УДК 004.41

РОЗШИРЕННЯ UML СПЕЦИФІКАЦІЇ ДЛЯ МОДЕЛЮВАННЯ СЕМАНТИЧНИХ ОБ'ЄКТІВ

О.В. Новицький

В статті запропоновано відображення діаграми класів UML до дескриптивної логіки діалекту SHOIQ. Запропоновано розширення нотацій UML, шляхом використання стереотипів, для максимального наближення семантичних конструкцій. Вказано на причини та проблеми що виникають при такому відображенні.
Ключові слова: UML, Дескриптивна логіка, semantic web, валідація діаграм класів UML.

В статье предложено отображения диаграммы классов UML в дескриптивную логику диалекта SHOIQ. Предложено расширение нотаций UML путем использования стереотипов, для максимального приближения семантических конструкций. Указано на причины и проблемы, которые возникают при таком отражении.
Ключевые слова: UML, дескриптивная логика, semantic web, валидация диаграмм классов UML.

In the article, propose mapping from UML class diagrams to SHOIQ. descriptive logic. Made approach through extension the UML notations by using stereotypes as close to semantic structures. Specified on the causes and problems that arise in such mapping.
Key words: UML, description logick, semantic web, validation class diagram uml.

Вступ

Електронні бібліотеки (ЕБ) наразі функціонують у мережі Інтернет. Центральним об'єктом, що є носієм інформації в ЕБ – інформаційний ресурс. Існують різноманітні підходи до моделювання, управління і публікації інформаційних ресурсів в ЕБ. Значна частина підходів до управління інформаційними ресурсами пов'язана з Semantic Web. Наприклад, для публікації семантичних даних, широко розповсюдження набула ідеологія зв'язаних даних (LD) (Linked Data, 2015). Зв'язані дані дають можливість використання Інтернету для підключення відповідних даних, що раніше не були пов'язані між собою. Або більш конкретно, LD – це підхід, що використовується для опису методів виявлення спільного використання та підключення частин даних, пошуку інформації та знань в Semantic Web, використовуючи URIs, SPARQL і RDF (Linked Data, 2015) (Berners-Lee, 2009).

На даний момент, у дослідників не існує чіткого розуміння, що являє собою ЕБ в середовищі Semantic Web. Багато робіт, були присвячені даній проблемі. Зокрема, в (Sure and Studer, 2005) робиться більше акцент на побудову та використання онтологій для ЕБ. В (Alotaibi, 2010) розглядається розвиток ЕБ у напрямку семантичних соціальних ЕБ, в яких значна увага приділяється обміну знань та більш потужним механізмам взаємодії користувачів. Розробка моделі даних ЕБ, в рамках підходу LD, дасть змогу наблизити електронні бібліотеки до повноцінної реалізації Semantic Web.

Однак, коли виникає питання розробки програмного забезпечення, в тому числі ЕБ, одним із центральних аспектів є формалізація вимог до програмного продукту. У такій ситуації, широкого застосування набула UML (англ. Unified Modeling Language) – це уніфікована мова моделювання, що використовується у парадигмі об'єктно-орієнтованого програмування. UML є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. Вона є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, так званої UML-моделі. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей, як інтерпретованого коду, можлива кодогенерація.

UML є стандартизованим та формалізованим засобом для розробки та аналізу програмного забезпечення. Для підтримки розробки великих додатків існують складні CASE інструменти, що забезпечують зручні умови для редагування, зберігання і доступу до діаграм UML. Однак в таких інструментах відсутні засоби інтелектуального аналізу. Для UML характерні надлишковість та протиріччя, які важко виявити. Вважається, що UML діаграма класів, є одним з найбільш важливих компонентів UML. Для перевірки моделей, створених за допомогою діаграми класів UML, необхідно провести над цією моделлю логічні судження. В UML немає вбудованих засобів для її верифікації та валідації. Проте існують певні підходи вирішення цієї проблеми (Волович and Дерюгина, 2015). Використавши семантичний підхід і здійснивши відображення моделі UML до OWL, верифікація та валідація зводяться до перевірки на несуперечність, здійсниміcть, класифікацію та реалізацію концептів.

В даній роботі ми намагаємося розшити UML для більш повноцінного відображення UML в OWL та навпаки.

