DOI: https://doi.org/10.15276/ict.01.2024.15
УДК 004.43

Управління та редагування UML-документів: структура, інтеграція, автоматизація

Нікітченко Максим Ігорович
Аспірант каф. Інженерії програмного забезпечення
ORCID: https://orcid.org/0009-0007-9560-7057; maksym.nikitchenko@gmail.com
Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

Unified Modeling Language (UML) посідає важливе місце в сучасному процесі розроблення програмного забезпечення (ПЗ), пропонуючи стандартизовані методи візуального моделювання, аналізу та проектування складних систем. У цій роботі розглядаються основні аспекти зберігання, редагування та інтеграції UML-документів з інтегрованими середовищами розробки (IDE). Основна увага приділяється порівнянню двох основних форматів зберігання UML-моделей – XML Metadata Interchange (XMI) і JavaScript Object Notation (JSON). XMI, будучи офіційним стандартом, розробленим Object Management Group, забезпечує високий ступінь деталізації та сумісність із різними UML-інструментами, що робить його кращим для великих і складних проектів, які потребують підтримки повної специфікації UML. Водночас JSON вирізняється простотою і гнучкістю, що робить його придатним для проектів, де важлива швидкість розроблення і легкість інтеграції, хоча він і поступається XMI в можливостях опису складних аспектів UML-моделей.

У роботі детально аналізуються методи редагування UML-документів, включно з ручним редагуванням і автоматизованими підходами, з використання програмних інтерфейсів додатків і скриптів. Ручне редагування корисне у випадках, коли потрібно внести невеликі зміни в структуру UML-документа, однак воно може бути трудомістким і схильне до помилок під час роботи з великими проектами. Автоматизація, навпаки, надає ефективні інструменти для масового редагування та генерації UML-елементів, що істотно прискорює розробку та мінімізує ймовірність помилок, тим паче, що сучасні інструменти, такі як StarUML, пропонують необхідні інструменти для програмної зміни UML-документів, інтегруючи їх у процеси розроблення.

Поєднання UML-документів з інтегрованими середовищами розробки також відіграє ключову роль у підвищенні ефективності розробки, адже це дає змогу автоматизувати генерацію коду на основі UML-моделей, підтримувати синхронізацію між кодом і діаграмами, а також полегшує візуалізацію та документування архітектурних рішень. Незважаючи на значні переваги, існують виклики, пов'язані з обмеженою підтримкою всіх можливостей UML і потенційними конфліктами між моделями та кодом. Це лише підкреслює необхідність подальших досліджень у цій галузі.

Перспективи подальших досліджень включають розробку нових методів та інструментів для роботи з UML-документами, поліпшення їхньої інтеграції з середовищами розробки, а також використання штучного інтелекту для автоматизації аналізу та проектування UML-моделей. Також варто приділити увагу використання UML у процесах неперервної інтеграції, що може значно підвищити гнучкість та адаптивність розробки програмного забезпечення. Таким чином, дослідження структури зберігання, редагування та інтеграції UML-документів являє собою важливий напрямок для оптимізації процесів проектування і розробки сучасних програмних систем.

Ключові слова: UML, структура UML-документа; редагування UML; автоматизація проектування; зворотне проектування; моделювання програмного забезпечення; діаграми класів; IDE; інструменти UML; JSON; XMI

Актуальність. У сучасних умовах розроблення програмного забезпечення UML (Unified Modeling Language) посідає важливе місце як стандартна мова візуального моделювання, забезпечуючи можливість створення архітектурних моделей систем на різних рівнях абстракції. UML активно використовується на стадіях аналізу та проектування, надаючи потужні засоби для формалізації вимог, моделювання структур і поведінки систем, а також документування архітектурних рішень. Завдяки своїй гнучкості та широкому поширенню, UML став де-факто стандартом для моделювання складних систем, включно з тими, що реалізують паралельні та розподілені процеси [1].

