**Uniwersytet Adama Mickiewicza w Poznaniu**
Wydział Matematyki i Informatyki

*Maciej Gawinecki*

# Modelowanie użytkownika na podstawie interakcji z systemem opartym o technologie WWW

*Praca magisterska napisana pod kierunkiem*
*prof. dra hab. Zygmunta Vetulaniego*


Poznań, 2005

**Streszczenie**. Praca na temat wykorzystania metod modelowania użytkownia w kontekście Semantycznej Sieci WWW i przy wsparciu inteligentnych agentów programowych. Obejmuje włączenie całego procesu modelowania użytkownika do przykładowego systemie wspomagania podróży. Poruszany jest wpływ interfejsu WWW na rodzaj możliwych do obserwowania zachowań użytkownika, semantyczny sposób reprezentacji modelu użytkownika i sposób jego uczenia oparty na pozytywnej informacji zwrotnej. Analizowana jest kwestia inicjalizacji takiego modelu za pomocą stereotypowania i problem stworzenia wiarygodnych stereotypów. Omówione jest także rozwiązanie problemu współistnienia dwóch jednostek proaktywnych w systemie: użytkownika i jego agenta osobistego.

**Słowa kluczowe**: modelowanie użytkownika, system wieloagentowy, Semantyczna Sieć WWW, środowisko współdzielonej inicjatywy, stereotypowanie, system rekomendujący, system wspomagania podróży.

## *Podziękowania*

*Pragnę podziękować prof. dr hab. Zygmuntowi Vetulaniemu za krytyczne uwagi: merytoryczne i redaktorskie. Dziękuję również prof. Marcinowi Paprzyckiemu za zarażenie mnie entuzjazmem do tematyki agentów, ontologii i personalizacji, za nie gasnący idealizm i konstruktywną krytykę. Pani mgr Joannie Ochniak (z Wyższej Szkoły Hotelarstwa i Gastronomii w Poznaniu), dr hab. Andrzejowi Masłowskiemu (z Instytutu Rynku Wewnętrznego i Konsumpcji w Warszawie) oraz dr hab. Michałowi Chmarze (z Instytutu Socjologii UAM w Poznaniu) – za pomoc w przygotowywaniu badań dotyczących rynku gastronomii w Polsce. Dr Waldemarowi Wołyńskiemu (z Zakładu Rachunku Prawdopodobieństwa i Statystyki Matematycznej mojego wydziału) za pomoc przy rozwijaniu technik analizy statystycznej. Za pomoc przy roztrząsaniu spraw technicznych – zespołowi pracującemu* systemem wspomagania podróży*, 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: Mateuszowi Kruszykowi i Michałowi Szymczakowi za budujące dyskusje i pomysły przy projektowaniu systemu. Dziękuje również Pawłowi Kaczmarkowi, współautorowi dodatku (A) dotyczącego problemu identyfikacji użytkownika w systemie wieloagentowym. Pracownikom Zakładu Lingwistyki Informatycznej i Sztucznej Inteligencji mojego wydziału – dzięki nim praca ta jest bardziej przystępna i zasadna. Wiktorowi Moderau za opracowanie grafiki do interfejsu użytkownika stworzonej aplikacji. Wreszcie całej rzeszy ludzi, których tu nie wymieniłem, a którzy codziennie odpowiadają na pytania ludzi takich jak ja – na listach dyskusyjnych i w ośrodkach akademickich na całym świecie.*

*Rodzinie i przyjaciołom – za wsparcie.*

*Maciej Gawinecki*

*Poznań, 6 grudnia 2005 r.*

# Spis treści

# Wstęp

*Zagraniczny turysta dociera do Poznania. Zaraz po przybyciu ma ochotę na zjedzenie w jakiejś przyzwoitej restauracji, a z braku znajomości języków obcych decyduje się na wykorzystanie Internetu do znalezienie adresu najbliższego lokalu. Wyszukiwarka zwraca listę restauracji polecających bigos, flaki i golonkę - a więc dania mięsne - nie pamiętając o tym, że rzeczony turysta to zagorzały wegetarianin.*

Kiedy na początku lat 50-tych dwudziestego wieku Calvin Mooers, pionier nauki o informacji, ukuł termin *wyszukiwanie informacji*[1], zwrócił uwagę również na problemy związane z tym pojęciem: (1) W jaki sposób reprezentować i organizować informację w sposób intelektualny[2]? (2) Jak określić intelektualne wyszukiwanie? i wreszcie (3) Jakich systemów i technik użyć dla tych celów? (Mooers 1951). Problemy te są wciąż żywe, o czym świadczy gwałtowny wzrost ilości przypadkowej informacji w Internecie z jednoczesnym ograniczeniem możliwości znalezienia właściwej. Tim Berners-Lee proponuje Semantyczną Sieć WWW[3], która koncentrując się wokół pojęcia ontologii, rozwiązuje problem inteligentnej reprezentacji informacji. Poszukiwana informacja musi być właściwa[4] nie w sensie świeżości, ale odnoszenia się do kontekstu: (a) zewnętrznego, podyktowanego aktualnym problem lub sytuacją użytkownika i (b) wewnętrznego, wynikającego z jego upodobań, stanu emocjonalnego i sposobu postrzegania świata (Saracevic *et al.* 1997). Gdyby w przedstawionej wyżej sytuacji system znał kontekst użytkownika, czyli miał na jego temat określone wyobrażenie, mógłby zaproponować restaurację serwującą kuchnię jarską.

Jak twierdzi Alfred Kobsa (Kobsa 1990), jedną z najlepiej zbadanych technik wspomagania wyszukiwania informacji jest *modelowanie użytkownika*. Jest to proces tworzenia wyobrażenia o użytkowniku, nazywanego modelem lub profilem, w oparciu o informacje, które użytkownik dostarcza na drodze interakcji z systemem. Kluczowym zadaniem jest określenie sposobu reprezentacji modelu. W systemach, w których informacja zorganizowana jest w postaci terminów z ontologii, naturalnym staje się stworzenie modelu użytkownika, który odnosi się do nich w sposób bezpośredni.

---

[1] Ang. *information retrieval* (IR).

[2] W literaturze dotyczącej teorii zarządzania wiedzą, a więc tej, którą współtworzył Mooers, określenie „intelektualny" odnosi się do intelektu ludzkiego i w omawianym kontekście oznacza coś, co naśladuje naturę intelektu. Zatem reprezentacja intelektualna ma stanowić projekcję reprezentacji wiedzy przez umysł człowieka, a intelektualne wyszukiwanie – naśladować jego działanie przy wyszukiwaniu informacji.

[3] Ang. *Semantic Web*.

[4] Ang. *relevant*.

Celem tej pracy jest dokładne określenie konstrukcji takiego modelu, zdefiniowanie całego procesu modelowania użytkownika oraz jego integracja z istniejącym *systemem wspomagania podróży*[5]. System ten stanowi aplikację idei internetowej agencji turystycznej na polu Semantycznej Sieci WWW i agentów programowych.

W rozdziale pierwszym definiuję pojęcie modelowania użytkownika i terminy pokrewne. Opisuję również znane techniki konstrukcji modelu, kładąc szczególny nacisk na podejścia oparte na interakcji opartej o strony WWW. Rozdział drugi opisuje technologie składające się na Semantyczną Sieć WWW, a których znajomość jest potrzebna do zrozumienia proponowanego rozwiązania. Rozdział trzeci podsumowuje stan dzisiejszy *systemu wspomagania podróży*. W rozdziale czwartym przedstawiam ideę proponowanego rozwiązania, analizuję jego zalety i problemy wynikające z integracją z poznanymi wcześniej technologiami. W końcu rozdział piąty opisuje szczegółowe rozwiązanie, zarówno od strony algorytmu modelowania użytkownika, jak i od strony implementacyjnej. W ostatnim, szóstym rozdziale, dzielę się wnioskami, problemami wynikłymi z założonego podejścia i propozycjami lepszych rozwiązań.

[5] Ang. *travel support system* jest międzynarodowym projektem akademickim, kierowanym przez prof. Marcina Paprzyckiego ze Szkoły Wyższej Psychologii Społecznej Warszawie. Nad projektem pracują zmieniające się zespoły studentów z Uniwersytetu Adama Mickiewicza w Poznaniu, Politechniki Warszawskiej oraz Oklahoma State University w USA. Więcej informacji o aktualnym stanie projektu czytelnik znajdzie na stronie: `http://mpaprzycki.swps.edu.pl/mp/cvr/research/agent.html`.

# Rozdział 1

# Modelowanie użytkownika

Początków techniki modelowania użytkownika należy doszukiwać się w pracach Cohena i Rich z końca lat 70-tych (Cohen i Perrault 1979; Rich 1979a; Rich 1979b). Od tego czasu powstało wiele systemów zdolnych do adaptowania się do preferencji użytkownika, a w końcu lat 90-tych dostrzeżono potencjał techniki modelowania użytkownika w handlu elektronicznym (Cohen i Perrault 1998). Każdy z systemów wykorzystujących tą technikę, nawet jeżeli realizuje ten sam scenariusz, zachowuje się inaczej. Zarysujemy najpierw kilka ogólnych sytuacji obrazujących jej działanie.

Na stronie przewodnika *Zagat Survey*[1] turysta, spragniony kulinarnych wrażeń, może znaleźć dla siebie restaurację, określając rodzaj preferowanej kuchni, wystrój restauracji, jej sąsiedztwo oraz położenie. W odpowiedzi dostaje listę najlepszych lokali, które spełniają jego wymagania. Restauracje wybierane są wyłącznie w oparciu o oceny konsumentów, wiadomo bowiem, że ich głosy – stałych bywalców, a nie recenzje gastronomicznych krytyków – są najbardziej obiektywne i wiarygodne[2]. Bazą przewodnika jest ankieta on-line, w której restauracje podlegają ocenie punktowej w czterech kategoriach: jakości serwowanych dań, wystroju wnętrza, obsługi oraz wymaganej zasobności portfela. Podstawę punktacji stanowi średnia wyciągana z ocen respondentów. A jeśli ostatecznie turysta znajdzie w serwisie odpowiednią restaurację, to może być ona za droga. Wtedy, niestety, proces wyszukiwania musi zacząć od początku.

Alternatywę dla powyższego rozwiązania stanowi system *Entree* (Burke 2002), w którym turysta może skrytykować wysokość ceny serwowanych posiłków i w rezultacie otrzymać listę restauracji tańszych, ale o podobnej kuchni i atmosferze. Wystarczy, że kliknie odpowiedni przycisk (*„less $$"* na rysunku 1.1), a system, w oparciu o posiadaną bazę restauracji i znajomość podobieństw między różnymi kuchniami znajdzie lepsze rozwiązanie. System jest zatem lepszy od serwisu *Zagat Survey*: reaguje na krytykę i potrafi porównywać restauracje. Niestety nie potrafi się uczyć: *Entree* zapisuje krytykę i wybory użytkowników, ale w żaden sposób nie wykorzystuje tego doświadczenia. Bazuje wyłącznie na ograniczonej, ustalonej (arbitralnie przez autora) wiedzy.

Co nie zostało zrobione w pierwszej wersji, zostało poprawione przez autora w dru-

[1] *Przewodnik Zagat Survey – online*: `http://www.zagat.com/`.

[2] Paradowski, Michał. 2005. *Europe's Top Restaurants 2005*: `http://katalogi.gastrona.pl/art/article_3617.php`.


Rysunek 1.1: Ekran systemu Entree, pozwalający na wyszukiwanie restauracji przez krytykę (rysunek pochodzi z (Burke 2002)).

giej. *EntreeC* (Burke 2002) nie tylko pozwala na krytykę jednego z atrybutów lokalu, ale także porównuje, jakie zdanie mieli inni użytkownicy odnośnie proponowanych restauracji. I tak, kiedy turysta łączy się z systemem i zaczyna szukać jakiejś restauracji w Bangkoku, za punkt startowy (a więc punkt odniesienia) podaje ulubioną restaurację w rodzinnym mieście, „Duetto" w Poznaniu. Lokal serwuje kuchnie: włoską i wegetariańską. Odpowiednią, najwyżej rekomendowaną w Bangkoku jest „Don Giovanni", podająca jedzenie włoskie i owoce morza. Okazuje się jednak, że poszukujący jest wegetarianinem i krytykuje proponowaną przez system kuchnię wracając do propozycji dla jaroszy. Chwilę później, przed podobnym wyborem staje pewna turystka, zaczynając poszukiwania od tej samej restauracji „Duetto". Ponieważ rekomendacja, zaproponowana wspomnianemu wcześniej turyście, została źle oceniona, jego opinia zostanie wzięta pod uwagę i tym razem system od razu zaproponuje turystce lokal wegetariański.

Ostatecznie zwrócone rekomendacje bliżej odpowiadają potrzebom użytkownika. Ale system wciąż nie zna swojego użytkownika. Bazuje raczej na podobieństwie jego problemu do sytuacji obserwowanych w trakcie interakcji z innymi użytkownikami. Przypatrzmy się zatem jeszcze jednemu systemowi.

W projekcie *IRES*[3] (Montaner *et al.* 2002) użytkownik, przy pierwszy spotkaniu z systemem rejestruje się w systemie. Dzięki temu będzie mógł być później identyfikowalny i traktowany w sposób indywidualny. W trakcie wyszukiwania ma szansę krytyki proponowanych restauracji, dzięki czemu system wie, które rekomendacje odpowiadają jego gustom. Zapamiętuje więc indywidualne wybory i gdy pojawia się pytanie o no-

[3] *On The Integeration Of Restaurant Services*

wy lokal, ocenia go na podstawie podobnych doświadczeń użytkownika. W wypadku wątpliwości, tzn. gdy system nie potrafi na podstawie zebranego doświadczenia użytkownika jednoznacznie zdecydować o rekomendacji danej restuaracji, wtedy zdaje się na ocenę wynikającą z doświadczeń innych użytkowników.

Kryteria, którymi użytkownik kieruje się przy wyszukiwania restauracji mogą być mu znane (i wtedy może je podać jako zapytanie w formularzu), jak i nie uświadomione (i wtedy oczekuje od systemu, że ten „domyśli" się w czym rzecz). Może być też tak, że użytkownik w procesie wyszukiwania dla własnej wygody nie podaje pewnych warunków wyszukiwania. Zawsze jednak poszukuje informacji odpowiadającej jego potrzebom. Jak wspomnieliśmy we wstępie, w celu spełnienia tych potrzeb, należy uwzględnić kontekst zewnętrzny i wewnętrzny użytkownika. Jest oczywistym, że system może go znać wyłącznie w jednym z dwóch przypadków: (a) użytkownik sam go określi (podając np. listę swoich zainteresowań) lub (b) obserwując zachowanie użytkownika w interakcji z nim. W każdym z nich system otrzymuje *informację zwrotną*, na przykład w postaci krytyki wysokości cen serwowanych potraw (systemy *Entree*, *EntreeC*). W tym wypadku system reaguje na bieżąco na pojawiający się kontekst i ewentualnie zapamiętuje zachowania użytkowników. Ale, jak wspomniano, nie kojarzy odnotowanych zachowań z konkretnym użytkownikiem. Aby rozwiązać ten problem Elaine Rich zaproponowała w 1979 roku odróżnianie użytkowników i tworzenie indywidualnych, długoterminowych *modeli (profili) użytkowników* (Rich 1979b).

Model użytkownika to jego obraz, sposób, w jaki system go postrzega. Proces prowadzący do stworzenia tego obrazu nazywamy *modelowaniem użytkownika*. Rich przedstawia swój pomysł na przykładzie programu grającego rolę bibliotekarza o imieniu Grundy, który próbuje zaproponować potencjalnemu czytelnikowi książkę odpowiadającą jego gustom. Zadając serię pytań, próbuje wybadać, jakie są cechy użytkownika. Przykładowy, nieco skrócony, dialog może wyglądać następująco (Rich 1979b):

> **GRUNDY**: *Jakimi słowami określiłby Pan siebie?*
> **UŻYTKOWNIK**: *niekonwencjonalny otwarty bezpośredni szczery dowcipny wytrwały ryzykant*

W ciągu dalszej rozmowy Grundy dostrzega u użytkownika cechy typowe dla stereotypowego ekscentryka:

$$Ekscentryk = [potrzeba\_ryzyka : 0.9, niekonwencjonalnosc : 0.7, \ldots]$$

i dlatego – być może zbyt pochopnie – rekomenduje:

> **GRUNDY**: *Czy czytałeś już „Zen i sztuka utrzymania motocykla" Roberta Pirsiga?*

Profil może również stanowić zbiór doświadczeń użytkownika. We wspomnianym systemie *IRES* na profil każdego użytkownika składa się wektor widzianych restauracji i ich oceny[4]:

$$Profil = [(restauracja_1, ocena_1), (restauracja_2, ocena_2), \ldots]$$

[4]Ze względu na potrzebę przejrzystości przedstawiam uproszczoną wizję systemu.

Ocena jest funkcją nie tylko bezpośredniej krytyki użytkownika, ale i czasu poświęconego na oglądanie witryny danej restuaracji.

Ludzie, zazwyczaj, zachowują się w stosunku do innych adekwatnie do wyobrażenia, jaki na ich temat sobie zbudowali, traktując każdego indywidualnie. A w jaki sposób zbudowany model może być wykorzystany przez system rekomendujący restauracje? Przyjrzyjmy się rozwiązaniom w pokrewnych dziedzinach.

W e-commerce wykorzystuje się *technologię „push"*, dzięki której użytkownik otrzymuje z serwera świeże informacje, ale tylko te, które leżą w sferze zainteresowań użytkownika, czyli są *spersonalizowane*. Dostarczanie informacji nie wymaga od użytkownika ciągłego sprawdzania witryny z informacjami – nad aktualizacją danych czuwa serwer. Gwoli ścisłości, pomysł *selektywnego rozpowszechniania informacji*[5] nie jest nowy; Luhn, informatyk w IBM wypracował go już w 1961 (Luhn 1961). Zakres informacji, które użytkownik chciał otrzymywać, musiał określić sam.

Mniej więcej w tym samym czasie Jonathan Robbin, programista i wykładowca socjologii, opracował metodę, dzięki której zainteresowania, gusta i inne cechy klienta można wywnioskować wyłącznie z jednej danej – kodu pocztowego[6]. Dokładniej, każdemu kodowi pocztowemu odpowiada obszar, którego mieszkańców można scharakteryzywać za pomocą trzech czynników: miejsca zamieszkania, rasę i wysokości dochodu. Robbin, przez połączenie swoich badań z danymi konsumentów i ich adresami pocztowi mógł określić rynki zbytu dla różnych produktów i wybrać konsumentów do reklamy wysyłanej pocztą. Aby sprzedawać tak zebrane bazy danych klientom rządowym i biznesowym, założył firmę Claritas. Bazując na spisie ludności z jednego roku Claritas stworzyła 40 potencjalnych rankingów na rynkach kodów pocztowych. Do każdej z tych kategorii została przypisana grupa ludności. Grupy te zostały uporządkowane według rentowności, która decyduje o ich potencjale rynkowym. W efekcie powstał podział amerykańskiego rynku na skupiska wygranych i przegranych[6].

Trzy wspomniane czynniki mogą określać profil klienta – będącego w ekonomii portretem typowego beneficjenta danej usługi czy towaru. Podobnie przy filtrowaniu informacji można wykorzystać profil użytkownika do dostarczaniu mu informacji odpowiedniej. Pomysł ten wykorzystał Pazzani (Pazzani 1999), dzieląc grupy użytkowników o wspólnych preferencjach kulinarnych na skupiska. Dla każdego tworzony jest reprezentant o cechach (takich jak: płeć, wiek, kod pocztowy, stan zatrudnienia i wykształcenie) dominujących w danym skupisku. W momencie rejestracji użytkownik zostaje przydzielony do skupiska z reprezentantem najbardziej, pod względem podanych cech demograficznych, do niego zbliżonym. Zakłada się, że preferencje użytkownika odpowiadają wspólnym dla skupiska upodobaniom i takie też będą rekomendacje restauracji.

Według przedstawionego modelu cechy demograficzne użytkownika determinują jego preferencje w sposób stały i nie biorą pod uwagę wpływu czasu na ich zmiany.

[5] Ang. *selective dissemination of information*.

[6] Quinn, Lois M. i Pawasarat, John. 2001. *Confronting Anti-Urban Marketing Stereotypes: A Milwaukee Economic Development Challenge*: `http://www.uwm.edu/Dept/ETI/purchasing/markets.htm`.

Rozwiązanie znajdziemy w codziennym życiu. Użytkownik, poszukując dobrej restauracji, może poradzić się swoich znajomych, a więc ludzi do niego podobnych. Co więcej, zebrane opinie może wyważyć i wybrać, jego zdaniem, najlepszą. Idea ta przyświecała stworzeniu odpowiedniej techniki dla systemów rekomendujących. Znajomi danego użytkownika znajdywani są na podstawie podobieństwa profilu, czyli – na przykład – wspólnych doświadczeń. Im bliżej do danego użytkownika, tym jego opinia bardziej się liczy. Podobnie w systemie *IRES* zaufanie do opinii innych użytkowników (zbudowane na zasadzie weryfikacji ich rekomendacji) decyduje o podobieństwie między nimi.

Podsumowując, budujemy model użytkownika na bazie jego cech demograficznych, obserwując jego zachowanie w systemie i poznając opinie. Wykorzystujemy go, aby dostarczyć mu informacje spersonalizowaną. Polega to na filtrowanie informacji, poprzez analizę co najmniej dwóch z trzech następujących elementów: profilu własnego użytkownika, profili innych użytkowników i cech oferowanych produktów.

## 1.1 Terminologia

Na potrzeby dalszej dyskusji uporządkujemy potrzebne definicje, bazując na pracach (Chen i Magoulas 2005; Oard 1997).

### 1.1.1 Wyszukiwanie i filtrowanie informacji

*Wyszukiwanie informacji* pozwala użytkownikowi odnaleźć potrzebną mu informację poprzez zadawanie odpowiednich zapytań do systemu. Tak znaleziona informacja może być poddana *technikom filtrującym*, prowadzącym do poprawienia jej selekcji. Rezultat ten osiągany jest poprzez przyjmowanie ograniczeń, dyktowanych najczęściej przez kontekst sytuacyjny i/lub model użytkownika (Chen i Magoulas 2005).

### 1.1.2 Personalizacja

Personalizacja przez niektórych jest określana jako możliwość ustalania i zapisywania profili decydujących w sposób bezpośredni o wyświetlanej informacji i sposobie jej prezentacji, na przykład na stronach portali *My Yahoo*[7] czy *MSN.com*[8]. Część ludzi uznaje za personalizację dostarczanie informacji zawierającej chociaż jedną daną związaną z odbiorcą, jak chociażby spotykana często w poczcie elektronicznej wiadomość: *„Mam dla Ciebie Karolu specjalną ofertę...”*. W każdym z tych przypadków różni użytkownicy widzą odmienne rzeczy. Na potrzeby tej pracy przyjmiemy definicję proponowaną przez Chena i Magoulasa (Chen i Magoulas 2005):

> *„Personalizacją nazywamy proces pozwalający na dopasowanie interfejsu oraz adaptację funkcjonalności, struktury, treści i modalności celem sprostania preferencjom pojedynczego użytkownika.”*[9]

[7] *My Yahoo!*: `http://my.yahoo.com/`.

[8] *MSN.com*: `http://www.msn.com/`.

[9] Definicja ta w oryginale przytoczona została w kontekście Systemu Zarządzania Wiedzą, czyli systemu informacyjnego do zarządzania wiedzą organizacji (KMS) (Chen i Magoulas 2005). Na potrzeby pracy adaptujemy tą definicję również na inne obszary personalizacji.

Personalizacja treści należy do najintensywniej badanych obszarów personalizacji [10]. W tym znaczeniu będziemy używać terminu „personalizacja" w dalszej części tej pracy.

### 1.1.3 Model użytkownika i modelowanie użytkownika

Według Chena i Magoulasa (Chen i Magoulas 2005):

> *„Modelem użytkownika nazwiemy taki model, który opisuje odpowiednie aspekty użytkownika prowadzącego interakcję z systemem. Zależnie od dziedziny aplikacji, może zawierać on wiedzę o preferencjach użytkownika, umiejętnościach, zdolnościach, celach, potrzebach i zainteresowania."*

Budowanie tego modelu, czyli modelowanie użytkownika, polega na gromadzeniu danych o nim w celu personalizacji jego interakcji z systemem.

### 1.1.4 Systemy rekomendujące

Kobsa wskazuje na kilku obszarów zastosowania modelowania użytkownika, wśród których za najbardziej płodny i najintensywniej badany uznaje systemy do filtrowania informacji (Kobsa 1990). Wśród nich, obok witryn czasopism internetowych i list Usenetu, funkcjonują systemy rekomendujące, definiowane następująco (Chen i Magoulas 2005):

> *„Systemem rekomendującym nazywamy system, który z wielkiej przestrzeni możliwych opcji wybiera i prezentuje użytkownikowi te, które opowiadają jego potrzebom."*

Zatem to, co odróżnia systemy rekomendujące od prostych systemów wyszukujących to kryterium zindywidualizowania informacji (Burke 2002). Każdy z przedstawionych wyżej systemów: *Zagat Survey*, *Entree*, *EntreeC*, *IRES* i system Pazzaniego pozwalają znaleźć informację. Wszystkie też, poza pierwszym, spełniają definicję systemu rekomendującego.

System rekomendujący dostarcza informacji spersonalizowanej wykorzystując techniki filtrowania informacji (IF). Co warto podkreślić, techniki te mogą, ale nie muszą działać w oparciu o zbudowany model użytkownika. Kryterium filtrowania informacji może być również oparte na krytyce, którą użytkownik wysunął w stosunku do wcześniejszych rekomendacji, na przykład w systemie *Entree*. Co więcej, nawet jeśli korzystamy z modelu użytkownika, to nie musi on być trwały i wyrażony wprost, w postaci listy ulubionych restauracji czy ich cech. Można bowiem za obraz użytkownika uznać jego chwilowe potrzeby i szukać innych użytkowników, którzy w przeszłości przejawiali podobne zachcianki. Wniosek z tego wyciągnęli twórcy księgarni internetowej *Amazon*[11]: skoro klienci kupujący podobne przedmioty mają zbieżne gusta, to można proponować im towary zgodnie z regułą *koszyka z zakupami*: *„Ludzie którzy kupili produkt X, kupili też produkt Y"* . Zatem, gdy w księgarni internetowej *Amazon* zapytamy o „The Magus" Johna Fowlesa, dostaniemy następującą odpowiedź:

[10] Pretschner, Alexander i Gauch, Susan. 1999. *Personalization on the Web*: `http://citeseer.nj.nec.com/pretschner99personalization.html`.

[11] *Amazon*: `http://www.amazon.com`.

`Customers who bought this book also bought`

- `The French Lieutenant's Woman (French Lieutenant's Woman) by John Fowles`
- `Siddhartha by HERMANN HESSE`
- `The Collector (Back Bay Books) by John Fowles`
- `The Ebony Tower by John Fowles`
- `Under the Net by Iris Murdoch`
- `Dirt Music : A Novel by Tim Winton`

W dalszej części pracy będziemy rozważać modelowanie użytkownika wyłącznie w kontekście systemów rekomendujących, w których model użytkownika będzie rozumiany tak, jak określiła go Rich: jako trwały i długoterminowy opis użytkownika (Rich 1979b).

## 1.2 Taksonomia systemów rekomendujących

Analiza istniejących systemów rekomendujących pozwala określić dwie fazy funkcjonowania: (a) *tworzenie i rozwijanie profilu* oraz (b) *eksploatacja profilu*. Pierwsza faza wymaga od projektanta podjęcia decyzji. Po pierwsze, należy określić *reprezentację profilu*, aby system w ogóle funkcjonował. Po drugie – wybrać technikę *inicjalizacji profilu*, po to, aby system mógł wspierać użytkownika od samego początku działania. Po trzecie – sprecyzować rodzaj przechwytywanej *informacji zwrotnej* od użytkownika, świadczącej o zainteresowaniu lub jego braku danym przedmiotem. I po czwarte, należy ustalić techniki *uczenia profilu* na podstawie informacji zwrotnej[12]. Rysunek 1.2 przedstawia relacje między tymi decyzjami.

Kiedy w wyniku pierwszej fazy otrzymujemy profil użytkownika, system rekomendujący może przystąpić do rekomendowania użytkownikowi przedmiotów. Wymaga to szacowania poziomu zainteresowania *aktywnego użytkownika*[13] *aktywnym przedmiotem*[14]. W tym celu analizuje się zebrane informacje (profil aktywnego użytkownika, profile innych użytkowników oraz informacje o przedmiotach). Decyzja, w jaki sposób to zrobić, wymaga wybrania odpowiedniej techniki filtrowania informacji (IF). Zatem w fazie eksploatacji należy określić:

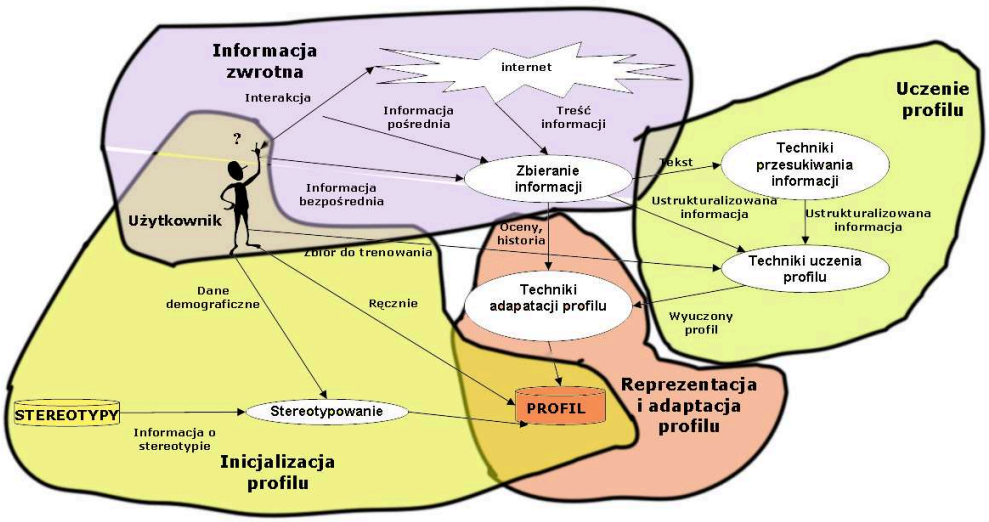
- metodę filtrowania informacji, która bazuje albo na *korelacji użytkownik-przedmiot* (w przypadku technik typu *item-centered*) albo *użytkownik-użytkownik* (w przypadku technik typu *user-centered*) i w związku z tym
- technikę obliczania jednej z powyższych korelacji.

W literaturze tematu to właśnie nazwa techniki filtrowania informacji określa zazwyczaj typ systemu rekomendującego. W związku z tym mamy systemy wykorzystujące metody demograficzne, oparte na treści, kolabaratywne i inne. Rysunek 1.3

[12] Autorzy pracy (Montaner *et al.* 2003) wskazują również na potrzebę określenia techniki *adaptacji profilu*, wzbogacających poprzedni krok o poszerzanie profilu poprzez dodawanie nowych elementów i zapominanie starych. W tej pracy ta kwestia nie będzie omawiana.

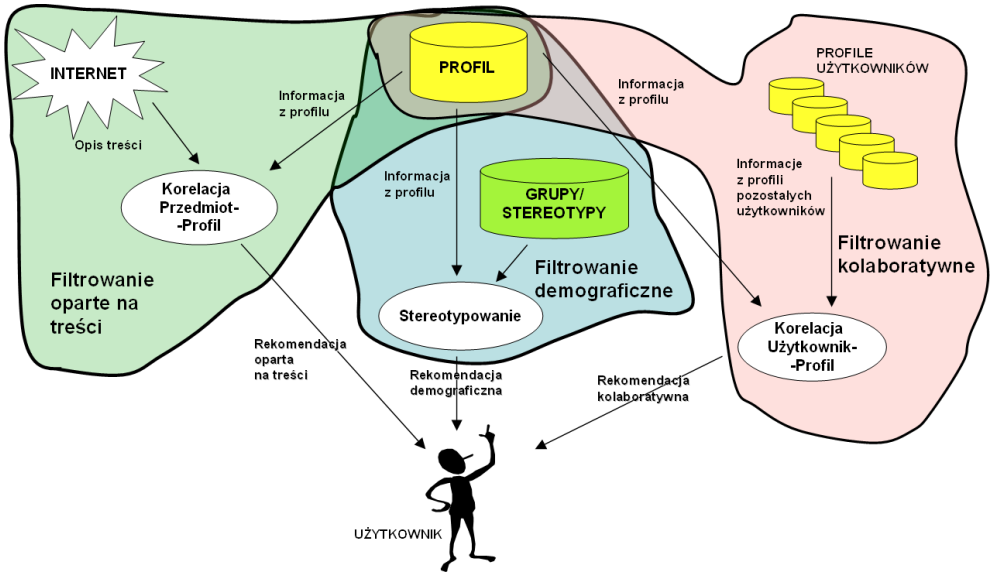
[13] Ang. *active user*.

[14] Ang. *active item*.




Rysunek 1.2: Tworzenie profilu i jego utrzymanie (opracowano na podstawie (Montaner *et al.* 2003)).

prezentuje ogólny obraz fazy eksploatacji.




Rysunek 1.3: Eksploatacja profilu (opracowano na podstawie (Montaner *et al.* 2003)).

Przedstawiony podział systemu oparty został o taksonomię przedstawioną w (Montaner *et al.* 2003). Taksonomia ta ma charakter funkcjonalny, to znaczy określa, jak jakie zadania stoją przed systemem. Szczegółowy opis tych zadań znajduje się w kolejnych paragrafach.

### 1.2.1 Reprezentacja profilu

Wybór reprezentacja profilu stanowi kluczową decyzję przy tworzeniu modelu użytkownika i decyduje o technikach stosowanych w kolejnych etapach konstruowania sys-

temu. Jak wspomnieliśmy we wstępie, istnieją systemy, które nie budują modelu użytkownika explicite, zapamiętują jedynie jego historię, zakupy czy oddane oceny; z drugiej strony istnieją systemy, które wykorzystują te informacje za podstawę uczenia się właściwego modelu użytkownika. W związku z tym wyróżniamy dwa sposoby reprezentacji profilu użytkownika (Rotaru 2005):

- *opartej na pamięci*[15], to jest na zapamiętanych zachowania,
- *opartej na modelu*[16], to jest profilu wyrażonym explicite.

### Reprezentacja oparta na pamięci

W podejściu opartym na pamięci system zapamiętuje interakcje z użytkownikiem jako:

- *historię* (zachowań, nawigacji w obrębie witryny czy zakupów)[17] lub
- *macierz ocen użytkownik-przedmiot*

Wśród systemów opartych na historii do najpularniejszych należy przedstawiany już *Amazon*[18]. Podobne podejście stosuje *WebSell* (Cunningham *et al.* 2001), który w profilu użytkownika przechowuje dwie listy: jedną dla zakupionych przedmiotów ocenionych pozytywnie, drugą – dla przedmiotów ocenionych negatywnie.

W przypadku macierzy ocen każda jej komórka postaci $(u, i)$ przechowuje ocenę przedmiotu $i$ postawioną przez użytkownika $u$.

W profilu typu *memory-based* nie stosuje się etapu uczenia: zebrane informacje są wykorzystywane w procesie jego eksploatacji. Zależnie od używanej techniki wykorzystywane są one do znajdywania podobieństwa między:

- przedmiotami (*item-centered*[19]), gdzie przedmioty opisane są przez zbiór atrybutów (na przykład obsada filmu, serwowana kuchnia w restauracji) lub zbiór słów kluczowych (na przykład w przypadku polecania artykułów prasowych lub recenzji filmowych),
- lub użytkownikami (*user-centered*[20]), gdzie odmienna perspektywa zmusza do szukania podobieństwa między użytkownikami na podstawie podobnych ocen czy zakupów.

W systemach opartych na reprezentacji profilu typu memory-based aktywny przedmiot lub też aktywny użytkownik musi być porównany (w celu określenia podobieństwa) z każdym innym przedmiotem lub użytkownikiem, odpowiednio. W przypadku zastosowań komercyjnych, kiedy to ilość przetwarzanych danych rośnie w dużym tempie, takie podejście wymaga dużej mocy obliczeniowej i przestrzeni fizycznej (Rotaru 2005). Dlatego obsłużenie pojedynczego użytkownika i rekomendacja (eksploatacja profilu) może się niekorzystnie wydłużyć.

[15] Ang. *memory-based*.
[16] Ang. *model-based*.
[17] Zobacz paragraf 1.2.3, aby zapoznać się zachowaniami możliwymi do obserwowania.
[18] *Amazon*: `http://www.amazon.com`.
[19] Czyli skoncentrowanymi wokół przedmiotu.
[20] Czyli skoncentrowanymi wokół użytkownika.

**Reprezentacja oparta na modelu**

Alternatywą dla podejścia opartego na pamięci stanowi stworzenie modeli explicite. Proces uczenia takiego modelu (najczęściej, kiedy użytkownik nie jest podłączony do systemu) może być długi, natomiast sam proces rekomendacji, czyli szacowania aktywnego przedmiotu, staje się relatywnie krótszy.

Najprostszym rozwiązaniem wydaje się *wektor średni*. W momencie rekomendacji zamiast porównywać aktywny przedmiot ze wszystkimi innymi, obliczamy odległość w stosunku do wektora średniego, którego każdy z atrybutów reprezentuje średnią wartości atrybutu wszystkich innych przedmiotów, którymi interesował się aktywny użytkownik. Wartości mogą być typu binarnego (prawda lub fałsz), gdy wskazują obecność preferowanej cechy, lub typu rzeczywistego, kiedy symbolizują częstotliwość, odpowiedniość lub prawdopodobieństwo występowania określonej cechy. Przykładowo *Webmate* (Chen i Sycara 1998) używa wielowymiarowych wektorów do opisu zawartości dokumentów. Każdy dokument charakteryzowany jest przez zbiór słów i częstotliwość, z jaką występują w tym dokumencie, a profil użytkownika stanowi przekrój dokumentów widzianych przez użytkownika.

Do reprezentacji informacji w profilu stosuje się także struktury znaczeniowo bogatsze:

- *ważone n-gramy*[21], pozwalają określić współwystępowania pewnych cech, a więc na przykład kontekst słowa.
- *ważone sieci skojarzeniowe*[22] stanowią graf, w których waga każdej krawędzi opisuje siłę skojarzenia między incydentnymi z nią wierzchołkami. Wykorzystanie technik aktywacji i propagacji sygnału wzdłuż krawędzi pozwala na przeprowadzanie wnioskowań i szukanie informacji pokrewnych.
- *ważone sieci semantyczne* stanowią rozwinięcie powyższych, poprzez wprowadzenie różnego typu relacji (bycia podklasą, synonimem, hiperonimem itp.)[23].

Niektóre profile mają charakter *klasyfikatorów*, skracając w ten sposób proces szacowania zainteresowania aktywnym przedmiotem. Zaliczmy tutaj sieci neuronowe, drzewa decyzyjne, reguły wnioskowania i sieci Bayesa.

### 1.2.2 Inicjalizacja profilu

Zanim system pozna preferencje użytkownika, musi upłynąć pewien okres czasu, zależny od intensywności interakcji użytkownika z systemem. W związku z tym wyłania się problem rekomendacji przedmiotów nowemu użytkownikowi (problem *zimnego startu*[24]). W celu jego rozwiązania stosuje się szereg różnych technik mających na celu inicjalizację profilu.

[21] Ang. *weighted n-grams*.
[22] Ang. *weighted associative networks*.
[23] Sieci semantyczne jako struktury zwane *ontologiami* opisujemy w paragrafie 2.2, a w paragrafie 4.1.1 opisujemy ich zastosowanie w systemach rekomendujących.
[24] Zobacz paragraf 1.3.

### Ręczna inicjalizacja

W tego typu systemach użytkownik jest proszony o podanie wprost swoich zainteresowanie w postaci słów kluczowy, tematów itp. Zaletą takiego podejścia jest przejrzystość działania systemu: rekomendowane przedmioty odpowiadają dokładnie temu, co podał wprost użytkownik. Do wad tego rozwiązania należy konieczność włożenia dużego wysiłku przez użytkownika oraz fakt, że wielu użytkowników nie ma świadomości co do wszystkich swoich zainteresowań, zatem nie potrafi określić ich wprost (Montaner *et al.* 2003).

### Zestaw treningowy

Zestaw treningowy stanowi zbiór przykładów interakcji użytkownika z systemem, na podstawie których inicjalizowany jest profil. Przykładowo, użytkownik proszony jest o ocenienie konkretnych przykładów jako odpowiadające lub nie jego zainteresowaniom. Uzyskane informacje wykorzystywane są w jednym z przedstawionych dalej algorytmów uczenia. Problem, który rodzi to podejście, polega na trudności z wyborem reprezentatywnych przykładów. Może to prowadzić do mało precyzyjnych wyników (Montaner *et al.* 2003).

### Stereotypowanie

Powszechnie przez stereotyp rozumie się uogólnienie na temat osoby lub grupy osób. Takie uogólnienie pozwala uzupełnić obraz określonej osoby, w miejscach gdzie informacje na jej temat są niekompletne. Proces tworzenia modelu takiej osoby nazywamy stereotypowaniem[25]. W systemach rekomendujących stereotypowanie jest metodą pozwalającą na wstępną ocenę danej osoby poprzez zaklasyfikowanie jej do kategorii i zaproponowanie rekomendacji na podstawie stereotypów, które z tą kategorią są związane. Stereotypy zatem zawierają zbiór typowych hipotez na temat członków każdej z kategorii (McCauley *et al.* 1980). Wykorzystanie stereotypów w systemach rekomendujących korzystających z profili użytkowników zostało zaproponowane przez Rich w omawianym już systemie Grundy (Rich 1979b). W systemie tym użytkownik zostaje przyporządkowany do jednego lub więcej stereotypów. Dany stereotyp jest wykorzystywany do rekomendacji w chwili, kiedy wystąpią odpowiednie warunki aktywujące[26]. Użytkownicy mogą być zaklasyfikowani do jednego lub więcej stereotypów: (a) na podstawie odpowiedzi, których udzielają systemowi przy pierwszym użyciu lub (b) na podstawie zachowania, które przejawiają w trakcie interakcji z systemem. Jak zwykle w przypadku zmuszania użytkownika do udzielania odpowiedzi, należy pamiętać, że może to być dla niego irytujące. Użytkownicy mogą mieć również problem z udzieleniem poprawnych informacji na swój temat (Rich 1979b)[27].