Огляд досліджень з проблематики верифікації UML

Існують різні методології верифікації UML моделей діаграм класів (Волович and Дерюгина, 2015). Так, найбільш простим є підхід, пов'язаний з побудовою драйвера (Макгрегор and Сайкс, 2002). Даний підхід дозволяє формально оцінити правильність створення діаграми класів і виявити найбільш

характерні помилки. Ще одним підходом до верифікації діаграми класів є метод шаблонів, який запропонований в (Sturm, Balaban and Maraee, 2010). Поняття шаблону, в даному підході, відрізняється від добре відомого і зрозумілого поняття шаблону проектування. У роботі було запропоновано декілька шаблонів помилок для виявлення протиріч в моделі. В роботі (Thalheim, 1993) розглянуто підхід на основі побудови ідентифікаційного графа. Суть даного методу полягає в побудові ідентифікаційного графа, що представляє собою орієнтований граф. Вузлами даного графа є класи, а також асоціативні зв'язки між ними, а дуги пов'язують асоціації з тими класами, між якими на діаграмі класів і вказані відповідні асоціативні зв'язки.

Представлення інформації про домен засобами OWL та UML

Мови OWL та UML були розроблені для різних цілей. OWL призначена для подання знань про інформаційну складову предметної області, UML розроблено перш за все для підтримки розробки програмного забезпечення. Хоча мови різні, але основна їх ціль є формальне представлення знань. В ранніх роботах (Cali et al., 202), (Cranefield and Purvis, 1999), (Brockmans et al., 2004) було висвітлено, яким чином можливо визначати онтологію засобами UML. В UML є кілька вбудованих засобів для формалізації семантики. Наприклад, Object Constraint Language (OCL) є декларативною мовою описання правил, що використовуються в UML, яка, на відміну від UML, має лінійну нотацію, а не графічну. На жаль так само, як і в UML в OCL, відсутня формальна модель семантики, теоретична модель та формальний доказ теорем. І тому, в UML не може бути механізмів для автоматизації міркувань.

Водночас, консорціум Object Management Group розробив метамодель визначення онтології (Ontology Definition Metamodel – ODM), що визначає набір мета-моделей UML (Object Management Group, 2009) і профілів для представлення UML в RDF, і OWL. У UML профілі в специфікації ODM адаптують нотації UML, щоб надати форму візуального представлення для RDF і OWL.

В роботах (Simmonds, 2003), (Daniela, Diego and Giuseppe, 2005) були здійснені спроби щодо трансформування діаграми класів UML до дескриптивної логіки, проте в цих роботах не знайшла місце повнота відображення.

Представлення знань та інформації про розробку програмного забезпечення, як правило, мають різні цілі при записі інформації. Робляться відповідні, але різні припущення щодо інтерпретації мовних висловів, чи заяв. Безліч припущень впливають на семантичну відповідність між мовними конструкторами і їх нотаціями. Якщо в OWL такі невідповідності можуть бути усунені, а вирази більш однозначні, то в UML вони більш різноманітні. UML дозволяє різні інтерпретації конструкцій мови в залежності від точки зору.

Є певні суттєві відмінності між UML та OWL:

- в діаграмі класів UML ми працюємо з припущенням про закритість світу. Тобто всі твердження, що не були явно вказані, є хибними. Натомість OWL використовує припущення відритого світу. Припущення ж про відкритість світу, в цьому випадку, припускає, що деяке твердження є ні істинним, ні хибним;

- UML має поняття профілів, що дозволяють розширювати мета-моделі елементів UML. Водночас OWL немає відповідної конструкції. У більшості випадків профілі UML використовуються для визначення стереотипів, щоб розширити класи. Представлення цих стереотипів може бути відображено до OWL через створення ряду нових класів та узагальнення тверджень. Однак більша частина профілю UML доволі специфічна і вимагає відповідні правила перетворення, адаптовані для певного профілю;

- абстрактні класи UML не можуть бути перетворені в OWL. Якщо клас визначається як абстрактний в UML, то це означає що екземпляри цього класу (об'єкти) не можуть бути створені. В OWL не закладено функцію, що вказувала б на те, що клас не повинен містити екземплярів. Одним із підходів до збереження семантики абстрактного класу є використання DisjointUnion. Це буде гарантувати, що будь-який екземпляр, який належить до підкласу, також відноситься до абстрактного суперкласу. Тим не менш, це не забороняє створювати екземпляри абстрактного суперкласу;

- в UML видимість елементів моделі може бути зменшена шляхом маркування їх як public, private, і т. д. Також можна оголосити елементи UML моделі, які доступні тільки для читання. В OWL відсутній механізм управління, щоб обмежити доступ до елементів моделі. OWL онтології також не можуть містити будь-яких операцій;

