# Tworzenie diagramów klas UML w oparciu o ontologie dziedzinowe OWL 2

Małgorzata Sadowska

*Promotor*: prof. dr hab. inż. Zbigniew Huzar

*Promotor pomocniczy:* dr inż. Bogumiła Hnatkowska

# Plan prezentacji

I. Zakres rozprawy

II. Wprowadzenie
   - UML i OWL 2 - podobieństwa i różnice
   - Transformacje OWL 2 ↔ UML
   - Normalizacja ontologii OWL 2

III. Tworzenie diagramów klas UML w oparciu o ontologie OWL 2

IV. Modyfikacja diagramów klas

V. Podsumowanie

# I. ZAKRES ROZPRAWY

# Przedmiot rozprawy

**Tytuł rozprawy**

Tworzenie i walidacja diagramów klas UML
z wykorzystaniem ontologii dziedzinowych w OWL 2

**Teza rozprawy**

Wykorzystanie ontologii dziedzinowych sprzyja szybszemu tworzeniu modeli biznesowych i podnosi ich jakość semantyczną

# Cele rozprawy

1. Opracowanie metody wydobywania (fragmentów) diagramów klas UML z ontologii wyrażonych w języku OWL 2
2. Opracowanie metody automatycznej weryfikacji diagramów klas UML względem ontologii wyrażonych w języku OWL 2, w celu usprawnienia walidacji diagramów
3. Opracowanie narzędzia, które implementuje proponowane metody

# Metodyka badań

1. Systematyczny przegląd literatury w zakresie reguł transformacji pomiędzy elementami diagramów klas UML oraz aksjomatami wyrażonymi w języku OWL 2
2. Eksperyment mający na celu empiryczną ocenę opracowanego narzędzia do tworzenia i walidacji diagramów klas UML
3. Analiza statystyczna uzyskanych wyników eksperymentu

# II. WPROWADZENIE

# UML i OWL 2 - podobieństwa i różnice

Podobieństwa

- semantyka
- podobne konstrukcje

  np. klasa, instancja

  generalizacja / SubClassOf

  liczność (Multiplicity/Cardinality)

  typ wyliczeniowy (enumeracja / DatatypeDefinition + DataOneOf)

  itp.

OWL 2:

- Open-World Assumption
- No Unique Name Assumption

UML:

- Closed-World Assumption
- Unique Name Assumption

- różne konstrukcje

  np. EquivalentClasses,

  klasa abstrakcyjna, operacje, itp.

# Transformacje OWL 2 ↔ UML

*Tworzenie diagramu w oparciu o ontologię*

OWL → UML

$$\frac{Ax_{e:E}}{e:E} \quad Cr_{e:E}$$

*Weryfikacja diagramu względem ontologii*

UML → OWL

$$\frac{e:E}{Ax_{e:E}} \quad Vr_{e:E}$$

$E$ - kategoria elementu diagramu klas UML

$e$ - pojedynczy element diagramu klas UML, $e:E$

$Ax_{e:E}$ - zbiór aksjomatów na który przekształcany jest pojedynczy element $e:E$ poprzez reguły transformacji

