https://doi.org/10.31891/2219-9365-2024-79-5
УДК 004

КРУПА Дмитро
Національний університет «Львівська політехніка»
https://orcid.org/0009-0002-6087-9988
e-mail: dima-krupa@ukr.net

ОЦІНКА СУМІСНОСТІ ОКРЕМИХ КОМПОНЕНТ ВЕБ-СЕРВЕРІВ З ВИКОРИСТАННЯМ ОНТОЛОГІЧНОГО ПІДХОДУ ТА НЕЧІТКОГО ЛОГІЧНОГО ВИВЕДЕННЯ

У статті описаний метод перевірки сумісності окремих компонент зі використанням онтологічного підходу та нечіткого логічного виведення. Запропонована структура методу оцінки сумісності об'єктів між собою. Наведений приклад використання онтологічного підходу до оцінки сумісності компонент. Запропоновані нечіткі змінні та модифіковано метод нечіткого логічного виведення. Наведений приклад роботи методу.

Ключові слова: сумісність компонент, онтологічний підхід, нечітке логічне виведення

KRUPA Dmytro
Lviv Polytechnic National University

EVALUATION OF THE COMPATIBILITY OF INDIVIDUAL COMPONENTS OF WEB SERVERS USNING THE ONTOLOGICAL APPROACH AND FUZZY LOGICAL INFERENCE

System compatibility is significant problem in software development, especially when systems can be changed independently. To check their compatibility, their application programming interfaces (APIs) should be compatible. But not only APIs ensure compatibility, software developers also should be sure that their apps fit logically. This process can be done using manual process of checking systems and their interfaces or by covering both apps with integration and API tests. Additionally, expert should proof that systems are compatible and their OOP structure fits to both business models.

In this article, system compatibility problem was raised and described. Was suggested method to check object compatibility. Object compatibility is initial step to check interface compatibility. Also, ontology approach is described and suggested how it can be used in method. In suggested method ontologies can be used by experts or ontology reasoning checks can be executed. Ontology assessment was divided into structural and logical parts. Where structural part can be done by automated check to ensure that requested object is compatible with ontology concept. Logical part of assessment can be done by using ontology reasoning or expert interaction.

Was suggested structure of base object for representation each instance that should be checked. This is initial step for creating compatibility checking process and/or framework,

Beside this, fuzzy logical inference was modified to fits suggested methods. Some base rules were given to explicitly show how logical deduction works in this case. Also, expert's intervention in the process has more impact than automatic check to decrease possibility of mistake.

Also, simple example was processed using suggested method to presents how it works. In this example was checked compatibility between objects "Room" and "Cabinet". Was suggested that they are for different systems and checks if these objects are compatible.

Keywords: component compatibility, ontology approach, fuzzy logical inference.

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

При розробці комплексних рішень для бізнесу часто виникає потреба інтеграції із іншими системами для спрощення виконання завдань, які обумовлюються бізнес вимогами. Наприклад, відправка емейлів, збереження файлів, редагування відео, тощо. Дані сервіси повністю вирішують поставлену задачу, а системі, яка з ними інтегрується, достатньо лише надати дані для опрацювання.

Окремо можемо виділити вимоги, що потребують інтеграції з іншими системами, які оперують тими ж самими або схожими поняттями. Для спрощення розуміння проблеми у статті буде розглянуто серверні технології, зокрема інтеграція двох систем на основі серверів, бізнес яких працює в одній області та оперують тотожними або схожими поняттями.

Із наведених прикладів такі поняття як «емейл», «файл», «відео» є загальновідомими, тому при їх використанні не потрібні спеціальні знання у даній області. Але виникає проблема, коли дві системи інтегруються між собою та оперують складними, нетривіальними або неоднозначними поняттями як, наприклад, «замовлення товару» чи «доставка». В одній системі поняття «доставка» може бути визначено як адресна доставка до покупця, а в іншій – завантаження товару у фургон перевізника на складі.

