

Object-Role Modeling – prezentacja ekspresywności w kontekście porównania z UML

Maciej Sobczak

1 Wstęp

Notacja ORM jest uznanym narzędziem (językiem oraz procedurą postępowania) wykorzystywanym do analizy koncepcyjnej systemów informatycznych.

Biorąc pod uwagę fakt, że analiza koncepcyjna często jest mylona z projektowaniem systemu na poziomie logicznym i na poziomie struktury statycznej, analitycy tradycyjnie wybierają jako swoje narzędzie notację ER lub diagram klas UML. Okazuje się, że w ten sposób analiza koncepcyjna jest obszarem, w którym notacja ORM i diagram klas UML konkurują ze sobą i równocześnie jest to miejsce gdzie notacja ORM okazuje się być znacznie bardziej ekspresywna i uniwersalna, operując na nieco innym poziomie abstrakcji.

Niniejszy dokument przedstawia ekspresywność notacji ORM i jednocześnie jej przewagę nad notacją UML w kontekście analizy koncepcyjnej¹. Diagram klas UML można traktować jako należący do rodziny notacji ER, dzięki czemu uwagi dotyczące UML w dużym stopniu odnoszą się też to popularnych notacji ER.

¹Notacja UML obejmuje kilka rodzajów diagramów, z których tylko dwa (diagram klas i diagram obiektów) mają zastosowanie do analizy danych. Pozostałe diagramy UML opisują dynamikę systemu lub sposób jego fizycznej realizacji i służąc do innych zastosowań nie stanowią konkurencji dla ORM w obszarze analizy danych.

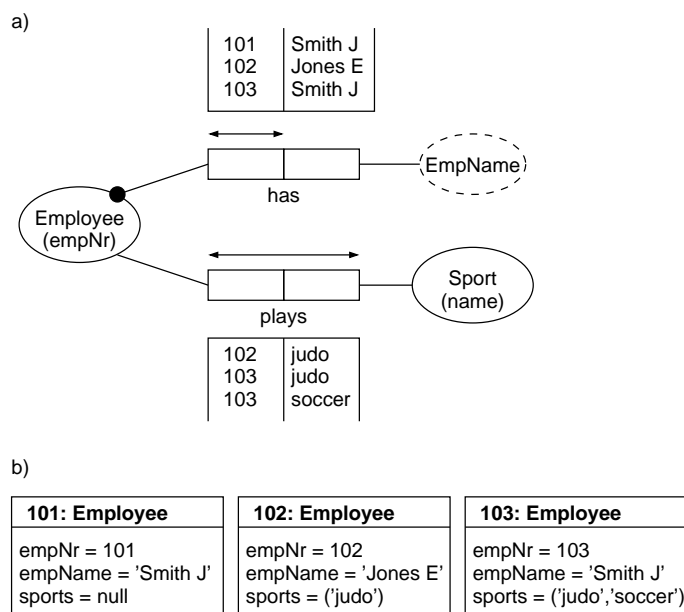
2 Przykłady ekspresywności ORM

2.1 Weryfikacja przez przykładowe populacje

ORM pozwala na użycie tabel faktów, w których można umieścić przykładowe populacje odpowiadających im relacji. Znacznie ułatwia to wyjaśnienie i dyskusję statycznych więzów istniejących na diagramie. Ponadto, użycie *kontrprzykładów* pozwala na szybką weryfikację więzów.

UML posiada bardzo ograniczone możliwości przedstawienia przykładowych instancji klas – służą do tego diagramy obiektów. Problem polega na tym, że umieszczenie na diagramie kilku lub więcej obiektów tej samej klasy czyni diagram nieczytelnym – jest to konsekwencją użycia atrybutów w klasach i ich graficznej reprezentacji. W UML każdy obiekt jest reprezentowany jako osobny prostokąt na diagramie (wraz z atrybutami), podczas gdy w ORM każda instancja relacji jest reprezentowana jako jeden wiersz w tabeli faktów.

Bez możliwości łatwego przedstawienia całych zbiorów instancji trudniej jest zrozumieć i zweryfikować więzy unikalności.