$Vr_{e:E}$ - zbiór reguł weryfikacji dla elementu UML $e:E$

$Cr_{e:E}$ - zbiór reguł sprawdzających dla elementu UML $e:E$

$Cr_{e:E} \subseteq Vr_{e:E}$

# Normalizacja ontologii

Tabela: Normalizacja dot. aksjomatów wyrażeń klasowych

| ID | Aksjomat zastępowany | Aksjomat(y) zastępujący(-e) |
|---|---|---|
| 1 | EquivalentClasses( $CE_1 \dots CE_i \dots CE_j \dots CE_N$) and $1 \le i \le j \le N$ and $N \ge 3$ and $CE_i = CE_j$ | EquivalentClasses( $CE_1 \dots CE_i \dots CE_N$) and $1 \le i \le N$ and $N \ge 2$ |
| 2 | EquivalentClasses( $CE_1 \dots CE_N$) and $1 \le i \le N$ and $N \ge 2$ | EquivalentClasses ( $CE_i$ $CE_j$ ) and $i,j \in \{1,N\}$ and $i \ne j$ and $N \ge 2$ |
| 3 | EquivalentClasses( $CE_1$ $CE_2$ ) | SubClassOf( $CE_1$ $CE_2$ ) SubClassOf( $CE_2$ $CE_1$ ) |
| 4 | DisjointClasses( $CE_1 \dots CE_i \dots CE_j \dots CE_N$) and $1 \le i \le j \le N$ and $N \ge 3$ and $CE_i = CE_j$ | DisjointClasses( $CE_1 \dots CE_i \dots CE_N$) and $1 \le i \le N$ and $N \ge 2$ |
| 5 | DisjointClasses( $CE_1 \dots CE_N$ ) and $N \ge 2$ | DisjointClasses( $CE_i$ $CE_j$ ) and $i,j \in \{1,N\}$ and $i \ne j$ and $N \ge 2$ |
| 6 | DisjointClasses( $CE_1$ $CE_2$ ) | SubClassOf( $CE_1$ ObjectComplementOf( $CE_2$ ) ) SubClassOf( $CE_2$ ObjectComplementOf( $CE_1$ ) ) |
| 7 | DisjointUnion( C $CE_1 \dots CE_i \dots CE_j \dots CE_N$) and $1 \le i \le j \le N$ and $N \ge 3$ and $CE_i = CE_j$ | DisjointUnion( C $CE_1 \dots CE_i \dots CE_N$) and $1 \le i \le N$ and $N \ge 2$ |
| 8 | DisjointUnion( C $CE_1 \dots CE_N$ ) and $N \ge 2$ | EquivalentClasses( C ObjectUnionOf ( $CE_1 \dots CE_N$ ) ) DisjointClasses( $CE_1 \dots CE_N$ ) and $N \ge 2$ |

