AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Grzegorz Rogus, Piotr Szwed, Jan Werewka

Porównanie metodyk zarządzania projektami PMBOK i Scrum przy użyciu modeli ontologicznych

Część II/III

Laboratorium Informatyki, Katedra Automatyki, AGH, Wydział EAIiE

Seminarium PMI Oddz. Kraków i Explicite KA AGH, 17 czerwca 2010

Project Management Institute
Poland Chapter

AGH

Spis treści

- Co to jest ontologia?
- Dlaczego tworzymy ontologię?
- Komponenty ontologii
- Proces tworzenia ontologii – krok po kroku

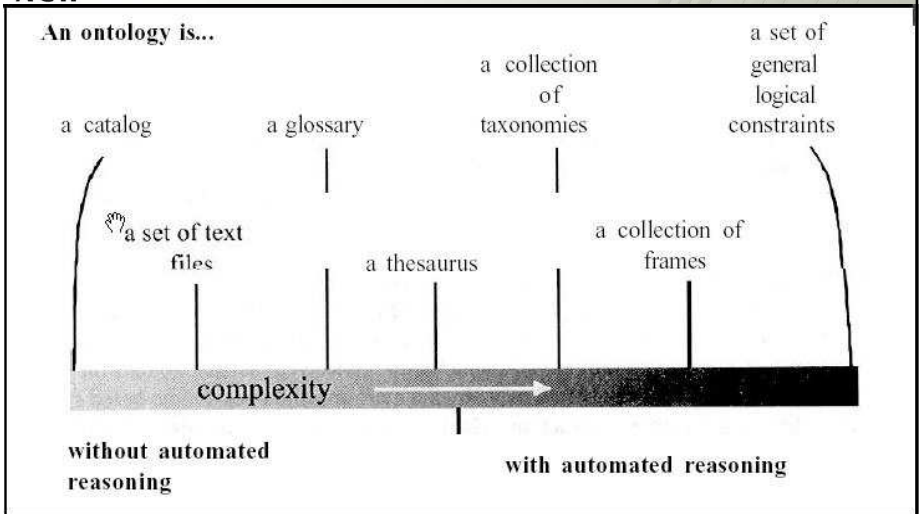
Project Management Institute
Poland Chapter

Ontologia – definicje

- Ontologia:
  - teoria bytu, podstawowy dział filozofii zajmujący się badaniem charakteru i struktury rzeczywistości (Arystoteles, Metafizyka, IV, 1)
  - określa rodzaje istniejących obiektów, właściwości, zdarzeń, procesów, relacji
  - opis rzeczywistości umożliwiający wyczerpującą klasyfikację bytów
- Kluczowe zagadnienia:
  - Istnienie, identyczność, właściwość, realność / hipotetyczność, zależność, możliwość, konieczność, powszechność / wyjątkowość, rzecz / proces, zdarzenie, czas, przestrzeń, przyczyna, ilość
  - Ontologia - opisanie rzeczywistości/związków (a nie wytłumaczenie)
  - **Credo ontologiczne** – aby stworzyć dobre reprezentacje należy mieć wiedzę o reprezentowanych obiektach i procesach – odnosić się do nich

Project Management Institute Poland Chapter

Zakres ontologii



Project Management Institute Poland Chapter

Ontologia

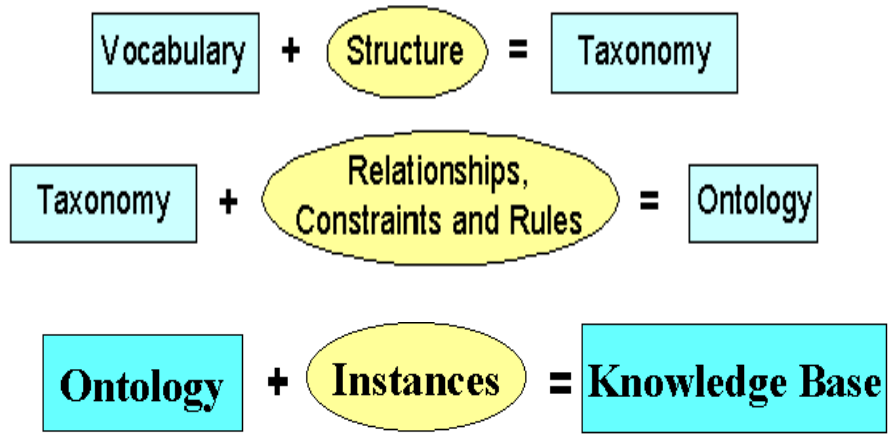
- **Taksonomii** - jest to kolekcja pojęć pomiędzy, którymi zostały określone relacje tworzące hierarchię tych pojęć.
- **Tezaurus** - jest kolekcją pojęć pomiędzy, którymi występują nie tylko relacje tworzące hierarchię tych pojęć, ale również dodatkowe relacje znaczeniowe (semantyczne) takie jak synonimy, homonimów, itp.

Ontologia to reprezentacja klas bytów i zależności między nimi.

Uwaga!!!
Ontologia to rodzaj modelu, a nie katalog przedmiotów danej dziedziny.

Project Management Institute Poland Chapter

Ontologia a baza wiedzy



Project Management Institute Poland Chapter

Pojęcie Ontologii w informatyce oficjalna definicja

Grubber (93) Studer (98)

Formalną, jednoznaczną specyfikacją wspólnej warstwy pojęciowej

- **Zrozumiałą dla aplikacji (możliwość odczytania)**
- **Koncepcje, właściwości, funkcje, aksjomaty są zdefiniowane jawnie**
- **dzielona wiedza**
- **abstrakcyjny model pojęć z rzeczywistości**

Project Management Institute Poland Chapter

Inżynieria Ontologii

- Ontologia jest to jawny opis pewnej dziedziny zawierający:
  - pojęcia
  - własności
  - ograniczenia własności i atrybutów
  - instancje (ale niekoniecznie)
- Ontologia wprowadza:
  - jednolitą terminologię
  - jednolitą interpretację (definicję) pojęć

