Uniwersytet im. Adama Mickiewicza w Poznaniu
Wydział Matematyki i Informatyki

Michał Szymczak

Modelowanie przy pomocy technologii ontologicznych

Praca magisterska pisana pod kierunkiem
prof. dra hab. Zygmunta Vetulaniego.

Poznań, 2006

Podziękowania

Pragnę podziękować prof. dr hab. Zygmuntowi Vetulaniemu za ważne, krytyczne i budujące uwagi: merytoryczne i redaktorskie. Dziękuję również dr. Marcinowi Paprzyckiemu za zarażenie mnie entuzjazmem do tematyki agentów i ontologii oraz zapałem do pracy przy zadaniach związanych z tymi tematami. Ponadto, dziękuję dr. Paprzyckiemu za kluczowe wskazówki merytoryczne. Zespołowi pracującemu nad systemem wspomagania podróży – za pomoc przy tworzeniu i ocenie ontologii opracowywanego systemu, w szczególności Minorowi Gordonowi (z Wydziału Informatyki Politechniki Berlińskiej), Jimmy'emu Wright (z Oklahoma State University w Tulsa, USA) i dr Marii Ganzhe (z Wydziału Administracji Elbląskiej Uczelni Humanistyczno-Ekonomicznej). Kolegom z zespołu: Mateuszowi Kruszykowi za budujące dyskusje na seminariach magisterskich oraz Mladence Vukmirovic i mgr Maciejowi Gawineckiemu za pomysły przy projektowaniu ontologii i ciekawą współpracę.

Michał Szymczak
Poznań, 28 sierpnia 2006 r.

Spis treści

1. Wstęp

Wybór tematu pracy magisterskiej wynika z mojego zainteresowania zagadnieniami sztucznej inteligencji, a w szczególności rozproszonymi systemami wieloagentowymi. Koncepcja Sieci Semantycznej (ang. Semantic Web) zaproponowana przez Berners'a-Lee [W3C, 1998b], [W3C, 1998c], obejmuje swoim zakresem interesujące mnie zagadnienia. Jej podstawą jest inteligentna reprezentacja informacji za pomocą metamodeli danych dostarczonych przez ontologie. Celem wprowadzenia takiego, bazującego na ontologiach, opisu danych jest zwiększenie możliwości rozwoju Internetu pod względem przechowywania i wyszukiwania informacji [Daconta et al., 2003].

Samo zagadnienie ontologii nie jest nowe w nauce. Jest to, w najbardziej ogólnym rozumieniu tego słowa, nauka o bytach, która intrygowała filozofów już w czasach antycznych. Jako gałąź filozofii ontologia rozwija się od 2500 lat. W informatyce pojawiła się bardzo niedawno i od tego czasu jest przedmiotem dyskusji dotyczących zakresu ontologii w informatyce, metod ich tworzenia oraz relacji między ontologiami filozoficznymi a informatycznymi. Analiza roli ontologii w filozofii oraz podejść do tworzenia ontologii stosowanych przez przedstawicieli tej dziedziny nauki dają możliwość dostrzeżenia analogii między ontologiami tworzonymi przez filozofów oraz przez informatyków.

Punktem wyjścia dla rozważań na temat tworzenia ontologii na potrzeby technologii informatycznych w niniejszej pracy magisterskiej będzie zarys różnych podejść do tworzenia ontologii stosowanych przez filozofów. Na ich podstawie zostaną określone podobieństwa i różnice między ontologią klasyczną i informatyczną. To porównanie podkreśla głównie pragmatyczną motywację tworzenia ontologii w informatyce oraz wynikającą z niej rolę tych ontologii. Z uwagi na praktyczny charakter ontologii w informatyce, bardzo istotne są aktualnie stosowane metody i techniki ich budowy. Analiza najbardziej popularnych z tych technik była podstawą do budowy własnej ontologii Systemu Wspomagania Podróży [Angryk et al., 2001], [Gordon i

Paprzycki, 2005], [Gordon et al., 2005]. W ramach pracy magisterskiej powstała ontologia Systemu Wspomagania Podróży, a dokładniej ontologia dziedziny miejsca zakwaterowania, która wraz z ontologiami dziedzin transportu i gastronomii została zintegrowana do ontologii dziedziny podróży. Dobór systemu tworzonego dla użytkowników Internetu jako motywacji dla ontologii dobrze zilustruje teoretyczną i praktyczną stronę procesu budowy ontologii. Dodatkowo, rola ontologii w powyższym systemie pozwoli pokazać nowe cechy i możliwości systemu wynikające z ich zastosowania.

W świetle powyższego wstępu układ zagadnień poruszanych w pracy magisterskiej przedstawia się następująco. W Rozdziale 2 została nakreślona historia rozwoju ontologii jako gałęzi filozofii. W Rozdziale 3 zdefiniowano ontologie w informatyce, opisano ich rolę, rodzaje na przykładach istniejących ontologii, metody tworzenia oraz porównano je z ontologiami klasycznymi. Przytoczono również informacje na temat Sieci Semantycznej, które są wprowadzeniem pozwalającym lepiej zrozumieć potrzebę tworzenia ontologii w informatyce. Rozdział 4 dotyczy technicznej strony rozwoju ontologii w informatyce, jak również języków ontologicznych. Rozdział 5 jest poświęcony przykładowi i metodologii tworzenia ontologii Systemu Wspomagania Podróży. Zawarty jest w nim opis procesu tworzenia ontologii dziedziny podróży (w szczególności dziedziny miejsca zakwaterowania) oraz integracji z dziedzinami gastronomii i transportu. Na potrzeby dalszego wykorzystania opracowanych ontologii w ramach pracy magisterskiej zostało stworzone narzędzie do generowania dokumentacji ontologii, którego opis znajduje się również w Rozdziale 5 i Załączniku C. Żaden ze stworzonych w ramach niniejszej pracy wyników praktycznych nie jest w chwili obecnej publicznie dostępny. W Rozdziale 6 przedstawiam wnioski płynące z poprzednich rozdziałów, z praktycznego zastosowania ontologii oraz podsumowanie wyników własnych.

2. Geneza i rozwój ontologii w filozofii

2.1. Pochodzenie terminu

Termin ontologia ma swoje korzenie w języku greckim, pochodzi od słów ὄντος – istnienie oraz λογία – nauka. Pierwsze wystąpienie łacińskiego odpowiednika (*ontologia*) w literaturze można przyjąć na początek siedemnastego stulecia. Jak jednak wynika z prac Smith'a oraz Øhrstrøm'a [Øhrstrøm et al., 2005], [Smith, 2003] dokładna data pozostaje kwestią sporną.

Barry Smith datuje pierwsze wystąpienie słowa ontologia na rok 1613. Miało się ono pojawić równolegle w dziełach dwóch autorów, które zostały wtedy opublikowane. Pierwszym z nich jest Jacob Lorhard, a jego książka ma tytuł *Theatrum philosophicum*. Drugi z kolei, Rudolf Göckel, profesor logiki na uniwersytecie w Marburgu, w swoim dziele zatytułowanym *Lexicon philosophicum* napomina o ontologii jako filozofii istnienia, w oryginale: *ontologia, philosophia de ente* [Formatology, 2000b], [Smith, 2003]. Øhrstrøm twierdzi, że data podana przez Smith'a jest niezgodna z prawdą, gdyż według niego, słowo ontologia zostało po raz pierwszy wydrukowane w roku 1606 na okładce dzieła Jacoba Lorharda zatytułowanym *Ogdoas Scholastica*, które miało być podręcznikiem do nauki i filozofii opartych na doktrynie wiary protestanckiej. Mimo, iż książka ta nie zawiera dalszych odwołań do tego terminu, to z pozostałej twórczości Lorharda można wywnioskować, że rozumiał on ontologię zgodnie z jej pierwotnym znaczeniem [Formatology, 2000b], [Øhrstrøm et al., 2005].

2.2. Zarys historii rozwoju ontologii w filozofii

2.2.1. Zmienność ontologii w czasie

O ile sam termin został przyjęty na początku siedemnastego wieku, o tyle ontologia jako nauka, zajmująca się istnieniem rozwijała się już od czasów starożytnych. [Leinfallner et al., 1982] opisują ontologię jako naukę podlegającą zmianom pod wpływem rozwoju nauki i techniki. Według nich łączy ona

filozofię z innymi dziedzinami nauki, przy czym nie może istnieć w oderwaniu ani od nich, ani od innych gałęzi filozofii. Ontologia dziedziczy po stanie wiedzy i dorobku naukowym epoki ogólną strukturę świata. Historia filozofii pokazuje, że ontologia nigdy nie rozwiązała ani nie starała się rozwiązać pytań o struktury naszego świata niezależnie od innych gałęzi filozofii lub dyscyplin naukowych [Leinfallner et al., 1982].

Arystoteles jako przedstawiciel metafizyków przedkładał badanie świata rzeczywistego nad postrzeganie świata idei. W myśl jego teorii, ogólne właściwości rzeczy, które stanowią o ich niezmiennej formie, muszą zostać ustalone w oparciu o proces poznawczy. Stworzył on pierwszy system ontologiczny w postaci ontologii istnień [Arystoteles], [Leinfallner et al., 1982]. Natomiast, w przypadku filozofii Christiana Wolffa ontologia ma charakter typowo interdyscyplinarny. Podstawy dla tego typu rozważań dały ogólne zasady logiki opracowane przez Leibniza. Wolff podkreśla potrzebę wyodrębnienia nauki opracowującej warsztat terminologiczny dla wszystkich innych nauk [Leinfallner et al., 1982], [Wolff, 1728], [Formatology, 2000a]. Ontologia Jensa Krafta rozszerza opracowania Wolffa i traktuje ontologię nie tylko jako technikę, lecz bardziej jako środowisko fundamentalnych twierdzeń dotyczących rzeczywistości. Uczeni zajmujący się ontologią w osiemnastym wieku (w tym również Kraft) dostrzegali istotny punkt wyjścia do rozważań nad zagadnieniami religii i etyki właśnie w zrozumieniu rzeczywistości. Wierzyli, że budując odpowiednio ontologię można pomóc ludzkości w tworzeniu lepszego społeczeństwa [Øhrstrøm et al., 2005]. Kolejny ważny krok w rozwoju ontologii był rezultatem rozwoju logiki formalnej oraz filozofii analitycznej, dla których kulminacyjny moment rozwoju przypadł na wiek dwudziesty. Jednym z ważniejszych filozofów tego okresu był Wittgenstein, według którego ontologie szczegółowe dają lepszy obraz rzeczywistości niż ontologie ogólne [Leinfallner et al., 1982]. Podobnie twierdził Quine, według którego każda dziedzina naukowa niesie ze sobą jakąś cząstkę ontologii, która jest tak naprawdę siecią teorii lokalnych (dostarczonych przez konkretne dziedziny). Jednocześnie uznawał on za niewłaściwe wprowadzać porządek

między teorie lokalne w celu unifikacji ontologii, lecz uznawał za konieczne formalizować ontologie dziedzinowe za pomocą logiki pierwszego rzędu [Quine, 1953], [Smith, 2003].

Z przykładów ontologii Krafta i Wolffa wynika, że rolą ontologii było przede wszystkim umożliwić porozumienie między różnymi dyscyplinami, dostarczyć podstawy interdyscyplinarnego konsensusu. Z kolei Wittgenstein i Quine wymagali specjalistycznej wiedzy dyscyplinarnej i gruntownej analizy dziedziny w celu dostarczenia kompletnej ontologii.

2.2.2. Definicja ontologii

Encyklopedia PWN [Encyklopedia, 2005] podaje następujące definicje ontologii.

1. W sensie pierwotnym termin ten był używany od XVII-tego wieku zamiennie ze starszą nazwą metafizyka. Odnosił się do arystotelesowskiej „filozofii pierwszej" jako *nauki o bytach jako bytach*, następnie jej rozważań w teorii bytu zajmującej się bytem w jego naturze powszechnej.
2. W fenomenologii R. Ingardena ontologia jest rozumiana jako odróżniona od metafizyki nauka aprioryczna, która w analizie zawartości idei odkrywa i ustala związki konieczne między czystymi jakościami idealnymi. Ze względu na upowszechnienie fenomenologicznego rozumienia ontologii, w którym utożsamia się ją z analizą pojęć współcześnie wielu filozofów unika używania tego terminu w stosunku do klasycznej metafizyki, będącej analizą realnego bytu.
3. W logice termin *ontologia* oznacza system rachunku nazw Leśniewskiego.

O ile dwie pierwsze definicje pozostają w bliskim związku z filozofią, o tyle trzecie z podanych przez autorów tej encyklopedii znaczeń wydaje się odbiegać od ontologii w pierwotnym rozumieniu tego słowa. Wyjaśnia to autor [Lejewski, 1958], który twierdzi, iż koncepcja logiki Leśniewskiego była

całkowicie ontologiczna w prawdziwie klasycznym rozumieniu i całkowicie uzasadnia takie użycie terminu ontologia w logice.

Dokładna definicja ontologii, podana przez Smith'a i bazująca na definicji Ingardena brzmi następująco [Smith, 2003], [Ingarden, 1964], [Ingarden, 1973]:
Ontologia jest to nauka, gałąź filozofii, traktująca o tym, co jest, o rodzajach i strukturze obiektów, własności, zdarzeń oraz relacji w każdym aspekcie rzeczywistości. Termin ten jest używany niekiedy zamiennie z metafizyką. Z kolei w rozumieniu Ingardena ma on szersze znaczenie i dotyczy również bytów, które mogą istnieć.

Definicja ta jest zbliżona do podanej w encyklopedii PWN [Encyklopedia, 2005]. Według Smith'a ontologia powinna dostarczyć ostatecznych i wyczerpujących klasyfikacji istnień we wszystkich sferach bytu. Ostateczność dotyczy możliwości udzielenia odpowiedzi na pytania: (1) o klasy istnień potrzebne, do dostarczenia kompletnego opisu i wyjaśnienia wszystkich zjawisk we wszechświecie; lub (2) o klasy istnień wymagane do udowodnienia najbardziej podstawowych twierdzeń. Wyczerpujący charakter klasyfikacji ma polegać na uwzględnieniu w niej wszystkich możliwych jednostek, bytów, czy zjawisk [Smith, 2003].

2.2.3. Podejścia do ontologii

Smith wyróżnia dwa podstawowe podejścia do ontologii, na podstawie których można podać najbardziej ogólną klasyfikację ontologii powstających przez 2500 lat historii filozofii. Te podejścia to:

1. Adekwatność;
2. Redukcjonizm.

Adekwatność w przypadku podejścia do ontologii to poszukiwanie systematyki istot w zgodności z rzeczywistością na wszystkich poziomach, od mikrofizycznego po kosmologiczny (przykładami ontologii adekwatnych mogą być ontologie Arystotelesa, Krafta i Wolffa).

Z kolei u podstaw redukcjonizmu leży postrzeganie rzeczywistości przez pryzmat, jednego, uprzywilejowanego istnienia. Taka ontologia powstaje przez dekompozycję rzeczywistości na mniejsze cząstki lub przez redukcję mnogości istnień (do redukcjonistów zaliczani są między innymi Quine i Wittgenstein) [Smith, 2003].

Te dwa podejścia do ontologii wyznaczają dwie podstawowe metody rozwoju ontologii. W pierwszej z nich analizie zostaje poddany ogół rzeczy, a celem jest opis wszystkich istnień. W drugim podejściu do budowy ontologii najważniejsza jest szczegółowa i dokładna analiza konkretnej dziedziny.

Niezależnie od tych podejść, rolą ontologii jest dostarczenie takiego opisu zamierzonej dziedziny lub całego świata, który pozwoliłby budować twierdzenia o danej dziedzinie lub całym świecie. Z kolei wspólne twierdzenie dla dziedziny lub świata mogłyby pozwolić zredukować problemy komunikacyjne [Smith, 2003].

3. Geneza i rozwój ontologii w informatyce

3.1. Definicja ontologii w informatyce

3.1.1. Ontologia

Uważana obecnie za klasyczną i jednocześnie najczęściej przytaczana przez wielu znanych badaczy definicja ontologii w odniesieniu do informatyki została zaproponowana przez Grubera w 1993 roku. Brzmi ona następująco:
Ontologia jest formalną, jednoznaczną specyfikacją dzielonej (wspólnej) konceptualizacji [Gruber, 1993].

Fensel podaje jedną z możliwych interpretacji tejże definicji. Według niego *konceptualizacja* to abstrakcyjny model dowolnego fenomenu świata rzeczywistego, który rozpoznaje istotne koncepty tego fenomenu. *Jednoznaczność* odnosi się do tego, że wykorzystane koncepty oraz sposób ich użycia mają być określone wprost. Z kolei *formalność* ontologii niesie ze sobą możliwość ich odczytania przez system aplikacji [Fensel, 2004].

Z uwagi na fakt, iż elementarnym składnikiem pojęcia ontologii według definicji Grubera jest pojęcie konceptu, wymagane jest zdefiniowanie również tego terminu. Jedną z definicji konceptu podaje Wikipedia:
Koncept to abstrakcyjna, uniwersalna idea, wyobrażenie lub byt, które wyznaczają kategorię lub klasę bytów, zdarzeń lub relacji. Abstrakcyjny charakter konceptu polega na pominięciu różnic między rzeczami znajdującymi się w jednakowym zasięgu [Wikipedia, 2006a].

Według Fensela podstawowym zadaniem ontologii w procesach inżynierii wiedzy jest zapewnienie możliwości konstrukcji modelu pewnej dziedziny. Bowiem ontologia dostarcza zbiór termów i relacji przy pomocy, których można zamodelować tę dziedzinę. [Fensel, 2004].

John Sowa podobnie pojmuje ontologię, a zwłaszcza jej powiązanie z dziedziną, której dotyczy:

Przedmiotem ontologii jest studium kategorii bytów, które istnieją lub mogą istnieć w danej dziedzinie. Produkt tego studium, nazywany ontologią. Jest to katalog typów rzeczy, które zakłada się, że istnieją w rozważanej domenie D z perspektywy osoby, która używa języka L do opisu D [Sowa, 2000].

Z punktu widzenia inżynierii wiedzy, teoria ontologii może być pojmowana w relacji z konkretną dziedziną. Dodatkowo może przyjmować pewien punkt widzenia tej dziedziny, pozwalający określić zadania, które powinna spełnić ontologia w danej dziedzinie [Øhrstrøm et al., 2005].

3.1.2. Przyjęcie ontologii

Aby umożliwić wymianę danych i zasobów oraz usprawnić wyszukiwanie danych ontologie powinny być *przyjmowane* przez strony zainteresowane nowymi możliwościami kooperacji. Jest to odniesienie do *wspólnego* charakteru ontologii określonego przez Grubera.

Ontologia składa się z definicji pojęć i relacji zachodzących między nimi. Mówimy, że agenci (ludzie lub systemy aplikacji) *przyjmują ontologię* (Fensel używa angielskiego terminu: commit themselves to ontology), gdy zgadzają się na logiczne, spójne i konsekwentne korzystanie z niej na potrzeby wzajemnej komunikacji. Agenci dzielący konceptualizacje nie muszą dzielić się całą bazą wiedzy. W rzeczywistości, od agenta, który *przyjął ontologię* nie wymaga się odpowiedzi na wszystkie pytania sformułowane w oparciu o dzieloną konceptualizację [Fensel, 2004], [Gordon et al., 2005].

Aby ontologia mogła zostać przyjęta przez jak najszersze grono agentów często prace nad nią są prowadzone przy współpracy zespołu ludzi, ponieważ wiedza dotycząca konkretnej dziedziny powinna zostać wypracowana w sposób kompromisowy. Nie powinna być narzuconymi ustaleniami jednostki [Fensel, 2004].

3.1.3. Rola ontologii

Podana definicja ontologii Grubera, jej interpretacja przez Fensela oraz podobne do nich spostrzeżenia Sowa'y i Øhrstrøma zakładają spojrzenie na

ontologię przez pryzmat dziedziny, której opis ma ona dostarczyć. Takie rozumienie ontologii różni się od klasycznego. Jej główna rola ma polegać na umożliwieniu usprawnienia komunikacji i wymiany danych z konkretnej dziedziny, a nie dostarczenia opisu wszystkich bytów [Øhrstrøm et al., 2005], [Smith, 2003].

W odniesieniu do definicji ontologii pochodzącej od Grubera, której podstawą jest wyczerpujący opis dziedziny, możemy mówić o *ontologiach kierowanych wymaganiami aplikacji*. W rzeczywistości, dziedzina ontologii będzie zależała od przewidzianych funkcjonalności i wymagań aplikacji, w której ma zostać wykorzystana. Zatem, tworzona konceptualizacja musi spełniać wymagania aplikacji, na której potrzeby budowana jest konkretna ontologia, przedstawiając przy tym niejednostronny schemat opisywanej dziedziny. W odróżnieniu od klasycznego rozumienia ontologii jest to podejście pragmatyczne.

Gruber zaproponował następującą tezę: *W systemach Sztucznej Inteligencji (ang. Artificial Intelligence) istnieje to, co daje się reprezentować* [Gruber, 1995]. W myśl tej tezy ontologie tworzone na potrzeby systemów informatycznych mogą reprezentować byty oderwane od obiektów świata rzeczywistego. Mogą się natomiast obejmować koncepty, języki oraz modele istniejących w ludzkiej wyobraźni (ang. mental model) [Smith, 2003]. Tak, więc, ontologie opracowywane na potrzeby dziedzin, których dotyczą dokumenty dostępne w pewnym stopniu w Internecie, nie muszą reprezentować prawdy o świecie rzeczywistym. Wręcz przeciwnie, mają umożliwić opis istniejących w sieci zasobów, które często nie są wiernymi odbiciami obiektów ze świata rzeczywistego.

Same ontologie zyskały zastosowanie w sztucznej inteligencji by umożliwić współdzielenie i ponowne wykorzystanie wiedzy. Dlatego nie należy mylić ontologii ze schematem baz danych ani modelem programowania obiektowego. Ontologia dostarcza przede wszystkim częściową teorię pewnej dziedziny, która może być *przyjęta* przez ludzi lub aplikacje komputerowe. Z kolei schemat baz danych to opis struktury, w której dane będą

przechowywane. Podobnie w przypadku modelu programowania obiektowego, jest on specyfikacją struktury obiektów określonej dziedziny, jednak ta struktura nie musi oddawać w żaden sposób rzeczywistości, ani tym bardziej nie musi być ustalana i akceptowana przez szersze grono agentów.

3.2. Zastosowanie ontologii w Internecie – idea Sieci Semantycznej

W roku 1989 z inicjatywy Tim'a Berners-Lee rozpoczął się projekt nazwany: Word Wide Web (WWW) [W3C, 1989]. Przyjęty entuzjastycznie przez użytkowników stał się medium, bez którego trudno wyobrazić sobie dzisiejszy świat. Obecnie jest zazwyczaj nazywany Internetem. Rosnąca popularność tego projektu przyczyniła się do ogromnego przyrostu informacji w Internecie. Ilość dokumentów w roku 1995 była szacowana na zaledwie siedem i pół miliona [Webjunction, 1995]. Osiem lat później, w 2003 roku wynosiła w przybliżeniu kilka miliardów [Davies et al., 2003]. Dziewięć lat po stworzeniu projektu sieci WWW pojawiły się pierwsze plany jego rozszerzenia do koncepcji Sieci Semantycznej [W3C, 1998a], [W3C, 1998b], [W3C, 1998c]. Publikacje wprowadzające tę ideę pojawiły się kilka lat później [Berners-Lee et al., 2001], [Fensel, 2004]. Podstawowymi celami Sieci Semantycznej są:

1. Uczynienie z sieci bardziej kolaboratywnego medium.
2. Sprawienie by była zawartość sieci była łatwiej przetwarzana, przez systemy aplikacji komputerowych.

Berners-Lee stwierdza, iż nie chodzi tu o nauczenie komputerów ludzkiej mowy, lecz o dokładniejsze opisywanie tworzonych dokumentów [Daconta et al., 2003]. Najważniejszym składnikiem Sieci Semantycznej, dzięki któremu zaproponowana wizja ma zostać zrealizowana, są *inteligentne dane*.

Pomysł na *inteligentne dane* opiera się na opakowaniu dowolnych danych opisem utworzonym zgodnie z XML, a więc takim, który byłby łatwy do przetworzenia przez program komputerowy. Ontologie dostarczają terminologii na potrzeby takiego opisu. Może być ona zaczerpnięta z różnych, niekoniecznie zależnych od siebie, sklasyfikowanych dziedzin. Dodatkowo,

ontologie muszą wprowadzać relacje logiczne między terminami, tak by zapewnić głęboki poziom analizy danych (najbardziej *inteligentne dane*) oraz usprawnić metody wyszukiwania i inteligentnego wnioskowania przez programy komputerowe.

Tak, więc Sieć Semantyczną można porównać do opartego na ontologiach systemu zarządzania wiedzą, w którym ontologie dostarczają narzędzi do formalnego opisu wiedzy. Na bazie tego opisu aplikacje komputerowe mogą dokonywać wyszukiwania i przetwarzania danych. To właśnie dzięki ontologiom możliwa ma być integracja miliardów istniejących obecnie w sieci WWW heterogenicznych dokumentów [Davies et al., 2003].

Ontologie bywają określane mianem technologicznego rdzenia Sieci Semantycznej. W tym rozumieniu ontologie mają zapewnić współdzieloną interpretację pewnych dziedzin i umożliwić komunikację zarówno między ludźmi jak i systemami aplikacji [Fensel, 2004].

Połączenie wielu dziedzin, zaproponowanie odpowiednich ontologii na tyle ogólnych by można było odnosić się przez nie do jak największej liczby ontologii szczegółowych jest jedną z podstawowych idei Sieci Semantycznej [Daconta et al., 2003].

3.3. Rodzaje i przykłady ontologii w informatyce

Pierwsze ontologie budowane na potrzeby systemów zarządzania wiedzą powstawały od połowy lat 80. XX wieku. Szerokie zainteresowanie tą nauką przekładające się na powstanie licznych ontologii pozwala wyodrębnić ich typy. Poniżej zostają przedstawione i zilustrowane przykładami podstawowe typy ontologii w informatyce [Fensel, 2004], [Guarino, 1998], [Heijst et al., 1997]:

3.3.1. Ontologie ogólne (ang. top-level, generic lub common-sense ontologies)

Ich celem jest ujęcie ogólnej wiedzy o świecie dostarczając podstawowych konceptów dla takich rzeczy jak: czas, przestrzeń, stan, zdarzenie, itd.