Rysunek 1: Przykładowe populacje: a) ORM, b) UML

Rysunek 1 przedstawia przykładowe populacje w obu notacjach². Łatwo można zauważyć na diagramie ORM, że przykładowa populacja górnego predykatu (*has*) sugeruje więzy unikalności relacji $n:1$, podczas gdy populacja dolnego predykatu (*plays*) sugeruje więzy unikalności oznaczające relację $n:m$.

Wyciągnięcie takich wniosków z diagramu obiektów UML wymaga znacznie więcej uwagi i przy większych diagramach może być niepraktyczne lub wręcz niemożliwe.

2.2 Spójność reprezentacji

2.2.1 Koncepcja relacji unarnych

ORM nie ogranicza arności relacji i pozwala na tworzenie relacji o dowolnej arności, łącznie z relacjami unarnymi. W UML relacje unarne nie są możliwe, zamiast nich używa się atrybutów Boolean, co jest mniej spójne, niż predykatowe podejście ORM.

2.2.2 Reprezentacja graficzna

W UML relacje unarne, binarne (proste linie pomiędzy klasami) i wyższych rzędów (romb połączony z klasami) mają różne reprezentacje graficzne. W ORM wszystkie rodzaje relacji mają jednakowe reprezentacje – rola jest prostokątem a ilość prostokątów w rzędzie wyznacza arność predykatu.

2.2.3 Werbalizacja relacji

Istotnym ograniczeniem związanym z niespójnością reprezentacji relacji w UML jest fakt, że relacje ternarne i wyższych rzędów nie mogą być werbalizowane do postaci zdań w języku naturalnym. Znacznie utrudnia to weryfikację modelu danych i jego dyskusję z ekspertem, który nie musi mieć wykształcenia informatycznego.

W oróżnieniu od UML, każda relacja w notacji ORM ma przynajmniej jeden predykat pozwalający na werbalizację dowolnej relacji. Co więcej, ORM pozwala na użycie dowolnego sposobu werbalizacji w sensie pozycji nazwy predykatu i nazwy typu obiektu w zdaniu. W przypadku języka angielskiego nie ma to wielkiego znaczenia, ale np. w języku japońskim czasowniki występują na końcu zdania i

²Rysunek pochodzi z [2].

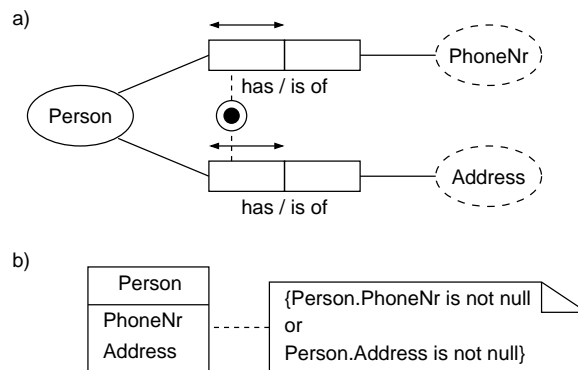
możliwość werbalizacji relacji w dowolnej postaci (*prefix*, *infix*, *postfix* lub *mixfix*) jest wtedy cechą pożądaną.

W UML można werbalizować jedynie relacje binarne (nazwa asocjacji wyznacza werbalizację dla każdej instancji relacji) i tylko w postaci *infix*.

2.3 Role sumarycznie obowiązkowe

ORM pozwala na definiowanie tzw. więzów ról sumarycznie obowiązkowych (ang. *disjunctive mandatory constraint*). Znaczenie takich więzów jest takie, że dany obiekt musi wziąć udział w co najmniej jednej z wymienionych ról.

UML nie posiada takiego narzędzia. Możliwość opisu relacji w formie atrybutu lub asocjacji komplikuje ten problem jeszcze bardziej, gdyż takie więzy nie muszą wtedy dotyczyć tylko atrybutów, ale mogą odnosić się do dowolnej kombinacji atrybutów i asocjacji. UML pozwala na tworzenie dodatkowych opisów w formie komentarzy lub OCL (ang. *Object Constraint Language*), ale nie jest to obowiązkowe, co sprzyja tworzeniu nieprzenośnych modeli. Ponadto użycie OCL do zdefiniowania takich więzów nie daje się łatwo werbalizować i przez to jest trudne w weryfikacji.



