УДК 681.3.06

МОДЕЛЕ-ОРІЄНТОВНИЙ ПІДХІД ДО СТВОРЕННЯ СИСТЕМ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАННИХ

О.А. Алєксєєнко

Інститут кібернетики ім. В.М. Глушкова НАН України
03680, Київ-187, проспект Академіка Глушкова, 40.
Тел.: (044) 526 6422, e-mail: a.aleksey@ua.fm

Розглядається моделе-орієнтовний підхід до створення систем інтелектуального аналізу даних заснованих на Байєсівських мережах довіри. Запропоновано схему застосування даного підходу, з урахуванням всіх етапів, процесів, вхідних і вихідних даних. Розроблена формальна мова опису трансформацій моделей, заснована на предикатах першого порядку, технологіях трансформації XML-документів та стандартах обміну мета-даними XMI. Описано спосіб застосування стандарту MathML для опису Байєсівських алгоритмів на рівні моделей систем інтелектуального аналізу даних.

The model-oriented approach is examined to the development of the systems of intellectual analysis of data (SIAD), based on the Bayesian belief networks. The usage scheme of this approach is offered, taking into account all his stages, processes, input and output data. A formal language is developed for description of model transformations, based on the first-order predicate logic, technologies of XML documents transformation and metadatas exchange based on XMI standards. The MathML standart usage for formal definition of the Bayesian mechanism at the different modeling level of the SIAD is described.

Вступ

З огляду на все більш широке коло задач як у державному секторі, так і у приватному, що вирішуються за допомогою систем інтелектуального аналізу даних (СІАД) заснованих на Байєсівських мережах довіри (БМД) [1−5] з'явилася необхідність більш детального дослідження алгоритмів, що використовуються у цих системах. З'явилася необхідність у розробці підходу, що дозволить швидко та якісно створювати нові, та змінювати існуючі СІАД. Такий підхід має охоплювати весь процес розробки, починаючи з опису математичної моделі майбутньої системи, закінчуючи створенням прототипів та робочих версій у рамках певної платформи. У роботах [6−9] запропоновано використовувати моделе-орієнтовний підхід до створення СІАД. Важливою вимогою до цього підходу є можливість автоматично підтримувати на всіх рівнях проектування зв'язок між математичною моделлю та кінцевим продуктом. Велика увага приділялася аспектам формалізації основних етапів застосування моделе-орієнтовного підходу з метою їх часткової або повної автоматизації. У рамках **запропонованого підходу було запропоновано** використання деяких існуючих технологій та стандартів, таких як Model-Driven Architecture (MDA) [10], XML Metadata Interchange (XMI) [11], Unifidet Modeling Language (UML) [12], Object Constraint Language (OCL) [13]. Описані методи використання цих технологій на різних етапах. У роботі [9] запропонована мова для опису трансформацій типу «модель-модель».

Мета данної роботи розглянути такі аспекти моделе-орієнтовного підходу як роль мета-моделей на різних етапах проектування, питання створення специфічних для заданої предметної області (СІАД) мета-моделей та механізмів опису математичних властивостей об'єктів СІАД. Використання мовних засобів для опису правил трансформації.

1. Вимоги до моделе-орієнтовного підходу до створення СІАД заснованих на БМД

Під час розробки моеделе-орієнтовного підходу до створення СІАД були враховані найбільш вагомі й необхідні вимоги, серед тих, що зазвичай ставлять перед технологіями розробки програмних систем. Запропонований підхід мав одночасно підвищити швидкість розробки СІАД та поліпшити якість кінцевої реалізації на конкретній платформі алгоритмів та структур даних описаних математичною мовою.

Для підвищення швидкості необхідно використовувати формальні механізми трансформації моделей, таким чином, щоб на пів автоматично переходити від узагальненого вигляду системи, що описує не конкретну систему, а певні властивості класу систем, до уточненого, що описує СІАД з урахуванням конкретних вимог. Це дозволить по-перше, підвищити кількість програмного коду, який буде використаний повторно, по-друге, на автоматичному рівні багаторазово використовувати шаблони проектування характерні для певного класу систем.