Project Management Institute Poland Chapter

Budowa ontologii a modelowanie obiektowe

Ontologia

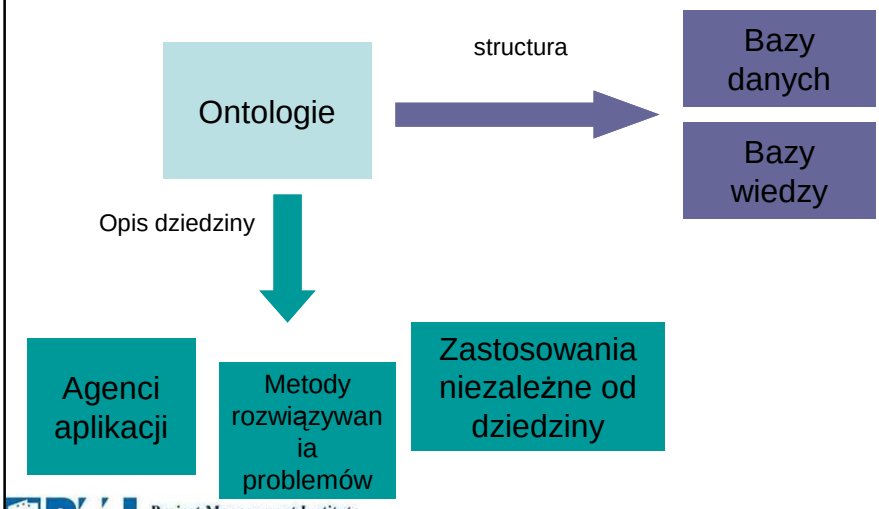
- Odzwierciedla strukturę otaczającej nas rzeczywistości
- Koncentruje się na strukturze pojęcia
- Nie interesuje się aktualną fizyczną reprezentacją modelowanego pojęcia

Modelowanie obiektowe

- Odzwierciedla strukturę danych oraz kodu
- Opisuje zazwyczaj zachowanie (metody)
- Opisuje fizyczną reprezentację danych (long int, char, etc.)

Project Management Institute Poland Chapter

Wykorzystanie ontologii



Project Management Institute Poland Chapter

Ontologie w systemach informacyjnych

- Obszary informatyki odnoszące się do ontologii:
  - inżynieria programowania
    - Struktury danych – reprezentacje 'rzeczy' i programy - procesy
    - Model obiektowy
    - Współdzielenie kodu – biblioteki, moduły
  - Systemy baz danych
    - Projektowanie struktur baz danych – modelowanie konceptualne
  - Sztuczna inteligencja; przetwarzanie języka naturalnego (NLP), systemy eksperckie
  - Sieci semantyczne

Project Management Institute Poland Chapter

Rodzaje ontologii

- Upper Ontology - opisują bardzo ogólne pojęcia takie jak: przestrzeń, czas, sprawa, obiekt, zdarzenie, akcja itp., które są niezależne od problemu czy dziedziny. Wydaje się celowym budowanie zunifikowanych ontologii warstw górnych obejmujących jak największe społeczności użytkowników.

Przykłady :

SUMO (Suggested Upper Merged Ontology)

WordNet

CYC

Project Management Institute Poland Chapter

Przykład istniejącego systemu ontologii – SUO/SUMO

- SUO – Standard Upper Ontology
  - SUO – grupa robocza IEEE mająca za zadanie stworzyć SUMO (http://suo.ieee.org)
- SUMO – Suggested Upper Merged Ontology
  - SUMO powstała z integracji wielu publicznie dostępnych ontologii
  - SUMO ma docelowo zawierać 1000-2500 pojęć i około 10 zdań definiujących dla każdego pojęcia
  - Ontologia dostępna w sieci: http://ontology.teknowledge.com/
  - SUO-KIF – język podstawowy SUMO
    - KIF (Knowledge Interchange Format) - 1992 Stanford – Mike Genesereth
    - Zaprojektowany jako narzędzie wymiany i integracji wiedzy; elastyczne, czytelne dla człowieka i maszyny
    - wariant rachunku predykatów 1 rzędu, podobny do LISPa
    - semantyka teorii zbiorów
    - Mechanizmy meta-językowe
    - 'konceptualizacja': 1.zbiór obiektów mających istnieć w świecie, 2.zbiór właściwości, relacji i funkcji jako zbiór uporządkowanych par
    - Obiekt: indywiduum, zbiór lub klasa
    - Świat zawiera: liczby zespolone, obiekt będący wartością funkcji dla niesensownej kombinacji argumentów funkcji, wszystkie skończone listy i zbiory obiektów, słowa kluczowe i wyrażenia KIF

Project Management Institute Poland Chapter

SUO/SUMO –wybrane elementy

- Pojęcia podstawowe
  - Fizyczne, Abstrakcyjne
- Obiekty
  - ZłożonyZCzęści, Region, Collection, Agent
- Procesy
  - Dwuobiektowy, Intencjonalny, Ruch, ZmianaWewnętrzna,
- Abstrakty
  - ZbiórLubKlasa, Relacja, Proposition, Ilość,
- Struktury
  - M.in.: instancja, domena, poddomena, zakres, partycja, rozdzielny
- Podstawowe relacje binarne
  - Symetryczne, asymetryczne, antysymetryczne, itd.
  - Role: agent, cel, źródło, itd.
- Przedmioty
  - M.in.:Ubranie,dzieło sztuki, budynek
  - Urządzenia: instrument muzyczny, broń, maszyna, itd.
- Liczby
- Jednostki
- Organizmy
- Pojęcia czasowe
  - Przedział czasu
  - Punkt czasowy