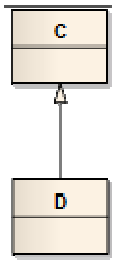
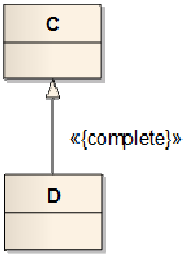
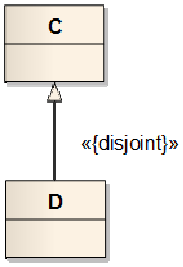
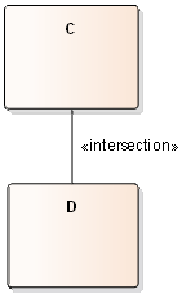
- в OWL можливо визначати властивості об'єкту на рівні онтології. І такі властивості не обов'язково повинні бути прив'язані до класу. Водночас, UML пропонує два способи прив'язки атрибутів. Через атрибут класу та через асоціації. Як випливає з назви, в першому випадку, атрибут класу відноситься до класу. Асоціації визначені на рівні пакету діаграми класів. Тим неменше, такі асоціації повинні мати на полюсах класи як типи. Тому UML асоціації не повністю підходять для представлення загальних властивостей об'єкту.

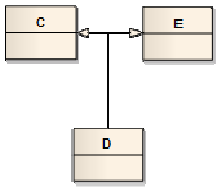
При моделюванні тверджень про об'єкти реального світу обидві мови використовують відповідні конструктори (наприклад, класи і об'єктів). OWL та UML виділяють відмінність між термінологічними (інтенсіональними) і екстенсіональними (твердженнями) знаннями.

Судження про UML діаграму класів використовуючи *SHOIQ*

В наслідок концептуальних розбіжностей між UML та OWL, не можливо побудувати однозначне відображення. В деяких роботах (Berardia, Calvanese and De Giacomo, 2005) є спроби розширити OWL до відповідного діалекту, що відповідав би UML. Ми в своєму підході намагаємося розшити UML через додаткові нотації та стереотипи, щоб збільшити повноту відображення. І це дозволить проектувати формальні специфікації у вигляді діаграми класів через онтології OWL. Ми робимо відображення основних конструкцій UML до дескриптивної логіки та відповідних конструкцій OWL. В UML клас A представляється за допомогою атомного концепту A (табл. 1).

Таблиця 1

UML	**Дескриптивна логіка**	**Абстрактний синтаксис OWL**
	A	Owl:Class. Атомний клас.
	$D \sqsubseteq C$	rdfs:subClassOf Фундаментальний конструктор, який оголошений в RDF Schema, визначає, що клас D є підкласом C.
	$D \equiv C$	owl:equivalentClass
	$D \sqsubseteq \neg C$	owl:disjointWith
	$C \sqcap D$	owl:intersectionOf

UML	**Дескриптивна логіка**	**Абстрактний синтаксис OWL**
	$D \sqsubseteq C \sqcup E$	D subClassOf unionOf C and E

Механізми розширення UML. Стереотипи

Надалі, для розвинення відображення, ми будемо використовувати механізми розширення. Механізми розширення – це вбудовані в мову способи розширити можливості мови. В UML включено багато методів так, щоб мова виявилася придатною в різних контекстах і предметних областях. Механізми розширення дозволяють визначати нові елементи моделі на основі існуючих.

Таких механізмів три:

- мічені значення;
- обмеження;
- стереотипи.

Ці механізми не є незалежними, вони тісно пов'язані між собою.

Мічене значення (tagged value) – це пара, ім'я властивості і значення властивості, яка може бути додана до будь-якого стандартного елементу моделі.

Обмеження (constraint) – це логічне твердження щодо значень властивостей елементів моделі.

Стереотип (stereotype) – це визначення нового елемента моделювання на основі існуючого елемента моделювання.

Визначення стереотипу проводиться таким чином. Взявши за основу деякий існуючий елемент моделі, до нього додають нові мічені значення (розширюючи тим самим внутрішнє представлення), нові обмеження (змінюючи семантику) і доповнення, тобто нові графічні елементи (визначаючи нотацію).

Генералізація

Узагальнення або генералізація – це відношення між двома сутностями, одна з яких є окремим випадком (спеціалізацією) іншої. Тобто це є направлене відношення між більш загальним класом і специфікованим класом. Генералізація в UML 2 має властивість наслідування, це означає, що клас який наслідує більш загальний клас, також наслідує його структуру та поведінку.