[Friedman-Noy i Hafner, 1997]. W konsekwencji są one ważne w wielu dziedzinach. Przykłady:

- Suggested Upper Merged Ontology (SUMO) – ontologia wysokiego poziomu (top-level) rozwinięta przez grupę roboczą IEEE; zawiera definicje ogólnych terminów, które mogą być wykorzystane do stworzenia konkretnych ontologii dziedzinowych; sama ontologia SUMO powstała przez integrację konceptów występujących w: (a) ontologii dostępnych na serwerze Ontolingua [Ontolingua, 1995], (b) ontologii wysokiego poziomu John'a Sowa'y [Sowa, 2000], (c) ontologii wysokiego poziomu Russell'a i Norvig'a [Russel i Norwig, 1995] (d) aksjomów temporalnych James'a Allen'a [Allen, 1984], (e) formalnej teorii dziur Casati'ego i Varzi'niego [Casati i Varzi, 1995], (f) ontologii obszarów granicznych Barry'ego Smitha [Smith, 1994], [Smith, 1995], (g) i różnych reprezentacjacj formalnych opisanych w [Borgo et al., 1996], [Borgo et al., 1997], [Pease, 1997], [Schlenoff et al., 2000]; formalizmem przyjętym do zapisu tej ontologii jest KIF [Niles i Pease, 2001] [SUO WG, 2003].
- WordNet – system referencji leksykalnych stworzony przez naukowców związanych z Princeton University; jest to słownik angielskich wyrazów przyporządkowanych do kategorii odpowiadających częściom mowy; w każdej z tych kategorii wyrazy są zorganizowane według ich znaczenia i połączone relacjami semantycznymi, takimi jak: bycie synonimem, bycie antonimem, bycie hiponimem, bycie częścią (ang. meronymy); według [WordNet, 2006a] obecnie zawiera on ok. 150 tysięcy słów a w porównaniu do tradycyjnych słowników przedstawiających głównie listę słów, dostarcza znacznie więcej gdyż jest oparty na konceptach; głównymi cechami WordNet są: (a) darmowa dostępność w sieci Internet pod adresem [WordNet, 2006b], (b) niezależność od jakiejkolwiek dziedziny, (c) definicja semantyki przy pomocy języka naturalnego, a nie formalnego; mały stopień formalizacji WordNet utrudnia jego

wykorzystanie w systemach aplikacji służących do zarządzania wiedzą [Fensel, 2004].

- CYC – ontologia formalna, powstająca od 1985 roku w ramach badań nad sztuczną inteligencją; jej celem jest dostarczenie wiedzy zdroworozsądkowej systemom aplikacji; architektura tej ontologii oparta jest na grupowaniu pojęć w mikroteorie, które pozwalają określić zależność kontekstową; innymi słowy to, co jest prawdziwe w jednej mikroteorii, nie musi być prawdziwe w drugiej; taki podział pozwala wprowadzić strukturę do globalnej bazy wiedzy dającą spójność i możliwości zarządzania; CYC składa się z setek tysięcy konceptów, ale jedynie część z nich, ok. 3000 jest publicznie dostępna pod nazwą OpenCYC [Gordon et al., 2005], [Fensel, 2004], [CYCORP, Inc., 2002], [OpenCYC, 2002].

3.3.2. Ontologie reprezentacyjne (ang. representational ontologies)

Ontologie te nie dotyczą żadnej konkretnej dziedziny. Definiują jednostki reprezentacyjne i nie specyfikują obiektów, które mogą być nimi reprezentowane.

- *Frame Ontology* – umożliwia opisanie wiedzy w sposób zorientowany obiektowo lub ramowo; zawiera definicje takich konceptów jak: *frame* (odpowiada *klasie* w modelowaniu zorientowanym obiektowo), *slot* (odpowiada *atrybutowi* w modelowaniu zorientowanym obiektowo), *slot constraint* (pozwala definiować warunki dla atrybutów) [Gruber, 1993].

3.3.3. Ontologie dziedzinowe (ang. domain ontologies)

Opisują wiedzę ważną w obrębie konkretnej dziedziny, na przykład: ontologia turystyki, muzyki, kina lub restauracji. Istnieje bardzo wiele przykładów ontologii dziedzinowych, które są dostępne publicznie. Powstają one w ramach projektów akademickich oraz przemysłowych. Poniżej zostały przedstawione wybrane przykładowe ontologie tego typu:

- Dziedzina walut – ontologia serwisu kursów i konwersji walut *Cambia*; szczegółowa ontologia pozwalająca opisać waluty, ich kurs, zadanie konwersji walut po kursach z określonych dni, a nawet subskrypcję takiego zadania konwersji; umożliwia komunikację z tym serwisem; same wiadomości są standardowymi wiadomościami FIPA Agent Communication Language (FIPA ACL) [FIPA, 2002b] z zawartością określoną przez tę ontologię według specyfikacji FIPA SL Content Language [FIPA, 2002c].
- Dziedzina turystyki – ontologia projektu *Harmonize*, którego celem jest umożliwienie organizacjom związanym z turystyką wymianę informacji bez konieczności zmiany ich formatu w lokalnych repozytoriach lub bazach danych; sama ontologia podzielona jest na poddziedziny takie jak: gastronomia, wydarzenie, koncepty geograficzne, polityka, itd.; są one tylko częściowo związane z dziedziną podróży, lecz mogą zostać włączone do ontologii dziedziny podróży w razie potrzeby [Gordon et al., 2005], [DERI, 2004].
- Dziedzina turystyki – ontologia projektu Travel Agent Game in Agentcities (TAGA); jego celem była demonstracja technologii Agentcities oraz Sieci Semantycznej; opracowana w ramach tego projektu ontologia definiuje podstawowe koncepty związane z dziedziną turystyki, na potrzeby środowiska agentowego; niestety mała szczegółowość wyklucza jej zastosowanie jako kompletnej ontologii dziedziny podróży [Gordon et al., 2005], [Agentcities, 2002a].
- Dziedzina turystyki – specyfikacja Open Travel Alliance (OTA); została opracowana by dostarczyć: (a) wspólny język terminologii związanej z podróżowaniem, (b) mechanizm wymiany informacji między członkami organizacji związanych z turystyką; jest próbą stworzenia kompletnej ontologii dziedziny podróży z perspektywy biznesowej; pomimo tego, że słowo ontologia nie zostało użyte w opisie projektu, można dostrzec w specyfikacji OTA cechy komplementarnej ontologii; specyfikacja OTA jest tworzona przez

liderów rynku turystyki oraz przedstawicieli rynku informatycznego takich, jak: IBM, czy Microsoft [Gordon et al., 2005], [OTA, 2001], [OTA, 2006].

- Dziedzina integracji przedsiębiorstw – ontologia TOronto Virtual Enterprise (TOVE) [Fox et al., 1993], [Fox i Gruninger, 1997], [TOVE, 2005] jest przykładem ontologii dziedzinowej oraz ontologii zadania; celem projektu TOVE było stworzenie podstawowego, zdatnego do ponownego użytku modelu danych, który: (a) dostarczy wspólną konceptualizację dla agentów działających w obrębie przedsiębiorstwa, (b) będzie precyzyjnie definiował każdy koncept będący elementem ontologii, (c) formalnie zdefiniuje koncepty umożliwiając automatyczną dedukcję odpowiedzi na pytania dotyczące przedsiębiorstwa, (d) zapewni definicję graficznej reprezentacji konceptów; w konsekwencji TOVE dostarcza możliwą do ponownego zastosowania reprezentację konceptów (ontologię) z dziedziny integracji przedsiębiorstw [Fensel, 2004].
- Dziedzina geodezji – ontologia WGS84 Geo Positioning pozwala opisać wysokość nad poziomem morza, długość i szerokość geograficzną zgodnie ze standardem World Geodetic System 1984 (WGS84) [WGS, 1984], opublikowana przez konsorcjum Word Wide Web, dostarczające światowych standardów w dziedzinie Internetu [W3C, 2004a], [W3C, 2006b].

3.3.4. Ontologie meta danych (ang. Meta data ontologies)

Dostarczają terminologię do opisu zawartości informacji umieszczanej w Internecie.

- Dublin Core Metadata Initiative – otwarte forum rozwoju standardu meta danych informacji przechowywanej w sieci Internet, mające na celu zwiększenie możliwości operowania informacjami i zaspokojenie szerokiej gamy modeli biznesowych [DCMI, 1995]

3.3.5. Ontologie metod i zadań (ang. method and task ontologies)

Ontologie zadań dostarczają konceptualizacje potrzebne do opisu konkretnych zadań. Przykładowo hipoteza jest elementem ontologii diagnozy. Z kolei ontologie metod zawierają terminy pozwalające opisać wybraną metodę rozwiązywania problemów (ang. Problem Solving Method – PSM) [Fensel i Groenboom, 1997], [Studer et al., 1996].

Powyżej przedstawione rodzaje ontologii w informatyce pozwalają zauważyć wspólne kierunki rozwoju ontologii z podejściami stosowanymi przez filozofów. Ta uwaga zostanie rozwinięta w następnej sekcji.

3.4. Podobieństwa ontologii informatycznych i filozoficznych

Na przestrzeni 2500 lat rozwoju ontologii jako gałęzi filozofii pojawiały się twierdzenia o konieczności zbudowania ontologii ogółu bytów świata (Arystoteles, Kraft, Wolff, Ingarden) oraz twierdzenia przeciwstawne, według których powstanie jednej spójnej ontologii *wszystkiego* jest niemożliwe (Wittgenstein, Quine). Podobnie w informatyce, próby stworzenia ontologii ogólnych (CYC, SUMO) mają opozycję w postaci ontologii dziedzinowych oraz opinii o trudności utworzenia jednej ontologii, która spełni wymagania wszystkich dziedzin [Fensel, 2004], [Vetulani, 2004].

Podstawowy podział ontologii w filozofii przebiegał według rozpiętości dziedziny ontologii. Zostały wyróżnione dwa podejścia: adekwatność i redukcjonizm (sekcja 2.2). Analogiczne kierunki można zauważyć w informatyce. Z jednej strony rozwijane są ontologie ogólne i reprezentacyjne (CYC, SUMO), a z drugiej dziedzinowe i zadań (Cambia, Harmonize, TOVE). Strategie rozwoju tych ontologii nazywają się odpowiednio: (a) *od ogółu do szczegółu* (ang. top-down approach) oraz (b) *od szczegółu do ogółu* (ang. bottom-up approach).

Charakterystyczne dla pierwszego z tych dwóch podejść jest rozwijanie ontologii przez znalezienie w pierwszej kolejności takiej ogólnej klasyfikacji bytów, która pozwoli opisać każdy szczegół w dowolnej dziedzinie. Innymi

słowy, polega na rozwinięciu ontologii szczegółowych na podstawie stabilnej ontologii ogólnej, wspólnej dla wszystkich dziedzin.

Z kolei podejście *od szczegółu do ogółu* wymaga zastosowania odwrotnej kolejności. Na początku rozwijane są ontologie dziedzinowe i dopiero na ich podstawie zostają wyciągnięte wnioski na temat postaci ontologii ogólniejszej.

Scharakteryzowane powyżej dwa podejścia są punktem wyjścia dla przeprowadzenia analizy bardziej szczegółowych metod rozwoju ontologii w informatyce.

3.5. Rozwój ontologii w informatyce

Rola ontologii w informatyce wynikająca z ich zastosowania w systemach aplikacji i Sieci Semantycznej polega przede wszystkim na dostarczeniu precyzyjnych, *przyjmowanych* przez różnych agentów ontologii szczegółowych oraz będących dla nich spoiwem ontologii ogólnych (sekcja 3.2). Aby wyznaczona rola została spełniona potrzebne są poprawnie określone metody budowy ontologii. Smith proponuje wzięcie pod uwagę następujących ogólnych punktów kluczowych dla metod rozwoju ontologii w informatyce [Smith, 2003]:

- zmniejszenie nacisku na oparte na konceptualizacji tworzenie substytutów obiektów może przynieść wymierne korzyści praktyczne.
- dobre konceptualizacje to takie, które przejrzyście oddają rzeczywistość i mają szanse być zintegrowane do jednego uniwersalnego systemu ontologicznego.
- dobre konceptualizacje korzystają z dorobku nauk przyrodniczych.
- tworzenie globalnej ontologii musi być iteracyjnym procesem składającym się z faz konstrukcji, krytyki, testów i poprawiania.

3.5.1. Podejście mieszane

Określone w sekcji 3.4 dwa podstawowe podejścia do procesu tworzenia ontologii są podejściami skrajnymi. Dzięki temu można łatwo określić ich zalety i wady.

Przy zastosowaniu podejścia *od ogółu do szczegółu* można mieć pewność, że określona ontologia ogólna będzie miała zamierzony kształt i pozwoli przeprowadzać wnioskowanie między dziedzinami szczegółowymi. Jednak, istnieje ryzyko niemożliwości znalezienia zadowalającej ontologii ogólnej. Metoda ta była jedną ze stosowanych przez filozofów zajmujących się ontologiami w 2500 letniej historii zachodniej filozofii i do tej pory nie pozwoliła wypracować takiej ontologii ogólnej, na którą zgodziłaby się większość filozofów [Niles i Pease, 2001]. Ponadto Fensel uważa za mało prawdopodobne powstanie jednej ontologii, która spełniłaby wymagania wszystkich kontekstów aplikacji i dziedzin. Podobnie wątpliwe jest w jego opinii by jedna ogólna ontologia została *przyjęta* przez tak szerokie grono, jakim jest społeczność internetowa [Fensel, 2004].

Z kolei rozpatrując podejście *od szczegółu do ogółu*, istnieje ryzyko niemożliwości wykonania ontologii ogólnej poprzez generalizację konceptów, jako powód zostaje podany brak ujednoliconej i powszechnie akceptowalnej metody tworzenia ontologii dotyczących konkretnych dziedzin (rozwijane są one z myślą o konkretnej dziedzinie i jej wewnętrznych potrzebach). Zdecydowaną zaletą tej metody jest dobra realizacja ontologii dziedzinowych [Russel i Norwig, 1995].

Ze względu na wady i zalety obu przedstawionych powyżej podejść autorzy [Vetulani, 2004] i [Niles i Pease, 2001] proponują podejście *mieszane* wykorzystujące zalety obydwu podejść, pozwalające na określenie właściwej ontologii ogólnej oraz dokładnych ontologii szczegółowych. W ten sposób powstała między innymi ontologia SUMO. W pierwszej kolejności zostały określone kategorie ogólne, pod które były włączane kategorie coraz bardziej szczegółowe. Jednocześnie miała miejsce redukcja niepotrzebnych kategorii ogólnych i ewentualne dodawanie nowych. Ostatecznie ontologia wysokiego poziomu ma spełnić funkcję wieszaka, na którym można odnaleźć każdą z ontologii dziedzinowych [Niles i Pease, 2001]. Podobnie *systemy ontologiczne* przedstawione w [Vetulani, 2003b] dotyczą systemu złożonego z ontologii ogólnych oraz ontologii szczegółowych. W takim systemie ontologie

szczegółowe powinny być kompatybilne z ontologią ogólną oraz powinny wyróżniać się między sobą systemem kategorii i relacji. Vetulani postuluje potrzebę lingwistycznej motywacji dla budowy ontologii ogólnej [Vetulani, 2003a]. Kategorie ontologii ogólnej powinny w jak najmniejszym stopniu umowne. Najważniejszą funkcją języka jest umożliwienie komunikacji, a w szczególnym przypadku umożliwienie wymiany informacji dotyczących świata. To właśnie z systemu znaków konkretnego języka powinna zostać zaczerpnięta wiedza o strukturze pojęciowej wspólnej dla społeczności posługującej się danym językiem. Vetulani uważa, że pomimo różnic występujących nawet między językami w zbliżonych kulturach, istnieje możliwość stworzenia w oparciu o kategorie pojęciowe wybranego języka ontologii ogólnej niezależnej od konkretnego języka [Vetulani, 2004].

3.5.2. Dynamiczne sieci znaczeń

Rozszerzenie podejść *od ogółu do szczegółu* i *od szczegółu do ogółu* zostało zaproponowane w [Fensel, 2004]. Fensel formułuje następujące wymagania funkcjonalne dotyczące ontologii:

- Umożliwienie przetworzenia informacji przez systemy aplikacji (definicja formalnej semantyki dla informacji).
- Umożliwienie połączenia zawartości przetwarzalnej przez systemy aplikacji ze znaczeniem zrozumiałym dla człowieka, przez wspólną terminologię (definicja semantyki dla świata rzeczywistego).

W skrócie, ontologie są rozwijane po to, by zapewnić dzielenie oraz ponowne wykorzystanie (ang. sharing and reuse) wiedzy między agentami, niezależnie od ich ludzkiej lub komputerowej (ang. artificial) natury. Spełnienie powyższych wymagań jest możliwe, według Fensela, przez połączenie dwóch cech na etapie projektowania, tworzenia i wdrażania ontologii:

1. Architektura sieci znaczeń

   Ontologie powinny mieć architekturę sieci złożonej z węzłów będących ontologiami dziedzinowymi o lokalnym zasięgu. Narzędzia rozwiązujące konflikty między heterogenicznymi definicjami i wspierające *przeplatanie*

(ang. interweaving) lokalnych teorii są podstawą działania i skalowalności tej architektury. Ich działanie powinno polegać na dopuszczeniu i unifikacji pokrywających się ontologii lokalnych opierających się nawet na sprzecznych konceptualizacjach. Aby możliwa była wymiana informacji niezależnie od dziedziny, zadania czy barier socjologicznych odseparowane lokalne teorie muszą zostać przeplecione. W szczególności potrzebne są narzędzia służące do: (a) definiowania lokalnych modeli dziedzinowych, (b) przeplatania modeli lokalnych między sobą. Nowo tworzone ontologie nie powinny istnieć odrębnie, lecz winny zostać przyłączone do sieci, w której możliwe będzie istnienie pokrywających się konceptualizacji.

2. Dynamika

Obecnie ontologie, gdyby miały być wykorzystywane do wymiany informacji między agentami (ludzkimi bądź komputerowymi), powinny odzwierciedlać porozumienie międzydziedzinowe. Istotne są dwie strony takiego konsensusu: (a) ontologie są jego przesłanką oraz (b) ontologie są jego rezultatem. Żadnego z powyższych aspektów nie należy ignorować, gdyż są one równoważnymi warunkami rozwoju ontologii. W celu rozwijania ontologii potrzebne są protokoły procesu rozwojowego a ich kluczowymi elementami powinny być:

- Polityka wersji dokumentów – zapewnia stabilność systemu przy ciągłych zmianach wersji ontologii; po pojawieniu się nowej wersji ontologii *przyjęte* wcześniej wersje tej ontologii będą mogły być nadal wykorzystywane, co jest niezbędne do poprawnego działania zależnych o nich aplikacji.
- Modele procesu biznesowego rozwoju ontologii – pozwalają ustalić przebieg *przyjmowania* ontologii przez różnych agentów w sposób abstrahujący od szczegółów budowy danej ontologii; zwiększają możliwość osiągnięcia konsensusu między dowolnymi agentami.

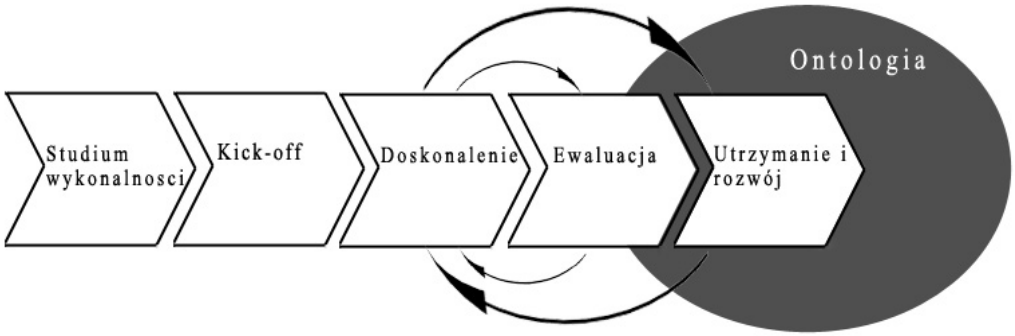
Podejście *dynamicznych sieci znaczeń* polega na projektowaniu i tworzeniu takich ontologii dziedzinowych, które po wdrożeniu w różnych systemach o zasięgu lokalnym będą mogły zostać wykorzystane na potrzeby innych systemów. Nadawanie numerów wersji ontologiom zwiększy stabilność

systemów korzystających z konkretnej ontologii systemów i pozwoli rozwiązywać ewentualne błędy występujące podczas uaktualniania ontologii. Podejście dynamicznych sieci znaczeń, kładzie duży nacisk na powstanie narzędzi umożliwiających wielokrotne definiowanie różnych pojęć.

Połączenie architektury sieci i dynamiki dodaje ontologii wartość możliwości zmienności w czasie. Dzięki temu podejście to spełnia podstawowe założenie ontologii w filozofii dotyczące zmienności ontologii w czasie, jej uzależnienia od stanu wiedzy. Przy czym należy podkreślić, że podejście *dynamicznych sieci znaczeń* w żaden sposób nie ogranicza możliwości konstrukcji ontologii dziedzinowych i ogólnych. Z tych względów teoria Fensela może być dobrą podstawą rozważań nad metodami budowy ontologii.

3.5.3. Cykl życia ontologii (ang. ontology life-cycle)

Dynamiczny charakter rozwoju ontologii oddaje również przedstawiona w książce [Sure i Studer, 2003] metoda rozwoju ontologii uzależnionej od potrzeb aplikacji (ang. application driven ontology development) – czyli ontologii dziedzinowej. Polega na wkomponowaniu procesu budowy ontologii w pięć etapów przedstawionych na Rysunku 3.1 powszechnych w rozwoju aplikacji.



Rysunek 3.1. Cykl życia ontologii.

Cykl życia ontologii jest oparty o powszechnie stosowaną metodę rozwoju produktów, w tym aplikacji komputerowych. Ontologie będące produktami informatycznymi, powstającymi na potrzeby konkretnych aplikacji powinny spełniać stawiane im przez te aplikacje wymagania. *Cykl życia ontologii*

określa etapy tworzenia ontologii i przeprowadzane w ich ramach procesy. Oto opis zadań i procesów wykonywanych w ramach *cyklu życia ontologii*:

1. Studium wykonalności

   Pozwala zbudować podstawę decyzyjną w sprawach ekonomicznych i technologicznych oraz związanych z wykonalnością całego projektu, którego część stanowi opracowywana ontologia. W szczególności w ramach tego zadania podejmowane są decyzje o zasadności tworzenia danej ontologii, wyborze narzędzi do jej rozwoju oraz doborze ludzi mających pracować nad nią.

2. Kick-off

   Głównym produktem tej fazy jest dokument zawierający specyfikację wymagań ontologii (ang. Ontology Requirements Specification Document; w skrócie ORSD). Powinien precyzować następujące zagadnienia:

   - Dziedzina i cel ontologii;
   - Wskazówki projektowe;
   - Źródła wiedzy;
   - (Potencjalnych) użytkowników i scenariusze wykorzystania;
   - Kompetencje ontologii;
   - Aplikacje wspierane przez ontologię.

3. Doskonalenie

   W ramach tego etapu powstaje dojrzała, formalna i zorientowana na aplikację ontologia. Kluczowe w ramach tej fazy są dwa procesy:

   - Wydobycie wiedzy z dokumentacji ORSD;
   - Pozyskanie wiedzy od ekspertów w dziedzinie, która jest przedmiotem danej ontologii;
   - Sformalizowanie ontologii.

4. Ewaluacja

   Pozostaje w ścisłym związku z poprzednim etapem. Polega na ocenie technicznej ontologii opracowanej w ramach Doskonalenia oraz dokumentacji powstałej w wyniku faz poprzednich. Powinna być połączeniem analizy zgodności stworzonej ontologii z dokumentacją oraz

raportami z testów prototypu systemu. Im więcej elementów analizy tym bardziej wartościowe dane wejściowe mogą być dostarczone dla ponownego przeprowadzenia etapu Doskonalenia.

5. Utrzymanie i rozwój

   Jest to w dużej mierze proces organizacyjny polegający na zastosowaniu odpowiedniej polityki zmiany wersji, dodawania lub usuwania ontologii. Istotne są na tym etapie również decyzje personalne, co do osób zajmujących się utrzymaniem i rozwojem danej ontologii.

Metoda *cyklu życia ontologii* jest specyfikacją podejścia *dynamicznych sieci znaczeń* dla ontologii dziedzinowych. Określa iteracyjny proces rozwoju ontologii dziedzinowej. Etapy Doskonalenia, Ewaluacji oraz Utrzymania i Rozwoju są kolejnymi iteracjami procesu tworzenia wiarygodnej ontologii. Metoda ta podkreślając potrzebę wprowadzania ciągłych ulepszeń do samej ontologii, zwiększa *przyjęcia* jej przez możliwie dużą grupę agentów. Na podstawie *cyklu życia ontologii* mogą powstawać nie tylko ontologie dziedzinowe, lecz również środowiska wspomagające tworzenie ontologii dziedzinowych.

3.5.4. Podsumowanie metod

Wybór wyłącznie metody *od ogółu do szczegółu* niesie zbyt duże ryzyko nie spełnienia wymagań dziedziny. Z kolei samo *od szczegółu do ogółu*, choć zakłada dokładne spełnienie tych wymagań, to nie zapewnia ani integracji dziedzin, ani dalszego rozwoju konceptualizacji w czasie. Metoda *mieszana* zwiększa możliwość przeprowadzenia integracji ontologii w celu wierniejszego odzwierciedlenia rzeczywistości. Metoda ta, podobnie jak *dynamiczna sieć znaczeń* i *cykl życia ontologii* zakładają dogłębną analizę i spełnienie wymagań dziedziny przez wykorzystanie odkryć nauk specjalistycznych w zakresie danej dziedziny. *Dynamiczna sieć znaczeń* dodaje pogłębianie analizy istniejącej ontologii określając jej dynamiczny charakter. *Cykl życia ontologii* precyzuje ten aspekt definiując w trzech ostatnich etapach iteracyjny proces *polepszania* ontologii. Dodatkowo *dynamiczna sieć znaczeń* daje możliwość stworzenia bardzo bogatej ontologii, oddającej jak najlepiej

rzeczywistość. Architektura sieci, ułatwia wprowadzenie wszelkich rozszerzeń, dodawanie nowych ontologii lokalnych (dziedzinowych) oraz wykorzystywanie istniejących. Węzłami *dynamicznej sieci znaczeń* powinny być ontologie dziedzinowe. *Cykl życia ontologii* może być dla takiej sieci metodą tworzenia ontologii-węzłów.

4. Technologia budowy ontologii

Poprzednie rozdziały dotyczyły teorii ontologii i metod ich tworzenia. Ontologie tworzone na potrzeby aplikacji komputerowych wymagają formalnego języka, przy pomocy którego można je budować [Fensel, 2004]. W niniejszym rozdziale zostaną zanalizowane obecne standardy internetowe na bazie języka eXtensible Markup Language (XML) oraz ich przydatność ze względu na tworzenie ontologii.

4.1. XML

Sam język XML służy do opisu dokumentów z wykorzystaniem prostej składni opartej na znacznikach (ang. Tag). Z kolei przy pomocy XML Schema (XSD) można definiować typy i struktury opisywanych danych. Jest to aktualne podejście do określania struktury dokumentów XML. Wypiera ono wykorzystywane do niedawna Dokument Type Definition (DTD) [Baclawski, 2006], [W3C, 2000], [W3C, 2004b], [W3Schools, 1999].

Dokumenty XML mają strukturę drzewa, jednak hierarchia przedstawiona w takim dokumencie nie ma określonego znaczenia. Może ona oznaczać zarówno podzbiór, podklasę, członkostwo, lub innego typu relację [Baclawski, 2006].

W XML istnieje dziedziczenie typów, jednak nie można określić dziedziczenia wielobazowego. Dodatkowo każde dziedziczenie musi zostać zamodelowane wprost. Ważną cechą języka XML i XSD jest to, że nie można w nim zamodelować wprost relacji *is-a*. Relacja ta określa dziedziczenie atrybutów przez klasy potomne po klasach bazowych oraz dziedziczenie instancji klas przez klasy bazowe po klasach potomnych. W XSD powyższe aspekty relacji *is-a* mogą być jedynie zamodelowane sztucznie [Fensel, 2004]. Dodatkowym jego mankamentem jest mała elastyczność dokumentów XML w stosunku do schematu struktury skojarzonego z danym dokumentem [Baclawski, 2006].