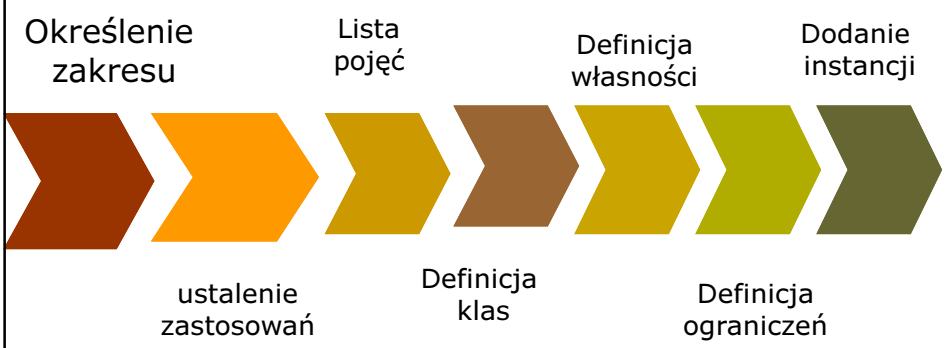
Project Management Institute Poland Chapter

Rodzaje ontologii

- **Ontologie domenowe i ontologie zadań** - opisują , odpowiednio, słownictwo ogólne związane z wybraną domeną (np. medycyna, informatyka) lub ogólne zadania, czynności (np. diagnozowanie, ewaluacja) poprzez specjalizacją termów dla ontologii górnych warstw (np. ontologia dla systemów informacyjnych).
- **Ontologie specyficzne dla aplikacji** - opisują koncepty zależne zarówno od domeny jak i od specyficznych zadań , które zazwyczaj reprezentują specjalizację ontologii obu typów. Koncepty te często oznaczają role, jakie pełnią jednostki domenowe podczas wykonywania czynności, takie jak jednostka wymienna lub komponent zapasowy

Project Management Institute Poland Chapter

Proces budowy ontologii



Project Management Institute Poland Chapter

Gotowe ontologie

- Ogólnodostępne biblioteki ontologiczne
  - Biblioteka DAML ontology (www.daml.org/ontologies)
  - Biblioteka Ontolingua ontology (www.ksl.stanford.edu/software/ontolingua/)
  - Protégé ontology (protege.stanford.edu/plugins.html)
- Upper ontologie
  - IEEE Standard Upper Ontology (suo.ieee.org)
  - Cyc (www.cyc.com)

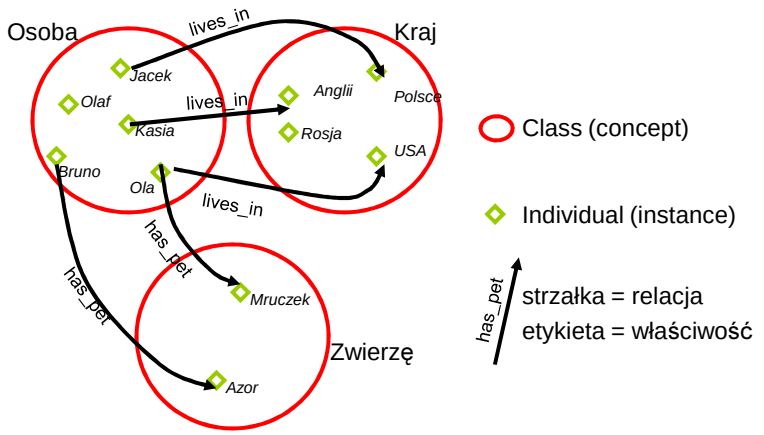
Project Management Institute Poland Chapter

OWL - język definiowania ontologii

- **OWL** (ang. *Web Ontology Language*) jest językiem ze składnią opartą na XML, a semantyką opartą na tzw. logice deskrypcyjnej (ang. *description logics*).
- Stanowi on rozszerzenie RDF (ang. *Resource Description Framework*).
- Służy do reprezentacji i przetwarzania danych w sieci WWW.
- OWL umożliwia opisywanie danych w postaci ontologii i budowanie w ten sposób tzw. Semantycznego Internetu.
- Istnieją trzy odmiany języka OWL:
  - OWL Lite;
  - OWL DL (rozszerzenie OWL Lite);
  - OWL Full (rozszerzenie OWL DL).
- OWL został uznany za standard przez W3C w lutym 2004 roku.

Project Management Institute Poland Chapter

Komponenty Ontologii



Podstawowe komponenty
Klasy

Mammal, Tree, Person, Building, Fluid, Company

- Klasa jest zbiorem indywiduów
- Klasa jest kolekcją elementów o podobnych właściwościach
- Inna nazwa "Type", "Concept", "Category"
- Przynależność do klasy zależy od logicznego opis, a nie jego nazwę
- Klasy nie muszą reprezentować tylko obiektów fizycznych, mogą pełnić rolę reprezentacji abstrakcyjnej np. klasa jako kontener „**rzeczy mających kolor czerwony**"
- Reprezentacja klas możliwy jest na 6 sposobów. Najprostszy z nich to tzw. **Named Class.** Pozostałę nazywane są jako klasy anonimowe **Anonymous Class**

Podstawowe komponenty
Klasy

Klasy są budowane za pomocą opisów które specyfikują warunki jakie muszą być spełnione przez każdy element klasy aby stać się członkiem tej grupy

- **Named Class –** tworzymy klasę i przypisujemy jej nazwę**.**
- **Anonymous Class –** zbudowane w oparciu o elementy dodatkowego opisu klasy **:**
  Intersection, Union, ComplementOf class
  restriction class - universal, existential, cardinality, hasValue
  enumeration class

  Named Class + Anonymous class = complex class description (**klasy złożone**)

Project Management Institute Poland Chapter

Komponenty ontologii
Indywidua

- Indywidua są to obiekty znajdujące się w interesującej nas klasie
- Prezentuje się jej także jako instancje klas
- "Instance", "Object"
- Indywidua mogą należeć do wielu klas (wielodziedziczenie)

Project Management Institute Poland Chapter

Komponenty ontologii
właściwośći (properties)

hasPart, isInhabitedBy, isNextTo, occursBefore

- Właściwości są wykorzystywane do przedstawienia zależności pomiędzy użytymi instancjami klasy, typami danych.
- Opis relacji pomiędzy instancjami dwóch klas
- Relacje w ontologii są relacjami binarnymi :
  Subject → predicate → Object
  Individual a → hasProperty → Individual b
  Ala → posiada → kotka