Якщо говорити про сумісність компонент для серверних технологій, то проблема зводиться до сумісності між запитами (а саме об'єктами), що передаються між двома системами. Вони повинні не лише мати необхідні дані, але і логічно (з точки зору бізнесу) підходити для виконання завдання.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Зазвичай перевірка сумісності веб-серверів є складним та затратним процесом. Для перевірки сумісності компонент можна використовувати інтеграційній тести [8], але вони будуть ефективні лише при незмінності однієї із компонент.

Проте виникає проблема, коли системи, що були інтегровані одна в іншу, починаються змінюватися і їхня інтеграція перестає працювати коректно. Важливим завданням є не лише перевірка факту сумісності двох веб-серверів, але і надання максимальної інформації про їхню взаємодію та що саме потрібно змінити, щоб система в цілому почала функціонувати як було заплановано.

При представленні знань про домену область кожної із компонент у вигляді онтологій варто зазначити що онтологія, як і сама система, теж може та має еволюціонувати щоб завжди чітко відповідати своїй сфері застосування [3].

Проблема сумісності компонент та пропозиції її вирішення була розглянуті у роботах [4–7]. Ці методи дозволяють оцінити та забезпечити сумісність серверів між собою. Але не завжди є можливість створити програмне рішення яке повністю покриє усі можливі проблеми із сумісністю компонентів зважаючи на обмеженість ресурсів (проблема «залізного трикутника») [9]. Тому потрібно розробити просте та гнучке рішення, яке буде допомагати розробникам забезпечувати сумісність систем між собою.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є розробити метод визначення ступеня сумісності між класами, використовуючи нечітке виведення та онтологічний підхід.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

У даній статті будемо користуватися загальновідомими термінами ООП: клас, об’єкт, поле [10]. Також система, з якою планується інтеграція, матиме об’єкти, які ми вважатимемо еталонними, бо саме з ними будемо порівнювати власні об’єкти. Важливим є наявність онтології, яка формалізовано відображає структуру системи з якою відбувається інтеграція.

Створимо структуру методу, що буде надавати оцінку сумісності класів, зважаючи на їхню роль в бізнес-домені.

Структура методу

- Потрібно представити два об’єкти як множину полів з метаданими
- Визначити поля, що є обов’язковими для порівняння
- Додати обмеження по типу та значеннях
- Використання онтологічного підходу для оцінки сумісності класів
- Використати нечітке виведення для перевірки відповідності об’єкту еталонному
- Надати оцінку сумісності класів

Представлення об’єктів як множину полів з метаданими

Для початку представимо поле об’єкту як новий об’єкт із метаданими.

Розглянемо клас як множину характеристик, а об’єкт – як набір понять «ключ-значення». Також створимо універсальний об’єкт (тут і далі будемо його називати MetaObject), який буде інкапсулювати метадані поля для подальшого порівняння. Такий об’єкт може містити дані: назва поля, тип поля, список допустимих значень, список обмежень, шлях у ієрархії до найабстрактнішого класу (наприклад, у Java – це клас Object).

Для порівняння сумісності(відповідності) двох полів потрібно представити їх у форматі MetaObject та використати функцію, яка даватиме числове представлення сумісності двох полів.

Function checkCompatibility(x,y) = [0;1],
де x та y – екземпляри класу MetaObject, [0;1] – межі результату обчислення.

Реалізація функції checkCompatability може змінюватися в залежності від вимог проекту.
Структура MetaObject у форматі псевдокоду:

```
Об'єкт MetaObject {
  Поле: {
    Ім'я: "Name",
    Тип: "string"
  },
  Поле: {
    Ім'я: "Type",
    Тип: "string"
```

```
    },
    Поле: {
      Ім'я: "allowedValues",
      Тип: "Array"
    },
    Поле: {
      Ім'я: "restrictions",
      Тип: "Array"
    },
    Поле: {
      Ім'я: "hierarchy",
      Тип: "String"
    }
}
```

1. Використання онтологічного підходу

Для того, щоб визначити чи відповідає порівнюваний клас предметній області, необхідно перевірити його відповідність онтології, зокрема як він відповідає еталонному класу. У цьому кроці необхідна оцінка експерта, що порівняє запитуваний клас із відповідним йому в онтології [2].