Зі зростанням складності систем, що розробляються, виникає необхідність у глибшому аналізі та автоматизації роботи з UML-документами. Самі по собі UML-діаграми є важливими елементами в процесі проектування, але їхня ефективність безпосередньо залежить від структури зберігання і можливості редагування. Це і робить актуальним дослідження цієї теми. Розуміння формату зберігання UML-документів, а також методів їхнього автоматизованого аналізу та зміни відкриває нові можливості для підвищення

якості та ефективності розроблення програмних систем. На поточний момент більшість інструментів зосереджено на візуальному моделюванні, залишаючи редагування на рівні файлів на другому плані. Ситуація, що склалася, вимагає розроблення нових інструментів, здатних усунути вузькі місця в роботі з UML-документами, включно з автоматизацією редагування та інтеграцією цих процесів із сучасними середовищами розроблення IDE. Необхідність у таких інструментах стає дедалі очевиднішою, особливо в контексті підтримки складних і розподілених систем, де найменша помилка в моделі може призвести до серйозних наслідків на етапі реалізації.

Мета дослідження – розробка методів та інструментів, які розв'язують проблеми, пов'язані зі зберіганням, редагуванням та автоматизацією роботи з UML-документами. Важливим аспектом є інтеграція цих методів з IDE, що дасть змогу мінімізувати ручну працю, зменшити вірогідність помилок і забезпечити тісніший зв'язок між процесами моделювання та розробки. Ця робота націлена на усунення наявних обмежень і поліпшення якості та ефективності проектування програмних систем.

Структура зберігання UML-документів. Однією з критичних задач під час використання UML у розробці програмного забезпечення є забезпечення надійного та ефективного зберігання й оброблення діаграм у форматі, придатному для автоматизації та інтеграції з іншими інструментами розроблення. Недоліки в структурі зберігання можуть призвести до значних проблем, включно із втратою даних і складністю в редагуванні, що робить правильний вибір формату зберігання ключовим фактором успішності проекту. У сучасних UML-інструментах для зберігання діаграм найчастіше використовують формати XMI і JSON. Ці формати забезпечують представлення моделей UML, що дає змогу легко передавати й обмінюватися даними між різними інструментами та платформами.

Нижче наведена Таблиця із порівнянням основних характеристик цих форматів.

XMI є офіційним стандартом, розробленим Object Management Group [2], і надає високий ступінь деталізації та сумісність із різними UML-інструментами. Цей формат ідеально підходить для складних і великих проектів, які потребують повної підтримки специфікації UML. Однак XMI має складну структуру, що робить його менш зручним для ручного редагування та обробки, особливо у випадках, коли потрібне швидке внесення змін.

JSON вирізняється простотою та гнучкістю, що робить його кращим у проектах, де важливі швидкість розробки та легкість інтеграції. Цей формат зручний для швидкого парсингу і редагування, що спрощує роботу з UML-документами на етапах прототипування або в умовах вимог, що динамічно змінюються. Мінус – JSON поступається XMI в підтримці повної специфікації UML і може не охоплювати всі можливі аспекти й деталі моделей, що може призвести до неповного передавання моделі та виникнення помилок у подальшому процесі розроблення для великих і складних проектів.

Тому вибір між XMI і JSON залежить від специфіки проекту: XMI краще підходить для складних систем із високим рівнем деталізації, тоді як JSON – для легших рішень [3].

Таблиця. **Порівняння критеріїв розглянутих структур**

Характеристика	XMI	JSON
Високий ступінь деталізації	+	-
Сумісність із різними UML-інструментами	+	-
Підтримка повної специфікації UML	+	-
Легкість опрацювання та парсингу	-	+
Простота в редагуванні вручну	-	+
Універсальність в обміні даними	+	+
Гнучкість і адаптованість	-	+
Великий обсяг і складність структури	+	-

Нижче наведено приклад структури зберігання UML-документів за допомогою JSON-документа, використовуваного в StarUML для зберігання окремого елементам діаграми класів.

```
{
  "_type": "UMLClass",
  "_id": "AAAAAAGRp/BsT8IgaTk=",
  "name": "Example",
  "visibility": "public",
  "attributes": [{
        "_type": "UMLAttribute",
        "_id": " attr1",
        "name": "username",
        "visibility": "public"
    }
  ],
  "operations": [{
        "_type": "UMLOperation",
        "_id": " op1",
        "name": "isAuthenticated",
        "visibility": "public"
    }
  ]
}
```