Project Management Institute Poland Chapter

Właściwości

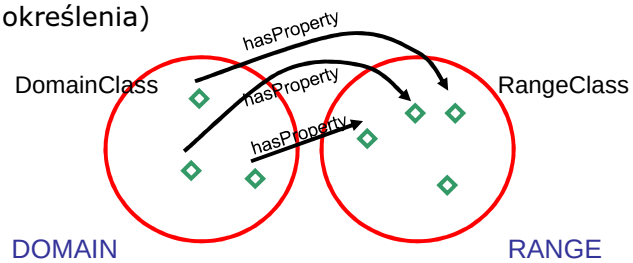
- Object Property – dotyczą indywiduów
- Datatype Property – wiążą indywidua z typami danymi (int, string, float etc)
- Annotation Property – używane do powiązania metadata do klas, indywiduów czy właściwości

Project Management Institute Poland Chapter

Dziedzina (domain) & zakres (range) we właściowościach

Mamy relacje typu:
obiekt → hasProperty (predykat) → określenie

- Dziedzina to zbiór który zostaje odwzorowany w zbiór zasięgu poprzez daną własność - klasa subject_individual (obiekty)
- Zakresem jest zbiór, którego elementy będą wartościami właściwości - klasa object_individual (określenia)



DOMAIN RANGE

class → właściwość → Dozwolone wartości

Project Management Institute Poland Chapter

Dziedziczenie klas

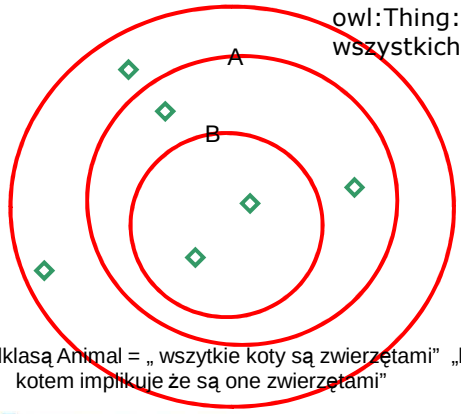
Klasy mogą być zorganizowane w hierarchię nadklasy i podklasy znaną jako taksonomia

- class jest zbiorem elementów -> subclass jest jego podzbiorem
- Jabłko (apple) jest podzbiorem zbioru owoców (Fruit)

Project Management Institute Poland Chapter

Subsumption (zawieranie)

- relacja typu "is-a" Superclass/subclass
- **Wszyscy** członkowie podklasy mogą być traktowani jako członkowie nadklasy



owl:Thing: jest superclass dla wszystkich klas w ontologii OWL

A zawiera B

- A jest nadklasą (superclass) B
- B jest podklasą (subclass) A
- Wszyscy członkowie B są także członkami A

is-a jest relacją przechodnią

B jest podklasą A

C jest podklasą B

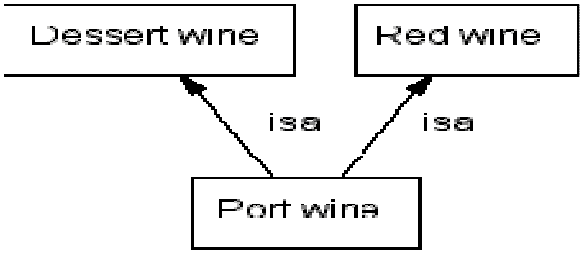
C jest podklasą A

cat jest podklasą Animal = „ wszytkie koty są zwierzętami" „bycie kotem implikuje że są one zwierzętami"

Project Management Institute Poland Chapter

Wielodziedziczenie

- Klasa może mieć wiele nadklas
- Podklasa dziedziczy wtedy właściwości i ograniczenia od wszystkich nadklas



Project Management Institute Poland Chapter

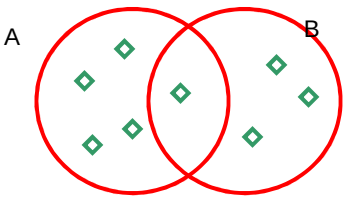
AGH

Klasy złożone

Project Management Institute
Poland Chapter

AGH

Unia klas



- "disjunction"
- A OR B OR C
- This ⊔ That ⊔ TheOther

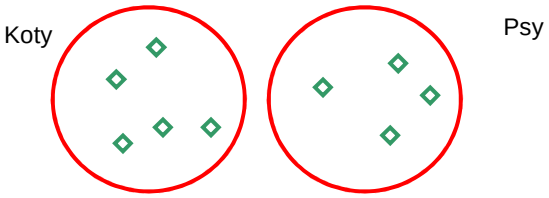
A⊔B zawiera wszystkie elementy klasy A , klasy B oraz wszystkie elementy należące do części wspólnej A i B

Klasa nie może zawierać żadnych instancji innych niż te znajdujące się w klasach pokrywających

Project Management Institute
Poland Chapter

Disjointness (rozłączność)

- Takie wymaganie modelujemy za pomocą **rozłączności**



Klasy są rozłączne gdy nie mają wspólnych instancji

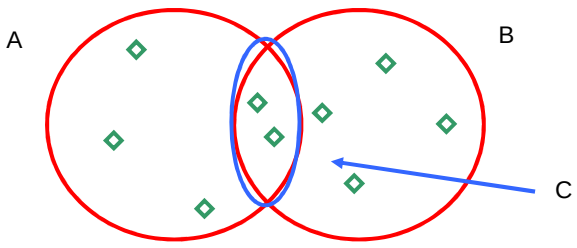
Klasy rozłączne nie mogą dziedziczyć po tych samych nadklasach

Rozłączność modeluje się jawnie nadając klasom odpowiednie własności

Project Management Institute Poland Chapter

Intersection Class (przecięcie)

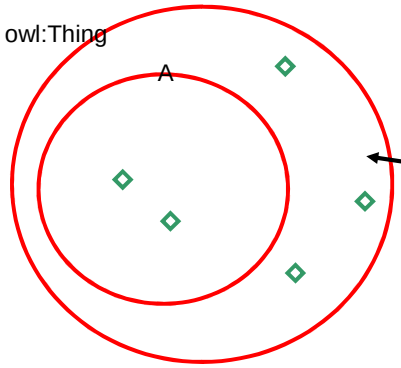
- "conjunction"
- A AND B AND C
- This ⊓ That ⊓ TheOther
- C ≡ A and **B**