Питання підвищення якості реалізації математичних абстракцій (алгоритмів, структур даних) слід розглядати, як питання підвищення якості комунікації між математиками, що займаються дослідженням відповідних алгоритмів та розробниками кінцевих систем. Для цього необхідно створити проміжний рівень моделювання який дозволить будувати схему функціонування майбутньої СІАД базуючись в першу чергу на математичному апараті, а не на термінах об'єктно-орієнтовної розробки (ООР). Очевидно, для остаточного вирішення цього питання, необхідна наявність певного формального підходу до перетворення (трансформації) таких моделей у моделі ООР.

2. Етапи застосування моделе-орієнтовного підходу до створення СІАД

На рис. 1 показані всі етапи що мають бути включені в моделе-орієнтовний підхід до створення СІАД. Зображені всі документи (данні) та всі процеси (операції), що задіяні у підході. Далі розглянемо кожний з етапів, його основні компоненти та процеси, вхідні та вихідні документи, його значення для підходу. Найбільшу увагу приділимо першому етапу, так як він не розглядався у попередніх роботах, та етапу трансформації – найбільш складному етапу.

Етап мета-моделювання. Як вищезазначено, для поліпшення комунікації між математиками, що досліджують СІАД, та розробниками кінцевого продукту необхідно створити проміжний рівень моделювання. Цей рівень дозволить з одного боку використовувати математичні абстракції, як елементи моделі. З іншого боку має існувати можливість прозорого переходу від проміжної моделі (специфічної для певної предметної області) до моделі специфічної для певної реалізації (платформи). У якості платформо-залежної моделі може виступати UML модель. Що стосується специфічної для обраної предметної області (СІАД) моделі, не існує широко поширених мовних або графічних засобів для опису таких моделей. Тому пропонується, використовуючи стандарт мета-моделюваняя MOF 2.0, запропонований консорціумом Object Management Group (OMG) [14], створити власну мета-модель, що буде задовольняти усім вимогам до побудови кінцевих моделей СІАД виражених у математичних термінах.

На практиці, для створення специфічної для предметної області мета-моделі (Domain Specified Meta-Model, DSMM) пропонується використання будь-якого редактору UML, що підтримує стандарт UML 2.0 та XMI 2.1.

Існує можливість змінити, або доповнити мета-модель UML, щоб покращити її орієнтованість на певну платформу реалізації. Також слід зазначити, що у будь-якому випадку, для автоматизації процесу необхідна наявність мета-моделі специфічної для платформи реалізації (Realization Specified Meta-Model, RSMM). В не залежності від того використовуватиметься мета-модель UML у чистому вигляді, чи з певним розширенням, чи буде створена довільна мета-модель для опису реалізації – вона має бути виражена у термінах MOF, та на фізичному рівні представлена у форматі XMI. Єдність засобів мета-моделювання, моделювання та форматів збереження результатів цих процесів полегшують формалізацію процесів трансформації, а таким чином і їх автоматизацію. Дозволяють уніфіковано описувати алгоритми трансформації моделей на різних етапах та у різних напрямках.

Також слід зазначити, що під час проектування DSMM створюється одноразово, і використовується для побудови (моделювання) усіх систем в рамках обраної предметної області. RSMM створюється для кожної обраної технології або платформи, при умові використання в якості RSMM мета-моделі UML, вона може лишатися без змін, для моделювання всіх реалізацій СІАД, що підпорядковуються об'єктно орієнтовному програмуванню.

Етап створення узагальненої моделі. Цей етап, вимоги до нього, та рекомендації до створення узагальненої моделі описані в [6, 7]. Враховуючи, що у підхід включено поняття DSMM головним завданням розробника СІАД на цьому етапі моделювання є включення загальних, для певного класу математичних абстракцій що визначені у мета-моделі.

У рамках цього етапу, може бути створена довільна кількість узагальнених моделей, кожна з яких описуватиме певний клас СІАД, та відповідно створюватиме свою гілку в процесі розробки, яка матиме незалежні наступні етапи, та результат. Оскільки в цій та інших роботах ми розглядаємо СІАД засновані на Байєсівських мережах довіри (БМД), як основну предметну область, то під час практичного застосування підходу буде створено відповідну DSMM, та одноразово узагальнену модель, що визначатиме певний клас алгоритмів БМД.

