двома поданнями. Він має поєднувати в собі підтримку гетерогенних форматів подання моделі, використання верифікованої інформаційної бази, інкрементальність змін та двосторонню синхронізацію з відновленням локальних залежностей.

Викладення основного матеріалу дослідження

У межах даного дослідження під UML-моделлю з двома поданнями розуміється модель, яка містить два взаємопов'язані подання: структурне та поведінкове. Це можна описати наступним виразом:

$$M = (M_S, M_B, \mu),$$

де M_S – це множина елементів структурного подання моделі, що представлені у форматі JSON, M_B – це множина елементів поведінкового подання моделі, що представлені у форматі XMI згідно зі специфікацією UML, $\mu \subseteq M_S \times M_B$ – відношення відповідності між множинами структурних елементів M_S та поведінкових M_B елементів моделі.

Запропоноване відображення μ пов'язує кожен поведінковий елемент із одним або кількома структурними елементами, від яких він залежить. Воно реалізується через спеціалізовані атрибути трасування (наприклад, jsonRef, triggerSource), що явно вказують на відповідний ідентифікатор у JSON-поданні.

Оскільки модель може змінюватися з обох боків, визначаються множини локальних змін:

– $\Delta M_S \subseteq M_S$ – множина структурних елементів, які були додані, змінені або видалені;
– $\Delta M_B \subseteq M_B$ – множина змінених поведінкових елементів відповідно.

Для подальшої локалізації валідаційних перевірок і побудови синхронізації використовується орієнтований граф залежностей $G = (V, E)$, де множина вузлів V містить усі елементи M_B, а множина ребер E фіксує залежності між ними. У графі виділяють три класи залежностей:

– структурні – описують синтаксичні зв'язки між елементами діаграм;
– семантичні – відображають логіку виконання поведінки;
– трасувальні – забезпечують відповідність із структурними елементами M_S.

Граф G дозволяє визначити множину елементів, які слід повторно перевірити після внесення змін. Це суттєво скорочує обсяг валідаційних операцій і дозволяє побудувати інкрементальний механізм синхронізації.

Далі, перш ніж переходити до побудови механізмів синхронізації між структурним і поведінковим поданнями, необхідно забезпечити верифікацію такої UML-моделі як передумову її коректного функціонування. У цьому контексті під верифікацією розуміється підтвердження відповідності моделі заздалегідь визначеним валідаційним правилам, що формалізують допустимі структури та поведінкові патерни відповідно до семантики UML.

Структурна частина моделі представлена у форматі JSON відповідно до запропонованої метамоделі. Валідаційні правила для цієї частини були раніше систематизовані у [1], зокрема обов'язковість імен для сутностей, коректність типізації атрибутів і зв'язків, єдність ідентифікаторів у межах одного простору імен, відповідність класів, атрибутів і операцій їх метатипам. Ці правила інтерпретуються як множина предикатів $\sigma_1, \sigma_2, \ldots, \sigma_n$, кожен з яких перевіряється над множиною елементів M_S.

Формальна модель валідації виглядає наступним чином:

$$\forall m \in M_S : i = 1n\sigma_i(m) = \text{true}.$$

Поведінкова частина представлена у форматі XMI, який реалізує специфікацію UML 2.5.x з розширенням у вигляді спеціального профілю. Валідаційні інваріанти для поведінкових діаграм наведені в [2] і охоплюють діаграми станів (перевірка умов переходу, унікальності початкового стану, наявності переходів), діаграми діяльності (коректність потоків керування, відповідність типів вхідних/вихідних даних) та діаграми послідовностей (зв'язність між повідомленнями, послідовність активацій, узгодженість із Lifeline).

Нехай $\psi_1, \psi_2, \ldots, \psi_k$ – предикати валідації поведінки над множиною M_B. Тоді модель задовольняє поведінкову верифікацію, якщо:

$$\forall b \in M_B : j = 1k\psi_J(b) = \text{true}.$$

Ключовою вимогою використання цих двох подань є семантична відповідність між структурою та поведінкою. Вона реалізується через відображення μ, яке визначає, які елементи M_B мають бути пов'язані з відповідними елементами M_S. Правила відповідності включають:

– наявність посилань jsonRef, triggerSource в елементах M_B;
– відповідність типів дій, викликів або повідомлень до відповідних методів або сигналів у структурі;
– відповідність параметрів і типів даних між InputPin та структурними атрибутами або операціями.