Можна спостерігати, як JSON описує різні елементи UML-діаграми, включно з класами, їхніми атрибутами, методами. Але, як уже було сказано, JSON обмежений у плані можливості опису складних аспектів моделі. Щоб це показати, нижче наведено приклад XMI-структури для аналогічної діаграми класів, і з додаванням деяких особливостей, які є у XMI. Як можна побачити, використання XMI надає більш високий рівень деталізації.

```
<uml:Class xmi:id="class1" name="Example" visibility="public" isAbstract="false">
  <ownedAttribute xmi:id="attr1" name="username" visibility="public"/>
  <ownedOperation xmi:id="op1" name="isAuthenticated" visibility="public"/>
  <generalization xmi:id="gen1" general="AAAAAAFF+qBWK6M3Z8Y="/>
  <ownedBehavior xmi:type="uml:StateMachine" xmi:id="stateMachine1"
name="ExampleStateMachine">
    <region xmi:type="uml:Region" xmi:id="region1">
      <subvertex xmi:type="uml:State" xmi:id="state1" name="Active"/>
      <subvertex xmi:type="uml:State" xmi:id="state2" name="Inactive"/>
      <transition xmi:type="uml:Transition" xmi:id="trans1" name="Activate"
source="state2" target="state1"/>
    </region>
  </ownedBehavior>
</uml:Class>
```

Крім елементів, ідентичних JSON, фрагмент XMI-документа також описує поведінку класу у вигляді кінцевого автомата, що моделює різні стани об'єкта і переходи між ними.

Елемент <ownedBehavior xmi:type=”uml:StateMachine” ...> представляє скінченний автомат, що належить класу “ExampleStateMachine". Усередині автомата визначено регіон зі

станами “Active” і “Inactive”, між якими здійснюється перехід “Activate”, що дає змогу керувати поведінкою об'єкта залежно від його поточного стану.

JSON може описувати подібні структури, якщо це потрібно, але менш формалізовано і з меншою деталізацією. Наприклад, можна використовувати JSON для представлення станів і переходів, однак він не надає стандартних засобів для опису таких моделей, що робить його менш зручним для складних UML-сценаріїв.

Буде це виглядати в такому вигляді:

```
{
  "type": "StateMachine",
  "name": "ExampleStateMachine",
  "states": [
    {
      "id": "state1",
      "name": "Active"
    },
    {
      "id": "state2",
      "name": "Inactive"
    }
  ],
  "transitions": [
    {
      "name": "Activate",
      "source": "state2",
      "target": "state1"
    }
  ]
}
```

Таким чином, вибір формату для зберігання UML-документів безпосередньо залежить від специфіки проекту і вимог до деталізації моделі. Неправильний вибір може призвести до серйозних наслідків, включно зі зниженням ефективності розроблення, ускладненням процесу інтеграції та підвищенням ризику виникнення помилок на пізніх стадіях проекту. XMI, з його високим ступенем деталізації, кращий для великих і складних систем, тоді як JSON надає гнучкість і простоту, що необхідна в умовах вимог, що динамічно змінюються, і швидкого розроблення.

Редагування UML-документів. Редагування UML-документів є складним і критично важливим процесом, від якого залежить здатність моделей адаптуватися до вимог, архітектури та стандартів проектування, що постійно змінюються. Помилки або недостатня гнучкість у редагуванні можуть призвести до серйозних проблем на етапах реалізації та тестування, що підкреслює необхідність використання ефективних методів для управління змінами. Нижче розглядаються основні методи редагування UML-документів: ручне редагування та автоматичне [4].

У деяких випадках, особливо коли використовуються прості діаграми, розробники можуть вручну змінювати вміст файлів UML-документів. Наприклад, додавання нового класу або атрибута можна здійснити шляхом вставки відповідного блоку до файлу, чи то JSON, чи то XMI, чи то інша структура даних, що використовується, що може бути корисним у ситуаціях, коли необхідно внести точкові зміни, що не вимагають використання графічного редактора. Проте воно несе в собі ризики, пов'язані з людським фактором, адже внесена

зміна вимагає ретельної перевірки, адже найменша помилка може призвести до порушення цілісності моделі та виникнення труднощів під час подальшої інтеграції з іншими системами.