На цьому етапі нас цікавлять дві оцінки: оцінка відповідності структурі та оцінка логічної відповідності.

Оцінку відповідності структурі можна отримати програмним шляхом: чи клас відповідає концепту, має усі його властивості та дотримується його аксіом. Наприклад, *Кімната* $\sqsubseteq$ *∃має.Вікно*.

Оцінку логічної відповідності може надати експерт, або спробувати співставити назву та властивості об'єкта із концептом, використовуючи тезаурус з конкретної предметної області. Наприклад, "surname" можна замінити словом "lastname".

2. Оцінка сумісності класів

Для роботи із заданими параметрами використаємо визначення нечіткої змінної, що складається із трійки параметрів <a, X, A> [1], де

a – ім'я змінної

X – універсальна множина

A – нечітка підмножина множини X, що описує обмеження на значення змінної a (μ_A(x)))

Таким чином у нас сформовано три нечіткі змінні:

<*оцінка основних властивостей*, [0;1], (μ_{A1} (X)) >

<*оцінка додаткових властивостей*, [0;1], (μ_{A2} (X))>

<*оцінка відповідності онтології*, [0;1], (μ_{A3} (X))>

Наступним кроком створимо правило, яке на основі оцінок визначає ступінь логічної відповідності. Оскільки ми працюємо із нечіткими даними, використаймо нечітке логічне виведення.

Для спрощення розглянемо еталонний об'єкт E та порівнюваний об'єкт A.

$\mu_{A1}(X)$=∑i *checkRequiredCompatibility*(xi, yi)

$\mu_{A2}(X)$=∑i *checkOptionalCompatibility*(xi, yi)

Логічна оцінка = *min*(оцінка експерта, програмна оцінка)

Структурна оцінка = *min*(оцінка експерта, програмна оцінка)

*μA3(X) = 0.65 * логічна оцінка + 0.35 * структурна оцінка*

Коефіцієнти 0.65 вказує на перевагу логічній відповідності, яка надається експертом у галузі, для якої розроблена онтологія, а 0.35 – на важливість відповідності структурі онтології, що необхідно для розробки програмного рішення.

1. Фазицікація

А) *оцінка основних властивостей*

Функція приналежності:

- НеВідповідає (x) = max(0, min(1, 1- x));
- ЧастковоВідповідає (x) = 1 - |2x – 1|
- Відповідає (x) = max(0, min (x, 1);

Б) *оцінка додаткових властивостей*

Функція приналежності:

- НеВідповідає (x) = max(0, min(1, 1- x));
- ЧастковоВідповідає (x) = 1 - |2x – 1|
- Відповідає (x) = max(0, min (x, 1);

В) оцінка відповідності онтології

Оцінка відповідності онтології може як надаватися експертом у нечіткому вигляді і не потребувати

фазицікації, так і може бути у вигляді чіткого числа, що необхідно фазифікувати.

Функція приналежності:

- НеВідповідає (x) = max(0, min(1, 1- x));
- ЧастковоВідповідає (x) = 1 - |2x – 1|
- Відповідає (x) = max(0, min (x, 1);

2. Пошук логічного висновку

Сформуємо правила формату «ЯКЩО … ТО …» для пошуку логічного висновку [2].