Тому, модель вважається верифікованою, якщо M_S задовольняє всі структурні інваріанти, M_B задовольняє всі поведінкові інваріанти, а також якщо всі елементи M_B, що мають трасувальні посилання, відображаються на коректні елементи M_S і виконуються узгоджувальні правила.

Ця верифікована база виступає точкою відліку для всіх подальших інкрементальних оновлень і є необхідною умовою запуску двосторонньої синхронізації, яка не повинна порушувати верифікованість моделі після застосування змін.

Далі, опираючись на вищевказаний формальний опис UML-моделі, необхідно сформувати теорему інкрементальної консистентності такої UML-моделі з двома поданнями.

Теорема.

Нехай існує наступна UML-модель:

$$M = (M_S, M_B, \mu),$$

де M_S – структурне подання моделі (у форматі JSON), M_B – поведінкове подання моделі (у форматі XMI), $\mu \subseteq M_S \times M_B$ – відношення відповідності між множиною структурних M_S та поведінкових M_B елементів моделі.

Також нехай:

Σ – сукупність формальних обмежень консистентності у вигляді набору OCL-інваріантів, які визначають допустимість моделей;

$\Delta \subseteq M_S \cup M_B$ – множина елементів, що зазнали змін внаслідок інкрементального редагування;

Affected(Δ, μ) $\subseteq M_S \cup M_B$ – замикання змінених елементів через відношення відповідності μ, тобто множина всіх елементів, для яких існує зв'язок із елементами з Δ або суміжність за структурними/поведінковими відношеннями.

Тоді, якщо після зміни елементів із Δ усі інваріанти з Σ виконуються для кожного елемента з множини Affected(Δ,μ), то модель M загалом також задовольняє всі інваріанти з Σ, тобто

$$(\forall e \in \mathit{Affected}(\Delta, \mu) : M \vDash \sum(e)) \Rightarrow M \vDash \sum.$$

Доведення від супротивного:

Припустимо протилежне твердженню теореми: нехай після змін у моделі виконується

$$\forall e \in \mathit{Affected}(\Delta, \mu) : M \vDash \sum(e),$$

але вся модель неконсистентна, тобто:

$$\exists \sigma \in \sum : M \nvDash \sigma.$$

За визначенням, кожен інваріант σ формується локально, тобто його істинність залежить лише від обмеженої множини елементів:

$$\mathrm{Context}(\sigma) \subseteq M_S \cup M_B.$$

З того, що $M \nvDash \sigma$, випливає, що хоча б один з елементів контексту Context(σ) порушує вимогу цього інваріанту.

Але якщо інваріант σ був порушений, і цей інваріант залежить від деякого елемента e, тоді або $e \in \Delta$, тобто був змінений безпосередньо, або $e \in \mathit{Affected}(\Delta, \mu)$, тобто зміни торкнулися суміжного елемента.

Це суперечить нашому припущенню, що всі інваріанти виконуються на $\mathit{Affected}(\Delta, \mu)$.

Отже, припущення про те, що $M \nvDash \sum$ при $M \nvDash \sum(e)$ $\forall e \in \mathit{Affected}(\Delta, \mu)$, є хибним.

В результаті, ця теорема дозволяє обмежити обчислювальну перевірку консистентності лише підмножиною $\mathit{Affected}(\Delta, \mu)$, що істотно знижує складність верифікації в умовах інкрементального редагування.

Запропонований метод двосторонньої синхронізації забезпечує підтримку узгодженості між структурним поданням моделі та поведінковим поданням шляхом виконання інкрементальних оновлень. Передбачається, що початкова модель вже верифікована, тобто задовольняє всі формальні інваріанти відповідно до встановлених валідаційних правил для обох подань. Зміни в одній частині моделі автоматично ініціюють відповідне пропагування (поширення) змін до іншої частини моделі, забезпечуючи логічну цілісність без повної перебудови моделі.

Завдання полягає в оновленні пов'язаного подання та перевірці локальних інваріантів для збереження консистентності моделі M. При цьому необхідно досягти актуалізацію відображення μ, перевірку лише зачеплених елементів (інкрементально) та збереження семантичної відповідності між структурою і поведінкою.

Метод передбачає чотири основні кроки.

Крок 1. Локалізація змін. Визначається об'єднана множина змінених елементів:

$$\Delta = \Delta M_S \cup \Delta M_B.$$

Будується множина зачеплених елементів:

$$\mathit{Affected} = x \in \Delta \mathit{Affected}(x),$$

що визначається за допомогою графа залежностей між елементами моделі.