Узагальнення між класом C та його дочірнім класом C_1 може бути представлено за допомогою твердження включення в ALCQI, а саме у вигляді $C_1 \sqsubseteq C$. Ієрархія класів, як показано на рис. 1, може бути представлена твердженнями $C_1 \sqsubseteq C, \ldots C_n \sqsubseteq C$.

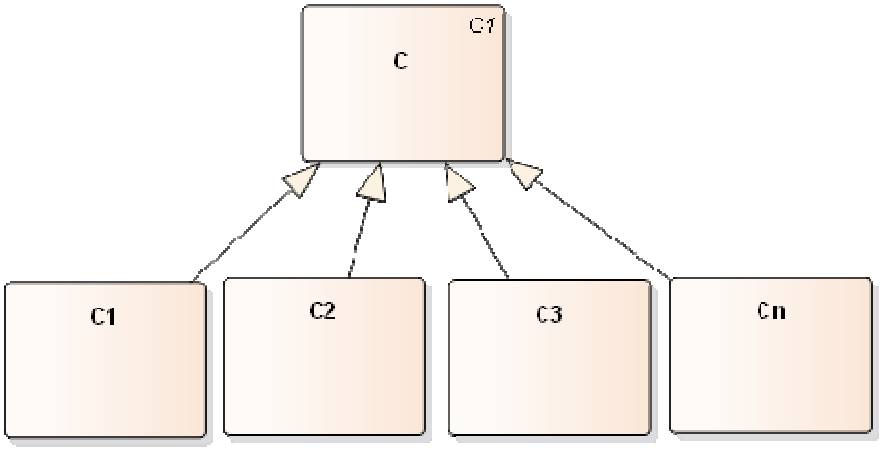


Рис. 1. Ієрархія класів

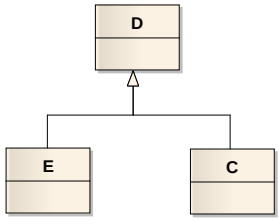
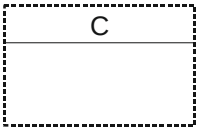
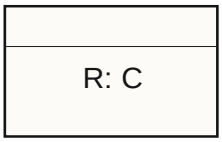
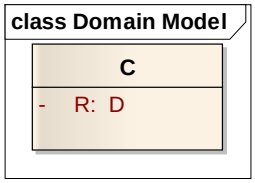
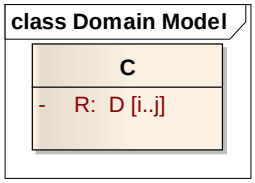
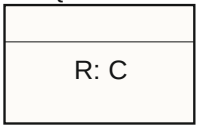
Кожна генералізація в специфікації UML 2 має дві основні властивості: це покриття, що може бути повним (**complete**) або неповним (**incomplete**); та властивість перетину, яка може бути з перетином (**overlapping**) або без перетину (**disjoint**).

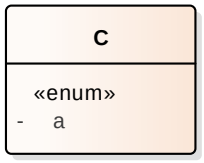
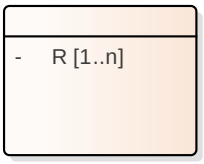
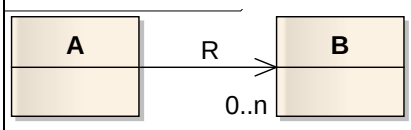
Повнота означає, що кожен екземпляр, більш загального класу, повинен належати принаймні до одного із специфікованих класів.

Перетин визначає, чи можуть специфіковані класи мати спільні екземпляри, тобто у випадку наявності хоча б одного спільного екземпляру говорять про генералізацію з перетином.

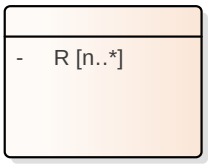
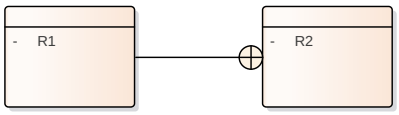
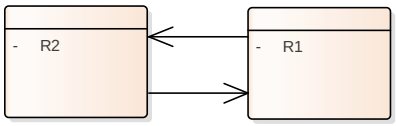
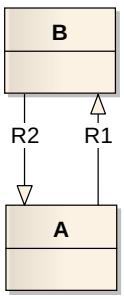
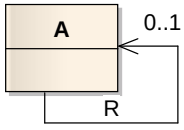
В UML 2.x кожна генералізація є неповною. Відмінність між UML 2.x та UML 2.5 полягає у відсутності та наявності перетину відповідно. Тому, якщо ми хочемо досягнути відсутності перетину, то обмеження класів $C_1 \ldots. Cn$, що не перетинаються, може бути змодельоване як $Ci \sqsubseteq \neg Cj$, де $1 \le i < j \le n$. Водночас як повнота узагальнення може бути виражена як $C \sqsubseteq C_1 \sqcup \ldots. \sqcup C_n$ (табл. 2).