4.2. Resource Description Framework (RDF)

Standard ten służy do zapisu reprezentowania informacji w Internecie i posiada składnię zdefiniowaną w języku XML, która umożliwia definiowanie znaczenia opisywanych obiektów bez potrzeby określania struktury dokumentu. Jest to infrastruktura, która umożliwia zakodowanie, wymianę i ponowne wykorzystanie strukturalnych metadanych. Silniki wyszukiwarek, inteligentni agenci, brokerzy informacji, przeglądarki oraz ludzie mogą wykorzystać informację semantyczną dostarczoną w formacie RDF. Model danych RDF jest oparty na trzech typach obiektów: *podmiotach*, *orzeczeniach* i *dopełnieniach* ustawionych w uporządkowane trójki [Brickley et al., 1998].

- *Podmiot* jest jednostką (ang. entity), do której można odwołać się przez adres Uniform Resource Locator (URL) lub Uniform Resource Identifier (URI) [Wikipedia, 2006b], [Wikipedia, 2006c]. Zasoby są elementami opisywanymi w zdaniach RDF.
- *Orzeczenie* definiuje relację binarną między zasobami i/lub wartościami atomowymi zdefiniowanymi jako podstawowe typy danych XML.
- *Dopełnienie* specyfikuje wartość orzeczenia dla danego podmiotu.

Przykład RDF w uproszczonej notacji *orzeczenie*(*podmiot*) = *dopełnienie*.

```
Autor(http://atos.wmid.amu.edu.pl/~d124124) = X
Imię(X) = Michał
Email(X) = d124124@atos.wmid.amu.edu.pl
```

Gdzie *X* oznacza aktualny lub wirtualny URI Michała. Tak zapisana informacja oznacza, że Autorem strony internetowej o podanym w pierwszym zadaniu adresie jest zasób X, jego imię to Michał a adres email to d124124@atos.wmid.amu.edu.pl.

Do opisu zasobów w języku RDF można wykorzystywać słowa pochodzące z dowolnych przestrzeni nazw, mających własne URI. Same terminy są najczęściej zdefiniowane za pomocą języka RDF Schema (RDFS).

4.2.1. RDFS

Język ten służy do opisu terminów, których można używać podczas tworzenia dokumentów RDF. Innymi słowy, daje możliwość specyfikacji terminów, które posłużą jako konkretne części zdania w opisie zasobów w dokumencie RDF. Słowom tym można nadawać znaczenie i definiować relacje między nimi. Dodatkowo, daje możliwość umieszczania komentarzy w języku naturalnym dla każdego wprowadzanego terminu.

Takie zestawy terminów można budować przez wykorzystanie podstawowych klas, atrybutów i ograniczeń określonych w RDFS.

Podstawowymi klasami (dopełnieniami) są:

- *Resource* – klasa wszystkich podmiotów;
- *Property* – klasa wszystkich orzeczeń;
- *Class* – klasa wszystkich dopełnień;
- *Literal* – klasa wartości dosłownych (łańcuch lub liczby);
- *Datatype* – klasa typów danych, podklasa *Literal*;
- *XMLLiteral* – klasa wartości dosłownych określonych w specyfikacji XML Schema.

Podstawowe atrybuty (orzeczenia) to:

- *subClassOf* – relacja bycia podklasą;
- *subPropertyOf* – relacja bycia podatrybutem;
- *range* – relacja definiująca ograniczenie zbioru dopełnień do określonych klas dla słowa zdefiniowanego jako orzeczenie;
- *domain* – relacja definiująca ograniczenie zbioru podmiotów do określonych klas dla słowa zdefiniowanego jako orzeczenie;
- *type* – relacja określająca klasę danego podmiotu;
- *label* – relacja określająca opis w języku naturalnym;
- *comment* – relacja określająca opis w języku naturalnym.

Dodatkowo w definiowaniu konceptualizacji można wykorzystać terminy z istniejących konceptualizacji zewnętrznych [W3C, 2004c].

Przykład definicji pokoju hotelowego w RDFS:

Standardowy nadgłówek XML:

```
<?xml version="1.0"?>
```

Deklaracja wykorzystywanych konceptualizacji, ich przestrzeni nazw:

```
<rdf:RDF
    xmlns:rvt="file:///E:/Ontologies/HotelInfrastructureCodes/RoomViewType#"
    xmlns:rlt="file:///E:/Ontologies/Location/IndoorLocation#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:rmt="file:///E:/Ontologies/HotelInfrastructure/RoomType#"
    xmlns:base="file:///E:/Ontologies/HotelInfrastructure/HotelRoom#">
```

Definicja klasy pokoju hotelowego, dziedziczącej po klasie typu pokoju wraz z komentarzem w języku naturalnym:

```
<rdf:Description
rdf:about="file:///E:/Ontologies/HotelInfrastructure/HotelRoom#HotelRoom">
     <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
     <rdfs:subClassOf
rdf:resource="file:///E:/Ontologies/HotelInfrastructure/RoomType#RoomType"/>
     <rdfs:comment>Record of a particular hotel room.
                        Its location, type, etc.</rdfs:comment>
</rdf:Description>
```

Definicja atrybutu numeru dla klasy pokoju hotelowego o wartościach z nieujemnych całkowich:

```
<rdf:Description
rdf:about="file:///E:/Ontologies/HotelInfrastructure/HotelRoom#number">
     <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
     <rdfs:comment>Hotel room number.</rdfs:comment>
     <rdfs:domain
rdf:resource="file:///E:/Ontologies/HotelInfrastructure/HotelRoom#HotelRoom"/>
     <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>
</rdf:Description>
```

Definicja atrybutu lokalizacji wewnętrznej dla klasy pokoju hotelowego o wartościach będących instancjami klasy IndoorLocation:

```
<rdf:Description
rdf:about="file:///E:/Ontologies/HotelInfrastructure/HotelRoom#roomLocation">
     <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
     <rdfs:comment>Location on the site.</rdfs:comment>
     <rdfs:domain
rdf:resource="file:///E:/Ontologies/HotelInfrastructure/HotelRoom#HotelRoom"/>
     <rdfs:range
rdf:resource="file:///E:/Ontologies/Location/IndoorLocation#IndoorLocation"/>
</rdf:Description>
```

Definicja atrybutu widoku z okna pokoju dla klasy pokoju hotelowego o wartościach będących instancjami klasy RoomViewType:

```
<rdf:Description
rdf:about="file:///E:/Ontologies/HotelInfrastructure/HotelRoom#roomViewType">
     <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
     <rdfs:comment>Room view information.</rdfs:comment>
     <rdfs:domain
rdf:resource="file:///E:/Ontologies/HotelInfrastructure/HotelRoom#HotelRoom"/>
     <rdfs:range
rdf:resource="file:///E:/Ontologies/HotelInfrastructureCodes/RoomViewType#RoomV
iewType"/></rdf:Description></rdf:RDF>
```

4.2.2. N3

Ten język stanowi zwięzłą alternatywę dla RDF zapisanego w składni XML wprowadzając jednocześnie kilka elementów zwiększających jego ekspresywność. Podstawą tego języka są zdania oparte na trójkach uporządkowanych analogiczne z trójkami RDF. Z kolei głównymi celami N3 są [W3C, 1998d]:

- Optymalizacja wyrażania opisu danych i logiki w tym samym języku.
- Ujęcie i wyrażenie wszystkich elementów języka RDF za pomocą uproszczonej składni.
- Cytowanie umożliwiające budowanie zdań o zdaniach (których podmiotem lub dopełnieniem są inne zdania).
- Przejrzystość zapisu i intuicyjność interpetacji odczytu.

Te cele zostały osiągnięte przez:

- Umożliwienie skrótowego zapisu URI z wykorzystaniem prefiksów (podobnie do przestrzeni nazw w XML).
- Dodanie wielu dopełnień dla danej pary podmiotu i orzeczenia przez wykorzystanie przecinka.
- Dodanie wielu orzeczeń do danego podmiotu z wykorzystaniem średnika.
- Tworzenie pustych węzłów (ang. blank node) z określonymi orzeczeniami (w takim zdaniu brak konkretnego podmiotu, istnieją wyłącznie orzeczenia). Pusty węzeł można porównać do zdań z kwantyfikatorem egzystencjonalnym.
- Tworzenie formuł składających się z wielu zdań N3.
- Wprowadzenie zmiennych i kwantyfikatorów do tworzenia reguł.
- Budowę prostej i logicznej gramatyki.

Przykład definicji pokoju hotelowego w notacji N3:

Deklaracja wykorzystywanych konceptualizacji, ich przestrzeni nazw:

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rlt: <Location/IndoorLocation#>.
@prefix rmt: <HotelInfrastructure/RoomType#>.
@prefix rvt: <HotelInfrastructureCodes/RoomViewType#>.
```

```
@prefix base: <HotelInfrastructure/HotelRoom#>.
```

Definicja klasy pokoju hotelowego, dziedziczącej po klasie typu pokoju wraz z komentarzem w języku naturalnym:

```
base:HotelRoom a rdfs:Class;
     rdfs:subClassOf rmt:RoomType;
     rdfs:comment "Record of a particular hotel room.Its location, type, etc.".
```

Definicja atrybutu numeru dla klasy pokoju hotelowego o wartościach ze zbioru liczb nieujemnych całkowich:

```
base:number  a rdf:Property;
     rdfs:comment "Hotel room number.";
     rdfs:domain base:HotelRoom;
     rdfs:range xsd:nonNegativeInteger.
```

Definicja atrybutu lokalizacji wewnętrznej dla klasy pokoju hotelowego o wartościach będących instancjami klasy IndoorLocation:

```
base:roomLocation  a rdf:Property;
     rdfs:comment "Location on the site.";
     rdfs:domain base:HotelRoom;
     rdfs:range rlt:IndoorLocation.
```

Definicja atrybutu widoku z okna pokoju dla klasy pokoju hotelowego o wartościach będących instancjami klasy RoomViewType:

```
base:roomViewType  a rdf:Property;
     rdfs:comment "Room view information.";
     rdfs:domain base:HotelRoom;
     rdfs:range rvt:RoomViewType.
```

4.3. Web Ontology Language (OWL)

OWL, podobnie jak RDF, jest językiem dedykowanym dla aplikacji, które wymagają przetworzenia zawartości dokumentu, a nie tylko jej prezentacji dla ludzi. Jest oparty na RDF i podobnie do RDFS pozwala definiować wyrazy oraz relacje między nimi. Jednak, przede wszystkim rozszerza RDF wprowadzając następujące elementy:

- Ograniczenia mocy zbioru wartości poszczególnych atrybutów klas.
- Specyfikacje typów atrybutów.
- Atrybuty tranzytywne i symetryczne, np. tożsamość (org. sameAs).
- Relacje między klasami (np. suma, część wspólna, rozłączność, równość, liczebność).
- Klasy wyliczeniowe.
- Metadane ontologii.
- Rozszerzania innej ontologii deklarowane wprost.

OWL występuje w trzech wersjach o odpowiednio malejącej ekspresywności i rosnącej wydajności przetwarzania: Full, DL, Lite [Baclawski, 2006], [W3C, 2004d], [W3C, 2004e].

4.3.1. OWL Full

Klasy zdefiniowane w tym wariancie OWL są traktowane jako równoważne z klasami zdefiniowanymi w RDF. Dodatkowo możliwe jest wykorzystanie wszystkich konstrukcji ograniczeń mocy zbiorów OWL. Takie połączenie sprawia, że czas przetwarzania dokumentu napisanego w takim języku jest w najgorszym przypadku nieskończony.

4.3.2. OWL DL

Pozwala korzystać z dowolnych konstrukcji językowych OWL, jednak pod ścisłymi warunkami, gwarantując tym samym skończony czas przetwarzania wszystkich dokumentów OWL DL. Nie można nic więcej powiedzieć o tym czasie przetwarzania [Baclawski, 2006]. Nie wszystkie klasy RDF można reprezentować w tym wariancie OWL.

4.3.3. OWL Lite

Narzuca większe ograniczenia w korzystaniu z konstrukcji językowych niż OWL DL, dzięki temu czas przetwarzania dokumentów OWL Lite jest w najgorszym przypadku wykładniczy. Podobnie jak w przypadku OWL DL, nie wszystkie klasy RDF można reprezentować w tym wariancie OWL.

4.4. XML, RDF(S) i OWL a ontologie

Model dziedziczenia w języku XML jest zdecydowanie uboższy niż w RDF(S) i OWL. Mając dany dokument XML oraz schemat jego struktury zdefiniowany w XSD, nie można przeprowadzić jednoznacznego wnioskowania. Jeśli ontologia miałaby zostać przedstawiona za pomocą XML i XSD języki te musiałyby zostać wzbogacone o definicje wyrażeń prostych i sposób ich reprezentacji w liniowej składni XML [Baclawski, 2006], [Fensel, 2004], [Szymczak et al., 2006].

RDF(S) dostarcza gotowe rozwiązanie. Definiuje prymitywy potrzebne do budowy ontologii i dostarcza składnię do jej konstrukcji w oparciu o język XML. Zatem w języku RDF(S) można tworzyć ontologie, a wykorzystanie XML do pisania ontologii wymagałoby stworzenia duplikatu RDF(S) [Fensel, 2004].

Podstawowym ograniczeniem RDF(S) jest brak standardu opisu aksjomatów logicznych. Można w tym języku definiować klasy i atrybuty, lecz nie ma możliwości definiowania złożonych relacji przy pomocy aksjomatów. W związku z tym, ontologie tworzone wyłącznie przy pomocy mechanizmów dostarczonych przez RDF(S) są formalnymi teoriami dziedzin, które nie zawierają żadnych informacji o konceptach oprócz:

- Definicji klas i atrybutów.
- Hierarchii klas i atrybutów.
- Komentarzy w języku naturalnym dla dowolnych klas, atrybutów lub instancji.

Powyżej wymienione braki języka RDF(S) wypełnia język OWL, który pozwala definiować teorie o większej ekspresywności wyrażeń. OWL rozszerza język RDF o elementy wymienione w sekcji 3 tego rozdziału. Zatem, ontologie napisane w tym języku są formalnymi teoriami dziedzin, które oprócz informacji zapewnianych przez mechanizmy RDF(S) dają dodatkowo możliwość definiowania:

- Relacji złożonych między klasami w oparciu o działania na zbiorach.
- Szczegółowych informacje o wersji ontologii (metadane ontologii).
- Ograniczeń liczebności klas.
- Ograniczeń liczebności wartości atrybutów.
- Rozszerzalności ontologii zewnętrznej.

W obu językach: RDF(S) i OWL można napisać ontologię, decyzja o wyborze języka zależy wyłącznie od potrzeb aplikacji, w której ta ontologia ma zostać wykorzystana.

5. Opracowana ontologia

Ontologia powstająca w ramach tej pracy magisterskiej ma zostać wykorzystana w Systemie Wspomagania Podróży (ang. Travel Support System – TSS). W ogólnym ujęciu jest to rozproszony system agentowy dający możliwość zaplanowania podróży w oparciu o dane dostępne obecnie w Internecie [Angryk et al., 2001], [Gordon i Paprzycki, 2005].

Układ tego rozdziału jest następujący: sekcja 1 zawiera opis systemu TSS, jego wymagań oraz roli, jaką spełnić ma w nim tworzona ontologia; w następnej kolejności zostaje przedstawiona metodologia jej budowy; sekcja 3 to szczegółowy opis zastosowania powyższej metodologii w praktyce do konstrukcji ontologii systemu TSS.

5.1. System Wspomagania Podróży

5.1.1. Cel projektu

Obecnie, w Internecie znajduje się bardzo duża ilość informacji związanych z dziedziną podróży. Chcąc jednak zaplanować podróż wyłącznie w oparciu o dane dostępne w Internecie należy poświęcić temu zdecydowanie więcej czasu, niż zlecając organizację wyjazdu. Dane dotyczące środków transportu, miejsc zakwaterowania i lokali gastronomicznych są przedstawione bardzo różnorodnie i znalezienie interesujących informacji okazuje się w wielu wypadkach trudne. Chcąc przykładowo zorganizować wycieczkę po krajach śródziemnomorskich, szukając miejsc zakwaterowania takich jak schroniska lub małe, regionalne hotele, należy liczyć się z tym, że (a) samo odnalezienie ich stron w Internecie może nie być łatwe, (b) informacja, którą na nich znajdziemy będzie za każdym razem inaczej przedstawiona i nie zawsze kompletna pod względem naszych wymagań. System TSS ma umożliwić sprawną organizację podróży bez konieczności błądzenia przez użytkownika po różnorodnych stronach internetowych.

W rzeczywistości, ze względu na obecny stan rozwoju sieci WWW, wspomniane strony internetowe z dziedziny podróży są bardzo rozproszone i bardzo słabo usystematyzowane. W konsekwencji, często ciężko odnaleźć poszukiwaną informację a możliwości automatyzacji procesu organizacji podróży operującego bezpośrednio na danych dostępnych w Internecie są małe. Głównym celem projektu Systemu TSS jest przede wszystkim stworzenie systemu umożliwiającego taką właśnie automatyzację.

W zakres projektu wchodzą przede wszystkim następujące zagadnienia: (a) ewolucja sieci WWW w stronę Sieci Semantycznej, (b) integracja danych pochodzących z różnych źródeł, (c) dostarczanie użytkownikowi systemu interesujących go danych, (d) konstrukcja profili użytkowników oraz (e) funkcjonowanie mobilnych agentów programowych w środowisku semantycznie opisanych danych i skonstruowanych profili użytkowników [Angryk et al., 2001], [Gawinecki et al., 2005a], [Gawinecki et al., 2005b], [Gawinecki et al., 2005c], [Gawinecki et al., 2005d], [Gawinecki, 2005], [Gordon et al., 2005], [Gordon i Paprzycki, 2005], [Nagrodkiewicz i Paprzycki, 2005]. Wymiernymi wynikami projektu są opisy istniejących ograniczeń budowy systemu i poszczególnych jego elementów oraz propozycje rozwiązań niwelujących te ograniczenia, zarówno teoretyczne jak i praktyczne.

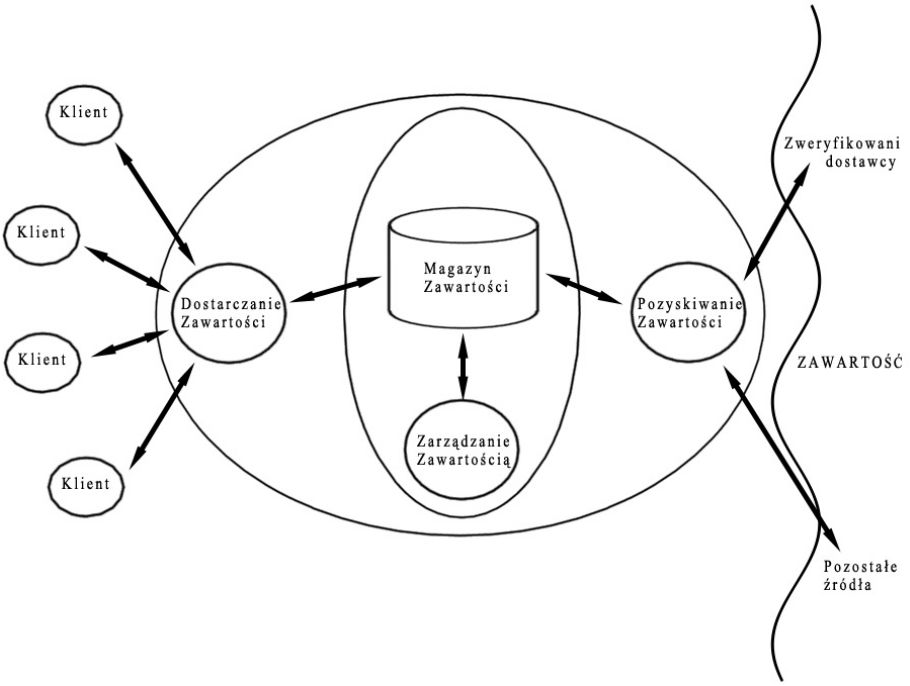
5.1.2. Funkcjonalności systemu TSS

Mając na uwadze cel projektu oraz przyjmując punkt widzenia użytkownika można wyspecyfikować następujące funkcjonalności systemu TSS:

- Tworzenie profilu użytkownika.
- Gromadzenie i dostarczanie użytkownikowi heterogenicznych danych o: (1) bazie hotelowej (zakwaterowanie); (2) bazie restauracji (gastronomia); (3) połączeniach lotniczych, autobusowych oraz kolejowych (komunikacja).
- Udostępnienie informacji o geograficznym położeniu opisywanych obiektów.

5.1.3. Proponowana architektura Systemu

Na przestrzeni ostatnich szesnastu lat powstało wiele projektów zastosowania paradygmatu programowania agentowego na potrzeby systemów wspomagania organizacji podróży, lecz zdecydowana większość z nich zakończyła się na fazie projektowania lub szczątkowych implementacji. Projekt systemu TSS powstał w 2001 [Angryk et al., 2001] i od tego czasu ewoluował. Wynikiem doświadczeń zebranych w trakcie czasu trwania projektu jest szczegółowa architektura Systemu zaproponowana w [Gordon i Paprzycki 2005]. Ogólny zarys tej architektury daje Rysunek 5.1.



Rysunek 5.1. Architektura Systemu TSS

Kluczowymi elementami dla całej architektury są podsystemy: pozyskiwania, zarządzania i dostarczania zawartości. Oto ogólny opis komponentów zawartych w Rysunku 5.1 [Gawinecki et al., 2005b]:

Zweryfikowani Dostawcy Zawartości *(ZDZ; ang. Verified Kontent Providers)*. Problemem związanym z pozyskiwaniem danych z Internetu jest ich dynamiczny charakter i niestabilność [Nwana i Ndumu, 1999]. Dla odróżnienia danych „godnych zaufania", czyli pochodzących ze sprawdzonych

i stabilnych źródeł zostali wyróżnieni właśnie *Zweryfikowani Dostawcy Zawartości* [Abramowicz i Kalczyński, 2002].

Pozostałe źródła (*ang. Other Sources*). W ten sposób rozróżniono wszystkie nie zweryfikowane źródła informacji; mimo, że nie są one „godne zaufania" nie powinny pozostać poza systemem, z uwagi na to, że w momencie upowszechnienia Sieci Semantycznej informacje pochodzące z tych stron internetowych będą łatwo interpretowalne przez system; informacje zebrane z *Pozostałych Źródeł* mogą nabrać szczególnie ważnego znaczenia.

Podsystem Pozyskiwania Zawartości (*PPZ; ang. Content Collection Subsystem*). Składa się z wielu agentów programowych wyszukujących informacje i opakowujących je opisem semantycznym w postaci trójek RDF. Zbiory takich trójek, odpowiadające konceptom zdefiniowanym w ontologii, są następnie wysyłane do centralnego repozytorium opartego na środowisku JENA [Jena, 2003a], [Jena, 2003b].

Podsystem Zarządzania Zawartością (*PZZ; ang. Content Management Subsystem*). Odpowiada za zarządzanie danymi w repozytorium opartym na środowisku JENA; rozpoznaje on między innymi stopień kompletności trójek nadesłanych przez *PPZ*.

Podsystem Dostarczania Zawartości (*PDZ; ang. Content Delivery Subsystem*). Jest odpowiedzialny za dostarczenie użytkownikowi spersonalizowanych danych na podstawie złożonego przez niego zapytania.

Klienci umożliwiają użytkownikom dostęp do systemu za pośrednictwem Internetu. Powinni działać na różnorodnych terminalach wyposażonych w różne platformy.

Niezależnie od systemu TSS powstał projekt systemu rezerwacji biletów lotniczych [Vukmirovic et al., 2006a]. Jednak przewidziane w jego ramach funkcjonalności dają podstawy do jego integracji z systemem TSS. Byłby on jego komplementarną częścią. Kwestia integracji tych dwóch systemów sprowadza się do opracowania ontologii pozwalającej opisywać zarówno dane,

które do tej pory przechowywane w ramach Systemu TSS oraz dane związane z rezerwacją biletów lotniczych [Vukmirovic et al., 2006b].

5.1.4. Rola ontologii

Ontologia dla systemu TSS ma zapewnić terminologię i podstawę wnioskowania dla logiki biznesowej całego systemu. Będzie podstawą dla porozumienia między agentami działającymi w ramach systemu TSS. Dzięki opracowanej ontologii będzie można nadać sens związanej z podróżowaniem informacji istniejącej obecnie w Internecie. Tak by powstały inteligentne dane (w rozumieniu inteligentnych danych wymaganych przez Sieć Semantyczną, sekcja 3.2) dostępne dla użytkowników systemu.

Ogólnie ontologia systemu TSS ma zapewnić teorię dziedziny podróży, spełniającą wymagania systemu. Należy podkreślić, że jest to przykład ontologii dziedzinowej kierowanej wymaganiami aplikacji, w tym przypadku dotyczącej dziedziny podróży przyjmującej punkt widzenia turysty/podróżnika. Nie jest to ontologia ogólna, której celem byłoby opisanie ogółu rzeczy, tj. wszystkich bytów możliwych na świecie. Co więcej nie daje ona nawet możliwości opisu wszystkich ważnych aspektów dziedziny podróży. Przykładowo, terminologia zdefiniowana w ramach ontologii systemu TSS nie daje możliwości opisu miejsca zakwaterowania z punktu widzenia właściciela na potrzeby administracyjne.

W szczególności, ontologia pełni w tym systemie następujące role:

- Definiuje obiekty, które powinny być wyszukane przez podsystem *PPZ*;
- Definiuje obiekty, które mogą być dostarczone użytkownikom przez podsystem *PDZ*;
- Definiuje obiekty, które mogą być przechowywane w centralnym repozytorium; zarządza nimi podsystem *PZZ*;
- Klasyfikuje te obiekty i definiuje relacje między nimi, przez co umożliwia przeprowadzenie wnioskowania.

Odpowiednie podsystemy (*PPZ*, *PZZ* i *PDZ*) powinny być na tyle elastyczne by umożliwić wraz ze zmianą wersji ontologii, automatyczną zmianę struktury zawartości dostarczanej, przechowywanej lub pobieranej z centralnego repozytorium. Dodatkowo, w przypadku integracji z systemem rezerwacji biletów lotniczych ontologia zapewnia jednolitą teorię dziedziny podróży wraz ze sposobem oznaczania danych krążących w całym systemie TSS wzbogaconym o moduł rezerwacji biletów.

5.1.5. Technologia implementacji komponentów systemu

Implementacja komponentów systemu TSS jest możliwa dzięki dwóm środowiskom: Java Agent DEvelopment Framework (JADE).

JADE

Jest to środowisko ułatwiające programowanie systemów wieloagentowych zaimplementowane w języku Java. Jest w pełni zgodne ze specyfikacjami Foundation for Inteligent Physical Agents (FIPA) [FIPA, 2006]. Główną jego zaletą jest przyspieszenie faz testowania oraz osadzania aplikacji-agentów, możliwe dzięki interfejsom graficznym wchodzącym w skład tego środowiska. Architektura tego środowiska daje możliwość jego rozszerzania o serwery baz danych oraz inne środowiska, w tym środowisko Jena.

Jena

Środowisko to służy do tworzenia aplikacji dla Sieci Semantycznej w języku Java. Dostarcza przede wszystkim bogaty interfejs programowania aplikacji (API) dedykowany do czytania, zapisywania i przetwarzania dokumentów RDF (w dowolnej notacji) oraz OWL (w dowolnym wariancie). Dodatkowo, zapewnia możliwość odpytywania (ang. querying) dokumentów zapisanych w RDF przez interfejs silnika SPARQL [W3C, 2006a] oraz serializację dokumentów RDF do bazy danych (możliwe do wykorzystania serwery to: MySQL, HSQLDB, PostgreSQL, Oracle, Microsoft SQL Server).

5.2. Tworzenie ontologii systemu TSS

Tworzona ontologia ma umożliwiać opis dziedziny podróży z punktu widzenia podróżnika, zatem jest to ontologia dziedzinowa. Proces jej rozwoju w dużej mierze opiera się o *cykl życia ontologii* zaprezentowany w podsekcji 3.5.3, jednak zawiera elementy *podejścia mieszanego* (podsekcja 3.5.1), a jej celem jest stworzenie *dynamicznej sieci znaczeń* (podsekcja 3.5.2). Poniższa lista prezentuje kluczowe etapy procesu tworzenia ontologii systemu TSS z uwzględnieniem roli ontologii oraz funkcjonalności tego systemu:

1. Określenie wymagań dziedziny – wykorzystanie podejścia *od szczegółu do ogółu*; zapewnienie pokrycia dziedziny podróży zgodnie z wymaganiami systemu TSS:
    a. wydzielenie zamkniętych poddziedzin;
    b. badanie ich wymagań.
2. Określenie wymagań technologicznych i wybór technologii spełniającej określone wymagania – porównanie języków ontologicznych opisanych w Rozdziale 4 pod kątem spełniania określonych wymagań ontologii dziedziny podróży.
3. Budowa ontologii:
    a. implementacja ontologii poddziedzin;
    b. integracja ontologii poddziedzin;
    c. wykorzystanie ontologii zewnętrznych.
4. Publikacja ontologii:
    a. wymagania publikacji;
    b. tworzenie publikacji.

5.2.1. Wymagania dziedziny

Biorąc pod uwagę funkcjonalności systemu TSS oraz przewidziane dla ontologii role należało określić jej wymagania. W celu ich precyzyjnej definicji rozważamy zarówno wymagania dziedziny oraz wymagania technologiczne. Pierwsze z tych wymagań są istotne w świetle sekcji 3.2 oraz 3.5. Pozwolą one zbadać poprawność ontologii na wysokim poziomie względem funkcjonalności

określonych w 5.1.2. Określenie tych wymagań jest jednym z elementów przedstawionego w podsekcji 3.5.3 *cyklu życia ontologii*. Pozwolą one opracować poprawną, z punktu widzenia systemu TSS, ontologię dziedziny podróży. Dodatkowym wymaganiem pozadziedzinowym jest konieczność odwołania się do konceptów wyższego poziomu, szerzej wyjaśniona w sekcjach 3.4 i 3.5.

W [Gordon et al., 2005] została zaproponowana pierwsza wersja ontologii podróży. Jednak ze względu na jej zbyt ogólny (mało szczegółowy) charakter oraz nie spełnienie podstawowych wymagań systemu została odrzucona przez kierownika projektu. Jedynym wykorzystanym z jej elementów jest analiza serwisów rezerwacji hotelowych, która dała podstawy bardziej precyzyjnemu zgłębianiu dziedziny podróży, a w szczególności dziedziny miejsca zakwaterowania.

Poddziedziny i ich wymagania

Ze względu na rozpiętość dziedziny podróży na pierwszym etapie tworzenia nowej wersji ontologii zostało zastosowane podejście *od szczegółu do ogółu*. W oparciu o funkcjonalności[*] zostały wyróżnione trzy podstawowe poddziedziny:

1. Miejsce zakwaterowania
2. Gastronomia
3. Transport

Każdy z tych modułów był rozwijany w dużym stopniu jako niezależna ontologia w oparciu o opracowania specjalistyczne oraz, w przypadku

[*]Funkcjonalności systemu TSS:

- Tworzenie profilu użytkownika.
- Gromadzenie i dostarczanie użytkownikowi heterogenicznych danych o:
  - Bazie hotelowej (zakwaterowanie);
  - Bazie restauracji (gastronomia);
  - Połączeniach lotniczych, autobusowych oraz kolejowych (komunikacja);
- Udostępnienie informacji o geograficznym położeniu opisywanych obiektów.

transportu lotniczego, doświadczenia zawodowe osoby zajmującej się rozwojem tej części ontologii.

Wymagania poddziedziny miejsca zakwaterowania

Ontologia dziedziny podróży zapewnia możliwość opisania dowolnego miejsca zakwaterowania, uwzględniając wszelkie możliwe informacje, które są obecnie prezentowane na tradycyjnych stronach internetowych sieci hotelowych, pojedynczych hoteli, moteli, kempingów, pól namiotowych, czy też schronisk. Ważne jest by nie ograniczała ona w żaden sposób prezentacji jakiegokolwiek z typów zakwaterowania, a jednocześnie dawała klarowny obraz. Powinna być również zwięzła, podobna do streszczeń publikowanych przez popularne serwisy zajmujące się wyszukiwaniem zakwaterowania. Z tego ostatniego powodu, wymagania dziedziny miejsca zakwaterowania zostały określone w pierwszej kolejności w oparciu o analizę następujących stron i serwisów internetowych [Gordon et al., 2005]:

- Travelocity [Travelocity, 1996],
- Venere.com [Tenere, 1995],
- HotelClub [HotelClub, 2000],
- Hotel Reservation Service [HRS, 2006],
- Web-Hotels.com [WebHotels, 2006],
- TravelWeb [TravelWeb, 2006],
- TravelCiti [TravelCiti, 2006].

Nie są to przypadkowe serwisy, w listopadzie 2004 roku były one wśród pierwszych dziesięciu wyników wyszukiwania przez www.google.com dla zapytanie *hotel reservation*. Porównane zostały opisy warszawskiego hotelu *Le Royal Meridien Bristol* zwrócone przez każdy z wyżej wymienionych serwisów. W wyniku tego porównania została otrzymana następująca lista szczegółów opisu miejsca zakwaterowania:

1. Adres
2. Akceptowane środki płatności
3. Akceptowane waluty
4. Atrakcje w pobliżu

5. Doba hotelowa
6. Gwarantowane taryfy
7. Ilość pokoi
8. Kategoria/typ miejsca zakwaterowania
9. Mapa z lokalizacją miejsca
10. Nazwa
11. Opcje wyżywienia
12. Polityka rezerwacji
13. Polityka względem zwierząt
14. Pozycja w rankingu (ilość gwiazdek)
15. Typy pokoi i ich ceny
16. Wyposażenie miejsca zakwaterowania
17. Wskazówki dojazdu
18. Zdjęcia miejsca

To właśnie na podstawie powyższej listy atrybutów miejsca zakwaterowania powstał moduł pierwszej wersji ontologii dziedziny podróży (szczegółowo opisanej w [Gordon et al., 2005]) odpowiedzialny za reprezentację miejsca zakwaterowania. Jego podstawowymi wadami były: (a) ograniczenie modelu opisu miejsca zakwaterowania do modelu przyjętego przez serwisy rezerwacji hotelowych, (b) nadanie wszystkim kluczowym atrybutom wartości w formie napisów w języku naturalnym. Te błędy wynikały przede wszystkim z mało precyzyjnej analizy dziedziny miejsca podróży.

W związku z tym, dodatkowej analizie zostały przeze mnie poddane strony internetowe sieci hoteli, gdyż to głównie z nich będą pozyskiwane dane do systemu. Strony te dostarczają większego zasobu informacji, który musiał zostać porównany z analizowanymi uprzednio serwisami rezerwacji hotelowych. Dzięki temu reprezentacja danych o miejscach zakwaterowania nie ogranicza się w systemie TSS do modelu tożsamego z przyjętym przez serwisy rezerwacji hotelowych. Z wielu światowych koncernów hotelarskich zostały wybrane cztery prezentujące najbogatszy zestaw informacji. Są nimi:

- Marriot Hotels [Marriot, 1996],
- Sofitel Hotels [Sofitel],
- InterContinental Hotels Group [ICHotels, 2001],
- Sheraton Hotels & Resorts [Sheraton 2006].

W porównaniu do serwisów rezerwacji hotelowych strony te dostarczają bardziej szczegółowych informacji. Poniżej znajduje się lista elementów nieuwzględnionych przez żaden z wymienionych serwisów rezerwacji hotelowych:

19. Informacje o dostępności pokoi w interesujących użytkownika terminach
20. Informacje o kuchni
21. Informacje o płatnych usługach na terenie miejsca zakwaterowania
22. Informacje o pokojach konferencyjnych
23. Informacje o restauracji
24. Informacje o środkach zabezpieczeń
25. Informacje o udzielanych zniżkach
26. Informacje o wsparciu dla osób upośledzonych
27. Usługi dla klientów biznesowych
28. Wyposażenie pokoi

Te informacje są wartościowe z punktu widzenia zarówno branży hotelarskiej jak i potencjalnych gości hotelowych, zatem muszą być uwzględnione w ontologii systemu TSS. Ponadto, zgodnie z wymaganiami funkcjonalnymi systemu potrzebne są również informacje dotyczące położenia geograficznego miejsca zakwaterowania. Istotne jest, by podkreślić już na tym etapie, iż wymienione kategorie mają wyznaczyć zbiór maksymalny pozwalający opisać dowolnego typu miejsce zakwaterowania (hotel, pole namiotowe, schronisko młodzieżowe).

Ustalony zakres informacji potrzebnych do opisania obiektu, można rozumieć jako zbiór jego atrybutów. Mając taki zbiór dla dowolnego miejsca zakwaterowania należy opracować zestawienie podstawowych wartości, jakie mogą one przyjmować. Jeśli założymy, że wartościami atrybutów są napisy w

języku naturalnym, wówczas sprowadzamy możliwość interpretacji ich znaczeń do interpretacji języka naturalnego. Natomiast w wypadku, gdy mogą one przyjmować wartości konkretnych typów (również opisanych w ontologii) to łatwość ich interpretacji przez agentów programowych znacznie wzrasta. Dodatkowo, zbiór możliwych wartości powinien być standardem powszechnie akceptowanym przez środowisko osób związanych z tą dziedziną.

Wymaganie to jest motywowane nie tylko zwiększeniem wydajności wyszukiwania informacji w centralnym repozytorium i budowania bardziej szczegółowej bazy wiedzy, lecz przede wszystkim definicjami i interpretacjami ontologii przedstawionymi rozdziale 3. Wynika z nich, że budowana konceptualizacja musi być *wspólna* (ang. shared) – zaaprobowana przez grono osób będących specjalistami w danej dziedzinie.

Wymagania poddziedzin gastronomii i transportu

Ontologia restauracji (pozwalająca opisać różne rodzaje lokali gastronomicznych) została stworzona na podstawie projektu ChefMoz. Jej wymagania i etapy rozwoju zostały szczegółowo opisane w [Gawinecki et al., 2005a], [Gawinecki, 2005].

Prace nad ontologią transportu rozpoczęły się od stworzenia ontologii systemu rezerwacji biletów lotniczych, w momencie pisania tego tekstu istnieje pierwsza wersja tej ontologii opisana w [Vukmirovic et al., 2006b]. Standardy wykorzystane przy projektowaniu tego fragmentu ontologii opierają się na specyfikacjach International Air Transport Association (IATA) [IATA, 2006a], [IATA, 2006b], [IATA, 2006c]. Konstrukcja tego fragmentu pozwala go rozbudować na potrzeby ogólniejszej ontologii dla całej dziedziny transportu.

5.2.2. Wymagania technologiczne i wybór języka

Wymagania technologiczne

Z kolei wymagania technologiczne są wynikiem analizy niezbędnego poziomu precyzji dla ontologii systemu TSS. Pozwalają wybrać jeden język ontologiczny na potrzeby tworzenia całej ontologii. W ramach projektu

systemu TSS zostały zdefiniowane wstępne wymagania technologiczne ontologii [Gordon et al., 2005]:

1. Łatwość zrozumienia i wykorzystania ontologii przez człowieka.
2. Możliwość zrozumiałość i interpretacji ontologii przez agentów programowych.

Zostały one następnie rozszerzone o następujące wymagania [Szymczak et al., 2006]:

3. Dziedziczenie wielobazowe.
4. Wsparcie dla reguł wnioskowania, w szczególności dla relacji „*is-a*”, innymi słowy:
    a. Dziedziczenie atrybutów *z góry w dół* (ang. top-down);
    b. Dziedziczenie instancji *z dołu do góry* (ang. bottom-up).
5. Możliwość wykorzystania istniejących ontologii.

Innymi słowy, język ontologiczny wybrany do budowy ontologii systemu TSS powinien umożliwić agentom (niezależnie od ich ludzkiej bądź programowej natury): (a) interpretować zasoby (ang. resources) jako reprezentantów poszczególnych klas, (b) wnioskować klasę instancji bazowej oraz atrybuty klasy potomnej (c) definiować nowe klasy oraz ich własności, (d) dołączać etykiety w języku naturalnym, (e) organizować hierarchię klas, (f) wykorzystywać istniejące definicje konceptów, bądź całe konceptualizacje [Szymczak et al., 2006].

Wybór języka

Z trzech opisanych w rozdziale 4 języków formalnych jedynie XML z XSD nie spełniają postawionych powyżej wymagań technologicznych. Pomimo tego, że na języku XML oparte są składnie języków OWL i RDF(S), to nie posiada on możliwości definiowania semantyki, w szczególności nie można w nim explicite określić hierarchii obiektów oraz relacji „*is-a*”. Dodatkowo, struktura języka XML nie może różnić się od jej definicji w XSD, co sprawia, że język ten nie jest elastyczny w odróżnieniu od RDF(S) i OWL [Szymczak et al., 2006].

Do zastosowania w systemie został ostatecznie wybrany RDF(S), gdyż jego składnia oparta jest na trójkach uporządkowanych (ang. tripple), w których każdy element ma swoją rolę, co zapewnia przejrzyste wnioskowanie. Ponadto, możliwość definiowania wartości zasięgu dowolnego atrybutu wprowadza dodatkowe zasady wnioskowania. Zgodnie z informacjami podanymi w Rozdziale 4, OWL również zapewnia wszystkie te zasady. Jednak wszystkie dodatkowe, takie jak: definiowanie ograniczeń i liczebność wartości atrybutów w konkretnych obiektach, czy zróżnicowane typy klas i atrybutów stwarzają narzut, który nie został uwzględniony w wymaganiach technologicznych.

Poza tym formalizm RDF(S) wydaje się gwarantować zwiększone możliwości ponownego wykorzystania raz opracowanej ontologii, gdyż jest on nastawiony zdecydowanie bardziej na konceptualizację dowolnej dziedziny niż na definiowanie, niekiedy zawiłych, ograniczeń dotyczących konceptów i ich atrybutów. Istotny jest również fakt, iż wydajność narzędzi analizy składniowej (ang. parser) i wnioskowania (ang. reasoner) przeznaczonych dla języka RDF(S) jest znacząco lepsza od wydajności analogicznych narzędzi dla języka OWL [Baclawski, 2006]. Jest to spowodowane właśnie złożonymi zasadami wnioskowania języka OWL. Zaletą wyboru RDF(S) jest spełnienie wszystkich postawionych wymagań technologicznych oraz lepsza wydajność narzędzi analizujących i wnioskujących dla tego języka niż dla OWL.

5.2.3. Budowa ontologii

Budowa ontologii poddziedzin

Ontologia restauracji powstała z potrzeby dostarczania użytkownikom danych o lokalach gastronomicznych. Dane w formie RDF/XML, są udostępniane przez serwis *ChefMoz dining guide* będący gałęzią projektu *DMOZ*. Niestety, pomimo tego, że dane są dostępne w formacie RDF nie istnieje publiczna wersja ontologii, według której dane o lokalach gastronomicznych w ramach projektu *ChefMoz* są opisywane. Dlatego też, ontologia ta została odtworzona w oparciu o analizę opisów danych pobieranych z serwerów *ChefMoz* [Gawinecki, 2005], [Gawinecki et al., 2005a].