Rysunek 2: Więzy sumarycznie obowiązkowe: a) ORM, b) UML

Przykładem zastosowania więzów ról sumarycznie obowiązkowych w ORM jest rysunek 2. Na rysunku tym pokazano też próbę wyrażenia równoważnego modelu w UML, gdzie więzy są „zdefiniowane” w komentarzu. Przedstawia on system, w którym osoba musi mieć zdefiniowany adres lub numer telefonu lub też obie te informacje równocześnie, ale nie jest dozwolona sytuacja, kiedy obie nie są znane.

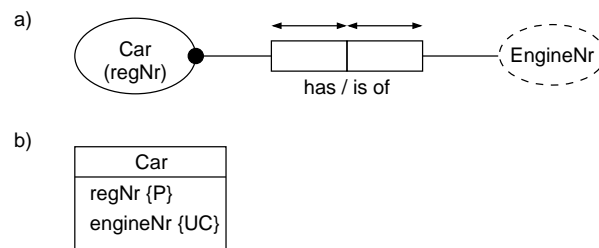
2.4 Więzy unikalności atrybutu

UML nie pozwala na opisanie więzów unikalności atrybutu (ang. *attribute uniqueness constraint*), czyli zastrzeżenia, że dana wartość atrybutu może być przypisana tylko jednej instancji klasy. ORM pozwala na użycie więzów unikatowych 1:1, które takie zadanie spełniają.

UML pozwala na stworzenie własnego rozszerzenia w tym celu (np. dodanie napisu "UC" w nawiasach za nazwą atrybutu), ale jest to rozwiązanie nieprzenośne i często niemożliwe do wykonania przy użyciu dostępnych narzędzi CASE. W praktyce diagramy UML w ogóle nie używają tego typu rozszerzeń, co istotnie ogranicza możliwości modelowania więzów unikalności.

Jedną z możliwości w UML jest rezygnacja z modelowania roli jako atrybutu i zastosowanie asocjacji, dla której można zdefiniować unikalność wartości. Takie rozwiązanie wskazuje jednak na istotną wadę notacji UML (i ER) przejawiającą się właśnie w niestabilności diagramów przy dodawaniu nowych elementów.

Problemem zbliżonym koncepcyjnie do więzów unikalności atrybutu jest zdefiniowanie schematu referencyjnego dla danego typu, czyli określenie sposobu identyfikowania obiektów.



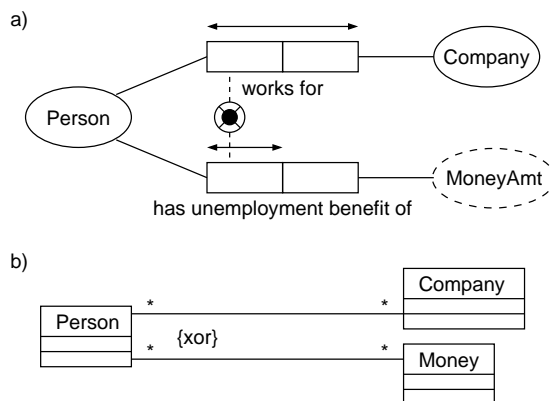
Rysunek 3: Więzy unikalności atrybutu: a) ORM, b) UML

Rysunek 3 przedstawia prosty diagram ORM i próbę jego wyrażenia w UML. Zgodnie z tym rysunkiem, samochód jest identyfikowany przez numer rejestracyjny, ale jego numer silnika również powinien być unikalny. Zarówno schemat referencyjny jak i unikalność atrybutu zostały zdefiniowane w UML przy pomocy rozszerzeń.

2.5 Więzy wyłączości

ORM pozwala na tworzenie więzów wyłączości (ang. *exclusion constraints*) na dowolnej liczbie ról, niekoniecznie pojedynczych. W ogólności, więzy wyłączości można zdefiniować na wielu kompatybilnych sekwencjach ról.

UML nie posiada takiego narzędzia, zdając się na komentarze lub opisy OCL.