Під час розробки узагальненої моделі, визначається можлива кількість результуючих систем. Тобто чи менше вимог та властивостей закладено в узагальнену модель, тим більше результуючих систем може бути створено на її основі, з одного боку, і тим більше уваги необхідно буде приділите кожній уточнюючий моделі. Дійсною буде і зворотня залежність. Отже визначення ступеню узагальнення моделі покладається на розробників і залежить від цілей, що ставляться перед проектом.

Етап трансформації моделей. Частково цей етап описаний в [6−8]. Але з урахування включення довільних мета-моделей у моделе-орієнтовний підхід даний етап потребує додаткового опису. Визначення терміну «трансформація моделей» у рамках моделе-орієнтовного підходу виглядатиме так:

трансформація моделей – це процес, що передбачає перетворення моделі описаної у термінах мета-моделі A, та реалізуючої множину функціональних вимог B, у модель що описана у термінах мета-моделі A_1, та реалізує множину функціональних вимог B_1. При цьому можлива одна з рівностей. $A = A_1$, тоді говоримо, що трансформація проходить у рамках однієї мета-моделі, але з уточненням вимог. $B = B_1$, трансформація проходить у рамках різних мета-моделей, але із сталим набором вимог.

З урахуванням цього визначення алгоритм виконання етапу трансформації моделей у загальному вигляді показано на рис. 2.

Зауважимо, що фактично множини вимог B та B_1 представлені лише існуючою моделлю. Тобто, наявність множини B_1 та її елементи можуть бути визначені лише після успішної трансформації у рамках однієї мета-моделі з уточненням вимог.