### Pusty profil

Niektóre systemy (przykładowo *WebSell* (Cunningham *et al.* 2001)) nie rozwiązują problemu zimnego startu, używając pustego profilu początkowego, który zostaje wy-

[25] Grobman, Gary M. 1990. *Stereotypes and Prejudices*: `http://www.remember.org//guide//History.root.stereotypes.html`.

[26] Ang. *trigger*.

[27] Zobacz również uwagi dotyczące braku zaufania użytkownika do Internetu w paragrafie 1.2.3.

pełniony dopiero w trakcie interakcji użytkownika z systemem.

### 1.2.3 Informacja zwrotna

Przyjrzyjmy się przykładowej sytuacji. Osoba planująca zakup roweru będzie poszukiwać informacji na temat części rowerowych, marek i cen, ale kiedy już podejmie decyzję o tym, jaki model kupić, przestanie się tym interesować. Aby system znał bieżące zainteresowania, potrzeby i możliwości użytkownika, musi otrzymywać od niego informację zwrotną. W polskiej literaturze ten element systemu rekomendującego nazywany jest *informacją relewatną*[28] (Próchnicka 2000).

Różne zachowania mogą świadczyć w odmiennym stopniu o zainteresowaniu przedmiotem. Pamiętajmy również, że ludzie często wyrażają swoje opinie nie wprost. Rodzi się pytanie, jak interpretować takie sygnały i kiedy świadczą one o wyraźnym zainteresowaniu? Idąc dalej: kiedy brak reakcji można odczytać jako brak zainteresowania? O tym wszystkim będzie mowa w kolejnych paragrafach.

#### Analiza

W tabeli 1.1 znalazł się szereg zachowań uznanych przez Nicholsa i Kobsa'e za możliwe do obserwacji (Nichols 1998; Kobsa *et al.* 2001).

| | *Operacja (zmienna)* | *Wyjaśnienie lub przykład* |
|---|---|---|
| 1. | Zakup internetowy (cena) | ... w sklepie internetowym |
| 2. | Ocena (wysokość) | szacowanie lub rekomendacja |
| 3. | Wielokrotne użycie (liczba) | wysyłanie tej samej kartki |
| 4. | Zapis / druk | zapis dokumentu do siebie |
| 5. | Usunięcie | skasowanie przedmiotu |
| 6. | Podanie referencji do | cytowanie lub inne odniesienie się |
| 7. | Odpowiedź (czas) | odpowiedzenie na coś |
| 8. | Zaznaczenie | dodanie do listy "Interesujące" |
| 9. | Analizowanie / czytanie (czas) | obejrzenie całości |
| 10. | Przejrzenie streszczenia (czas) | obejrzenie streszczemia |
| 10. | Dojrzenie | przelotne spojrzenie na |
| 10. | Skojarzenie | nie dostrzeżony rezultat zapytania |
| 11. | Zapytanie | analiza słów kluczowych w zapytaniu |
| | Operacje wyboru | np. klikanie w historii nawigacji |

Tabela 1.1: Zachowania możliwe do obserwowania w trakcie interakcji z systemem rekomendującym (opracowane na podstawie (Nichols 1998; Kobsa *et al.* 2001)).

*Zakup* przez Internet jest uważany za wskaźnik najwyższego zainteresowania produktem. Informację tą wykorzystuje serwisna przykład *Amazon*[29].

Z kolei, w systemie *Syskill & Webert* (Pazzani *et al.* 1996) możemy *ocenić* bezpośrednio dokumenty z pewnej dziedziny na trzystopniowej skali jako *„gorące"*, *„letnie"*

[28] Ang. *relevance feedback*.

[29] Przykład był już przytaczany wcześniej w paragrafie 1.1.4.

i *„zimne”*. Analizując sugestie użytkownika system bada popularność słów zawartych w dokumentach. Nichols (Nichols 1998) wskazuje również na znaczenie samego faktu dokonania lub nie dokonania oceny, w sytuacji, gdy taka *operacja* była możliwa do przedsięwzięcia. Sąd taki może być dokonywany na skali symbolicznej (*„interesujacy”* lub *„nieinteresujący”*, *„lubię”* lub *„nienawidzę”*) lub numerycznej (*Amazon*). W kilku systemach użytkownicy mają możliwość umieszczenia własnego komentarza tekstowego, np. w serwisie wiadomości *GroupLens* (Resnick *et al.* 1994). W internetowych sklepach komputerowych, np. *Komputroniku*[30] użytkownik może dać wyraz swojej satysfakcji (lub jej braku) z zakupionego sprzętu. System prezentuje komentarze o danym elemencie użytkownikom, co ma ułatwić podjęcie decyzji o zakupie. Mimo, że taka sugestia wydaje się bardzo pomocna, to obowiązek przeczytania i oceny, do jakiego stopnia komentarz jest pozytywny lub negatywny, spada na użytkownika.

Podobnie kłopotliwe jest określenie własnych zainteresowań, jak to się dzieje w serwisach filtrowania wiadomości takich, jak *SIFT* (Yan i Garcia-Molina 1995) czy *PointCast Network*[31]. W systemach tych formularze (kwestionariusze) pozwalają na zebranie bezpośrednio od użytkownika informacji na temat jego preferencji, poziomu umiejętności czy znajomości danej dziedziny.

Intuicyjne wydaje się obserwowanie *wielokrotnego użycia*, analogicznie do wypożyczania książek w bibliotece – najpopularniejsze wydawane są najczęściej. Dokumenty, które użytkownik chce zachować dla jakiś późniejszych celów, mogą być *drukowane* lub *zapisywane* na dysku (np. w serwisie rekomendującym muzykę *SmartRadio* (Hayes i Cunningham 1999; Hayes i Cunningham 2000)), a o niechęci wobec obiektu może świadczyć decyzja o jego *usunięciu*, na przykład skasowanie wiadomości w Usenet. W tym kontekście możemy mówić również o *operacjach odpowiedzi* i *podania referencji do*, gdzie o zainteresowaniu daną wiadomością świadczy liczba jej cytowań, odniesień do niej, a także czas poświęcony na przygotowanie odpowiedzi na nią. Analogia jest naturalna: zainteresowania globalnego (ogólnospołecznego) daną pracą naukową dowodzi liczba jej cytowań, a popularności strony internetowej – liczba odnośników odwołujących się do niej[32].

Kategorie *operacji*: *analizowanie / czytanie*, *przejrzenie streszczenia* odnoszą się do tej samej czynności czytania dokumentu, a zmienną świadcząca o zainteresowania jest czas czytania. Systemy rekomendujące z powodzeniem bazują na tej metodzie, jak chociażby wyszukiwarka newsów *Morita & Shinoda* (Morita i Shinoda 1994). Efektywność serwisu i metody potwierdzają autorzy podobnego systemu, *GroupLens* (Konstan *et al.* 1997), stwierdzając, że przewidywania o preferencjach użytkownika oparte na badaniu czasu spędzonego na czytaniu są niemal tak samo dokładne jak te, oparte na bezpośrednich ocenach dokumentów przez czytającego. Wnioski te jednak na zbudowane są na mało praktycznym modelu, co zostanie szerzej wyjaśnione w dalszych rozważaniach.

Analiza *zapytania* w wyszukiwarce (słów kluczowych) może być wykorzystana przy wyszukiwaniu podobnych fraz (Koenig 1990). Zauważmy również, że podanie przez

[30] *Komputronik*: `http://www.komputronik.pl/`.
[31] *Pointcast Network*: `http://www.pointcast.com/`.
[32] *Google*: `http://www.google.com/`.

użytkownika określonych słów w zapytaniu np. o dokument, dobrze charakteryzuje jego preferencje: zwykle najbardziej odpowiedni będzie ten dokument, który zawiera najwięcej spośród podanych słów.

Wyróżnioną kategorię stanowią *operacje wyboru*, przez które Kobsa (Kobsa *et al.* 2001) rozumie przede wszystkim klikanie na odnośnik. Podejście to jest wykorzystane przede wszystkim w systemach rekomendującym strony w Internecie, jak *Letizia* (Lieberman 1995) czy *Personal Web Watcher* (Mladenic 1996), w których kliknięcie na odnośnik oznacza zainteresowanie tematem kryjącym się pod nim. Ciąg wybieranych odnośników składa się na historię nawigacji, która badana jest np. w systemie filtrującym newsy internetowe *ACR News* (Mobasher *et al.* 2000). Tu, w odróżnieniu od dwóch poprzednich systemów, liczy się kolejność wybieranych odnośników.

Do *operacji wyboru* należy również wliczyć takie zachowania jak maksymalizowanie, minimalizowanie czy zmianę rozmiaru okna z dokumentem. W wyszukiwarce newsów *Anatagomy* (Sakagami *et al.* 1997) powiększenie okna z artykułem jest przyjmowane za dowód zainteresowania.

### Zachowania wprost i nie-wprost

Wyobraźmy sobie taką sytuację: użytkownik poszukuje tekstu na temat globalnego ocieplenia. Przegląda tytuły, a dla każdego, który wyda się mu interesujący, czyta streszczenie. W końcu dociera do najtrafniejszego. Dalej szuka informacji o okolicznościach zatopienia statku Tytanik, ale tym razem proszony jest o zaznaczenie tych dokumentów, które ocenia jako istotne. Które z tych podejść jest bardziej efektywne? Czy to, w którym system widzi związek między czytaniem streszczenia a zainteresowaniem i odpowiednio do tego, proponuje użytkownikowi dokumenty zbliżające go do odpowiedzi na jego pytania? Czy też to, w którym system uczy się o użytkowniku na podstawie bezpośrednio podanej oceny? Opisana wyżej sytuacja przedstawia fragment eksperymentu, który przygotowali autorzy systemu *WebDocSum* (White *et al.* 2002; White *et al.* 2001) po to, aby odpowiedzieć na zadane wyżej pytania. Różnice między trafnością polecanych tematów były nieznaczne, więc oba podejścia postanowiono do pewnego stopnia traktować zamiennie.

W rzeczywistości postawiony problem od dawna nurtuje badaczy przedmiotu. Zaprezentowane podejścia w interakcji z systemem wpisują się w jedną z poniższych kategorii zachowań:

- *reakcje nie-wprost* (informacja pośrednia, ang. *implicit feedback*), nie naruszające naturalnego ciągu nawigacji w ramach aplikacji; system wnioskuje o preferencjach użytkownika obserwując jego interakcję z aplikacją;
- *reakcje wprost* (informacja bezpośrednia, ang. *explicit feedback*), w celu jej zdobycia system wymusza na użytkowniku podjęcia określonej akcji, np. ocenienie przedmiotu.

Plusem informacji typu *explicit feedback* jest jej prostota. Zachowanie systemu jest przewidywalne. Kiedy na przykład w serwisie *SIFT* (Yan i Garcia-Molina 1995) artykuł

zostanie dostarczony do użytkownika, może on odgadnąć, dlaczego akurat ten został mu podsunięty. Dzieje się tak, ponieważ system jest skrajnym przykładem stosowania techniki *explicit feedback* – w trakcie rejestracji użytkownik zobowiązany jest do podania swoich zainteresowań w postaci słów kluczowych, zajmujących go tematów itp. Tym samym od użytkownika wymaga się dodatkowego wysiłku. A ponieważ zainteresowania człowieka zmieniają się wraz z upływem czasu, to musi on wciąż na nowo modyfikować w systemie informacje o swoich preferencjach. W dodatku ludzie nie zawsze potrafią dokładnie określić, czym się interesują, jeśli ich preferencje pozostają nieuświadomione. Bardziej komfortowe w obsłudze wydaje się więc ocenianie przedmiotów na bieżąco w trakcie oglądania (czytania). W praktyce również to podejście ma kilka wad (Montaner *et al.* 2003):

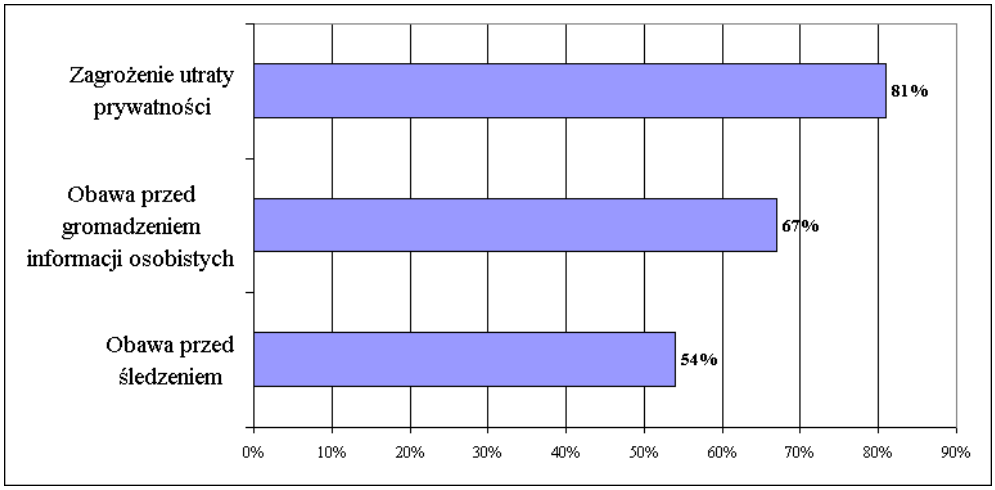
- skala numeryczna może być nieadekwatna do reakcji opisujących stosunek człowieka do przedmiotu,
- podane oceny nie są niezależne, jak to się często zakłada, ale zmieniają się w czasie, jak chociażby w przypadku, gdy użytkownik ma zaopiniować kilka artykułów: kolejny z nich może być oceniony niżej, jeśli wcześniej obejrzane satysfakcjonowały użytkownika,
- użytkownicy niechętnie udzielają opinii; Pazzani przytacza, że jedynie 15% użytkowników jest skłonnych do zrobienia tego, nawet jeśli byliby do tego zachęcani (Pazzani i Billsus 1997). Użytkownicy w ogóle są niechętni do podejmowania działań nie prowadzących ich bezpośrednio do zamierzonych celów, jeśli nie mają z tego natychmiastowych korzyści, nawet jeśli te korzyści w dłuższej perspektywie czasowej byłyby gwarantowane (Carroll i Rosson 1987). A jeśli udzielą informacji, to nie ma gwarancji, że dane będą poprawne. Wynika to z braku zaufania do polityki ochrony prywatności witryn internetowych (patrz rysunek 1.4 i rysunek 1.5). Użytkownik woli zataić te informacje lub podać fałszywe[33] tylko po to, aby przemieścić się dalej w systemie (Weihberg 2003).

Mimo szeregu problemów *explicit feedback* należy pamiętać, że jest to technika wysoce wydajna (Salton i Buckley 1997; Buckley i Salton 1995).

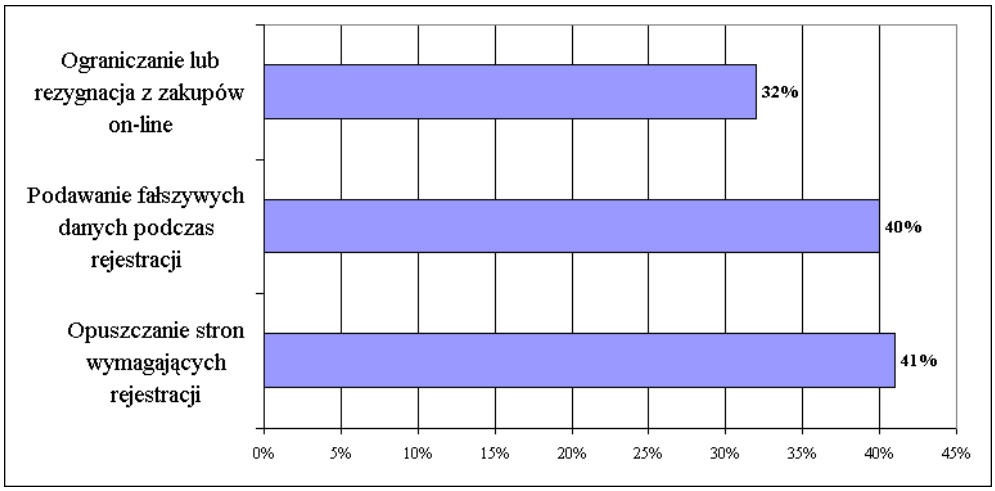
W przypadku *reakcji nie-wprost* wydajność ta nie została stanowczo udowodniona. Ten typ reakcji ma jednak jedną decydującą przewagę w postaci komfortu interakcji. *Implicit feedback* nie rozprasza ani nie denerwuje. Użytkownik klika interesujące go odnośniki, powiększa okna z ciekawymi artykułami, ale nie angażuje się w żadne dodatkowe akcje. Z definicji system nie inicjuje żadnej interakcji w stronę użytkownika i – co dowiedziono – z zachowania potrafi wywnioskować jego zainteresowania (Chatterjee *et al.* 2003). Należy pamiętać, że wnioski wyciągane z reakcji nie-wprost opierają się jedynie na przypuszczeniach co do motywów działania użytkowników.

Jest tak na przykład w przypadku przewidywania preferencji użytkownika na podstawie obserwacji czasu poświęconego na czytanie artykułu. Choć zostało to nawet

[33] *Reasons for Not Registering*: `http://www.cc.gatech.edu/gvu/user_surveys/survey-1998-10/graphs/privacy/q48.htm` oraz `http://www.thestandard.com/article/display/0,1151,235,00.html`.



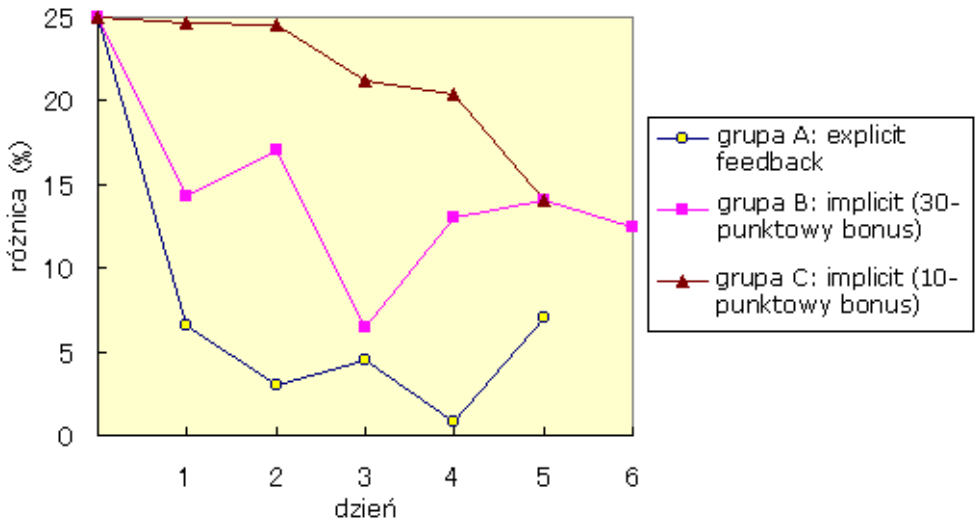

Rysunek 1.4: Zagrożenia niepokojące użytkowników korzystających z Internetu (opracowany na podstawie (Kobsa 2002)).




Rysunek 1.5: Środki ostrożności zachowywane przez użytkowników korzystających z Internetu (opracowany na podstawie (Kobsa 2002)).

udowodnione doświadczalnie przez autorów wyszukiwarki newsów *Morita & Shinoda* (Morita i Shinoda 1994), to warto zwrócić uwagę, że uczestnikom eksperymentu kazano wyłącznie czytać artykuły, nie odchodzić od terminala czy robić sobie przerwę na kawę lub na odebranie rozmowy telefonicznej. W praktyce użytkownik może nawet nie czytać otwartego artykułu, jeśli jest on zasłonięty przez okna innych aplikacji czy innych artykułów. Bardziej znaczące może być wyciąganie z czasu interakcji negatywnych wniosków o braku zainteresowania. Jeśli użytkownik przerwie wczytywania prezentacji, filmu czy strony internetowej i wróci do poprzedniej, jest to znaczący przejaw jego niechęci (Kobsa *et al.* 2001).

W systemach obserwujących ruchy myszką, przewijanie, powiększanie czy minimalizowanie okien z dokumentami problemem pozostaje ogromna ilość informacji wymagająca czasochłonnego przetworzenia. Najpierw należy określić w jakim celu te dane mają być gromadzone (np. do oceny zainteresowań użytkownika w danym zbiorze stron internetowych) (Kobsa *et al.* 2001). Skuteczność tego typu systemów została dotychczas potwierdzona jedynie w środowisku laboratoryjnym (ale wyniki wydają się obiecujące (Montaner *et al.* 2003)). System wyszukiwania newsów *Anatagomy* (Sakagami *et al.* 1997) za kliknięcie odnośnika do tekstu lub powiększenie okna z samym artykułem przyznaje punkty zainteresowania. Autorzy skonfrontowali to podejście z wersją systemu opartą o *reakcje wprost*. Okazało się, że aby system rekomendował trafnie, liczba punktów przyznawanych przez system za pojedynczą reakcję użytkownika powinna być adekwatna do zainteresowania użytkownika (rysunek 1.6).




Rysunek 1.6: Porównanie *reakcji wprost* i *nie-wprost* w systemie *Anatagomy* (opracowana na podstawie (Sakagami *et al.* 1997)).

Zatem personalizacja z wykorzystaniem *implicit feedback* potrafi być skuteczniejsza niż w przypadku *explicit feedback*, co podważa dotychczas udowadnianą wydajność tej drugiej. Która z nich jest ostatecznie lepsza? W jaki sposób je porównywać, skoro wyniki przedstawianych do tej pory eksperymentów (Sakagami *et al.* 1997; White *et al.* 2002; White *et al.* 2001; Salton i Buckley 1997; Buckley i Salton 1995) dają wzajmnie sprzeczne wnioski? Tylko pozornie. Zanim jednak wyjaśnię dlaczego, zastanówmy się,

czy nie warto połączyć obu metod w jeden twór, nazywany w literaturze *podejściem hybrydowym* (Montaner *et al.* 2003)).

Załóżmy, że w systemie obserwującym *implicit feedback* użytkownik chce od czasu do czasu wyrazić swoją opinię bezpośrednio. System ocenił oglądany artykuł jako przeciętny, ale użytkownik chce podciągnąć ocenę w górę, ponieważ tekst jest według niego wyjątkowo ciekawy. Idąc za tą sugestią, autorzy systemu *Anatagomy* stworzyli kolejny eksperyment, w którym użytkownicy mogli wyrazić opinię bezpośrednio, ale zasadniczo system uczył się na podstawie ich *reakcji nie-wprost*.

Zbliżamy się powoli do kwestii rozwiązania problemu, która z przedstawionych technik jest najbardziej skuteczna. Wynik nie zależy wyłącznie od wyboru kategorii reakcji (*wprost* czy *nie-wprost*), ale także od modelu ich zbierania i wyciągania wniosków. Nichols (Nichols 1998) porównał dwa systemy oceniające zainteresowanie użytkownika na podstawie czasu czytania wiadomości: *GroupLens* (Resnick *et al.* 1994) i *Morita & Shinoda* (Morita i Shinoda 1994). Pierwszy (model 1) zbiera oceny użytkownika oraz *reakcje nie-wprost*, następnie porównuje je i na podstawie ich zbieżności przedstawia swoje rekomendacje. Drugi (model 2) zbiera najpierw *reakcje nie-wprost* od użytkownika i na ich podstawie buduje wstępne wnioski. Następnie użytkownik wraca do przeczytanych już dokumentów i ocenia je. Dopiero wtedy system porównuje oba typy reakcji i wyciąga końcowe wnioski. Żadne z rozwiązań nie jest idealne. W modelu 1 korelacja pomiędzy *reakcjami wprost* a *nie-wprost* może być niska, a więc o małym znaczeniu. W modelu 2 użytkownik wystawia oceny przedmiotom w momencie, kiedy już wszystkie przejrzał. Mając całościowy obraz przedstawionej oferty, może wydawać inne sądy niż w modelu 1.

**Wniosek 1.2.1** *System powinien przyjmować zarówno* reakcje wprost*, jak i* nie-wprost*, przy czym te pierwsze powinny być wyżej punktowane.*

**Wniosek 1.2.2** *Nie należy użytkownika zmuszać do dokonywania oceny wprost, ale należy dać mu szansę jej postawienia.*

### Sugestie pozytywne i negatywne

A co będzie, jeśli użytkownik nie przeczyta wszystkich artykułów i będzie wybierał tylko spośród tych tytułów, które zmieściły się w oknie? Aby znaleźć rozwiązanie, należy odpowiedzieć na bardziej ogólne pytania:

1. Czy brak reakcji użytkownika w stosunku do danego przedmiotu można rozumieć jako niską ocenę w stosunku do niego – *sugestię negatywną* (ang. *negative feedback*)?

2. Czy nie wystarczy, że system zbiera informacje wyłącznie na temat tego, co użytkownik lubi – *sugestie pozytywne*?

3. Czy przy tak silnych ograniczeniach budowanie przypuszczeń o preferencjach użytkownika jest zasadne?

Odpowiadając na pytanie pierwsze wielu autorów systemów opartych o *implicit feedback* (*WebWatcher* (Armstrong *et al.* 1995; Joachims *et al.* 1997), *NewsDude* (Billsus

i Pazzani 1999)) założyło, że obiekty niewidziane lub nieklikнięte nie odpowiadają gustowi użytkownika. Podział ten ma charakter heurytystyki i stawia na równi obiekty niewidziane (nieklikнięte) z nieinteresującymi. Takie rozwiązanie wydaje się błędne, zwłaszcza w przypadku przeglądarki WWW, w której fakt nie-kliknięcia elementu nie musi znaczyć, że użytkownik uznał go za nieodpowiedni. Jest to raczej związane z faktem, że wiele stron nie mieści się w całości na ekranie, a elementy, które znajdują się poza nim, są oglądane przez relatywnie małą grupę użytkowników[34]. Rozwiązaniem wydaje się podejście zaproponowane we wspomnianym już systemie *Letizia* (Lieberman 1995), aby za odnośniki nieinteresujące użytkownika uznać tylko te, które bezpośrednio sąsiadują z odnośnikami klikniętymi. Alternatywnie, wyzwaniem staję się taka kompozycja strony, aby najważniejsze informacje były wyświetlane bez konieczności przewijania ekranu. W systemie personalizującym nie tylko treść, ale i formę wyświetlania (zależną np. od przeglądarki, rozdzielczości ekranu) jest to możliwe, ale tylko do pewnego stopnia, narzuconego przez możliwości techniczne medium użytkownika (np. wielkość ekranu telefonu komórkowego).

Widać więc, że uzyskanie *sugestii negatywnych*[35] jest często problematyczne (Schwab i Pohl 1999). Być może zbieranie tego rodzaju informacji nie jest potrzebne. W systemach opartych o *reakcję wprost* wskazywanie, że nie jest się zainteresowanym może być kłopotliwe dla użytkownika. Autorzy systemu *SurfAgent* (Somlo i Howe 2003) pokazali, że niewielki zysk efektywności w przypadku *negative feedback* nie wydaje się warty dodatkowego wysiłku użytkownika (informowania, że nie jest się zainteresowanym). Eksperyment polegał na ocenie dwóch wersji systemu. Pierwszej, w której użytkownik sugeruje systemowi zainteresowanie kliknięciem guzika *„feedback button"* i drugiej, pozwalającej użytkownikowi okazywać również brak zainteresowania. Gwoli obiektywności, istnieją również systemy, w których użycie *negative feedback* powoduje spadek dokładności działania systemu (np. system *ELFI* (Kobsa *et al.* 2000)).

Ponadto istnieją systemy, w których *negative feedback* niezbędny. Holte i Yan (Holte i Yan 1996) uważają, że niesie on ze sobą dramatyczną poprawę działania systemu. Ich system pozwala na wyszukiwanie klasy m.in. według nazw funkcji, które są przez nią implementowane. W przypadku, gdy dwie funkcje są podobne (co do nazw), a użytkownik poszukuje jednej (a drugą zauważalnie odrzuca), to sugerowanie się przez system tylko pozytywnym feedbackiem dałoby efekt niezgodny z oczekiwaniem użytkownika.

Wykorzystanie sugestii negatywnych wydaje się potrzebne z jeszcze jednego powodu. Otóż większość systemów rekomendujących do uczenia profilu wykorzystuje algorytmy typu *Machine Learning* (Mitchell 1997), które z założenia dzielą zbiór przedmiotów na dwie części: pozytywną i negatywną.

Wydaje się zatem, że dwa czynniki mają wpływ na wykorzystanie sugestii negatywnych: środowisko interakcji i metoda uczenia. Wrócimy do tego problemu w paragrafie 4.4.1, kiedy poznamy kontekst, w jakim będziemy projektowali metodę modelowania użytkownika.

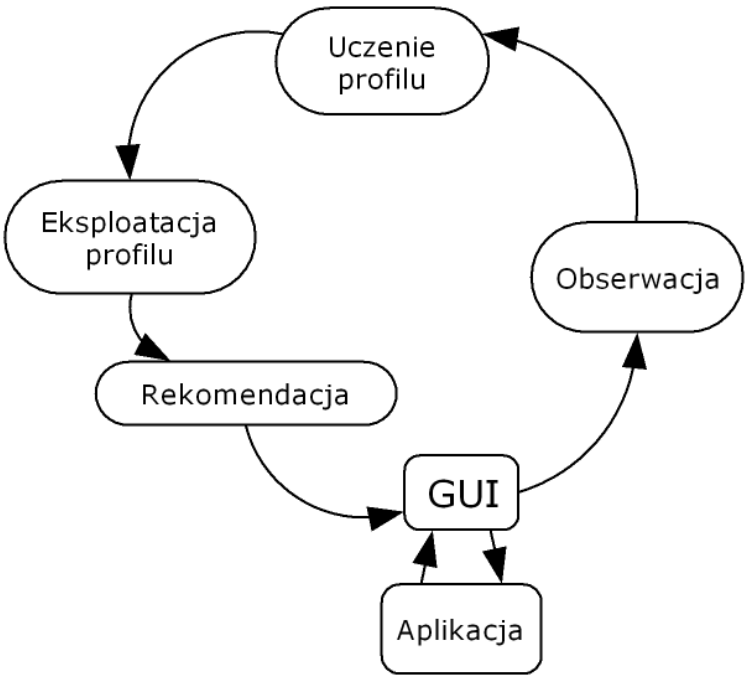
[34] Nielsen, Jakob. 1996. *Top Ten Mistakes in Web Design*: `http://www.useit.com/alertbox/9605.html`.

[35] Ang. *negative feedback*.

### 1.2.4 Uczenie i eksploatacja profilu

Procesy uczenia i eksploatacji profilu są ze sobą ściśle związane (zobacz rysunek 1.7). Informacje o przedmiotach polecanych użytkownikowi w procesie rekomendacji są wyświetlane za pomocą GUI aplikacji. Informacje o reakcjach użytkownika na te rekomendacje trafiają w procesie obserwacji do aplikacji. Reakcje te stanowią podstawę uczenia profilu. Dzięki ciągłemu procesowi uczenia wyniki rekomendacji (procesu eksploatacji profilu) stają się bardziej trafne.

Każda technika rekomendacji, a więc uczenia i eksploatacji profilu, wymaga określenia (a) *zaplecza danych*, które system musi posiadać przed rozpoczęciem całego procesu rekomendacji, (b) *danych wejściowych*, jakie użytkownik przekazuje systemowi w ramach informacji zwrotnej i (c) *algorytmu*, który korzysta z danych wejściowych i zaplecza danych w celu wypracowania sugestii. Na tej podstawie takiego podziału można wyróżnić trzy główne techniki rekomendacji[36], prezentowane w tabeli 1.3.




Rysunek 1.7: Proces rekomendacji w systemie.

#### Techniki kolaboratywne

Techniki kolaboratywne[37] należą do najbardziej dojrzałych i najszerzej implementowanych spośród technik rekomendacji (Burke 2002). Ich ideą jest rekomendowanie tych samych przedmiotów użytkownikom o zbieżnych opiniach. Dokładniej, eksploatacja profilu polega na wyznaczeniu sąsiedztwa użytkowników podobnych do użytkowni-

[36]Burke prezentuje dodatkowo dwa inne rodzaje technik: *oparte na wiedzy* (ang. *knowledge-based*) i *oparte na użyteczności* (ang. *utility-based*) (Burke 2002), a Montaner omawia technikę *opartą na opiniach* (ang. *opinion-based*) (Montaner *et al.* 2002). Żadna z nich nie jest przedmiotem tej pracy.

[37]Nazywane też *social filtering* (Malone *et al.* 1987), *people-to-people correlation* (Schafer *et al.* 1999) lub oparte na klice (ang. *clique-based*) (Kobsa *et al.* 2001).

| *Technika* | *Zaplecze* | *Wejście* | *Przetwarzanie* |
|---|---|---|---|
| *Kolaboratywna* | Oceny od $U$ o przedmiotach w $I$. | Oceny od $u$ o przedmiotach w $I$. | Znajdź w $U$ użytkowników podobnych do $u$ i ekstrapoluj ich oceny na temat $i$. |
| *Oparta na treści* | Cechy przedmiotów w I. | Oceny od $u$ o przedmiotach w $I$. | Generuj klasyfikator, który odpowiada zachowaniom $u$ i użyj go na $i$. |
| *Demograficzna* | Informacje demograficzne o $U$ i ich oceny o przedmiotach w $I$. | Informacje demograficzne od $u$. | Znajdź użytkowników w $U$, którzy są demograficznie podobni do $u$ i ekstrapoluj ich oceny na temat $i$. |

Tabela 1.3: Techniki rekomendacji: $U$ – zbiór wszystkich użytkowników, $u$ – użytkownik aktywny, $I$ – zbiór wszystkich przedmiotów, $i$ – przedmiot aktywny (opracowane na podstawie (Burke 2002)).

ka aktywnego (za pomocą ustalonej metody korelacji), a następnie ekstrapolacji opinii sąsiedztwa na temat aktywnego przedmiotu. W odróżnieniu od technik na opartych na treści[38] techniki kolaboratywne potrafią polecać przedmioty niezależnie od ich fizycznej reprezentacji. Ma to szczególne znaczenie, gdy przedmiotem rekomendacji są filmy, przedstawienia, książki itp., o których zainteresowaniu decydują osobiste gusta i względy estetyczne, trudne do zdefiniowania w sposób formalny (Burke 2002). Z drugiej strony strony brak konieczności jawnej reprezentacji może prowadzić do utraty przejrzystości w działaniu technik kolaboratywnych (Herlocker *et al.* 2000). Techniki te stanowią swoiste „czarne skrzynki" – wyrocznie, których rekomendacji nie można kwestionować. W takiej sytuacji użytkownik nie może wskazać, którym rekomendacjom ufa, a w które wątpi, co może mieć szczególne znaczenie w przypadku systemów doradzających w dziedzinach bardziej ryzykownych (na przykład przy inwestowaniu na giełdzie papierów wartościowych) (Montaner *et al.* 2003). Brak konieczności jawnej reprezentacji i – w związku z tym – niemożności określenia podobieństwa między przedmiotami prowadzi do tzw. *problemu nowego przedmiotu*[39]. W systemach, w których liczba dostępnych przedmiotów znacznie przewyższa liczbę użytkowników, prowadzi to do *problemu rozrzedzenia*[40]: część przedmiotów pozostaje nie oceniona lub oceniona przez liczbę użytkowników niewystarczającą do trafnej rekomendacji.

W wielu systemach wykorzystujących techniki kolaboratywne profil jest reprezen-

[38] Patrz dalej.
[39] Zobacz paragraf 1.3.2.
[40] Ang. *sparsity problem.*

towany w postaci macierzy ocen użytkownik-przedmiot. Wpływa to na wydajność systemu, gdyż brak potrzeby wnioskowania o potrzebach użytkownika pozwala uniknąć etapu uczenia (Montaner *et al.* 2003).

### Techniki demograficzne

Techniki demograficzne, podobnie jak kolaboratywne, bazują na korelacji między użytkownikami. W ten sposób system poleca podobnym ludziom te same przedmioty, z tą jednak różnicą, że podobieństwo określane jest na podstawie danych demograficznych. Przykładowo, system *LifeStyle Finder* potrafi określić styl życia użytkownika na podstawie (a) danych demograficznych podanych przez niego oraz (b) informacji zawartych w amerykańskim spisu ludności, informacji o prenumeratach, zakupach i wynikach kwestionariuszy (Krulwich 1997). Wadą technik demograficznych jest konieczność uzyskania od użytkownika danych demograficznych, dlatego autorzy wspomnianego systemu ograniczyli się do kilku niezbędnych pytań. Alternatywnie, Pazzanini zaproponował technikę demograficzną polegająca na zebraniu informacji o użytkownikach poprzez analizę słów kluczowych na ich stronach domowych (Pazzani 1999). W ten sposób system Pazzaniniego uczy się profili klientów, a dokładniej stereotypów klientów wybranych restauracji[41]. Stereotypowanie, czyli generalizacji zainteresowań użytkowników, nie uwzględnia indywidualnych preferencji użytkownika i może prowadzić to zbyt ogólnych rekomendacji (Montaner *et al.* 2003). Co więcej, brak procesu uczenia i adaptacji profili w tego typu technikach nie pozwala na dostosowanie profilu użytkownika do zmieniających się w czasie zainteresowań użytkownika[42] (Kobsa *et al.* 2001; Koychev 2000).

### Techniki oparte na treści

Techniki oparte na treści[43] zakładają, że użytkownik przy wyborze odpowiedniego przedmiotu, kieruje się jego charakterystyką. System musi znać charakterystyki wszystkich przedmiotów, które widział użytkownik i polecać tylko te, które są do nich podobne, a więc o zbliżonej charakterystyce. Z tego powodu ten rodzaj technik nosił początkowo nazwę *filtrowania kognitywnego*[44] (Malone *et al.* 1987). Techniki oparte na treści są szeroko stosowane w systemach filtrujących dokumenty, takich jak *NewsWeeder* (Lang 1995). W przypadku przedmiotów o ustalonym zestawie atrybutów (na przykład filmy opisywane przez: reżysera, typ i obsadę aktorską) używane jest określenie technik *opartych na atrybutach*[45]. Niestety, techniki te nie znają tej cechy przedmiotu, która zaważyła o opinii użytkownika. Może być też tak, że taka cecha jest niemożliwa do zinterpretowania przez system, jak na przykład porywający styl artykułu prasowego.

Uczenie profilu tego rodzaju techniką wymaga od systemu ekstrakcji cech, słów kluczowych lub innych informacji (zależnie od typu rekomendowanych przedmiotów).

[41] Dlatego często w literaturze termin „stereotypowanie" stosowany jest zamiennie z terminem „techniki demograficzne".

[42] Aby rozwiązać ten problem Kobsa i Fink wskazują możliwość okresowej reklasyfikacji użytkowników. Proces ten jest jednak czasowo i obliczeniowo złożony i może powodować opóźnienia w zmianie profili użytkowników (Kobsa *et al.* 2001).

[43] Ang. *content-based.*

[44] Ang. *cognitive filering.*

[45] Ang. *feature-based.*

Wyekstrahowane informacje stanowią podstawę do dzielenia użytkowników na grupy, w obrębie który preferowane są podobne przedmioty. W procesie tym wykorzystuje się jeden z omawianych wcześniej klasyfikatorów[46].

Eksploatacja profilu wymaga określenia metody korelacji użytkownik-przedmiot.

## 1.3 Problem zimnego startu

Obok wspomnianych wcześniej problemów towarzyszących technikom rekomendujących istotną kwestię stanowi wprowadzenie do systemu nowego użytkownika lub nowego przedmiotu (Burke 2002; Rotaru 2005). Jest to tak zwany *problem zimnego startu*[47].

### 1.3.1 Problem nowego użytkownika

Ponieważ rekomendacje oparte są na analizie historii zachowań pojedynczego użytkownika (techniki oparte na treści) lub porównywaniu jej z historiami innych użytkowników (techniki kolaboratywne) w początkowej fazie interakcji system może dostarczać użytkownikowi mało trafne rekomendacje. W efekcie prowadzi to do frustracji użytkownika i braku wiary w możliwości systemu. Rozwiązań jest kilka: wykorzystanie różnych technik inicjalizacji profilu przedstawionych wcześniej (stereotypowanie, zestaw pytań kontrolnych lub przedmiotów do oceny) czy wykorzystanie dodatkowo innych technik rekomendacji (opartych na wiedzy lub na użyteczności) (Burke 2002).

### 1.3.2 Problem nowego przedmiotu

Problem nowego przedmiotu występuje w systemach, w których ilość pojawiających się przedmiotów do rekomendowania przewyższa znacznie ilość użytkowników (na przykład w systemach polecających artykuły prasowe): każdy użytkownik jest w stanie ocenić jedynie kilka przedmiotów. Dodatkowo użytkownik oceniając przedmiot jako pierwszy, nie zyskuje w ten sposób żadnych bezpośrednich korzyści, ponieważ wstępne oceny nie poprawiają jeszcze możliwości znalezienia pokrewnych przedmiotów czy użytkowników. Konieczne jest zatem zachęcanie użytkowników do oceniania nowych przedmiotów[48] (Burke 2002). W przypadku technik opartych na treści (atrybutach) rozwiązaniem wydaje się również porównanie cech nowego przedmiotu z cechami znanych już przedmiotów.

## 1.4 Łączenie różnych technik personalizacji

Hybrydowe systemy rekomendujące łączą dwie lub więcej technik rekomendujących celem osiągnięcia wyższej wydajności i wyeliminowania ograniczeń tych z nich, które zostały wymienione w paragrafie 1.3. Najczęściej łączy się jedną z technik kolaboratywnych, aby wyeliminować problem zimnego startu. Tabela 1.4 przedstawia różne sposoby takiego łączenia .

[46] Patrz paragraf 1.2.1.

[47] Znany również pod nazwą *cold start* lub *ramp-up* problem.

[48] Problem ten określany jest mianem *pierwszego oceniającego* (ang. *early rater*).

| *Metoda łączenia* | *Opis* |
|---|---|
| Ważenie | Wyniki (głosy) z kilku technik rekomendujących są łączone, aby utworzyć jedną rekomendację. |
| Przełączanie | System przełącza się między technikami rekomendującymi zależnie od aktualnej sytuacji. |
| Mieszanie | Rekomendacje z kilku różnych różnych systemów rekomendujących są prezentowane jednocześnie. |
| Kombinowanie cech | Cechy z różnych źródeł rekomendowanych danych trafiają do jednego algorytmu rekomendującego. |
| Kaskadowa | Jeden system rekomendujący poprawia wyniki otrzymane w wyniku pracy drugiego. |
| Powiększanie cech | Wyjście jednej techniki staje się wejściem dla innej. |
| Poziom meta | Model wyuczony przez jeden system rekomendujący staje się wyjściem dla drugiego. |

Tabela 1.4: Metody łączenia różnych technik rekomendujących (opracowano na podstawie (Burke 2002)).