Rysunek 4: Więzy wyłączości: a) ORM, b) UML

Istnieje możliwość określenia więzów wyłączości pomiędzy dwoma binarnymi asocjacjami (jak na rysunku 4), ale nie jest już możliwe określenie takich więzów pomiędzy dwoma atrybutami lub pomiędzy atrybutem i asocjacją. Ponadto znaczenie takich więzów jest nieco inne w ORM i UML:

- W ORM istnieją dwa *niezależne* więzy: więzy ról sumarycznie obowiązkowych i więzy ról wykluczających się. Można je stosować niezależnie od siebie i jeśli oba rodzaje więzów są zdefiniowane dla tych samych ról (lub sekwencji ról) to w efekcie otrzymujemy więzy obowiązkowe, wykluczające się (*XOR*).
- W UML istnieje tylko modyfikator *xor*, który można zastosować pomiędzy binarnymi asocjacjami. Nie ma osobnego symbolu graficznego dla więzów wykluczania.

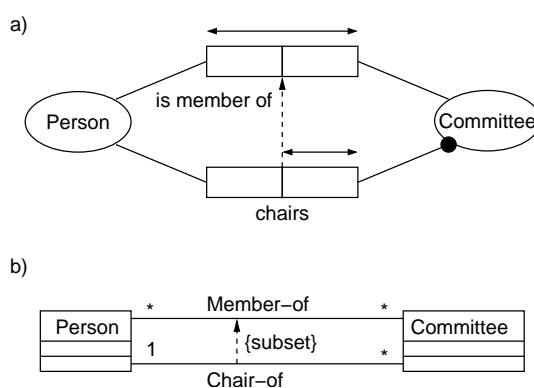
Podjęcie zastosowane w notacji ORM, gdzie relacja jest jedyną konstrukcją służącą do przechowywania danych a więzy są ortogonalne i mogą być stosowane osobno lub razem, procentuje tutaj znacznie większymi możliwościami.

2.6 Więzy podzbiorów

UML pozwala na zdefiniowanie więzów podzbiorów (ang. *subset constraint*), ale tylko w kontekście całych asocjacji.

Możliwości notacji ORM są tutaj znacznie większe – ORM pozwala na tworzenie takich więzów pomiędzy każdymi kompatybilnymi sekwencjami ról, nawet jeśli chodzi tylko o podzbiór większych relacji.

W notacji ORM dwa więzy podzbiorów zastosowane równocześnie w przeciwnych kierunkach są równoważne więzom równości zbiorów (ang. *equality constraints*). UML nie posiada żadnego graficznego symbolu dla takich więzów.



Rysunek 5: Więzy podzbiorów: a) ORM, b) UML

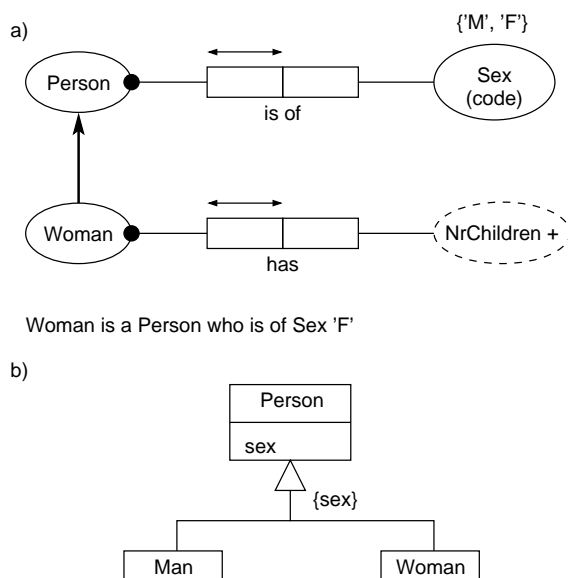
Rysunek 5 prezentuje więzy podzbiorów w ORM i UML.

2.7 Dziedziczenie i definicje podtypów

ORM wymaga, aby podklasy były formalnie zdefiniowane w tym sensie, że dla każdej podklasy musi istnieć reguła pozwalająca na stwierdzenie, czy dany obiekt można uznać za obiekt danej klasy. Dzięki temu jest jasne, jakie kryteria determinują przynależność obiektu do pewnej podklasy – może to być nawet wykorzystane w narzędziach automatycznie generujących zapytania.