Таблиця 2

UML	Дескриптивна логіка	Абстрактний синтаксис OWL
	$(E \sqcup C) \sqsubseteq D$	owl:unionOf
	$\neg C$	owl:complementOf(C)
	$\forall R.C$	owl:allValuesFrom
	$C \sqsubseteq \forall R.C$	C subClassOf Restriction on allValuesFrom
	Мультиплікативне відношення $[i..j]$ для ролі R класу формалізуються наступним чином $C \sqsubseteq (\ge iR.D) \sqcap (\le jR.D)$ Коли j рівно *, то ми відкидаємо другий кон'юкт. Коли мультиплікативне відношення має вигляд $[0...*]$, тоді опускають ціле твердження. Коли мультиплікативне відношення має вигляд $[1...1]$, то твердження матиме наступний вигляд $C \sqsubseteq \exists R. \top \sqcap (\le 1R)$.	owl:restriction(R maxCardinality(n)) owl:restriction(R minCardinality(n))
{existential restriction} 	$\exists R.C$	owl:restriction(R someValuesFrom(C))

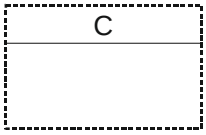
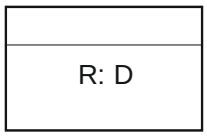
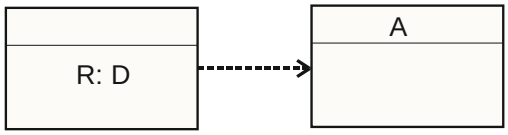
	$C(a)$	owl:hasValue
	$\leq n R$	restriction(R maxCardinality(n))
 Направлена асоціація	$A \sqsubseteq B \sqcap \leq n R$	A subClassOf Restriction on R maxCardinality N

Закінчення табл. 2

UML	Дескриптивна логіка	Абстрактний синтаксис OWL
	$\geq n R$	owl:restriction(R minCardinality(n))
	$R_1 \sqsubseteq R_2$	rdfs:SubPropertyOf(R1 R2)
	$R_1 \equiv R_2$	owl: EquivalentProperties(R1. R2)
	$\exists A.R_1 \equiv B.R^{-}_2$	owl:inverseOf
	$A \sqsubseteq A \leq 1 R$	owl:FunctionalProperty

Розглянемо більш детально деякі конструкції. Для досягнення більшої повноти відображення ми вводимо додаткові нотації в UML (табл. 3).

Таблиця 3

Нотація	Назва нотації	Опис нотації
	Заперечення існування класу	Нотація означає, що даний клас в предметній області, яка моделюється, існувати не може.
	Доповнення	Нотація позначає існування властивості певного типу.
	Залежність. Доповнення класу	Означає, що клас A має властивість R типу D.

Асоціації з асоціативним класом

Подібно до того, як об'єкти класу можуть бути записані за допомогою атрибутів, асоціативне відношення також може мати атрибути. UML дозволяє представляти інформацію такого характеру за допомогою класів асоціацій. Клас асоціації – це асоціація, яка одночасно є класом. Подібно зв'язкам асоціації, екземпляри класу асоціації мають індивідуальність, пов'язаної з тими об'єктами, між якими вони проводяться. Подібно звичайним класам, класи асоціацій можуть мати атрибути та операції, і брати участь в асоціаціях.

Для представлення мультиплікативного відношення A з асоціативним класом (рис. 2.), необхідно використати якісне обмеження кардинальності ролі (qualified number restrictions).