# Rozdział 2

# Semantyczna Sieć WWW

Wyobraźmy sobie następujący scenariusz[1]: poszukujemy w Internecie artykułu o polskich mnichach, co jest równoznaczne z faktem poszukiwania informacji o polskich zakonnikach. Dla człowieka istnienie synonimów jest oczywiste, dla maszyny przetwarzającej zapytanie w obecnym środowisku WWW – nie. Ostatecznie więc dostajemy zawężoną ilość informacji.

Problemem jest też nadmiar informacji. Załóżmy, że planujemy podróż z Poznania do Bangkoku w połowie września. Hipotetycznie, musielibyśmy zorientować się, jakim środkiem transportu moglibyśmy dostać się do miasta docelowego, wybrać przewoźnika i odnaleźć jego witrynę. Kiedy dokonalibyśmy już wszystkich formalności dotyczących rezerwacji biletu, należałoby jeszcze zarezerwować pokój hotelowy. To z kolei wymagałoby odnalezienia serwisów hoteli w centrum Bangkoku, a spośród nich wybrania najtańszego. No i jeszcze wynajem samochodu... Scenariusz ten zapowiada żmudny proces wyszukiwania, wypełniania formularzy i oceniania wartości informacji.

Problem w tym – jak zauważa North (North i Hermans 2000) – że choć niemalże wszystko w sieci jest czytelne dla komputerów, to niemalże nic nie jest dla nich zrozumiałe. Z jednej strony przyczynia się do tego anarchizm globalnej sieci: brak jasno określonych struktur informacji i relacji między nimi, a z drugiej - wymieszanie czystej informacji ze sposobem jest formatowania (chociażby HTML). Paradoks ten dostrzega również Tim Berners-Lee. Ten sam, który w 1989 wymyślił sieć WWW, dziesięć lat później tworzy wizję nowej „pajęczyny": *Semantyczna Sieć WWW*[2], mającą stanowić rozwiązanie dotychczasowego problemu. W3C (WWW Consorcium), którego dyrektorem jest Berners-Lee, rozpoczyna projekt badawczy w tym kierunku i w 2004 ogłasza, że *Semantic Web* jest gotowa do wykorzystania:

> *"Celem inicjatywy* Semantic Web *jest stworzenie uniwersalnego medium do wymiany informacji, gdzie dane mają być współdzielone i przetwarzane przez zautomatyzowane narzędzia w sposób równie skuteczny, jak to czynią ludzie. W skali sieci, jutrzejsze programy musza być zdolne do współdzielenia i przetwarzania danych, nawet jeśli zostały zaprojektowane zupełnie niezależnie. Semantyczna Sieć WWW zgrabnie połączy zarządzanie informacją osobistą, integrację aplikacji biznesowych oraz globalną wymianę da-*

[1] Przykłady przywołuję za (Krukowski 2005; Nowak 2004).

[2] Ang. *Semantic Web*; inna często spotykane tłumaczenie to *Semantyczny Internet*.

> *nych: komercyjnych, naukowych i kulturowych. Dla organizacji, jednostek i społeczności możliwość umieszczania w sieci informacji rozumianych przez maszyny staje się sprawą priorytetową."*[3]

W idealnym świecie *Semantycznej Sieci WWW* scenariusz zaplanowania podróży do Bangkoku byłby zdecydowanie inny. Trzeba byłoby uruchomić wyszukiwarkę semantyczną i wpisać w oknie dialogowym zapytanie w języku naturalnym: *„Znajdź środek transportu do Bangkoku, zakup na niego bilet, zarezerwuj..."*. Wszystkim zajęłaby się potężna maszyneria. Zachwycony tą wizją Ibaraki, przewodniczący *iGen Knowledge Solutions, Inc.*, konstatuje:

> *„Mamy do czynienia z rewolucją, rewolucją czyniącą sieć znaczącą, zrozumiałą i przetwarzalną przez maszyny, niezależnie od tego, czy jest oparta na intranecie, ekstranecie czy też internecie.* Semantic Web*, bo o niej mowa, poprowadzi nas w stronę spojrzenia na wiedzę jako oś «wszystkiego»."*(Daconta *et al.* 2003)

Z drugiej strony powstały przede wszystkim specyfikacje, którym sceptycy zarzucają nadmierny poziom skomplikowania i słaby formalizm. Narzędzia rozwijane w ich ramach są niedojrzałe, a każde z nich operuje specyficznym dla siebie API i językiem zapytań. Nie wydaje się też, aby Semantyczna Sieć WWW miała zastąpić dotychczasową sieć WWW. Nie mniej jednak jej potencjał został dostrzeżony przez Unię Europejską, która dofinansowywała program SWAP-Europe[4]. Jego rezultatem stało się m.in. stworzenie silniejszych specyfikacji, bardziej uniwersalnych narzędzi i zaprezentowanie licznych aplikacji z użyciem Semantycznej Sieci WWW.

## 2.1 Konstrukcja Semantycznej Sieci WWW

Semantyczna Sieć WWW to, tak naprawdę, sieć WWW wzbogacona o semantykę – dane zrozumiałe i przetwarzalne przez maszynę. Sieć ta stanowi przestrzeń informacyjną, w której zasoby identyfikowane są przez referencje URI (*Uniform Resource Identifier*[5], *Ujednolicony Identyfikator Zasobu*[6]). Dzięki rozproszeniu może rozwijać się podobnie jak hipertekst we wczesnym stadium WWW. W chwili, gdy do zasobu (witryny WWW, książki, osoby czy czegokolwiek innego) zostanie przyporządkowana referencja URI, będą mogły odwoływać się do niego inne osoby, tworząc w ten sposób skomplikowaną sieć zależności.

Na rysunku 2.1 prezentowana jest architektura *Semantic Web*, zaproponowana przez W3C. Najniższa warstwa, obejmująca URI i Unicode, pozwala na identyfikowanie zasobów, stworzenie podstawy przestrzeni informacyjnej. Powyżej znajduje się warstwa dostarczająca wspólną składnię (XML, przestrzenie nazw[7] i XML Schema). Kolejne dwie warstwy: RDF i RDF Schema oraz *ontologie*[8] pozwalają na opisywanie relacji

[3] *Semantic Web Activity Statement*: `http://www.w3.org/2001/sw/Activity`.

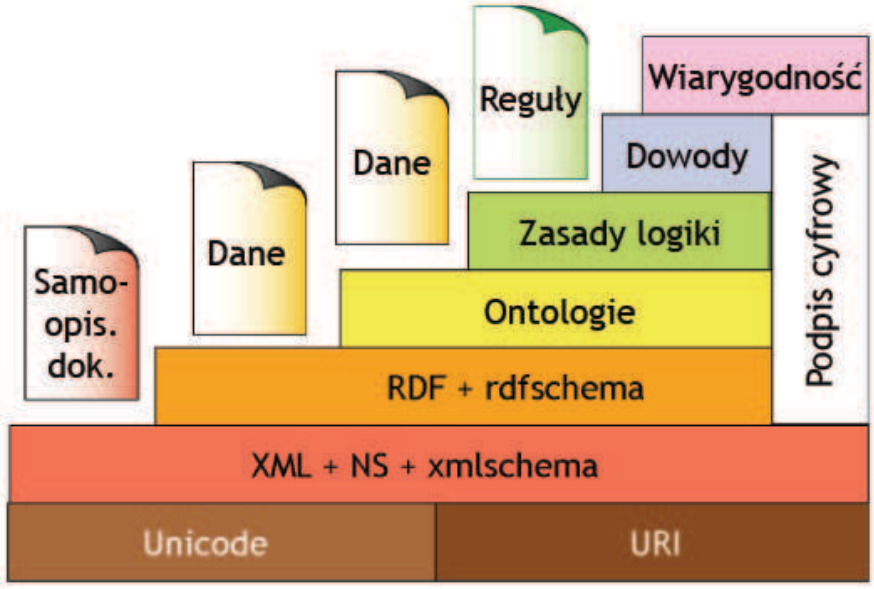
[4] *Semantic Web Standardization in Europe (Brickley* et al. *2004)*

[5] Dawniej także *Universal Resource Identifier*, czyli *Uniwersalny Identyfikator Zasobu*.

[6] Polskie nazwy dotyczące *Semantycznej Sieci WWW* podaję za (Harold 2000; North i Hermans 2000; Sokulski 2002).

[7] Ang. *namespaces* (NS).

[8] Ang. *ontology vocabulary*.




Rysunek 2.1: Architektura *Semantycznej Sieci WWW* (opracowano na podstawie *Semantic Web – XML2000 – slide «Architecture»*: `http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html`).

między obiektami i stanowią trzon wymienialności danych. RDF i RDF Schema i stanowią zbiór reguł, dzięki którym *ontologie* mogą łączyć wiedzę pochodzącą z różnych światów i na jej podstawie dokonywać wnioskowania. Zasady logiki[9], podane w wyższej warstwie, określają wspólny język wnioskowania, a dowody[10] – kroki potrzebne do osiągnięcia danego wniosku.

Cztery ostatnio przedstawione warstwy wspierane są przez podpis elektroniczny[11]. Stanowi on mechanizm pozwalający sprawdzić jak bardzo wiarygodna jest dana informacja; zweryfikować, kto jest autorem danego stwierdzenia w ontologii lub dostarczonego dowodu.

W odróżnieniu do WWW, której naczelną zasadą jest: *„Ktokolwiek może powiedzieć cokolwiek na temat czegokolwiek”*, *Semantyczna Sieć WWW* ma stanowić system, gwarantujący jakość informacji i wniosków. Dlatego Semantyczna Sieć WWW zawiera warstwę zaufania (wiarygodności)[12], która ma stanowić zbiór mechanizmów gwarantujących wiarygodność wyciągniętych wniosków.

Wróćmy do warstwy ontologii. To na jej poziomie przenoszona będzie prawdziwa informacja. Nią wymieniać się mają *agenci*, których rozwój – jak przewiduje Tim Berners-Lee – ujawni prawdziwą siłę Semantycznej Sieci WWW. Agenci to programy zdolne do zbierania informacji z odmiennych źródeł, przetwarzania jej i wymieniania się nią między sobą. Do całej maszynerii użytkownik ma mieć dostęp poprzez interfejs WWW, który – jak się wydaje – pozostaje najważniejszym sposobem dostępu do sieci.

[9] Ang. *logic*.
[10] Ang. *proof*.
[11] Ang. *digital signature*.
[12] Ang. *trust*.

Z punktu widzenia projektanta to właśnie te trzy elementy: ontologia, agenci i interfejs WWW określają metodologię projektowania aplikacji w środowisku Semantycznej Sieci WWW.

## 2.2 Ontologie i ich rola

Termin *ontologia* wywodzi się z filozofii, gdzie definiowany jest jako dziedzina zajmująca się odpowiedzią na zasadnicze pytania: (a) co istnieje? oraz (b) jeśli to, co istnieje, można rozdzielić na części, to czym są te składniki i jakie są relacje między nimi? W kontekście informatycznym odpowiedzią na te pytania może być wiedza oparta na konceptualizacji (a) obiektów, konceptów i innych jednostek, dla których zakłada się, że istnieją w określonym obszarze zainteresowania oraz (b) relacji między nimi (Genesereth i Nilsson 1986). Ta konceptualizacja jest abstrakcyjnym i uproszczonym obrazem świata, który chcemy modelować (reprezentować) w określonym celu. W tym sensie, każda baza informacji, system oparty na informacji czy też agent informacyjny wpisuje się w pewną konceptualizację, wprost lub nie-wprost[13]. Tą konceptualizację, jeśli jest określona wprost, nazwiemy *ontologią*[14].

W samej informatyce termin *ontologia* pojawił się już w 1967 roku, w temacie modelowania danych, ale dopiero w dobie *Semantic Web* zyskał szersze zainteresowanie[15]. *Ontologie* są często przyrównywane do taksonomicznych hierarchii klas, definicji klas i relacji zawierania. Jednak, hierarchie te są jedynie podzbiorem o wiele szerszej klasy jednostek konceptualnych. Nawet więcej, *ontologie* nie są ograniczone do definicji w tradycyjnym sensie logiki, to jest wprowadzającym tylko terminologię pozbawioną wiedzy o świecie [5]. Definicje terminów w ontologii pozwalają na określenie, w jaki sposób mają być one interpretowane i poprawnie używane. Widać tu więc pewne podobieństwo do schematów baz danych. Należy jednak pamiętać, że *ontologia* dostarcza przede wszystkim teorii na temat pewnej dziedziny, a nie struktury, w jakich dane mają być przechowywane. Zatem w stosunku do schematów baz danych różnice dotyczą (Fensel 2003):

- *języka* do definiowania *ontologii*, który jest syntaktycznie i semantycznie bogatszy od języka opisującego schematy baz danych,
- *struktury* informacji określonej przez *ontologię*; ontologia jest tylko częściowo ustrukturalizowanym zapisem w języku naturalnym i nie ma charakteru tabeli,
- *terminologii*, która musi być powszechnie stosowana ze względu na wymienialność informacji.

Dla dalszych rozważań musimy ustalić spójną terminologię. *Konceptem* nazywać będziemy każdy obiekt w ontologii, niezależnie od tego czy jest on ogólną klasą obiek-

[13] Gruber, Tom. 1993. *What is an Ontology?*: `http://www-ksl.stanford.edu/kst/what-is-an-ontology.html`.

[14] W literaturze przedmiotu podawane są także inne definicje *ontologii*. Dyskusja na temat ich zasadności została przedstawiona w (Bassara 2004).

[15] *Wikipedia: Ontologia. Kontekst informatyczny (w oparciu o sieci semantyczne)*: `http://pl.wikipedia.org/wiki/Ontologia`.

tów (np. koncept *Restauracji*), czy też konkretną instancją tej klasy (np. koncept *Restauracja „Verona”*). W języku do reprezentowania ontologii koncept utożsamiamy z *zasobem*[16]. Świat opisujemy za pomocą zdań zbudowanych z konceptów, np.

*Restauracja „Verona” serwuje włoską kuchnię.*

### 2.2.1 Język reprezentowania ontologii – RDF

Przedstawione powyżej zdanie chcielibyśmy zapisać w bardziej sformalizowany sposób, na przykład w postaci logicznej relacji wiążącej podmiot z dopełnieniem za pomocą funkcji orzeczenia:

$$serwuje(restauracjaVerona, wloskaKuchnia)$$

W odpowiedzi *Semantic Web* dostarcza technologię RDF[17], która pozwala zapisywać zdania (*stwierdzenia*[19]) w postaci *trójki* [20]:

$$Triple(RestauracjaVerona, serwuje, WloskaKuchnia)$$

Każda trójka stanowi kompletny i unikalny fakt.

Wszystkie elementy stwierdzenia są zasobami identyfikowanymi przez referencję URI. Na potrzeby przykładu załóżmy, że URI *podmiotu* to:

```
<http://gastronomia.pl\#RestuaracjaVerona> .
```

Niech pozostałe elementy zdania funkcjonują w tej samej *przestrzeni nazw*:

```
http://gastronomia.pl\#
```

Zatem

```
<http://gastronomia.pl#RestauracjaVerona>
  <http://gastronomia.pl#serwuje>
          <http://gastronomia.pl#WloskaKuchnia> .
```

Specyfikacja RDF nie zobowiązuje projektanta, aby wskazane URI odwoływało się do faktycznie istniejącego adresu. Ale dzięki unikalności URI, do tego samego *podmiotu*, *orzeczenia* i *dopełnienia* można odwoływać się w innych stwierdzeniach. Dodamy zatem kilka dodatkowych faktów do ontologii:

```
<http://gastronomia.pl#RestauracjaVerona>
  <http://gastronomia.pl#serwuje>
     <http://gastronomia.pl#WloskaKuchnia> .
<http://gastronomia.pl#RestauracjaVerona>
  <http://gastronomia.pl#serwuje>
     <http://gastronomia.pl#Piwo> .
<http://gastronomia.pl#RestauracjaVerona>
```

[16] Ang. *resource*.

[17] *Resource Description Framework*, czyli *Szkielet Opisu Zasobów*; rekomendowana przez W3C[18]. W polskiej literaturze występują też tłumaczenia: *Ramowy Opis Zasobów* (North i Hermans 2000) i *Metoda Opisu Zasobów* (Harold 2000), od poprzedniego angielskiego rozwinięcia skrótu *Resource Description Format*.

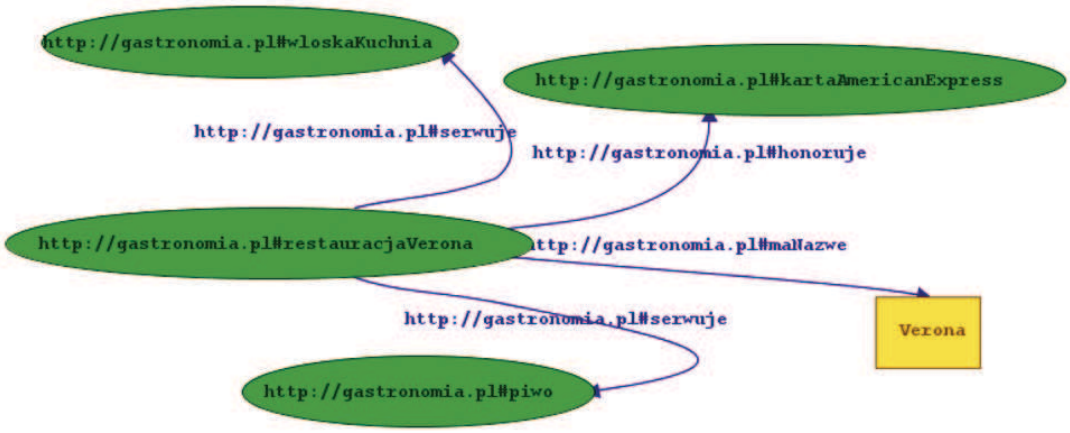
[19] Ang. *statement*.

[20] Ang. *triple*.

```
  <http://gastronomia.pl#honoruje>
      <http://gastronomia.pl#KartaAmericanExpress> .
http://gastronomia.pl#RestauracjaVerona>
  <http://gastronomia.pl#maNazwe>
      "Verona" .
```

Wiemy zatem, że przykładowa restauracja serwuje także piwo i honoruje karty American Express. W ostatnim stwierdzeniu *dopełnienie* nie jest *zasobem*, ale – w drodze wyjątku dopuszczanego przez RDF – *literałem*[21].

Poczyńmy jeszcze jedną obserwację. Przedstawiona powyżej notacja jest nazywana N3 i jest, obok XML, jednym ze sposobów zapisu (serializacji) RDF[22]. Często mówimy, że serializujemy graf RDF, np. do postaci RDF/XML. Graf jest kolejnym przykładem wizualizacji RDF. Na rysunku 2.3 prezentujemy reprezentację ontologii o restauracji „Verona" w postaci grafu. Sposób interepretacji określa rysunek 2.2.




Rysunek 2.2: Przykładowy graf RDF (przygotowane w IsaViz).

### 2.2.2 Jak przedstawiać informację o informacji – RDF Schema

Za pomocą RDF przedstawiłem opis konkretnej instancji klasy restauracji. Restauracja jest *klasą*[23], *serwuje*, *honoruje* i *maNazwę* są *właściwościami*[24] o określonych ograniczeniach, a np. sama *WloskaKuchnia* jest wartością, która mieści się w ramach tych ograniczeń. Formalizmem, który pozwala określić klasy, ich właściwości i ograniczenia, jest rozszerzenie RDF zwane RDF Schema (RDFS)[25]. Można z dużym przybliżeniem powiedzieć, że RDF pozwala na określenie relacji na poziomie instancji, natomiast

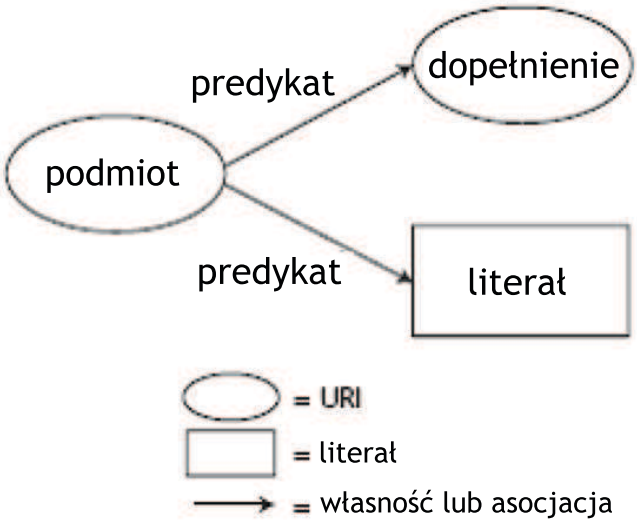
[21] Ang. *literal*, ciąg znaków.

[22] Notacja N3 jest polecana przez Tima Berners-Lee (*Notation 3 – Ideas about Web architecture*: `http://www.w3.org/DesignIssues/Notation3.html`) przy projektowaniu ontologii, ale to RDF/XML i jego skrócona postać (RDF/XML-ABBREV) jest oficjalnie rekomendowana przez W3C (*RDF/XML Syntax Specification*: `http://www.w3.org/TR/rdf-syntax-grammar`). W tej pracy będziemy posługiwać się notacją N3, która jest prostsza w czytaniu i bardziej zwarta.

[23] Ang. *class*

[24] Ang. *property*.

[25] Szczegółowa specyfikacja dostępna jest pod *RDF Vocabulary Description Language 1.0: RDF Schema*: `http://www.w3.org/TR/rdf-schema/`.




Rysunek 2.3: Idea konstrukcji grafu RDF (opracowano na podstawie (Daconta *et al.* 2003)).

RDFS – na poziomie klas (Daconta *et al.* 2003).

Definiowanie relacji między klasami można rozumieć jako modelowanie typów danych. Model danych proponowany przez RDFS jest tym samym modelem, który używany jest w programowaniu obiektowym[26]. Przypomnijmy zatem, że możliwe jest definiowanie klas. Klasę rozumiemy jako grupę obiektów o wspólnej charakterystyce. Obiektami nazywamy instancje klas. Na charakterystykę składają się pola danych, w RDFS nazywane *właściwościami* (nie definiuje się metod). Analogicznie, możliwe jest dziedziczenie właściwości przez klasy potomne.

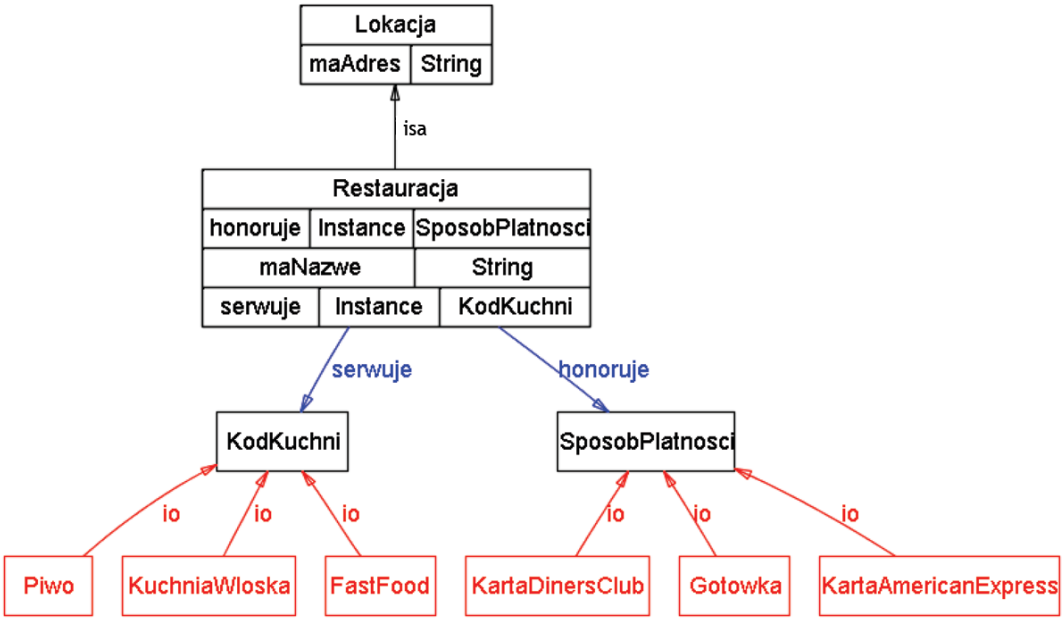
Hierarchię klas w RDFS można przedstawić graficznie, podobnie jak to się czyni w programowaniu obiektowym przy użyciu diagramów UML. Na rysunku 2.4 pokazany jest diagram dla ontologii opisującej restauracje.

*Restauracja* jest klasą z określonymi wprost właściwościami, np. pole *serwuje* mówi o typie serwowanej kuchni. Właściwość ta może przyjmować wartości w zakresie określonym przez klasę *KodKuchnii*, a ściślej przez predefiniowane instancje tej klasy: *Piwo*, *KuchniaWloska* i *FastFood*. Klasa *Restuaracja* dziedziczy z klasy rodzica, *Lokacji* właściwość *maAdres* o dopuszczalnych wartościach typu *String*.

Jak już wspomnieliśmy, RDFS jest rozszerzeniem gramatyki RDF o nowe pojęcia, pozwalającym określać zależności na poziomie klas. Trzy najważniejsze to: *zasób* (rdf:Resource), *właściwość* (rdf:Property) i *klasa* (rdfs:Class). W rzeczywistości wszystkie one są klasami, co możemy zapisać w notacji N3 jako:

```
rdf:Resource   rdf:type  rdfs:Class  .
rdf:Property   rdf:type  rdfs:Class  .
rdf:Class      rdf:type  rdfs:Class  .
```

[26] Ang. *object-oriented programming* (OOP).




Rysunek 2.4: Przykładowy diagram ontologii: *io* jest skrótem od angielskiego *instance of* – instancja czego; *isa* jest skrótem od angielskiego *is a(n)*, czyli *jest czymś*, jak w zdaniu: *„Restauracja jest Lokacją”* (przygotowane w Protégé).

gdzie rdf:type oznacza, że *coś posiada określony typ*. Podobnie możemy zdefiniować własną klasę:

```
:Lokacja rdf:type rdfs:Class .
```

RDFS pozwala na określanie relacji między klasami, takich jak:

- bycie podklasą innej klasy (rdfs:subClassOf), np.:

  ```
  :Restauracja rdfs:subClassOf :Lokacja .
  ```

- bycie właściwością innej klasy (rdfs:domain), a ściślej przynależenie do jej domeny, np.:

  ```
  :serwuje rdf:type   rdf:Property .
  :serwuje rdf:domain :Restauracja .
  ```

- ograniczenie dopuszczalnych wartości (rdfs:range) za pomocą instancji innej klasy, np.:

  ```
  :serwuje          rdfs:range :KodKuchni .
  :KuchniaWloska    rdf:type   :KodKuchni .
  :Piwo             rdf:type   :KodKuchni .
  :FastFood         rdf:type   :KodKuchni .
  ```

- inne.

W RDFS definiujemy właściwości i określamy do jakiej klasy przynależą. Inaczej jest w programowaniu obiektowym, gdzie klasę definiujemy od razu ze wszystkim, co w niej się zawiera. Jest to kluczowa różnica w modelowaniu klas: w RDFS mamy do

czynienia z podejściem *bottom-up*, a w OOP – *top-down*. To właśnie ta cecha decyduje o możliwości stałego rozwijania ontologii, bez niszczenia istniejącej struktury[27].

### 2.2.3 T-Box i A-Box

Uporządkujmy teraz naszą wiedzę. Za pomocą RDFS i RDF potrafimy podać zarówno definicję Restauracji (typowe cechy, właściwości którymi może się charakteryzować), jak i opis jej konkretnej realizacji (na przykład opis restauracji-pubu „13 Kotów" w Poznaniu, która serwuje takie a takie piwo). Zbiór definicji konceptów oraz ich ograniczeń będziemy nazywać *wiedzą terminologiczną* (T-Box[28]), natomiast jest realizacją, opis konkretnych instancji – *wiedzą faktyczną* (A-Box[29]) (Horrocks i Sattler 2002).

### 2.2.4 Jak przeszukiwać informacje – RDQL

Choć *Semantic Web* jest ciągle w fazie rozwoju, to wiele portali udostępnia już swoje dane w postaci RDF. Jednym z nich jest *Chefmoz dining guide*[30], katalog 260 tysięcy restauracji, barów i innych lokali gastronomicznych. Aby przeszukiwać tak ogromne zasoby danych w postaci RDF, potrzebny był odpowiedni mechanizm. Różni producenci proponowali różne języki zapytań. Dopiero niedawno, w wyniku prac prowadzonych przez labolatorium Hewlett Packard, powstał język zapytań dla RDF – RDQL[31]. Język uzyskał wsparcie W3C i został zaimplementowany w wielu systemach wyciągających informacje z grafów RDF[32].

Składnię języka prześledzimy na krótkim przykładzie, który pozwali nam wyłuskać informacje z katalogu podobnego do wspomnianego serwisu *Chefmoz*. Załóżmy, że poszukujemy adresu restauracji w Poznaniu, która serwuje kuchnię włoską, honoruje karty American Express i jest wysoko oceniana przez innych (wysoka nota w skali 1-10). Poszukiwana restauracja określona jest zatem przez trzy fakty (stwierdzenia):

```
SELECT ?adres
WHERE (?rest , <gas:serwuje>, <gas:WloskaKuchnia),
      (?rest , <gas:honoruje>, <gas:AmericanExpress),
      (?rest , <gas:oceniana>, ?ocena),
      (?rest , <gas:maAdres>, ?adres)
AND
      ?ocena > 6
USING gas FOR <http://gastronomia.pl#>
```

Zmienna ?rest wiąże wszystkie ze sobą te zasoby, które spełniają warunki stwierdzeń. Jedyną wolną zmienną jest ?adres i to ona jest zwracana w wyniku zapytania.

[27] Idea RDF i RDF Schema jest prosta, natomiast dokładna specyfikacja jest dość skomplikowana. Dobre wprowadzenie do tematu znajduje się w *RDF Primer*: `http://www.w3.org/TR/rdf-primer`. Powers uczy krok po kroku w (Powers 2003), natomiast w (Davies *et al.* 2003) znajduje się dobre opracowanie aplikacji RDF do tworzenia ontologii w *Semantic Web*. Z polskiej literatury dostępne są krótkie opracowania w czasopismach (Sokulski 2002; Krukowski 2005) i skromne wzmianki w książkach (North i Hermans 2000; Harold 2000).

[28] Ang. *terminological knowledge*.

[29] Ang. *assertional knowledge*.

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

[31] *RDQL - A Query Language for RDF*: `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/`.

[32] Np. program Jena, o którym mówimy dalej, przy okazji omawiania narzędzi.

### 2.2.5 Wnioskowanie na podstawie ontologii

Ontologia określa również pewne *reguły wnioskowania*[33]. Mogą być one wyrażone wprost lub mieć charakter „strukturalny" (Hendler 1999). Na podstawie przedstawionego wyżej przykładu, możemy przeprowadzić rozumowanie, że skoro *Restauracja* jest klasą pochodną *Lokacji* i *Lokacja* posiada jakiś adres (właściwość *maAdres*), to *Restauracja* również musi go posiadać. Powyższy wniosek możemy wyabstrahować do ogólnej zasady, że jeśli *R1* jest klasą i ma właściwość *P*, a *R2* jest klasą pochodną klasy *R1*, to *R2* również ma właściwość *P*.

### 2.2.6 Praca z ontologiami – narzędzia

Aby wykorzystać siłę ontologii w praktyce, aby tworzyć i przetwarzać je efektywnie, potrzebne są narzędzia wspierające *Semantic Web*[34]. Takie narzędzia powinny pozwalać na:

- tworzenie grafów RDF(S) za pomocą GUI techniką WYSIWYG[35],
- wizualizację grafów RDF(S),
- operowaniem na plikach RDF(S): (de)serializacja i konwersja między różnymi notacjami (RDF/XML, RDF/N3),
- zapytywanie grafów RDF(S)
- przeprowadzanie wnioskowania w grafach RDF(S),
- dostęp do danych w formacie RDF(S) z poziomu popularnych języków programowania.

Poniżej przedstawiam narzędzia łączące w sobie powyższe funkcje. Są to projekty typu *Open Source*[36], w związku z czym posiadają bogatą i szczegółową dokumentację implementacji.

#### Close World Machine

Liczną grupę tworzą *systemy eksperckie*, które pozwalają na przeprowadzanie wnioskowania w określonej dziedzinie[37]. Jednym z nich jest CWM (*Close World Machine*)[38], napisany w języku Python. Wydaje się, że jest to pierwszy i najbardziej uniwersalne narzędzie wśród systemów eksperckich *Semantic Web*, co nie dziwi, zważywszy na fakt, że system ten został stworzony przez W3C. Aplikacja pozwala na konwersję między różnymi notacjami RDF i przeprowadzanie różnego typu wnioskowań. Wbudowany silnik umożliwia sprawdzenie poprawności syntaktycznej RDF, brakuje natomiast walidacji RDFS.

[33] Ang. *inference rules*.

[34] Ang. *Semantic Web enabled*, SWESE. 2005. Workshop on Semantic Web Enabled Software Engineering (SWESE). *Z: 4th International Semantic Web Conference (ISWC 2005)*.

[35] Ang. *What You See Is What You Get* (co znaczy dosłownie *To Co Widzisz Jest Tym Co Otrzymasz*) to akronim stosowany dla określenia metod, które pozwalają uzyskać wynik w wydruku (zapisie do pliku) identyczny lub bardzo zbliżony do obrazu na ekranie.

[36] *Wolne Oprogramowanie*.

[37] Stąd często określane są mianem: *reasoning*, *rule-based* czy *knowledge-based*. Szczegółową analizę tego typu aplikacji czytelnik znajdzie w (Baczyńska i Kumanowska 2004).

[38] *FIPA Specifications*: `http://www.fipa.org/specifications/index.html`.

### Jena

Programista Javy może skorzystać z systemu JENA[39], platformy do tworzenia aplikacji dla *Sieci Semantycznej*. Aplikacja wyszła z labolatorium Hewlett Packard i stąd też używa wspomnianego języka zapytań RDQL. Dobrze zaplanowany interfejs pozwala na szybką integrację danych semantycznych z aplikacją. JENA pozwala na parsowanie i zapisywanie we wszystkich formatach RDF, zawiera kilka mechanizmów wnioskowania. Znalazł zastosowanie między innymi w projekcie *E-Person*[40], gdzie *e-osoba* (Ang. *e-person.*), czyli agent reprezentujący osobę w sieci, dzieli pewne informacje w systemie peer-to-peer z innymi agentami. Dane przechowywane są w formacie RDF.

### IsaViz

W labolarium W3C powstał także ISAVIZ[41], program do przeglądania i tworzenia modeli RDF reprezentowanych przez graf. Aplikacja posiada interfejs pozwalający na płynną nawigację w grafie i edycję modeli techniką WYSIWYG (rysowanie łuków, elips i prostokątów symbolizujących elementy RDF, patrz rysunek 2.2). Pozwala również importować z i eksportować do plików w różnych formatach RDF, a także renderować grafy RDF do plików SVG i PNG.

### Protége

Narzędziem łączącym funkcje edytora graficznego i systemu ekspertowego[42] jest PROTÉGÉ[43], aplikacja rozwinięta na uniwersytecie w Stanford w USA. Aplikacja pozwala na tworzenie ontologii za pomocą kolejnych zakładek: klasy, właściwości[44] i instancje. Na przejrzystość i łatwość obsługi wpływa fakt, że tworzone klasy prezentowane są od razu w formie hierarchii. Siłą PROTÉGÉ są wtyczki pisane przez użytkowników z całego świata. Jedną z nich jest ONTOVIZ TAB, pozwalająca na renderowanie w postaci diagramu hierarchii klas i właściwości (patrz rysunek 2.4).

## 2.3 Systemy wieloagentowe i ich rola

> *„Biały Dom oświadczył, że zwiększone wydatki na technologię [...] mogą być spożytkowane dla przykładu, na tworzenie «inteligentnych agentów», chodzących po Internecie w celu zbierania informacji"* (APNews 1999).

Jak już wspomniałem, istotny element Semantic Web stanowi technologia agentów programowych. Idea samego agenta funkcjonowała już w latach '80, jednak dopiero w 1994 nastąpił przełom, kiedy to pojawiła się wizjonerska praca, opisująca agentów jako „rewolucję w rozwoju oprogramowania" (Ovum 1994). Według Jenningsa (Jennings 1999) ich rola jest dwoista. Z jednej strony stanowią nowe podejście przy projektowaniu

[39] *Jena – A Semantic Web Framework for Java*: `http://jena.sourceforge.net/`.
[40] *e-Person - Personal Information Infrastructure*: `http://www.hpl.hp.com/semweb/e-person.htm`. 8 stycznia 2004.
[41] *IsaViz: A Visual Authoring Tool for RDF*: `http://www.w3.org/2001/11/IsaViz/`.
[42] Więcej informacji na temat narzędzi do tworzenia ontologii czytelnik znajdzie w (Wlekły 2004).
[43] *Protégé*: `http://protege.stanford.edu/`.
[44] W aplikacji nazywane: *slots*.

skompliwanych systemów rozproszonych, z drugiej – spoiwo łączące rozmaite dyscypliny Sztucznej Inteligencji. W kontekście Semantic Web oznacza to możliwość tworzenia inteligentnych jednostek zdolnych do współpracy i przetwarzania informacji. Pierwszym krokiem do zrozumienia tej dwoistości będzie zdefiniowanie pojęcia agenta.

### 2.3.1 Czym jest agent programowy?

Wooldridge (Wooldridge 1997) przytacza następującą definicję[45]:

> *„Agent stanowi skapsułkowany system komputerowy, rezydujący w pewnym środowisku i zdolny do elastycznego, autonomicznego podejmowania akcji w tym środowisku tak, aby sprostać założonym celom."*

Rozwijając definicję, agenci stanowią *identyfikowalne jednostki* o jasno wytyczonych granicach i interfejsach. Posiadają *zdolności społeczne*: otrzymują informacje ze środowiska, w którym przebywają (w tym od innych agentów) i na podstawie tych informacji oddziaływują na środowisko. Każdy agent posiada określone cele, do których osiągnięcia dąży. Jest *autonomiczny*: co prawda, informacje ze środowiska wprowadzają agenta w określony stan, ale to on posiada nad nim kontrolę i nad zachowaniem, jakie podejmie w tym stanie. Te trzy elementy: środowisko, celowość działania i autonomiczność wskazują na kolejne cechy: agent musi być zarówno *reaktywny* (reagować na zmiany w środowisku), jak i *proaktywny* (adekwatnie do tych zmian przyjmować nowe cele)[46].

### 2.3.2 Skąd inteligencja w agencie?

Przedstawiony powyżej model działania agenta jest możliwy w sytuacji, gdy posiada on umiejętność *wnioskowania*[47] (Galant i Paprzycki 2005). W literaturze cecha ta jest rozumiana bardzo szeroko. Agent winien posiadać pewien stopień inteligencji, która przejawiać ma się wykorzystaniem bazy wiedzy, zdolnościami wnioskowania i uczenia, czyli poszerzania i aktualizowania nabytej wiedzy. Proces uczenia ma sens jedynie wtedy, jeśli zbliża go do ustalonego celu. Stąd w literaturze pojawiło się pojęcie *inteligentnych agentów programowych*[48].

W praktyce projektowanie i implementowanie inteligencji agenta zależy przede wszystkim do problemu, do którego jest on wykorzystywany. Wróćmy do przykładu organizacji podróży do Bangkoku, przedstawionego na początku tego rozdziału. Gdyby zadaniem tym miał zająć się rzeczywisty agent z biura turystycznego, chcielibyśmy przede wszystkim, aby był *komunikatywny*, tzn. potrafił rozmawiać z klientem na temat podróży i w tym samym języku. Po drugie agent musi być *kompetentny*[49], czyli zdolny do wykonania powierzonego mu zadania i to w szerokim tego słowa znaczeniu, czyli ma nie tylko znaleźć połączenia do Bangkoku, ale także zarezerwować bilety na lot, wynająć hotel

[45] W literaturze przedmiotu możemy znaleźć szereg innych definicji. Dyskusja nad ich zasadnością pojawia się (Paprzycki 2003).

[46] Wooldridge, Michael i Jennings, Nicholas R. 1995. *Intelligent Agents: Theory and Practice.* `http://www.doc.mmu.ac.uk/STAFF/mike/ker95/ker95-html` (Hypertext version of Knowledge Engineering Review paper).

[47] Ang. *reasoning.*

[48] Ang. *Intelligent Software Agent.*

[49] Ang. *capable.*

etc. Po trzecie agent musi być *autonomiczny*, to jest zdolny działać bez nadzoru klienta, ale także gotowy do inicjowania dyskusji nad poszczególnymi aspektami podróży. Agent musi również mieć zdolność *adaptacji*[50], uczyć się, jakiego typu hotelów klient lub klienci nie lubią itp. Podobną propozycję przedstawił Hendler (Hendler 1999), żądając od agenta programowego takich samych cech jak od pracownika agencji turystycznej[51]. Przyjrzyjmy im się pokrótce.

### Komunikatywność

Zdolność komunikacji wyraża się nie tylko na poziomie słów i języka, ale także na poziomie wiedzy. Rozmowa stanowi wymianę myśli, ale jest możliwa tylko wtedy, jeśli rozmawiający posiadają podstawową wiedzę w omawianej dziedzinie[52]. W rozwiązaniu tego problemu wykorzystywane są najczęściej ontologie. Wiedza w nich zgromadzona pozwala na spójne i jednoznaczne definiowanie tych samych pojęć, a wykorzystanie zawartych w nich reguł wnioskowania pozwala na znajdowanie rozwiązań, np. za pomocą wspomnianych systemów eksperckich.

Skoro ontologia jest czytelna dla maszyny (patrz RDF), to pozwala agentowi manipulować słowami, które dla użytkownika niosą pewną zrozumiałą informację. Agent nie musi rozumieć tej informacji, w głębokim tego słowa znaczeniu, ale potrafi nią manipulować w taki sposób, że użytkownik będzie w stanie zrozumieć efekt tej manipulacji (rozumowania formalnego).