Крок 2. Пропагування змін. Для кожного зміненого елемента виконується спрямоване оновлення пов'язаних об'єктів:

– якщо зміна відбулася в M_S, оновлюються або створюються відповідні елементи в M_B (наприклад, клас → Lifeline, атрибут → InputPin);

– якщо зміна в M_B, виконується оновлення відповідних елементів M_S (наприклад, Transition → булевий атрибут, Message → операція або сигнал);

– в обох випадках актуалізуються трасувальні зв'язки μ.

Крок 3. Локальна верифікація. На основі теореми інкрементальної консистентності перевіряються лише інваріанти, які залежать від елементів із множини Affected. Якщо всі локальні перевірки успішні, модель вважається узгодженою. У протилежному випадку зміна відхиляється або маркується як така, що потребує доопрацювання.

Крок 4. Оновлення відповідностей. Зміни у відношенні μ включають додавання пар для новостворених елементів та вилучення або редагування пар після видалення або зміни елементів.

Запропонований метод володіє кількома ключовими властивостями, що забезпечують його ефективність та надійність. По-перше, він є *інкрементальним*, оскільки перевірці підлягає лише та частина моделі, яку зачепили локальні зміни, що значно зменшує обчислювальну складність. По-друге, гарантується його *завершуваність*, тобто метод завжди зупиняється, оскільки він не створює рекурсивного розширення множини змін. Крім того, для *уникнення циклів* нескінченної синхронізації застосовуються спеціальні маркери походження змін (Push або Pull). Нарешті, метод підтримує *семантичну узгодженість*: поведінкові елементи, створені внаслідок структурних змін, зберігають повну відповідність до типів, параметрів і залежностей, визначених у структурному поданні M_S.

Для демонстрації ефективності інкрементального методу синхронізації було проведено серію тестів на вручну побудованих UML-моделях, які реалізують типові сценарії з прикладної галузі систем управління замовленнями в електронній комерції. Всі ці моделі відображають логіку обробки замовлень, перевірки клієнтів, обліку складу й комунікації з зовнішніми сервісами. При цьому варто враховувати, що хоч діаграми діяльності, станів і послідовностей належать до поведінкових засобів моделювання, вони прямо посилаються на елементи структури через спеціальні механізми трасування, такі як jsonRef, TriggerSource, PayloadType (табл. 1), завдяки чому забезпечується тісна семантична взаємодія між двома поданнями.

Таблиця 1

Характеристики тестових моделей

Назва моделі	Елементи M_S	Елементи M_B	Типи діаграм M_B	Сценарій	Механізми трасування
Model A	12 класів, 18 атрибутів	13 елементів	Діаграма станів	Життєвий цикл замовлення	jsonRef для станів, TriggerSource для подій
Model B	9 класів, 12 операцій	17 елементів	Діаграма діяльності	Авторизація клієнта	jsonRef, PayloadType на InputPin
Model C	7 класів, 10 зв'язків	15 елементів	Діаграма послідовностей	Виклик платіжного сервісу	TriggerSource для повідомлень, jsonRef на lifeline

Трасувальні зв'язки у кожному випадку встановлювались вручну з урахуванням семантики сценарію та структурної специфікації.

На першому етапі для кожної моделі було визначено повний набір застосовних інваріантів, що включав структурні (унікальність ID, валідність типів у JSON), поведінкові (досяжність станів, коректність потоків) та міжподаннєві правила консистентності (SignatureMatches, AnchorExists). Повна перевірка цього набору слугувала базовим показником. На другому етапі в кожну модель вносилася атомарна локальна зміна (Δ), що імітувала типові дії розробника, наприклад, редагування атрибута в M_S або додавання переходу в M_B. Після цього запускався інкрементальний алгоритм, який за допомогою графа залежностей визначав мінімальну підмножину зачеплених елементів *Affected*(Δ) (табл. 2). Перевірці підлягали лише інваріанти, що стосувалися цієї підмножини.

Таблиця 2

Порівняння повної та інкрементальної перевірки інваріантів

Модель	Тип зміни	Інваріантів загалом	Перевірено (повністю)	Перевірено (інкрементально)
A	M_B	7	7	3
B	M_S	9	9	4
C	$M_B + M_S$	11	11	5

Інкрементальний підхід у середньому дозволив зменшити кількість перевірених інваріантів на 56 %, що прямо зменшує навантаження при ручному аналізі та значно пришвидшує зворотний зв'язок у разі розширення чи зміни моделі.