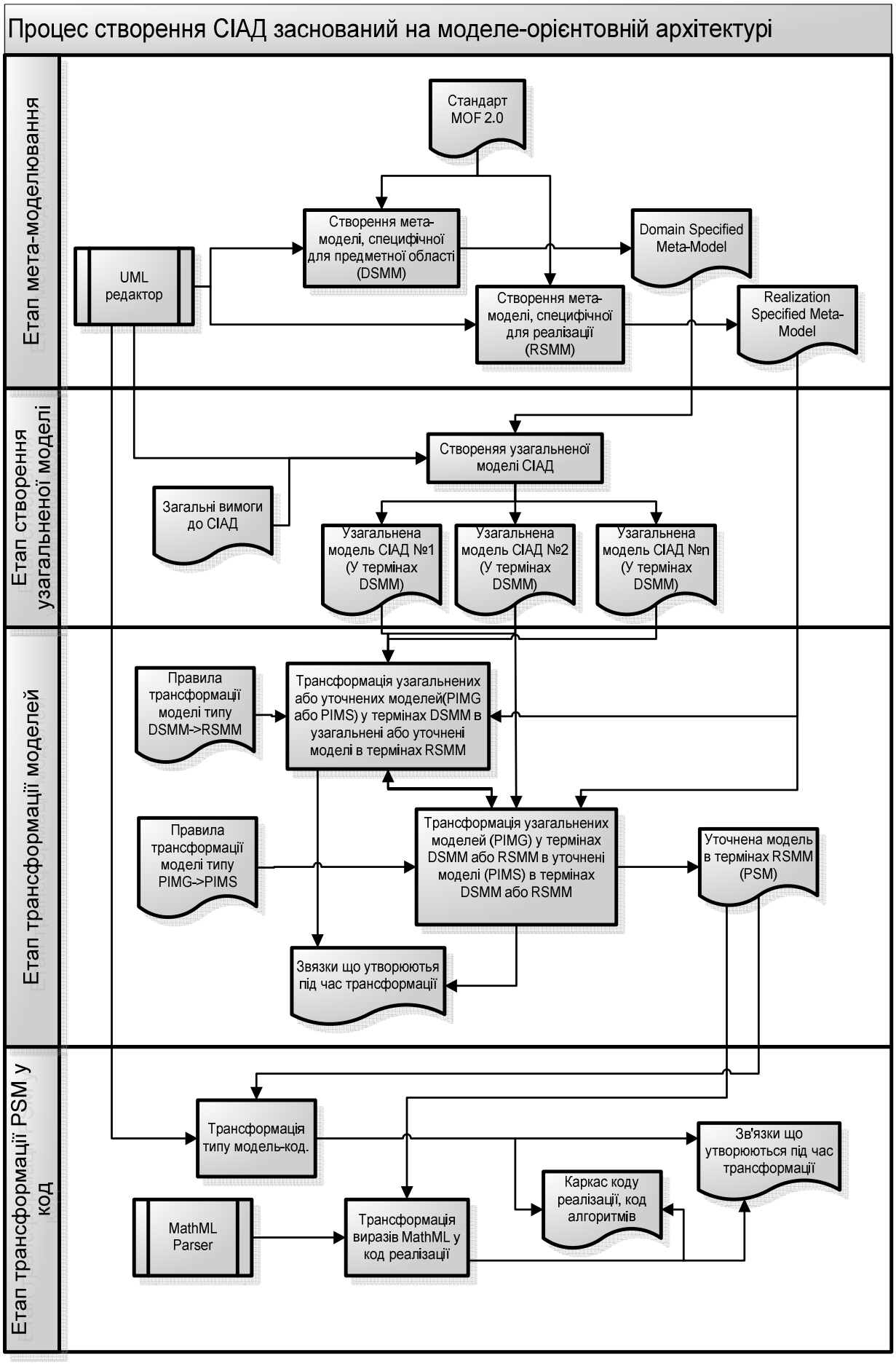

Рис. 1. Етапи застосування моделе-орієнтовного підходу до створення СІАД

В алгоритмі наведеному на рис. 2 до виконання блоку з функцією M = Trfnsformation(M, RR), вираз $B = B_1$ будемо вважати не рівним при наявності не пустої множини правил RR, адже саме вони визначають уточнені вимоги до моделі, а отже структуру і наповненість множини B_1.