### Kompetencja

Aby agent był kompetentny, musi być w stanie podejmować działania w środowisku, w którym funkcjonuje. Musi nie tylko doradzać, ale także posiadać zdolność do współpracowania z innymi usługami i źródłami informacji, np. wykupienia biletu (poprzez współpracę z serwisem pozwalającym płacić poprzez podanie numeru karty kredytowej) czy też przetwarzania informacji znajdującej się w Internecie. Wydaje się, że rozwiązania proponowane w ramach Semantycznej Sieci WWW spełniają wszystkie wymagania związane z pojęciem kompetencji agenta: format danych, które ma przetwarzać agent (przykładowo RDF/XML) jest dla niego przyjazny.

### Autonomiczność

Całkowita autonomiczność agenta stoi w opozycji do interfejsu użytkownika, który pozwalają kontrolować zachowanie systemu w sposób bezpośredni. W debacie[53] na temat wyższości jednej z opcji nad drugą Shneiderman twierdzi, że tylko bezpośrednia kontrola daje użytkownikowi poczucie kontroli, kompetencji i przewidywalności. Człowiek cały czas „trzyma rękę na sterze" aplikacji. W tym wypadku jest jednak obciążony

[50] Ang. *adaptivite*.

[51] Hendler używa tutaj określenia *agenta internetowego*, który kontaktuje się z użytkownikiem przez interfejs strony WWW i czerpie wiedzę zgromadzoną w Internecie.

[52] Kwestia języka (zwłaszcza użycia języka naturalnego) jest oddzielnym problemem i nie będzie poruszana w tej pracy.

[53] Maes, Pattie, Miller, Jim i Shneiderman, Ben. 1997. *Intelligent Software Agents vs. User-Controlled Direct Manipulation: A Debate*. `http://www.acm.org/sigchi/chi97/proceedings/panel/jrm.htm`.

obowiązkiem pamiętania swoich preferencji i potrzeb.

W przypadku, gdy ten obowiązek przechodzi na autonomicznego agenta pojawia się problem *zaufania* (*„Czy agent postąpi tak, jak ja bym postąpił?”*) i *kompetencji* (*„Czy agent podejmie właściwą decyzję, co do – na przykład – zakupu biletu w wybranej linii lotniczej?”*). Rozwiązaniem są systemy o mieszanej strukturze. Maes (Maes 1994) proponuje wykorzystanie agenta, który uczy się od użytkownika jakie działania może podejmować samodzielnie, a które wykonywać za jego pozwoleniem.

#### Adaptacyjność

Zdolność agenta do adaptacji polega na szukaniu równowagi między znajdywaniem rozwiązywania problemu a sprawdzaniem granic tego, co użytkownik lubi. Rozwiązaniem jest uczenie się (a) jego upodobań i (b) możliwości środowiska. Przypadek (a) może polegać na budowania modelu użytkownika i wykorzystaniu go do personalizacji zwracanej informacji (patrz rozdział 1.1.2). W przypadku (b) zdolność do adaptacji zachowania polega na reagowaniu na zmiany w środowisku. Przykładowo, jeśli np. witryna z której agent czerpał informację przestaje istnieć, agent powinien przestać ją odwiedzać. Jak zwykle w praktyce, wykorzystana metoda adaptacji zależy od rozważanego problemu i może wykorzystywać algorytmy uczenia maszynowego, reguły wnioskowania, elementy statystyki lub inne.

### 2.3.3 Komunikacja między agentami

Zdolności społeczne agenta decydują o jego umiejętności współpracy z innymi agentami[54]. Agenci osiągają swoje cele komunikując się z innymi agentami w ściśle ustalonym języku, ACL[55]. *Akt komunikacji* stanowi przykład *akcji*, czyli reprezentacji zachowania, które agent może przejawiać. W praktyce akt komunikacji polega na wysłaniu przez agenta *wiadomości ACL* (komunikatu) do drugiego agenta. Sekwencja takich aktów, dokonywanych między dwoma lub więcej agentami, składa się na *konwersację*.

#### Protokoły interakcji

W celu uproszczenia konstrukcji agentów ustalane są *protokoły interakcji*. Protokół taki określa typy możliwych do wysłania wiadomości. Dokładniej, określa dla każdej akcji jej *wstępne warunki wywołania*[56] oraz *efekt racjonalny*[57], stanowiący reprezentację efektu, jakiego agent może spodziewać się w wyniku przeprowadzenia akcji.

Przykładowo, protokół FIPA Request – stanowiący formalny opis obsługi żądania zleconego przez agenta *Inicjatora*[58] agentowi *Partnerowi*[59] – określa co następuje:

[54] Dalszy tekst stanowi przede wszystkim opis specyfikacji FIPA (*FIPA Specifications*: `http://www.fipa.org/specifications/index.html`), określającej zasady projektowania systemów wieloagentowych.

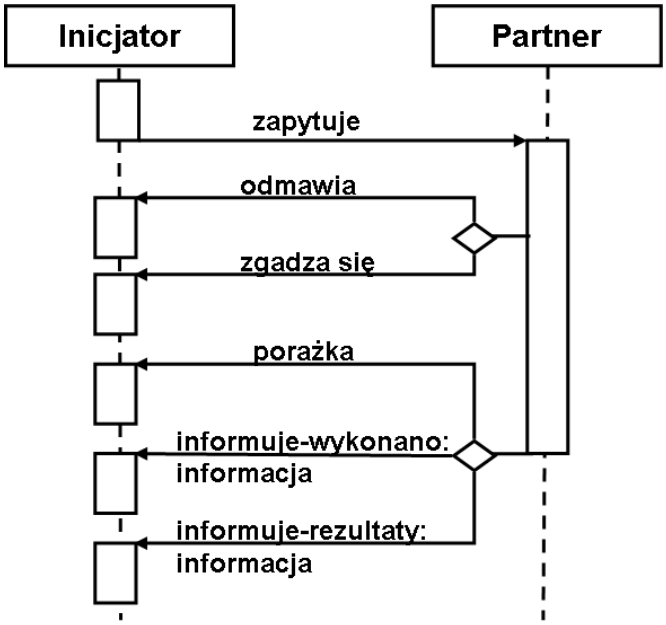
[55] Ang. *Agent Communication Language*, specyfikacja znajduje się pod `http://www.fipa.org/repository/aclspecs.html`.

[56] Ang. *Feasibility Preconditions*.

[57] Ang. *Rational Effect*.

[58] Ang. *Initiator*.

[59] Ang. *Participant*.




Rysunek 2.5: Protokół FIPA Request (opracowane na podstawie *FIPA Request Interaction Protocol Specification*: `http://www.fipa.org/specs/fipa00026/SC00026H.pdf`).

- jako wstępny warunek wywołania ustala typ wiadomości wysyłanej przez Inicjatora jako *żądanie* (*request*)
- a jako efekt racjonalny, czyli spodziewaną odpowiedź od Partnera, *zgodę* (*agree*) lub *odmowę* (*refuse*) wykonania żądania, a po wykonaniu żądania albo szczęśliwie zwraca wynik (*inform-done* lub *inform-result*), albo informaje o porażce (*failure*).

Opis protokołu FIPA Request za pomocą diagramu AUML[60] znajduje się na rysunku 2.5.

#### Kontekst rozmowy

Model komunikacji zakłada, że agenci zaangażowani w rozmowę będą dzielić wspólną ontologię, czyli kontekst[61]. Dzięki temu możliwe jest zawarcie w treści wiadomości ACL komunikatów zrozumiałych dla obu stron. Treść taka zapisywana jest za pomocą jednego z języków: FIPA-SL (najbardziej powszechny), FIPA-CCL, FIPA-RDF oraz FIPA-KIF.

### 2.3.4 Społeczność agentów – systemy wieloagentowe

Dobrym obrazem koncepcji współdziałania agentów jest drużyna piłki nożnej. W zawodach RoboCup programy rozgrywają mecze, w których poszczególni zawodnicy-agenci odbierają sygnały ze środowiska (patrzą, co się dzieje na boisku), wpływają na nie (kopią piłkę, poruszają się), porozumiewają się ze sobą, przewidują swoje działania. Jednocześnie posiadają własne cele (np. bramkarz broni bramki, napastnik strzela gole) oraz cel nadrzędny – wygrać mecz.

[60] Język służący do modelowania zachowań agentów (Agent UML), dokładny opis znajduje się pod *Agent UML Web Site*: `http://www.auml.org/`.

[61] *FIPA Ontology Service Specification*: `http://www.fipa.org/specs/fipa00086/XC00086C.html`.

Grupa autonomicznych agentów, z których każdy dąży do wypełnienia własnych celów lub też współpracuje z innymi agentami w celu osiągnięcia globalnego efektu, stanowi społeczność. Społeczność tą nazywamy *systemem wieloagentowym*[62]. Społeczność agentów rozsiana jest na jednej lub większej liczbie *Platform Agentowych*[63]. Platforma taka zapewnia infrastrukturę, w której agent może być umieszczony. Agent musi być zarejestrowany na platformie, jeżeli chce prowadzić komunikację z agentami na tej i innych platformach. W ramach platformy funkcjonują następujące usługi:

- *Kanał Komunikacyjny Agenta*[64] (KKA), pozwalający na rutowanie komunikatów w obrębie tej samej i innych platform,
- *System Zarządzania Agentem*[65] (SZA)), m. in. zarządzający tworzeniem, usuwaniem, zawieszaniem, przywracaniem, potwierdzaniem i migrowaniem agentów na platformie,
- *Usługę Katalogową*[66] (UK), dostarczająca usługę „żółtych stron" dla agentów.

### 2.3.5 Agentowo zorientowana metodologia programowania

Jak powiedziałem, każdy agent w społeczności dąży do osiągniecia indywidualnego celu i/lub celu wspólnego dla danej społeczności. Dlatego na każdego agenta należy patrzeć z dwóch perspektyw: *wewnętrznej*[67] – definiującej indywidualne cele i *zewnętrznej*[68] – określającej jego rolę w osiągnięciu globalnego efektu. Każda z nich jest odbiciem odpowiedniej cechy agenta: proaktywności lub reaktywności; obie decydują o sposobie konstrukcji agenta. Programowanie obiektowe nie dostarcza żadnej metodologii pozwalającej na spojrzenie z obu perspektyw (Zambonelli *et al.* 2000).

Można jednak analizować dobrze znane techniki rozwiązywania podobnych problemów inżynierii programowania obiektowego i szukać podobieństw z głównymi cechami podejścia agentowego, jak to uczynił Jennings (Jennings 1999). Przytacza on za (Booch 1994) zasadnicze metody projektowania systemów złożonych:

- *dekompozycja*, czyli dzielenie problemu na mniejsze podproblemy, łatwiejsze do zarządzania i możliwe do rozwiązania we względnej izolacji,
- *abstrakcja*, jako proces upraszczania modelu systemu,
- *organizacja*, pozwalająca na określenie i zarządzanie relacjami między wydzielonymi podproblemami.

Złożony system może być postrzegany w formie hierarchii podsystemów. Najniżej w hierarchii znajdują się komponenty wykonujące prymitywne zadania. Z często pojawiających podsystemów wyobstrahowano najczęściej pojawiające się schematy, tzw. *wzorce projektowe*, których użycie upraszcza projektowanie i pozwala uniknąć powtarzanych

[62] Ang. *multi-agent system* (MAS).
[63] Ang. *Agent Platform* (AP).
[64] Ang. *Agent Communication Channel* (ACC).
[65] Ang. *Agent Managing System* (AMS).
[66] Ang. *Directory Facilitator* (DF).
[67] Zwanej także *mikro* lub *intra-agent*.
[68] Zwanej także *makro* lub *inter-agent*.

w przeszłości błędów. *Dekompozycja* prowadzi do takiego podziału na abstrakcyjne podsystemy, z których każdy zajmuje się oddzielnym problemem. W programowaniu agentowym agent doskonale realizuje rolę komponentu wykonującego najprymitywniejsze zadania, a kolektyw agentów – rolę podsystemu.

Idąc dalej, *interakcje* między podsystemami mogą być odzworowane poprzez wymianę wiadomości między agentami, a powiązania i organizacja – poprzez protokoły tworzenia, utrzymywania i niszczenia kolektywów agentów. Poziom abstrakcji, określony przy pod podejściu agentowym jest wyższy niż przy paradygmacie programowania obiektowego, dlatego że zachowanie agentów odpowiada w sposób bliższy ludziom, których pracę mają wykonywać lub wspierać.

Najtrudniejszy, moim zdaniem, jest problem organizacji wielu perspektyw, a zatem wielu punktów kontroli i współzawodniczących zadań. Został on rozwiązany dzięki ustaleniu wspólnego języka ACL: podsystemy (a więc agenci i całe grupy agentów) mogą być projektowane i rozwijane niezależnie na każdym z poziomów hierarchii systemu.

Przy ocenie efektywności w przypadku agentów należy przede wszystkim wziąć pod uwagę koszta komunikacji. Lokowanie każdej funkcji systemu w oddzielnym agencie może doprowadzić do spadku wydajności systemu ze względu na wzrost ilości wymiany informacji (Tusiewicz 2003).

Mimo pewnych związków z programowaniem obiektowym programowanie agentowe musiało doczekać się nowych metodologii projektowania. Jedną z bardziej obiecujących jest metodologia *Prometheus*.

### Metodologia Prometheus

Metodologia *Prometheus* wspomaga tworzenie systemu agentowego w całej rozciągłości: od zaplanowania ogólnych celów systemu, aż po szczegółowe opisanie zachowań agentów. Bazuje na na wieloletnich doświadczeniach ośrodków akademickich (skupionych w organizacji The Agent Oriented Software Group[69]) i praktyce przemysłowej (Padgham i Winikoff 2002; Perepletchikov 2004).

Metodologia wyróżnia trzy fazy projektowania, stanowiące kolejne odmienne poziomy abstrakcji: specyfikacji systemu, projektowania architektury i projektowania szczegółowego (patrz rysunek 2.6). Każda z nich składa się z poszczególnych kroków, które mogą być iterowane wielokrotnie w celu wyelimowania błędów.

*Faza specyfikacji systemu* zaczyna się od zdefiniowania *udziałowców*[70] (rysunek 2.7)[71]. Dla każdego udziałowca definiujemy *przypadki użycia*[72], wraz z możliwym *wejściem*[73]

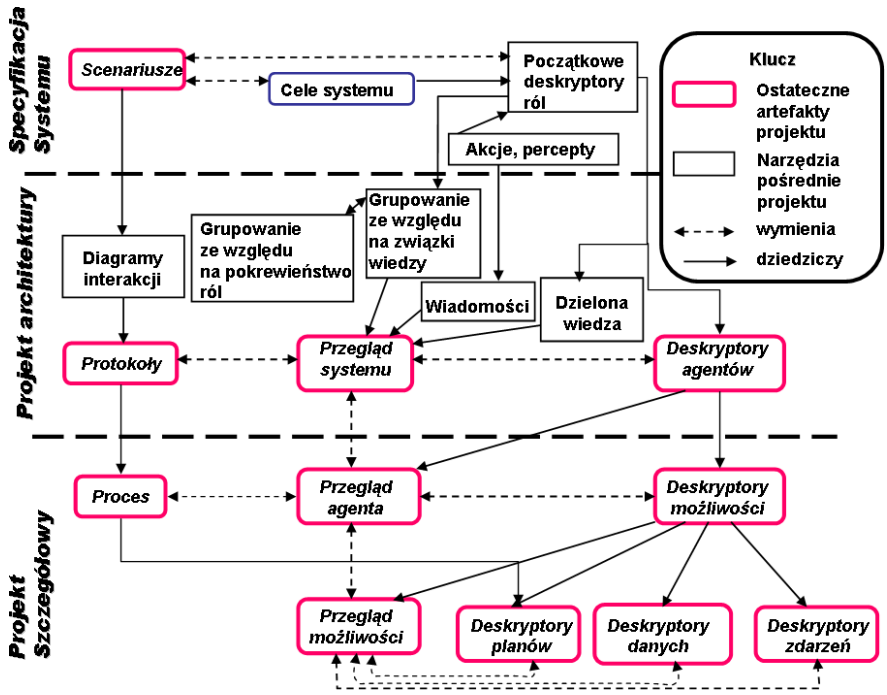
[69] The Agent Oriented Software Group (AOS) jest czołowym międzynarodowym projektantem i dostawcą oprogramowania do tworzenia i rozwijania aplikacji agentowych.

[70] Ang. *stakeholders*, w programowaniu obiektowym określani mianem aktorów.

[71] Faza ta została zmodyfikowana przez autorów w stosunku do pierwowzoru z rysunku 2.6.

[72] Ang. *uses cases*.

[73] Ang. *input*.




Rysunek 2.6: Schemat budowy systemu agentowego w metodologii Prometheus (opracowano na podstawie (Padgham i Winikoff 2002)).

i *wyjściem*[74]. Każdy przypadek użycia wyznacza pewien abstrakcyjny *cel*[75]. Realizacją tego celu zajmują się *scenariusze*[76], których udziałowiec może być inicjatorem lub może być w nie zaangażowany przez system. W scenariuszu rolę wejście pełni *sygnał*[77], a reakcją końcową na niego, czyli ustalonym wyjściem, jest *akcja*[78].

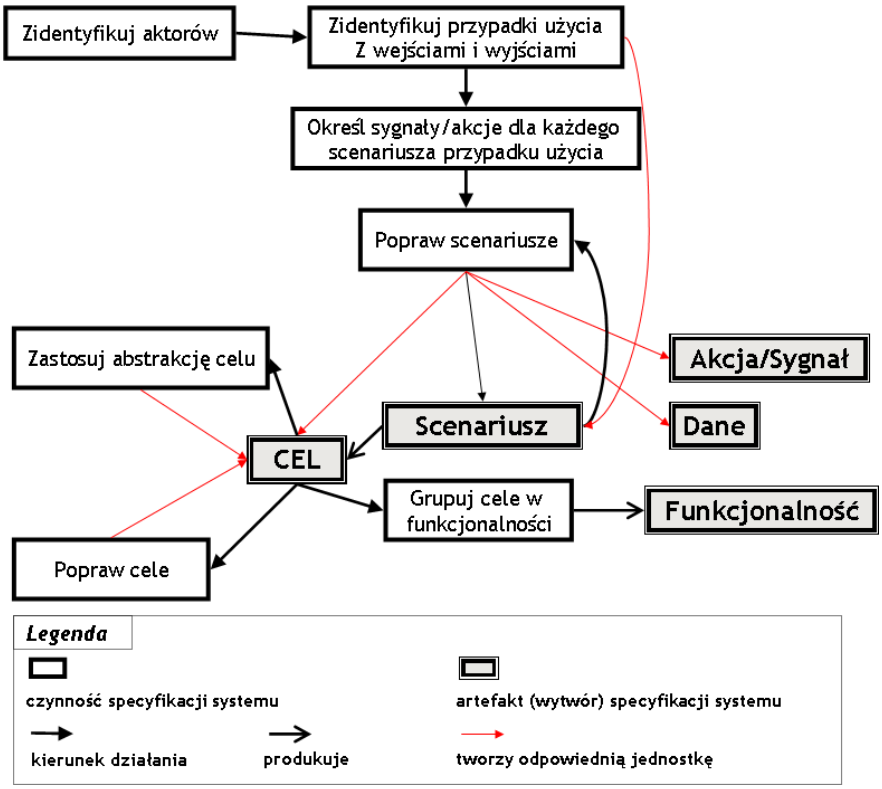
Cele, które system ma osiągać, grupujemy w spójne grupy, czyli *role*, określane często jako funkcjonalności.

W *fazie projektowania architektury* musimy podjąć decyzję, jacy *agenci* powinni działać w systemie. Analizując wnioski z poprzedniej fazy każdemu agentowi przyznajemy role. Role grupujemy, kierując się kryteriami łączenia i spójności, typowymi dla programowania obiektowego. Bierzemy więc pod uwagę, w tak podanej kolejności, następujące czynniki:

- używane dane i wiedzę,
- pokrewieństwo agentów,
- częstotliwość interakcji.

W przypadku danych rozpatrujemy, jakie funkcjonalności, z jakich *danych*[79] korzystają, czy do nich zapisują, czy z nich czytają. Przykładowo, jeśli role zapisujące do

[74] Ang. *output*.
[75] Ang. *goal*.
[76] Ang. *scenarios*.
[77] Ang. *percept*.
[78] Ang. *action*.
[79] Ang. *data*.




Rysunek 2.7: Zmodyfikowana faza specyfikacji systemu w metodologii Prometheus (opracowano na podstawie (Perepletchikov 2004)).

tego samego źródła danych będą przyporządkowane różnym agentom, wymagana będzie synchronizacja dostępu. Należy więc zminimalizować liczbę sytuacji, w który dane są współdzielone.

Następnie ustalamy, który agent na który sygnał powinien reagować jaką akcją. W tym momencie istotne staje się określenie interakcji między agentami. W tym celu wykorzystujemy *diagramy interakcji* zapożyczone z programowania obiektowego. Ustalamy, jakie *wiadomości* wysyłają między sobą oraz jakie są ich cele i zawartość. W tym momencie definiujemy również *protokoły interakcji*, stanowiące sekwencje wysyłanych wiadomości.

Końcową fazą jest *projektowanie szczegółowe*, pozwalająca określić wewnętrzną strukturę agenta i sposób wypełnienia zadania, które mają do osiągnięcia w systemie. Ustalone w fazie projektowania architektury role stają w kontekście agenta jego *kompetencjami*[80]. W ramach każdej kompetencji określamy plany potrzebne do jej wypełnienia. Każdy plan wyzwalany jest w wyniku otrzymania wiadomości lub sygnału, a reakcją na to jest ustalona w wcześniej akcja lub wysłanie wiadomości. Należy również zdefiniować z jakich danych i w jaki sposób plan korzysta.

Niestety, język modelowania używany przez metodologię Prometheus nie został jeszcze ostatecznie udokumentowany, w związku z czym użytkownicy mogą mieć problemy

[80] Ang. *capabilities*.

ze zrozumienia precyzyjnych określeń i prawidłowego używania konstrukcji[81].

Autorzy zapewniają również napisane w Javie narzędzie Prometheus Design Tool (PDT), wspierające tworzenie systemów w metodologii Prometheus[82]. Mimo pewnych niedociągnięć od strony interfejsu oraz drobnych błędów w kodowaniu, aplikacja zdecydowania ułatwia tworzenie systemu. Dodatkowo pozwala sprawdzić spójność zaprojektowanego systemu za pomocą serii testów[83].

### 2.3.6 Jak implementować agentów? – platforma JADE

Do tej pory powstało na świecie ponad 80 środowisk wspierających programowanie agentowe. Większość z nich zaimplementowano w języku Java, pozwalającym uniezależnić się od otoczenia systemowego i sprzętowego. Niestety brak kompatybilności między nimi utrudnia łączenie rozwiązań przygotowanych w każdym z tych środowisk. Kompatybilność byłaby możliwa, gdyby systemy te zbudowane były wokół standardu komunikacji międzyagentowej, jakie wyznacza na przykład FIPA[84]. JADE[85][86], rozwijana w laboratoriach Telecom Italiana, jest platformą zgodną z tymi standardami. Dodatkowo, dostarcza pakiet wspomagający rozwijanie agentów.

#### Funkcjonowanie platformy

W JADE *platforma* może obejmować kilka współdziałających ze sobą maszyn wirtualnych Java (Java Virual Machine). Każda maszyna wirtualna stanowi *kontener*, w którym funkcjonują agenci. Zgodność ze standardami FIPA oznacza obecność wymaganych serwisów SZA, KKA i UK[87]. Platforma dostarcza kanał komunikacyjny pozwalający na wysyłanie wiadomości ACL przez agentów. Wspiera mobilność agentów, to jest migrację ich stanu i kodu agentów między kontenerami i między połączonymi ze sobą platformami. Wykorzystanie technologii JAVA RMI pozwala na tworzenie aplikacji rozproszonej poprzez dodawanie do platformy kontenerów na stacjach zdalnych. Na rysunku 2.8 znajduje się przykład współdziałających ze sobą platform i kontenerów.

#### Implementacja zadań agenta

Agent w JADE jest reprezentowany przez klasę `Agent`. Stworzenie agenta wykonującego przyznane mu zadania polega na implementacji jego zachowań, reprezentowanych przez instancje klasy `Behaviour` i jej pochodnych. JADE dostarcza mechanizmu zarządzania kolejnością wywoływania zachowań, przedstawionym na rysunku 2.9.

[81] Cervenka, Radovan. 2003. *Prometheus Design Tool*. `http://www.auml.org/auml/documents/Prometheus030402.doc`.

[82] *Prometheus Design Tool - application site*: `http://www.cs.rmit.edu.au/agents/pdt/`.

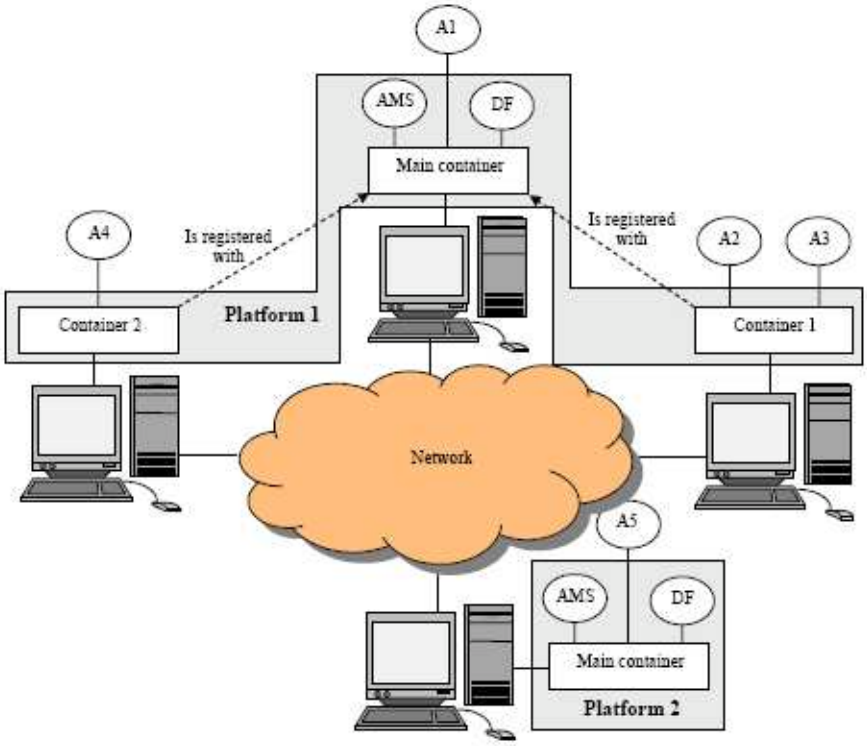
[83] Ang. *consistency cross-checking*.

[84] *Foundation for Intelligent Physical Agents* – organizacja skupiająca twórców technologii agentowej. Jej celem jest tworzenie standardów związanych z komunikacją i współpracą agentów w systemach wieloagentowych. Swoim członkom stawia wymagania, które powodują, że systemy rozwijane przez różne organizacje mogą z sobą współpracować.

[85] Java Agent Development Framework.

[86] *Jade - Java Agent DEvelopment Framework*: `http://jade.tilab.com/`.an Open Source platform for peer-to-peer agent based applications.

[87] Omawianych w paragrafie 2.3.4.




Rysunek 2.8: Platformy agentowe i kontenery (opracowano na podstawie (Caire 2003)).

### Wsparcie dla ontologii

W związku z potrzebą ustalenia wspólnego kontekstu rozmowy[88] JADE dostarcza wsparcia dla ontologii. Dostarczone mechanizmy pozwalają na (Caire 2004):

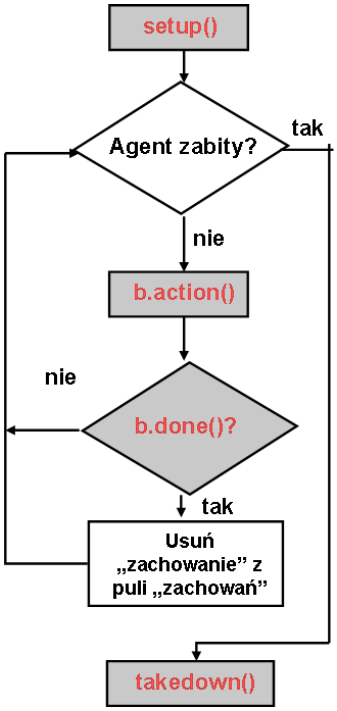
- tworzenie własnych ontologii (w językach SL i LEAP),
- odwzorowanie konceptów ontologii przez klasy w Javie,
- zawieranie w wiadomości ACL konceptów ontologii (*eksternalizacja*) i wyłanianie ich z powrotem do odpowiednich klas w Javie (*internalizacja*).

Trzecia możliwość daje ogromne usprawnienie przy tworzeniu systemu agentowego: obiekty Java są proste w manipulowaniu, a w postaci łańcucha tekstowego – łatwe do przesyłania (patrz rysunek 2.10).

### Brak bezpośredniego wsparcia dla inteligencji

Jedną z niewielu wad platformy JADE jest brak mechanizmów wspierających „inteligencję", czyli planowanie i wnioskowanie. Na szczęście użycie za podstawę Javy pozwala na stosunkowo prostą współpracę z systemami eksperckimi zaimplementowanymi w tym języku.

[88] Patrz paragraf 2.3.3.




Rysunek 2.9: Przebieg obsługi wątku agenta. Metody zaznaczone szarym tłem musi zaimplementować programista (opracowano na podstawie (Caire 2003)).

## 2.4 Interfejs WWW i jego rola

Swój sukces[89] sieć zawdzięcza organizacji informacji w postaci *hipertekstu*, dzięki czemu użytkownicy mogą poruszać się w obrębie i między dokumentami za pomocą *hiperodnośników*[90]. Hipertekst bazuje na dwóch technologiach, które determinują nawigację użytkownika w sieci WWW: protokół HTTP i język opisu stron HTML.

### 2.4.1 Protokół HTTP

HTTP[91] jest oparty na *architekturze klient-serwer* (patrz rysunek 2.11) i określa formę żądań[92] i odpowiedzi[93] między komputerami. Protokół określa cztery proste metody pozwalające na:

- pobieranie z serwera dokumentów (GET), określonych przez *dynamiczny* URL[94], określający dokładny adres dokumentu (*statyczny* URL) i dodatkowe parametry (tzw. *querystring*),
- pobieranie informacji o dokumentach (HEAD),

[89] Richter, Agnieszka. 2001. *Historia Internetu*: `http://www.kailastudio.com.pl/design/htm/article/historia.htm`.

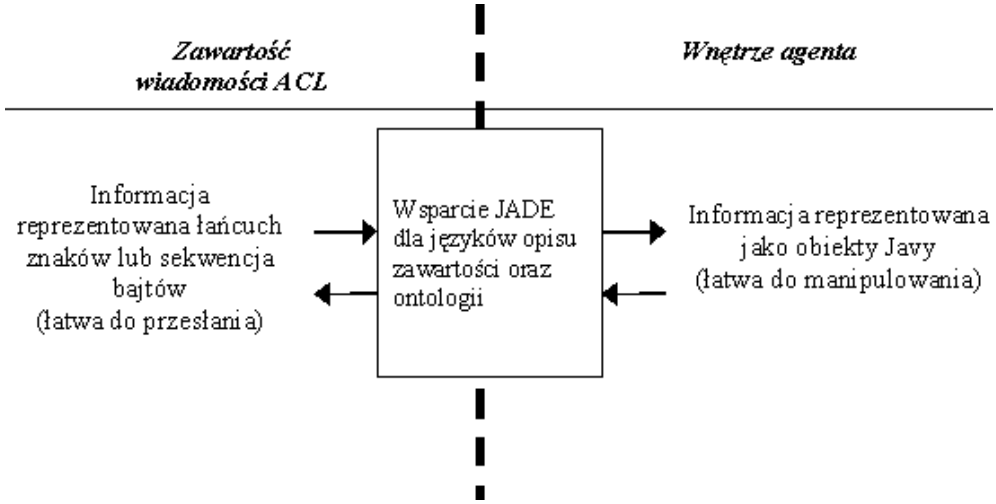
[90] Ang. *hiperlinks*

[91] Ang. *Hypertext Transfer Protocol*, specyfikacja znajduje się pod *HTTP - Hypertext Transfer Protocol*: `http://www.w3.org/Protocols/`.

[92] Ang. *request*

[93] Ang. *response*.

[94] Ang. *Uniform Resource Locator*, oznacza zunifikowany format odnośników do zasobów (głównie w Internecie), specyfikacja znajduje się pod *Uniform Resource Locators (URL)*: `http://www.ietf.org/rfc/rfc1738.txt`.




Rysunek 2.10: Konwersja między konceptami ontologii a odpowiadającymi im klasami Java przy użyciu JADE (opracowane na podstawie (Caire 2004)).

- wysyłanie danych z formularzy (PUT i POST).

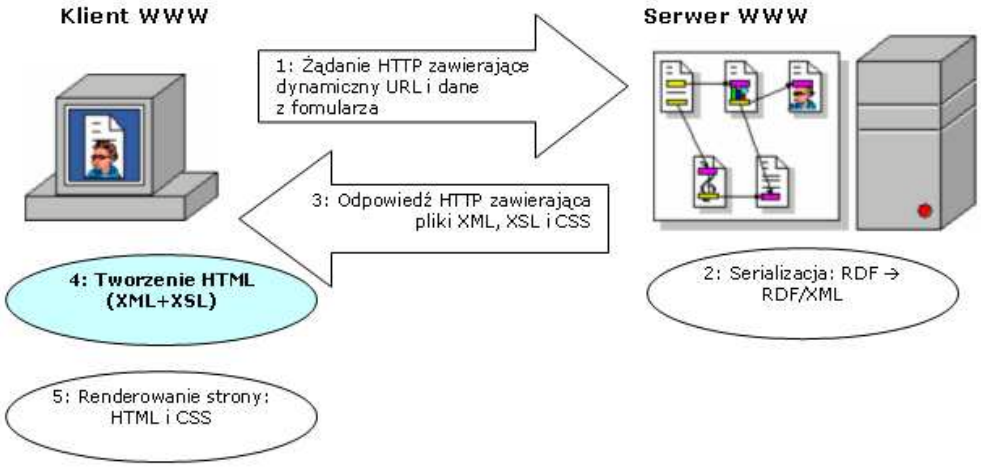
Protokół HTTP wymaga *proaktywności* klienta, tzn. że nawiązanie połączenia następuje wyłącznie z inicjatywy użytkownika (wysłanie żądania) i po dostarczeniu odpowiedzi (dokumentu wraz z nagłówkami) jest, zazwyczaj, zrywane. Cała ta operacja ma charakter *transakcji*, a informacje o niej przepadają po zamknięciu połączenia. W związku z tym protokół nazywany jest *bezstanowym*[95].

Bezstanowość protokołu nie pozwala na identyfikowanie użytkownika, co w procesie personalizacji jest zadaniem kluczowym, pozwalającym dopasowywać się do jego doświadczeń. Dla serwera HTTP, żądania wysłane czy to z komputera z Chin, czy z Polski niczym się nie różnią, poza numerem IP. Niestety numer IP nie pozwala na jednoznaczne rozpoznanie użytkownika (pod tym samym numerem może występować kilka osób). Dlatego w językach takich jak PHP, ASP czy JSP wprowadzono pojęcie sesji. Każdemu użytkownikowi odpowiada odpowiedni obiekt sesji, przechowywany po stronie serwera. Identyfikacja użytkownika polega na każdorazowym przekazywaniu identyfikatora sesji za pomocą metod POST/GET lub *ciasteczek*[96]. Żadne z tych rozwiązań nie jest idealne. Metody POST/GET wymagają dodawania identyfikatora sesji do każdego formularza i każdego odnośnika na stronie. Natomiast część przeglądarek (w tym telefony udostępniające WAP), nie obsługują ciasteczek lub pozwalają na ich wyłączenie.

### 2.4.2 Język opisu zawartości dokumentu – HTML

Dokument zwracany w odpowiedzi na żądanie zazwyczaj określony jest w formacie HTML[97]. Syntaktyka języka pozwalająca na jednoczesny opis wyglądu, zawartości i powiązań (odnośników) dokumentów hipertekstowych zadecydowała o sukcesie WWW w jej pierwszym pokoleniu. W chwili obecnej ta cecha stała się jednak problemem: treść

[95] Ang. *stateless*.
[96] Ang. *cookies*.
[97] Ang. *Hypertext Markup Language*, specyfikacja dostępna jest pod *W3C HTML Home Page*: `http://www.w3.org/MarkUp/`.




Rysunek 2.11: Proponowany obieg informacji w Semantic Web.

nie jest opisana w sposób ułatwiający poszukiwanie informacji. Pierwszym krokiem stało się wprowadzenie arkuszy styli CSS[98], nadal jednak część informacji wymieszana była ze sposobem formatowania. Prawdziwym rozwiązaniem stał się XML[99], jednak i on nie doprowadził do wyparcia HTML, ponieważ XML służy do opisu danych, a HTML – sposobu ich wyświetlania. Aby transformować XML do postaci HTML, po stronie klienta wykorzystywane jest technologia: XSL(T)[100].

Na rysunku 2.11 znajduje się schemat obiegu informacji w rozwijającej się sieci WWW. W kroku 4. zachodzi transformacja XML do HTML z użyciem szablonu XSL. W chwili obecnej istnieje możliwość, aby odciążyć klienta i dostarczać mu gotowy plik HTML, a obowiązek transformacji przenieść na serwer. Jedną z nich jest Apache Cocoon[101], serwlet Javy pozwalający z plików XML tworzyć dokumenty o różnych formatach wyjściowych, np. HTML za pomocą szablonów XSL. Podobnie działa Raccoon[102], serwer napisany w Pythonie, z tym że za źródło transformacji traktuje plik RDF/XML, a rolę szablonów pełnią pliki RxSLT.

### 2.4.3 Interakcja z interfejsem WWW

Obecnie sieć WWW (World Wide Web) ewoluuje do *sieci drugiego pokolenia* [103], zwanej *Semantyczną Siecią WWW*. Nie mniej jednak interfejs WWW wciąż wydaje się najważniejszym sposobem dostępu do jej zasobów, a rozwój wielu technologii nie zmienił faktu, że przeglądarki internetowe udostępniają ciągle te same sposoby interakcji z siecią WWW(Helic 2001):

[98] Ang. *Cascading Style Sheets*, opisane w *Cascading Style Sheets (CSS)*: `http://www.w3.org/Style/CSS/`.

[99] Ang. *Extensible Markup Language*, dokładniej opisane w *Extensible Markup Language (XML)*: `http://www.w3.org/XML/`.

[100] Ang. *Extensible Stylesheet Language (Transformation)*, dokładniej opisane w *Extensible Stylesheet Language Family (XSL)*: `http://www.w3.org/Style/XSL/`.

[101] Cocoon. 2005. *The Apache Cocoon Project*: `http://cocoon.apache.org/`.

[102] *Raccoon*: `http://rx4rdf.liminalzone.org/Racoon`.

[103] Ang. *second-generation web*.

1. *szperanie*[104], czyli podążanie za odnośnikami,

2. formułowanie *zapytań* w celu osiągnięcia określonego celu.

[104] Ang. *browsing*, nazywane także *point-and-click*, czyli *wyceluj i kliknij*.

# Rozdział 3

# System wspomagania podróży

Potrzeba stworzenia inteligentnego serwisu pomagającego turyście w zaplanowanie podróży jest podyktowana nadmiarem informacji i brakiem ich usystematyzowania. Dowodzą tego sytuacje (przedstawione w rozdziałach 1 i 2), w obliczu których staje turysta. Przytaczana analogia między agentem biura podróży a agentem programowym pozwala wierzyć, że aplikacja technologii Semantycznej Sieci WWW w dziedzinie turystyki powinna rozwiązać ten problem. Jednak w ciągu ostatnich 15 lat wiele było prób stworzenia odpowiedniego, kompetentnego systemu, a większość z nich nie wyszła poza fazę projektu. Główną przyczyną, jak pokazuje doświadczenie, jest ciągła ewolucja i niepewność technologii Semantycznej Sieci WWW.

Projekt *systemu wspomagania podróży* jest przede wszystkim analizą tych problemów i próbą wykorzystania tych rozwiązań, które na dzień dzisiejszy wydają się najlepsze i najpewniejsze. Do najważniejszych, rozważanych przez Foundation of Intelligent Physical Agents, należą[1]:

- *wyszukiwanie informacji*: udostępnienie bazy danych oraz przeszukiwanie ich za pomocą interfejsu WWW,
- *planowanie*: system powinien poruszać się nie tylko w obrębie własnej dziedziny, ale także współpracować z aplikacjami pokrewnymi, kalendarzami, pocztą elektroniczną i innymi narzędziami wspomagania pracy biurowej,
- *mobilność użytkownika końcowego*: mobilność związana jest (a) z możliwością przemieszczania się turysty, (b) z możliwością zmiany urządzenia służącego do kontaktu z systemem i (c) problemami z komunikacją w przypadku braku połączenia,
- *mobilność agentów*: związana z mobilnością użytkownika, wymaga od systemu wsparcia dla transferu kodu agentów lub ich stanów w obrębie sieci,
- *łączone i konkurujące ze sobą usługi*: porównywanie właściwości, negocjowania kosztów i czasu,
- *modelowanie użytkownika*: analiza preferencji użytkownika i dostosowanie się do nich.