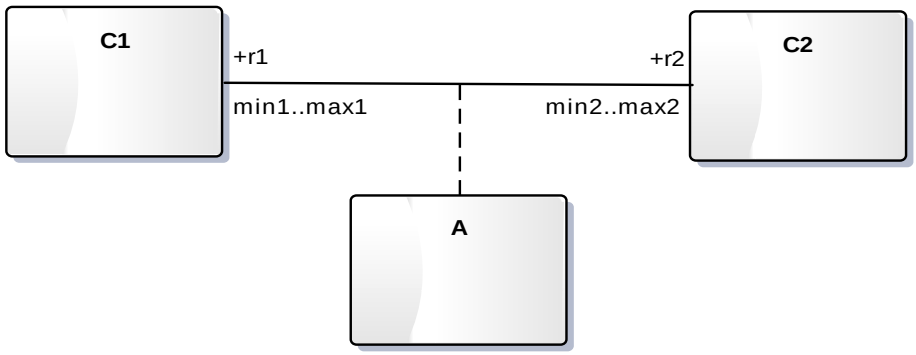


Рис. 2. Асоціації з асоціативним класом

Кожен екземпляр класу C_1 асоційований з екземплярами класу C_2, через асоціативний клас A. При цьому клас A може мати певні атрибути.

Відношення через асоціативний клас може мати n-арність на полюсах. В такому випадку, ми отримуємо відношення:

$$C_1 \sqsubseteq (\geq min_1 r_1^-.A) \sqcap (\leq max_1 r_1^-.A),$$

$$C_2 \sqsubseteq (\geq min_2 r_2^-.A) \sqcap (\leq max_2 r_2^-.A).$$

Розглянемо випадок, коли ми маємо асоціативне відношення між класами *A* та *D* через клас *C* з асоціацією *R* (рис. 3).

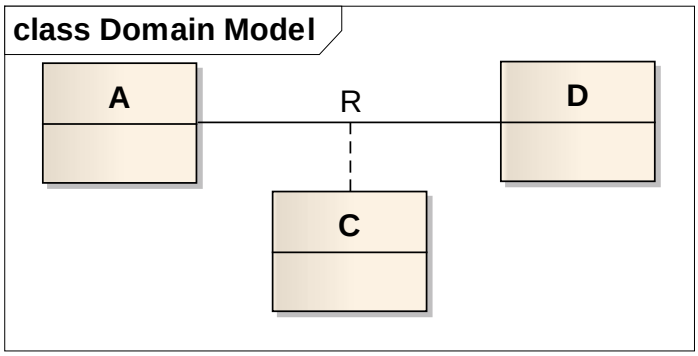


Рис. 3. Відношення з асоціацією R

В цьому випадку, щоб побудувати співставлення на ДЛ, є певні складності. Така діаграма класів UML є насправді окремим випадком n-арного відношення. В ДЛ SHOIQ немає конструкторів для моделювання n-арного відношення, тому вводять додатковий клас C' (рис. 4), який дає змогу будувати тільки бінарні відношення між додатковим класом та зв'язаними класами.

На ДЛ SHOIQ даний запис буде представлений наступним чином:

$$A \sqsubseteq \forall R1.C'$$
$$C' \equiv \exists R2.D \sqcup \forall R3.C$$
$$D \sqsubseteq \forall R2.C'.$$

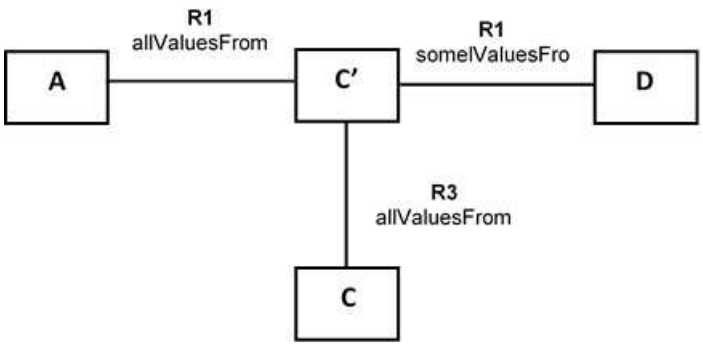


Рис. 4. Моделювання n-арного відношення

Асоціації в UML

Кожні бінарні асоціації (чи агрегації) між класами C_1 і C_2 мають вигляд (рис. 5).