Prace nad ontologią podróży lotniczej postępują od stosunkowo niedawna. Po przeprowadzeniu analizy specyfikacji IATA [IATA 2006a], [IATA 2006b], [IATA 2006c] zostały opracowane wymagania i funkcjonalności. Ponadto został zaprojektowany szkielet tej ontologii, wyróżniający koncepty potrzebne do jej pełnej implementacji [Vukmirovic et al., 2006a], [Vukmirovic et al., 2006b].

Aby spełnić wymagania określone w 5.2.1 ontologia powinna pozwalać opisać miejsce zakwaterowania z uwzględnieniem wszystkich wymienionych elementów jego charakterystyki. Należy zauważyć, że otrzymany w ten sposób zbiór atrybutów miejsca zakwaterowania będzie zbiorem maksymalnym atrybutów publikowanych obecnie w Internecie w celu promocji dowolnego miejsca zakwaterowania. Zatem, zakładając dodatkowo, że dane o miejscach zakwaterowania będą pozyskiwane przez podsystem *PPZ* z Internetu od dostawców *ZDZ*, ontologia zbudowana w oparciu o listę atrybutów zdefiniowaną w 5.2.1 pozwoli pozyskać, przechować i odzyskać dokładne dane dotyczące dowolnego miejsca zakwaterowania.

Istotnym wymaganiem postawionym w 5.2.1 jest zbudowanie zbiorów typów dla określonych atrybutów miejsca zakwaterowania, zaakceptowanego przez środowiska związane zawodowo z dziedziną podróży.

Do realizacji podobnego celu przyczynia się organizacja Open Travel Alliance (OTA) [OTA, 2001], która skupia czołowych przedstawicieli branży turystycznej oraz IT. Od 2001 roku, dwa razy do roku, publikuje ona specyfikacje wiadomości i ich zawartości wykorzystywane w systemach rezerwacji biletów lotniczych, miejsc hotelowych oraz samochodów w wypożyczalniach. Każda specyfikacja składa się z dokumentów XSD dla typów wiadomości, definicji typów zawartości tych wiadomości w dokumentach XML oraz szczegółowej dokumentacji. Do tworzenia ontologii miejsca zakwaterowania, a konkretnie typów wartości atrybutów, została wykorzystana wersja z 2004 roku, z uaktualnieniem typu InformationType z 2005 roku (obie są publicznie dostępne na stronie internetowej OTA [OTA, 2001]). Jeszcze jedną zaletą wykorzystania specyfikacji OTA jest możliwość

rozbudowy systemu TSS o moduł translacji wiadomości między standardem OTA a ontologią Systemu.

Dodatkowo, w ramach ontologii miejsca zakwaterowania zostały stworzone moduły ontologii pozwalające opisać dane osobowe, kontakty telefoniczne oraz dane adresowe. Powstały one na potrzeby przyszłej integracji z pozostałymi ontologiami poddziedzin w celu zwiększenia precyzji opisu tych elementów. Tabela 5.1 zawiera opis wszystkich elementów składających się na ontologię miejsca zakwaterowania i określa ich funkcje w ontologii.

Klasa/Element	**Opis**
Adres	Definiuje adres. Umożliwia podanie zarówno adresu w formie ciągu znaków jak i ciągu elementów składających się na poprawny adres (ulica, dom, mieszkanie, miasto, kraj, kod pocztowy, itd.).
Restauracja	W ramach miejsca zakwaterowania może funkcjonować lokal gastronomiczny. Jest to bezpośrednie odniesienie do ontologii restauracji stworzonej w ramach pracy magisterskiej Gawineckiego [Gawinecki, 2005].
Typ usług biznesowych	Definicja możliwych dostępnych usług biznesowych. Przykładowe usługi to: ogólnodostępny faks, kopiarka, modem, tablica, dostęp do Internetu. Kod OTA: BUS.
Kategoria segmentu	Określa możliwe segmenty miejsc zakwaterowania ze względu na standard, koszt i przeznaczenie. Przykładowe segmenty to: luksusowy, ekonomiczny, turystyczny. Kod OTA: SEG.
Kontakty	Umożliwia szczegółową definicję kontaktów. Zawiera definicje podstawowych: typów lokalizacji kontaktu (np. biuro, rezerwacje, dom; kod OTA: PLT), typów przeznaczenia kontaktu (np. dzienny, nocny, ratunkowy; kod OTA: PUT), typów usług dla kontaktu (np. parking, pralnia; kod OTA: CSC), typów technologii kontaktu (np. telefon, email; kod OTA: PTT).

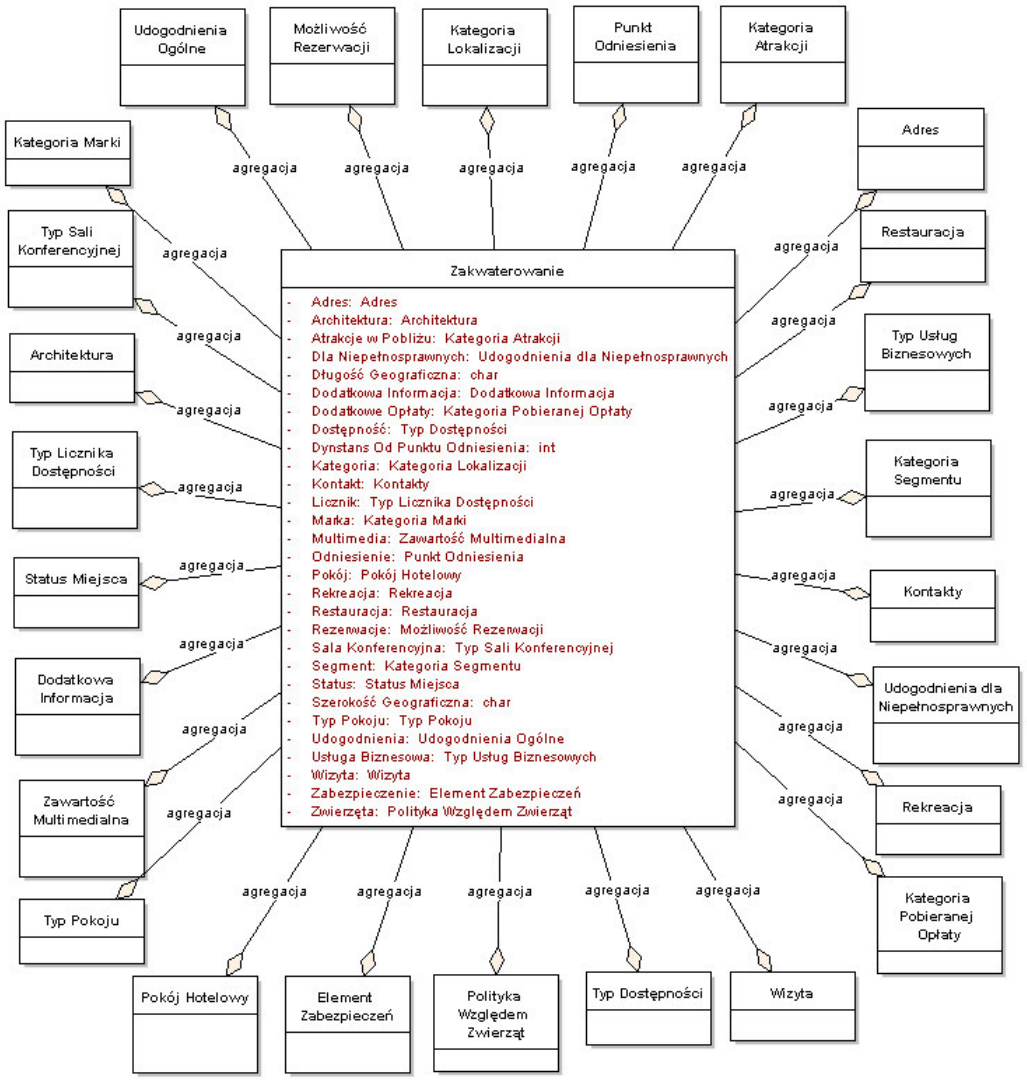
Udogodnienia dla niepełnosprawnych	Umożliwia definicje dostępnych usług i udogodnień dla osób niepełnosprawnych. Przykładowe udogodnienia to: uchwyty w łazienkach, ilość dostępnych wózków na kółkach. Kod OTA: PHY.
Rekreacja	Definiuje możliwe usługi rekreacyjne dostępne na terenie obiektu (np. strzelnica, pole golfowe, basen; kod OTA: RST) oraz szczegóły związane z tymi usługami (np. głębokość basenu, program dla dzieci; kod OTA: REC).
Kategoria pobieranej opłaty	Określa usługi, za które możliwe jest pobieranie dodatkowej opłaty w miejscu zakwaterowania. Przykładowe kategorie to: tenis, pralnia, faks, parking. Kod OTA: RCC.
Wizyta	Pozwala opisać pobyt danej osoby w miejscu zakwaterowania przy pomocy zbioru pól szczegółowych takich jak: okres wizyty, dane osoby, kontakt osoby, metoda rezerwacji, faktura/rachunek za wizytę. Przy czym rachunek może być opisany w sposób szczegółowy przez wartości następujących pół: waluta, kwota, taryfa, odprowadzony podatek.
Typ dostępności	Określa w jakim stopniu miejsce zakwaterowania jest dostępne dla gości, np: częściowo lub całkowicie.
Polityka względem zwierząt	Umożliwia opis stosowanej polityki względem zwierząt. Przykładowo: tylko psy, tylko koty, żadnych zwierząt. Kod OTA: PET.
Element zabezpieczeń	Definiuje możliwe środki bezpieczeństwa w miejscach zakwaterowania. Przykładowe elementy zabezpieczeń to: podwójne zamki na drzwiach, oświetlony parking. Kod OTA: SEC.
Pokój hotelowy	Umożliwia opisanie jednostkowego pokoju w miejscu zakwaterowania. Określa jego standard (dziedziczy wszystkie atrybuty po Typie Pokoju), numer, dokładną

	lokalizację wewnątrz miejsca zakwaterowania (np. skrzydło zachodnie, przy basenie; kod OTA: RLT), typ widoku z okna (np. na lotnisko, na morze, na basen; kod OTA: RVT).
Typ pokoju	Pozwala zdefiniować standard pokoju przez specyfikację jego ceny, zawartości (np. radio z budzikiem, lodówka; kod OTA: RMA), typ łóżka (np. podwójne, polowe, łoże małżeńskie; kod OTA: BED) i nazwy standardu.
Typ środka płatności	Określa dostępne typy środków płatności. Kod OTA: PMT.
Typ menu	Zawiera definicje występujących rodzajów kuchni. Opracowany częściowo na podstawie kodów OTA (CUI), listy rodzajów kuchni opracowanej na potrzeby projektu *ChefMoz* oraz ontologii restauracji [AgentCities, 2002b]
Zawartość multimedialna	Definiuje możliwe załączniki multimedialne wraz z ich lokalizacją. Kod OTA: CTT.
Dodatkowa informacja	Umożliwia załączenie informacji w języku naturalnym, dotyczącej szczegółów określonych przez Typy Dodatkowej Informacji (np. informacje dla podróżnych przyjeżdżających późną porą, informacje o podatkach; kod OTA: ADT).
Status miejsca	Określa, w jakim stanie jest miejsce zakwaterowania. Przykłady statusu: otwarte, zamknięte, przed otwarcie. Kod OTA: HST.
Typ licznika dostępności	Pozwala wyświetlić informacje ilościowe dotyczące jednego z wyspecyfikowanych aspektów miejsca zakwaterowania. Przykładowe typy liczników: goście, dostępne pokoje. Kod OTA: CNT.
Architektura	Określa styl architektoniczny miejsca zakwaterowania. Przykładowo: nowoczesny, wiktoriański, brazylijski. Kod OTA: ARC.

Typ sali konferencyjnej	Definiuje dostępne sale konferencyjne, ich kształt (np. teatralny, litera U; kod OTA: MRF) oraz cechy (np. projektor, kamera wideo; kod OTA: MRC).
Udogodnienia ogólne	Umożliwia opis dostępnych na terenie miejsca zakwaterowania udogodnień dla gości. Przykładami takich udogodnień są: bankomat, odźwierny, maszyna do lodu, kasyno, restauracja. Kod OTA: HAC.
Możliwość rezerwacji	Określa politykę rezerwacji. Wymieniane możliwości to między innymi: zalecana, wymagana, nie wymagana.
Kategoria lokalizacji	Definiuje ogólną charakterystykę okolicy, w jakiej położone jest dane miejsce zakwaterowania. Przykładowe kategorie to: plaża, miasto, góry, jezioro. Kod OTA: LOC.
Punkt odniesienia	Określa charakterystyczny punkt terenu w stosunku, do którego podawana jest informacja o dojeździe. Przykładowe typy punktów odniesienia to: autostrada, miasto, lotnisko. Kod OTA: IPC.
Kategoria atrakcji	Umożliwia opisanie różnych atrakcji występujących w pobliżu opisywanego miejsca zakwaterowania. Przykładowe kategorie to: kościół, festiwal, stadion. Kod OTA: ACC.

Tabela 5.1. Elementy wykorzystane w opisie miejsca zakwaterowania

Poniższy schemat ilustruje klasę umożliwiającą opisanie dowolnego miejsca zakwaterowania – *Zakwaterowanie*, jej atrybuty oraz powiązania z innymi, zdefiniowanymi klasami.



Rysunek 5.2. Klasa Zakwaterowanie jej atrybuty i agregaty.

Integracja powstałych ontologii poddziedzin [Szymczak et al., 2006]

Szczątkowe wzajemne wykorzystanie konceptów między opracowanymi ontologiami trzech dziedzin zostało uwzględnione w dokumentach [Gawinecki et al., 2005a], [Vukmirovic et al., 2006b]. Polegało ono na możliwości opisu restauracji hotelowej oraz położenia przykładowej restauracji w sąsiedztwie lotniska, hotelu lub jeszcze innej restauracji.

Po opracowaniu ontologii dziedzinowych lub, w przypadku ontologii transportu, specyfikacji konceptów potrzebnych do jej implementacji, została przeprowadzona dogłębna integracja tych konceptualizacji. Czyli, zgodnie ze stosowanym podejściem *od szczegółu do ogółu*, abstrakcja szczegółów

dziedziny. Integracja ta była procesem iteracyjnym, składającym się z następujących etapów:

1. Rozpoznanie części wspólnej między ontologiami;
2. Wyszczególnienie części wspólnej;
3. Zbudowanie lub odświeżenie relacji z wyszczególnionymi i zmodyfikowanymi elementami.

Integracja w takim znaczeniu może być rozumiana jako unifikacja opisu. Pierwszą z części wspólnych miedzy ontologiami, która podległa tak zdefiniowanemu procesowi integracji był opis lokalizacji (położenia geograficznego, adresu, charakterystyki okolicy) miejsca zakwaterowania, restauracji i portu lotniczego. Każdy z tych trzech konceptów dotyczy miejsca, najczęściej budynku, zajmującego wyznaczony teren, którego położenie na kuli ziemskiej można określić współrzędnymi geograficznymi. Miejsce takie na ogół leży na terytorium kraju na terenie, którego obowiązują adresy w jednakowym formacie. Istotne jest również, by umożliwić opisanie okolicy konkretnego miejsca. Wszystkie elementy opisu lokalizacji były uwzględnione zarówno w ontologii miejsca zakwaterowania, lokalu gastronomicznego jak i podróży lotniczej, jednak były reprezentowane przez inne atrybuty klas zakwaterowania, restauracji oraz lotniska. W celu unifikacji opisu zostały usunięte analogiczne atrybuty z trzech wymienionych powyżej klas, a w zamian została utworzona klasa LokalizacjaZewnetrzna (org. *OutdoorLocation*) zawierająca atrybuty opisu lokalizacji. W ostatnim etapie integracji klasy zakwaterowania, restauracji i lotniska zostały powiązane relacją *podklasy* (ang. sub-class) z nowo utworzoną LokalizacjaZewnetrzna (org. *OutdoorLocation*). Sama relacja *podklasy* w wykorzystywanym do budowy ontologii języku RDF(S) zapewnia dziedziczenie wszystkich atrybutów klasy nadrzędnej przez klasy potomne.

Przeprowadzona w ten sposób generalizacja zapewnia ujednolicony sposób opisu dowolnego miejsca posiadającego adres i współrzędne geograficzne. Daje możliwość rozszerzania ontologii podróży o klasy takich miejsc jak kino, teatr, sala koncertowa, stadion, sala gimnastyczna, sklep, dworzec autobusowy

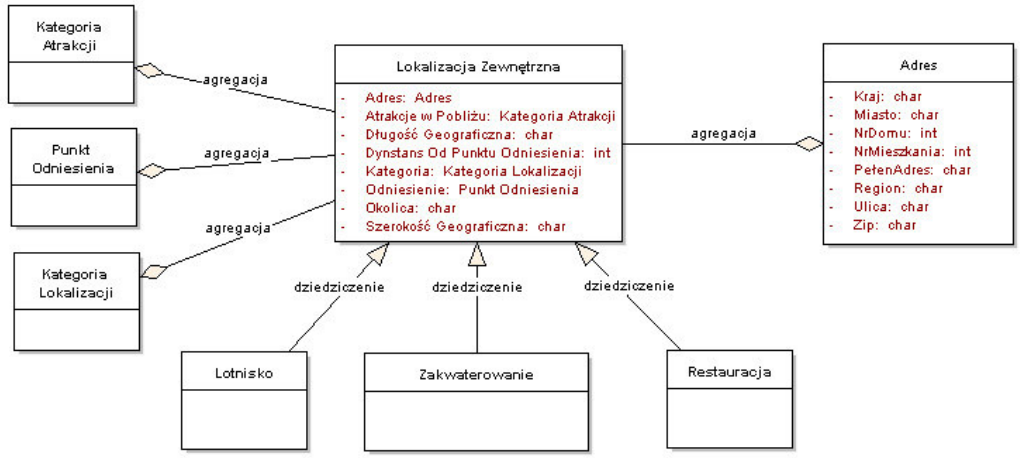
lub kolejowy. Wszystkie nowo tworzone klasy będą integralnymi składnikami ontologii podróży, jednoznacznie kwalifikowanymi jako lokalizacje posiadające współrzędne geograficzne, adres i charakterystykę otoczenia – będą specyfikacjami klasy LokalizajcaZewnetrzna. Tabela 5.2 zawiera atrybuty klasy lokalizacji wraz z ich opisem.

Klasa	**Atrybut**	**Opis**
LokalizacjaZewnetrzna		Klasa główna reprezentująca koncept lokalizacji, budynek, plac, obiekt natury, itp.
	adres	Dokładne informacje adresowe. Wartość: instancja *AddressRecord*.
	atrakcje	Atrakcje znajdujące się w pobliżu lokacji. Wartość: instancja *AttractionCategory*.
	odniesienie	Punkt odniesienia. Wartość: instancja *IndexPoint*
	dystans	Odległość od punktu odniesienia. Wartość: naturalna.
	kategoria	Kategoria lokalizacji, np. jezioro, teren miejski, plaża. Wartość: instancja *LocationCategory*.
	okolica	Opis słowny okolicy danej lokacji. Wartość: ciąg znaków.
	długość	Długość geograficzna. Wartość: rzeczywista.
	szerokość	Szerokość geograficzna. Wartość: rzeczywista.

Tabela 5.2. Klasa LokalizacjaZewnętrzna i jej atrybuty.

Rysunek 5.3 przedstawia relacje klas Zakwaterowania, Restauracji i Lotniska z klasą LokalizacjaZewnetrzna (org. *OutdoorLocation*). Dodatkowo ilustruje agregaty wykorzystywane w klasie LokalizacjaZewnetrzna. Klasy-agregaty

widoczne po lewej stronie ilustracji są definicjami kategorii atrakcji i lokalizacji oraz punktów odniesienia opracowanymi na podstawie specyfikacji OTA.



Rysunek 5.3. Relacja klas zapewniających opis lokalizacji.

Kolejną wspólną częścią ontologii miejsca zakwaterowania oraz podróży lotniczej jest koncept zniżki i taryfy. Obie poddziedziny dziedziny podróży wykorzystują ten koncept umożliwiając podanie:

- Typu zniżki/taryfy;
- Kwoty lub procentu zniżki/taryfy;
- Krótkiej charakterystyki zniżki/taryfy;
- Warunków jej uzyskania.

Różnica tkwi w możliwych wartościach typu zniżki. Ontologia miejsca zakwaterowania powstała w oparciu o specyfikację OTA, a ontologia podróży lotniczej w oparciu o dokumentację stworzoną przez IATA. Z tych specyfikacji pochodzą możliwe wartości atrybutu *typu zniżki/taryfy*. Tabela 5.2 zawiera porównanie typów wymienianych przez OTA i IATA.

Według specyfikacji OTA	**Według specyfikacji IATA**
Dla przewodnika wycieczki	Dla przewodnika wycieczki
Dla emeryta	Dla emeryta
Rządowa	Rządowa
Promocyjna	Promocyjna

Grupowa	Grupowa
Rodzinna	Rodzinna
Wojskowa	Wojskowa
AAA – najwyższa korporacyjna	Dla pracownika linii lotniczych
AARP (ang. American Association of Retired Persons)	Dla dzieci
Dla uczestników konferencji	Euro26
Korporacyjna	Dla niemowląt
Na dzień roboczy	Studencka
Weekendowa	

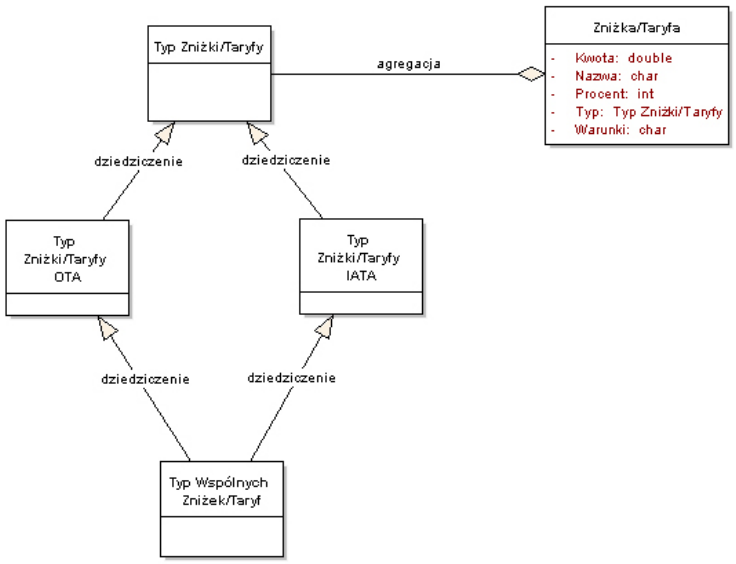
Tabela 5.3. Porównanie typów zniżek i taryf wyróżnianych w specyfikacjach OTA oraz IATA.

Wszystkie wymienione typy zniżek były konkretnymi instancjami klas TypyZnizekIATA (org. *IATADiscountTypes)* oraz TypyZnizekOTA (org. *OTADiscountTypes)* będącymi modułami ontologii odpowiednio podróży lotniczej i miejsca zakwaterowania. W momencie integracji obu tych ontologii następuje redefinicja pierwszych siedmiu typów zniżek/taryf wymienionych w tabeli 5.3. Aby tego uniknąć, należy dokonać wyróżnienia tożsamych typów jako instancji odrębnej klasy.

W przypadku unifikacji opisu lokalizacji wyróżnione zostały atrybuty klas, a klasa je zawierająca została klasą nadrzędną dla klas reprezentujących zakwaterowanie, restauracje i lotnisko. Odwrotnie w przypadku typów zniżek i taryf. Klasa typów wspólnych dla dziedzin miejsca zakwaterowania i restauracji musi zostać podklasą TypyZnizekIATA (org. *IATADiscountTypes)* oraz TypyZnizekOTA (org. *OTADiscountTypes)*, gdyż instancje klas podrzędnych są automatycznie instancjami klas nadrzędnych. Dodatkowo została wprowadzona klasa nadrzędna dla wszystkich typów zniżek zwiększająca możliwości przeprowadzenia wnioskowania. Tabela 5.4 zawiera charakterystykę wszystkich klas istniejących w zintegrowanej ontologii podróży wykorzystywanych do opisu zniżek i taryf, Rysunek 5.4 przedstawia relacje między nimi.

Klasa	Atrybut	Opis
Zniżki		Klasa główna reprezentująca koncept zniżki bądź taryfy.
	Procent	Procent przysługującej ulgi. Najczęściej stosowany zamiennie z atrybutem *kwota.* Wartość: rzeczywista.
	Kwota	Przysługująca ulga wyrażona konkretną kwotą. Najczęściej stosowany zamiennie z atrybutem *procent*. Wartość: rzeczywista.
	Typ	Typ danej zniżki lub taryfy. Wartość: instancja *TypZniżki*
	Nazwa	Opis słowny. Wartość: ciąg znaków.
	Warunki	Słowny opis warunków uzyskania ulgi. Wartość: ciąg znaków.
TypZnizki		Klasa nadrzędna wszystkich typów zniżek i taryf.
TypyZnizekOTA		Klasa nadrzędna wszystkich typów zniżek i taryf wyróżnionych w specyfikacji OTA. Bez elementów wspólnych z IATA.
TypyZnizekIATA		Klasa nadrzędna wszystkich typów zniżek i taryf wyróżnionych w specyfikacji IATA. Bez elementów wspólnych z OTA.
ZnizkiWspolneOtaIata		Klasa wspólnych zniżek i taryf dla specyfikacji OTA i IATA.

Tabela 5.4. Moduł opisu zniżek i taryf ontologii dziedziny podróży.



Rysunek 5.4. Hierarchia klas zniżek i taryf.

Przedstawione powyżej przykłady unifikacji opisów lokalizacji oraz taryf obrazują dwa przypadki: (a) redefinicji atrybutów, (b) redefinicji wartości atrybutów integrowanych ontologii szczegółowych. W pierwszym konflikt nazw został rozwiązany przez generalizację atrybutów wspólnych, w drugim natomiast przez specyfikację klas oraz przeniesienie tożsamych instancji.
Pozostałymi zunifikowanymi w trakcie integracji konceptami były: środek płatności, typ kuchni oraz użytkownik. Nie różniły się one od siebie jednak niczym poza nazwami atrybutów, wobec czego ich unifikacja sprowadziła się do wybrania jednej z klas z ontologii poddziedzinowych, ustanowienia jej jako obowiązującej w całej dziedzinie i ewentualnej zmiany nazwy atrybutów.

Wykorzystanie ontologii zewnętrznych [Szymczak et al., 2006]
Dwie ontologie opracowane poza pracami na systemem TSS zostały włączone do ontologii systemu TSS. Są nimi: SpatialThing [W3C, 2006b] i CurrencyOntology [Cambia, 2004].

Pierwsza z nich pozwala reprezentować obiekt zajmujący konkretne współrzędne geograficzne umieszczony na konkretnej wysokości nad poziomem morza. Ontologia ta jest napisana w czystym RDF(S). Została opublikowana przez W3 Consortium [W3C, 2004a]. Zważywszy na jej przeznaczenie, ontologia ta może być określona jako ontologia wysokiego

poziomu. Koncept, który opisuje jest znacznie mniej szczegółowy niż wszystkie uwzględnione w omawianej ontologii dziedziny podróży. Dlatego, klasa LokalizacjaZewnetrzna, posiadająca do tej pory własne atrybuty pozwalające opisać położenie geograficzne, została określona jako podklasa SpatialThing. Dzięki temu wszystkie instancje klas będących podklasami LokalizacjiZewnetrzenej (zakwaterowanie, restauracja, lotnisko) są rozpoznawalne jako instancje klasy SpatialThing. Ontologia systemu TSS i jej kluczowe koncepty zostają połączone z ontologią zewnętrzną przyjętą jako standard opisu pozycji geograficznej obiektów przestrzennych.

Druga z wymienionych ontologii jest udostępniana przez serwis wymiany walut Cambia. Zawiera szczegółowe elementy charakterystyki kursu i wymiany walut, w tym opis konceptu waluty. Z uwagi na dużo bardziej precyzyjny charakter tej ontologii od opracowanego na potrzeby ontologii systemu modułu reprezentującego walutę, została ona wykorzystana jako element ontologii dziedziny podróży. W porównaniu z wykorzystywanym do tej pory modułem opisu waluty ontologia serwisu Cambia umożliwia dodatkowo:

- codzienny opis kursu walut;
- zamówienie konwersji pewnej ilości X waluty A na walutę B;
- zamówienie informacji o kursie walut.

Jedyną przeszkodą w bezpośrednim wykorzystaniu ontologii serwisu Cambia były elementy składniowe pochodzące z narzędzia, w którym została ona stworzona, jednak po prostej transformacji w narzędziu Protégé [Protégé, 2006] została uzyskana ontologia w czystym RDF(S).

Dokładne definicje w notacji N3 wszystkich klas ontologii dziedziny podróży wraz z opisem ich roli w systemie zostały zawarte w Załączniku A. Wszystkie elementy pochodzące ze specyfikacji OTA zostały odpowiednio oznaczone, a elementy specyfikacji, które nie zostały wykorzystane w ontologii są wymienione w Załączniku B. Ontologia przedstawiona w Załączniku A jest w wersji po przeprowadzonej integracji.

5.2.4. Publikacja ontologii

Po zakończeniu prac nad budową ontologii podróży zostanie ona opublikowana i oceniona ze względu na: (a) spełnienie funkcjonalności Systemu, (b) spełnienie wymagań dziedziny, (c) faktyczne możliwości wykorzystania w systemach zewnętrznych. Jest to element konieczny prawidłowego *cyklu życia ontologii* po opracowaniu pierwszej stabilnej wersji ontologii. Jednak poprawna ocena ontologii tych rozmiarów może stanowić problem dla osoby niepracującej nad jej rozwojem. Dodatkowym czynnikiem utrudniającym jej analizę jest rozproszona architektura tej ontologii. Składa się ona z wielu modułów umieszczonych w różnych przestrzeniach nazw (ang. namespace). Skuteczność oraz poprawność analizy ontologii dokonanej przez ekspertów związanych z daną dziedziną może zagwarantować jej poprawny rozwój w czasie oraz możliwości dalszego wykorzystania ontologii systemu TSS, również w systemach zewnętrznych. Aby umożliwić przeprowadzenie sprawnej oceny i uzyskanie wsparcia dla dalszego rozwoju danej ontologii potrzeba prostej i intuicyjnej publikacji stworzonej ontologii. Obecnie dostępne narzędzia, będące w większości rozszerzeniami aplikacji Protégé, nie dostarczają możliwości stworzenia takiej publikacji dla ontologii systemu TSS.

Narzędzia te nakładają ograniczenia: (a) językowe – DocGen tworzy publikacje jedynie z ontologii OWL, (b) architektoniczne – HTML Export generuje błędną dokumentację dla ontologii składającej się z większej niż 1 ilości plików źródłowych. Wykluczają one bezpośrednie zastosowanie wymienionych aplikacji, gdyż ontologia dziedziny podróży opracowana na potrzeby systemu TSS jest napisana w języku RDF(S), a jej główną cechą architektoniczną jest rozbicie architektury na wiele modułów.

Należy podkreślić również, że nie ma obecnie dostępnych publicznie narzędzi umożliwiających obejście przedstawionych w poprzednim akapicie ograniczeń. Zarówno aplikacje transformujące RDF(S) do OWL jak i programy przeprowadzające integrację modułów ontologii są jeszcze we wczesnych fazach rozwoju i generują dużą ilość błędów krytycznych, które mają następnie odbicie w błędnie wygenerowanej dokumentacji.

Wymagania publikacji

W celu określenia wymagań, jakie ma spełniać publikacja ontologii należy rozważyć potencjalne grupy jej odbiorców. Biorąc pod uwagę zaproponowaną metodologię tworzenia ontologii (5.1.2) oraz wymagania systemu TSS można wyróżnić następujące grupy:

- Projektanci i programiści podsystemów *PPZ*, *PZZ* i *PDZ*;
- Osoby mające kontakt zawodowy z dziedziną podróży;
- Projektanci i twórcy systemów ontologicznych, którzy po znalezieniu ontologii chcą ją wykorzystać dla własnych potrzeb.

Projektantów i programistów komponentów systemu TSS zainteresują przede wszystkim znaczenia poszczególnych klas i ich atrybutów dodane przez twórcę ontologii w formie napisów w języku naturalnym. Stworzone przez nich systemy będą musiały przyporządkować dane ze stron internetowych do instancji odpowiednich klas ontologii lub dostarczyć użytkownikowi danych, których ten sobie zażyczy.

Osoby należące do dwóch pozostałych grup również będą zainteresowane znaczeniami elementów ontologii. Profesjonaliści z danej dziedziny zwrócą dodatkowo uwagę na stopień pokrycia dziedziny, jaki zapewnia dana ontologia. Z kolei twórcy systemów ontologicznych spojrzą na daną ontologię szczególnie wnikliwie, zwłaszcza pod kątem zaproponowanej hierarchii klas i atrybutów. Będą musieli też określić stopień przydatności znalezionej ontologii na potrzeby własnych systemów.

W związku z powyższym, dokumentacja ontologii musi pozwalać na swobodną nawigację po całej ontologii, umożliwiając w szczególności podgląd:

- listy wykorzystanych modułów ontologii (ontologii szczegółowych, zewnętrznych, itp.);
- list wszystkich klas i atrybutów w danej ontologii;
- list wszystkich klas i atrybutów zdefiniowanych w danym module ontologii;

- komentarzy wszystkich klas i atrybutów;
- list atrybutów własnych i odziedziczonych przez daną klasę;
- list instancji bezpośrednich i pośrednich danej klasy;
- typów wartości wszystkich atrybutów.

Kluczowym aspektem dokumentacji spełniającej powyższe wymagania będzie swoboda nawigacji między każdym elementem dokumentacji.

Narzędzie do tworzenia publikacji ontologii

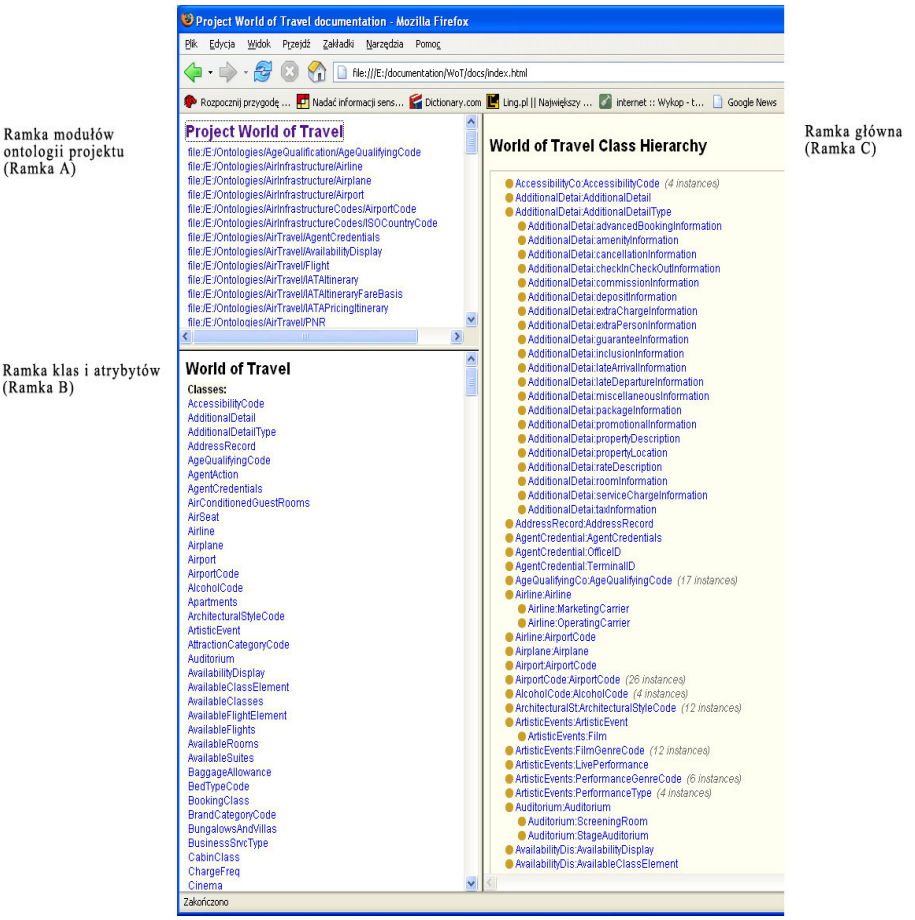
Aplikacja generująca dokumentację ontologii złożonej z wielu modułów napisanych w języku RDF, spełniająca wszystkie wymienione wyżej wymagania publikacji nie jest obecnie dostępna publicznie. W związku z tym została stworzona w ramach tej pracy magisterskiej. Dokumentacja tej aplikacji została zamieszczona w Załączniku C.

Jest napisana w języku Java. Opiera się na wykorzystaniu środowiska Jena opracowanego w laboratoriach rozwojowych Hewlett Packard oraz Interfejsu Programowania Aplikacji Protégé stworzonego na Uniwersytecie w Stanford.

Protégé umożliwia stworzenie dokumentacji dla pojedynczego pliku (węzła sieci ontologii) RDF(S), przy czym dokumentacja ta nie spełnia wszystkich wymagań postawionych w poprzednim paragrafie. Jena dostarcza możliwości analizy składniowej RDF(S) oraz interfejs umożliwiający operowanie na plikach tego formatu. Jedną z funkcji tego interfejsu jest tworzenie sumy zawartości plików RDF(S). Przy pomocy interfejsu zawartego w Jena wszystkie moduły ontologii zostały zintegrowane, a na podstawie wyniku integracji została utworzona dokumentacja z wykorzystaniem biblioteki Protégé. Biblioteka ta wymagała poprawek umożliwiających rozróżnienie instancji bezpośrednich i pośrednich dla klas.

Przykładowa dokumentacja

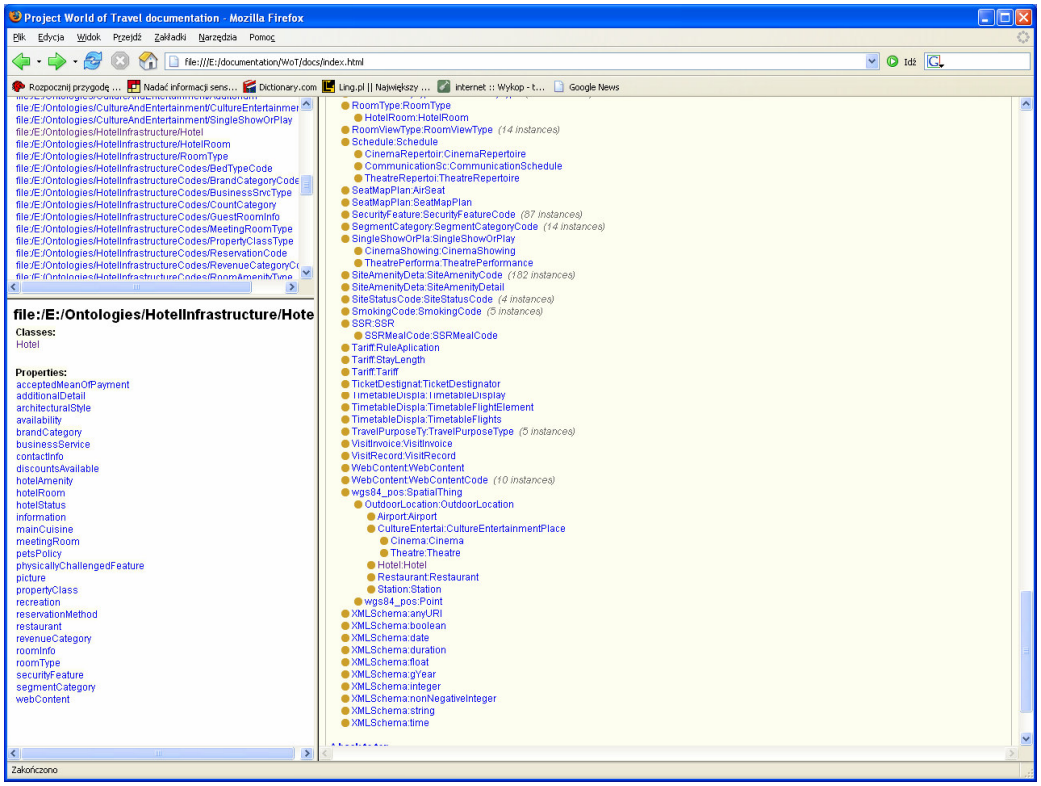
Na stronie startowej (Rysunek 5.5) widać trzy podstawowe elementy dokumentacji ontologii, które zostały wyróżnione jako ramki. Dzięki temu możliwa jest nawigacja na każdym z tych trzech elementów jednocześnie.



Rysunek 5.5. Strona główna dokumentacji

Ramka A zawiera lokalizację modułów, z których składa się ontologia. Są one deklarowane przez użytkownika aplikacji przed wygenerowaniem dokumentacji. W Ramce B początkowo zostają wyświetlone wszystkie klasy i atrybuty istniejące w ontologii. Ramka C domyślnie wyświetla hierarchię klas ontologii. Zawartość ostatniej ramki została w całości wygenerowana przy pomocy poprawionej biblioteki aplikacji Protégé. Po wybraniu dowolnego modułu w Ramce A w B zostaną wyświetlone klasy i atrybuty zdefiniowane wyłącznie w tym module. Rysunek 5.6 pokazuje przykład wybrania modułu o

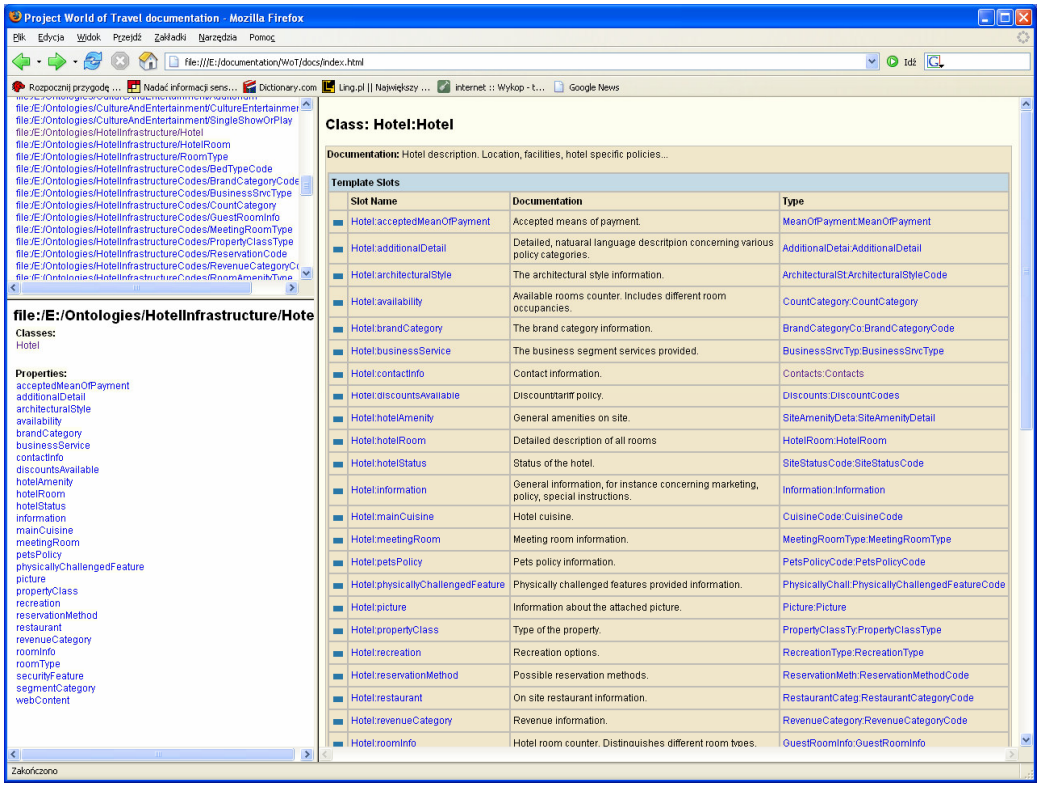
nazwie *Hotel*. W Ramce B zostają pokazane klasa i atrybuty zdefiniowane w tym module.



Rysunek 5.6. Klasy i atrybuty modułu Hotel.

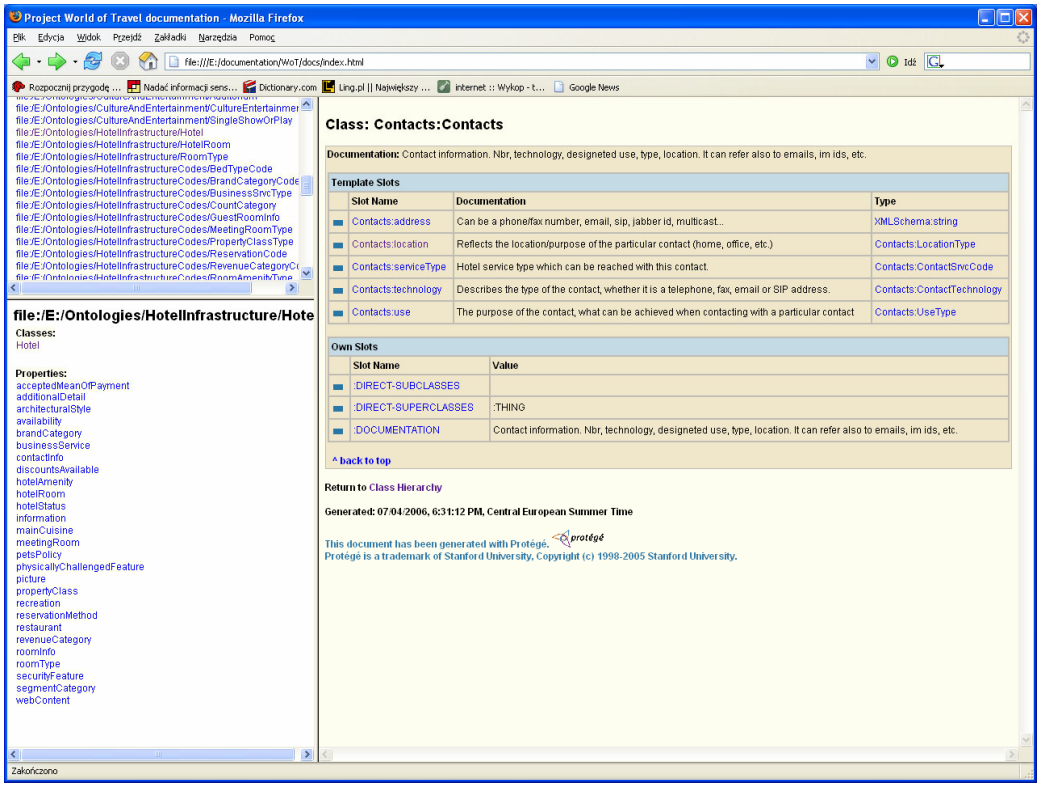
Dokumentacja umożliwia obejrzenie szczegółowego opisu każdej klasy. Rysunek 5.7 ilustruje przykład wyboru klasy *Hotel* i jej szczegółowej dokumentacji zawierającej:

- Opis w języku naturalnym dla danej klasy.
- Listę wszystkich atrybutów własnych.
- Listę wszystkich atrybutów dziedziczonych.
- Opis w języku naturalnym dla każdego atrybutu.
- Wartości przyjmowane przez każdy z atrybutów.
- Listę instancji bezpośrednich.
- Listę instancji pośrednich.
- Listę klas dziedziczących z danej klasy.
- Listę klas, po których dana klasa dziedziczy.



Rysunek 5.7. Szczegółowa dokumentacja klasy Hotel.

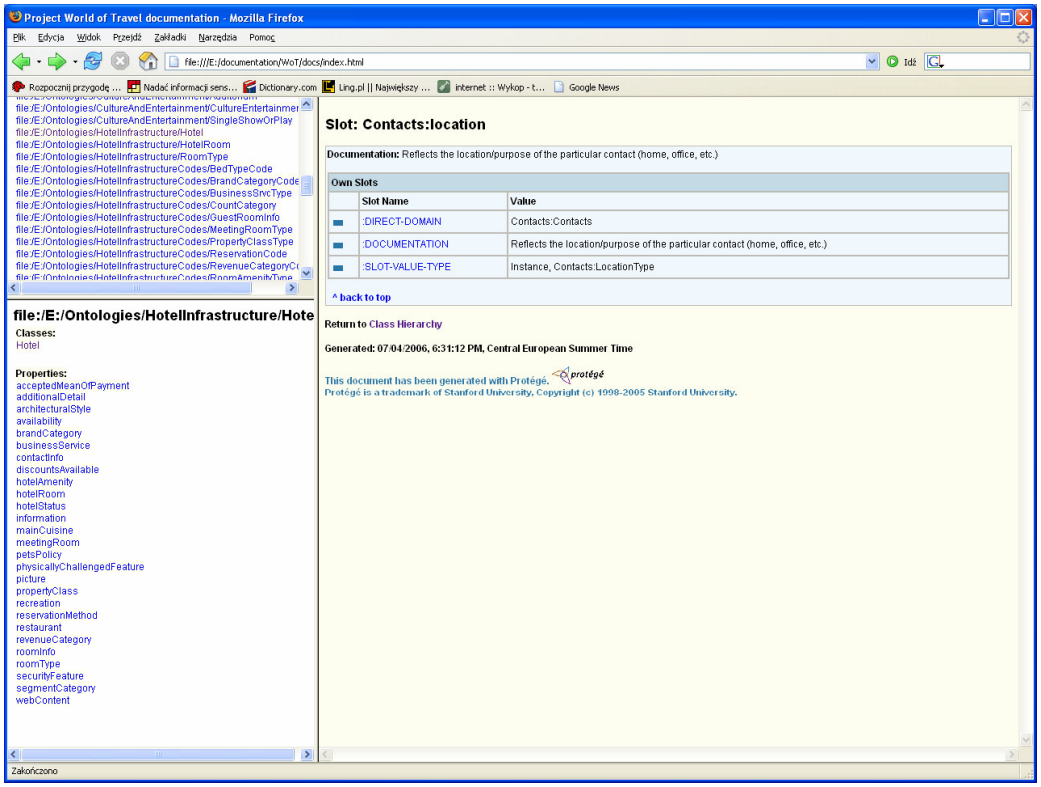
Z widoku szczegółowego jednej klasy można przejść widoku szczegółowego dowolnej klasy będącej wartością atrybutu. Rysunek 5.8 pokazuje widok szczegółowy klasy *Contacts*, będącej wartością atrybutu *contactInfo* klasy *Hotel*. Na tym rysunku widać również stopkę autorską biblioteki aplikacji Protégé.



Rysunek 5.8. Szczegółowa dokumentacja klasy Contacts.

Podobnie, z widoku szczegółowego dowolnej klasy można przejść do dowolnego widoku szczegółowego dowolnego atrybutu danej klasy. Rysunek 5.9 przedstawia taki widok atrybutu *location* klasy *Contacts*. Elementy w nim przedstawione to:

- Opis w języku naturalnym dla danego atrybutu.
- Lista klas, do opisu których dany atrybut jest wykorzystywany.
- Lista klas będących wartościami danego atrybutu.



Rysunek 5.9. Widok szczegółowy atrybutu location klasy Contacts

6. Wnioski

Najważniejszymi wnioskami płynącymi z porównania ontologii filozoficznych z informatycznymi jest różnica ich roli oraz zastosowania. Jako gałąź filozofii, ontologia ma dostarczyć opisy bytów (nie tylko tych istniejących, ale i tych, których istnienie jest możliwe). Z kolei w informatyce, jej rolą jest zapewnienie opisu semantycznego dla różnego rodzaju zasobów elektronicznych. Te zasoby mogą w pewnych przypadkach reprezentować byty świata rzeczywistego.