Project Management Institute Poland Chapter

ComplementOf Class (dopełnienie)

- "Negacja" "Not"
- Not Something
- ¬ A

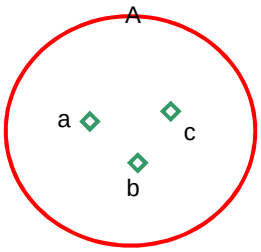


Wszystkie obiekty z tego zbioru należą do ¬A

Project Management Institute Poland Chapter

enumeration Class (wyliczenie)

- Polega na jawnym podaniu (wyliczeniu) wszystkich elementów klasy



Project Management Institute Poland Chapter

Własności i restrykcje

Project Management Institute
Poland Chapter

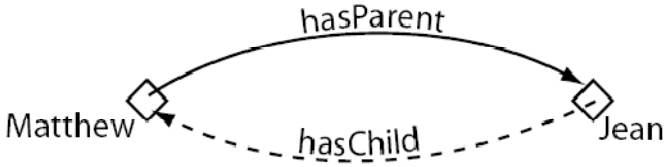
Typy właściwości

- **Inverse** (odwrotna)
- **Functional** (funkcjonalna)
- **Inverse Functional** (funkcjonalna odwrotna)
- **Symmetric** (symetryczna)
- **Transitive** (przechodnia)

Project Management Institute
Poland Chapter

Relacja odwrotna

- Dzięki tej relacji można udowodnić następującą zależność dla obiektów a, b i własności X oraz odwrotnej do niej własności Y. JeśliX(a, b) jest poprawne to również Y(a, b) jest poprawne

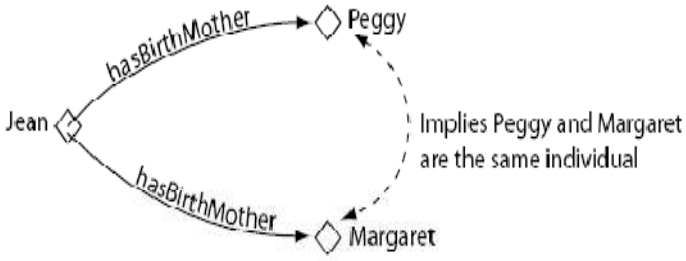


Project Management Institute Poland Chapter

Relacja funkcjonalna

- Jeśli relacja X jest relacją funkcjonalną dla następujących obiektów a,b,c, gdzie X(a,b) oraz wiedząc że X(a,c) to możemy dojść do wniosku że b=c.

Dla podanej dziedziny zakres musi być unikalny

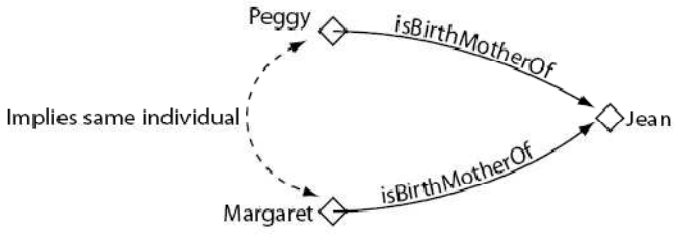


Poland Chapter

Relacja funkcjonalna odwrotna

- Jeśli relacja jest funkcjonalna odwrotna to jest relacja odwrotna jest funkcjonalna
- Własność X dla obiektów a,b,c mamy X(a,b) oraz X(c,b) zatem poprawne jest stwierdzenie że a=c.

Dla podanego zakresu dziedzina musi być unikalna



Project Management Institute Poland Chapter

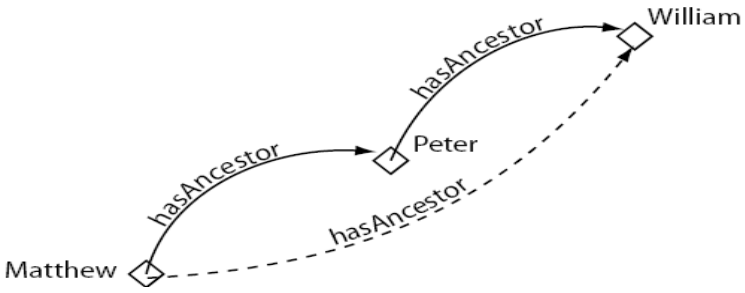
- Relacja funkcjonalna odwrotna a funkcjonalana

	dziedzina	zakres	przykład
Funkcjonalna	dowolna	unikalny	hasFather: A hasFather B, A hasFather C →B=C
Funkcjonalna odwrócona	unikalny	dowolna	hasID: A hasID B, C hasID B →A=C

Project Management Institute Poland Chapter

Relacja Przechodniości

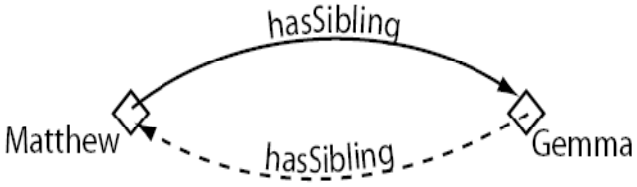
- Są do dyspozycji trzy obiekty a,b,c i dla danej własności X zdefiniowana została następująca zależność X(a,b) oraz X(b,c). Na podstawie własności przechodniości można stwierdzić prawdziwość faktu X(a,c)



Project Management Institute Poland Chapter

Relacja symetryczna

- Jeśli własność X jest symetryczna oraz X(a,b) to możemy stwierdzić że X(b,a) jest również prawdziwe.



Project Management Institute Poland Chapter

Czego jeszcze brak?