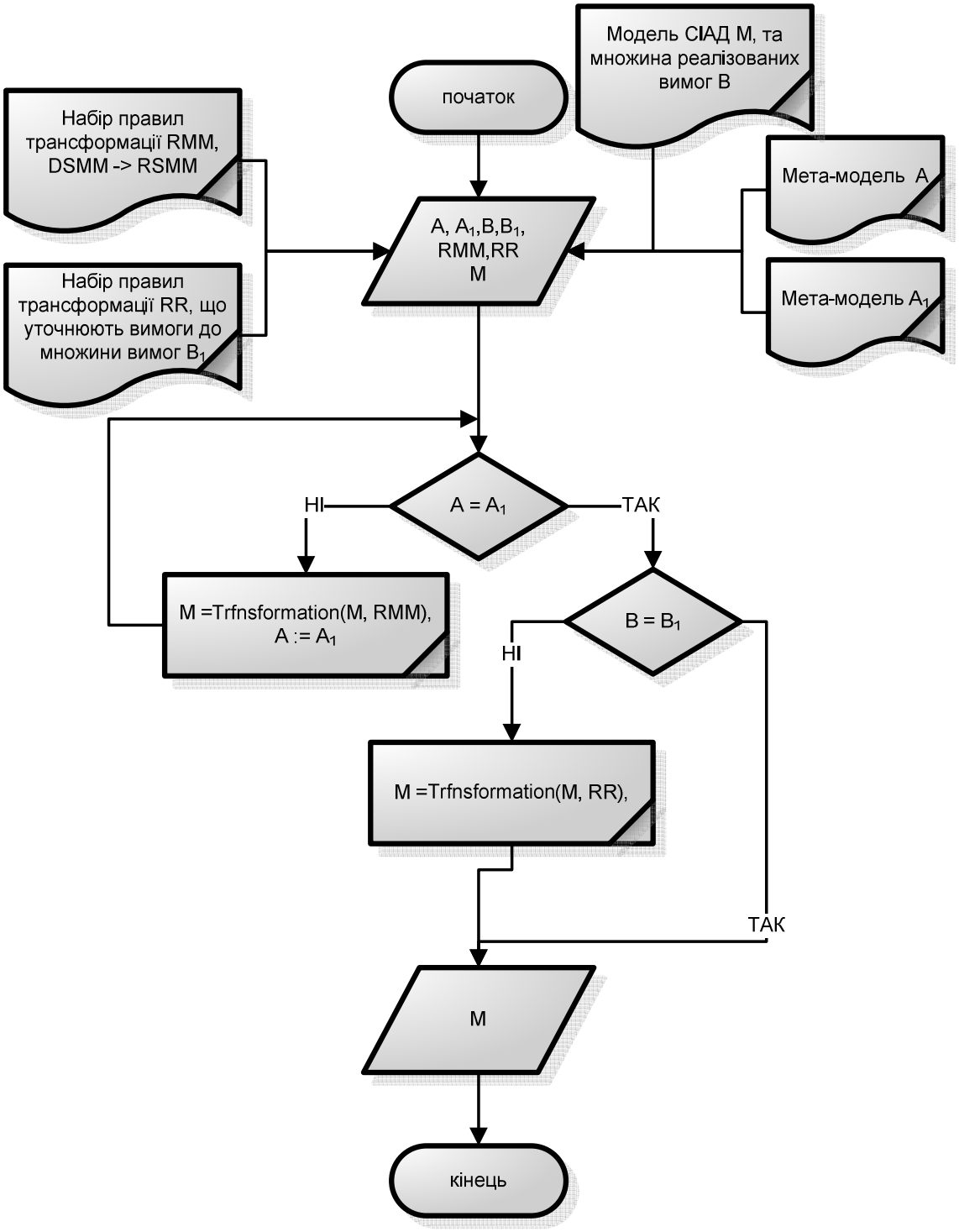

Рис. 2. Алгоритм виконання етапу трансформації моделей у загальному вигляді

Також очевидно, що перебіг трансформації залежить від вхідних даних, тобто користувач визначає який саме тип трансформації необхідно провести, шляхом вибору мета-моделей та правил трансформації.

На етапі трансформації для кожного виконання функції Trfnsformation(Model, Rules) створюється відповідна структура, що містить трансформаційні зв'язки. Тобто, посилання на вхідну і вихідну модель, їх мета-моделі, інформацію про всі кроки виконання правила, та всі проміжні й кінцеві множини елементів моделі які приймали участь у трансформації.

Після проведення всіх необхідних трансформацій, при умові, що хоча б одна за них була проведена на одній мета-моделі, але з уточненням вимог і хоча б одна в рамка різних мета-моделей без уточнення вимог користувач отримає уточнену платформо-залежну модель (Platform Specified Model, PSM).

Етап трансформації PSM у код. На цьому етапі отримана PSM з уточненими вимогами трансформується в каркас коду реалізації на обраній платформі. За умови, що у якості RSMM обрано мета-модель UML або її розширення результатом трансформації може бути набір класів з атрибутами та порожніми методами, структура БД. Таку трансформацію можна виконати за допомогою існуючих Open Source та комерційних редакторів UML. Необхідно доповнити функціонал такого програмного забезпечення, щоб підчас трансформації створювалася відповідна структура, що міститиме трансформаційні зв'язки аналогічно до попереднього етапу. У випадку якщо RSMM значно відхиляється від класичної UML моделі, і трансформація існуючими засобами неможлива, виникає необхідність створити додаткову мета-модель, що буде описувати елементи реалізації як синтаксис конкретної мови програмування та платформи, і провести трансформацію типу модель-модель, адже трансформація типу модель код є особливим випадком трансформації модель-модель. Той факт, що довільну RSMM можна трансформувати у платформо-залежний об'єктно орієнтовний код, витікає з того, що згідно запропонованого підходу RSMM виражена у термінах MOF.

За наповнення методів певним функціоналом відповідатиме розробник СІАД, і код цих методів буде створюватись у ручному режимі. Але за допомогою збереження трансформаційних зв'язків існуючий код, можна буде автоматично підіймати на більш високий рівень абстракції, і включати його в усі попередні моделі, що містять опис відповідної функціональності. Таким чином при зміні правил трансформації, на певному кроці, після того як код реалізації вже існує, в нову PSM, а відповідно і у каркас класів потрапить максимально можлива множина реалізованих методів.

Такий підхід дозволяє на автоматичному рівні використовувати і певні шаблони проектування СІАД і конкретні фрагменти коду реалізації. Очевидним недоліком є те, що опис конкретних математичних алгоритмів (формул, функцій) залишається поза процесом трансформації, і фактично розробник реалізує математичній вираз включений у певний метод в ручну, що може призвести до виникнення помилок під час реалізації, відстежити які буде досить важко, через експериментальний характер розроблюваних систем. Вирішенням проблеми є включення математичних виразів визначених засобами MathML [15].