Metody stosowane w filozofii przez 2500 lat nie dały możliwości stworzenia jednej powszechnie obowiązującej ontologii. Był to najważniejszy cel filozofów zajmujących się ontologią i mimo, że nie został on osiągnięty, stosowane przez filozofów metody stanowią dobrą podstawę dla metod informatycznych. W informatyce bardzo ważny jest pragmatyczny charakter ontologii. Są one z reguły rozwijane w ramach konkretnych systemów, ale powinny również umożliwiać opis wiedzy na potrzeby aplikacji funkcjonujących poza danymi systemami.

Za właściwe podejścia do rozwoju ontologii na potrzeby aplikacji internetowych uważam metody wykorzystujące koncepty: *dynamicznej sieci znaczeń* oraz *cyklu życia ontologii*. Obie te podejścia kładą nacisk na zmienność ontologii w czasie. Dodatkowo, pierwsze z nich kładzie nacisk na ich rozproszony charakter, co daje realną możliwość rozwoju ontologii w Internecie, który jest środowiskiem bardzo zróżnicowanym i dynamicznym.

Zakładany praktyczny charakter ontologii informatycznych wymaga stosowania standardowych języków ontologicznych, w szczególności RDF(S) oraz OWL. Są one na tyle elastyczne, że nie ograniczają możliwości budowy ontologii. Jednocześnie, są formalizmami, które można łatwo przetwarzać za pomocą aplikacji komputerowych. Same ontologie umożliwiające opisywanie danych na potrzeby użytkowników Internetu powinny być publicznie dostępne, aby projektanci systemów aplikacji mogli skorzystać z istniejących ontologii w nowo powstających systemach.

O ile istnieją narzędzia pozwalające edytować ontologie w językach RDF(S) oraz OWL, o tyle brakuje obecnie publicznie dostępnych narzędzi do budowy ontologii w wymienionych wyżej językach, z uwzględnieniem *cyklu życia ontologii* oraz *dynamicznej sieci znaczeń*. Ich rozwój jest kluczowym elementem wprowadzenia ontologii do systemów aplikacji oraz Internetu. Takie narzędzia przyczyniłyby się do upowszechnienia ontologii w szerokiej gamie rozwiązań biznesowych, a to z kolei daje szansę na ich dalszy, pozaakademicki rozwój.

Zastosowanie ontologii w praktyce daje większe możliwości eksploracji danych. W najbardziej ogólnym zakresie ontologie umożliwiają budowę terminologii z uwzględnieniem relacji między występującymi w nich terminami. Dane opisane przy pomocy tych słów przy pomocy wybranego formalizmu mogą zostać zinterpretowane przez aplikacje przetwarzające semantycznie opisane dane. Definicja i implementacja logiki biznesowej takich aplikacji może abstrahować zupełnie od samych danych, powinna bazować na terminach i relacjach ujętych w ontologii.

Z kolei, z punktu widzenia rozproszonych systemów zarządzania wiedzą, ontologie zapewniają:

- większą przenośność danych,
- rozszerzalność systemów o nowe komponenty korzystające z opisanych danych,
- odnajdywanie danych na podstawie opisu semantycznego, przeprowadzenie wnioskowania,
- łatwe udostępnianie danych systemom zewnętrznym.

W szczególności ontologia opracowana przeze mnie w ramach pracy magisterskiej powstała w celu opisu danych dla dziedziny miejsca zakwaterowania z punktu widzenia podróżnika. Ostateczna, zintegrowana wersja ontologii dziedziny podróży, której element stanowi ontologia dziedziny miejsca zakwaterowania została umieszczona na płycie kompaktowej dołączonej do pracy magisterskiej. Wersja ontologii dziedziny miejsca zakwaterowania otrzymana przeze mnie po przeprowadzeniu integracji

z ontologiami dziedzin transportu lotniczego i restauracji, wraz z elementami wspólnymi tych dziedzin znajduje się w Załączniku A. Aktualna wersja ontologii dziedziny podróży umożliwia jej zastosowanie w systemie TSS lub dowolnym innym systemie, na którego potrzeby okaże się ona wystarczająca. Ontologia ta lub jej poszczególne elementy mogą również być wykorzystane jako składniki innych ontologii. Właśnie w celu właściwego wykorzystania tej ontologii powstała szczegółowa dokumentacja wygenerowana przez narzędzie do tworzenia dokumentacji ontologii napisanej w standardzie RDF(S). Generator dokumentacji został stworzony w ramach niniejszej pracy magisterskiej.

Wykaz źródeł

Książki i publikacje

[Abramowicz i Kalczyński, 2002] Abramowicz, W., Kalczyński, P. J., *Building and Taking Advantages of the Digital Library for the Organizational Data Warehouse*. W: Cobb, M. et. al. (eds.), *Proceedings of the Sec-ond Southern Conference on Computing*, Hattiesburg, Mississippi, USA, CD 2002.

[Allen, 1984] Allen J.F., *Towards a general theory of action and time.* W: *Artificial Intelligence*, 23, 123-154, 1984.

[Angryk et al., 2001] Angryk, R., Kołodziej, K., Fiedorowicz, I., Paprzycki, M., Cobb, M., Ali, D., Rahimi, S., *Development of a travel support system based on intelligent agent technology*. W: Niwiński, S. (ed.), *Proceedings of the PIONIER 2001 Conference* , Technical University of Poznań Press, Poznań, Poland, 243-255, 2001.

[Arystoteles] Arystoteles, *Metafizyka*, rozdział 9.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J. and Lassila, O., *The semantic web*. Scientific American, May, 2001.

[Borgo et al., 1996] Borgo, S., Guarino, N., Masolo, C., *A Pointless Theory of Space Based on Strong Connection and Consequence*. W: Aiello, L. C., Doyle, J. (eds), *Principles of Knowledge Representation and Reasoning (KR96)*, Morgan Kaufmann, 1996.

[Borgo et al., 1997] Borgo, S., Guarino, N., Masolo, C., *An Ontological Theory of Physical Objects*. W: Ironi, L. (ed.), *Proceedings of Eleventh International Workshop on Qualitative Reasoning (QR'97)*, Cortona (Italia), 3-6 Giugno, 223-231, 1997.

[Brickley et al., 1998] Brickley, D., Guha, R., Layman, A. (eds), *Resource Description Framework (RDF) Schema Specification*, W3C Working Draft, 1998.

[Casati i Varzi, 1995] Casati, R., Varzi, A., *Holes and Other Superficialities*, MIT Press Cambridge, MA, 1995.

[Daconta et al., 2003] Daconta, M.C., Obrst, L.J., Smith, K.T, *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Indianapolis, Indiana, USA: Wiley Publishing, Inc, 2003.

[Davies et al., 2003] Davies, J., Fensel, D. i Harmelen, F. van, *Towards the Semantic Web: Ontology-Driven Knowledge Management*. John Wiley & Sons 2003.

[Encyklopedia, 2005] Encyklopedia Gazety Wyborczej, Wydawnictwo Naukowe PWN, 2005.

[Fensel, 2004] Fensel, D., *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag Berlin Heidelberg, 2004.

[Fensel i Groenboom, 1997] Fensel, D., Groenboom, R., *Specifying knowledge-based systems with reusable components*. W: *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering (SEKE'97)*, Madrid, 1997.

[Fox et al., 1993] Fox, M. S., Chionglo, J.F., Fadel, F.G., *A common sense model of the enterprise.* W: *Proceedings of 2nd Industrial Engineering Research Conference*, Norcross, GA, USA, 1993.

[Fox i Gruninger, 1997] Fox, M.S., Gruninger, M., *On ontologies and enterprise modeling*. W: *Proceeding of the International Conference on Enterprise Integration Modelling Technology'97*, Springer, 1997.

[Friedman-Noy i Hafner, 1997] Fridman-Noy, N., Hafner, C.D., *The State of the Art in Ontology Design*, AI Magazine, 18(3):53-74, 1997.

[Gawinecki, 2005] Gawinecki, M., Vetulani, Z., Modelowanie użytkownika na podstawie interakcji z systemem opartym o technologie WWW. Praca magisterska broniona na Wydziale Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu, 2005.

[Gawinecki et al., 2005a] Gawinecki, M., Gordon, M., Paprzycki, M., Szymczak, M., Vetulani, Z., Wright, J., *Enabling Semantic Referencing of Selected Travel Related Resources*. W: Abramowicz, W. (ed.), *Proceedings of the 8 th International Conference on Business Information Systems* (BIS 2005), Poznań University of Economics Press, Poznań, Poland, 271-288, 2005.

[Gawinecki et al., 2005b] Gawinecki, M., Gordon, M., Nguyen, N.T., Paprzycki, M., Szymczak, M., *RDF Demarcated Resources in an Agent Based Travel Support System*. W: Goliński, M. et. al. (eds.), *Informatics and Effectiveness of Systems*, PTI Press, Katowice, 303-310, 2005.

[Gawinecki et al., 2005c] Gawinecki, M., Gordon, M., Nguyen, N. T., Paprzycki, M., Vetulani, Z., *Ontologically Demarcated Resources in an Agent Based Travel Support System*. W: Katarzyniak, R. K. (ed.) *Ontologies and Soft Methods in Knowledge Management*, Advanced Knowledge International, Adelaide, Australia, 219-240, 2005.

[Gawinecki et al., 2005d] Gawinecki, M., Kruszyk, M., Paprzycki M., *Ontology-based Stereotyping in a Travel Support System*. W: *Proceedings of the XXI Fall Meeting of Polish Information Processing Society* , PTI Press, 73-85, 2005.

[Gordon i Paprzycki, 2005] Gordon, M. i Paprzycki, M., *Designing Agent Based Travel Support System*. W: *Proceedings of the ISPDC 2005 Conference*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2005.

[Gordon et al., 2005] Gordon, M., Kowalski, A., Paprzycki, M., Pełech, T., Szymczak, M., Wąsowicz, T., *Ontologies in a Travel Support System*. W: Bem, D. J. et. al. (eds.) *Internet 2005* , Technical University of Wroclaw Press, 285-300, 2005.

[Gruber, 1993] Gruber, T.S., *A translation approach to portable ontology specifications*. W: *Knowledge Acquisition*, 5, 199-220, 1993.

[Gruber, 1995] Gruber, T. R., *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*. W: *International Journal of Human and Computer Studies*, 1995. 43(5/6), 907–928. An outline of motivations for the development of ontologies.

[Guarino, 1998] Guarino, N., *Formal Ontology in Information Systems.* IOS Press, Amsterdam, 1998.

[Heijst et al., 1997] Heist, G. van, Schreiber, A., Wielinga, B., *Using Explicit Ontologies in KBS Development*, Interantional Journal of Human-Computer Studies, 46:183-292, 1997.

[Hesse, 2002] Hesse, W., *Ontologie(n)*, GI Gesellschaft für Informatyk e.V. Informatyk-Lexikon, 2002. Dostępny w Internecie pod adresem: http://www.gi-ev.de/informatik/inf-lex-ontologien.shtml

[IATA, 2006a] IATA Reservations Service Manual, 23rd Edition, Effective 1st June, 2006.

[IATA, 2006b] IATA Airline Coding Directory – Airline Destignators, Dec 1, 2005 until Dec 1, 2006

[IATA, 2006c] IATA Standard Schedules Information Manual, Mar 1, 2006 until Sep 30, 2006

[Ingarden, 1964] Ingarden, R., *Time and Modes of Being*. Tłumaczenie: Michejda, H.R. Springfield, Ill.: Thomas, C., Translated extracts from a masterly four-volumework in realist ontology entitled *The Problem of the Existence of the World*, 1964.

[Ingarden, 1973] Ingarden, R., *The Literary Work of Art: An Investigation on the Borderlines of Ontology, Logic, and Theory of Literature,* Evanston: Northwestern University ress, 1973.

[Leinfallner et al., 1982] Leinfellner, W., Kraemer, E., Schank J.(eds.), *Preface by The Editors*. W: *Language and Ontology. Proceedings of the Sixth International Wittgenstein Symposium. 23th to 30th August 1981 Kirchberg am Wechsel (Austria)*, Wien, Hölder-Pichler-Tempsky, 18-20, 1982.

[Lejewski, 1958] Lejewski, Cz., On Lesniewski's Ontology. W: Ratio tom I, n.2, 150-176, 1958, (Wydrukowane ponownie w: Srzednicki, J.T.J., Rickey, V. F. (eds), Lesniewski' Systems. Ontology and Mereology. The Hague, Martinus Nijhoff, 123-148, 1984).

[Øhrstrøm et al., 2004] Øhrstrøm, P., Schärfe, H., *A Priorean Approach to Time Ontologies.* W: Wolff, K.E. et al. (eds.), *Proceedings of ICCS 2004*, LNAI 3127, 388-401, 2004.

[Øhrstrøm et al., 2005] Øhrstrøm, P., Andersen, J., Schärfe, H., *What Has Happened to Ontology*. Dau, W: F., Mugnier, M.-L., Stumme, G. (Eds), *ICCS 2005*. Springer Verlag, LNAI 3596, 425 – 438, 2005.

[Nagrodkiewicz i Paprzycki, 2005] Nagrodkiewicz, P., Paprzycki, M., *Automated Travel Planning*. W: *Proceedings of the XXI Fall Meeting of Polish Information Processing Society* , PTI Press, 205-212, 2005.

[Niles i Pease, 2001] Niles, I., Pease, A., *Origins of the Standard Upper Merged Ontology: A Proposal for the IEEE Standard Upper Ontology*. W: *Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology*, Seattle, Washington, 2001.

[Nwana i Ndumu, 1999] Nwana, H., Ndumu, D., *A perspective on software agents research*. W: *The Knowledge Engi-neering Review*, 14, 2, 1-18, 1999.

[Quine, 1953] Quine, W. V. O., *On What There Is*. W: *From a Logical Point of*

View, New York: Harper & Row, 1953.

[Russel i Norwig, 1995] Russell, S., Norwig, P., *Artificial Intelligence: A Modern Approach*, 1995.

[Schlenoff et al., 2000] Schlenoff, C., Gruninger, M., Tissot, F., Valois, J., Lubell, J., Lee, J., *The Process of Specification Language (PSL): Overview and Version 1.0 Specification.* W: NISTIR 6459, Natioanal Institute of Stardards and Technology, Gaithersburg, MD, 2000.

[Smith, 1994] Smith, B., *Fiat Objects*. W: Guarino, N., Vieu, L., Pribbenow, S. (eds), *Parts and Wholes: Conceptual Part-Whole Relations and Formal Mereology, 11th European Conference on Artificial Intelligence*, Amsterdam: European Coordinating Committee for Artificial Intelligence, 15-23, Amsterdam 1994.

[Smith, 1995] Smith, B., *Mereotopology: A Theory of Parts and Boundaries*. W: *Data and Knowledge Engineering*, 20, 287-303.

[Smith, 2003] Smith, B., *Preprint version of chapter "Ontology"*, in L. Floridi (ed.), *Blackwell Guide to the Philosophy of Computing and Information*, Oxford: Blackwell, 155–166, 2003.

[Sowa, 1984] Sowa, J. F., *Conceptual Structures: Information Processing in Mind and Machine*, Addison Wesley, Reading, MA, 1984.

[Sowa, 2000] Sowa, J. F., *Knowledge Representation. Logical, Philosophical and Computational Foundations.* Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[Studer et al., 1996] Studer, R., Eriksson, H., Gennari, J.H., Tu, S.W., Fensel, D., Musen, M., *Ontologies and the configuration problem-solving methods*. W: Gaines, B.R., Musen, M.A. (eds), *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, 1996.

[Sure i Studer, 2003] Sure, Y., Studer, R., *A Methodology for Ontology-based Knowledge Management*. W: Davies, J., Fensel, D. i Harmelen, F. van, *Towards the Semantic Web: Ontology-Driven Knowledge Management*. John Wiley & Sons 2003.

[Szymczak et al., 2006] Szymczak, M., Gawinecki, M., Vukmirovic, M., Paprzycki, M., *Ontological reusability in state-of-the-art semantic languages*, w opracowaniu.

[Vetulani, 2003a] Vetulani, Z., *Linguistically Motivated Ontological Systems*. W: Callaos, N. et al. (eds), *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, vol. XII (Information Systems, Technologies and Applications: II), 395-400 Int. Inst. of Informatics and Systemics, Orlando, Florida, USA, 2003.

[Vetulani, 2003b] Vetulani, Z., *Systemy ontologiczne wobec rozwoju Dziedzin.* W: Kłopotek, M. i Tchórzewski, J. (eds), *Materiały XIII Ogólnopolskigo Konwersatorium "Sztuczna Inteligencja. Organizacje wirtualne"*, AI-18'2003, Nr 22, str. 73-84, Wyd. Akademii Podlaskiej, Siedlce, 2003.

[Vetulani, 2004] Vetulani, Z., *Komunikacja człowieka z maszyną. Komputerowe modelowanie kompetencji językowej*, Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2004.

[Vukmirovic et al., 2006a] Vukmirovic M., Ganzha M., Paprzycki M., *Developing a Model Agent-based Airline Ticket Auctioning System*. W: *Proceedings of the IIPWM Conference* w opracowaniu.

[Vukmirovic et al., 2006b] Vukmirovic M., Szymczak M., Ganzha M., Paprzycki, M., *Utilizing Ontologies in an Agent-based Airline Ticket Auctioning System*. W: *Proceedings of the 28th ITI Conference*, w opracowaniu.

Źródła internetowe

[Agentcities, 2002a] *Agentcities Web*, http://www.agentcities.org

[AgentCities, 2002b] *Restaurant ontology, version 4*, http://www.csd.abdn.ac.uk/research/AgentCities/ontologies/restaurant-v4.daml

[ChefMoz, 2005] *ChefMoz dining guide*, http://chefdmoz.org/.

[CYCORP, Inc., 2002] *CYCORP, Inc.*, http://www.cyc.com/

[DCMI, 1995] *Dublin Core Metadata Initiative (DCMI)*, http://dublincore.org/

[DERI, 2004] *E-Tourism Working Group Site*, http://deri.at/research/projects/e-tourism

[FIPA, 2002a] *Foundation for Intelligent Physical Agents*, http://www.fipa.org/

[FIPA, 2002b] *FIPA Agent Communication Language Specifications*, http://www.fipa.org/repository/aclspecs.html

[FIPA, 2002c] *FIPA SL Content Language Specification*, http://www.fipa.org/specs/fipa00008/SC00008I.html

[Formatology, 2000a] *What is Ontology? Definitions by leading philosophers*, http://www.formalontology.it/section_4.htm

[Formatology, 2000b] *Notes on the History of Ontology*, http://www.formalontology.it/history.htm

[Formatology, 2000c] *Ontology. A resource guide for philosophers*, http://www.formalontology.it/index.htm

[Baclawski, 2006] *Baclawski, K., Versatile Information Systems - Tutorial on the Semantic Web*, colab.cim3.net/file/wor**k**/SICoP/2006-04-2728/KBaclaw**k**si04282006.ppt

[Cambia, 2004] *Cambia Service*, http://zurich.agentcities.whitestein.ch/Services/Cambia.html

[IATA, 2004] *International Air Transport Association*, http://www.iata.org/

[ICHotels, 2001] *InterContinental Hotels Group Hotel Reservations*, http://www.ichotelsgroup.com/

[Jena 2003a] *Jena 2 Ontology API – General concepts*, http://jena.sourceforge.net/ontology/index.html#generalConcepts

[Jena 2003b] *Jena Documentation*, http://jena.sourceforge.net/documentation.html

[Jena 2003c] *Jena Semantic Web Framework*, http://jena.sourceforge.net/

[Marriot, 1996] *Hotel, resort and vacation information from Marriott.com*, http://marriott.com/default.mi

[Ontolingua, 1995] *Stanford KSL Network Services*, http://ontolingua.stanford.edu

[OpenCYC, 2002] *OpenCYC*, http://www.cyc.com/opencyc

[OTA, 2001] *OpenTravel Alliance*, http://www.opentravel.org/

[OTA, 2006] *OpenTravel Alliance*, http://www.opentravel.org/members.cfm

[Protégé, 2006] *The Protégé Ontology Editor and Knowledge Acquisition System*, http://protege.stanford.edu/

[Sheraton, 2006] *Sheraton Hotels & Resorts* http://www.starwoodhotels.com/sheraton/index.html

[Sofitel] *Sofitel hotels & resorts: The Official Site for Booking luxury hotels & resorts on-line*, http://www.sofitel.com/hotel-cms/gb/accueil/index.shtml

[TOVE, 2005] *TOVE Ontologies*, http://www.eil.utoronto.ca/tove/toveont.html

[TravelCiti, 2006] *Vacation Deals, Last Minute Hotel + Flight Travel Packages*, http://www.travelciti.com

[TravelWeb, 2006] *TravelWeb.com*, http://www.travelweb.com

[SUO WG, 2003] *Standard Upper Ontology Working Group*, http://suo.ieee.org.

[W3C, 1989] *The original proposal of the WWW, HTMLized*, http://www.w3.org/History/1989/proposal.html

[W3C, 1993] *Web Naming and Addressing Overview*, http://www.w3.org/Addressing/

[W3C, 1998] *W3C Semantic Web*, http://www.w3.org/2001/sw/

[W3C, 1998a] *Web design issues; What a semantic can represent*, http://www.w3.org/DesignIssues/RDFnot.html

[W3C, 1998b] *Web Architecture from 50,000 feet*, http://www.w3.org/DesignIssues/Architecture.html

[W3C, 1998c] *Semantic Web roadmap*, http://www.w3.org/DesignIssues/Semantic.html

[W3C, 1998d] *Notation3 (N3) A readable RDF syntax*, http://www.w3.org/DesignIssues/Notation3.html

[W3C, 2000] *W3C XML Schema*, http://www.w3.org/XML/Schema

[W3C, 2004a] *About W3C*, http://www.w3.org/Consortium/

[W3C, 2004b] *XML Schema PART 0: Primer Second Edition*, http://www.w3.org/TR/xmlschema-0/

[W3C, 2004c] *RDF Vocabulary Description Language 1.0: RDF Schema*, http://www.w3.org/TR/rdf-schema/

[W3C, 2004d] *Web Ontology Language*, http://www.w3.org/2004/OWL/

[W3C, 2004e] *OWL Web Ontology Language*, http://www.w3.org/TR/owl-features/

[W3C, 2005] *Design Issues for the World Wide Web*, http://www.w3.org/DesignIssues/Overview.html

[W3C, 2006a] *SPARQL Query Language for RDF*, http://www.w3.org/TR/rdf-sparql-query/

[W3C, 2006b] *WGS84 Geo Positioning: an RDF vocabulary*, http://www.w3.org/2003/01/geo/wgs84_pos

[W3Schools, 1999] *XML Schema Tutorial*, http://www.w3schools.com/schema/default.asp

[WebHotels, 2006] *Cheap Hotels - Discount Hotel Reservations*, http://www.web-hotels.com

[Webjunction, 1995] *Number of documents in the world wide web?*, http://lists.webjunction.org/wjlists/web4lib/1995-October/002069.html

[WGS, 1984] *World Geodetic* System, http://www.wgs84.com/

[Wikipedia, 2001a] *Ontologia – Wikipedia*, http://pl.wikipedia.org/wiki/Ontologia

[Wikipedia, 2006a] *Reference.com/Encyclopedia*, http://www.reference.com/browse/wiki/Concept

[Wikipedia, 2006b] *Uniform Resource Locator*, http://en.wikipedia.org/wiki/URL

[Wikipedia, 2006c] *Uniform Resource Identifier*, http://en.wikipedia.org/wiki/Uniform_Resource_Identifier

[Wikipedia, 2006d] *Resource Description Framework*, http://en.wikipedia.org/wiki/Resource_Description_Framework

[WordNet, 2006a] *WNSTATS(7WN)*, http://wordnet.princeton.edu/man/wnstats.7WN

[WordNet, 2006b] *WordNet – Princeton University Cognitive Science Laboratory*, http://wordnet.princeton.edu/

Załącznik A. Źródła ontologii

Ontologia miejsca zakwaterowania składa się z autonomicznych modułów, które mogą być wykorzystane do opisu zasobów w systemach zewnętrznych. Poniżej przedstawiam moduły i zdefiniowane w ich ramach klasy, zaimplementowane w notacji N3. Klasy będące odpowiednikami poszczególnych jednostek ze specyfikacji OTA zawierają w komentarzu odpowiednią uwagę. Dla nich zostanie przytoczona wyłącznie definicja i komentarz. Wszystkie ich instancje równoważne kodom OTA będą pominięte, ze względu na ich dużą ilość. Pełna implementacja ontologii miejsca zakwaterowania jest dostępna na załączonym do niniejszej pracy magisterskiej krążku CD.

A.1. Moduł infrastruktury

Klasa *Hotel* jest podstawową klasą ontologii miejsca zakwaterowania. Umożliwia opisanie dowolnego typu kwaterunku. Wszystkie jej atrybuty przyjmują wartości typów złożonych ujętych również w tej ontologii.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

@prefix dis: <Payment/Discounts#>.
@prefix rmc: <HotelVisitCodes/ReservationMethodCode#>.

@prefix ctt: <SiteFeatures/WebContent#>.
@prefix adt: <SiteFeatures/AdditionalDetail#>.
@prefix arc: <SiteFeatures/ArchitecturalStyleCode#>.
@prefix sam: <SiteFeatures/SiteAmenityDetail#>.
@prefix sec: <SiteFeatures/SecurityFeatureCode#>.
@prefix ssc: <SiteFeatures/SiteStatusCode#>.
@prefix seg: <HotelInfrastructureCodes/SegmentCategoryCode#>.
@prefix pet: <SiteFeatures/PetsPolicyCode#>.
@prefix phc: <Contacts/Contacts#>.
@prefix phy: <SiteFeatures/PhysicallyChallengedFeatureCode#>.
@prefix pic: <SiteFeatures/Picture#>.
@prefix inf: <SiteFeatures/Information#>.

@prefix res: <Restaurant/RestaurantCategoryCode#>.
@prefix cui: <Restaurant/CuisineCode#>.

@prefix bcc: <HotelInfrastructureCodes/BrandCategoryCode#>.
@prefix bus: <HotelInfrastructureCodes/BusinessSrvcType#>.
@prefix prp: <HotelInfrastructureCodes/PropertyClassType#>.
@prefix rcc: <HotelInfrastructureCodes/RevenueCategoryCode#>.
@prefix mrt: <HotelInfrastructureCodes/MeetingRoomType#>.
@prefix room: <HotelInfrastructure/RoomType#>.
@prefix roominf: <HotelInfrastructureCodes/GuestRoomInfo#>.
@prefix ccat: <HotelInfrastructureCodes/CountCategory#>.

@prefix rec: <Recreation/RecreationType#>.
```

```
@prefix siz: <VehicleSize/SizeOfVehicle#>.
@prefix loc: <Location/OutdoorLocation#>.
@prefix pmt: <Payment/MeanOfPayment#>.
@prefix hroom: <HotelInfrastructure/HotelRoom#>.
@prefix base: <HotelInfrastructure/Hotel#>.

base:Hotel a rdfs:Class;
     rdfs:subClassOf loc:OutdoorLocation;
     rdfs:comment "Hotel description.
     Location, facilities, hotel specific policies...".

base:additionalDetail a rdf:Property;
     rdfs:comment "Detailed, natuaral language descritpion
                         concerning various policy categories.";
     rdfs:domain base:Hotel;
     rdfs:range adt:AdditionalDetail.

base:architecturalStyle a rdf:Property;
     rdfs:comment "The architectural style information.";
     rdfs:domain base:Hotel;
     rdfs:range arc:ArchitecturalStyleCode.

base:brandCategory a rdf:Property;
     rdfs:comment "The brand category information.";
     rdfs:domain base:Hotel;
     rdfs:range bcc:BrandCategoryCode.

base:businessService a rdf:Property;
     rdfs:comment "The business segment services provided.";
     rdfs:domain base:Hotel;
     rdfs:range bus:BusinessSrvcType.

base:webContent a rdf:Property;
     rdfs:comment "Attached multi-media content.";
     rdfs:domain base:Hotel;
     rdfs:range ctt:WebContent.

base:availability a rdf:Property;
     rdfs:comment "Available rooms counter.
                  Includes different room occupancies.";
     rdfs:domain base:Hotel;
     rdfs:range ccat:CountCategory.

base:discountsAvailable a rdf:Property;
     rdfs:comment "Discount/tariff policy.";
     rdfs:domain base:Hotel;
     rdfs:range dis:DiscountCodes.

base:roomInfo  a rdf:Property;
     rdfs:comment "Hotel room counter.
                  Distinguishes different room types.";
     rdfs:domain base:Hotel;
     rdfs:range roominf:GuestRoomInfo.

base:hotelAmenity a rdf:Property;
     rdfs:comment "General amenities on site.";
     rdfs:domain base:Hotel;
     rdfs:range sam:SiteAmenityDetail.

base:roomType a rdf:Property;
     rdfs:comment "Room types.";
     rdfs:domain base:Hotel;
     rdfs:range room:RoomType.

base:hotelRoom a rdf:Property;
     rdfs:comment "Detailed description of all rooms";
     rdfs:domain base:Hotel;
     rdfs:range hroom:HotelRoom.

base:hotelStatus a rdf:Property;
     rdfs:comment "Status of the hotel.";
     rdfs:domain base:Hotel;
```

```
     rdfs:range ssc:SiteStatusCode.

base:information  a rdf:Property;
     rdfs:comment "General information,
                  for instance concerning marketing,
                  policy, special instructions.";
     rdfs:domain base:Hotel;
     rdfs:range inf:Information.

base:mainCuisine  a rdf:Property;
     rdfs:comment "Hotel cuisine.";
     rdfs:domain base:Hotel;
     rdfs:range cui:CuisineCode.

base:meetingRoom a rdf:Property;
     rdfs:comment "Meeting room information.";
     rdfs:domain base:Hotel;
     rdfs:range mrt:MeetingRoomType.

base:acceptedMeanOfPayment a rdf:Property;
     rdfs:comment "Accepted means of payment.";
     rdfs:domain base:Hotel;
     rdfs:range pmt:MeanOfPayment.

base:petsPolicy a rdf:Property;
     rdfs:comment "Pets policy information.";
     rdfs:domain base:Hotel;
     rdfs:range pet:PetsPolicyCode.

base:physicallyChallengedFeature a rdf:Property;
     rdfs:comment "Physically challenged features provided information.";
     rdfs:domain base:Hotel;
     rdfs:range phy:PhysicallyChallengedFeatureCode.

base:picture a rdf:Property;
     rdfs:comment "Information about the attached picture.";
     rdfs:domain base:Hotel;
     rdfs:range pic:Picture.

base:contactInfo a rdf:Property;
     rdfs:comment "Contact information.";
     rdfs:domain base:Hotel;
     rdfs:range phc:Contacts.

base:propertyClass a rdf:Property;
     rdfs:comment "Type of the property.";
     rdfs:domain base:Hotel;
     rdfs:range prp:PropertyClassType.

base:recreation a rdf:Property;
     rdfs:comment "Recreation options.";
     rdfs:domain base:Hotel;
     rdfs:range rec:RecreationType.

base:reservationMethod  a rdf:Property;
     rdfs:comment "Possible reservation methods.";
     rdfs:domain base:Hotel;
     rdfs:range rmc:ReservationMethodCode.

base:restaurant a rdf:Property;
     rdfs:comment "On site restaurant information.";
     rdfs:domain base:Hotel;
     rdfs:range res:RestaurantCategoryCode.

base:revenueCategory  a rdf:Property;
     rdfs:comment "Revenue information.";
     rdfs:domain base:Hotel;
     rdfs:range rcc:RevenueCategoryCode.

base:securityFeature a rdf:Property;
     rdfs:comment "Provided security measures.";
     rdfs:domain base:Hotel;
```

```
     rdfs:range sec:SecurityFeatureCode.

base:segmentCategory a rdf:Property;
     rdfs:comment "Segment category.";
     rdfs:domain base:Hotel;
     rdfs:range seg:SegmentCategoryCode.
```