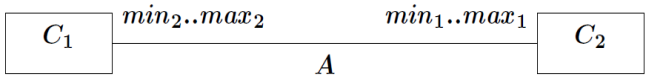


Рис. 5. Бінарна асоціація

Такі асоціації можуть бути трансформовані до твердження дескриптивної логіки через використання атомарної ролі *A* разом з твердженням включення:

$$\top \sqsubseteq \forall A.C_2 \sqcap \forall A^-.C_1.$$

Мультиплікативність асоціативного відношення *A* формалізується наступними твердженнями:

Кожний екземпляр концепту C_1 зв'язаний через атомну роль A з, принаймні, $\min_1$ і не більше $\max_1$ екземплярами концепту C2:

$$C1 \sqsubseteq (\geq \min 1A) \sqcap (\leq \max 1A).$$

Кожний екземпляр концепту C_2 зв'язаний через атомну роль A^- з, принаймні, $\min_2$ і не більше $\max_2$ екземплярами концепту C_1.

$$C_2 \sqsubseteq (\geq min_2 A^-) \sqcap (\leq max_2 A^-).$$

Слід відмітити, що агрегація в UML – це особливий вид бінарної асоціації без асоціативного класу.

Висновок

В роботі розглянуто проблематику верифікації діаграм класів UML за допомогою відображення до OWL. В даному підході також передбачена можливість зворотного відображення. Ця методика надає можливість верифікувати діаграму класів UML через механізми судження, що суттєво розвинуті в дескриптивній логіці. Представлене відображення не є однозначним, в наслідок того, що UML була з самого початку націлена на визначення формальних специфікації для розробки ПЗ з властивістю гнучкості та простотою нотацій. Так гнучкість та неоднозначність призвела до наслідків важкості автоматизованого аналізу діаграми класів UML. Зокрема, асоціація в UML описує набір кортежів, чиї значення відносяться до типу екземплярів. Іншими словами, асоціація специфікує семантичні відношення, які можуть виникнути між екземплярами певних типів. За замовчуванням, асоціація є ненаправленою, проте при нотаціях може бути використаний (OMG, 2010) символ %, який просто визначає напрямок читання і допомагає інтерпретувати асоціацію. І окремо виділяються направлені асоціації, в яких чітко вказується напрямок відношень між класами. Такі слабкі формальні відмінності не дозволяють побудувати однозначне та повне відображення.

1. *Alotaibi S.* 'The 4th Saudi International Conference' // Semantic Web Technologies for Digital Libraries: From Libraries to Social Semantic Digital Libraries (SSDL), Over Semantic Digital Libraries (SDL)., Manchester. 2010.
2. *Berardia, , Calvanese, D. and De Giacomo, G.* 'Reasoning on UML class diagrams', Artificial Intelligence. – 2005. – Vol. 168, N 1–2. P. 70–118.
3. *Berners-Lee, T.* (2009) *Linked Data*, [Online], Available: HYPERLINK "http://www.w3.org/DesignIssues/LinkedData.html" http://www.w3.org/DesignIssues/LinkedData.html [Jul 2015].
4. *Brockmans S., Volz R., Eberhart A. and Löffler.* 'Visual modeling of owl dl ontologies using uml', in The Semantic Web – ISWC 2004, Springer.
5. *Cali A., Calvanese D., De Giacomo G. and Lenzerini M.* 'A Formal Framework for Reasoning on UML Class Diagrams' // in Foundations of Intelligent Systems, Springer Berlin / Heidelberg. – 2002.
6. *Cranefield and Purvis* 'Uml as an ontology modelling language' // In Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99). – 1999.
7. *Daniela B., Diego C. and Giuseppe D.* 'Reasoning on UML class diagrams' // Artificial Intelligence. – 2005. – Vol. 168, N 1–2.
8. *Dereferencing HTTP URIs* (2007), Oct, [Online], Available: HYPERLINK "http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14" http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14.
9. *Fielding R.T.* (2005) *[httpRange-14] Resolved*, Jan, [Online], Available: HYPERLINK "http://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html" http://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html.
10. *Linked Data* (2015), Jul, [Online], Available: HYPERLINK "http://www.w3.org/standards/semanticweb/data" http://www.w3.org/standards/semanticweb/data.
11. *Object* Management Group (2009) Documents associated with Ontology Definition Metamodel (ODM) Version 1.0, May, [Online], Available: HYPERLINK "http://www.omg.org/spec/ODM/1.0/" http://www.omg.org/spec/ODM/1.0/ [Jul 2015].
12. *OMG* (2010) 'OMG Unified Modeling LanguageTM (OMG UML), Superstructure' Object Management Group, Available: http://www.omg.org/spec/UML/2.3/Superstructure.
13. *Simmonds J.* (2003) Consistency Maintenance of UML Models with Description Logics, Brussel.
14. *Sturm A., Balaban M. and Maraee A.* 'Management of Correctness Problems in UML Class Diagrams Towards a Pattern-Based Approach', International Journal of Information System Modeling and Design. – 2010. – Vol. 1, N 4. – P. 24–47.
15. *Sure Y. and Studer.* 'Semantic Web technologies for digital libraries', Library Management. – 2005. – Vol. 4/5, N 26. – P. 190–195.
16. *Thalheim B.* 'Foundations of entity-relationship modeling', Annals of Mathematics and Artificial Intelligence. – 1993. – Vol. 7, N 1–4, Available: 10.1007/BF01556354.
17. *Волович М.Е., Дерюгина, О.А.* 'Верификация UML программных систем', Cloud of Science. – 2015. – Vol. 2, № 1, Available: 2409-031X.
18. *Макгрегор Д., Сайкс Д.* Тестирование объектно-ориентированного программного обеспечения. – 2002, ТИД "ДС".