UML nie umożliwia tworzenia takich definicji. Istnieje możliwość określenia dyskryminatora w relacji dziedziczenia, ale nie implikuje on żadnych późniejszych ograniczeń (czyli można np. stworzyć populację podklas, która będzie nielegalna w sensie wybranego dyskryminatora). Co więcej, dyskryminator w UML może być jednym z atrybutów klasy bazowej (np. typu wyliczeniowego),

ale nie można w ten sposób stworzyć definicji podklas zawierających np. warunki arytmetyczne (np. *BigCity is a City that has Population > 500000* – to jest bez problemu definiowalne w ORM, przy użyciu dokładnie takiego zapisu).



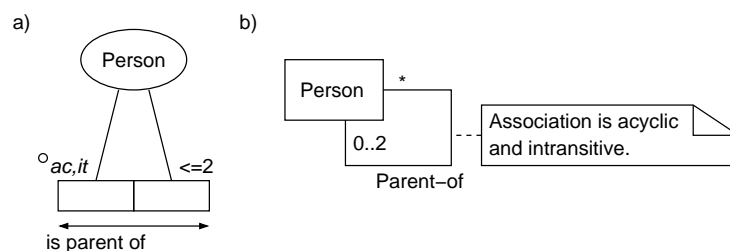
Rysunek 6: Definicje podtypów: a) ORM, b) UML

Rysunek 6 prezentuje przykłady definicji podklas w obu notacjach. Tylko w przypadku ORM wymagana definicja podtypu pozwala na zagwarantowanie poprawności danych.

2.8 Więzy pierścieniowe

ORM pozwala na zdefiniowanie specjalnych więzów w relacjach, gdzie jedna entycja występuje w wielu rolach (ang. *ring constraints*). UML w ogóle nie posiada takiego narzędzia, i tego typu więzy muszą być definiowane jako tekstowe komentarze.

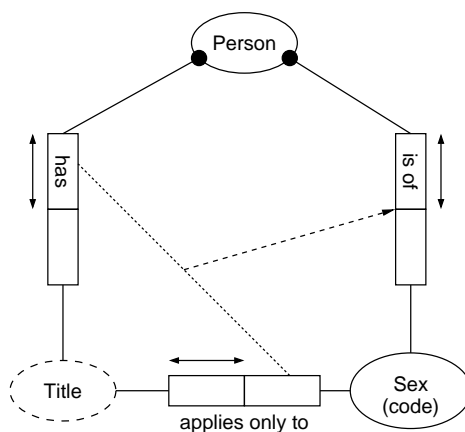
Rysunek 7 przedstawia prostą relację pierścieniową i więzy zdefiniowane w ORM. UML nie pozwala na definiowanie więzów pierścieniowych i na rysunku został do tego celu wykorzystany komentarz.



Rysunek 7: Więzy pierścieniowe

2.9 Więzy złączeń

ORM pozwala na zdefiniowanie więzów złączeń (ang. *join constraints*). UML w ogóle nie posiada takiego narzędzia.



Rysunek 8: Więzy złączeń

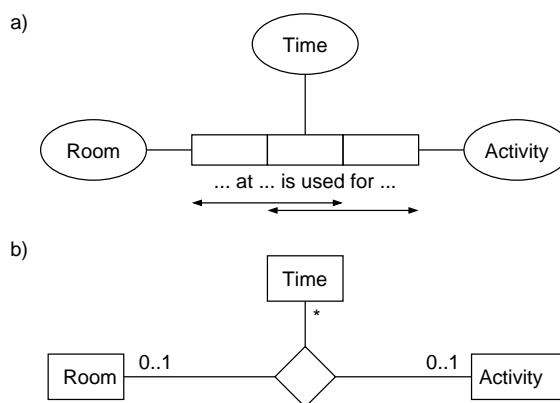
Rysunek 8 prezentuje przykład użycia więzów złączeń w ORM³. Ich znaczenie na tym rysunku jest takie, że osoba ma płeć i tytuł (np. *Mr*, *Ms*, *Mrs*, *Lady*, itp.), ale tytuł posiadany przez osobę musi zawierać się w zbiorze tytułów dozwolonych dla jej płci (czyli np. mężczyzna nie może mieć tytułu *Lady*). Więzy złączeń wyraża się w rachunku relacji jako zawieranie się odpowiednich złączeń, w tym przypadku chodzi o zawieranie się złączenia relacji *has* i *applies only to* w relacji *is of*.