3. Застосування MathML

MathML – Mathematical Markup Language [15], мовний засіб запропонований консорціумом W3C в першу чергу, як засіб обміну математичними формулами на машинному рівні. Пізніше цей стандарт почали активно використовувати для коректного відображення математичних формул на Web-сторінках. Мова MathML базується на стандарті XML, в якості математичних операцій використовуються відповідні, визначені стандартом тегі XML, а в якості змінних і констант значення цих тегів. Структура та порядок обчислень легко вписуються у деревовидну структуру XML формату. Завдяки такому підходу вирази описані за допомогою MathML, виражають не лише математичний синтаксис а й семантику, тобто існує можливість однозначно формально визначити послідовність виконання операцій, та змінні що в них задіяні. Ще однією перевагою MathML є наявність широкого кола готових Open Source компонентів, як для синтаксичного розбору MathML файлів, так і для візуалізації та візуального редагування формул описаних за допомогою MathML. Прикладом таких компонентів може бути MathML .NET Control [16], MathFlow [17], DragMath [18].

Далі наведено приклад запису виразу $\sqrt{b^2 - 4ac}$:

```
<msqrt>
    <mrow>
        <msup>
            <mi>b</mi>
            <mn>2</mn>
        </msup>
        <mo>-</mo>
        <mrow>
            <mn>4</mn>
            <mo>&InvisibleTimes;</mo>
            <mi>a</mi>
            <mo>&InvisibleTimes;</mo>
        <mi>c</mi>
        </mrow>
    </mrow>
</msqrt>
```

Крім наявності широкого кола компонентів для візуального редагування математичних формул та автоматичного створення відповідних файлів, стандарт MathML має ще декілька переваг. По-перше цей стандарт підтримується групою W3C, а відповідно постійно оновлюється, і є досить поширеним стандартом для обміну математичними даними на машинному рівні. По-друге, заздалегідь визначена деревовидна структура результуючого коду математичних виразів полегшує процеси синтаксичного розбору MathML. По-третє, MathML є підстандартом XML, то для нього дійсні всі існуючі засоби трансформації XML, такі як XSLT та XQuery, можливості розширення за допомогою додаткових тегів.

4. Опис трансформації. Мова опису трансформації

Одним з найскладніших процесів, серед тих що входять у запропонований підхід, є безпосередньо процес застосування правил трансформації [6] (функція Trfnsformation(M, RMM) на рис. 2), що неодноразово виконується на етапі трансформації моделей. Для створення функції застосування правил необхідна певна формальна мова для запису цих правил.

Основні вимоги до мови опису трансформацій. У загальному вигляді, ці вимоги можна описати наступним чином:

- універсальність, формальність і повнота. Мова опису трансформації має надавати можливість описати будь-яку трансформацію для будь-якої моделі підпорядкованої обраній метамоделі (в нашому випадку метамоделі UML). При цьому опис має бути формалізованим настільки, щоб була можливість його автоматичного виконання;
- наочність правил трансформації. Правила трансформації задані цією мовою мають бути наочними не лише для програміста, що їх розробив, а і для спеціалістів у заданій предметній області;
- підтримка цілісності при модифікації. Необхідно мати можливість підтримувати зв'язок між початковою і кінцевою моделями, для того щоб коректно відображати зміни початкової моделі у кінцевих, спираючись на правила трансформації.

Існуючі підходи до трансформації. Імперативний опис процесу трансформації з використанням будь-якої алгоритмічної мови. При цьому в середовище розробки на етапі його створення вбудовуються певний набір трансформацій, чи трансформаційних шаблонів, які пізніше можуть бути використані користувачем. Недолік цього підходу в неможливості додавати нові або змінювати існуючи трансформації, що робить розробника залежним від конкретного інструменту.

Використання математичного апарату для трансформації графів. Недолік такого підходу в тому, що UML-модель несе семантичне навантаження відмінне від формального графу. Це змусить користувача, який задає правила трансформації, постійно робити перехід від графу до відповідної йому UML-моделі.

Трансформація XML документів створених за стандартом XMI [11] (XML Metadata Interchange). XMI дозволяє представити будь-яку модель UML у вигляді XML документу. Існує декілька способів трансформації XML документів, наприклад XSLT або XQuery. При такому підході користувач якій створює правила трансформації повинен використовувати терміни XML, і орієнтуватися в першу чергу на трансформацію XML документа, а не моделі UML яку він представляє.