# III. TWORZENIE DIAGRAMÓW KLAS

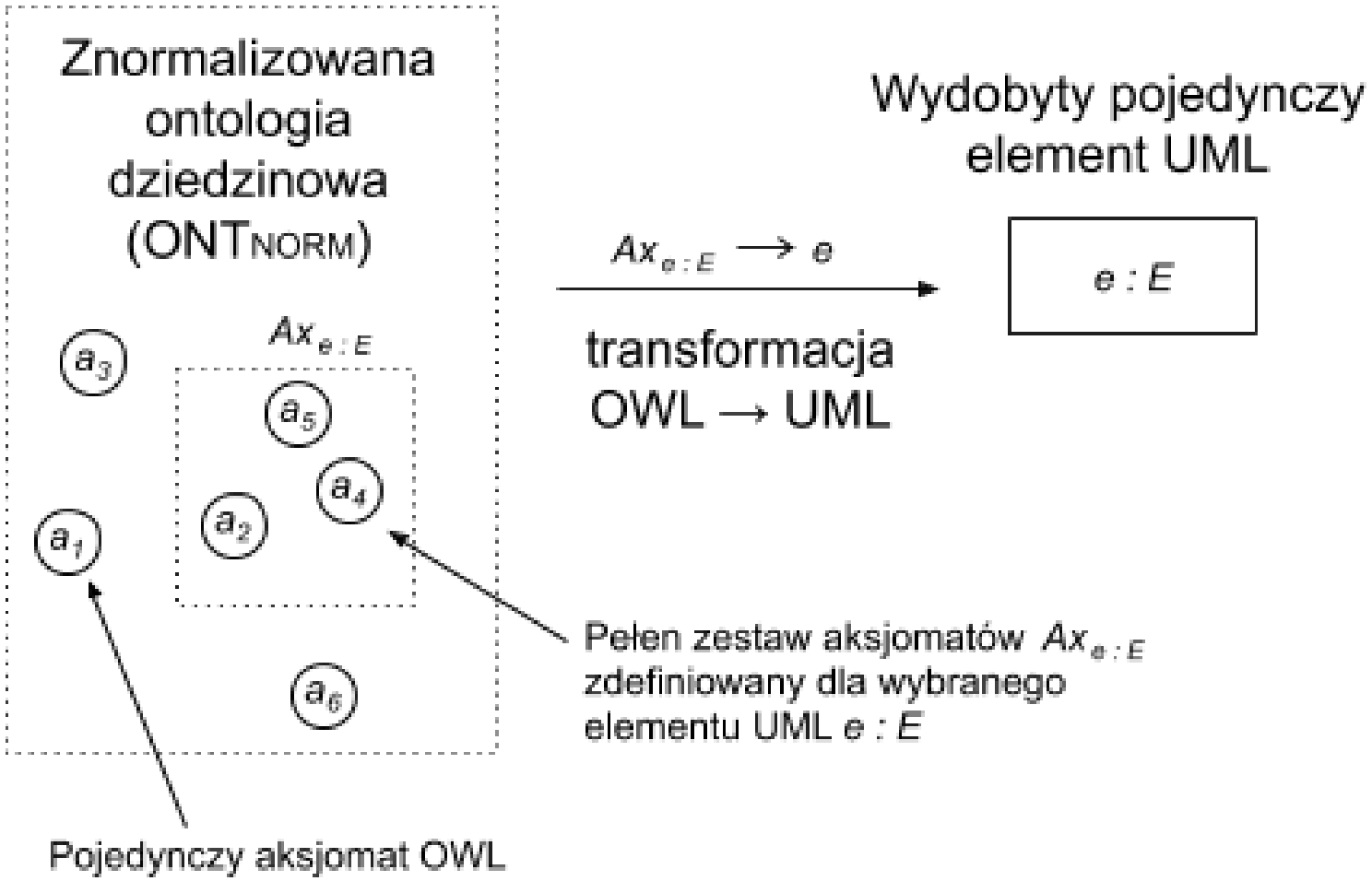
# Tworzenie diagramów w oparciu o ontologie

**Podejście „bezpośrednie”** – Metoda bezpośredniego wydobycia elementów UML wykorzystuje pełne zdefiniowane reguły transformacji

**Podejście „rozszerzone”** – Metoda rozszerzonego wydobycia elementów UML, mająca zastosowanie w przypadku niekompletnych ontologii, umożliwia na wydobycie również takich elementów UML, które nie są w pełni zdefiniowane w ontologii

# III a. TWORZENIE DIAGRAMÓW KLAS PODEJŚCIE „BEZPOŚREDNIE”

# Podejście „bezpośrednie”



# Podejście „bezpośrednie”

**Krok 1:** Normalizacja wybranej ontologii

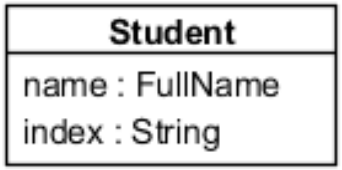
**Krok 2:** Wydobycie elementów UML z ontologii znormalizowanej bazując na regułach transformacji i regułach sprawdzających

**Krok 2A:** Narzędzie automatycznie proponuje listę wszystkich pojęć dziedzinowych występujących w ontologii

**Krok 2B:** Osoba modelująca wybiera potrzebne pojęcia z listy mając na uwadze słownik pojęć i wymagania

**Krok 2C:** Po dokonaniu wyboru, narzędzie automatycznie tworzy diagram klas UML

# Podejście „bezpośrednie” – Przykład 1




| ID | Aksjomaty transformacyjne |
|---|---|
| | **dotyczące klasy UML** |
| TA1 | Declaration( Class( :*Student* ) ) |
| | **dotyczące atrybutów UML** |
| TA2 | Declaration( ObjectProperty( :*name* ) ) |
| TA3 | Declaration( DataProperty( :*index* ) ) |
| TA4 | ObjectPropertyDomain( :*name* :*Student* ) |
| TA5 | DataPropertyDomain( :*index* :*Student* ) |
| TA6 | ObjectPropertyRange( :*name* :*FullName* ) |
| TA7 | DataPropertyRange( :*index* xsd:string ) |