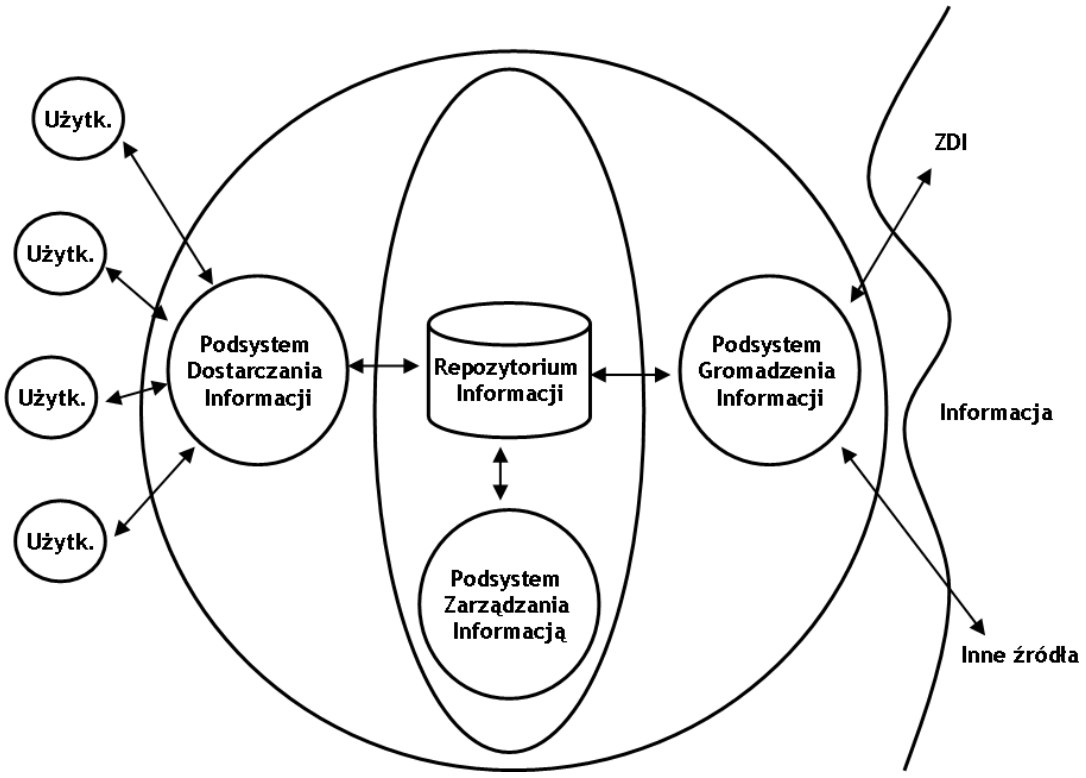
[1] *FIPA Personal Travel Assistance Specification*: `http://www.fipa.org/specs/fipa00080/XC00080A.html#_Toc505481930`.

## 3.1 Aktualny stan projektu

Prace nad projektem zaczęły się w 2001 (Ali *et al.* 2001; Galant *et al.* 2002; Angryk *et al.* 2002). Architektura systemu na dzień dzisiejszy (Gordon i Paprzycki 2005) została przedstawiona na rysunku 3.1. Determinują ją trzy funkcjonalności:

- *gromadzenie informacji*[2] – agregacja informacji potrzebnych w systemie od zaufanych dostawców i ze źródeł internetowych,
- *zarządzanie informacją*[3] – czuwanie nad poprawnością i aktualnością zgromadzonych danych,
- *dostarczenie informacji*[4] – dostarczanie użytkownikowi spersonalizowanej informacji, spełniającej określone przez niego kryteria.

Realizacją każdej z tych funkcjonalności zajmuje się oddzielny podsystem.




Rysunek 3.1: Architektura *systemu wspomagania podróży* (opracowane na podstawie (Gordon i Paprzycki 2005)).

Projekt wykorzystuje platformę agentową JADE[5]. Jej wydajność potwierdziły testy, które przeprowadziliśmy w roku 2004 (Chmiel *et al.* 2004).

[2] Ang. *Content Collection.*
[3] Ang. *Content Management.*
[4] Ang. *Content Delivery.*
[5] Omawianą w paragrafie 2.3.6.

### 3.1.1 Podsystem Gromadzenia Informacji

Podsystem (PGI) zajmuje się gromadzeniem informacji. Jej wiarygodność gwarantują *Zweryfikowani Dostawcy Informacji*[6], do których wliczamy dostawców oferujących treść w ustalonej formie (w postaci stopek RSS[7]) oraz statyczne strony WWW, z których informacja jest ekstrahowana okresowo. Drugorzędną rolę pełnią *inne źródła* internetowe. Ich ilość oraz różnorodność formy i treści ogranicza możliwości całkowitego ich przeszukania. Dodatkowo należy wziąć pod uwagę, że mogą one zawierać dane niegodne zaufania.

Zebrane dane trafiają do *Repozytorium Informacji*[8] w formie semantycznie ustrukturalizowanej (RDF). W chwili obecnej Repozytorium (nazywane dalej *Bazą Danych*) zawiera informacje o ponad 260 tysiącach lokalach gastronomicznych na świecie. Dane te udostępnia serwis *Chefmoz dining guide*[9] w formie pliku RDF/XML. Przy tak ogromnej ilości informacji nie brakuje błędów semantycznych i syntaktycznych, z których tylko te drugie udało nam się naprawić – w sposób zautomatyzowany (Gawinecki *et al.* 2005a).

Serwis nie udostępnia także żadnej ontologii (nazywanej dalej *ontologią domeny*, czy też *ontologią dziedziny wiedzy*[10]), definiującej sposób przetwarzania danych, a jedynie krótki opis użytych konceptów. W związku z tym, opierając się na dostarczonych danych dokonaliśmy odtworzenia potrzebnej ontologii (Gawinecki *et al.* 2005a; Gawinecki *et al.* 2005b). Jej fragment opisuje rysunek 2.4[11].

Aby zapewnić zgodność z ontologią, dane zostały przeformatowane. Poniżej prezentujemy opis jednego z lokali z bazy restauracji (w notacji N3):

```
:Poland_WP_Poznan_Avanti__Restauracja_Bar_Kawiarnia__1052601155 a
    :Restaurant;
  loc:city           "Poznan";
  loc:country        "Poland";
  loc:crossStreet    "main hall";
  loc:fax            "+48 (61) 866 28 70";
  loc:neighbourhood  "Dworzec PKP - Railway Station";
  loc:phone          "+48 (61) 866 87 41";
  loc:state          "WP";
  loc:streetAddress  "PKP Poznan Glowny, ul. Dworcowa 1";
  loc:zip            "61-801";
  res:URL            "http://www.avanti.poznan.pl/wpodrozy.htm";
  res:alcohol        :NoAlcoholServed;
  res:cuisine
                     res:CafeteriaCuisine,
                     res:FastFoodCuisine,
                     res:PolishCuisine;
```

[6] Ang. *Verified Content Providers*.

[7] Umowna rodzina języków znacznikowych do przesyłania nagłówków wiadomości. Więcej informacji znajdzie czytelnik pod *RSS (protocol)*: `http://en.wikipedia.org/wiki/RSS_(protocol)`.

[8] Ang. *Content Storage*.

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

[10] Ang. *domain knowledge ontology*.

[11] Jej zapis w postaci N3 znajduje się w załączniku B.1.

```
res:description   "Acceptable source of food if you have just
    arrived at the Poznan railway station and waiting for
    another train in known/unknown direction. At the bar you can
     get fast food, go upstairs if you are looking for a more
    quiet place - cafe is OK, but the the place called "
    restaurant" is more like a fast food again.";
res:dress          res:Casual;
res:hours          "24h";
loc:locationPath   "Poland/WP/Poznan";
res:parking        res:OwnParkingLot;
res:parsedHours    "0-24|0-24|0-24|0-24|0-24|0-24|0-24";
res:price          mon:Inexpensive;
res:reservations   res:NotAcceptedReservations;
res:smoking        res:NotPermittedSmoking;
res:title          "Avanti, Restauracja-Bar-Kawiarnia".
```

W chwili obecnej do przetwarzania danych RDF wykorzystywany jest system JENA[12].

### 3.1.2 Podsystem Zarządzania Informacją

Podsystem (PZI) zarządza Repozytorium Informacji, wykonując co następuje:

- uzupełnia niepełne informacje, na przykład brakujący numer telefonu lokalu,
- aktualizuje dane podatne na zmiany, jak chociażby repertuar kina czy godziny otwarcia muzeum.

### 3.1.3 Podsystem Dostarczania Informacji

Celem podsystemu (PDI) jest dostarczanie użytkownikowi spersonalizowanej informacji odpowiadającej jego zapytaniu. Jest to w tej chwili najdalej posunięta w rozwoju część projektu.

W chwili obecnej podsystem realizuje dwa scenariusze: dostarcza formularz wyszukiwania restauracji oraz znajduje restauracje odpowiednio do podanych kryteriów.

Użytkownik może połączyć się z systemem za pomocą urządzenia posiadającego dostęp do Internetu. Za odebranie żądania odpowiedzialny jest *Agent Proxy*, który przekazuje je dalej do *Agenta Osobistego*. Agent Osobisty reprezentuje użytkownika w systemie i w jego imieniu dokonuje obsługi żądania. Zlecenie wyszukanie odpowiedniej restauracji przekazuje do *Agenta Serwisu Restauracyjnego* i tak uzyskane wyniki posyła do przekształcenia do *Agenta Transformującego Widok*. W ten sposób otrzymuje dane w formacie akceptowalnym przez medium użytkownika (np. HTML, WML). Rezultat całej operacji jest przekazywany z powrotem do klienta za pomocą odpowiedniego Agenta Proxy. Szczegółowy przebieg tego scenariusza zostanie przedstawiony w paragrafie 3.3.

[12] Omawiany w paragrafie 2.2.6.

### Urządzenia klienckie

W założeniach systemu medium klienta powinno być *cienkie*, tzn. dysponować wyłącznie najpotrzebniejszymi możliwościami pozwalającymi na: wyświetlenie interfejsu systemu i komunikację z nim. W chwili obecnej możliwe jest połączenie za pomocą przeglądarki WWW lub telefonu komórkowego obsługującego WAP. Połączenie następuje za pomocą protokołu HTTP. Interfejs opisany jest za pomocą języka – odpowiednio – HTML lub WML. W fazie rozwoju jest obecnie wsparcie dla technologii agentów na telefonach komórkowych.

### Agent Proxy (APr)

Dla każdego medium istnieje Agent Proxy[13], stanowiący zewnętrzny interfejs systemu. Do jego zadań należy: bezpośrednie przyjmowanie żądania użytkownika, przetłumaczenie żądania na formę wiadomości ACL, przekazanie wiadomości do Agenta Osobistego i odebranie niego wyników, wysłanie odpowiedzi z powrotem do klienta. Taka funkcjonalność pozwalający na uniezależnienie pracy systemu od rodzaju medium klienta.

### Agent Osobisty (AO)

Agent Osobisty[14] do wykonania żądania przyjętego od użytkownika deleguje siebie i innych agentów. W planowanym rozwoju systemu ma być odpowiedzialny za personalizację zwracanej informacji.

### Agent Serwisu Restauracyjnego (ASR)

Agent[15] zapewnia innym agentem dostęp do Repozytorium Informacji z restauracjami.

### Agent Transformujący Widok (ATW)

Agent[16] odpowiedzialny za transformację danych semantycznych (RDF) do formatu rozumianego przez medium klienckie (HTML, WML itp.).

## 3.2 Opis szczegółowy strony klienckiej

Podsystem Dostarczania Informacji jest także nazywany stroną kliencką. Poza zaimplementowaniem funkcji wspomagających wyszukiwanie restauracji zawiera szereg szczegółowych rozwiązań wspomagających rozwój system przez kolejnych projektantów. Dokładny opis konstrukcji z systemu wraz z uzasadnieniem podjętych decyzji znajduje się w pracy (Kaczmarek 2005), tutaj natomiast opisujemy architekturę w sposób przeglądowy.

[13] Ang. *Proxy Agent* (PrA).
[14] Ang. *Personal Agent* (PA).
[15] Ang. *Restaurant Service Agent* (RSA), określany też jako *Database Agent* (DBA), czyli Agent Bazodanowy.
[16] Ang. *View Transformer Agent* (VTA), nazywany też *Transformer Agent* (TA).

### 3.2.1 Architektura Model-Widok-Kontroler

W sytuacji, gdy w systemie występuje wiele sposobów (mediów) do interakcji z nim, naturalnym wydaje się oddzielenie warstwy prezentacji od warstwy danych. Odpowiedzią jest wzorzec projektowy *Model-Widok-Kontroler*[17][18], zaadaptowany na potrzeby systemu wieloagentowego, co przedstawia tabela 3.1.

| *Warstwa wzorca* | *Rola* | *Agenci w systemie* |
|---|---|---|
| *Model* | Udostępnia zasadniczą logikę biznesową systemu, przeprowadzając operacje na danych. | Agent Serwisu Restauracyjnego |
| *Widok* | Odpowiada za prezentację danych użytkownikowi. | Agent Transformujący Widok |
| *Kontroler* | Tłumaczy żądania użytkownika na operacje Modelu i wybiera odpowiedni Widok jako odpowiedź. | Agent Osobisty |

Tabela 3.1: Realizacja wzorca projektowego Model-Widok-Kontroler w systemie.

#### Realizacja Modelu

Realizacją modelu zajmuje się Agent Serwisu Restauracyjnego. Na podstawie paramerów krzekazanych w żadaniu wyszukuje w Repozytorium odpowiednich restauracji.

#### Realizacja Kontrolera

W standardowym podejściu *Kontroler* pełni dwie role: (a) przechwycenie żądania od użytkownika oraz (b) przetłumaczenie go na operację Modelu i wybranie odpowiedniego sposobu prezentacji (Widoku). W związku z różnicami w sposobie interakcji z różnymi mediami klienckimi funkcje te zostały rozdzielone do dwóch agentów: odpowiedni do medium klienta Agent Proxy pełni rolę (a), natomiast za rolę (b) odpowiedzialny jest Agent Osobisty.

Ponadto, wyodrębnienie Agent Proxy (Agent Proxy HTML i Agent Proxy WML ) pozwoliło na połączenie świata *agentowego*, który komunikuje się *wyłącznie* za pomocą wiadomości ACL, ze światem *nie-agentowym*, który posługuje się – przykładowo – protokołem HTTP. W chwili obecnej Agent Proxy ma charakter agenta-serwera HTTP[19].

[17] Ang. *Model-View-Controller*.

[18] *Design Patterns for Building Flexible and Maintainable J2EE Applications*: `http://java.sun.com/developer/technicalArticles/J2EE/despat/`.

[19] Dokładniejszy opis przyjętego rozwiązania, analizę problemu synchroniczności protokołu HTTP i asynchroniczności komunikacji międzyagentowej znajdzie czytelnik w pracach (Kaczmarek 2005; Kaczmarek *et al.* 2005; Gawinecki *et al.* 2003).

**Realizacja Widoku**

Funkcję Widoku w systemie realizuje Agent Transformujący Widok. Agent ten przyjmuje wiadomość ACL z danymi i opisem docelowego medium. Następnie żąda od serwera RACCOON[20] dokonania transformacji i zwraca wynik w postaci wiadomości ACL. W przypadku pojawienia się nowego scenariusza lub nowego medium konieczne jest napisanie nowe szablonu transformującego.

### 3.2.2 Protokół obsługi żądania

W systemie wielokrotnie pojawia się sytuacja, kiedy to agent A otrzymuje określone zadanie i podejmuje się go sam lub deleguje do niego agenta B. W każdym z tych przypadków agent A pełni rolę *Kontrolera*, w programowaniu obiektowym zdefiniowaną przez następujące funkcjonalności (Alur *et al.* 2001):

- akceptacja zapytań,
- wykonywanie operacji na zapytaniu,
- wybieraniu odpowiedniego delegata do realizacji zadania w ramach zapytania,
- przekazanie zapytania delegatowi,
- mechanizm obsługi błędów.

W związku z tym projektantowi rozwijającemu system należało zapewnić dwie możliwości: przyjmowanie zlecenia i delegowanie do dalszego wykonania tego zlecenia. W stworzonym systemie wygląda to następująco.

Agent A w otrzymanej wiadomości ACL znajduje nazwę *akcji* do wykonania i za pomocą klasy `HandlerSelector` na podstawie nazwy akcji dokonuje wyboru *zachowania*. Powołane zachowanie może zająć się samodzielnym wykonaniem akcji lub wspomóc się (częściowo lub całkowicie) funkcjonalnością innego agenta, na przykład agenta B. W tym celu inicjuje protokół *FIPA Request*[21] w stosunku do agenta B. Agent-odbiorca może żądanie odrzucić lub przyjąć i podjąć się jego wykonania – samodzielnie lub z pomocą innych agentów, oczywiście. Po wykonaniu zadania agent B informuje agenta A o wynikach zadania za pomocą wiadomości ACL.

Do projektanta należy decyzja, które zadanie agent może wykonać samodzielnie, a które przekazać dalej. Należy tu bowiem ocenić koszt interakcji z innymi agentami, posiadany dostęp do wymaganych danych i funkcjonalności.

Agenci komunikujący się ze sobą, niezależnie od szczegółów wykonywanego zadania, muszą w tych komunikatach określić nazwę akcji do wykonania lub jej rezultat oraz dane związane z sesją aktywnego użytkownika (typ medium klienckiego oraz – w przyszłości – identyfikator sesji i identyfikator użytkownika). W związku z tym naturalnym stało się stworzenie wspólnej ontologii rozmowy zwanej *SystemOntology*, której koncepty określone zostały w tabeli 3.2.

[20] Dokładniejszy opis serwera RACCOON znalazł się w rozdziale 2.4.2.
[21] Opisany w paragrafie 2.3.3.

| *Koncept* | *Znaczenie* |
|---|---|
| *MessageInfo* | Zawiera informacje o sesji aktywnego użytkownika. |
| *ConceptWithInfo* | Łączy klasę *MessageInfo* z jedną z poniższych. |
| *GenericAction* | Odpowiada ogólnej akcji bez jasno określonych danych. |
| *QueryAction* | Odpowiada akcji zapytującej: dane wejściowe mają postać: klucz-wartość. |
| *ActionResult* | Określa rezultat wykonanej akcji. |

Tabela 3.2: Ontologia systemowa.

Koncept *MessageInfo* zawiera pola określające nazwę wspomnianej akcji (*ActionName*) oraz typ medium klienckiego (*MediaType*). Natomiast koncepty *GenericAction* i dziedziczący z niego *QueryAction* zawierają pole na dane: *InputData*.

Dzięki wsparciu JADE dla ontologii[22] każdy koncept ontologii posiada swój odpowiednik w postaci klasy Java. Instancje klas `GenericAction` i `QueryAction` są wysyłane w ramach wiadomości *FIPA request*, natomiast instancja klasy `ActionResult` jest zwracana poprzez *FIPA inform* lub *FIPA failure*.

## 3.3 Scenariusz użycia

Celem lepszego zrozumienia działania systemu przedstawimy przykładowy scenariusz użycia. Załóżmy, że użytkownik poszukuje restauracji w Poznaniu, w której serwowana jest kuchnia włoska i honoruje kartę American Express. Wypełniony formularz zapytań przedstawia rysunek 3.2.

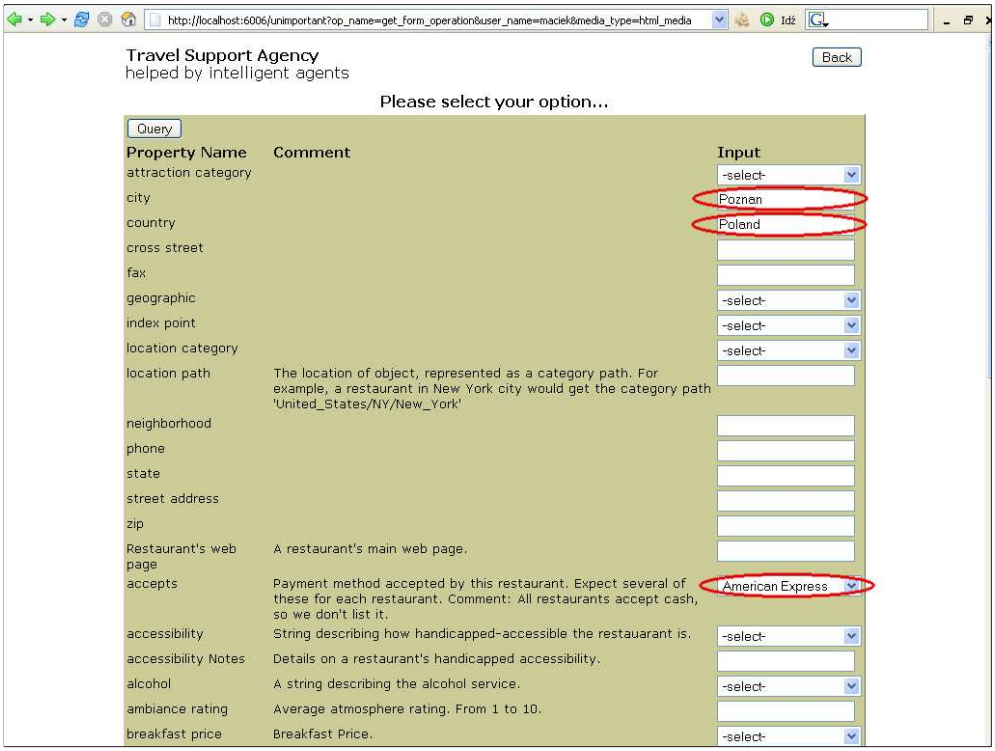
W wyniku zapytania użytkownik otrzymuje listę restauracji przedstawioną na rysunku 3.3.

### 3.3.1 Przepływ informacji

Od chwili wysłania formularza zapytań do momentu wyświetlania wyników system dokonuje realizacji żądania w formie przedstawionej na rysunku 3.4. Po naciśnięciu przycisku *Query* kryteria wyszukiwania z formularza (wraz z nazwą akcji odpowiedzialnej za jej wykonanie) trafiają za pomocą metody GET protokołu HTTP[23] do APr. Mają one postać następującego querystringu:

```
http://localhost:6006/?
http://www.agentlab.net/schemas/location#city = Poznan
http://www.agentlab.net/schemas/location#country = Poland
& http://www.agentlab.net/schemas/restaurant#cuisine =
    ItalianCuisine
& http://www.agentlab.net/schemas/restaurant#accepts =
    AmericanExpressCard
& op_name = find_restaurant_operation
```

[22]Patrz paragraf 2.3.6.
[23]Patrz paragraf 2.4.1.




Rysunek 3.2: Formularz zapytania w Podsystemie Dostarczania Informacji (bez personalizacji). Zakreślone zostały wybrane kryteria zapytania.

```
& media_type = html_media
```

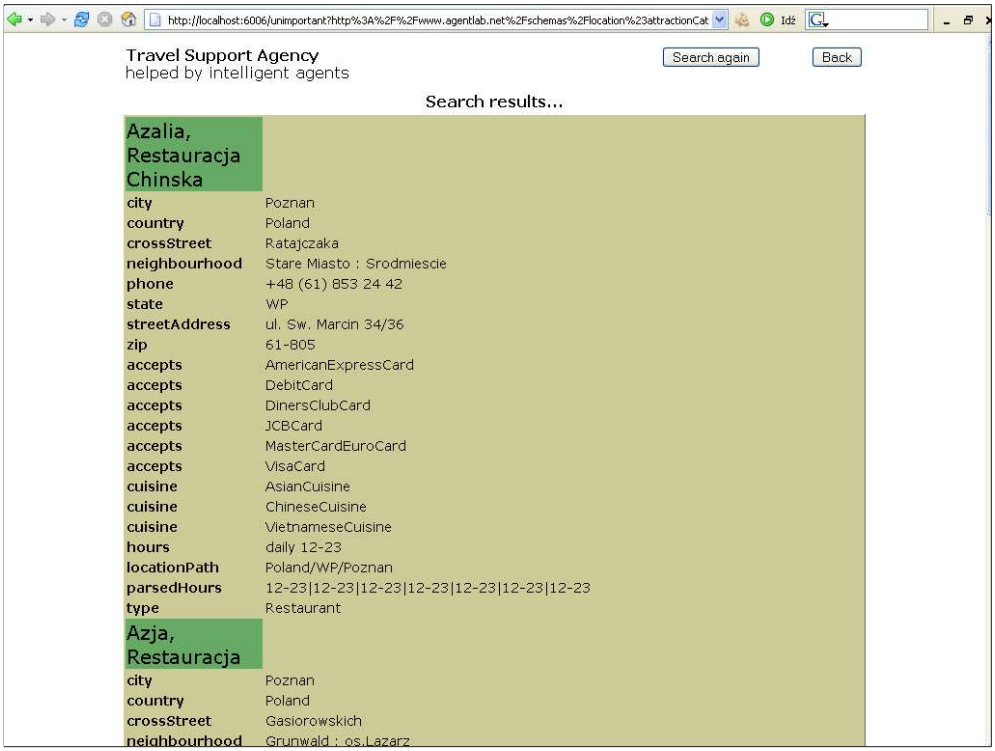
Zostaje on przetłumaczony na wiadomość ACL, o treści zgodnej z zaproponowaną *SystemOntology*:

| *Koncept* | *Wartość* |
|---|---|
| *MessageInfo* | W polu *MediaType* zostaje umieszczona wartość *html_media*, a w polu *ActionName* – *find_restuarant_operation*. |
| *QueryAction* | W polu *InputData* zostają umieszczone kryteria wyszukiwania. |

i przekazane do PA. On, a raczej jego zachowanie `FindRestaurantHandler`, wskazane przez wartość w polu *ActionName*, zajmuje się obsługą całego żądania. Przekazuje kryteria wyszukiwania do agenta bazodanowego (RSA), który tłumaczy je na postać zapytania RDQL[24] o następującej treści:

```
SELECT ?rest
WHERE
        (?rest, <loc:city>, 'Poznan'),
        (?rest, <loc:country>, 'Poland'),
        (?rest, <res:cuisine>, <res:ItalianCuisine>),
        (?rest, <res:accepts>, <mon:AmericanExpressCard>)
USING
```

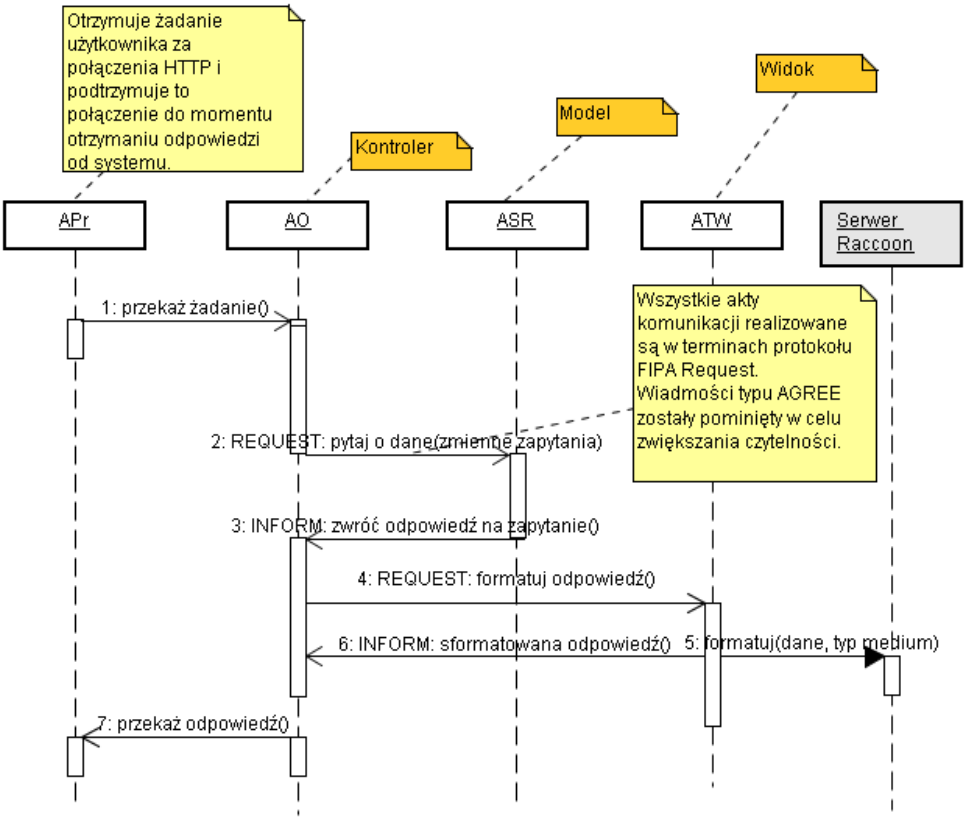
[24]O języku RDQL mówiliśmy w paragrafie 2.2.4.




Rysunek 3.3: Wyniki zapytania w Podsystemie Dostarczania Informacji (bez personalizacji).

```
loc FOR <http://www.agentlab.net/schemas/location#>,
mon FOR <http://www.agentlab.net/schemas/money#>,
res FOR <http://www.agentlab.net/schemas/restaurant#>
```

ASR zapytuje bazę danych za pomocą środowiska Jena, a otrzymaną odpowiedź w postaci RDF/XML pakuje do wiadomości ACL i odsyła z powrotem do AO. Następnie AO przekazuje te wyniki do ATW. ATW oprócz danych, wyłuskuje z wiadomości informację o medium klienta (*MediaType* w *MessageInfo*). Te informacje przekazuje do serwera Raccoon, który formatuje dane do postaci HTML i odsyła je z powrotem do ATW. Ostatecznie sformatowane wyniki trafiają do AO, dalej – do APr i użytkownika.




Rysunek 3.4: Diagram sekwencji dla obługi żądania użytkownika w Podsystemie Dostarczania Informacji (bez personalizacji). $APr$ – Agent Proxy, $AO$ – Agent Osobisty, $ASR$ – Agent Serwisu Restauracyjnego, $ATW$ – Agent Transformujący Widok. (opracowane na podstawie (Kaczmarek 2005))

# Rozdział 4

# Cel – profil użytkownika oparty na ontologii

Podsumujmy zebraną do tej pory wiedzę. Mooers tworząc pojęcie filtrowania informacji dostrzegł dwa zasadnicze problemy: (a) w jaki sposób przechowywać informację w sposób intelektualny[1] i (b) jakie techniki pozwalają na znajdywanie informacji odpowiedniej (relewantnej). Ontologie, dzięki wysokopoziomowym relacjom i towarzyszącym im technikom wnioskowania stanowią odpowiedź na pierwszą kwestię. Natomiast aby znaleźć informację relewantną, to znaczy odpowiadającą potrzebom użytkownika, należy tego użytkownika dobrze poznać. W tym celu wykorzystuje się wspomniane techniki z dziedziny modelowania użytkownika. Jak wskazał Kobsa, model użytkownika stanowi podstawę technik filtrowania informacji (Kobsa 1990). Rodzi się pytanie, jak reprezentować wiedzę o użytkowniku?

W tym celu proponujemy wykorzystanie ontologii do konstrukcji profilu. Nie tylko jako sposobu jego reprezentacji. Mamy tu na myśli *ontologię domeny*, która może stanowić punkt odniesienia dla profilu użytkownika. Takie rozwiązanie wydaje się dobre w nowoczesnym systemie informacyjnym (jakim jest opisywany *system wspomagania podróży*), w którym ontologie i RDF(S) grają kluczową rolę. Chcemy zatem włączyć do tak rozwijanego systemu element modelowania użytkownika. Podsystem Dostarczania Informacji będzie miał na celu dostarczanie użytkownikowi informacji spersonalizowanej. Przyjrzyjmy się zatem zaletom wykorzystania ontologii przy konstrukcji profilu oraz problemom związanym ze integracją funkcji modelowania użytkownika z istniejącym systemem. Szczegółowe rozwiązanie przedstawimy w kolejnym rozdziale.

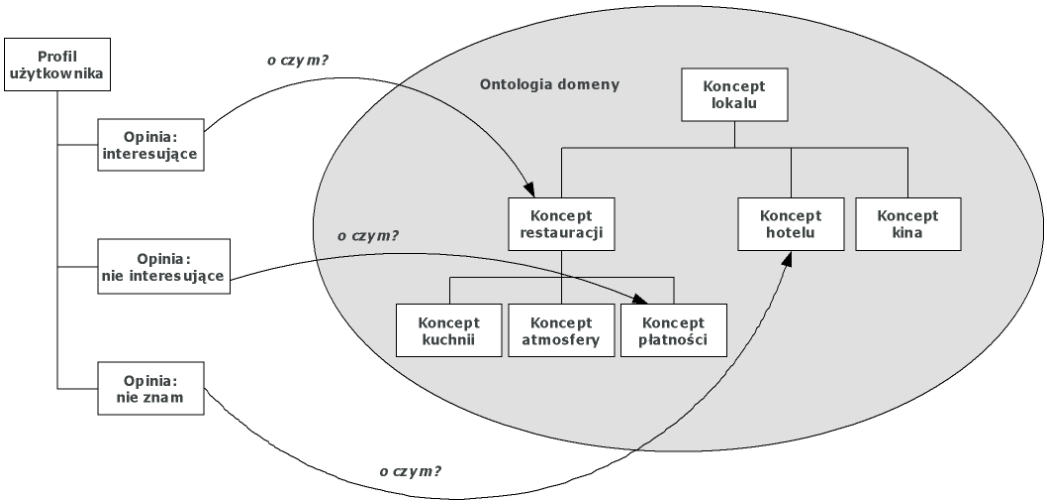
## 4.1 Wcześniejsze prace

Analizujemy tutaj krótko podobne rozwiązania dotyczące reprezentacji profilu i architektury systemu użytkownika.

[1] Znaczenie wyrazu „intelektualny” w omawianym kontekście zostało omówione w przypisach do wstępu.

### 4.1.1 Model typu „overlay”

Wykorzystanie struktury dziedziny (domeny) jako punktu odniesienia dla budowy systemu modelującego użytkownika nie jest nowym pomysłem. Domena stanowi sieć, w której koncepty (nazywane również tematami, elementami wiedzy lub obiektami) stanowią wierzchołki, a różne relacje między nimi tworzą krawędzie (Brusilovsky 1996). Model typu „overlay”, przedstawiony na rysunku 4.1, jest takim nałożeniem profilu użytkownika na *domenę* . Dla każdego konceptu domeny model użytkownika przechowuje wartość określającą wiedzę użytkownika o nim. Wartość ta może być typu binarnego (wiem-nie wiem), porządkowego (dobrze-średnio-słabo) lub w postaci prawdopodobieństwa szacującego znajomość konceptu. Model ten jest często nazywany *modelem studenta*, jako że został wymyślony między innymi na potrzeby modelowania wiedzy studentów (Greer i McCalla 1993).




Rysunek 4.1: Idea profilu użytkownika typu „overlay”.

### 4.1.2 Architektura

Każdy z systemów wspierających modelowanie użytkownika posiada komponent modelowania użytkownika (UMC[2]). Próchnicka i Fink analizując kwestię architektury systemów modelujących użytkownika biorą pod uwagę autonomiczność i niezależność komponentu modelowania użytkownika oraz samego modelu użytkownika (Próchnicka 2000; Fink 1996). Najgorszym rozwiązaniem pod tym względem jest *architektura monolityczno-lokalna*, w której aplikacja i komponent modelowania użytkownika stanowią jeden program. Wszelkie procesy przetwarzania informacji odbywają się sekwencyjnie, spowalniając interakcję z systemem, a powstały model użytkownika jest ściśle związany z systemem, w którym powstał. Korzystniejszym jest oddzielenie komponentu modelowania użytkownika od reszty systemu, jak to ma miejsce w *architekturze autonomomiczno-lokalnej*. Komponent modelowania użytkownika zajmuje się się realizacją zadań związanych z modelowania użytkownika dla wszystkich aplikacji na danym komputerze. W obu przypadkach komponent modelowania użytkownika ma charakter

[2]Ang. *User Modelling Component.*

zcentralizowany i – jak zauważa Kobsa – nie nadaje się do wtórnego zastosowania (Kobsa 2001). Naturalnym rozwiązaniem jest *architektura rozproszona*, w której znaczącym elementem są *serwery modelowania użytkownika*[3], przechowujące i zarządzające profilami użytkownika. Serwery modelowania użytkownika udostępniają swoje usługi innym komponentom systemu w obrębie sieci. Takie podejście sprzyja tworzeniu modeli bardziej kompleksowych i spójnych. W przypadku aplikacji takiej architektury w środowisku komercyjnym należy uwzględnić ryzyko dostępu do danych zawartych w profilu użytkownika przez osoby niepowołane niedozwolonego użycia tych danych (Kobsa i Pohl 1995). Aby ograniczyć to ryzyko autorzy autorzy omawianego już wcześniej systemu *IRES* umieścili profil i agenta osobistego (AO), zajmującego się nim, na maszynie użytkownika. AO potrafi kontaktować się z innymi usługami systemu dostępnymi w sieci (Montaner *et al.* 2002).

## 4.2 Zalety wykorzystania ontologii

W większości aplikacji modelujących użytkownika profil stanowi wewnętrzną część aplikacji, ograniczając możliwość dzielenia tego profilu z innymi aplikacjami. Autorzy (Bieliková i Kurus 2005) wymieniają wynikające z tego problemy:

- profil tego samego użytkownika musi być inicjalizowany oddzielnie dla każdego systemu rekomendującego
- brak przejrzystości zawartości profilu może doprowadzić do braku zaufania użytkownika.

Pytamy zatem, w jaki sposób rozwiązują te problemy ontologie i jakie są tego dalsze konsekwencje.

### 4.2.1 Przenośność i skalowalność profilu

Tworzenie profilu i jego uczenie jest kosztowne zarówno dla projektanta systemu, jak i jego użytkownika. Celem badaczy jest zatem tworzenie takich profili, które są niezależne od aplikacji i dziedziny wiedzy, w której się rozwijają. Proponowane rozwiązania polegają:

- związaniu modelu użytkownika z nim samym, a nie z konkretnym systemem (Kay 1999)
- standaryzacji struktury profilu, czyli ustanowieniu takiej jego semantyki, która będzie zrozumiała dla różnych systemów.

Samo stworzenie i udostępnienie ontologii profilu w powszechnie akceptowanym języku (RDF(S)) rozwiązuje problem standaryzacji.

Rozszerzenie dziedziny (przykładowo włączenie do ontologii domeny charakterystyki hoteli) nie wymusza tworzenia profilu użytkownika od początku, a jedynie zdobycie wiedzy czy też wyrobienie opinii o nowych konceptach. Charakterystyczny sposób konstrukcji ontologii typu bottom-up, czyli możliwość dodawania nowych konceptów w dziedzinie oraz opinii o nich w profilu, rozwiązuje problem od strony technicznej.

[3] Ang. *user model servers.*

### 4.2.2 Odkrywanie nowej wiedzy dzięki ontologiom

Kiedy w dziedzinie pojawiają się nowe koncepty, opinie użytkownika na ich temat są nieznane. Mogą być natomiast wywnioskowane na podstawie konceptów pokrewnych, co wykorzystują autorzy elektronicznego przewodnika *WebGuide* (Fink i Kobsa 2002). W systemie tym wykorzystana została struktura określająca hierarchię obiektów turystycznych. Przykładowo, w skład kategorii architektura wchodzą kościoły, zamki i budynki architektury współczesnej. Kiedy użytkownik jest zainteresowany zamkami i kościołami, to jest to dowód jego zainteresowania architekturą w ogóle, a być może również architekturą współczesną. Mamy zatem do czynienia z *propagacją* zainteresowania wzdłuż relacji określonych przez domenę.

Berger podaje przykład bardziej skomplikowanego systemu, w którym wykorzystywane są informacje semantycznie pokrewne (Berger *et al.* 2004). Wykorzystuje on strukturę podobną do ontologii (*sieć skojarzeniowa*[4]) łączącą terminy powiązane ze sobą znaczeniowo, na przykład łaźnie parową, basen i saunę. W sytuacji, gdy użytkownik poszukuje informacji o hotelu z sauną, system na pierwszych miejscach poleci hotele zawierające także podobne atrakcje.

Kobsa i Fink wskazują, że stosowanie wnioskowania na podstawie domeny pozwala uczyć profilu szybciej i poprawia jego skuteczność przy rekomendacji (Fink i Kobsa 2002).

### 4.2.3 Czytelność profilu

Ontologie pozwalają maszynom przetwarzać informację zrozumiałą dla ludzi. Kiedy profil jest przejrzysty i intuicyjny, wzrasta zaufanie i kooperatywność użytkownika względem systemu (Bieliková i Kurus 2005).

### 4.2.4 Łatwiejsza integracja z systemem

W systemie rozproszonym występuje co najmniej kilka jednostek, które przetwarzają tą samą informację w różnych celach (na przykład przetwarzanie historii zdarzeń przez różne techniki filtrowania informacji). Wykorzystanie ustalonej i jednoznacznej ontologii ułatwia komunikację, przykładowo system *OntobUM* (Razmerita *et al.* 2003) używa trzech ontologii:

- *ontologia domeny*, definiująca wiedzę w danej branży,
- *ontologia zdarzeń*, określająca semantykę zachowań użytkownika w interakcji z systemem,
- *ontologia użytkownika*, charakteryzująca strukturę jego zainteresowań, dla każdej techniki personalizacji inna.

## 4.3 Dodatkowe założenia. Stratyfikacja personalizacji

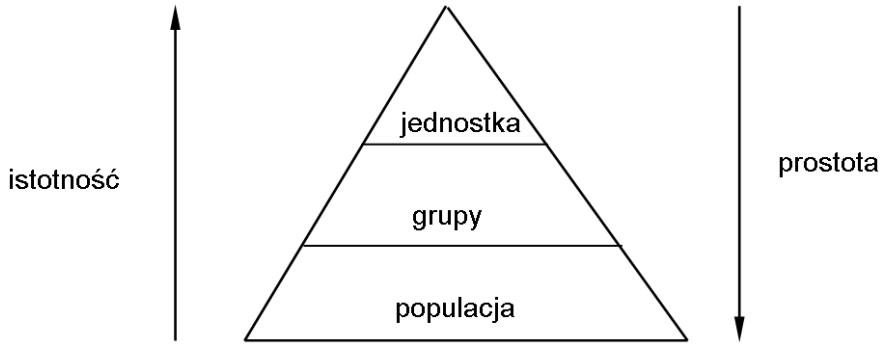
Zgodnie z założeniami systemu, personalizacja informacji ma mieć charakter *stratyfikacyjny* (Nistor *et al.* 2002). Oznacza to, że system informacyjny ma brać pod uwagę

[4]Patrz paragraf 1.2.1.

nie tylko indywidualne potrzeby użytkownika, ale także:

- zachowania cechujące grupę społeczną, w której się znajduje (*„Wszyscy studenci jeżdżą na wakacje do Finlandii”* lub bardziej szczegółowo *„Wszyscy jego znajomi uwielbiają oglądać horrory”*),
- oraz ogólne tedencje wyznaczające cele populacji w określonym czasie (*„Ostatnimi czasy dominuje moda na restauracje wietnamskie”*).

Oczywiście dla użytkownika najistotniejsza jest informacja przygotowana na podstawie jego indywidualnego doświadczenia (patrz rysunek 4.2). Jest ona jednak często trudna do zebrania (na przykład ze względu na początkową nieufność użytkownika do systemu[5]) lub skąpa (na przykład ze względu na krótki okres interakcji użytkownika z systemem – sparsity problem[6]). W tej sytuacji rozwiązaniem wydaje się wykorzystanie informacji bardziej ogólnych.