Означені недоліки цих підходів мають значення лише при самостійному їх використанню, але у поєднанні із специфічною мовою опису трансформацій їх використання може бути корисним. Наприклад, під час створення інтерпретатора для мови опису трансформацій.

Мова опису трансформацій. Кожен проект трансформацій (Transformation solution) складається з конфігураційного блоку (Configuration block) , так блоку трансформацій (Transformation block). Далі наведені синтаксичні нотації мови опису трансформацій за допомогою розширеної ФБН (форма Бекуса−Наура)

Transformation_solution ::= <Name> { <Configuration_block> <Transformation_block>* };

<Configuration_block> ::= Configuration_block { [Direct_style | OCL_style | XMI_Style] <Predicate_convert>* };

<Transformation_block> ::= Transformation_block <Name> { <Transformation_rule>* };

у конфігураційному блоці визначаються предикати, що можуть бути використані у трансформації та спосіб їх визначення. Пропонується використовувати декілька шляхів для визначення предикатів:

задані прямо – визначені виразами типу \vertex1\vertex2\...\vertexN на графі, що створений шляхом відображення класів метамоделі у вершині графа, а асоціативних зв’язків між класами в дуги;

задані мовою OCL – визначені за допомогою виразів OCL заданих на діаграмі класів метамоделі;

задані засобами XSLT/XPath – визначені за допомогою синтаксису XSLT/XPath на метамоделі конвертованої у XML файл за стандартом XMI.

Спосіб визначення предикатів залежить від відповідного параметра на початку блоку конфігурації.

<Predicate_convert> ::= <Predicate> = <Low_level_expression>;

Трансформаційний блок складається з правил трансформації, кожне з яких має ім'я, ліву та праву частину.

<Transformation_rule> ::= Transformation_rule <Name> { <Left_part> -> <Right_part> };

Ліва частина правла за допомогою виразу у термінах логіки предикатів визначає певну зону (шаблон) у вхідній моделі. За допомогою операторів if | if not визначається вимога щодо присутності або відсутності цього шаблону у вхідній моделі.

Права частина правла за допомогою виразу у термінах логіки предикатів визначає певну зону (шаблон) яку необхідно створити або видалити у кінцевій моделі. Необхідна дія визначається за допомогою операторів create або delete відповідно.

<Transformation_rule> ::= Transformation_rule <Name> { <Left_part> -> <Right_part> };
<Left_part> ::= if | if not <Predicate_expression>
<Right_part> ::= create | delete <Predicate_expression>

Обраний підхід дозволяє використовувати звичну для математиків, що займаються розробкою систем інтелектуального аналізу даних, нотацію логіки предикатів для створення правил трансформації. Перехід від предикатів, до сучасних засобів означення сутностей моделей у UML нотації, що задається у конфігураційному блоці, дозволяє під час створення інтерпретатора використовувати існуючи методи трансформації.

Також слід зазначити, що при використанні мета-моделі відмінної від мета-моделі UML, але визначеної у термінах MOF, наприклад, мета-модель систем інтелектуального аналізу даних заснованих на БМД, отримаємо мову опису трансформацій, що орієнтується на відповідну предметну область. Отже, запропонована мова підходить як для опису трансформацій у рамках різних мета-моделей, так і для опису трансформацій у рамках однієї мета моделі, але з уточненням вимог.

Ще одна перевага обраного підходу до створення мови опису трансформації, потенційна можливість переходу від текстового до графічного режиму визначення предикатів. Адже користувач може зробити це шляхом вибору асоціативних зв'язків у мета-моделі при умові, що вона задана у термінах UML. У такому випадку заповнення конфігураційного блоку можна виконати автоматично, адже перехід від UML нотації до XML документа формалізований у стандарті XMI.

Висновки

Запропонований у даній та попередніх роботах моделе-орієнтовний підхід до створення СІАД заснованих на БМД відповідає усім висунутим до нього вимогам, та задовольняє потреби як дослідників математичних апаратів так і розробників кінцевих систем.

Завдяки використанню у рамках запропонованого підходу механізмів мета-моделювання, існує можливість на початкових етапах розробки СІАД використовувати у якості елементів моделі математичні абстракції, які на відміну від абстракцій ООР, однозначно відповідають даній предметній області.