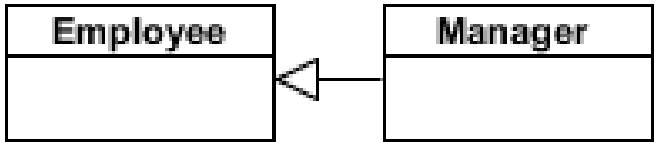
**UML → OWL 2**

| ID | Aksjomaty weryfikacyjne dotyczące atrybutów UML |
|---|---|
| VA1 | ObjectPropertyDomain( :*name* CE ), where CE ≠ :*Student* |
| VA2 | DataPropertyDomain( :*index* CE ), where CE ≠ :*Student* |
| VA3 | ObjectPropertyRange( :*name* CE ), where CE ≠ :*FullName* |
| VA4 | DataPropertyRange( :*index* DR ), where DR ≠ xsd:string |

**OWL 2 → UML**

| ID | Aksjomaty sprawdzające dotyczące atrybutów UML |
|---|---|
| - | BRAK |

# Podejście „bezpośrednie” – Przykład 2




| ID | Aksjomaty transformacyjne |
|---|---|
| **dotyczące klas UML** | |
| TA1 | Declaration( Class( :*Manager* ) ) |
| TA2 | Declaration( Class( :*Employee* ) ) |
| **dotyczące generalizacji UML** | |
| TA3 | SubClassOf( :*Manager* :*Employee* ) |

**UML → OWL 2**

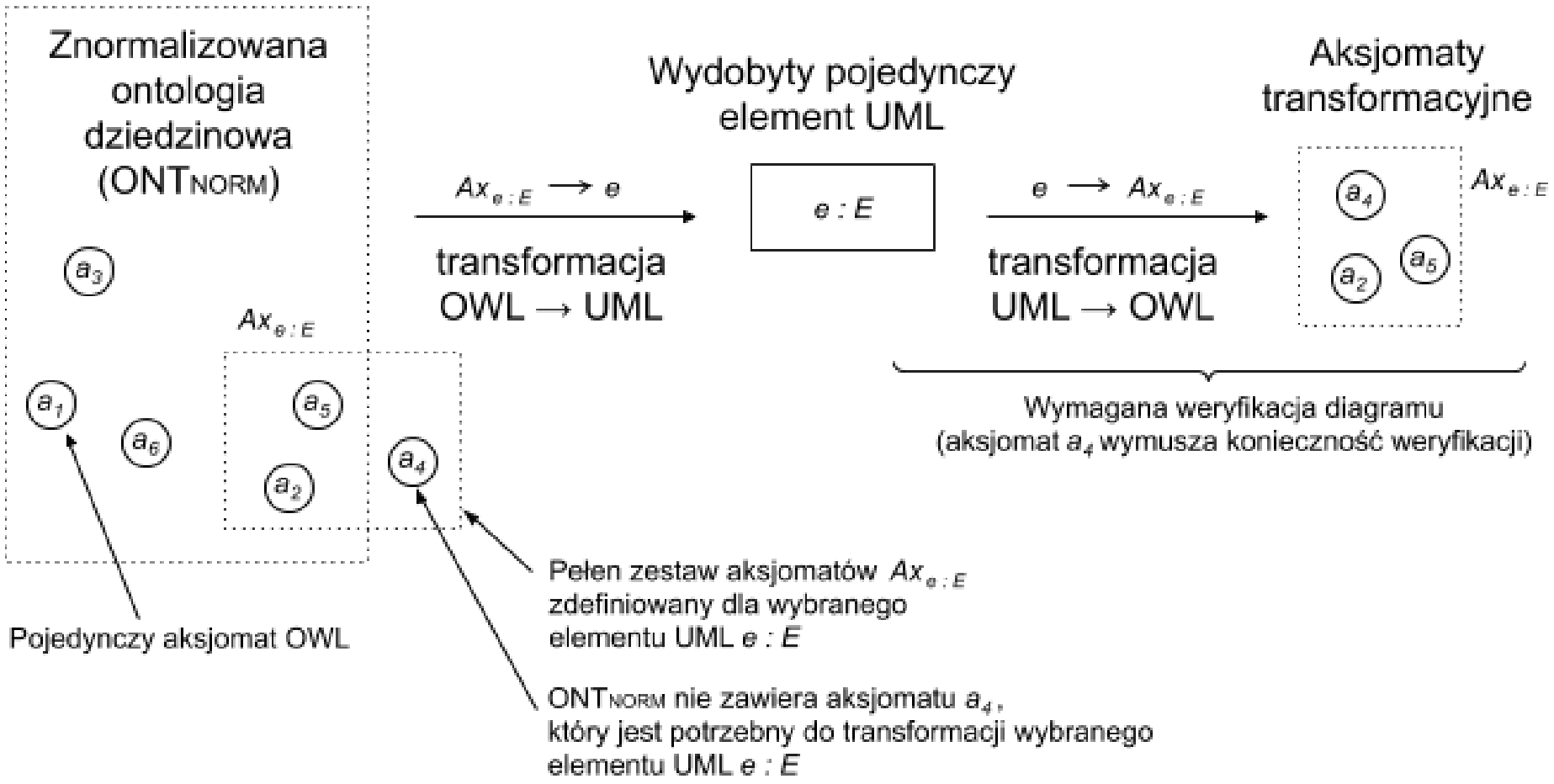
| ID | Aksjomat weryfikacyjny dotyczący generalizacji UML |
|---|---|
| VA1 | SubClassOf( :*Employee* :*Manager* ) |