Водночас автоматичне редагування має на увазі використання API або вбудовані засоби скриптування (наприклад, за допомогою JavaScript або Python) у сучасних інструментах, наприклад, у StarUML [5], які ж і дають змогу автоматично змінювати UML-документи. Автоматичне редагування стає незамінним в умовах сучасних проектів, де потрібні масові коригування або автоматична генерація певних частин моделі. Переваги автоматизації очевидні: вона дає змогу уникнути помилок, характерних для ручного редагування, і значно прискорює процес внесення змін, що особливо важливо в умовах суворих дедлайнів.

Варто також згадати, що деякі інструменти, наприклад, PlantUML, дають змогу автоматично генерувати UML-діаграми на основі текстового опису. Це зручно для CI/CD процесів, де документація має оновлюватися разом із кодом. Крім того, існують бібліотеки для роботи з XMI на Python, такі як pyecore, що дозволяють змінювати UML-документи, зберігаючи їх у форматі XMI. Або ж застосування ентропійного підходу, що може бути адаптовано для автоматизованого контролю за змінами в UML-документації [6]. Тим не менш, варто враховувати, що при редагуванні UML-документів на рівні файлів будь-які зміни можуть значно вплинути на візуальне представлення діаграм під час їхнього завантаження в графічний редактор. Скажімо, якщо змінити атрибут або метод класу, це може вплинути на відображення залежностей і зв'язків між елементами на діаграмі. Наприклад, додавання нової асоціації між класами вимагає не лише оновлення відповідних XML-блоків, а й коригування візуального відображення.

Вибір методу редагування UML-документів має ґрунтуватися на специфіці проекту та потенційних ризиках, пов'язаних зі змінами. У той час як ручне редагування може підходити для простих і малозначних змін, автоматизація процесу є надійнішим рішенням для великих і складних проектів, де важливими є точність, масштабованість і ефективність.

Інтеграція з IDE. Інтеграція UML-документів з інтегрованими середовищами розроблення значно спрощує та прискорює процес створення програмного забезпечення, даючи змогу розробникам генерувати код на основі UML-моделей, візуалізувати наявний код і підтримувати синхронізацію між кодом і діаграмами, що сприяє підвищенню якості архітектурних рішень. Незважаючи на переваги, інтеграція UML з IDE стикається з серйозними обмеженнями. Однією з головних перешкод є неповна підтримка всіх можливостей UML в існуючих IDE, що може призвести до невідповідностей і конфліктів між моделлю та вихідним кодом, і в результаті можуть викликати затримки в розробці, збільшувати ризики виникнення помилок і вимагати значних зусиль для підтримки синхронізації.

Для розв'язання цих проблем і поліпшення інтеграції UML з IDE необхідні нові підходи. Перспективними напрямками є вдосконалення механізмів синхронізації між моделями та кодом, що дасть змогу мінімізувати конфлікти та підтримувати актуальність моделей. Також варто розглянути технологію автоматизованої побудови програмних класів, що дозволяє формувати модель класів з набагато більшою інформаційною насиченістю, ніж традиційні UML-діаграми [7]. Запропоновані підходи значно скорочують кількість помилок та час розробки, що може бути адаптовано і для UML-документів в інших інформаційних системах.. На додаток до цього, використання штучного інтелекту для аналізу та оптимізації UML-документів може відкрити нові можливості для підвищення ефективності розробки та поліпшення архітектурних рішень. Тим паче, що на поточний момент вже відбуваються дослідження по використанню чат-ботів [8], досвід яких може бути використаним у сфері UML, та розробка методів для покращення та підвищення точності результатів [9].

Перспективи подальших досліджень. Подальші дослідження мають зосередитися на розв'язанні найактуальніших проблем і на розробці нових підходів, що дадуть змогу

подолати наявні обмеження UML і зробити процес розроблення програмного забезпечення ефективнішим і надійнішим.

Зокрема, такі напрями потребують особливої уваги:

1. Розширення функціональності UML-інструментів. Існують значні можливості для поліпшення наявних UML-інструментів, включно з розробленням нових форматів зберігання та методів аналізу моделей. Це може включати додавання підтримки нових мов програмування і технологій, а також поліпшення роботи з паралельними і розподіленими системами.