Rysunek 4.2: Stratyfikacja personalizacji w systemie (opracowana na podstawie (Nistor *et al.* 2002)).

Na każdym z poziomów personalizacji możliwe są różne techniki filtrowania informacji. Poszukując tendencji wśród populacji można wykorzystywać techniki typu data-mining, przetwarzające historię zachowań wszystkich użytkowników. Poszukiwanie preferencji typowych dla pewnych grup, zależy od założenia czym tak w istocie są te grupy, dlatego zarówno: klastrowanie z użyciem technik demograficznych i kolaboratywnych może być pomocne. W niniejszej pracy poświęcamy uwagę najwyższemu punktowi „piramidy” personalizacji: informacji indywidualnej.

Nie mniej jednak warto podkreślić, że włączenie do systemu pozostałych poziomów nie powinno nastręczać większych problemów. W ogólnym zarysie proponujemy metodą łączenia technik personalizacji (rozwiązanie hybrydowe[7]).

## 4.4 Ograniczenia środowiska systemu

Proces modelowania użytkownika jest ściśle związany ze środowiskiem i *systemem wspomagania podróży*. Rodzi to oczywiste ograniczenia, które należy rozważyć tworząc rozwiązanie.

[5]Patrz paragraf 1.2.3.
[6]Patrz paragraf 1.3.
[7]Opisywane w paragrafie 1.4.

### 4.4.1 Wpływ interfejsu na rodzaj zbieranej informacji zwrotnej

Istnieją systemy rekomendujące, które potrafią śledzić zachowania na poziomie ruchów myszą i współrzędnych klikania. System do wyszukiwania informacji w sieci, *ELFI* (Pohl i Nick 1999) odnotowuje interakcję z użytkownikiem za pomocą hierarchicznego nawigatora[8], system *TELLIM* (Joerding *et al.* 1998) – za pomocą kontrolki[9]. Jednakże system, w ramach którego budujemy personalizację, narzuca pewne ograniczenia techniczne co do kategorii *zachowań*, które mogą być obserwowane:

1. Przeglądarka WWW renderująca strony w formacie HTML pozwala na komunikację z użytkownikiem za pomocą formularzy i odnośników.

2. Serwer otrzymuje tylko te informacje, które przenoszone są przez protokół HTTP.

Pozwala to na śledzenie historii nawigacji (odnośników, czasu spędzonego na czytaniu strony) w obrębie witryny i danych podawanych w formularzu (ocena, kwestionariusz). Natomiast niemożliwe staje się odnotowanie takich zachowań jak zapisanie, wydrukowanie dokumentu, powiększanie okna, cofnięcie się do poprzednio oglądanej strony czy dodanie odnośnika do listy „Interesujące" w przeglądarce.

Użycie przeglądarki internetowej ogranicza możliwość zbierania sugestii negatywnych, których uniemożliwia użycie algorytmów typu Machine Learning[10].

### 4.4.2 Wpływ współdzielonej inicjatywy na interfejs

Rozważając autonomiczność agenta[11] doszliśmy do wniosku, że może to rodzić wątpliwości dotyczące interakcji z równie autonomicznym użytkownikiem. Wywnioskowaliśmy, że w systemie rekomendującym, oprócz *implicit feedback*, powinien funkcjonować też *explicit feedback*, który wymaga od systemu podejmowania inicjatywy[12]. W ogólności, problem ten określony jest w literaturze mianem *współdzielonej inicjatywy*[13], kiedy to różne współistniejące jednostki mają swoje cele, do osiągnięcia których potrzebne jest przejmowanie inicjatywy. Dzieje się tak na przykład w systemach dialogowych, w których naturalny dialog użytkownik-system, dyktowany jest przez tego pierwszego. Od czasu do czasu system może żądać od użytkownika podania pewnych danych, czy wykonania wielokrokowego scenariusza, co prowadzi przede wszystkich do problemów natury socjologicznej, a nie technicznej (Norman 1994). Należą do nich:

- ograniczenie autonomii użytkownika, możliwości jego decydowania o postępowaniu systemu,

- utrata przez użytkownika poczucia kontroli i odpowiedzialności za podejmowane akcje[14],

[8] Ang. *hierarchical nagivator*.
[9] Ang. *widget*.
[10] Patrz paragraf 1.2.3.
[11] Patrz paragraf 2.3.2.
[12] Paragraf 1.2.3.
[13] Ang. *mixed-initiative* (MI).
[14] Maes, Pattie, Miller, Jim i Shneiderman, Ben. 1997. *Intelligent Software Agents vs. User-Controlled Direct Manipulation: A Debate*. `http://www.acm.org/sigchi/chi97/proceedings/panel/jrm.htm`.

- utrata zaufania do funkcjonowania i wyników działania systemu[14] (Norman 1994).

Jak twierdzi Shneiderman tylko stosowanie bezpośredniej manipulacji[15] rozwiązuje te problemy, jednak w pewnych sytuacjach system powinien mieć możliwość przejęcia inicjatywy, oczywiście za przyzwoleniem użytkownika. Dzieje się tak na przykład w systemach rekomendujących, kiedy to do dobrego modelowania profilu użytkownika wymagana jest znajomość jego opinii na temat różnych zjawisk czy przedmiotów. Mówimy więc o potrzebie otrzymania *bezpośredniej informacji zwrotnej* (*explicit feedback*)[16].

Od strony technicznej problemu MI należy rozważyć w jakim kontekście technologicznym powstaje aplikacja. W przypadku komunikacji opartej na protokole HTTP nie ma możliwości bezpośredniego inicjowania nowego scenariusza przez system. Można co prawda przechwytywać żądania użytkownika i zwracać strony ze rozpoczętym przez system scenariuszem, prowadziłoby to jednak to utraty kontroli nad aplikacja przez użytkownika. Rozwiązaniem jest technologia *wypchnij i ściągnij*[17], zaproponowana w systemie informacyjnym opisanym w (Chou *et al.* 2003). W aplikacji tej użytkownicy sieci komórkowej informowani są o nowych, interesujących ich zdarzeniach w postaci krótkich anonsów-odnośników (wypychanie informacji), a jeśli są zainteresowani jednym z nich, mogą połączyć się z systemem poprzez odnośnik (ściąganie informacji). Zaletą tego rozwiązania jest danie w ręce użytkownika decyzji o przerwaniu jego scenariusza i wyboru jednego z proponowanych.

## 4.5 Problem inicjalizacji profilu. Tworzenie stereotypów

W proponowanym rozwiązaniu do inicjalizacji profilu użytkownika wybrałem metodę stereotypowania[18]. Efektywność tej metody zależy w dużej mierze od jakości stereotypów (Kobsa *et al.* 2001). W tym celu tworzone stereotypy można oprzeć na komercyjnych danych udostępnianych przez firmy lub wynikach sondaży prowadzonych przez ośrodki badania opinii społecznej. W przypadku, gdy takich danych brak, możemy podjąć się samodzielnego przeprowadzania sondażu wśród potencjalnych użytkowników. Rzetelne stereotypy można również uzyskać analizując, za pomocą algorytmów data mining, dane zebrane w trakcie dłuższej interakcji użytkowników z systemem. Można również oprzeć się na własnym doświadczeniu i literaturze specjalistycznej tworząc stereotypy "ręcznie"[19]. Poniżej rozważymy wady i zalety każdego z tych podejść.

### 4.5.1 Wykorzystanie danych komercyjnych

Wiele systemów rekomendujących opiera się na komercyjnych danych do stworzenia stereotypów. Dla przykładu LifeStyle Finder tworzy stereotypy styli życia opierając się na danych demograficznych takich jak np. posiadanie psa, oglądanie kanału telewizyjnego HBO czy comiesięczne zakup kanadyjskiej whisky. Informacje te pochodzą z

[15] Ang. *direct manipulation*.
[16] Kwestia ta poruszana była w paragrafie 1.2.3.
[17] Ang. *push-and-pull*.
[18] Zobacz paragraf 1.2.2.
[19] Ang. *hand-crafted*.

systemu do zarządzania konsumentami, który z kolej bazuje na danych z amerykańskiego spisu ludności, danych o prenumeratach i zakupach oraz wynikach ankiet na próbie ponad 40 tysięcy ludzi (Krulwich 1997). W przypadku gastronomii badania na temat profilu klienta prowadzone są tylko przez największe sieci restauracji i nie są udostępniane potencjalnej konkurencji[20].

### 4.5.2 Wykorzystanie wyników sondaży

W przypadku rynku gastronomicznego nie udało mi się znaleźć niekomercyjnych danych, pozwalających ocenić preferencje kulinarne zależne od wieku czy płci badanego. Sondaże przeprowadzane przez ośrodki badania opinii społecznej pozwalają jedynie ocenić zachowania dominujące w danej populacji (przykładowo, Polacy w porównaniu do Francuzów czy Niemców prawie w ogóle nie jadają poza domem (Andrzejewska 2005)) lub jedynie aspekty ekonomiczne (na przykład, że bezdzietne pary wydają więcej pieniędzy na jedzenie poza domem, niż rodziny z dziećmi[21]). W związku z tym mają one małe znaczenie praktyczne.

### 4.5.3 Analiza danych zebranych w trakcie interakcji z systemem

Analiza taka polega na poddaniu działaniu algorytmu typu data mining danych o zainteresowaniach użytkowników zebranych w pewnym okresie interakcji użytkowników z systemie. Algorytm taki dzieli użytkowników na grupy o podobnych preferencjach, a następnie dla każdej grupy tworzy uogólniony profil grupowy, który możemy nazwać stereotypem. Podobne rozwiązanie wykorzystywane jest w systemie Doppelgänger (Orwant 1995). Wadą takiego rozwiązania jest konieczność zebrania dużej ilości danych od użytkowników.

### 4.5.4 Przeprowadzenie ankiety i analiza jej wyników

W przypadku braku możliwości uzyskania dostępu do gotowych danych (na temat profili klientów lokali gastronomiczny) można przeprowadzić własną ankietę. Kwestia ta wiąże się przede wszystkim z dużym nakładem czasowym, którego wymaga organizacja całego przedsięwzięcia (Babbie 2003).

#### Problem praktyczny

Po pierwsze należy przygotować właściwe pytania. W tym celu oparłem się na literaturze dotyczącej rynku gastronomii i pomocy specjalistki[22]. Ankieta powinna składać się z dwóch części: charakterystyki restauracji i profilu klienta restauracji. Charakterystyka musi obejmować wyłącznie te dane, które występują w ontologii restauracji (utworzonej na podstawie bazy danych z Chefmoz).

[20] Powyższą opinię otrzymałem od specjalistów w tej branży: mgr Joanny Ochniak z Wyższej Szkoły Hotelarstwa i Gastronomii w Poznaniu i dr hab. Andrzeja Masłowski z Instytutu Rynku Wewnętrznego i Konsumpcji w Warszawie. Zapytywane przeze mnie sieci restauracji (Starbucks, McDonald's i KFC) odsyłały do ogólnikowych informacji statystycznych o małym znaczeniu praktycznym.

[21] *Restaurant Spending*: `http://www.restaurant.org/research/consumer/spending.cfm`.

[22] Wspomnianej już mgr Joanny Ochniak z Wyższej Szkoły Hotelarstwa i Gastronomii w Poznaniu.

Po drugie należy określić próbę ankietowanych. Próba ta musi być reprezentatywna, to znaczy wszyscy ankietowani muszą mieć takie same szanse, że do zostaną wybranii do tej próby (Babbie 2003). Nie może być zatem mowy o wybieraniu tylko tych restauracji, które leżą w centrum (kryterium dostępności). Dodatkowo podejście, w którym ankietowani są nie klienci restauracji, ale prowadzący je wydaje się lepsze ze względu na krótszy okres czasu potrzebny do przeprowadzenia badania i możliwość poznania preferencji klientów przychodzących do lokali w różnych godzinach (kryterium dostępności).

Po trzecie należy przeprowadzić analizę wyników i wyciągnąć wnioski. Należy zatem odpowiedzieć na pytanie: jakiego rodzaju klienci (określeni przez cechy demograficzne) odwiedzają restauracje określonego typu. Etap ten wymaga przyporządkowania ustalonym typom restauracji określonych cech, przykładowo autorzy w (Sala 2004) zakładają, że restauracja japońska posiada ogrody piaskowe i fontanny, serwuje kuchnię syczuańską, potrawy grillowane i smażone, a atmosferę tworzy rytualna obsługa i dekoracja wnętrz. Innymi słowy, żeby zbudować stereotyp użytkownika i odpowiedzieć na pytanie, kto chodzi do restauracji japońskich, musimy wpierw stworzyć stereotyp japońskiej restauracji.

#### Problem techniczny

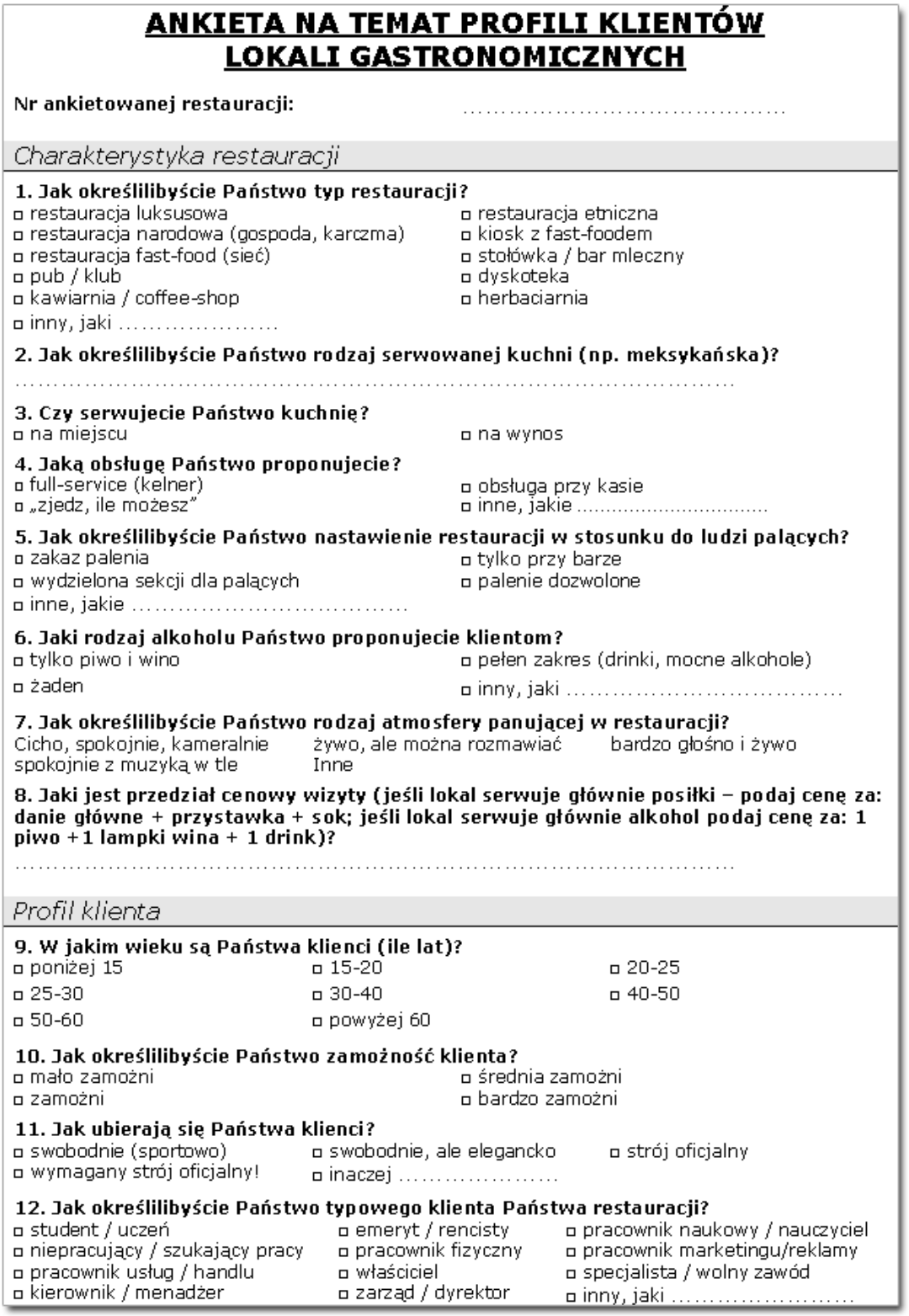
Problem techniczny wynika z braku wszystkich informacji o restauracjach. Baza danych dostarczana przez Chefmoz posiada najczęściej informacje o adresie restauracji, rodzaju serwowanej kuchni i akceptowanych formach płatności. Brakuje najczęściej opisu atmosfery, wysokości cen posiłków, a więc cech, które zostały zdefiniowane w ontologii restauracji.

#### Ankieta

Na rysunku 4.3 przedstawiam ankietę przygotowaną na potrzeby projektu. Przeprowadzenie całego procesu sondażu wymaga dużego nakładu czasowego i wiedzy potrzebnej do przeanalizowania jego wyników. W związku z tym zdecydowałem się na ręczne stworzenie stereotypów.

### 4.5.5 Ręczne tworzenie

Na rok 2001 wszystkie stereotypy w systemach modelujących użytkownika powstawały w sposób „ręczny", ale w oparciu o obserwację danych empirycznych (na przykład informacji o zakupach użytkowników) (Kobsa *et al.* 2001). Nie podparcie się takimi danymi może prowadzić do sytuacji, gdy skonstruowane stereotypy będą niezgodne z rzeczywistością. W jednym ze znanych mi systemów grupa „ekspertów", pracująca na adaptacją interfejsu użytkownika, poszukiwała stereotypu dla pań użytkowniczek. Niestety, przekonanie, że różowy kolor jest ulubionym większości kobiet okazało się mitem.



**ANKIETA NA TEMAT PROFILI KLIENTÓW LOKALI GASTRONOMICZNYCH**

**Nr ankietowanej restauracji:** .....

*Charakterystyka restauracji*

**1. Jak określilibyście Państwo typ restauracji?**
- ▫ restauracja luksusowa
- ▫ restauracja narodowa (gospoda, karczma)
- ▫ restauracja fast-food (sieć)
- ▫ pub / klub
- ▫ kawiarnia / coffee-shop
- ▫ inny, jaki .....
- ▫ restauracja etniczna
- ▫ kiosk z fast-foodem
- ▫ stołówka / bar mleczny
- ▫ dyskoteka
- ▫ herbaciarnia

**2. Jak określilibyście Państwo rodzaj serwowanej kuchni (np. meksykańska)?**

.....

**3. Czy serwujecie Państwo kuchnię?**
- ▫ na miejscu
- ▫ na wynos

**4. Jaką obsługę Państwo proponujecie?**
- ▫ full-service (kelner)
- ▫ „zjedz, ile możesz"
- ▫ obsługa przy kasie
- ▫ inne, jakie .....

**5. Jak określilibyście Państwo nastawienie restauracji w stosunku do ludzi palących?**
- ▫ zakaz palenia
- ▫ wydzielona sekcji dla palących
- ▫ inne, jakie .....
- ▫ tylko przy barze
- ▫ palenie dozwolone

**6. Jaki rodzaj alkoholu Państwo proponujecie klientom?**
- ▫ tylko piwo i wino
- ▫ żaden
- ▫ pełen zakres (drinki, mocne alkohole)
- ▫ inny, jaki .....

**7. Jak określilibyście Państwo rodzaj atmosfery panującej w restauracji?**

Cicho, spokojnie, kameralnie | żywo, ale można rozmawiać | bardzo głośno i żywo
spokojnie z muzyką w tle | Inne

**8. Jaki jest przedział cenowy wizyty (jeśli lokal serwuje głównie posiłki – podaj cenę za: danie główne + przystawka + sok; jeśli lokal serwuje głównie alkohol podaj cenę za: 1 piwo +1 lampki wina + 1 drink)?**

.....

*Profil klienta*

**9. W jakim wieku są Państwa klienci (ile lat)?**
- ▫ poniżej 15
- ▫ 15-20
- ▫ 20-25
- ▫ 25-30
- ▫ 30-40
- ▫ 40-50
- ▫ 50-60
- ▫ powyżej 60

**10. Jak określilibyście Państwo zamożność klienta?**
- ▫ mało zamożni
- ▫ zamożni
- ▫ średnia zamożni
- ▫ bardzo zamożni

**11. Jak ubierają się Państwa klienci?**
- ▫ swobodnie (sportowo)
- ▫ wymagany strój oficjalny!
- ▫ swobodnie, ale elegancko
- ▫ inaczej .....
- ▫ strój oficjalny

**12. Jak określilibyście Państwo typowego klienta Państwa restauracji?**
- ▫ student / uczeń
- ▫ niepracujący / szukający pracy
- ▫ pracownik usług / handlu
- ▫ kierownik / menadżer
- ▫ emeryt / rencisty
- ▫ pracownik fizyczny
- ▫ właściciel
- ▫ zarząd / dyrektor
- ▫ pracownik naukowy / nauczyciel
- ▫ pracownik marketingu/reklamy
- ▫ specjalista / wolny zawód
- ▫ inny, jaki .....

Rysunek 4.3: Ankieta na temat profili klientów lokali gastronomicznych.

# Rozdział 5

# Rozwiązanie

Przedstawiam praktyczne rozwiązanie procesu modelowania użytkownika wykorzystującego ontologię domeny jako podstawę do reprezentacji i uczenia profilu. Rozwiązanie to składa się z dwóch zasadniczych części: (a) teoretycznej, przedstawiającej proces modelowania użytkownika w terminach matematycznych oraz (b) praktycznej, opisującej proces włączone do ramach *systemu wspomagania podróży* funkcji modelowania użytkownika.

## 5.1 Mechanizm modelowania użytkownika

Poszukując metody modelowania użytkownika, postawiłem jej następujące wymagania:

- musi wykorzystywać ontologię domeny[1],
- musi opierać sie wyłącznie na pozytywnych sugestiach ze strony użytkownika i w związku z tych wykorzystuje przystosowany do tego celu algorytm[2],
- musi wykorzystywać bezpośrednią informację zwrotną[3],
- musi inicjalizować profil użytkownika w celu rozwiązania problemu zimnego startu[4],
- musi być skalowalna pod kątem liczby użytkowników i rozmiaru ontologii domeny (ze względu na jej przyszły rozwój).

Metoda zaproponowana przez Finka i Kobsa'e (Fink i Kobsa 2002) spełnia te warunki w znacznej części. W celu spełnienia wszystkich założonych wymogów dokonałem w niej następujących adaptacji i innowacji:

- użyłem ontologii zamiast prostej taksonomii do reprezentacji wiedzy i profilu użytkownika oraz odpowiednio zmodyfikowałem algorytm uczenia,
- zaprojektowałem nowy algorytm inicjalizacji profilu oparty na stereotypowaniu,

[1] Patrz paragraf 4.
[2] Patrz paragraf 1.2.3. oraz paragraf 4.4.1.
[3] Patrz paragraf 1.2.3.
[4] Patrz paragraf 1.3.

- uwzględniłem, obok funkcjonującej już w metodzie uczernia Finka i Kobsa'y obsługi informacji pośredniej, także obsługę informacji bezpośredniej (ale wyłącznie pozytywnej – ze względu na wymgania algorytmu uczenia),
- zaprojektowałem nowy algorytm rekomendacji eksploatujący nie tylko profil użytkownika, ale także bieżący kontekst użytkownika.

### 5.1.1 Reprezentacja profilu

Każdy zarejestrowany użytkownik systemu (identyfikowany przez unikalne *uid*) posiada $Profil$ składający się z *części opisowej*, czyli *Danych*, określających: jego wiek (datę urodzenia), zamożność, sposób ubierania i profesję oraz *części prognozowanej*, stanowiącej *ZbiorOpinii* użytkownika o konceptach *Domeny*.

$$Profil_{uid} = [Dane_{uid}, ZbiorOpinii_{uid}]$$
$$Dane_{uid} = [wiek, zamoznosc, ubior, profesja]$$
$$ZbiorOpinii_{uid} = \left\{Opinia_{uid}^{koncept}\right\}_{koncept \in Domena}$$

*Opinia* użytkownika wyraża zainteresowanie lub jego brak danym *konceptem*.

$$klasyfikacja \in \{interesujacy, nie\_interesujacy, nieokreslony\}$$
$$Opinia_{uid}^{koncept} = [koncept, p_{uid}^{i}, p_{uid}^{n}, p_{uid}^{w}, klasyfikacja]$$

Proces prognozowania opinii zachodzi w trakcie uczenia profilu, kiedy to następuje $klasyfikacja$ zainteresowania na podstawie wyliczonych prawdopodobieństw $p_{uid}^{i}$, $p_{uid}^{n}$, $p_{uid}^{w}$ [5].

**Przykład** Poniżej prezentujemy przykład oparty na stworzonej ontologii profilu użytkownika[6]. Załóżmy, że wybrany użytkownik o imieniu Karol (identyfikowany przez $uid = 14$) jest artystą i dlatego w procesie rejestracji wybrał profesję *SpecialistFreeLancer*. Ma 24 lata, jest zamożny (*Rich*) i ubiera się naturalnie (*NaturalDress*);

```
:KarolProfile a sys:UserProfile;
  :hasUserID            14;
  :hasUserProfileData   :KarolProfileData;
  :hasOpinionsSet       :KarolOpinions.

:KarolProfileData a :UserProfileData;
  :hasAge               24;
  :hasWealth            sys:Rich;
  :hasDress             sys:NaturalDress;
  :hasProfession        sys:SpecialistFreeLancer.
```

W dalszej części opisu rozwiązania przekonamy się, jakie opinie ma wybrany użytkownik (*KarolOpinions*).

### 5.1.2 Inicjalizacja profilu

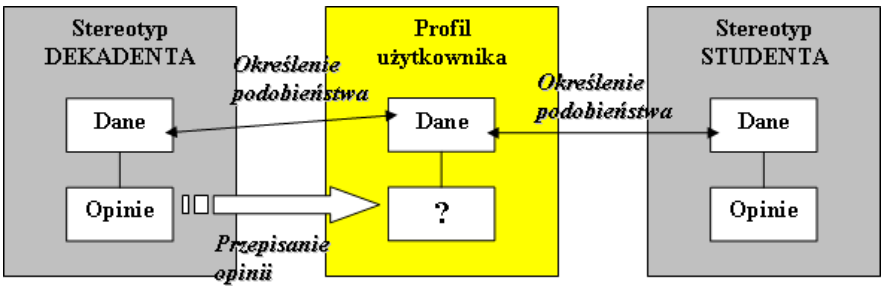
Aby rozwiązać problem zimnego startu[7] do inicjalizacji profilu wykorzystany został mechanizm *stereotypowania*.

[5] Zostaną one dokładnie opisane w paragrafie 5.1.4.
[6] opisanej w załączniku B.3
[7] Opisywany w 1.3.

### Stereotypowanie

Stereotypem nazywamy profil użytkownika skonstruowany przez projektanta systemu. Tak jak każdy profil, składa się on z *Danych* i *ZbioruOpinii*. Mówimy na przykład o *stereotypie artysty*[8], którym jest człowiek w wieku 20 do 35 lat, pracujący najczęściej w wolnym zawodzie. Charakteryzuje go naturalny lub elegancki ubiór oraz niewielka zamożność.

Dopasowanie stereotypu następuje przez porównanie $Danych_{stereotypu}$ z $Danymi_{uzytkownika}$, które użytkownik podał w momencie rejestracji. Opinie stereotypu najbardziej podobnego do użytkownika przeniesione zostają do jego profilu. Proces ten został przedstawiony na rysunku 5.1. W przypadku dopasowania użytkownika do stereotypu artysty, w profilu użytkownika znajdą się pozytywne opinie o lokalach serwujących kawę, herbatę, wino i słodycze, a także negatywne – na temat restauracji typu fast-food.




Rysunek 5.1: Proces stereotypowania.

**Konstrukcja stereotypu i obliczanie podobieństwa między danymi** Dane dwóch użytkowników będziemy określać symbolami $\hat{u}$ i $\hat{w}$, natomiast dane ustalonego stereotypu przez $\hat{S}$. Siłą stereotypu jest zdolność do opisywania więcej niż jednego użytkownika. Dzieje się to za sprawą struktury *Danych*, w których elementy mogą przyjmować więcej niż jedną wartość, w przeciwieństwie do elementów danych użytkownika. Wiek może być określony za pomocą przedziału, a profesja – zbiorem możliwych zawodów. Będziemy pisać, że $\hat{u}^{(f)} \in \hat{S}^{(f)}$, jeśli atrybut danych użytkownika mieści się w zakresie dopuszczanym przez stereotyp, na przykład wiek mieści się w określonym przedziale. Moglibyśmy powiedzieć, że użytkownik $u$ należy do stereotypu $S$, wtedy i tylko wtedy, gdy $\hat{u}^{(f)} \in \hat{S}^{(f)}$ dla każdego $f$. Ale przy tak „idealistycznym" podejściu mogłoby się okazać, że nie potrafimy dopasować do użytkownika żadnego ze znanych stereotypów. Tą niedogodność rozwiążemy wprowadzając pojęcie odległości ($d$) między danymi stereotypu ($\hat{S}$) a danymi użytkownika ($\hat{u}$), polegające na obliczeniu odległości między wartościami poszczególnych atrybutów. Każdy z atrybutów może przyjmować wartości jednego z typów[9]:

1. *Nominalnego*, rozróżniającego kategorie, czyli na przykład profesję nauczyciela od profesji bankowca. O elementach typu nominalnego można powiedzieć jedynie, że

[8] Stereotypy wymyślone na potrzebny tekstu są fikcyjne. Piszę o tym dalej.
[9] Teoria klasyfikacji typów danych oparta jest na tekście (Periklis 2002).

są sobie równe ($\hat{u}^{(f)} = \hat{w}^{(f)}$) lub nie ($\hat{u}^{(f)} \neq \hat{w}^{(f)}$). Stanowią one generalizację typu binarnego do domeny o więcej niż dwóch wartościach dyskretnych.

2. *Porządkowego*, różniącym się od nominalnego, bo pozwalającym uporządkować wartości ($\hat{u}^{(f)} < \hat{w}^{(f)}$, $\hat{u}^{(f)} = \hat{w}^{(f)}$, $\hat{u}^{(f)} > \hat{w}^{(f)}$). Niemożliwe jest natomiast obliczenie odległości pomiędzy poszczególnymi elementami. Możliwe do osiągnięcia wartości nazywamy stanami i w celu ich uporządkowania przyporządkowujemy każdemu rangę: $r_i^f \in [1, 2, \ldots, M^f]$ oznacza rangę wartości $i$ atrybutu $f$. Przykładowo, dla atrybutu zamożności możliwe stany to: mało zamożny ($r_0^f = 1$), średnio zamożny ($r_1^f = 2$), zamożny ($r_2^f = 3$), bardzo zamożny ($r_3^f = 4$). Mówimy o tym, że atrybut $\hat{u}^{(f)}$ ma wartość o randze: $r(\hat{u}^{(f)}) = r_i^f$. Symbolem $M^f$ oznaczamy rozmiar *domeny* atrybutu $f$, czyli ilość możliwych stanów: w przykładzie $M^f = 4$.

3. *Przedziałowego*[10], mierzącego wartości w sposób liniowy. Typ ten wzbogaca poprzedni typ o możliwość obliczania odległości między wartościami ($\hat{u}^{(f)} - \hat{w}^{(f)}$). Przykładem może być tutaj wiek użytkownika.

Ponieważ wartości atrybutów w danych stereotypu są zbiorami, określimy analogiczne typy:

1. Dla typów nominalnego i porządkowego określamy zbiór możliwych wartości: $\hat{S}^{(f)} = \left\{\hat{s}_1^{(f)}, \hat{s}_2^{(f)}, \ldots, \hat{s}_n^{(f)}\right\}$.

2. Dla typu przedziałowego określamy granice dopuszczalnego przedziału: $\hat{S}^{(f)} = \langle y_l, y_r \rangle$.

Zatem odległość $d(\hat{S}, \hat{u})$ definiujemy następująco:

$$d(\hat{S}, \hat{u}) = \frac{\sum_{f=1}^{k} w^f \delta_{\hat{S}\hat{u}}^f d_{\hat{S}\hat{u}}^f}{\sum_{f=1}^{k} w^f \delta_{\hat{S}\hat{u}}^f}$$

gdzie:

- wskaźnik $\delta_{\hat{S}\hat{u}}^f = 0$, jeśli brakuje jednego z elementów: $\hat{S}^{(f)}$ lub $\hat{u}^{(f)}$, to znaczy nie zostały określone w danych stereotypu lub użytkownika (ponieważ nie jest konieczne określania wszystkich atrybutów, na przykład ubioru); w przeciwnym wypadku $\delta_{\hat{S}\hat{u}}^f = 1$,

- $d_{\hat{S}\hat{u}}^f$ jest miarą odległości między $\hat{S}$ a $\hat{u}$ na poziomie atrybutu $f$, określoną następująco:

  - jeśli $f$ jest typu nominalnego, to: $d_{\hat{S}\hat{u}}^f = 0$, jeśli $\hat{u}^{(f)} \in \hat{S}^{(f)}$; w przeciwnym wypadku $d_{\hat{S}\hat{u}}^f = 1$;

[10] W przytaczanym tekście (Periklis 2002) autorzy opisuje również typ *ratio*, którego wartości w porównaniu do przedziałowego mają charakter bezwzględny. Tu traktujemy oba typy jako jeden.

  - jeśli $f$ jest typu przedziałowego, to: $d^f_{\hat{S}\hat{u}} = 0$, jeśli $\hat{u}^{(f)} \in \hat{S}^{(f)}$; w przeciwnym wypadku $d^f_{\hat{S}\hat{u}} = \frac{|m-\hat{u}^{(f)}|}{max^f - min^f}$, gdzie $m = \frac{1}{2}(y_1+y_2)$, a $max^f$ i $min^f$ to wartości maksymalna i minimalna (odpowiednio), możliwe do przyjmowania przez atrybut $f$.
  - jeśli $f$ jest typu porządkowego, to: $d^f_{\hat{S}\hat{u}} = 0$, jeśli $\hat{u}^{(f)} \in \hat{S}^{(f)}$; w przeciwnym wypadku dla $d = min_{\hat{s}^{(f)} \in \hat{S}^{(f)}} |r(\hat{s}^{(f)}) - r(\hat{u}^{(f)})|$ obliczamy: $d^f_{\hat{S}\hat{u}} = \frac{d}{M^f - 1}$.

- $w^f$ stanowi wagę istotności atrybutu $f$.

**Dobór atrybutów demograficznych i wag istotności** Na jakość stereotypowania wpływa dobór atrybutów demograficznych i wag świadczących o istotności każdego z nich przy znajdywaniu najbliższego stereotypu. Problem ten jest znacznie szerszy, ponieważ dotyczy podziału społeczeństwa na segmenty o określonych stylach życia. Badania pokazują, że arbitralne ustalenie uniwersalnych kryteriów takiego podziału może nie być możliwe (Hawkins *et al.* 1998). Przykładowo, w Polsce segmenty ludzi o określonych stylach życia mogą być inne niż wśród mieszkańców USA czy Grecji. Co więcej, nawet gdyby udało się ustalić normatywne kryteria takiego podziału dla mniejszej populacji (na przykład mieszkańców miast w Polsce), to stereotypowanie mogłoby prowadzić do błędnej klasyfikacji. Jest tak, ponieważ – jak twierdzą psychologowie Friedman i Reed – każdy człowiek ma własną teorię klasyfikacji (Reed i Friedman 1973). Innymi słowy: może nie istnieć konsensus co do przewagi wpływu jednego atrybutu nad wpływem innego przy podziale ludzi na ustalone grupy i na przykład część ludzi może uznać, że to wiek cechuje o odrębności grupy, a część – że wysokość dochodów. W praktyce, aby otrzymać wiarygodne dane, prowadzi się badania psychograficzne oparte na obszernych kwestionariuszach mierzących wskaźniki demograficzne, postawy, wartości, czynności, zainteresowania itp. Może być też tak, że istotność danego atrybutu zależy od liczby i doboru pozostałych.

Ostatecznie zaproponowałem cechy (atrybuty), które najczęściej pojawiały się w badaniach dotyczących preferencji gastronomicznych oraz w rozmowach z prowadzącymi restauracje na temat klientów, którzy ich odwiedzają. Wagi dobrałem bazując na własnej intuicji (zobacz tabelę 5.1). Należy pamiętać, że proces stereotypowania, choć może prowadzić do zbyt ogólnych lub nawet błędnych rekomendacji, to stanowi jednak tylko wstępną klasyfikację użytkownika, zanim kolejne techniki modelowania użytkownika skorygują jego profil.

| *Atrybut ($f$)* | *Typ* | *waga ($w^f$)* |
|---|---|---|
| wiek | przedziałowy | 2 |
| zamożność | porządkowy | 4 |
| sposób ubioru | porządkowy | 1 |
| profesja | nominalny | 2 |

Tabela 5.1: Wagi określające istotność atrybutu przy liczeniu odległości między danymi stereotypu a danymi użytkownika.

**Algorytm** Mając już zdefiniowaną miarę odległości $d$, możemy podać algorytm stereotypowania dla użytkownika o danych $\hat{u}$:

1. Znajdź *Stereotyp* o danych $\hat{S}$, dla którego $d(\hat{S}, \hat{u})$ jest najmniejsza.

2. Dla każdej
$$Opinii_{koncept} = [koncept, \ldots, klasyfikacja]$$
określonej w *Stereotypie* utwórz odpowiednią opinię w $Profilu$ użytkownika. Prawdopodobieństwa występujące w *Opinii* nie są w tym momencie istotne i zostaną określone w procesie uczenia profilu.

**Stereotypy** Zdecydowałem się na przygotowanie stereotypów w oparciu o zdobytą wcześniej wiedzą z dziedziny gastronomii i własne doświadczenie. Należy pamiętać, że stanowią one, zgodnie z definicją stereotypu, jedynie generalizację pewnych zachowań czy preferencji i w poszczególnych przypadkach mogą nie być trafne.

Każdy stereotyp w systemie, podobnie jak użytkownik, jest reprezentowany przez profil, którego ontologia zdefiniowana została w załączniku B.4. Składa się z danych demograficznych i określających sposób życia (`StereotypeProfileData`) oraz zbioru charakterystycznych opinii (`OpinionsSet`). Dla pierwszego z prezentowanych stereotypów, stereotypu *młodego człowieka*, podałem reprezentację wiedzy o nim (w notacji N3).

**Stereotyp młodego człowieka** *Młody człowiek* to uczeń w wieku do 15 lat. Nie ma jeszcze wyrobionego gustu co do preferowanego stroju. Zasobność jego portfela określona jest zamożnością jego rodziców i stąd obejmuje szeroki przekrój możliwych wartości.

*Młody człowiek* lubi hamburgery, hot-dogi, pizzę – jednym słowem jedzenie typu fast-food. Najlepiej, jeśli szukana restauracja oferuje menu specjalnie dla młodych klientów, podaje posiłki na powietrzu i posiada warunki do przyjmowania dużych grup znajomych (na przykład duże stoły). Młody człowiek płaci gotówką.

```
:YoungsterData a sys:StereotypeProfileData;
  sys:hasStereotypeID 1;
  sys:hasName "Youngster";
  sys:hasProfessionSet [sys:contains sys:StudentPupil, sys:
      UnemployedJobSeeker];
  sys:hasWealthSet [sys:contains sys:Rich, sys:NotRich, sys:
      AverageRich];
  sys:hasAgeSet [sys:hasLeftBound 0; sys:hasRightBound 15].

:YoungsterOpinions a sys:OpinionsSet;
  sys:containsOpinion
    [sys:about res:FastFoodCuisine;
    sys:hasClassification sys:Interesting],
    [sys:about res:HamburgerCuisine;
    sys:hasClassification sys:Interesting],
    [sys:about res:HotDogsCuisine;
    sys:hasClassification sys:Interesting],
```

```
    [sys:about res:PizzaCuisine;
    sys:hasClassification sys:Interesting],

    [sys:about mon:Cash;
    sys:hasClassification sys:Interesting],

    [sys:about res:KidFriendly;
    sys:hasClassification sys:Interesting],
    [sys:about res:Kids;
    sys:hasClassification sys:Interesting],
    [sys:about res:Kiosk;
    sys:hasClassification sys:Interesting],
    [sys:about res:LargeGroupsOk;
    sys:hasClassification sys:Interesting],
    [sys:about res:Outdoor;
    sys:hasClassification sys:Interesting],
    [sys:about res:Family;
    sys:hasClassification sys:Interesting];

  sys:hasStereotypeID 1.
```

**Stereotyp artysty** *Artysta* to dorosły człowiek, najczęściej w wieku między 20 a 35 lat. Pracuje jako specjalista w określonej dziedzinie albo wykonuje wolny zawód. Nie jest zamożny, ubiera się naturalnie lub elegancko.

*Artysta* w restauracjach szuka przede wszystkim sposobu na spędzenie czasu, rozmowę i stąd preferuje lokale serwujące kawę, herbatę, ciastka. Gustuje też w winiarniach. Krytycznie ocenia jedzenie typu fast-food.

**Stereotyp młodego sportowca** Młody sportowiec to najczęściej człowiek między 15 a 25 rokiem życia. Ubiera się sportowo, zatrudnia się najczęściej jako pracownik fizyczny. Może być też studentem (uczniem) lub bezrobotnym (poszukującym pracy). W związku z tym nie dysponuje dużymi pieniędzmi.

Sportowiec lubi się dobrze zabawić (lokale oferujące rozrywkę dla dorosłych i muzykę) i szybko zjeść – w swoich opiniach jest więc przeciwieństwem artysty. Jedyne co ich łączy to zamiłowanie do alkoholu, aczkolwiek młody sportowiec woli piwo. Nie ma nic przeciwko lokalom dyskryminującym pewne mniejszości społeczne.

**Stereotyp studenta** Student to człowiek między 18 a 26 rokiem życia, niezamożny i ubierający się w sposób naturalny.

Preferuje restauracje serwujące bardzo różnorodną kuchnię, od pizzy, poprzez jedzenie meksykańskie po potrawy kuchni azjatyckiej. Dobrze, jeśli taki lokal oferuje jedzenie na wynos lub dostarcza do domu. Do posiłków koniecznie musi być podawane piwo. Dla studenta restauracja, klub czy pub stanowi jednocześnie miejsce spędzania wolnego czasu i dobrze, jeśli można tam posłuchać muzyki, skorzystać z Internetu czy spróbować nowego trunku. Student płaci głównie gotówką.