**OWL 2 → UML**

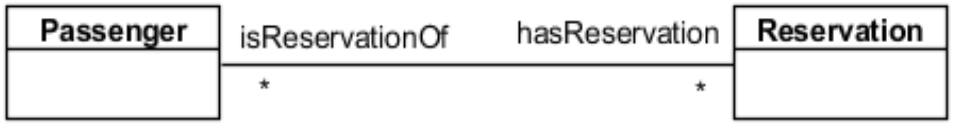
| ID | Aksjomat sprawdzający dotyczący generalizacji UML |
|---|---|
| CA1 | SubClassOf( :*Employee* :*Manager* ) |

# III b. TWORZENIE DIAGRAMÓW KLAS PODEJŚCIE „ROZSZERZONE”

# Podejście „rozszerzone”



# Podejście „rozszerzone” – Przykład 1




| ID | Aksjomaty transformacyjne |
|---|---|
| | **dotyczące klas UML** |
| A1 | Declaration( Class( :*Passenger* ) ) |
| A2 | Declaration( Class( :*Reservation* ) ) |
| | **dotyczące asocjacji UML** |
| A3 | Declaration( ObjectProperty( :*isReservationOf* ) ) |
| A4 | Declaration( ObjectProperty( :*hasReservation* ) ) |
| A5 | ObjectPropertyDomain( :*hasReservation* :*Passenger* ) |
| A6 | ObjectPropertyDomain( :*isReservationOf* :*Reservation* ) |
| A7 | ObjectPropertyRange( :*hasReservation* :*Reservation* ) |
| A8 | ObjectPropertyRange( :*isReservationOf* :*Passenger* ) |
| A9 | InverseObjectProperties( :*isReservationOf* :*hasReservation* ) |

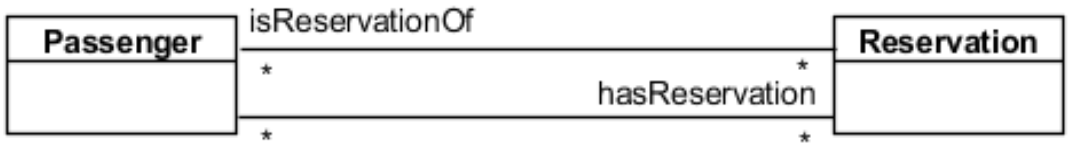
Brak aksjomatów A1-A4 nie wpływa na wynik transformacji