Створення проміжних узагальнених моделей, та їх трансформація в уточнені, дозволяє розробляти та підтримувати сімейства моделей, що мають низку спільних властивостей. Збереження трансформаційних зв'язків дозволяє автоматично застосовувати зміни в узагальнених моделях до уточнених. Інтеграція стандарту MathML у запропонований підхід, надає можливість використання звичних математичних записів під час створення моделей.

Механізм трансформації дозволяє за допомогою низки правил перейти від узагальненої моделі до уточненої, або від моделі описаної у рамках математичних абстракцій (алгоритми, змінні, функції) до моделі описаної у рамках ООП. Формальна мова опису трансформації надає можливості до автоматизації процесу трансформації, та полегшення процесу створення користувачем правил трансформації.

Генерація каркасу системи (оголошення класів, методів, атрибутів, інтерфейсів і т. ін.), спрощує процес інтерпретації кінцевої моделі на обраній платформі. Завдяки трансформаційним зв'язкам, що утворюються під час генерації каркасу системи підвищується ефективність повторного використання коду, що був створений в ручну, під час реалізації кінцевої моделі. Це відбувається завдяки механізмам, що відповідають за «підняття» готового коду на вищі рівні моделювання, та подальше його розповсюдження підчас створення нових кінцевих моделей у рамках одного сімейства та платформи.

Таким чином, запропонований моделе-орієнтовний підхід надає широке коло можливостей для експериментування з різними алгоритмами та структурами даних під час розробки СІАД що засновані на БМД.

1. *Pearls J.* Bayesian inference methods // Encyclopedia of Artificial Intelligence, Sec. Ed. − New York: Wiley-Interscience Publication, 1992. – Vol. 1, A. P. 89–98.
2. *Верьовка О.В., Парасюк І.М., Карпінка Є.С.* Концептуальні основи Байєсівської діагностики у розмитому інформаційному просторі при дзвоноподібних функціях належності // Проблемы программирования. – 2004. – № 2-3. – С. 328–333.
3. *Cozman F. G.* Credal networks, Artificial Intelligence. − 2000 − vol. 120. − P. 199–233.
4. *Верьовка О.В., Парасюк І.М., Карпінка Є.С.* Концептуальні основи Байєсівської діагностики у розмитому інформаційному просторі при дзвоноподібних функціях належності // Проблемы программирования. – 2004. – № 2-3. – С. 328–333.
5. *Сергієнко І.В., Парасюк І.М., Ершов С.В.* Нечіткий трансформаційний підхід до розробки програмних систем // Проблемы программирования. – 2004. – № 2-3. – С. 122–132.
6. *Парасюк І.М., Ершов С.В., Алєксєєнко О.А.* Трансформаційний підхід типу «модель-модель» для реалізації Байєсівських механізмів інтелектуального аналізу даних // Проблемы программирования. – 2006. – № 2–3. – С. 511–518.
7. *Алексеенко А.А.* Трансформационный подход типа «модель-модель» к проектированию систем интеллектуального анализа данных // Компютерная математика. – 2007. − № 3 − С. 77-86.
8. *Алексеенко О.А., Ершов С.В.* Трансформация моделей систем интеллектуального анализа данных на основе Model-Driving Architecture // TAAPSD'2006
9. *Алексеенко О.А.* Мовні засоби для опису трансформацій типу «модель-модель» при побудові систем інтелектуального аналізу даних // TAAPSD 2007
10. *OMG*/MDA Guide V1.0.1. OMG, Document 2003-06-01, 12th June 2003. – http://www.omg.org
11. *OMG*/XMI 2.1 - http://www.omg.org
12. *OMG*/UML Superstructure Specification, v2.0, Document -- formal/05-07-04, August 2005 – http://www.omg.org
13. *OMG*/OCL 2.0 Specification, Document -- ptc/05-06-06, June 2005 - http://www.omg.org
14. *OMG*/MOF 2.1 - http://www.omg.org
15. W3C/Mathematical Markup Language (MathML) Version 2.0 - http://www.w3.org/TR/MathML2/
16. *Soft4science*/MathML .NET Control, February 2005 - http://www.soft4science.com/products/MathMLControl/s4s_MathMLControl.html
17. *Design* Science/MathFlow, February 2005 - http://www.dessci.com/en/products/mathflow/
18. *Alex* Billingsley/DragMath, November 2007 - http://www.dragmath.bham.ac.uk/