Теоретична складність перевірки підтвердилась емпірично:

$$T(n) = O(k \cdot d),$$

де $k \ll n$.

Це дозволяє зберігати масштабованість навіть для більших моделей. Застосування графа залежностей виявилось корисним навіть у ручному режимі: воно дозволяє фокусуватися лише на критично важливих інваріантах, пов'язаних зі змінами.

Висновки

У роботі представлено метод двосторонньої синхронізації UML-моделі з дома поданнями, де структурна частина описується у форматі JSON, а поведінкова – у форматі XMI з використанням спеціалізованого UML-профілю. Основна увага зосереджена на забезпеченні узгодженості між цими поданнями в умовах локальних змін моделі. Запропонований підхід базується на формалізованій відповідності між структурними та поведінковими елементами, підтримує інкрементальне оновлення лише змінених фрагментів та гарантує збереження консистентності за допомогою перевірки інваріантів.

Порівняно з традиційними методами повної перевірки, інкрементальна стратегія дозволяє істотно знизити обчислювальні витрати. Результати експериментальної оцінки на прикладах із предметної області електронної комерції продемонстрували, що кількість інваріантів, які необхідно перевірити після локальної зміни, скорочується у 2–3 рази. Це особливо важливо в умовах застосування практик безперервної інтеграції, де ефективність перевірки відіграє критичну роль.

Метод дозволяє не лише виявляти порушення узгодженості, але й підтримує симетричну синхронізацію: кожна зміна в структурі або поведінці спричиняє відповідне пропагування змін до іншої частини моделі. Ця властивість відрізняє підхід від односторонніх трансформацій та забезпечує його застосовність у повноцінних системах управління еволюцією моделей.

Разом із перевагами, підхід має певні обмеження: він потребує дотримання правил трасування між поданнями, а також поки не охоплює всі типи UML-діаграм. Втім, завдяки гнучкості та чіткому формалізованому підґрунтю, запропоноване рішення можна розширювати та адаптувати під інші профілі та сценарії застосування. В результаті, метод створює надійну основу для узгодженого розвитку складних програмних моделей у сучасному інженерному середовищі.

Список використаної літератури

1. A systematic identification of consistency rules for UML diagrams / D. Torre et al. *Journal of Systems and Software*. 2018. Vol. 144. P. 121–142. URL: https://doi.org/10.1016/j.jss.2018.06.029

2. Cicchetti A., Ciccozzi F., Pierantonio A. Multi-view approaches for software and system modelling: a systematic literature review. *Software and Systems Modeling*. 2019. Vol. 18, no. 6. P. 3207–3233. URL: https://doi.org/10.1007/s10270-018-00713-w

3. Stevens P. Bidirectional model transformations in QVT: semantic issues and open questions. Software & Systems Modeling. 2008. Vol. 9, no. 1. P. 7–20. URL: https://doi.org/10.1007/s10270-008-0109-9

4. Formal analysis of model transformations based on triple graph grammars / F. HERMANN et al. Mathematical Structures in Computer Science. 2014. Vol. 24, no. 4. URL: https://doi.org/10.1017/s0960129512000370

5. Buchmann T., Westfechtel B. Using triple graph grammars to realise incremental round-trip engineering. IET Software. 2016. Vol. 10, no. 6. P. 173–181. URL: https://doi.org/10.1049/iet-sen.2015.0125

6. Mens T., Van Der Straeten R. Incremental Resolution of Model Inconsistencies. Recent Trends in Algebraic Development Techniques. Berlin, Heidelberg, 2007. P. 111–126. URL: https://doi.org/10.1007/978-3-540-71998-4_7

7. Oriol X., Teniente E. Incremental Checking of OCL Constraints with Aggregates Through SQL. Conceptual Modeling. Cham, 2015. P. 199–213. URL: https://doi.org/10.1007/978-3-319-25264-3_15

8. Knapp A., Mossakowski T. Multi-view Consistency in UML: A Survey. Graph Transformation, Specifications, and Nets. Cham, 2018. P. 37–60. URL: https://doi.org/10.1007/978-3-319-75396-6_3 (date of access: 11.08.2025).