**Stereotyp snoba** Snob, określany też mianem nowobogackiego, w krótkim okresie czasu zdobył duży majątek: ma od 25 do 40 lat i jest zamożny. Ubiera się elegancko. Pracuje najczęściej na wysokim stanowisku jako biznesmen, dyrektor przedsiębiorstwa lub pracownik reklamy.

Snob brzydzi się jedzeniem typu fast-food i polską kuchnią narodową. Jest za to eklektykiem: próbuje kuchni z różnych stron świata: Grecji, Włoch i Chin. Lubi dobre, markowe wina. Płaci wyłącznie kartą kredytową lub debetową.

**Stereotyp konserwatysty** Konserwatysta to dojrzały człowiek, powyżej 45 roku życia. Jest najczęściej pracownikiem fizycznym lub emerytem. Chociaż nie jest zamożny, to ubiera się elegancko.

Konserwatysta nie lubi nowości: nie gustuje w kuchni zagranicznej, nie lubi lokali typu fast-food i niechętnie spogląda na lokale, w których płaci się kartą płatniczą. Uwielbia za to polską kuchnię narodową, tłuste potrawy mięsne, zakrapiane mocnym alkoholem.

**Przykład** Prześledźmy teraz jak zostałby sklasyfikowany wybrany użytkownik. Okazuje się, że najbliżej mu do stereotypu artysty (tabela 5.2): $d(\hat{S}, \hat{u}) = 0.148$. W związku z czym wśród opinii w jego profilu pojawiają się następujące, przeniesione ze stereotypu:

| *Atrybut (f)* | *Dane* stereotypu artysty *($\hat{S}$)* | *Dane użytkownika ($\hat{u}$)* | *Zważona odległość ($w^f d^f_{\hat{S}\hat{u}}$)* |
|---|---|---|---|
| wiek | 20-35 | 24 | $0 = 2 \cdot 0$ |
| zamożność | niezamożny lub średnio zamożny | zamożny | $1.33 = 4 \cdot \frac{d}{4-1}$, gdzie $d = min(\|3-1\|, \|3-2\|) = 1$ |
| sposób ubioru | naturalny lub elegancki | naturalny | $0 = 1 \cdot 0$ |
| profesja | specjalista/wolny zawód | specjalista/wolny zawód | $0 = 2 \cdot 0$ |
| | | *Razem ($d(\hat{S}, \hat{u})$)* | $0.148 = \frac{0+0.1.33+0+0}{2+4+1+2}$ |

Tabela 5.2: Przykład obliczania odległości między danymi stereotypu artysty a danymi wybranego użytkownika ($uid = 14$), który wyłamuje się ze stereotypu artysty jedynie pod względem zamożności.

```
:KarolOpinions      a sys:OpinionsSet;
  sys:containsOpinion
    [sys:about res:CafeCoffeeShopCuisine;
      sys:hasClassification sys:Interesting;
      sys:hasNormalizedProbability 1 0].
    [sys:about res:CafeteriaCuisine;
      sys:hasClassification sys:Interesting;
      sys:hasNormalizedProbability 1.0].
    [sys:about res:TeaHouseCuisine;
      sys:hasClassification sys:Interesting;
      sys:hasNormalizedProbability 1.0].
```

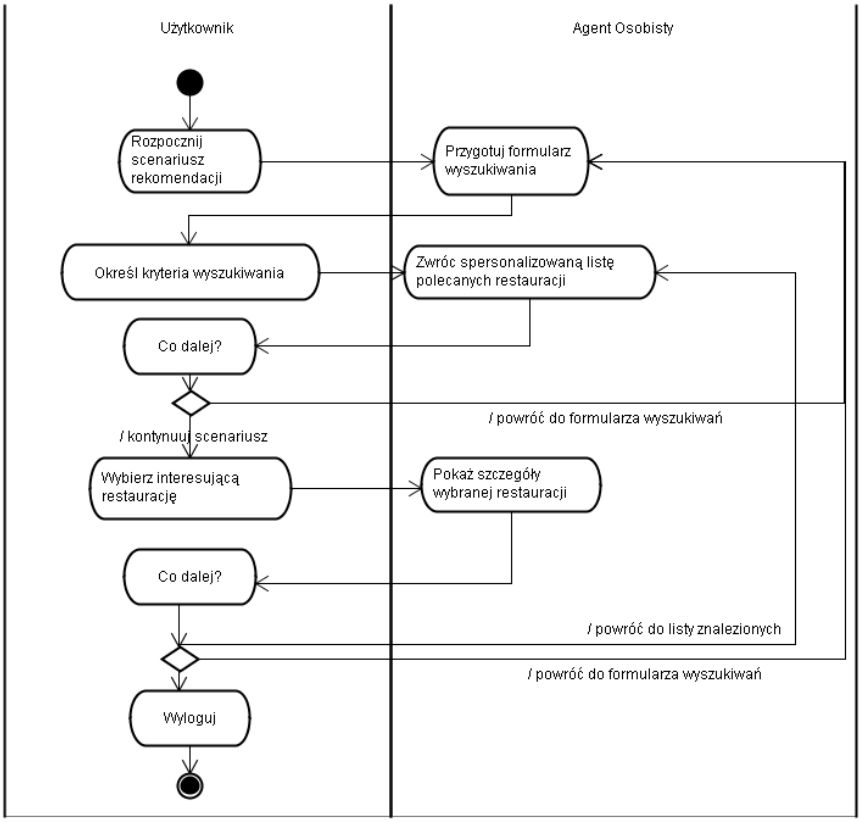
```
[sys:about res:WineBeer;
  sys:hasClassification sys:Interesting;
  sys:hasNormalizedProbability 1.0].
[sys:about res:WineList;
  sys:hasClassification sys:Interesting;
  sys:hasNormalizedProbability 1.0].
[sys:about res:WineTasting;
  sys:hasClassification sys:Interesting;
  sys:hasNormalizedProbability 1.0].
[sys:about res:Winery;
  sys:hasClassification sys:Interesting;
  sys:hasNormalizedProbability 1.0].
[sys:about res:ExtensiveWineList;
  sys:hasClassification sys:Interesting;
  sys:hasNormalizedProbability 1.0].
[sys:about res:Entertainment;
  sys:hasClassification sys:Interesting;
  sys:hasNormalizedProbability 1.0].
[sys:about res:FastFoodCuisine;
  sys:hasClassification sys:NotInteresting;
  sys:hasNormalizedProbability 0.0].
[sys:about res:HamburgerCuisine;
  sys:hasClassification sys:NotInteresting;
  sys:hasNormalizedProbability 0.0].
[sys:about res:HotDogsCuisine;
  sys:hasClassification sys:NotInteresting;
  sys:hasNormalizedProbability 0.0].
[sys:about res:PizzaCuisine;
  sys:hasClassification sys:NotInteresting;
  sys:hasNormalizedProbability 0.0].
```

Wybrany użytkownik, jak zakładamy, będzie poszukiwał kawiarni i herbaciarni, unikał natomiast będzie restauracji typu fast-food. W dalszej części opisu okaże, się czy te przewidywania się sprawdzą.

### 5.1.3 Informacja zwrotna

Informacja zwrotna posłuży do wyuczenia profilu. W celu zebrania takiej informacji system wysyła do użytkownika formularz, w którym może on określić swoje bieżące potrzeby co do preferowanej restauracji (rysunek C.3). W wyniku zapytania użytkownik otrzymuje spersonalizowaną listę proponowanych restauracji. Lista ta nie zawiera wszystkich informacji o restauracjach i aby je uzyskać, użytkownik wybiera jedną, która wyda mu się interesująca. Następnie może wrócić do listy znalezionych restauracji. W każdej chwili użytkownik może przerwać wyszukiwanie i powrócić do początku scenariusza lub wylogować się z systemu. Cały scenariusz przedstawiony jest na rysunku 5.2.

W dowolnym momencie użytkownik może również zdecydować się na odpowiedzenie na jedno z pytań z proponowanych przez AO. Pytania te dotyczą restauracji poleconym przez system w trakcie ostatniej sesji. Zakłada się bowiem, że od jej czasu użytkownik odwiedził co najmniej jedną z proponowanych restauracji. Użytkownik może oddać na restaurację jedynie pozytywny głos lub wstrzymać się od głosowania (rysunek C.4).




Rysunek 5.2: Diagram aktywności dla scenariusza rekomendowania restauracji.

Taka strategia podyktowana jest wykorzystanym algorytmem uczenia profilu, który analizuje wystąpienie jedynie pozytywnych zdarzeń.

### Bezpośrednia i pośrednia informacja zwrotna

Zgodnie z wnioskami wyciągniętymi w paragrafie 1.2.3 wykorzystujemy w systemie oba rodzaje informacji zwrotnej: *bezpośrednią* (explicit) i *pośrednią* (implicit). Przez informację bezpośrednią rozumiemy głosy oddane na co najmniej jedną z proponowanych wcześniej restauracji (będziemy to zachowanie określać mianem *głosowania*). Informacje pośrednia to kryteria wyszukiwania określone przez użytkownika (*zapytanie*) oraz restauracje wybierane w celu uzyskania dokładnych danych o nich (*wybranie*). Wszystkie te zachowania świadczą w sposób pozytywny o danej restauracji i jej cechach.

Należy w tym momencie podkreślić, że użytkownik nie jest zmuszany do podania informacji bezpośredniej[11]. W tym celu wykorzystywana jest została zaadaptowana technika *wypchnij-i-zwróć*[12]. W jej wyniku do każdej odpowiedzi systemu przygotowywane są wcześniej spreparowane pytania, a właściwie ich skrócone wersje (patrz rysunek C.3). Użytkownik sam podejmuje decyzje czy kontynuować rozpoczęty scenariusz, czy też przerwać go i odpowiedzieć na zadane pytanie.

[11]Kwestia korzyści tego podejścia została podana w paragrafie 1.2.3 przy okazji omawiania systemu *Anatagomy*.

[12]Opisana w paragrafie **??**.

**Polityka zadawania pytań**

Prześledźmy teraz przykładowe postępowanie użytkownika w trakcie scenariusza rekomendowania restauracji:

1. Użytkownik podaje kryteria (Q1) wyszukiwania.
2. System zwraca wyniki spersonalizowaną listę restauracji (L1), spełniającą kryteria (Q1).
3. Użytkownik wybiera z listy jedną z restauracji (R1).
4. Następnie wraca do listy znalezionych restauracji (L1) o wybiera restaurację (R2).
5. Decyduje się na powtórne określenie kryteriów wyszukiwania (Q2).
6. System zwraca listę L2, spełniającą kryteria Q2.
7. Użytkownik wybiera z listy restaurację R3.
8. Wylogowywuje się z systemu.

Możemy założyć, że po obejrzeniu szczegółowych informacji o R1 użytkownik, stwierdził, że mu się nie podoba. Pojawiają się również pytania: (a) czy po obejrzeniu danych o restauracji R2 użytkownik zdecydował się na powtórne określenie kryteriów wyszukiwania (Q2), bo nie był zadowolony z zaproponowanych rezultatów ? oraz (b) czy wylogowując się z systemu użytkownik zrezygnował się, czy też restauracja R3 wydała się na tyle interesująca, że dalsze wyszukiwanie nie było konieczne? Innymi słowy: czy każda akcja kończąca (*wylogowanie*) lub rozpoczynają wyszukiwanie (*rozpoczęcie-wyszukiwania*) jest sugestią pozytywną, czy negatywną? Burke (Burke 2002), przy okazji projektowania systemu *EntreeC*, założył, dość ryzykowanie, że pozytywną i umieścił ją na liście zachowań możliwych do obserwowania (zaproponowaną przez Nicholsa (Nichols 1998)[13]) na 9 miejscu istotności.

W związku z tym w kolejnej sesji AO powinien zapytać użytkownika o opinię na temat restauracji R2 i R3.

**Algorytm przygotowywania pytań** W momencie wylogowania się z systemu dokonywana jest analiza zachowań użytkownika odnotowanych w trakcie ostatniej sesji. Na wierzch stosu pytań zostają odłożone pytania o restauracją, o której szczegółowe dane pytał użytkownik tuż przed zakończeniem lub powtórnym rozpoczęciem procesu wyszukiwania.

**Reprezentacja zebranej informacji zwrotnej**

Wspomnieliśmy, że w trakcie interakcji z systemem użytkownik może wykonywać różne *zachowania*:

$$zachowanie \in \{glosowanie, zapytanie, \\ wybranie, rozpoczecie, wylogowanie\}$$

[13]Lista ta omówiona została w paragrafie 1.2.3.

Zachowanie takie zachodzi w ściśle określonym *Kontekscie*, czy to zdefiniowanym przez URI konceptu wybranej lub ocenionej restauracji, czy to przez słowa kluczowe wyznaczające kryteria zapytania. Definiujemy zatem:

$$Kontekst = \{Parametr\}$$
$$Parametr = [nazwa, wartosc]$$

gdzie nazwa jest konceptem należącym do *Domeny*, a wartość – konceptem albo należącym do *Domeny* albo podanym wprost przez użytkownika literałem (np. miasto *„Poznań"*). Ostatecznie określamy:

$$RozszerzoneZachowanie = [zachowanie, Kontekst]$$

*Zachowanie* wykonane przez użytkownika identyfikowanego przez *uid* w obrębie pewnej sesji (*sid*) o godzinie wyznaczanej przez *czas* nazywamy *Zdarzeniem* i dołączamy do *Historii* wszystkich zdarzeń.

$$Zdarzenie^{(czas,uid)} = [czas, uid, sid, RozszerzoneZachowanie]$$
$$Historia = \left\{ Zdarzenie^{(czas,uid)} \right\}_{(czas,uid)}$$

**Przykład** W tabeli 5.6 widzimy, że wybrany użytkownik (*uid* = 14) zaraz po rozpoczęciu wyszukiwania poprosił o przygotowanie rekomendacji restauracji podających jedzenie typu fast-food (wbrew przewidywaniom stereotypu) w Poznaniu. Wśród poleconych restauracji znalazła się restauracja „Avanti" [14]. Oprócz żądanego „szybkiego jedzenia" serwuje ona także kawę, co zapewne spodobało się użytkownikowi, bo postanowił zobaczyć jej szczegóły (zachowanie wyboru). Była to ostatnia, obejrzana przed wylogowaniem, restauracja i dlatego system postanowił zapytać o nią użytkownika w trakcie następnej sesji. Użytkownik zagłosował na nią pozytywnie.

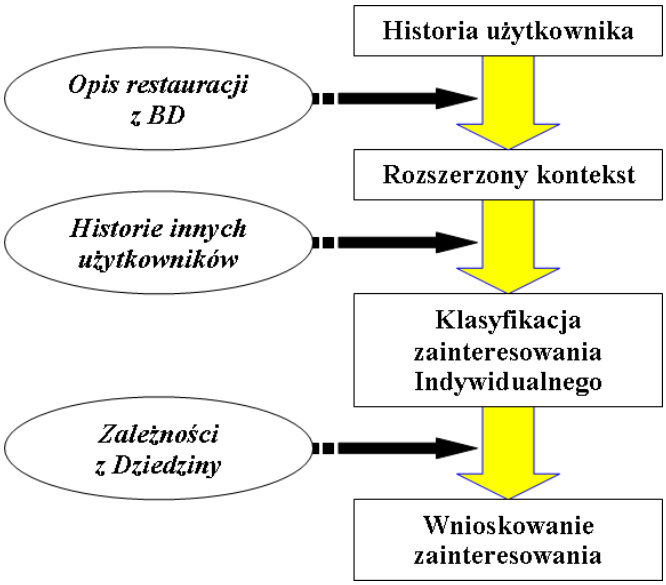
| *Zdarzenie* | | | | |
|---|---|---|---|---|
| *czas* | *uid* | *sid* | *RozszerzoneZachowanie* | |
| | | | *zachowanie* | *Kontekts* |
| 17:32 | 14 | XX | rozpoczęcie | |
| 17:34 | 14 | XX | zapytanie | <res:cuisine>=<res:FastFoodCuisine>, <res:city>="Poznan" |
| 17:42 | 14 | XX | wybranie | <event:hasTargetURI> =<db::Poland_WP_Poznan_Avanti_... |
| 17:45 | 14 | XX | wylogowanie | |
| 13:54 | 14 | YY | rozpoczęcie | |
| 14:04 | 14 | YY | głosowanie | <event:hasTargetURI> =<db::Poland_WP_Poznan_Avanti_... |

Tabela 5.3: Zapis zdarzeń (*Historii*) w dwóch kolejnych sesjach wybranego użytkownika (*uid* = 14). Wbrew przewidywaniom przeprowadzonych w trakcie stereotypowania, użytkownik preferuje także kuchnię typu fast-food.

[14] Opisana w paragrafie 3.1.1.

### 5.1.4 Uczenie profilu

Proces uczenia, przedstawiony na rysunku 5.3, składa się z trzech etapów: (a) rozszerzenia kontekstu, (b) klasyfikacji indywidualnej i (c) wnioskowania na podstawie ontologii domeny. Etap (a) prowadzi do rozszerzenie *Kontekstu* każdego *Zdarzenia* w *Historii* użytkownika, tak aby znalazła w nim się nie tylko referencja do konceptu będącego obiektem zachowania, ale również opis tego konceptu, wyciągnięty z *BD*. Jest oczywistym, że to cechy obiektu decydują o jego wyborze przez użytkownika[15] Przykładowo, jeśli obiektem zachowania jest pub „13 Kotów", honorujący karty MasterCard, to *Kontekst* musi uwzględniać również ten fakt. Etap (b) pozwala ocenić zainteresowanie ustalonego użytkownika (*uid*) względem różnych cech reprezentowanych przez *koncepty* w ontologii. Zainteresowania tego dowodzi częstotliwość występowania różnych zachowań względem danego *konceptu* w historii nawigacji użytkownika. Przykładowo, użytkownik może często pytać o kuchnię chińską, wybierać (klikać) restauracje, które ją serwują. Mówimy wtedy o zachowaniach dotyczących konceptu *kuchni chińskiej*. Na podstawie *Domeny* możemy w trakcie etapu (c) wywnioskować zainteresowanie użytkownika tymi konceptami, które nie zostały sklasyfikowane w poprzednim etapie. Dla przykładu zainteresowanie kuchnią chińską, meksykańską i wieloma innymi kuchniami może wskazywać na fakt, że użytkownik przy wyborze restauracji kładzie nacisk na rodzaj serwowanego jedzenia.




Rysunek 5.3: Proces uczenia.

#### Rozszerzenie kontekstu

Aby rozszerzenie kontekstu było możliwe, należy przygotować odpowiednią strukturę informacji. Chcemy, aby kolejne kolejne etapy uczenia operowały na konceptach zdefiniowanych wyłącznie w *Domenie*, a nie ich realizacjach w postaci konceptów z *BD* czy literałach. W tym celu wprowadzimy pojęcie *refleksji*.

[15] I stąd też techniki wykorzystujące to spostrzeżenie noszą nazwę opartych na atrybutach – paragraf 1.2.4.

**Definicja 5.1.1** *Niech $PrzestrzenKonceptow$ będzie sumą Domeny, $DB$ i możliwych literałów. Wtedy* ***refleksją*** *konceptu nazywamy takie przekształcenie $r : PrzestrzenKonceptow \rightarrow Domena$, że:*

- *jeśli $koncept \in Domena$ to $r(koncept) = koncept$,*
- *jeśli $koncept \notin Domena$, ale jest instancją klasy $klasa_{koncept}$ zdefiniowanej w Domenie, to $r(koncept) = klasa_{koncept}$*
- *w każdym innym przypadku, w tym kiedy koncept jest literałem, wartość $r(koncept)$ jest nieokreślona.*

*Skrótowo zapisujemy $r(koncept) \equiv \overline{koncept}$.*

Ustalmy również pojęcie refleksji dla *Stwierdzenia* opisującego restaurację $DB$.

**Definicja 5.1.2** *Refleksją $Stwierdzenia = [podmiot, orzeczenie, dopenienie]$ nazywamy operację:*

$$\overline{Stwierdzenie} = [\overline{podmiot}, \overline{orzeczenie}, \overline{dopenienie}]$$

*jeśli refleksja na poszczególnych elementach stwierdzenia jest określona.*

**Algorytm** Ustalmy $Zdarzenie^{(czas,uid)} = [czas, uid, sid, [zachowanie, Kontekst]]$.

1. Określamy dla niego $Kontekst^*$, do którego będziemy dodawać koncepty występujące w $Kontekscie$ oraz związane z nimi koncepty z $BD$. Wszystkie są poddawana uprzedniej refleksji (patrz *Algorytm rozszerzania kontekstu*).
2. Dla każdego *konceptu* należącego do $Kontekstu^*$ tworzymy $Zdarzenie^{*(czas,uid)}$:

$$Zdarzenie^{*(czas,uid)} = [czas, uid, sid, zachowanie, koncept]$$

3. Dodajemy wszystkie $Zdarzenia^{*(czas,uid)}$ do $Historii^{*uid}$ użytkownika.

Powyższy algorytm wykonywany jest dla każdego $Zdarzenia$ należącego do $Historii_{uid}$. W jego efekcie otrzymujemy:

$$Historia^{*uid} = \left\{ Zdarzenie^{*(czas,uid)} \right\}_{(czas,uid)}$$

**Algorytm rozszerzania kontekstu** Wyjaśnienia wymaga krok 1. Koncepty dodajemy do $Kontekstu^*$ zależnie od *zachowania*:

- jeśli jest to *zapytanie*, to dla każdego $Parametru = [nazwa, wartosc]$ należącego do $Kontekstu$ dodamy do $Kontekstu^*$ koncepty: $\overline{nazwa}$ oraz $\overline{wartosc}$ (jeśli określona).
- jeśli to *klikniecie* lub *ocena*, to jedyny $Parametr\ Kontekstu$ określa $URI_{restauracji}$. W tym celu dodajemy do $Kontekstu^*$, wszystkie elementy stwierdzeń z $BD$, w których $URI_{restauracji}$ gra rolę podmiotu. Operację tą powtarzamy, rekurencyjnie, to znaczy dla każdego dopełnienia szukamy zdań, w których jest ono podmiotem.

Uściślenia wymaga przypadek, kiedy to dopełnieniem jest anonimowy węzeł RDF, jak na przykład w wyrażeniu:

```
db:Poland_WP_Poznan_13_Kotow_Pub :serves [ :contains: :
    Alcohol] .
```

znaczącym tyle co: pub „13 Kotow" serwuje *coś* (nie wiemy co), co zwiera alkohol. To coś funkcjonuje w ontologii jako *wezęł anonimowy*[16] i nie jest dodawane do $Kontekstu^*$, dodaniu ulegają natomiast zdania z nim związane.

**Przykład** W tabeli 5.4 widzimy dokładnie, o co pyta, co wybiera i na co głosuje wybrany użytkownik ($uid = 14$). Do $Historii^*$ przeniesione zostały koncepty opisujące między innymi restaurację „Avanti".

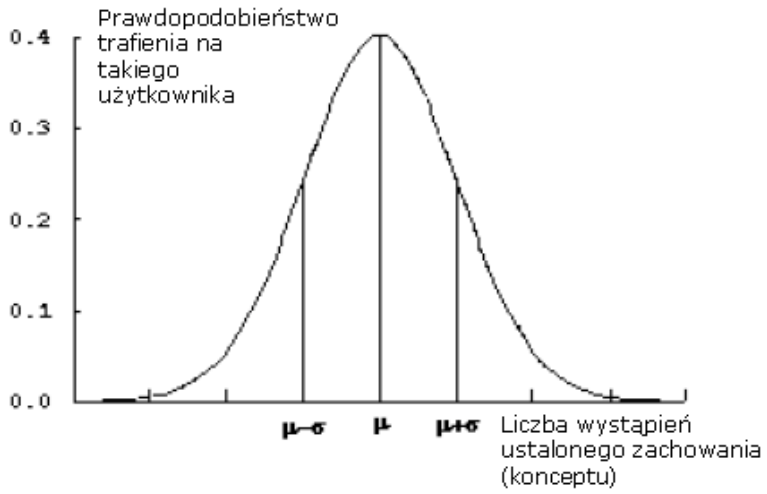
| $Zdarzenie^*$ | | | | |
|---|---|---|---|---|
| *czas* | *uid* | *sid* | *zachowanie* | *koncept* |
| 17:34 | 14 | XX | zapytanie | <loc:city> |
| 17:34 | 14 | XX | zapytanie | <res:cuisine> |
| 17:34 | 14 | XX | zapytanie | <res:FastFoodCuisine> |
| 17:42 | 14 | XX | wybranie | <res:cuisine> |
| 17:42 | 14 | XX | wybranie | <res:FastFoodCuisine> |
| 17:42 | 14 | XX | wybranie | <res:cuisine> |
| 17:42 | 14 | XX | wybranie | <res:CafeteriaCuisine> |
| 17:42 | 14 | XX | wybranie | <res:cuisine> |
| 17:42 | 14 | XX | wybranie | <res:PolishCuisine> |
| 17:42 | 14 | XX | wybranie | <res:alcohol> |
| 17:42 | 14 | XX | wybranie | <res:NoAlcoholServed> |
| 17:42 | 14 | XX | wybranie | <loc:city> |
| | | | ... | |
| 14:04 | 14 | YY | głosowanie | <res:cuisine> |
| 14:04 | 14 | YY | głosowanie | <res:FastFoodCuisine> |
| 14:04 | 14 | YY | głosowanie | <res:cuisine> |
| 14:04 | 14 | YY | głosowanie | <res:CafeteriaCuisine> |
| 14:04 | 14 | YY | głosowanie | <res:cuisine> |
| 14:04 | 14 | YY | głosowanie | <res:PolishCuisine> |
| 14:04 | 14 | YY | głosowanie | <res:alcohol> |
| 14:04 | 97 | YY | głosowanie | <res:NoAlcoholServed> |
| 14:04 | 14 | YY | głosowanie | <loc:city> |

Tabela 5.4: Przetworzony zapis zdarzeń ($Historia^*$). Widzimy, że często pojawia się koncept *FastFoodCuisine*.

### Klasyfikacja indywidualna

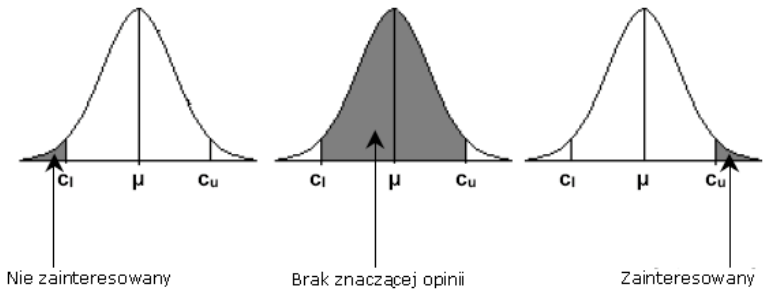
Dla ustalenia uwagi terminem *zachowanie* będziemy posługiwać się w odniesieniu do ustalonego konceptu. Częstotliwość występowania zachowań ma charakter relatywny i jest badana w stosunku do częstotliwości zachowań wszystkich użytkowników.

[16] Berners-Lee, Tim. 1998. *RDF Anonymous nodes and quantification.* `http://www.w3.org/DesignIssues/Anonymous.html`.




Rysunek 5.4: Rozkład normalny zainteresowań użytkownika przy ustalonym zachowaniu (koncepcie) (opracowano na podstawie (Fink i Kobsa 2002)).

Jednowymiarowa analiza istotności[17] jest statystyczną techniką opartą na założeniu, że występowanie zachowania (z wybranym konceptem) w historii nawigacji użytkownika ma rozkład normalny (rysunek 5.4). Jeśli koncept (zachowanie) pojawia się w indywidualnej historii użytkownika rzadziej niż w próbce losowej, użytkownik może nie być nim zainteresowany. I odwrotnie, jeśli pojawia się częściej niż w próbce losowej, zakłada się, że użytkownik jest nim zainteresowany. Aby określić fakt zainteresowania lub jego brak, wprowadzam dwie granice *przedziału ufności*: $c_l$ i $c_p$ dla niższej i wyższej granicy, odpowiednio. Jeśli aktualna liczba wystąpień zachowania w historii użytkownika jest poniżej $c_l$ lub powyżej $c_p$, to klasyfikujemy użytkownika jako preferującego lub nie dane zachowanie (koncept), odpowiednio. W przypadku, gdy wartość mieści się w przedziale pomiędzy $c_l$ i $c_p$, nie można sklasyfikować jednoznacznie użytkownika. Wszystkie trzy przypadki przedstawiono na rysunku 5.5.




Rysunek 5.5: Klasyfikacja zainteresowania użytkownika (opracowano na podstawie (Fink i Kobsa 2002)).

**Algorytm** W oparciu o te obserwacje wykonywany jest następujący algorytm dla każdego *konceptu* z ontologii:

1. Oszacuj *prawdopodobieństwo ogólne* ($p$) zainteresowania *konceptem* opierając się

[17] Tłumaczenia angielskich terminów z teorii prawdopodobieństwa podaję za (Oktaba 1969).

na historii wszystkich użytkowników ($Historia^*$).

2. Oszacuj *prawdopodobieństwo indywidualne* ($p^i_{uid}$) preferowania *konceptu* opierając się na historii badanego użytkownika ($Historia^*_{uid}$).

3. Wykorzystując *prawdopodobieństwo ogólne* i *prawdopodobieństwo indywidualne* oblicz prawdopodobieństwo zainteresowania konceptem danego użytkownika w relacji do rozkładu wystąpień tego *konceptu* u wszystkich użytkowników (dalej zwane *prawdopodobieństwem znormalizowanym,* $p^n_{uid}$*).*

4. Jeśli to możliwe, sklasyfikuj użytkownika jako zdecydowanie zainteresowanego lub nie zainteresowanego *konceptem.*

Pamiętamy, że różne rodzaje zachowań w różnym stopniu świadczą o zainteresowaniu użytkownika (paragraf 1.2.3). Należy odpowiednio wzmacniać wpływ określonego rodzaju zachowań na prawdopodobieństwo zainteresowania.

**Model matematyczny** Ustalmy *koncept*. Na zmienną $Historia^*$ możemy patrzeć, jak na sekwencję prób schematu Bernoulliego[18], w którym za pojedynczy sukces uznajemy występowania dowolnego rodzaju zachowania wobec ustalonego *konceptu*. Niech $k^{*koncept}$ będzie liczbą tych sukcesów $Historii^*$, a $N^*$ liczbą wszystkich zachowań w $Historii^*$. Schemat Bernoulliego możemy teraz scharakteryzować jako $b(N^*, k^{*koncept}, p)$, gdzie *prawdopodobieństwo ogólne* pojedynczego sukcesu szacujemy:

$$p(koncept) = \frac{k^{*koncept}}{N^*}$$

Analogicznie, aby obliczyć *prawdopodobieństwo indywidualne*, określamy $k^{*koncept}_{uid}$ jako liczbę zachowań wobec *konceptu* w $Historii^*_{uid}$, a $N^*_{uid}$ – jako liczbę wszystkich zachowań w historii użytkownika. Wtedy:

$$p^i_{uid}(koncept) = \frac{k^{*koncept}_{uid}}{N^*_{uid}}$$

W przypadku dużej liczby prób i sukcesów rozkład dwumianowy (Bernoulliego) $b(N, k, p)$ można przybliżyć rozkładem normalnym $N(np, np(1-p))$. W naszym przypadku szacujemy wartość oczekiwaną ($\mu$) i odchylenie standardowe ($\sigma$) w stosunku do $Historii_{uid}$ ustalonego użytkownika następująco:

$$\mu = p(koncept)N^*_{uid}$$
$$\sigma = \sqrt{p(koncept)(1 - p(koncept))N^*_{uid}}$$

Szukamy takiej konstrukcji matematycznej, która pozwoli nam ocenić prawdopodobieństwo zainteresowania użytkownika danym *konceptem* w relacji do występowania *konceptu* w zachowaniach wszystkich użytkowników. Ustandaryzujemy liczbę zachowań $k^{*koncept}_{uid}$ w relacji do całej populacji:

$$\overline{k^{*koncept}_{uid}} = \frac{k^{*koncept}_{uid} - \mu}{\sigma}$$

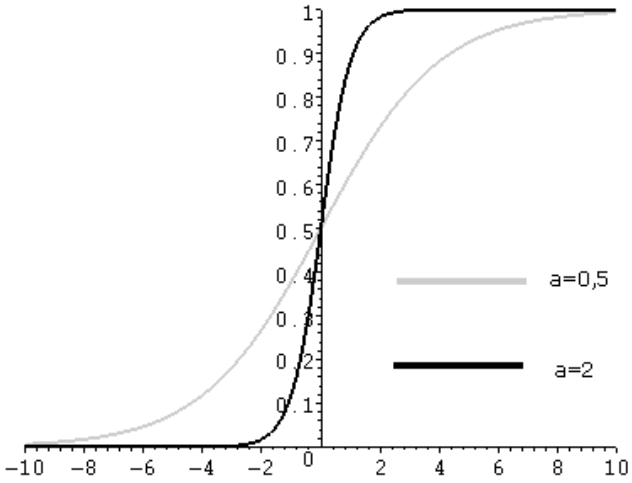
[18]Wszystkie oznaczenia z teorii prawdopodobieństwa pochodzą z (Carlson i Thorne 1997).

i określimy

$$p^n_{uid}(koncept) = f(\overline{k^{*koncept}_{uid}})$$

gdzie $f(t)$ jest funkcją sigmoidalną (rysunek 5.6) daną wzorem[19]:

$$f(t) = \frac{1}{1+e^{-at}}, a > 0$$




Rysunek 5.6: Wykres funkcji sigmoidalnej w zależności od parametru $a$.

Funkcja $f(t)$ jest odpowiednia, ponieważ:

1. Jej przeciwdziedzina jest określona na przedziale $[0, 1]$.
2. Inaczej niż w przypadku funkcji gęstości rozkładu normalnego, prawdopodobieństwo skrajnego braku zainteresowania przyjmuje wartości odmienne (niskie) do prawdopodobieństwa skrajnego zainteresowania (wartości wysokie).

Wiemy, że $k^{*koncept}_{uid}$ ma rozkład normalny z szacowaną wartością oczekiwaną $u$ i odchyleniem standardowym $\sigma$. Wykorzystując metodę estymacji przedziałowej wyznaczamy granice *przedziału ufności*:

$$c_{l/u} = \mu \mp Z_{\alpha/2}\sigma$$

gdzie $Z_{\alpha/2}$ jest statystyką z tablicy rozkładu normalnego.

Dokonujemy klasyfikacji *konceptu* następująco:

$$klasyfikacja(koncept) = \begin{cases} interesujacy & k^{*koncept}_{uid} > c_u \\ nie\ interesujacy & k^{*koncept}_{uid} < c_l \\ nie\ okreslony & \text{w pozostałym wypadku} \end{cases}$$

[19]W pracy (Fink i Kobsa 2002) współczynnik $a = 0.4$ został zaadaptowany przez autorów z prac nad wykorzystaniem uczenia maszynowego do modelowania użytkownika (zwłaszcza (Kobsa *et al.* 2000; Schwab i Pohl 1999)). Co do proponowanej funkcji sigmoidalnej: w komercyjnej wersji systemu, autorzy używają podobnej, aczkolwiek nie tej samej formuły.

Pamiętamy o tym, aby przed obliczeniem prawdpodobieństw $p^i_{uid}(koncept)$ i $p^n_{uid}(koncept)$ wzmocnić wpływ rodzaju zachowania na prawdopodobieństwo zainteresowania. Ustalmy wektor wag:

$$W = \langle w_{zachowanie_1}, w_{zachowanie_2}, \ldots \rangle$$

Niech $k^{zachowanie,koncept}_{uid}$ będzie liczbą zachowań (o ustalonym obiekcie *koncept*) w $Historii^*_{uid}$. Wtedy łączna liczba zachowań względem *konceptu* jest liczona za pomocą średniej ważonej:

$$k^{*koncept}_{uid} = \frac{\sum\limits_{zachowanie} w_{zachowanie} k^{zachowanie,koncept}_{uid}}{\sum\limits_{zachowanie} w_{zachowanie}}$$

Analogicznie określamy:

$$N^*_{uid} = \frac{\sum\limits_{zachowanie} w_{zachowanie} N^{zachowanie}_{uid}}{\sum\limits_{zachowanie} w_{zachowanie}}$$

$$k^{*koncept} = \frac{\sum\limits_{zachowanie} w_{zachowanie} k^{zachowanie,koncept}}{\sum\limits_{zachowanie} w_{zachowanie}}$$

$$N^* = \frac{\sum\limits_{zachowanie} w_{zachowanie} N^{zachowanie}}{\sum\limits_{zachowanie} w_{zachowanie}}$$

**Wagi istotności zachowań** Zachowania, które obserwuje system, w różnym stopniu odzwierciedlają zainteresowanie użytkownika. Najsilniejszym zachowaniem jest *głosowanie*, wysoko punktowane (waga: 30) jako reakcja wprost[20]. Niżej, ale zgodnie z tabelą oceniają istotność zachowań Nicholsa[21], znajdują się: wybranie (waga: 10) i zapytanie (waga 5).

**Przykład** W systemie jest zarejestrowanych 20 użytkowników, którzy w trakcie interakcji pozostawili w $Historii^*$ $N = 2000$ zdarzeń, w tym udział wybranego użytkownika ($uid = 14$) wyniósł $N_{14} = 100$ zdarzeń[22]. Będziemy chcieli oszacować, jak bardzo prawdopodobne jest, że interesuje się on jedzeniem typu fast-food (koncept *FastFood-Cuisine*).

[20]Istotność informacji typu explicit feedback została podkreślona przy okazji omawiania systemu *Anatagomy* w paragrafie 1.2.3.

[21]Paragraf 1.2.3.

[22]Tablica 5.4 przedstawia jedynie wycinek $Historii^*$.

| *Co obliczamy?* | *Dla populacji* | *Dla wybranego użytkownika* |
|---|---|---|
| zainteresowanie konceptem | $k^{*glosowanie,koncept}$=6<br>$k^{*wybranie,koncept}$=20<br>$k^{*zapytanie,koncept}$=10 | $k_{14}^{*glosowanie,koncept}$=4<br>$k_{14}^{*wybranie,koncept}$=10<br>$k_{14}^{*zapytanie,koncept}$=15 |
| zważone zainteresowanie konceptem | $k^{*koncept}=\frac{<30,10,5>^t<6,20,10>}{30+10+5}$<br>=9.5 | $k_{14}^{*koncept}=\frac{<30,10,5>^t<4,10,5>}{30+10+5}$<br>=5.4 |
| ogólną ilość zachowań | $N^{*glosowanie}$=373<br>$N^{*wybranie}$=801<br>$N^{*zapytanie}$=826 | $N_{14}^{*glosowanie}$=18<br>$N_{14}^{*wybranie}$=43<br>$N_{14}^{*zapytanie}$=39 |
| zważoną ogólną ilość zachowań | $N^*=\frac{<30,10,5>^t<373,801,826>}{30+10+5}$<br>=518.4 | $N_{14}^*=\frac{<30,10,5>^t<18,43,39>}{30+10+5}$<br>=25.8 |
| prawdpodobieństwo | $p(koncept)=\frac{9.5}{518.4}$<br>=0.18 | $p_{14}^i(koncept)=\frac{5.4}{25.8}$<br>=0.21 |

Sprawdzamy, jak bardzo ilość zachowań wybranego użytkownika dotyczących konceptu *FastFoodCuisine* odbiega od zachowań populacji (przy poziomie istotności 95%):

$$\mu = 0.18 \cdot 25.8 = 4.64$$
$$\sigma = \sqrt{0.18 \cdot 0.82 \cdot 25.8} = 1.95$$
$$c_l = 4.64 - 1.96 \cdot 1.95 = 0.82$$
$$c_r = 4.64 + 1.96 \cdot 1.95 = 8.47$$

Okazuje się, że $k_{14}^{*koncept} \in [0.82; 8.47]$, zatem pan użytkownik na tle populacji nie wyróżnia się pod tym względem i nie posiada zdecydowanej opinii. Obliczmy na przyszłość pradwopodobieństwo znormalizowane:

$$\overline{k_{14}^{*koncept}} = \frac{5.4 - 4.64}{1.95} = 0.389$$
$$p_{14}^n(koncept) = \frac{1}{1 + e^{-0.4 \cdot 0.389}} = 0.54$$

**Wnioskowanie na podstawie ontologii domeny**

W celu przeprowadzania wnioskowania wprowadzimy pojęcie relacji zależności między konceptami.

**Definicja 5.1.3** *Mówimy, że $koncept_B$ jest **zależny** od $konceptu_A$ (piszemy: $koncept_A \leftarrow koncept_B$), wtedy i tylko wtedy, gdy zachodzi jedna z poniższych sytuacji:*

- *$koncept_B$ jest podklasą $konceptu_A$ (B rdfs:subClass A)*
- *$koncept_B$ jest podwłaściwością $konceptu_A$ (B rdfs:subProperty A)*
- *$koncept_B$ jest właściwością $konceptu_A$ (B rdfs:property A)*
- *$koncept_B$ jest instancją $konceptu_A$ (B rdfs:type A)*
- *$koncept_B$ określa zakres możliwych wartości $konceptu_A$ (A rdfs:range B)*

Ontologia może być reprezentowana jako graf skierowany, w którym koncepty grają rolę węzłów, a relacja zależności wyznacza krawędzie między nimi. Graf ten nie musi być spójny. Na potrzeby przedstawionego niżej algorytmu zakładamy acykliczność *Domeny* oraz wykluczamy sytuacje, kiedy $koncept_A \leftarrow koncept_B$ i $koncept_C \leftarrow koncept_B$, czyli na przykład gdy $koncept_B$ jest podklasą dwóch różnych konceptów.

Przedstawiony poniżej algorytm opisuje *propagację zainteresowania w górę*, zaczynając od konceptów leżących najniżej w relacji zależności.

**Algorytm** Algorytm jest konfigurowany za pomocą dwóch współczynników:

- współczynnika istotności *prawdopodobieństwa znormalizowanego* $\alpha = 1.0$,
- współczynnika istotności *prawdopodobieństwa wywnioskowanego* $\beta = 0.7$.