Klasa HotelRoom (pol. Pokój hotelowy) pozwala opisać każdy pokój w danym miejscu zakwaterowania. Dziedziczy z klasy RoomType wszystkie atrybuty dotyczące konkretnego typu pokoju w danej lokalizacji a atrybuty własne klasy HotelRoom pozwalają na bardziej indywidualny opis każdego pokoju.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rlt: <Location/IndoorLocation#>.
@prefix rmt: <HotelInfrastructure/RoomType#>.
@prefix rvt: <HotelInfrastructureCodes/RoomViewType#>.
@prefix base: <HotelInfrastructure/HotelRoom#>.

base:HotelRoom a rdfs:Class;
     rdfs:subClassOf rmt:RoomType;
     rdfs:comment "Record of a particular hotel room.
                         Its location, type, etc.".

base:number  a rdf:Property;
     rdfs:comment "Hotel room number.";
     rdfs:domain base:HotelRoom;
     rdfs:range xsd:nonNegativeInteger.

base:roomLocation  a rdf:Property;
     rdfs:comment "Location on the site.";
     rdfs:domain base:HotelRoom;
     rdfs:range rlt:IndoorLocation.

base:roomViewType  a rdf:Property;
     rdfs:comment "Room view information.";
     rdfs:domain base:HotelRoom;
     rdfs:range rvt:RoomViewType.
```

Atrybuty klasy *RoomType* pozwalają opisać typ pokoju, który może obowiązywać dla wielu indywidualnych pokoi.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix btc: <HotelInfrastructureCodes/BedTypeCode#>.
@prefix rat: <HotelInfrastructureCodes/RoomAmenityType#>.
@prefix base: <HotelInfrastructure/RoomType#>.

base:RoomType a rdfs:Class;
     rdfs:comment "Type of a hotel room.
                  Bed standards and facilities provided in this standard.".

base:typeName a rdf:Property;
     rdfs:comment "Room type name.";
     rdfs:domain base:RoomType;
     rdfs:range xsd:string.

base:bedType   a rdf:Property;
     rdfs:comment "Bed type for the room type.";
     rdfs:domain base:RoomType;
     rdfs:range btc:BedTypeCode.

base:roomAmenity a rdf:Property;
     rdfs:comment "Amenities in rooms of particular type.";
     rdfs:domain base:RoomType;
```

```
     rdfs:range rat:RoomAmenityType.

base:price a rdf:Property;
     rdfs:comment "Price for a room of a particular type.";
     rdfs:domain base:RoomType;
     rdfs:range xsd:float.
```

A.2. Moduł kodów dla infrastruktury

Większość klas w tym module to odpowiedniki elementów wyróżnionych w specyfikacji OTA dających możliwość opisu miejsca zakwaterowania.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix base: <HotelInfrastructureCodes#>.

base:BedTypeCode a rdfs:Class;
     rdfs:comment "Bed standard. OTA - BED".

base:BrandCategoryCodea rdfs:Class;
     rdfs:comment "Brand standard. OTA - BCC".

base:BusinessSrvcType a rdfs:Class;
     rdfs:comment "Business service types. OTA - BUS".

base:CountCategory  a rdfs:Class;
     rdfs:comment "Information about the periodical changes of hotel rooms
               availability due to their inoccupancy. OTA - CNT".
base:GuestRoomInfo a rdfs:Class;
     rdfs:comment "Information about the total amount of an object in the
               hotel. OTA - GRI".
base:MeetingRoomType  a rdfs:Class;
     rdfs:comment "Meeting room description.".

base:meetingRoomFormat  a rdf:Property;
     rdfs:domain base:MeetingRoomType;
     rdfs:range base:MeetingRoomFormat.

base:meetingRoomFeat a rdf:Property;
     rdfs:domain base:MeetingRoomType;
     rdfs:range base:MeetingRoomCode.

base:MeetingRoomFormat a rdfs:Class;
     rdfs:comment "Format of a meeting room. OTA - MRF".

base:MeetingRoomCode a rdfs:Class;
     rdfs:comment "Features of a meeting room. OTA - MRC".

base:PropertyClassType        a rdfs:Class;
     rdfs:comment "Types of the site (what it phisically is
                  or what it consist of), eg. chalet, inn, castle, all suite.
                  Describes the type of a building the customer would live in.
                  OTA - PCT".

base:ReservationCode a rdfs:Class;
     rdfs:comment "Reservation options. OTA - RST".

base:RevenueCategoryCode  a rdfs:Class;
     rdfs:comment "The revenue policy. OTA - RCC".

base:RoomAmenityType a rdfs:Class;
     rdfs:comment "Facilities and features of a room. OTA - RMA".

base:RoomViewType  a rdfs:Class;
     rdfs:comment "Room window view types. OTA - RVT".

base:SegmentCategoryCode  a rdfs:Class;
     rdfs:comment "Hotel/brand segment category. Brand visibility/apearance.
               OTA - SEG".
```

A.3. Moduł rekreacji

W tym module główną rolę pełni klasa *RecreationType*. Umożliwia opisanie konkretnego usługi rekreacyjnej oraz szczegółów z nią związanych. Zbiór usług oraz szczegółów ich charakterystyki pochodzi ze specyfikacji OTA.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix base: <Recreation/RecreationType#>.

base:RecreationType  a rdfs:Class;
     rdfs:comment "Recreation facilities.
                  The service with its detailed information and features.".

base:recreationService a rdf:Property;
     rdfs:domain base:RecreationType;
     rdfs:range base:RecreationSrvcType.

base:recreationServDetail  a rdf:Property;
     rdfs:domain base:RecreationType;
     rdfs:range base:RecreationSrvcDetailCode.

base:RecreationSrvcType  a rdfs:Class;
     rdfs:comment "Recreation service types. OTA - RST".

base:RecreationSrvcDetailCode  a rdfs:Class;
     rdfs:comment "Details of the recreation service. OTA - REC".
```

A.4. Moduł kontaktów

Klasa *Contacts* odnosi się zarówno do samego adresu, jak i technologii w jakiej połączenie z danym adresem należy nawiązać. Pozwala również opisać przeznaczenie danego kontaktu, jego umiejscowienie oraz, jeśli przedstawia namiary na usługę, jej charakter.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix base: <Contacts/Contacts#>.

base:Contacts a rdfs:Class;
     rdfs:comment "Contact information. Nbr, technology, designeted use,
                  type, location. It can refer also to emails, im ids, etc.".

base:location a rdf:Property;
     rdfs:comment "Reflects the location/purpose of the particular contact
                  (home, office, etc.)";
     rdfs:domain base:Contacts;
     rdfs:range base:LocationType.

base:technology a rdf:Property;
     rdfs:comment "Describes the type of the contact,
                  whether it is a telephone, fax, email or SIP address.";
     rdfs:domain base:Contacts;
     rdfs:range base:ContactTechnology.

base:use a rdf:Property;
     rdfs:comment "The purpose of the contact, what can be achieved when
               contacting with a particular contact";
     rdfs:domain base:Contacts;
```

```
     rdfs:range base:UseType.

base:serviceType a rdf:Property;
     rdfs:comment "Hotel service type which can be reached with this contact.";
     rdfs:domain base:Contacts;
     rdfs:range base:ContactSrvcCode.

base:address a rdf:Property;
     rdfs:domain base:Contacts;
     rdfs:range xsd:string;
     rdfs:comment "Can be a phone/fax number, email, sip, jabber id,
               multicast...".

base:LocationType a rdfs:Class
     rdfs:comment "OTA - PLT".

base:UseType a rdfs:Class;
     rdfs:comment "OTA - PUT".

base:ContactSrvcCode a rdfs:Class;
     rdfs:comment "Services that can be contacted. OTA - CSC".

base:ContactTechnology a rdfs:Class;
     rdfs:comment "The medium through which a particular contact can be
               reached. Implies the format of the address property. OTA - PTT".
```

A.5. Moduł lokalizacji

OutdoorLocation to podstawowa klasa, z której mogą dziedziczyć wszystkie klasy opisujące obiekt przestrzenny i odniesienie do niego na mapie.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>.
@prefix adrec: <Location/AddressRecord#>.
@prefix base: <Location/OutdoorLocation#>.

base:OutdoorLocation a rdfs:Class;
     rdfs:subClassOf geo:SpatialThing;
     rdfs:comment "Outdoor location.
                  Geographical and urban references.".

base:address a rdf:Property;
     rdfs:comment "Address details.";
     rdfs:domain base:OutdoorLocation;
     rdfs:range adrec:AddressRecord.

base:attractionCategory a rdf:Property;
     rdfs:comment "Nearby attractions.";
     rdfs:domain base:OutdoorLocation;
     rdfs:range base:AttractionCategoryCode.

base:indexPoint a rdf:Property;
     rdfs:comment "Reference map point.";
     rdfs:domain base:OutdoorLocation;
     rdfs:range base:IndexPointCode.

base:indexPointDist a rdf:Property;
     rdfs:comment "Distance from the reference map point.";
     rdfs:domain base:OutdoorLocation;
     rdfs:range base:IndexPointCode.

base:locationCategory a rdf:Property;
     rdfs:comment "Location category.";
     rdfs:domain base:OutdoorLocation;
     rdfs:range base:LocationCategoryCode.

base:neighbourhood a rdf:Property;
```

```
     rdfs:label "Neighbourhood";
     rdfs:comment "The neighborhood the site is located in.";
     rdfs:range xsd:string;
     rdfs:domain base:OutdoorLocation.

base:AttractionCategoryCode a rdfs:Class;
     rdfs:comment "Possible nearby attractions. OTA - ACC".

base:IndexPointCode a rdfs:Class;
     rdfs:comment "Possible reference map points. OTA - IPC".

base:LocationCategoryCode a rdfs:Class;
     rdfs:comment "Possible location categories. OTA - LOC".
```