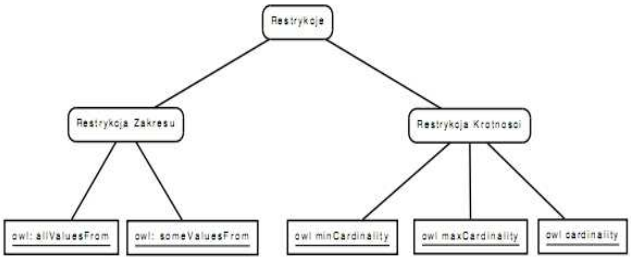
- Przedstawiony do tej pory model opisu ontologicznego jest dość ubogi semantycznie
- Brak w nim oprócz relacji oprócz typu "is kind of" (subsumption) oraz "is not kind of" (disjoint) innych relacji pozwalających na modelowanie różnych wzajemnych zależności

Project Management Institute
Poland Chapter

Restrykcje relacji

Właściwości są używane do tworzenia relacji związanych z dodatkowymi wymaganiami zwanymi restrykcje.

- Używa się ich do ograniczenia dostępu do klas
- Typy ograniczeń :
  - Restrykcje typu kwantyfikatory zakresu
    - Isnienia Existential quantifier
    - Uniwersalny Universal quantifier
  - Restrykcje ilościowe Cardinality restrictions
  - hasValue restrictions



Project Management Institute
Poland Chapter

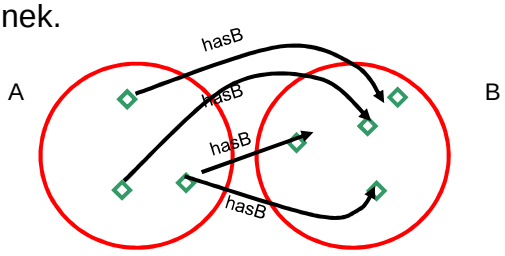
Typy restrykcji

∃	Existential, someValuesFrom	"Some", "At least one"
∀	Universal, allValuesFrom	"Only"
∋	hasValue	"equals x"
=	Cardinality	"Exactly n"
≤	Max Cardinality	"At most n"
≥	Min Cardinality	"At least n"

Project Management Institute Poland Chapter

Existential, someValuesFrom

Własność ta specyfikuje że przynajmniej jedna wartość opisująca daną własność spełnia ten warunek.



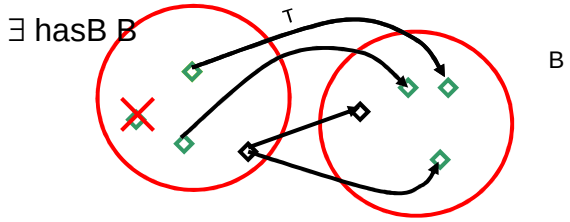
- Mamy restrykcję: ∃ hasB **B**
  dla klasy **A** traktowaną jako warunek konieczności

Każdy element klasy A musi mieć przynajmniej jedną relację z obiektem z klasy B

Project Management Institute Poland Chapter

Existential, someValuesFrom

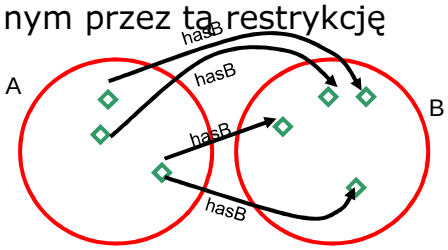
- Mamy restrykcję: ∃ hasB **B** dla klasy **A**



► Restrykcja ta wymaga aby nie istniał żaden element w klasie A który nie ma przynajmniej jednej relacji z obiektami z klasy B

Universal, allValuesFrom

Własność ta specyfikuje że dla każdej instancji klasy posiadającej tą własność, wartości tej właściwości zawierały się będą w zbiorze wskazanym przez tą restrykcję

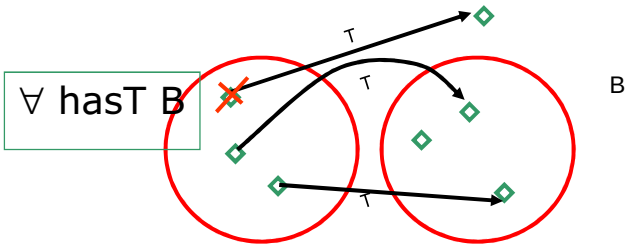


- Restrykcja : ∀ hasB B dla klasy A jako warunek konieczny

► Jeśli obiekt jest elementem klasy A to wymagane jest aby miał tylko relacje z obiektami z klasy B

Universal, allValuesFrom

- Własność ta specyfikuje że dla każdej instancji klasy posiadającej tą własność, wartości tej właściwości zawierały się będą w zbiorze wskazanym przez tą restrykcję

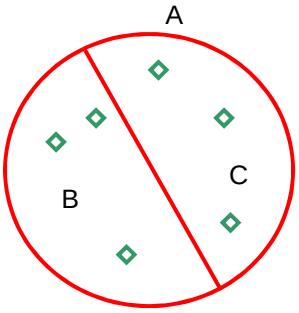


Każdy element klasy A musi mieć przynajmniej jeden element z klasy B

Project Management Institute Poland Chapter

Restrykcje z użyciem klas złożonych– ograniczenie zakresu

$A \equiv B \sqcup C$



- Zakres A jest pokryty przez B lub C
- Każdy element A musi być albo B albo C . Nikt inny nie może należeć do klasy A

Project Management Institute Poland Chapter

Sprawdzanie spójności

- Po stworzeniu dowolnej klasy do modelu dobrze było by sprawdzić logiczną spójność modelu
- Zadaniem tym zajmuje się automatycznie aplikacja zwana **Reasoner**

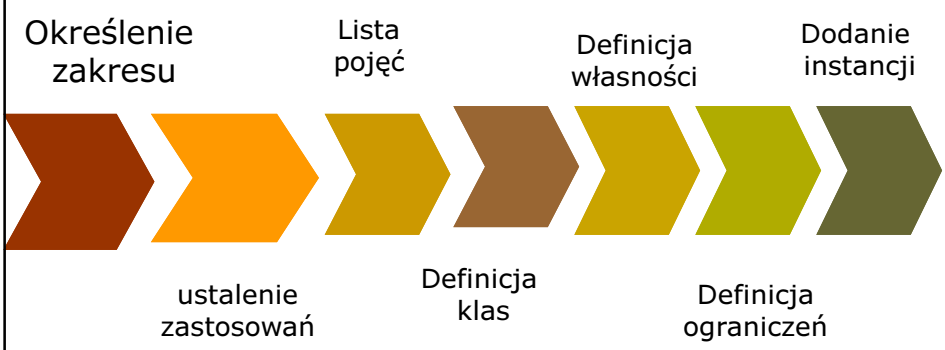
Project Management Institute Poland Chapter