Aksjomaty te zostaną automatycznie utworzone w procesie normalizacji

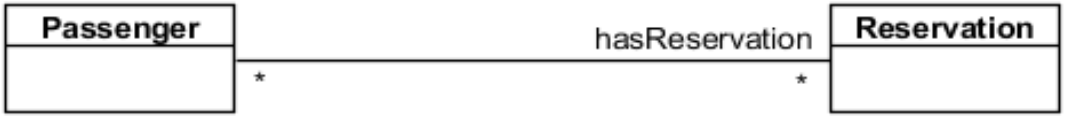
# Podejście „rozszerzone” – Przykład 1

| ID | Aksjomaty transformacyjne |
|---|---|
| A5 | ObjectPropertyDomain( :*hasReservation* :*Passenger* ) |
| A6 | ObjectPropertyDomain( :*isReservationOf* :*Reservation* ) |
| A7 | ObjectPropertyRange( :*hasReservation* :*Reservation* ) |
| A8 | ObjectPropertyRange( :*isReservationOf* :*Passenger* ) |
| A9 | InverseObjectProperties( :*isReservationOf* :*hasReservation* ) |

{A5, A6, A7, A8}

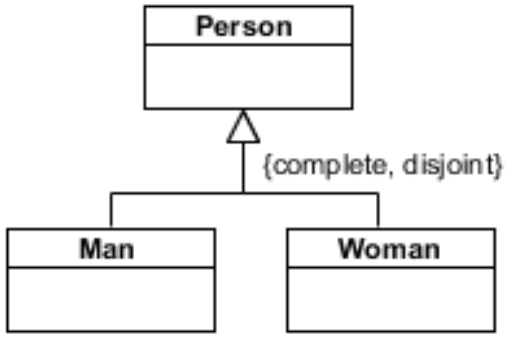



{A5, A7}




Aksjomaty po redukcji:
{A5, A6, A7, A8},
{A5, A6, A7, A9},
{A5, A7, A8, A9},
{A5, A6, A8, A9},
{A6, A7, A8, A9},
{A5, A7, A9},
{A6, A8, A9},
{A5, A7},
{A6, A8}

# Podejście „rozszerzone” – Przykład 2

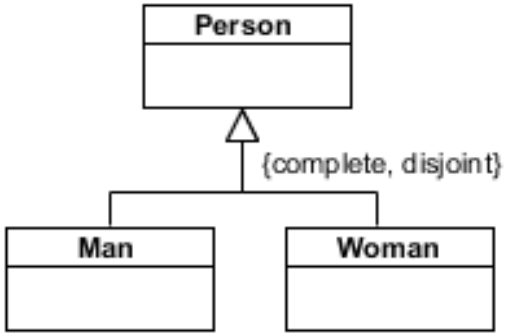



| ID | Aksjomaty transformacyjne |
|---|---|
| **dotyczące klas UML** | |
| A1 | Declaration( Class( :*Person* ) ) |
| A2 | Declaration( Class( :*Man* ) ) |
| A3 | Declaration( Class( :*Woman* ) ) |
| **dotyczące generalizacji** | |
| A4 | SubClassOf( :*Man* :*Person* ) |
| A5 | SubClassOf( :*Woman* :*Person* ) |
| **dotyczące zbioru generalizacji** | |
| A6 | DisjointUnion( :*Person* :*Man* :*Woman* ) |

Analogicznie - brak aksjomatów A1-A3 nie wpływa na wynik transformacji

# Podejście „rozszerzone” – Przykład 2

| ID | Aksjomaty transformacyjne |
|---|---|
| A4 | SubClassOf( :*Man* :*Person* ) |
| A5 | SubClassOf( :*Woman* :*Person* ) |
| A6 | DisjointUnion( :*Person* :*Man* :*Woman* ) |