9. Komleva N. O., Nikitchenko M. I. Method for Incremental Control of Consistency Between Structural and Behavioral Views of Software Architecture. AAIT. 2025. Vol. 8, no. 2. P. 162–177. URL: https://doi.org/10.15276/aait.08.2025.11

10. Комлева Н. О., Нікітченко М. І. Структурне подання UML-метамоделі у форматі JSON. Інформатика та математичні методи в моделюванні. 2025. Том 15, № 2. С. 205–217. URL: https://doi.org/10.15276/imms.v15.no2.205

11. Комлева Н. О., Нікітченко М. І. Поведінкове подання у форматі XMI в межах UML-метамоделі з двома поданнями. Вісник Хмельницького національного університету. Технічні науки. 2025. Том 5 (Подано до друку).

12. Нікітченко М. І. Аналіз форматів збереження UML-моделей у сучасних CASE-засобах. Технології та суспільство: взаємодія, вплив, трансформація : матеріали IV Міжнар. наук. конф. (Чернігів, 20 червня 2025 р.). Чернігів, 2025. URL: https://doi.org/10.62731/mcnd-20.06.2025

References

1. Torre, D., Labiche, Y., Genero, M., & Kassab, M. M. (2018). A systematic identification of consistency rules for UML diagrams. Journal of Systems and Software, 144, 121–142. https://doi.org/10.1016/j.jss.2018.06.029

2. Cicchetti, A., Ciccozzi, F., & Pierantonio, A. (2019). Multi-view approaches for software and system modelling: A systematic literature review. Software and Systems Modeling, 18(6), 3207–3233. https://doi.org/10.1007/s10270-018-00713-w

3. Stevens, P. (2008). Bidirectional model transformations in QVT: Semantic issues and open questions. Software & Systems Modeling, 9(1), 7–20. https://doi.org/10.1007/s10270-008-0109-9

4. Hermann, F., Ehrig, H., Golas, U., & Orejas, F. (2014). Formal analysis of model transformations based on triple graph grammars. Mathematical Structures in Computer Science, 24(4). https://doi.org/10.1017/s0960129512000370

5. Buchmann, T., & Westfechtel, B. (2016). Using triple graph grammars to realise incremental round-trip engineering. IET Software, 10(6), 173–181. https://doi.org/10.1049/iet-sen.2015.0125

6. Mens, T., & Van Der Straeten, R. (2007). Incremental resolution of model inconsistencies. In A. Tarlecki (Ed.), Recent trends in algebraic development techniques (pp. 111–126). Springer. https://doi.org/10.1007/978-3-540-71998-4_7

7. Oriol, X., & Teniente, E. (2015). Incremental checking of OCL constraints with aggregates through SQL. In P. Johannesson, M. L. Lee, S. W. Liddle, A. L. Opdahl, & Ó. Pastor López (Eds.), Conceptual modeling (pp. 199–213). Springer. https://doi.org/10.1007/978-3-319-25264-3_15

8. Knapp, A., & Mossakowski, T. (2018). Multi-view consistency in UML: A survey. In R. Heckel & G. Taentzer (Eds.), Graph transformation, specifications, and nets (pp. 37–60). Springer. https://doi.org/10.1007/978-3-319-75396-6_3

9. Komleva, N. O., & Nikitchenko, M. I. (2025). Method for incremental control of consistency between structural and behavioral views of software architecture. AAIT, 8(2), 162–177. https://doi.org/10.15276/aait.08.2025.11

10. Komleva, N. O., & Nikitchenko, M. I. (2025). Strukturne podannia UML-metamodeli u formati JSON. Informatyka ta matematychni metody v modeliuvanni, 15(2), 205–217. https://doi.org/10.15276/imms.v15.no2.205

11. Komleva, N. O., & Nikitchenko, M. I. (2025). Povedinkove podannia u formati XMI v mezhakh UML-metamodeli z dvoma podanniamy[Manuscript submitted for publication].

12. Nikitchenko, M. I. (2025, June 20). Analiz formativ zberezhennia UML-modelei u suchasnykh CASE-zasobakh [Paper presentation]. Technology and society: interaction, influence, transformation: IV International Scientific Conference, Chernihiv, Ukraine. https://doi.org/10.62731/mcnd-20.06.2025

Дата першого надходження рукопису до видання: 24.09.2025
Дата прийнятого до друку рукопису після рецензування: 21.10.2025
Дата публікації: 28.11.2025