I. Якщо оцінка основних властивостей відповідає І оцінка додаткових властивостей відповідає І оцінка відповідності онтології відповідає ТО системи повністю сумісні

II. Якщо оцінка основних властивостей *частково відповідає* І оцінка додаткових властивостей *відповідає* І оцінка відповідності онтології *не відповідає* ТО системи *не сумісні*

III. Якщо оцінка основних властивостей *відповідає* І оцінка додаткових властивостей *не відповідає* І оцінка відповідності онтології *не відповідає* ТО системи *потребують додаткової сумісності*

IV. Якщо оцінка основних властивостей *частково відповідає* І оцінка додаткових властивостей *не* відповідає І оцінка відповідності онтології *не відповідає* ТО системи *потребують додаткової сумісності*

V. Якщо оцінка основних властивостей *відповідає* І оцінка додаткових властивостей *не відповідає* І оцінка відповідності онтології *відповідає* ТО системи *сумісні*

Кількість та умови правил можуть змінюватися в залежності від вимог системи

3. Дефазифікація

Призначимо числові ваги вихідним термінам:

I. повністю сумісні = 1

II. не сумісні = 0

III. потребують додаткової сумісності = 0.5

Якщо ж активуються декілька правил, то будемо використовувати формулу: *Fazz = min (rule1, rule 2, rule 3, rule 4, rule 5).*

Причина полягає в тому, що якщо виникає сумнів в інтеграції систем – необхідно приділити цьому увагу та вирішити імовірну проблему.

3. Розглянемо роботу методу на прикладі

Як приклад розглянемо вищенаведений клас «Кімната» та чи буде він сумісним із класом «Кабінет».

Розглянемо клас «Кімната», як один із популярних прикладів, які використовуються для вивчення ООП. Характеристики (поля) «Кімнати» {«ширина», «довжина», «висота», «кількість вікон», «колір стін»}.

Перетворимо об’єкт класу «Кімната» та список об’єктів типу MetaObject.

```
{
    "Name": “ширина”
    "Type": “Number”,
    "allowedValues": [“x > 0”],
    "restrictions": [],
    "hierarchy": “Number -> Object”
}
```

Аналогічні об’єкти будуть розроблені для полів «довжина» та «висота». «Кількість вікон» буде відрізнятися типом даних: цілий тип (Integer) замість Number.

«Колір стін» можна перетворити у такий об’єкт:

```
{
    "Name": “колір стін”
    "Type": “String”,
    "allowedValues": [“білий”, “чорний”, “зелений”, “синій” … ],
    "restrictions": [],
    "hierarchy": “String -> Object”
}
```

Обов’язковими полями є: “ширина”, “довжина”, “висота” та “колір стін”. Додатковими: «кількість вікон».

Клас «Кабінет» має поля: «ширина», «довжина», «висота», «робочий стіл», «колір стін»}. Аналогічно перетворюємо його поля.

Онтологічний підхід:
Припустімо, що експертна оцінка «так, класи сумісні». Тобто,
Логічна оцінка = 1
Перевіримо виконання аксіом з концепту “Кімната” для об’єкту “Кабінет”.

- *Кімната⊑∃маєКолірСтін.Колір = true = 1*
- *Кімната⊑∃маєШирину.Ширина = true = 1*
- *Кімната⊑∃маєДовжину.Довжина = true = 1*
- *Кімната⊑∃маєВисоту.Висота = true = 1*
- *Кімната⊑∃маєВікно.Вікно = false = 0*

Для даного прикладу обчислимо структурну оцінку як середнє значення аксіом:
Структурна оцінка = 0.8

$\mu_{A1}(X)=\sum_i$ checkRequiredCompatibility(x_i, y_i) = 1
$\mu_{A2(}X)=\sum_i$ checkOptionalCompatibility(x_i, y_i) = 0
$\mu_{A3}(X)$ = 0.65 * 1 + 0.35 * 0.8 = 0.93

Результат фазифікації:
оцінка основних властивостей = Відповідає
оцінка додаткових властивостей = Не відповідає
оцінка відповідності онтології = Відповідає

Для даного прикладу застосовується правило №5 -> Системи сумісні між собою.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Розроблений метод дозволяє порівнювати об’єкти з однієї системи з бізнес-логікою іншої. Він надає універсальний інструмент перевірки сумісності об’єктів з використанням онтологічного підходу та з аналізом метаданих кожної властивості об’єкту.

Також було модифіковано структуру нечіткого логічного виведення, що повинно зменшити імовірність похибки завдяки використанню онтологій.