W oparciu o normalizację – aksjomat A6 jest semantycznie równoważny:

| | |
|---|---|
| A6a | EquivalentClasses( :*Person* ObjectUnionOf( :*Man* :*Woman* ) ) |
| A6b | DisjointClasses( :*Man* :*Woman* ) |

Aksjomaty po redukcji {A4, A5, A6b}

| | |
|---|---|
| A4 | SubClassOf( :*Man* :*Person* ) |
| A5 | SubClassOf( :*Woman* :*Person* ) |
| A6b | DisjointClasses( :*Man* :*Woman* ) |

# IV. MODYFIKACJA DIAGRAMÓW KLAS

# Powody potrzeby modyfikacji diagramu

- Niekompletna ontologia
- Różne poziomy abstrakcji ontologii i diagramu
- Ewolucja ontologii, np. dziedzina finansowa, prawna
- Ontologia „prosta” składająca się wyłącznie z kilku typów aksjomatów
- Różnice między językami UML a OWL 2

# Typowe modyfikacje diagramów

- uszczegółowienie diagramu,
  np. zmiana liczności z "*" na "M..N"
- uzupełnienie diagramu o nowe elementy UML,
  np. dodanie nowych klas lub atrybutów
- usunięcie wybranych elementów diagramu,
  np. klasy UML o nazwie Thing, odpowiednika owl:Thing