2. Глибока інтеграція з сучасними IDE. Дослідження можуть бути спрямовані на розробку методів, які ще тісніше інтегрують UML з IDE, забезпечуючи безперервну синхронізацію між UML-документами і вихідним кодом і, своєю чергою, формувати поліпшення механізмів як автоматичної генерації коду, так і зворотного проектування.

3. Застосування штучного інтелекту. ШІ, застосований для роботи з UML-документами – гарний шлях при нинішньому стані такої технології, як ChatGPT, незважаючи на її обмеження. Він добре працює при створенні простих UML-моделей, але не справляється з генерацією великих і складних [10]. Майбутні дослідження мають бути зосереджені на усуненні синтаксичних і семантичних помилок, а також на поліпшенні здатності ШІ працювати зі складними конструкціями UML, такими як множинне успадкування або асоціативні класи. Особливу увагу слід приділити розробленню методів, що дають змогу ШІ краще розуміти контекст і специфіку галузі, що призведе до створення більш точних і ефективних архітектур.

4. Інтеграція з DevOps і CI/CD (Continuous Integration/Continuous Deployment). Для підтримки постійного контролю за змінами в UML-документах можна використовувати систему версій, яку можна порівняти з процесом, що використовується в CI/CD, методологія якої і побудована для швидкого впровадження змін, завдяки чому зміни в UML-моделях будуть автоматично фіксуватися та узгоджуватися з основним репозиторієм вихідного коду [11]. Це не тільки поліпшить видимість процесу розробки, а й спростить версіонування UML-документів.

Таким чином, подальші дослідження в галузі UML-документів мають потенціал істотно поліпшити не тільки процеси проектування, а й увесь цикл розроблення програмного забезпечення, що може призвести до створення надійніших, гнучкіших та ефективніших інструментів.

Висновок. Під час цього дослідження було розглянуто ключові аспекти роботи з UML-документами, починаючи зі способів їхнього представлення і редагування, і закінчуючи перспективами подальших досліджень і поліпшень у сфері їхньої інтеграції з сучасними інструментами розробки. Автоматизація процесів редагування і глибоко інтегровані рішення з використанням API і скриптів дають змогу значно підвищити ефективність роботи з UML-моделями, особливо у великих і складних проектах. Ручне редагування залишається корисним інструментом для невеликих і простих змін, тоді як автоматизація відкриває можливості для масових коригувань і швидкої генерації нових елементів моделі.

Інтеграція UML-документів з IDE забезпечує не тільки зручну генерацію коду на основі моделей, а й підтримує синхронізацію діаграм з вихідним кодом, що підвищує якість і узгодженість архітектурних рішень. Водночас, залишаються виклики, пов'язані з неповною підтримкою всіх можливостей UML та конфлікти між моделями та кодом, що відкриває можливості для подальшого вдосконалення інтеграційних механізмів.

Перспективи подальших досліджень спрямовані на розвиток наявних інструментів і методів роботи з UML-документами. Уже згадані поглиблення інтеграції з сучасними IDE, а також розвиток форматів зберігання UML і застосування штучного інтелекту для аналізу та проектування можуть призвести до створення гнучкіших та інтелектуальних рішень для складних систем. Впровадження цих технологій у CI/CD і DevOps процеси також видається

перспективним напрямком, здатним істотно підвищити гнучкість розроблення програмного забезпечення і якість продукту, що випускається.

Тому, подальше дослідження і розвиток інструментів для роботи з UML-документами відкривають нові можливості як для оптимізації процесів проектування, так і для розробки програмного забезпечення, роблячи їх більш адаптивними, автоматизованими та ефективними.

СПИСОК ЛІТЕРАТУРИ

1. “The unified modeling language user guide”, ed. by R. James, J. Ivar. 2nd ed. *Upper Saddle River. NJ: Addison-Wesley*. 2005.

2. “OMG XMI 2.5.1. XML Metadata Interchange. Replaces OMG XMI 2.5; effective from 2015-06-01. Official edition”. *Object Management Group*. 2015.