Окрім того, було окреслено важливість залучення експертів як із сторони бізнесу, так і експертів з розробки програмних рішень для модифікації функцій та правил нечіткого виведення з метою відповідності потребам проекту.

Підхід, розглянутий у даній статті, є початковим кроком для розробки гнучкого та не ресурсозатратного рішення для перевірки сумісності систем що активно розробляються.

Література

1. Желдак Т. А. Нечіткі множини в системах управління та прийняття рішень / Т. А. Желдак, Л. С. Коряшкіна, С. А. Ус. – Дніпро : НТУ «Дніпровська політехніка», 2020. – 387 с.

2. Cingolani P. jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming / P. Cingolani, J. Alcal. // International Journal of Computational Intelligence Systems. – 2013.

3. Басюк Т. М. Онтологічний інжинірінг / Т. М. Басюк, В. В. Литвин, Д. Г. Досин. – Львів, 2016. – 264 с.

4. Liang Q. Ontology-Based Compatibility Checking for Web Service Configuration Management / Q. Liang, M. N. Huhns. – P. 15.

5. Craig D. C. Verification of Component Behavioral Compatibility / D. C. Craig, W. Zuberek. – 2007. – URL : https://www.researchgate.net/publication/4261561_Verification_of_Component_Behavioral_Compatibility

6. Pahl C. An ontology for software component matching / Claus Pahl // International Journal on Software Tools for Technology Transfer. – 2007.

7. Medjahed B. Context-based matching for Web service composition / B. Medjahed, Y. Atif // Distributed and Parallel Databases. – 2007.

8. Chan W. An Overview of Integration Testing Techniques for Object-Oriented Programs / W. Chan, T. Chen, T. Tse. – 2002.

9. Stojcetovic B. Project managament: cost, time and quality / Bojan Stojcetovic. – 2013. – URL : https://www.researchgate.net/publication/305462896_Project_managament_cost_time_and_quality.

10. Object-Oriented Programming in Computer Science / R. Yilmaz, A. Sezgin, S. Kurnaz, Y. Arslan // Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics. 2019. – С. 1439–1451.

ISSN 2219-9365

References

1. Zheldak T. A. Nechitki mnozhyny v systemakh upravlinnia ta pryiniattia rishen / T. A. Zheldak, L. S. Koriashkina, S. A. Us. – Dnipro: NTU «Dniprovska politekhnika», 2020. – 387 p.

2. Cingolani P. jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming / P. Cingolani, J. Alcal. // International Journal of Computational Intelligence Systems. – 2013.

3. Basiuk T. M. Ontolohichnyi inzhynirinh / T. M. Basiuk, V. V. Lytvyn, D. H. Dosyn. – Lviv, 2016. – 264 p.

4. Liang Q. Ontology-Based Compatibility Checking for Web Service Configuration Management / Q. Liang, M. N. Huhns. – C. 15.

5. Craig D. C. Verification of Component Behavioral Compatibility [Electronic resource] / D. C. Craig, W. Zuberek. – 2007. – Resource access mode: https://www.researchgate.net/publication/4261561_Verification_of_Component_Behavioral_Compatibility

6. Pahl C. An ontology for software component matching / Claus Pahl. // International Journal on Software Tools for Technology Transfer. – 2007.

7. Medjahed B. Context-based matching for Web service composition / B. Medjahed, Y. Atif. // Distributed and Parallel Databases. – 2007.

8. Chan W. An Overview of Integration Testing Techniques for Object-Oriented Programs / W. Chan, T. Chen, T. Tse. – 2002.

9. Stojcetovic B. Project managament: cost, time and quality [Electronic resource] / Bojan Stojcetovic. – 2013. – – Resource access mode: https://www.researchgate.net/publication/305462896_Project_managament_cost_time_and_quality.

10. Object-Oriented Programming in Computer Science / R.Yilmaz, A. Sezgin, S. Kurnaz, Y. Arslan // Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics / R.Yilmaz, A. Sezgin, S. Kurnaz, Y. Arslan., 2019. – p. 1439–1451.