³Rysunek ten pochodzi z [7].

2.10 Znaczenie więzów częstotliwości

UML określa więzy na minimalną ilość wystąpień na przeciwnym końcu asocjacji w stosunku do klasy, której dotyczą (ang. *far end*). W przypadku notacji ORM jest odwrotnie – wszystkie więzy są po stronie tego typu obiektu, którego dotyczą.

W rezultacie, w relacjach ternarnych i wyższych w UML istnieje poważny problem z ich zdefiniowaniem, zwłaszcza gdy minimum ma być inne niż 0 (a jeszcze trudniej, gdy minimum ma być większe od 1). W ORM taki problem nie występuje i więzy częstotliwościowe są zawsze definiowane i interpretowane tak samo, niezależnie od arności relacji, przy której są określone.



Rysunek 9: Więzy przy relacjach wyższych rzędów: a) ORM, b) UML

Rysunek 9 pokazuje dwa równoważne diagramy⁴. Znaczenie więzów jest następujące:

- Jedno zadanie w danym czasie może być wykonywane co najwyżej w jednym pokoju.
- Każdy pokój może być przydzielony do każdego zadania na dowolny czas.
- W jednym czasie jeden pokój może być wykorzystany do co najwyżej jednego zadania.

Innymi słowy, więzy na diagramie UML dotyczą nie tej klasy przy której się znajdują, ale wszystkich pozostałych klas w relacji. Np. więzy *0..1* przy kla-

⁴Rysunek pochodzi z [12].

się *Room* dotyczą nie klasy *Room*, ale pary (*Time, Activity*) – jest to pierwszy z wymienionych powyżej więzów.

Problem z takim podejściem wydaje się być oczywisty – określenie więzów dotyczących pojedynczej klasy (a w ogólności dotyczących mniej niż $n - 1$ klas, gdzie n jest liczbą klas w relacji) w takiej relacji nie jest w UML możliwe. Np. nie jest możliwe zdefiniowanie więzów, że każdy pokój ma być wykorzystany co najmniej dwa razy w „grafiku” lub że każde zadanie ma zostać rozpisane dokładnie w 3 różnych czasach. W notacji ORM nie ma problemu z takimi definicjami, gdyż więzy częstotliwościowe mają zawsze lokalny zasięg.

Literatura

- [1] Terry Halpin. *UML Data Models From An ORM Perspective: Part 1.*
<http://www.orm.net/>.
- [2] Terry Halpin. *UML Data Models From An ORM Perspective: Part 2.*
<http://www.orm.net/>.
- [3] Terry Halpin. *UML Data Models From An ORM Perspective: Part 3.*
<http://www.orm.net/>.
- [4] Terry Halpin. *UML Data Models From An ORM Perspective: Part 4.*
<http://www.orm.net/>.
- [5] Terry Halpin. *UML Data Models From An ORM Perspective: Part 5.*
<http://www.orm.net/>.
- [6] Terry Halpin. *UML Data Models From An ORM Perspective: Part 6.*
<http://www.orm.net/>.
- [7] Terry Halpin. *UML Data Models From An ORM Perspective: Part 7.*
<http://www.orm.net/>.
- [8] Terry Halpin. *UML Data Models From An ORM Perspective: Part 8.*
<http://www.orm.net/>.
- [9] Terry Halpin. *UML Data Models From An ORM Perspective: Part 9.*
<http://www.orm.net/>.

- [10] Terry Halpin. *UML Data Models From An ORM Perspective: Part 10*. <http://www.orm.net/>.
- [11] Terry Halpin. *A comparison of UML and ORM for data modeling*. <http://www.orm.net/>.
- [12] Terry Halpin. *Augmenting UML with Fact-orientation*. <http://www.orm.net/>.