Reasoner – maszyna wnioskująca

- Reasoner jest używany do wnioskowania o informacjach które nie są jawnie podane w ontologii
- Standardowe maszyny wnioskujące sprawdzają:
  - **Consistency Checking**
  - **Subsumption Checking**
  - **Equivalence Checking**
  - **Instantiation Checking**

Project Management Institute Poland Chapter

Proces budowy ontologii



Project Management Institute Poland Chapter

Przykład budowy ontologii: Animals & Plants

- Dog
- Cat
- Cow
- Person
- Tree
- Grass
- Herbivore
- Male
- Female

- Carnivore
- Plant
- Animal
- Fur
- Child
- Parent
- Mother
- Father

- Dangerous
- Pet
- Domestic Animal
- Farm animal
- Draft animal
- Food animal
- Fish
- Carp
- Goldfish

Project Management Institute Poland Chapter

Przykład: Animals & Plants

- Dog
- Cat
- Cow
- Person
- Tree
- Grass
- Herbivore
- Male
- Female

- Carnivore
- Plant
- Animal
- Fur
- Child
- Parent
- Mother
- Father

- Healthy
- Pet
- Domestic Animal
- Farm animal
- Draft animal
- Food animal
- Fish
- Carp
- Goldfish

Project Management Institute Poland Chapter

Wybór głównych koncepcji, dodanie abstrakcji, identyfikacja relacji,

- Living Thing
  - Animal
    - Mammal
      - Cat
      - Dog
      - Cow
      - Person
    - Fish
      - Carp
      - Goldfish
  - Plant
    - Tree
    - Grass
    - Fruit

- Modifiers
  - domestic
    - pet
    - Farmed
      - Draft
      - Food
  - Wild
  - Health
    - healthy
    - sick
  - Sex
    - Male
    - Female
  - Age
    - Adult
    - Child

- Relations
  - eats
  - owns
  - parent-of
  - …
- Definable
  - Carinvore
  - Herbivore
  - Child
  - Parent
  - Mother
  - Father
  - Food Animal
  - Draft Animal

Project Management Institute Poland Chapter

Po integracji

- Primitives
  - Living Thing
    - Animal
      - Mammal
        - » Cat
        - » Dog
        - » Cow
        - » Person
      - Fish
        - » Carp
          Goldfish
    - Plant
      - Tree
      - Grass
      - Fruit
- Modifiers
  - **Domestication**
    - **Domestic**
    - **Wild**
  - **Use**
    - **Draft**
    - **Food**
    - **pet**
  - **Risk**
    - **Dangerous**
    - **Safe**
  - **Sex**
    - **Male**
    - **Female**
  - **Age**
    - **Adult**
    - **Child**
- Relations
  - eats
  - owns
  - parent-of
  - …
- Definables
  - Carnivore
  - Herbivore
  - Child
  - Parent
  - Mother
  - Father
  - Food Animal
  - Draft Animal

Project Management Institute Poland Chapter

Ustalenia zakresu i dziedziny dla właściwości

- Animal *eats* Living_thing
  - *eats* dziedzina: Animal;
    zakres: Living_thing
- Person *owns* Living_thing except person
  - *owns* dziedzina: Person
    zakres: Living_thing & not Person
- Living_thing *parent_of* Living_thing
  - *parent_of:* dziedzina: Animal
    zakres: Animal

Project Management Institute Poland Chapter

Przykłady class definiowalnych które można zdefiniować na podstawie prymitywów i relacji

- Parent =
  Animal and *parent_of* some Animal
- Herbivore=
  Animal and *eats* only Plant
- Carnivore =
  Animal and *eats* only Animal

Project Management Institute Poland Chapter

Definiowanie dla poszczególnych klas ich własności

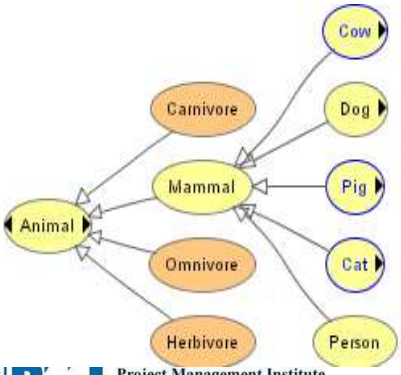
- Cows powinna być Herbivores (roślinożerna)
  Jak to zdefiniować?

  Cows are animals and, *amongst other things,*
  eat *some* plants and eat *only* plants

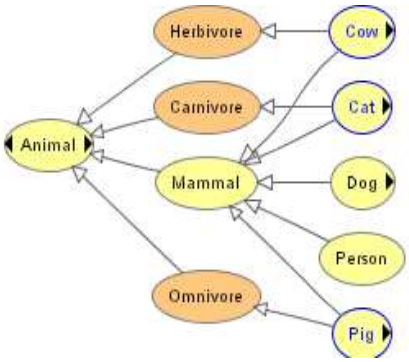
Project Management Institute Poland Chapter

Normalizacja – wprowadzanie polimorfizmu

- Przed klasyfikacją
  - drzewo



- Po klasyfikacji
  - Graf acykliczny



Project Management Institute Poland Chapter

Modyfikacje klas

- Modifiers
  - Domestication
    - Domestic
    - Wild
    - Feral
  - Risk
    - Dangerous
    - Risky
    - Safe
  - Sex
    - Male
    - Female
  - Age
    - Child
      - Infant
      - Toddler
    - Adult
      - Elderly

- Identyfikacja klas z właściwościami mającymi specyficzne wartości
  - Domestication
  - Risk
  - Sex
  - Age
- Stworzenie dodatkowych abstrakcyjnych klas
  - Age → Age_group