3. Benson T., Grieve G. “Principles of Health Interoperability. UML, XML and JSON”. Cham, 2020. p. 399–426. DOI: https://doi.org/10.1007/978-3-030-56883-2_21.

4. Gomaa H. “Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures”. *Cambridge University Press*. 2010.

5. “StarUML API Documentation: UML Module”. –Available from: https://files.staruml.io/api-docs/2.0.0/api/modules/uml/UML.html.

6. Gogunskii V. D., Kolesnikova K. V., Lukianov D. V. “Entropy analysis of organizations' knowledge systems on the example of project management standards”. *Applied Aspects of Information Technology.* 2022; 5 (2): 91–104. DOI: https://doi.org/10.15276/aait.05.2022.7.

7. Kungurtsev O. B., et al. “Automated object-oriented technology for software module development”. *Applied Aspects of Information Technology*. 2021; 4 (4): 338–353. DOI: https://doi.org/10.15276/aait.04.2021.4.

8. Kobets V. M., Kozlovskyi K. H. “Application of chat bots for personalized financial advice”. *Herald of Advanced Information Technology*. 2022; 5 (3): 229–242. DOI: https://doi.org/10.15276/hait.05.2022.18

9. Boyko N. I., Muzyka M. V. “Methods of analysis of multimodal data to increase the accuracy of classification”. *Applied Aspects of Information Technology*. 2022; 5 (2): 147–160. DOI: https://doi.org/10.15276/aait.05.2022.11.

10. Cámara J., et al. “On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML”. *Software and Systems Modeling*. 2023. DOI: https://doi.org/10.1007/s10270-023-01105-5.

11. Donca I.-C., et al. “Method for continuous integration and deployment using a pipeline generator for agile software projects”. *Sensors.* 2022; 22 (12): p. 4637. DOI: https://doi.org/10.3390/s22124637.

DOI: https://doi.org/10.15276/ict.01.2024.15
UDC 004.43

Managing and editing UML documents: structure, integration, automation

Maksym I. Nikitchenko
Postgraduate Student, Department of Software Engineering
ORCID: https://orcid.org/0009-0007-9560-7057; maksym.nikitchenko@gmail.com
Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine

ABSTRACT

Unified Modeling Language (UML) plays an important role in the modern software development process by offering standardized methods for visual modeling, analysis, and design of complex systems. This paper discusses the main aspects of storing, editing, and integrating UML documents with integrated development environments (IDEs). The main focus is on comparing the two main formats for storing UML models – XML Metadata Interchange (XMI) and JavaScript Object Notation (JSON). XMI, being an official standard developed by Object Management Group, provides a high degree of detail and compatibility with various UML tools, making it preferable for large and complex projects that require support for the full UML specification. At the same time, JSON is simple and flexible, which makes it suitable for projects where speed of development and ease of integration are important, although it is inferior to XMI in describing complex aspects of UML models.

The paper analyzes in detail the methods of editing UML documents, including manual editing and automated approaches, using APIs and scripts. Manual editing is useful when you need to make small changes to the structure of a UML document, but it can be time-consuming and error-prone when working with large projects. Automation, on the other hand, provides effective tools for mass editing and generating UML elements, which significantly speeds up development and minimizes the likelihood of errors, especially since modern tools such as StarUML offer the necessary tools for programmatically modifying UML documents by integrating them into development processes.

Combining of UML documents with integrated development environments also plays a key role in increasing development efficiency, as this allows automating code generation based on UML models, maintaining synchronization between code and diagrams, and facilitating visualization and documentation of architectural solutions. Despite the significant advantages, there are challenges associated with limited support for all UML features and potential conflicts between models and code. This only emphasizes the need for further research in this area.

Prospects for further research include the development of new methods and tools for working with UML documents, improving their integration with development environments, and using artificial intelligence to automate the analysis and design of UML models. Attention should also be paid to the using of UML with continuous integration processes, which can significantly increase the flexibility and adaptability of software development. Thus, the study of the structure of storing, editing, and integrating UML documents is an important area for optimizing the design and development of modern software systems.

Keywords: Unified Modeling Language (UML); UML document structure; UML editing; design automation; reverse engineering; software modeling; class diagrams; integrated development environment (IDE); UML tools, JavaScript Object Notation (JSON); XMI