References

1. Alotaibi, S. (2010) 'The 4th Saudi International Conference', Semantic Web Technologies for Digital Libraries: From Libraries to Social Semantic Digital Libraries (SSDL), Over Semantic Digital Libraries (SDL)., Manchester.
2. Berardia, , Calvanese, D. and De Giacomo, G. (2005) 'Reasoning on UML class diagrams', Artificial Intelligence, Vol. 168, N 1–2, P. 70–118.
3. Berners-Lee, T. (2009) Linked Data, [Online], Available: HYPERLINK "http://www.w3.org/DesignIssues/LinkedData.html" http://www.w3.org/DesignIssues/LinkedData.html [Jul 2015].
4. Brockmans, S., Volz, R., Eberhart, A. and Löffler, (2004) 'Visual modeling of owl dl ontologies using uml', in The Semantic Web – ISWC 2004, Springer.
5. Cali, A., Calvanese, D., De Giacomo, G. and Lenzerini, M. (202) 'A Formal Framework for Reasoning on UML Class Diagrams', in Foundations of Intelligent Systems, Springer Berlin / Heidelberg.
6. Cranefield , and Purvis, (1999) 'Uml as an ontology modelling language', in In Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99.
7. Daniela, B., Diego, C. and Giuseppe, D. (2005) 'Reasoning on UML class diagrams', Artificial Intelligence, Vol. 168, N 1–2.
8. Dereferencing HTTP URIs (2007), Oct, [Online], Available: HYPERLINK "http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14" http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14 .
9. Fielding, R.T. (2005) [httpRange-14] Resolved, Jan, [Online], Available: HYPERLINK "http://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html" http://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html .
10. Linked Data (2015), Jul, [Online], Available: HYPERLINK "http://www.w3.org/standards/semanticweb/data" http://www.w3.org/standards/semanticweb/data .
11. Object Management Group (2009) Documents associated with Ontology Definition Metamodel (ODM) Version 1.0, May, [Online], Available: HYPERLINK "http://www.omg.org/spec/ODM/1.0/" http://www.omg.org/spec/ODM/1.0/ [Jul 2015].

12. OMG (2010) 'OMG Unified Modeling LanguageTM (OMG UML), Superstructure' Object Management Group, Available: http://www.omg.org/spec/UML/2.3/Superstructure.
13. Simmonds, J. (2003) Consistency Maintenance of UML Models with Description Logics, Brussel.
14. Sturm, A., Balaban, M. and Maraee, A. (2010) 'Management of Correctness Problems in UML Class Diagrams Towards a Pattern-Based Approach', International Journal of Information System Modeling and Design, Vol. 1, N 4, P. 24–47.
15. Sure, Y. and Studer, (2005) 'Semantic Web technologies for digital libraries', Library Management, Vol. 4/5, N 26, P. 190–195.
16. Thalheim, B. (1993) 'Foundations of entity-relationship modeling', Annals of Mathematics and Artificial Intelligence, Vol. 7, N 1–4, Available: 10.1007/BF01556354.
17. Volovich M.E. and Deryugina O.A. (2015) 'Verification of UML software systems ', Cloud of Science, Vol. 2, N 1, Available: 2409-031X.
18. Makgregor D. and Sayks D. (2002) Testing Object-Oriented Software, TID "DS".

Про автора:

Новицький Олександр Вадимович,
молодший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 15.
Кількість наукових публікацій в іноземних виданнях – 4.
Індекс Гірша – 4.
http://orcid.org/0000-0002-9955-7882.

Місце роботи автора:

Інститут програмних систем НАН України,
03181, Київ-187, проспект Академика Глушкова, 40.
Тел.: (067) 44 5 173.
E-mail: alex.googl@gmail.com