Project Management Institute Poland Chapter

Specyfikacja wartości dla klas

- Klasa Dangerousness
- Tworzymy podklasy określające typu wartości
  - Dangerous, Risky, Safe
- Wszystkie są rozłączne
- Definiujemy dodatkowe właściwości dla klasy głównej
  - Dangerousness = Dangerous OR Risky OR Safe
- Definiujemy nową funkcjonalną właściwość has_dangerousness
  - Zakresem jest klasa główna Dangerousness
  - Dziedzina musi być specyfikowana oddzielnie jako Domain must be specified separately

Dangerous_animal =
Animal *and* has_dangerousness *some* Dangerous

Project Management Institute Poland Chapter

PROBLEMY Z PROJEKTAMI INFORMATYCZNYMI

- Przemysł informatyczny ma duże znaczenie gospodarcze i strategiczne.
- Należy dążyć by powstawanie oprogramowania będącego podstawowym wytworem tego przemysłu było efektywne i miało odpowiednią jakość.
- Wytwarzanie oprogramowania dokonywane jest zazwyczaj w oparciu o projekty.
- Okazuje się jednak, że istnieją duże problemy zarządzania projektami informatycznymi i nie są one rozwiązywane w sposób zadawalający.

Project Management Institute Poland Chapter

Raporty chaosu (Chaos Report)

- Dane amerykańskiej grupy konsultingowej Standish Group dotyczące wykonania projektów branży IT w Stanach Zjednoczonych.
- Statystyki te koncentrują się na sukcesie projektu bazującym na popularnym dla projektów trójkącie ograniczeń, który mówi, że projekt zakończył się sukcesem jeżeli został wykonany:
  - na czas (harmonogram),
  - przy zaplanowanym budżecie (koszty),
  - z wymaganymi funkcjami i własnościami (zakres).

Project Management Institute Poland Chapter

Porównanie metodyk zarządzania projektami PMBOK i Scrum przy użyciu modeli ontologicznych – Część II

- **Dziękuję z uwagę - Pytania?**

Project Management Institute Poland Chapter

Literatura

1. Marek Jaślan, Branża IT ma się dobrze, www.msipolska.pl/rynek_200702.php4?num=445, 2007
2. Tomasz Sańpruch, www.prnews.pl, Bankier.pl, 2007
3. Tomasz Szetyński, Rynek IT rośnie powoli - firmy, które chcą istnieć na rynku zmieniają swoją strategie działania, www.globaleconomy.pl/content/view/80/17/, 2010
4. The Standish Group, www.standishgroup.com
5. J. Laurenz Eveleens, Chris Verhofer: The rise and Fall of the Chaos Report Figures. IEEESoftware, 2010, pp. 30- 36
6. Barry Boehm, Richard Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Pearson Education, 204, p. 266
7. A Guide to the Project Management Body of Knowledge Fourth Edition (PMBOK® Guide), Approved American National Standard ANSI/PMI 99-001-2008, PMI 2008, s. 460.

Project Management Institute Poland Chapter

Literatura

8. The state of Agile Development, 3rd annual Survey:2008, Full Data Report Conducted June-July 2008. VersionOne Inc. www.VersionOne.com/acnewsletter, s. 18.
9. Werewka J., Scaling Management of IT Projects Depending on their Size and Complexity, 1st CEE Symposium on Business Informatics, ACS, Vienna 2009, s. 181-190.
10. Mobilizing HP, Project Management as an Executive Priority. Benchmark Implementation: Primavera, Case Study, 2004 Benchmarking Partners, pp. 9
11. Schwaber K., Beedle M: Agile software development with Scrum. Upper Saddle River, N.J., Prentice Hall, 2002
12. Franke, U.; Hook, D.; Konig, J.; Lagerstrom, R.; Narman, P.; Ullberg, J.; Gustafsson, P.; Ekstedt, M.; EAF2- A Framework for Categorizing Enterprise Architecture Frameworks, ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 2009. SNPD '09, pp.327 – 332
13. Gruber T.: A Translation Approach to Portable Ontology Specifications. Academic Press Ltd., 1993
14. Wikipedia: Ontology (information science), http://en.wikipedia.org/wiki/Ontology_(information_science)

Project Management Institute Poland Chapter

Literatura

15. Rumbaugh J., Jacobson I., Booch G. Unified Modeling Language Reference Manual, The (2nd Edition), Pearson Higher Education, 2004
16. Protégé: http://protege.stanford.edu/
17. Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net/
18. Euzenat J., Shvaiko P., Ontology matching, Springer Verlag 2007
19. Pinto, H. S., Gómez-Pérez, A., Martis, J. P. Some issues on ontology integration. Proceeding of the IJCAI-99 workshop on Ontologies and Problem-Solving, Stockholm, Sweden, 1999, s. 7.1–7.12.
20. Sowa, J. F. Principles of ontology, onto-std mailing list, Wed, 3 Dec 1997, http://www-ksl.stanford.edu/onto-std/mailarchive/0136.html.
21. Kaneiwa K., Iwazume M., Fukuda K.: An Upper Ontology for Event Classifications and Relations. Australian Conference on Artificial Intelligence 2007: 394-403

Project Management Institute Poland Chapter

Literatura

22. Allen J. F,. Ferguson G.: Actions and events in interval temporal logic. Technical Report TR521, Computer Science Department, University of Rochester, Rochester, New York 14627, July 1994.
23. Cohn M.: Agile Estimating and Planning, Prentice Hall/PTR, 2006, s.330
24. Agile Development Poster, http://pm.versionone.com/AgilePoster.html
25. Ivan Terziev, Atanas Kiryakov, Dimitar Mano, Base upper-level ontology (BULO) Guidance, EU-IST Project IST-2003-506826 SEKT
26. Daniel L. Moody, Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data & Knowledge Engineering 55 (2005) 243–276
27. Guarino, Nicola, Chris Welty, Identity, Unity, and Individuation: Towards a Formal Toolkit for Ontological Analysis, W. Horn, ed., Proceedings of ECAI-2000: The European Conference on Artificial Intelligence. Amsterdam:IOS Press. Pp. 219-223. August, 2000.

Project Management Institute Poland Chapter