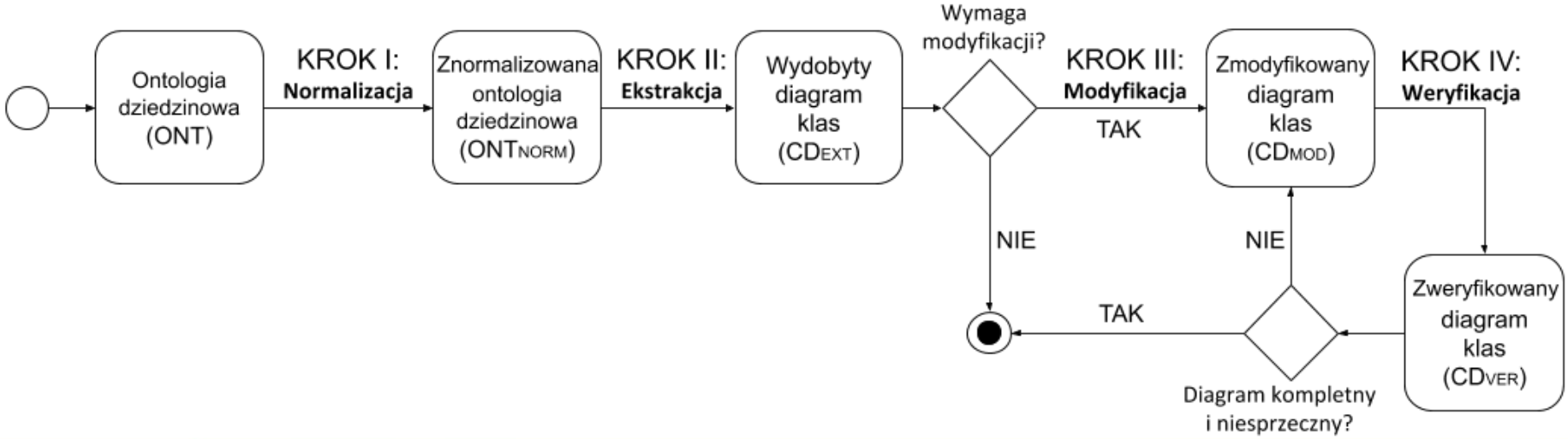
# Modyfikacja diagramu klas

**Krok 1:** Normalizacja wybranej ontologii

**Krok 2:** Wydobycie elementów UML z ontologii
(podejście „bezpośrednie” lub „rozszerzone”)

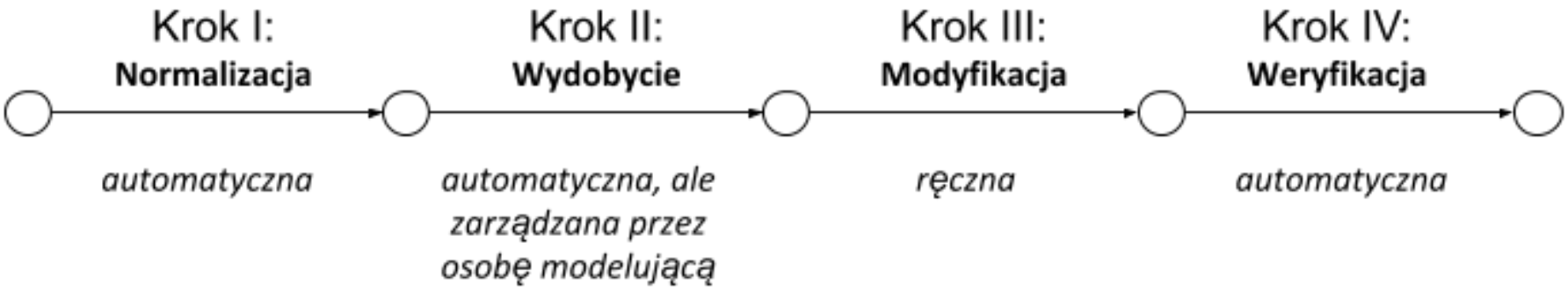
**Krok 3:** Modyfikacja diagramu klas (jeśli konieczna)

**Krok 4:** Weryfikacja zmodyfikowanego diagramu klas



# V. PODSUMOWANIE

# Metoda tworzenia diagramów klas
# - etapy wykonywane automatycznie i ręcznie



# Elementy oryginalne

- Metoda normalizacji - propozycja oryginalna
- Tworzenie diagramów:
  - podejście „bezpośrednie”
    - podejście bazuje na rozszerzeniu propozycji literaturowych o szerokie spektrum konstrukcji UML oraz o oryginalne reguły sprawdzające
  - podejście „rozszerzone” - propozycja oryginalna
- Automatyczna weryfikacja diagramów

  - propozycja oryginalna

# Publikacje (1/2)

## Artykuły opublikowane:


[1] M. Sadowska, “An Approach to Assessing the Quality of Business Process Models Expressed in BPMN”, In e-Informatica Software Engineering Journal, vol. 9, no. 1, pp. 57–77, 2015.

[2] Z. Huzar and M. Sadowska, “Towards Creating Complete Business Process Models”, In From Requirements to Software: Research and Practice, pp. 77-86, 2015.

[3] M. Sadowska and Z. Huzar, “Semantic Validation of UML Class Diagrams with the Use of Domain Ontologies Expressed in OWL 2”, In Software Engineering: Challenges and Solutions. Springer International Publishing, pp. 47–59, 2017.

[4] M. Sadowska, “A Prototype Tool for Semantic Validation of UML Class Diagrams with the Use of Domain Ontologies Expressed in OWL 2”, In Towards a Synergistic Combination of Research and Practice in Software Engineering. Springer, Cham, pp. 49–62, 2018.

# Publikacje (2/2)

## Artykuły opublikowane:


[5] M. Sadowska and Z. Huzar, “The method of normalizing OWL 2 DL ontologies”, In Global Journal of Computer Science and Technology, vol. 18, no. 2, pp.1-13, 2018.

[6] M. Sadowska and Z. Huzar, “Representation of UML Class Diagrams in OWL 2 on the Background of Domain Ontologies”, In e-Informatica Software Engineering Journal, vol. 13, no. 1, pp. 63–103, 2019.


## Artykuł w recenzji:


[7] M. Sadowska and Z. Huzar, “Creation of UML class diagrams on the basis of OWL 2 domain ontologies”, Software and Systems Modeling, 2020.

# Dziękuję za uwagę