A.6. Moduł cechy miejsca

Klasy tego modułu w większości odnoszą się do elementów specyfikacji OTA. Umożliwiają opisanie cech charakterystycznych dla zasobu reprezentującego dowolną lokalizację.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix base: <SiteFeatures #>.

base:AdditionalDetail a rdfs:Class;
     rdfs:comment "Details about certain policy. The message itself is
   delivered in natural language.
   OTA - ADT".
base:ArchitecturalStyleCode a rdfs:Class;
     rdfs:comment "Architectural style information. OTA - ARC".

base:Information  a rdfs:Class;
     rdfs:comment "Info attached.".

base:info a rdf:Property;
     rdfs:comment "Info text.";
     rdfs:domain base:Information;
     rdfs:range xsd:string.

base:type a rdf:Property;
     rdfs:comment "Info type.";
     rdfs:domain base:Information;
     rdfs:range base:InformationType.

base:InformationType  a rdfs:Class;
     rdfs:comment "Type of info attached. OTA - INF".

base:PetsPolicyCode a rdfs:Class;
     rdfs:comment "What about them. Pets allowances restrictions. OTA - PET".

base:PhysicallyChallengedFeatureCode a rdfs:Class;
     rdfs:comment "Facilities for disabled. OTA - PHY".

base:Picture  a rdfs:Class;
     rdfs:comment "The picture".

base:reference a rdf:Property;
     rdfs:comment "Picture location.";
     rdfs:domain base:Picture;
     rdfs:range xsd:anyURI.

base:description a rdf:Property;
     rdfs:comment "Picture descritption.";
     rdfs:domain base:Picture;
     rdfs:range xsd:string.
```

```
base:view a rdf:Property;
     rdfs:comment "Picture view description.";
     rdfs:domain base:Picture;
     rdfs:range base:PictureCategoryCode.

base:PictureCategoryCode  a rdfs:Class;
     rdfs:comment "What the picture shows. OTA - PIC".

base:SecurityFeatureCode  a rdfs:Class;
     rdfs:comment "Various safety measures. OTA - SEC".

base:SiteAmenityDetail a rdfs:Class;
     rdfs:comment "Facilities that can be found on-site and their location.".

base:amenity a rdf:Property;
     rdfs:domain base:SiteAmenityDetail;
     rdfs:range base:SiteAmenityCode.

base:location  a rdf:Property;
     rdfs:domain base:SiteAmenityDetail;
     rdfs:range rlt:IndoorLocation.

base:SiteAmenityCode  a rdfs:Class;
     rdfs:comment "Facilities that can be found on-site. OTA - HAC".

base:SiteStatusCode  a rdfs:Class;
     rdfs:comment "Site's status. OTA - HST".

base:WebContent a rdfs:Class;
     rdfs:comment "Videos, audios, etc.".

base:reference a rdf:Property;
     rdfs:comment "Media content location.";
     rdfs:domain base:WebContent;
     rdfs:range xsd:anyURI.

base:code a rdf:Property;
     rdfs:comment "Media content type.";
     rdfs:domain base:WebContent;
     rdfs:range base:WebContentCode.

base:WebContentCode a rdfs:Class;
     rdfs:comment "Videos, audios, etc. OTA - CTT".
```

A.7. Moduł wizyty

Klasa *VisitRecord* to opis wizyty konkretnej osoby w danym hotelu. Może być wykorzystana zarówno przez administratora konkretnego miejsca zakwaterowania jak i na potrzeby usługi rezerwacji miejsc w różnych miejscach zakwaterowania.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix hot: <HotelInfrastructure/Hotel#>.
@prefix rmt: <HotelInfrastructure/RoomType#>.
@prefix scd: <Schedule/Schedule#>.
@prefix trv: <Traveler/NameRecord#>.
@prefix tpt: <Traveler/TravelPurposeType#>.
@prefix gut: <Traveler/GuestType#>.
@prefix rmc: <HotelVisitCodes/ReservationMethodCode#>.
@prefix vin: <HotelVisit/VisitInvoice#>.
@prefix phc: <Contacts/Contacts#>.
@prefix base: <HotelVisit/VisitRecord#>.

base:VisitRecord a rdfs:Class;
```

```
     rdfs:comment "Record of particular hotel visit with payment,
   visit length and guest data.".

base:visitor   a rdf:Property;
     rdfs:comment "Personal information.";
     rdfs:domain base:VisitRecord;
     rdfs:range trv:NameRecord.

base:guestType a rdf:Property;
     rdfs:comment "Guest category.";
     rdfs:domain base:VisitRecord;
     rdfs:range gut:GuestType.

base:stayPeriod a rdf:Property;
     rdfs:comment "How long the person stays.";
     rdfs:domain base:VisitRecord;
     rdfs:range scd:Schedule.

base:at a rdf:Property;
     rdfs:comment "The accomodation.";
     rdfs:domain base:VisitRecord;
     rdfs:range hot:Hotel.

base:roomType   a rdf:Property;
     rdfs:comment "The room type the person stays at.";
     rdfs:domain base:VisitRecord;
     rdfs:range rmt:RoomType.

base:rm a rdf:Property;
     rdfs:label "Reservation Method";
     rdfs:domain base:VisitRecord;
     rdfs:range rmc:ReservationMethodCode.

base:invoice    a rdf:Property;
     rdfs:label "Invoice to pay";
     rdfs:domain base:VisitRecord;
     rdfs:range vin:VisitInvoice.

base:contactInfo    a rdf:Property;
     rdfs:label "Contact information";
     rdfs:domain base:VisitRecord;
     rdfs:range phc:Contacts.
```

Składnikiem opisu takiej wizyty jest faktura, której opis umożliwia klasa *VisitInvoice*.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix cur: <http://protege.stanford.edu/CurrencyOntology#>.
@prefix ftt: <Payment/FareTax#>.
@prefix hic: <Payment/Discounts#>.
@prefix hvc: <HotelVisitCodes/EventChage#>.
@prefix rcd: <HotelVisitCodes/RateCodes#>.
@prefix base: <HotelVisit/VisitInvoice#>.

base:VisitInvoice a rdfs:Class;
     rdfs:comment "Payment details.".

base:code a rdf:Property;
     rdfs:comment "Id of the invoice.";
     rdfs:domain base:VisitInvoice;
     rdfs:range xsd:integer.

base:currency a rdf:Property;
     rdfs:comment "Currency of the invoice.";
     rdfs:domain base:VisitInvoice;
     rdfs:range cur:Currency.
```

```
base:discount a rdf:Property;
     rdfs:comment "Discount applied. The rate applied.";
     rdfs:domain base:VisitInvoice;
     rdfs:range hic:Discounts.

base:tax  a rdf:Property;
     rdfs:comment "Tax information.";
     rdfs:domain base:VisitInvoice;
     rdfs:range ftt:FareTax.

base:ec a rdf:Property;
     rdfs:comment "Event Charge";
     rdfs:domain base:VisitInvoice;
     rdfs:range hvc:EventCharge.

base:beforeTax a rdf:Property;
     rdfs:comment "Amount before tax";
     rdfs:domain base:VisitInvoice;
     rdfs:range xsd:float.

base:afterTax a rdf:Property;
     rdfs:comment "Amount after tax";
     rdfs:domain base:VisitInvoice;
     rdfs:range xsd:float.

base:rateMode a rdf:Property;
     rdfs:comment "Rate mode. How the current price fits the average policy.";
     rdfs:domain base:VisitInvoice;
     rdfs:range rcd:RateMode.
```

A.8. Moduł finansów

Jego klasy są wykorzystywane do opisu polityki finansowej usług, cen oraz faktur. Klasa *MeanOfPayment* zawiera środki płatności wymienione w specyfikacji OTA oraz te, które zostały uwzględnione w projekcie *ChefMoz*. Klasa *Discounts* umożliwiaja opis zniżek według obu specyfikacji: IATA i OTA. Wprowadzone w tym celu relacje między konceptami zostały szczegółowo opisane w rozdziale 5. Podobnie zbudowana jest klasa *FareTax*, która należy do modułu finansów.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix cur: <http://protege.stanford.edu/CurrencyOntology#>.
@prefix base: <Payment#>.

base:MeanOfPayment a rdfs:Class;
     rdfs:comment "Mean of payment option. OTA - PMT".

base:FareTax a rdfs:Class;
     rdfs:comment "Taxes information".

base:currency a rdf:Property;
     rdfs:domain base:FareTax;
     rdfs:range cur:Currency;
     rdfs:comment "The currency information.".

base:percent a rdf:Property;
     rdfs:domain base:FareTax;
     rdfs:range xsd:float;
     rdfs:comment "The tax rate.".
```

```
base:amount a rdf:Property;
     rdfs:domain base:FareTax;
     rdfs:range xsd:float;
     rdfs:comment "The tax fare.".

base:info a rdf:Property;
     rdfs:domain base:FareTax;
     rdfs:range xsd:string.

base:code a rdf:Property;
     rdfs:domain base:FareTax;
     rdfs:range base:TaxCodes.

base:TaxCodes  a rdfs:Class;
     rdfs:comment "Base class for all tax codes.".

base:OTATaxCodes  a rdfs:Class;
     rdfs:subClassOf base:TaxCodes;
     rdfs:comment "Tax types specified by OTA. OTA - FTT".

base:IATATaxCodes a rdfs:Class;
     rdfs:subClassOf base:TaxCodes;
     rdfs:comment "Tax types specified by IATA".

base:Discounts  a rdfs:Class;
     rdfs:comment "Discount details.".

base:reductionPercent a rdf:Property;
     rdfs:domain base:Discounts;
     rdfs:range xsd:float;
     rdfs:label "Indicates how many percent less the reduced price is".

base:reductionAmount a rdf:Property;
     rdfs:domain base:Discounts;
     rdfs:range xsd:float;
     rdfs:label "Indicates static price reduction".

base:code a rdf:Property;
     rdfs:domain base:Discounts;
     rdfs:range base:DiscountTypes;
     rdfs:label "The official code of the discount".

base:name a rdf:Property;
     rdfs:domain base:Discounts;
     rdfs:range xsd:string;
     rdfs:label "The name od the discount".

base:conditions a rdf:Property;
     rdfs:domain base:Discounts;
     rdfs:range xsd:string;
     rdfs:label "Conditions that must be met in order to receive the discount".

base:DiscountTypes a rdfs:Class;
     rdfs:comment "The super class of all discount codes".

base:OTADiscountTypes  a rdfs:Class;
     rdfs:subClassOf base:DiscountTypes;
     rdfs:comment "Discount types specified by OTA. OTA - DIS".

base:IATADiscountTypesa rdfs:Class;
     rdfs:subClassOf base:DiscountTypes;
     rdfs:comment "Discount types which can be applied to air travel.".

base:CommonDiscountTypes a rdfs:Class;
     rdfs:subClassOf base:OTADiscountTypes, base:IATADiscountTypes;
     rdfs:comment "Class of all common discount codes for IATA
               and OTA discount codes specification".
```

A.9. Moduł restauracji

Podstawową klasą w tym module jest *Restaurant*, cała ontologia restauracji została opisana wyczerpująco w [Gawinecki, 2005]. W ramach ontologii miejsca zakwaterowania została wykorzystana cała ontologia restauracji i może być ona traktowana jako jej moduł.

Załącznik B. Pominięte elementy specyfikacji OTA.

Kody specyfikacji OTA z 2004 roku nie uwzględnione w ontologii.

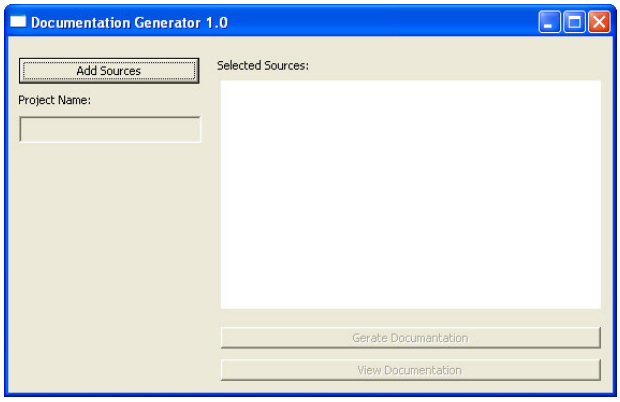
Pełna nazwa	Nazwa kodowa
Age Qualifying Code	AQC
Communication Location Type	CLT
Contact Location	CON
Decline Reason	DEC
Distribution Type	DTB
Inventory Block Type	IBT
Inventory Block Status	IBS
Inventory Count Type	INV
Profile Type	PRT
Rate Plan Type	RPT
Reference Point Category Code	REF
Restaurant Category Code	RES
Restaurant Srvc Info	RSI
Size	SIZ
Status	STS
Transportation Code	TRP

Załącznik C. Opis generatora dokumentacji.

C.1. Instrukcja obsługi

Stworzona aplikacja generatora dokumentacji jest przeznaczona dla systemów operacyjnych z rodziny Microsoft Windows. Do jej uruchomienia wymagana jest biblioteka o nazwie: `swt-win32-3139.dll`, która znajduje się na płycie dołączonej do niniejszej pracy magisterskiej. Bibliotekę tę należy umieścić w katalogu znajdującym się w zmiennej środowiskowej `path` systemu operacyjnego. Aplikację należy uruchamiać poleceniem: `runDocGen.bat`.

Po poprawnym uruchomieniu powinien zostać wyświetlony ekran przedstawiony na Rysunku C.1. Jest to interfejs graficzny generatora dokumentacji.



Rysunek C.1. Interfejs graficzny generatora dokumentacji.

W pierwszej kolejności należy wybrać źródła ontologii za pomocą guzika *Add Sources*, powinny być to pliki w formacie RDF(S) o rozszerzeniach *rdf*. Następnie należy wpisać nazwę projektu w pole tekstowe *Project Name*. Jeśli został wybrany co najmniej jeden plik źródłowy i została podana nazwa projektu guzik *Generate Documentation* stanie się dostępny. Po jego kliknięciu pojawi się zapytanie o ścieżkę, w której ma zostać stworzona dana dokumentacja, po jej wybraniu rozpocznie się proces tworzenia dokumentacji. Proces ten sygnalizowany jest ekranem informacyjnym, a po jego zakończeniu ekran informacyjny zostaje zamknięty i guzik o nazwie *View Documentation*, znajdujący się w głównym oknie interfejsu graficznego generatora, zostaje

uaktywniony. Po jego kliknięciu zostaje otwarta domyślna przeglądarka internetowa systemu, a w niej dokumentacja ontologii.

Po poprawnym zakończeniu procesu tworzenia dokumentacji w podkatalogu *docs* wybranego katalogu docelowego zostaje umieszczony plik *index.html*, który jest stroną startową dokumentacji. W podkatalogu *protage_sources* powinny z kolei znajdować się pliki *rdfs* i *rdf*, z których może zostać łatwo stworzony nowy projekt dla aplikacji Protégé.

C.2 Ogólna zasada działania

Po naciśnięciu guzika *Generate Documentation* aplikacja tworzy podkatalogi projektu.

Następnie wszystkie pliki źródłowe wybrane przez użytkownika zostają wczytane za pomocą komponentów środowiska Jena. Każdemu plikowi *rdf* odpowiada dokładnie jeden obiekt klasy *Model*. W trakcie czytania plików źródłowych zostają zapamiętane wszystkie zasoby występujące w badanych plikach wraz z nazwą przestrzeni nazw, w której zostały zdefiniowane.

Iteracyjnie, parami, na wszystkich obiektach klasy *Model,* zostaje przeprowadzona operacja sumy zbiorów zdefiniowanych w przestrzeniach nazw wyrażeń RDF. Warunkiem końcowym iteracji jest uzyskanie jednego obiektu klasy *Model*. Obiekt ten zostaje zapisany do pliku w formacie RDF w podkatalogu *protege_sources*.

W następnej kolejności, na podstawie stworzonego pliku będącego sumą wyrażeń ze wszystkich przestrzeni nazw zostaje stworzona nowa *baza wiedzy* Protégé. *Baza wiedzy* jest strukturą opartą na grafie, w której przechowywane są wyrażenia z podanego pliku źródłowego. Następnie zostaje stworzona dokumentacja HTML hierarchii klas oraz indywidualna dokumentacja dla każdej z klas i każdego z atrybutów znajdujących się w *bazie wiedzy* powstałej z pliku w formacie RDF.

Na końcu zostają stworzone: strona indeksu, strony wszystkich przestrzeni nazw oraz zasobów istniejących w danej przestrzeni.

C.3. Modyfikacje kodu źródłowego aplikacji Protégé

Usunięcie dynamicznego ładowania pliku konfiguracyjnego.

Polega na przeprowadzeniu modyfikacji konstruktora klasy *HTMLExport* znajdującej się w pakiecie: `edu.stanford.smi.protegex.export.html`. Deklarację ścieżki dostępu do pliku konfiguracyjnego *hmtlexport.properties* należy zmienić z:

```
String path = PluginUtilities.getPluginsDirectory().getPath() +
                                                File.separator +
              "edu.stanford.smi.protegex.standard_extensions" +
                                                File.separator +
                                                 "html_export" +
                                                File.separator +
                                      "htmlexport.properties";
```

Na:

```
String path = config.getApplicationDir() + File.separator
+"htmlexport.properties";
```

Dodanie możliwości wydruku instancji konkretnej klasy, które są dziedziczone nie wprost po danej klasie.

W tej samej klasie *HTMLExport* należy dodać następujące pola statyczne:

```
private final String IND_INSTANCE_LOWERCASE = "instance.indirect.lowercase";
private final String IND_INSTANCE_LOWERCASE_PLURAL =
"instance.indirect.lowercase.plural";
private final String IND_INSTANCE_UPPERCASE = "instance.indirect.uppercase";
private final String IND_INSTANCE_UPPERCASE_PLURAL =
"instance.indirect.uppercase.plural";
private final String DIR_INSTANCE_LOWERCASE = "instance.direct.lowercase";
private final String DIR_INSTANCE_LOWERCASE_PLURAL =
"instance.direct.lowercase.plural";
private final String DIR_INSTANCE_UPPERCASE = "instance.direct.uppercase";
private final String DIR_INSTANCE_UPPERCASE_PLURAL =
"instance.direct.uppercase.plural";
```

Dzięki temu w pliku konfiguracyjnym *htmlexport.properties* możemy definiować słowa wykorzystane w opisie instancji klas. Przykład takiej definicji to:

```
instance.indirect.lowercase=indirect instance
instance.indirect.lowercase.plural=indirect instances
instance.indirect.uppercase=Indirect Instance
instance.indirect.uppercase.plural=Indirect Instances
instance.direct.lowercase=direct instance
instance.direct.lowercase.plural=direct instances
instance.direct.uppercase=Direct Instance
instance.direct.uppercase.plural=Direct Instances
```

Następnie w metodzie *generateClassPage*, na końcu istrukcji wykonywanych w wypadku spełnienia warunku:

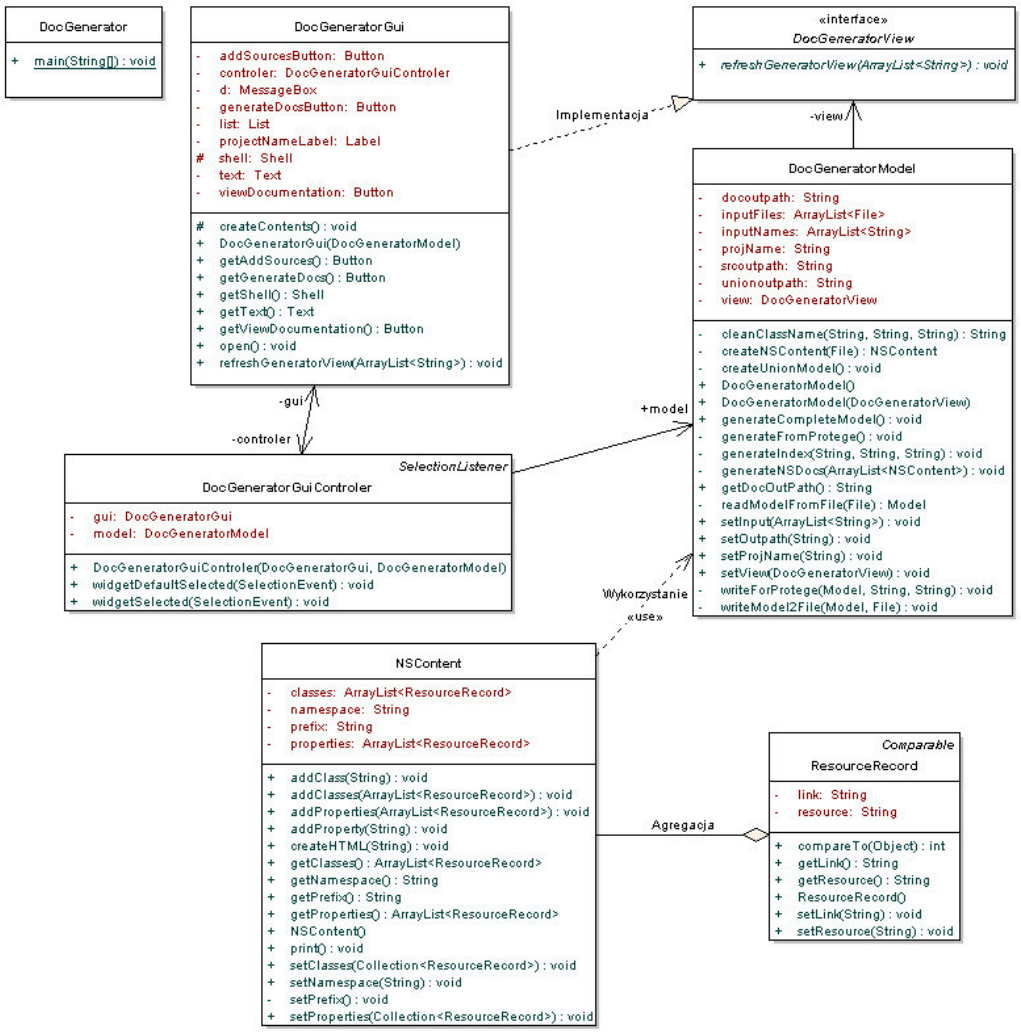
Należy dodać następującą instrukcję warunkową:

```
if (!config.getUseNumbering()) {
```

```
                    pw.println("<span class=\"sectionTitle\">" +
properties.getProperty(IND_INSTANCE_UPPERCASE_PLURAL) + "</span>");
                    printFrameIconList(pw, indirectInstances);
                } else {
                    pw.println("<dl>");
                    pw.println("<dt class=\"sectionTitle\">" +
properties.getProperty(IND_INSTANCE_UPPERCASE_PLURAL) + "</dt>");
                    pw.println("<dd>");
                    printNumberedInstanceList(pw, indirectInstances);
                    pw.println("</dd>");
                    pw.println("</dl><br>");
                }
```

C.4. Implementacja aplikacji generatora dokumentacji

Projekt aplikacji generatora dokumentacji opiera się o wzorzec projektowy Model View Control (MVC). Wzorzec projektowy (*ang. design pattern*) jest to powtarzalny element architektury oprogramowania, dzięki jego zastosowaniu można określić strukturę i zachowanie komponentów lub zestawów klas systemu. W szczególności, wzorzec MVC to rozwiązanie typowego problemu współpracy klas lub komponentów warstwy logiki biznesowej (*ang. model*) aplikacji z klasami lub komponentami odpowiedzialnymi za prezentację (*ang. view*) i kontrolę (*ang. control*) warstwy logiki biznesowej z poziomu graficznego interfejsu użytkownika (*ang. graphical user inteface – GUI*). Diagram klas generatora dokumentacji (Rysunek C.2) pokazuje między innymi, które klasy pełnią role modelu, prezentacji i kontrolera. Opis wszystkich klas zaimplementowanych w ramach tej aplikacji został umieszczony pod rysunkiem.



Rysunek C.2. Diagram klas aplikacji generatora dokumentacji.

Klasa *DocGenerator* służy wyłącznie do inicjalizowania aplikacji. Tworzy obiekty klas *DocGeneratorModel* i *DocGeneratorGui*.

DocGeneratorModel jest klasą stanowiącą rdzeń aplikacji. Jest odpowiedzialna za przeprowadzenie sumy wyrażeń zdefiniowanych w wejściowych plikach formatu RDF, stworzenie *bazy wiedzy* Protégé, jej zapis do pliku i ostatecznie za wygenerowanie dokumentacji ontologii. Przekazuje informacje o wybranych plikach źródłowych RDF do klasy odpowiedzialnej za prezentacje tych informacji. Udostępnia publiczne metody pozwalające na: (1) dodawanie plików źródłowych, (2) dodawanie obiektu prezentującego stan pól obiektu typu *DocGeneratorModel*, (3) definiowanie ścieżki wyjściowej do zapisu stworzonej dokumentacji, (4) podawanie nazwy projektu ontologii złożonego z

podanego zbioru plików wejściowych. Jest to implementacja roli *modelu* będącej elementem wzorca MVC.

DocGeneratorGui jest klasą odpowiedzialną za wyświetlenie interfejsu użytkownika oraz prezentację stanu obiektu klasy *DocGeneratorModel*. Klasa ta implementuje interfejs *DocGeneratorView*. *DocGeneratorModel* uruchamia metody tego interfejsu po zmianie stanu swoich pól, dzięki temu *DocGeneratorGui* prezentuje zmiany stanu *modelu*. Jest to implementacja roli *prezentacji* będącej elementem wzorca MVC. Wyświetlane przez obiekty tej klasy elementy interfejsu użytkownika zostały pokazane na Rysunku C.1.

Klasa *DocGeneratorGuiControler* obsługuje zdarzenia generowane przez obiekt klasy *DocGeneratorGui* powstałe w wyniku konkretnych akcji użytkownika. Po wybraniu odpowiednich przycisków interfejsu użytkownika, wyświetla okna dialogowe wyboru i zapisu plików. Wprowadzane nazwy i wybierane pliki przekazuje do obiektu klasy *DocGeneratorModel*, który obsługuje. Jest to implementacja roli *kontroli* będącej elementem wzorca MVC.

W obiektach klasy *ResourceRecord* przechowywana jest nazwa zasobu definiowanego w źródłowych plikach RDF oraz hiperłącze do szczegółowej dokumentacji dotyczącej danego zasobu.

NSContent to model wszystkich definicji zasobów pochodzących z konkretnego pliku źródłowego. Nazwy zasobów i hiperłącza do stron HTML zawierających ich dokładną charakterystykę są przechowywane w obiektach klasy *ResourceRecord* agregowanych w obiektach typu *NSContent*. Klasa ta posiada też metodę pozwalającą wygenerować pliki HTML będące spisami wszystkich zasobów dla pojedynczego pliku posegregowanymi ze względu na rolę pełnioną w ontologii: klasy lub atrybuty.