Dla każdego $konceptu_A \in Domena$, nie będącego liściem (a więc posiadającym koncepty zależne od niego), obliczamy *prawdopodobieństwo wywnioskowane* ($p^w_{uid}$):

$$p^w_{uid}(koncept_A) = \frac{1}{|I_{koncept_A}|} \sum_{koncept_{B_i} \in I_{koncept_A}} f(koncept_{B_i})$$

gdzie

$$I_{koncept_A} = \{koncept_{B_i} : koncept_A \leftarrow koncept_{B_i} \text{ i użytkownik posiada opinię o } koncept_{B_i}\}$$

$$f(koncept) = \frac{\alpha p^n_{uid}(koncept) + \beta p^w_{uid}(koncept)}{\alpha + \beta}$$

Graf przeszukujemy metodą *przeszukiwania w głąb*[23] (Cormen 2004), rozpoczynając od konceptu stojącego najwyżej w relacji zależności w każdym z podgrafów spójnych grafu *Domeny*.

Następnie dla każdego $konceptu_B$, dla którego $p^w_{uid}(koncept_B)$ jest nieznane ustawiamy:

$$p^w_{uid}(koncept_B) = p^w_{uid}(koncept_A)$$

gdzie $koncept_A \leftarrow koncept_B$.

| Cuisine<br>$p^n_{14} = 0.67$<br>$p^w_{14} =?$ | | | |
|---|---|---|---|
| CuisineCode<br>*(brak opinii)* | | | |
| GreeceCuisine<br>*(brak opinii)* | FastFoodCuisine<br>$p^n_{14} = 0.54$<br>$p^w_{14} =?$ | TeaHouseCuisine<br>$p^n_{14} = 1$<br>$p^w_{14} =?$ | PizzaCuisine<br>$p^n_{14} = 0$<br>$p^w_{14} =?$ |

Tabela 5.5: Początkowy model wybranego użytkownika ($uid = 14$).

| Cuisine<br>$p^n_{14} = 0.67$<br>$p^w_{14} = 0.21$ | | | |
|---|---|---|---|
| CuisineCode<br>$p^n_{14} = 0.0$<br>$p^w_{14} = 0.51$ | | | |
| GreeceCuisine<br>*(brak opinii)* | FastFoodCuisine<br>$p^n_{14} = 0.54$<br>$p^w_{14} = 0.51$ | TeaHouseCuisine<br>$p^n_{14} = 1$<br>$p^w_{14} = 0.51$ | PizzaCuisine<br>$p^n_{14} = 0$<br>$p^w_{14} = 0.51$ |

Tabela 5.6: Końcowy model wybranego użytkownika ($uid = 14$).

**Przykład** Przykładowo obliczamy:

### 5.1.5 Eksploatacja profilu

W scenariuszu rekomendowania restauracji system staje przed zadaniem oszacowania potencjalnego zainteresowania *aktywnego użytkownika* każdą ze znalezionych restauracji. O tym, jak wysokie jest ono, świadczy *temperatura* przedmiotu, obliczna na podstawie:

- cech rozpatrywanej restauracji ($URI_{restauracji}$),
- profilu aktywnego użytkownika ($Profil_{uid}$),
- oraz *bieżącego kontekstu*, czyli zbioru słów kluczowych towarzyszących zapytaniu:

$$BiezacyKontekst = \{Parametr\}$$

**Algorytm** Idea algorytmu jest prosta:

1. Dla każdego rozpatrywanego przedmiotu (restauracji), nazywanego aktywnym, oszacuj *temperaturę* (patrz *Algorytm szacowania temperatury*).
2. Rekomenduj użytkownikowi tylko $\gamma$ najwyżej ocenionych przedmiotów, sortując je według temperatury w porządku malejącym.

[23] Ang. *Depth-First Search*.

Algorytm jest konfigurowany za pomocą:

- *ilości rekomendowanych przedmiotów* $\gamma = 5$,
- *współczynnika istotności prawdopodobieństwa wywnioskowanego* $\alpha = 0.4$,
- *współczynnika istotności kontekstu* $\beta = 0.8$.

**Algorytm szacowania temperatury** Zbiór opinii w profilu użytkownika może być postrzegany jako graf acykliczny, niekoniecznie spójny, w którym rolę węzłów odgrywają opinie o konceptach, a krawędzie wyznaczone są przez relacje zależności z ontologii *Dziedziny*. Szacowanie będzie polegało na wyznaczeniu temperatury aktywnego przedmiotu, będącej sumą temperatur konceptów zawartych w jego opisie. Oto kolejne kroki:

1. W grafie $ZbioruOpinii_{uid}$ oznacz jako *aktywne* te opinie, które dotyczą konceptów występujących w stwierdzeniach o aktywnym przedmiocie, czyli takich, w których $URI_{restauracji}$ pełni rolę podmiotu. Jeśli dopełnienie tego stwierdzenia jest podmiotem w innym zdaniu, operację powtarzamy, rekurencyjnie.

2. Dla każdej $Opinii_{uid}^{koncept}$ definiujemy funkcję temperatury biorąc pod uwagę jej $klasyfikacje$:

$$temp(Opinia_{uid}^{koncept}) = \begin{cases} 1 & \text{gdy } koncept \text{ interesujacy} \\ 0 & \text{gdy } koncept \text{ nie interesujacy} \\ \alpha p_{uid}^{w}(koncept) & \text{w pozostałym wypadku} \end{cases}$$

   o ile opinia o danym koncepcie istnieje w profilu. W przeciwnym wypadku $temp(Opinia_{uid}^{koncept}) = 0$.

3. Dla każdej aktywnej *opinii*, której *koncept* **nie** pojawił się w *Kontekscie* jako *nazwa* lub *wartosc Parametru*, mnożymy temperaturę przez $\beta$.

4. Aktywne opinie, wyznaczają podgraf $ZbiorOpinii_{uid}^{(aktywne)}$. Każde

$$Stwierdzenie = [p, o, d] \in Domena$$

   którego co najmniej jeden z elementów jest obiektem *aktywnych opinii* określamy mianem *aktywnego* i obliczamy jego temperaturę:

$$temp(Stwierdzenie) = temp(Opinia_{uid}p) * temp(Opinia_{uid}^{o}) * temp(Opinia_{uid}^{d}) - 0.5$$

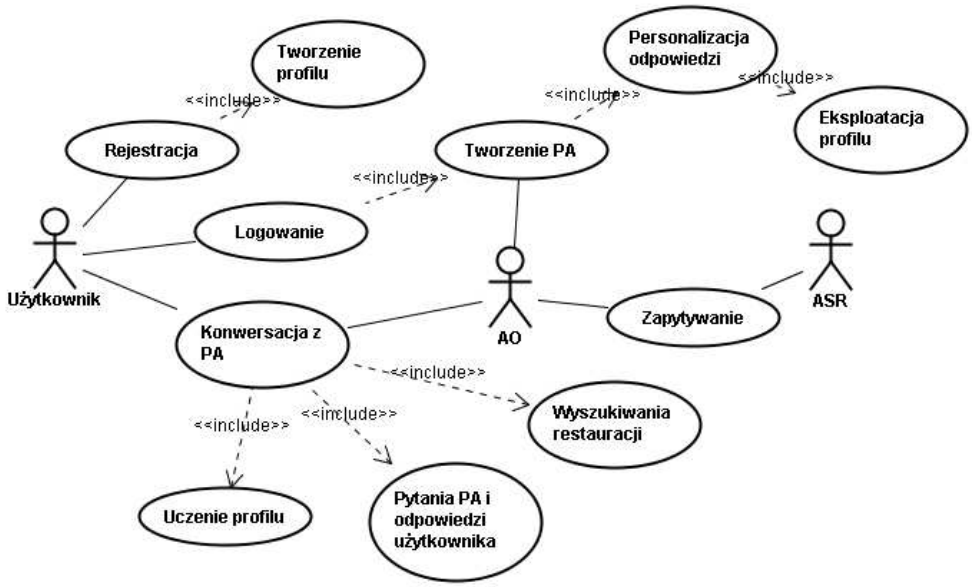
5. Ostateczną temperaturę dla $ZbiorOpinii_{uid}^{(aktywne)}$ wyznaczamy sumując temperatury *aktywnych stwierdzeń*

$$temp(ZbiorOpinii_{uid}^{(aktywne)}) = \sum temp(Stwierdzenie)$$

Wartość $temp(ZbiorOpinii_{uid}^{(aktywne)})$ jest jednocześnie szacowaną temperaturą aktywnego przedmiotu. Należy zauważyć, że $temp(Stwierdzenie) \in [-0.5, 0.5]$ i w związku z tym może zmniejszać ostateczną temperaturę dla $ZbiorOpinii_{uid}^{(aktywne)}$, dając w ten sposób wyraz opiniom o nieinteresujących konceptach.

## 5.2 Projekt podsystemu

Mając gotowy algorytm modelowania użytkownika wyzwaniem staje się jego realizacja w istniejącym środowisku *systemu wspomagania podróży*[24]. W tym celu wykorzystałem metodologię stosowaną przy programowaniu obiektowymi prezentowaną wcześniej metodologię Prometheus[25].




Rysunek 5.7: Diagram przypadków użycia dla podsystemu.

Diagram przypadków użyć przedstawiony został na rysunku 5.7. Każdy z przypadków użyć przekłada się na jedną lub kilka ról. Każdą z tych ról przyporządkowałem jednemu z agentów (rysunek 5.8), z jednym wyjątkiem „obsługi sesji", co podkreślamy w paragrafie A.

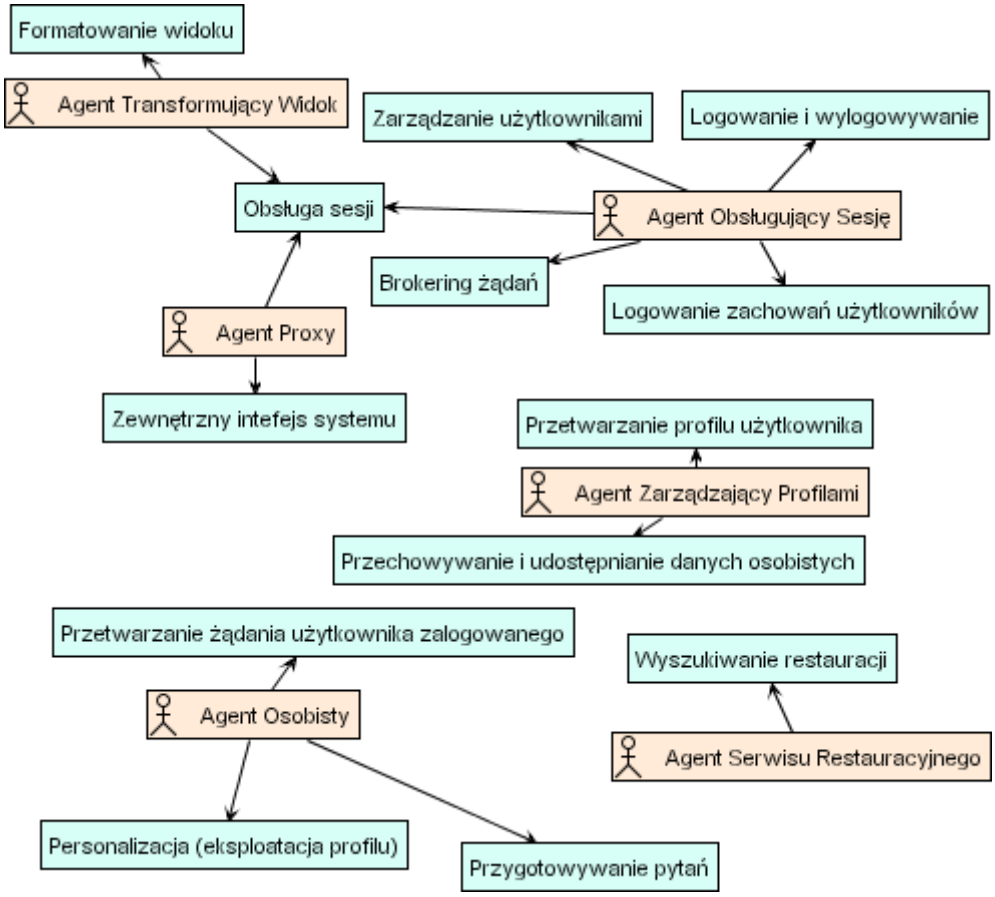
Realizując każdą z tych ról napotkałem trzy zasadnicze problemy:

- w jaki sposób identyfikować użytkownika w systemie[26],
- jak rozdysponować role związane z procesem modelowania użytkownika pomiędzy poszczególnych agentów,
- w jaki sposób agenci mają mieć dostęp do danych ontologicznych i w jaki sposób mogą je między sobą wymieniać.

### 5.2.1 Realizacja procesu modelowania użytkownika

Celem modelowania użytkownika, wynikającym wprost z definicji, jest stworzenie odpowiedniego profilu. W tym celu został w systemie wyróżniony *Agent Zarządzający*

[24] Omawianym w paragrafie 3.
[25] Paragraf 2.3.5.
[26] Odpowiedź na to pytanie znajduje się w paragrafie A.




Rysunek 5.8: Diagram przydziału ról do odpowiednich agentów (według metodologii Prometheus).

*Profilami*[27] (AZP).

AZP pełni dwie role:

- przetwarza profil użytkownika, to jest inicjalizuje go i uczy,
- przechowuje i oferuje innym agentom dostęp do danych osobistych: profile wszystkich użytkowników, profili wszystkich stereotypów oraz pytań pozostawionych przez AO.

W tabeli 5.7 zostały zebrane etapy modelowania użytkownika (według taksonomii przedstawionej w paragrafie 1.2) i agenci odpowiedzialni za ich realizację[28].

### Inicjalizacja profilu profilu

Inicjalizacja profilu została przedstawiona na diagramie A.1 przy okazji omawiania procesu rejestracji.

[27] Ang. *Profile Managing Agent* (PMA).

[28] Omawiając role AOS (paragraf A) określiłem jego udział w modelowaniu użytkownika. Omówiłem również rolę AO przy zadawaniu pytań (paragraf 5.1.3).

| *Etap* | *Agenci odpowiedzialni za* |
|---|---|
| *Inicjalizacja profilu* | AZP dokonuje stereotypizacji na podstawie stereotypów w bazie danych i danych z rejestracji otrzymanych od AOS. |
| *Informacja zwrotna* | AOS zapamiętuje wszystkie potrzebne zachowania użytkownika, także odpowiedzi na pytania, które przygotował AO. |
| *Uczenie profilu* | Przeprowadza AZP na podstawie danych od AOS i ASR. |
| *Eksploatacja profilu* | Dokonuje AO, pobrawszy uprzednio profil od AZP. |

Tabela 5.7: Etapy modelowania użytkownika i ich realizacja w systemie.

### Uczenie profilu

We wstępnych zamierzeniach uczeniem profilu miał zajmować się AO. Takie rozwiązanie posiadało jednak kilka istotnych wad. Po pierwsze, proces uczenia profilu wymaga przetworzenia danych wielu użytkowników, zasadne jest zatem gromadzenie statystyk na temat wszystkich użytkowników. Po drugie – po drugie proces uczenia jest kosztowny ze względu na czas i zasoby, a AO ma przede wszystkim służyć swojemu użytkownikowi bezpośrednią pomocą. Ponadto związanie go obowiązkiem uczenia na dłuższy czas ograniczyłoby jego mobilność.

Dlatego też cały proces uczenia został przerzucony do AZP. AO pobiera tylko profil od AZP w celu personalizacji odpowiedzi. Dzięki temu, czas uczenia profilu nie jest kwestią priorytetową, a jego eksploatacja może być dokonywana względnie szybko[29].

Proces uczenia przedstawia diagram na rysunku 5.9. Proces ten jest konfigurowany przez dwie zmienne:

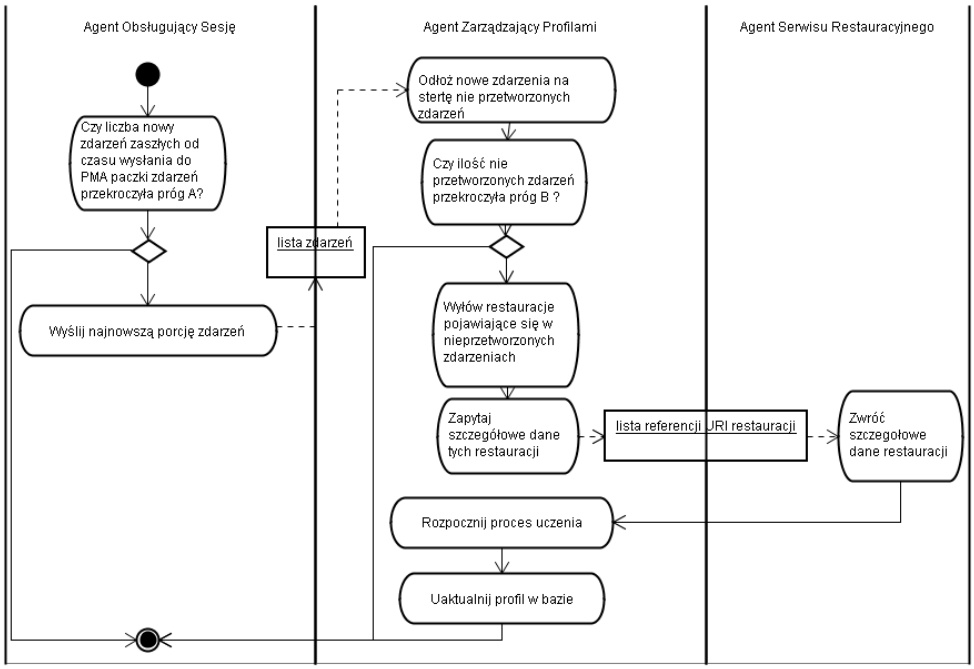
- *próg A*, określający maksymalną dopuszczalną ilość nowych zdarzeń, jaka może odróżniać bazę zdarzeń (`events-model`) znanych przez AOS, a bazą zdarzeń, o których wie AZP.
- *próg B*, określający dopuszczalną różnicę między ilością zdarzeń znanych AZP, a ilością *zdarzeń przepracowanych* w celu nauki profilu.

Wyjaśnienia wymaga termin *zdarzeń przepracowanych*. W wyniku uczenia pojedynczego profilu gromadzimy dane statystyczne nie tylko ma temat ilości zachowań dokonanych przez właściciela profilu, ale także informacje o innych użytkownikach, których statystyki są liczone między innymi w celu normalizacji wyników[30].

### Eksploatacja profilu

Rozważając projektowanie agentów[31] wyróżniłem dwie perspektywy: zewnętrzną i wewnętrzną. Na przykładzie AO możemy zobaczyć w jaki sposób one funkcjonują

[29]Problem ten poruszany był w paragrafie 1.2.1 przy okazji omawianie różnic między reprezentacją *opartą na modelu* i reprezentacja *opartą na pamięci*.
[30]Patrz paragraf 5.1.4.
[31]Zobacz paragraf 2.3.5.




Rysunek 5.9: Diagram aktywności dla scenariusza uczenia profilu.

i wpływają na siebie. Diagram na rysunku 5.10 przedstawia proces obsługi żądania wyszukania restauracji[32] z uwzględnieniem eksloatacji profilu. Diagram stanów na rysunku 5.11 przedstawia perspektywę wewnętrzną obrazującą cykl funkcjonowania AO, włącznie z realizacją obsługi żądania omówionego wyżej.

### 5.2.2 Dostęp agentów do danych ontologicznych

Sercem systemu są ontologie i wykorzystujące je dane semantyczne. Przedyskutujemy teraz postawione rozwiązania dostępu do nich i wiążące się z tym niedogodności.

#### Przydział ról do źródeł danych

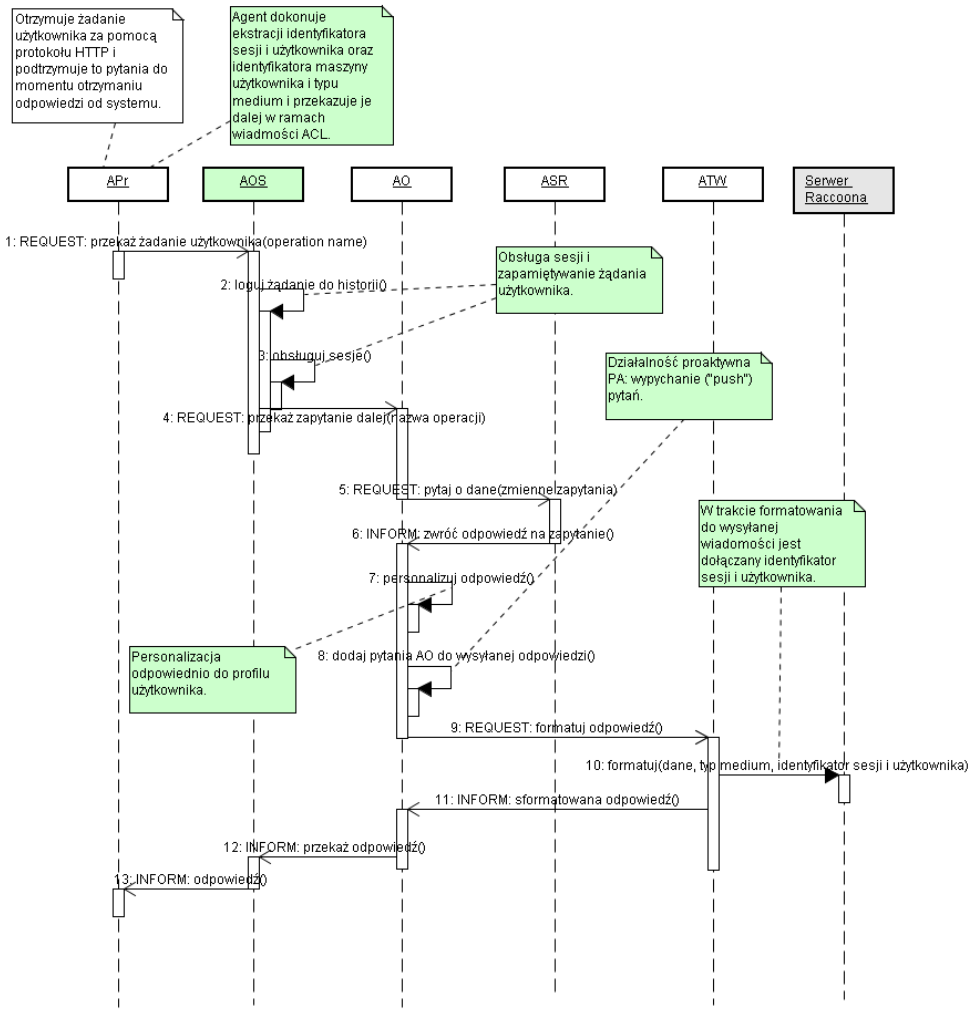
Zgodnie z metodologią Prometheus dokonałem analizy, z jakich danych jakie role najczęściej korzystają. Wyniki zostały przedstawione na rysunku 5.12. Należy pamiętać, że korzystanie z wybranych danych nie jest równoznaczne z bezpośrednim dostęp do nich – komunikacja ze źródłem danych może przebiegać za pomocą agenta stanowiącego interfejs tego źródła. Przykładowo AZP udostępnia agentowi AO profil jego użytkownika. Ostateczny efekt rozumowania przedstawia tabela 5.8.

#### Wzorce projektowe: Obiekt Dostępu do Danych i Obiekt do Transferu

Przyjrzyjmy się dwóm sytuacjom dostępu do danych ontologicznych przechowywanych w modelach Jeny:

1. *Dodanie nowych danych.* AOS otrzymuje nowe żądania od użytkownika i chciałby informacje o nich zapisać do modelu `events-model`. Żądania mają formę obiektów

[32] Warto, aby czytelnik porównał ten diagram z wcześniejszym rysunkiem 3.4 obejmującym Podsystem Dostarczania Informacji bez włączonej personalizacji.



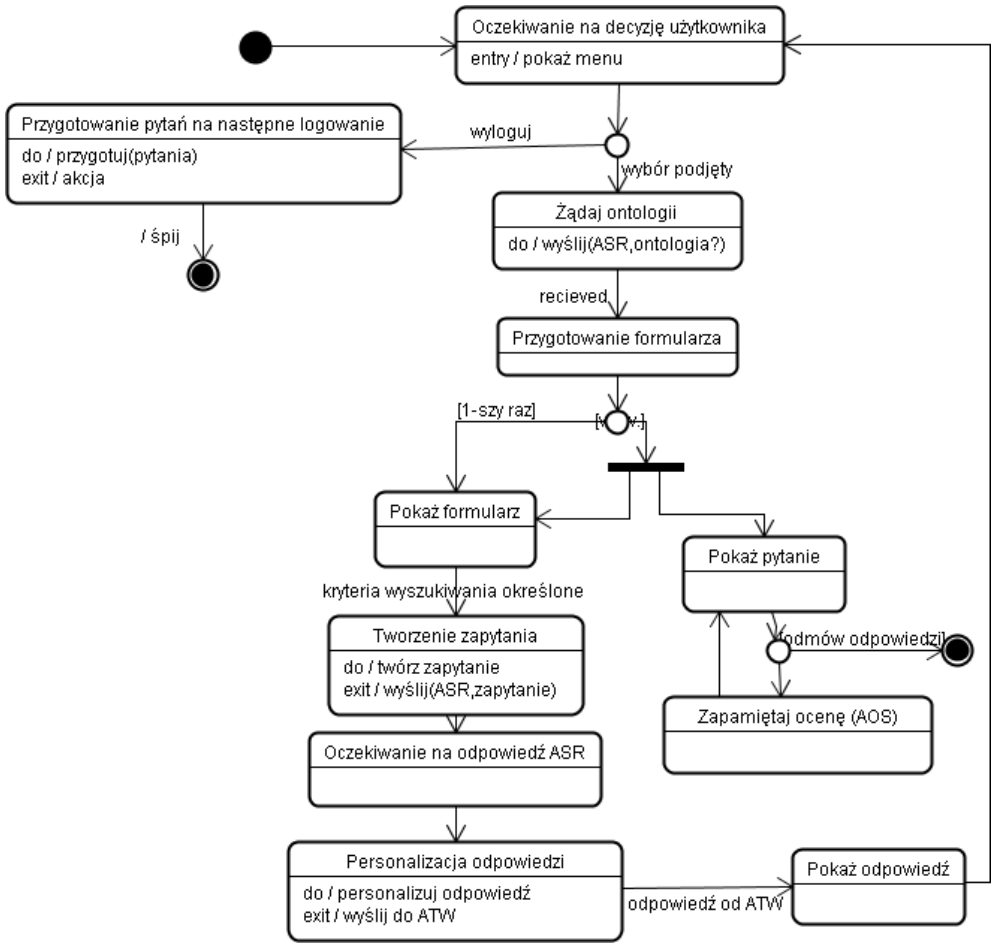

Rysunek 5.10: Diagram sekwencji dla obsługi żądania użytkownika w Podsystemie Dostarczania Informacji. *APr* - Agent Proxy, *AOS* – Agent Obsługujący Sesję, *AO* – Agent Osobisty, *ASR* - Agent Serwisu Restauracyjnego, *ATW* - Agent transformujący Widok

Javy, model jest zbiorem trójek RDF.

2. *Przesyłanie danych za pomocą wiadomości ACL.* AO chce przygotować pytania dla użytkownika na następną sesję, zatem musi uzyskać wcześniej informacje o zdarzeniach zaszłych w trakcie ostatniej sesji. W tym celu prosi AOS o przesłanie listy takich zdarzeń. AOS musi przekształcić znalezione trójki RDF do formy możliwej do przesłania za pomocą wiadomości ACL. A AO musi potrafić zdekodować odebraną wiadomość do postaci zbioru obiektów Javy, aby móc na nich operować.

Ponieważ powyższy scenariusz powtarza się w systemie bardzo często w wielu wariantach, zdecydowałem się opracować mechanizm normujący sposób dostępu do danych. W tym celu wykorzystałem wzorce projektowe: *Obiekt Dostępu do Danych*[33] i

[33]Ang. *Data Access Object* z *Core J2EE Patterns - Data Access Object*: `http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html`.




Rysunek 5.11: Diagram stanów dla AO, nie obejmujący startu agenta.

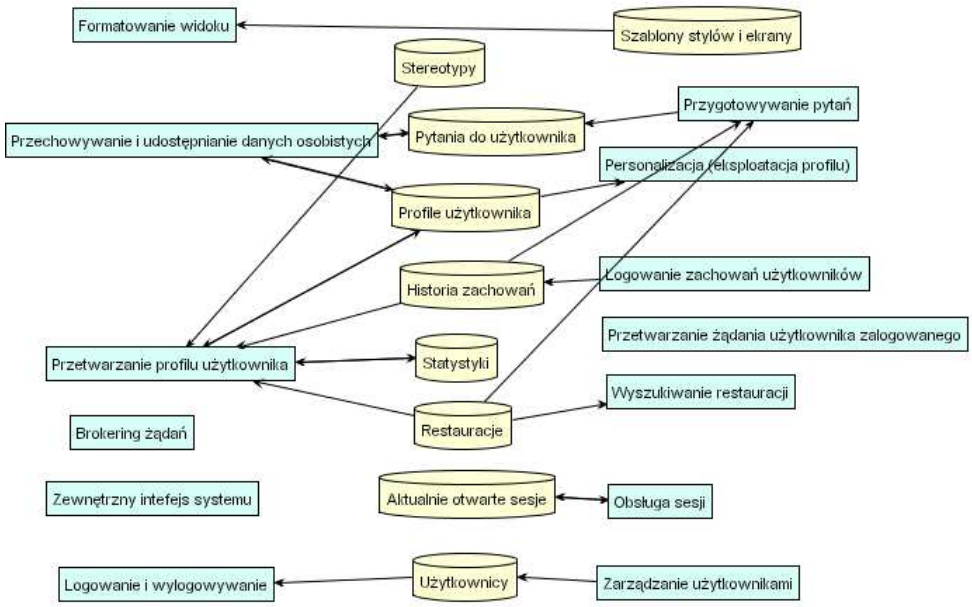
*Obiekt do Transferu*[34]. Wzorzec Obiektu Dostępu do Danych dostarcza stronie trzeciej (na przykład zachowaniu agenta) interfejs wysokopoziomowych funkcji pozwalających na operowaniu na danych bez potrzeby martwienia się o format i sposób dostępu do źródła tych danych. Obiekt do Transferu służy jako nośnik informacji między stroną trzecią a źródłem danych, a jego konwersją z i na format źródła danych zajmuje się odpowiedni Obiekt Dostępu do Danych.

Wróćmy do pierwszej sytuacji. Jej realizację przedstawia poniższy fragment kodu (w języku Java).

```
1 EventDAO eventDAO = PlatformManager.getEventDAO();
2 EventTO e = new EventTO( null );
3 e.setWhen( new Data() );
4 e.setSessionID ( sid );
5 eventDAO.insertEvent( e );
```

AOS otrzymuje od `PlatformManager` obiekt `eventDAO` udostępniający między innymi funkcję `insertEvent()`. Funkcja ta jako argument przyjmuje obiekt typu `EventTO`. Typ ten symbolizuje pojęcie zdarzenia i jest pochodną klasy `TransferObject` W wyni-

[34] Ang. *Transfer Object* z *Core J2EE Patterns - Transfer Object*: `http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html`.




Rysunek 5.12: Diagram obrazujący korzystanie z danych przez poszczególne role (według metodologii Prometheus). Strzałka idąca od roli do danych oznacza, że rola ta ma mieć możliwość zapisu do nich, strzałka odwrotna – możliwość odczytu. Strzałka w obie strona symbolizuje jednoczesną możliwość zapisu i odczytu.

ku działania kodu zdarzenie `e` zostaje dodane do bazy zdarzeń (`events-model`). Dalej chcielibyśmy odtworzyć drugą sytuację:

```
1 List events = eventDAO.getSessionEvents( sid );
2 DataManipulator eventDM = DMManager.getEventDM();
3 String rdfxml = eventDM.externalizeS( events );
```

Obiekt `events` stanowi listę obiektów typu `EventTO` i jest w kolejnych liniach serializowany (funkcja `externalizeS()`) do formatu RDF/XML. Format ten jest łatwy do przesłania w wnętrzu wiadomości ACL i może być zdeserializowany po drugiej stronie, tzn. u agenta AO za pomocą funkcji `internalize()`:

```
1 DataManipulator eventDM = DMManager.getEventDM();
2 List events = eventDM.internalize(rdfxml);
```

Do synchronizacji konceptów ontologicznych z klasami Javy wykorzystałem narzędzie `schemagen` dostarczane w pakiecie Jena. Pozwala ono konwertować plik ontologii na klasę statycznych pól odpowiadających konceptom klas, właściwości i instancjom ontologii.

### Ograniczenie mobilności agentów bazodanowych i usługowych

Większość przedstawionych w systemie agentów pełni rolę usługową i ma charakter reakcyjny, a nie proaktywny. ATW dostarcza z pomocą serwera Raccoon usługi formatującej, a ASR udostępnia dane na temat restauracji. Mimo iż, komunikacja z tymi usługami jest możliwa również zdalnie, to względy wydajnościowe wiążą agenta z daną

| *Agent* | *Trwałe źródła danych* |
|---|---|
| *AOS* | Model zarejestrowanych użytkowników (`users-model`), model zanotowanych zdarzeń (`events-model`). |
| *AZP* | Model profili użytkowników (`user-profiles-model`), model profili stereotypów (`stereotype-profiles-model`), model statystyk wypracowanych w wyniku uczenia (`statistics-model`), model pytań AO (`questions-model`). |
| *ASR* | Model restauracji z całej polski (`restaurants-model`) oraz model ontologii restauracji m(`restaurant-onto-model`). |

Tabela 5.8: Przydział trwałych źródeł danych do agentów. Przez trwałe źródła (ang. *persistent*) rozumiemy modele Jeny umieszczone w bazie MySQL.

maszyną, na której znajduje się dana usługa czy baza danych. Ogranicza to w dużym stopniu ich mobilność.

### 5.2.3 Platforma implementacyjna

System został zaimplementowany na platformie agentowej JADE3.3 z użyciem pakietu JADEX wspierającego debugowanie komunikacji międzyagentowej w środowisku JDK1.4.2. Dostęp do danych ontologicznych możliwy był dzięki użyciu środowiska Jena2.2, zaś same dane składowane są w tzw. trwałych modelach składowanych w bazie danych MySQL4.1.13, do których dostęp odbywa się za pomocą mostka JDBC (`mysql-connector-java-3.2.0-alpha`).

Implementacja polegała na stworzeniu serii agentów i ich zachowań realizujących kolejne scenariusze. Konieczne stało się także przeprogramowanie niektórych domyślnych klas platformy JADE. Stworzona została seria ontologii oraz ekranów interfejsu użytkownika w postaci szablonów XSL i plików HTML.

Aplikacja korzysta z serwera Raccoon (Rx4RDF i Rhizome 0.4.3).

# Rozdział 6

# Wnioski końcowe

Proces modelowania użytkownika pozwala na dostarczanie użytkownikowi informacji odpowiedniej do jego wymagań. W niniejszej pracy pokazałem, w jaki sposób można ten proces realizować w środowisku Semantycznej Sieci WWW z wykorzystaniem technologii agentów programowych. W rezultacie powstał podsystem systemu wspomagania podróży pozwalający na dostarczanie informacji spersonalizowanej.

W trakcie pracy nad projektem pojawiły się dwa zasadnicze problemy: (a) jak integrować technologie Semantycznej Sieci WWW z metodami modelowania użytkownika oraz (b) w jaki sposób włączać te metody do systemu wieloagentowego, opartego do na interfejsie WWW. Kluczem jest więc zrozumienie procesu modelowania użytkownika w ogóle. W wybranej przeze mnie taksonomii systemów rekomendujących (Montaner *et al.* 2003) autorzy podkreślają dwa zasadnicze etapy procesu modelowania użytkownika: (a) stworzenie i rozwijania modelu oraz (b) wykorzystanie modelu do filtrowania informacji. Konstrukcja obu etapów jest ściśle związana ze sposobem w jaki reprezentowany jest profil użytkownika. Z semantycznego punktu widzenia w środowisku, w którym polecane przedmioty definiowane zgodnie z ontologią dziedziny wiedzy (ontologią restauracji) naturalnym stało się wykorzystanie modelu typu *„overlay”*, w którym opinie użytkownika odnoszą się do niej w sposób bezpośredni (opinie o konceptach ontologicznych) (Brusilovsky 1996; Greer i McCalla 1993). Pod kątem syntaktyki zaproponowałem ontologię profilu zapisaną w formie RDF. Takie rozwiązanie ma dwie zasadnicze zalety: (a) pozwala na możliwość rozbudowy ontologii dziedziny (na przykład o domenę hotelarstwa) bez konieczności ponownego tworzenia i rozwijania profili użytkowników i (b) przyśpiesza proces uczenia profilu użytkownika dzięki wnioskowaniu na podstawie ontologii dziedziny (Fink i Kobsa 2002). Algorytm wnioskowania o zainteresowaniu na podstawie relacji z ontologii narzuca pewne ograniczenia co do rodzaju tych relacji: muszą mieć one charakter zależności (bycie podklasą, bycie instancją itp.) oraz niedopuszczalne są cykle. Tym samym ontologia dziedziny musi mieć charakter hierarchiczny. Potrzebne są metody, które wykorzystują znaczeniowo bogatsze typy relacji (bycie synonimem, częścią wspólną dwóch innych klas) i nie ograniczają konstrukcji ontologii dziedziny.

W systemie znajduje się *Repozytorium*, zawierające informacje o lokalach gastronomicznych w Polsce. Każdy lokal opisany jest przez szereg atrybutów zdefiniowanych w ontologii dziedziny. Dlatego też zdecydowałem się na użycie *metody rekomendacji*

*opartej na treści*. Polega ona na analizie częstotliwości występowania cech restauracji oglądanych i/lub ocenianych przez użytkownika i uczeniu na tej podstawie jego modelu. Zaletą przedstawionego podejścia jest możliwość oszacowania zainteresowania użytkownika nieznanym obiektem wyłącznie w oparciu o zbieżność między jego charakterystyką a opiniami użytkownika. Z drugiej strony mała ilość informacji w opisie niektórych obiektów może prowadzić do spadku efektywności metody. Niestety, informacje na temat restauracji, dostarczane przez serwis *Chefmoz*[1] są rozłożone nierównomiernie (na przykład niektóre restauracje opisane są wyłącznie za pomocą adresu). Jednym z rozwiązań możliwych do zastosowania w przyszłości byłoby stworzenie agentów pracujących w Podsystemie Gromadzenia Informacji i znajdujących brakujące dane. Alternatywnym podejściem jest również zaprzęgnięcie do współpracy z systemem *kolaboratywnych technik rekomendacji*, rekompensujących niedostatki metody opartej na treści.

Metodom opartym na treści towarzyszy również *problem zimnego startu*. Jako rozwiązanie przedstawiłem oryginalny algorytm stereotypizacji. Jak pokazało doświadczenie, samo opracowanie algorytmu nie stanowi problemu tak dużego, jak przygotowanie wiarygodnych stereotypów ludzi. Przedstawiłem możliwe podejścia: każde z nich wymaga uprzedniego zebrania wiedzy na temat dziedziny życia, w obrębie której tworzone mają być stereotypy. Dlatego jednym z rozważanych przeze mnie podejść było przeprowadzenie ankiety. W celu właściwej organizacji całego przedsięwzięcia rozmawiałem ze specjalistami (mgr Joanną Ochniak z Wyższej Szkoły Hotelarstwa i Gastronomii w Poznaniu oraz dr hab. Andrzejem Masłowskim z Instytutu Rynku Wewnętrznego i Konsumpcji w Warszawie). Wyniki otrzymane w potencjalnie zorganizowanej ankiecie pozwoliłyby na dopasowania cech różnych ludzi do określonych typów restauracji, ale to z kolei wymagałoby określenia stereotypów restauracji (na przykład stereotyp restauracji japońskiej). Zadanie to może być szczególnie trudne przy braku niektórych informacji na temat lokali, opisanych w serwisie *Chefmoz*. Dodatkowo, kwestią wymagającą konsultacji nie tylko specjalistów z branży gastronomicznej, ale i psychologów oraz socjologów byłoby ustalenie istotności i ilości atrybutów zdefiniowanych w algorytmie stereotypowania (Hawkins *et al.* 1998; Reed i Friedman 1973). Dlatego też ostatecznie zdecydowałem się na stworzenie stereotypów wyłącznie w oparciu o zgromadzoną wcześniej wiedzę oraz własną intuicję.

Szczególna uwaga należała się procesowi zbierania informacji zwrotnej, który warunkowany jest przez interfejs WWW. Stawia on użytkownika w roli inicjującego dialog z systemem, dlatego przyjąłem zasadę, że przy wyborze rodzaju zbieranej informacji należy kierować się przede wszystkim komfortem i potrzebami użytkownika. Do minimum ograniczyłem bezpośrednią informację zwrotną (tylko niezbędne dane demograficzne potrzebne w trakcie rejestracji użytkownika oraz możliwość swobodnego głosowania na restauracje), nie odbierając jednocześnie możliwości udzielenia jej. O sensowności takiego rozwiązania przekonuje doświadczenie autorów pokrewnego systemu *Anatagomy* (Sakagami *et al.* 1997) oraz przewaga istotności reakcji wprost nad reakcjami nie-wprost (Nichols 1998). W proponowanym rozwiązaniu wykorzystałem technikę *„wypchnij i ściągnij”* (Chou *et al.* 2003), która pozwala systemowi zadać py-

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

tanie o ocenę restauracji, a użytkownikowi podjąć decyzję o jej udzieleniu lub odmowie. Takie podejście stanowi równocześnie odpowiedź na sytuacje, w których dwie jednostki *proaktywne* (użytkownik i *Agent Osobisty*) wykazują inicjatywę (w środowisku o współdzielonej inicjatywie)[2]. W rozwiązaniu uwzględniłem również fakt, że to klient wysyła żądanie (inicjuje połączenie HTTP) i agent może przejąć inicjatywę wyłącznie w ramach odpowiedzi systemu na to nie.

Zbieranie informacji zwrotnej wymaga identyfikacji użytkownika i śledzenia jego zachowania. W tym celu zaprojektowałem i zaimplementowałem (razem z Pawłem Kaczmarkiem) rozwiązanie oparte na pojęciu sesji[3].

Jak twierdzą Próchnicka i Fink, środowisko rozproszone w naturalny sposób pozwala zrealizować ideę autonomiczności i niezależności komponentu modelowania użytkownika (Próchnicka 2000; Fink 1996). W projektowanym systemie wprowadziłem Agenta Zarządzającego Profilami, którego zachowania zajmują się procesem inicjalizacji i uczenia profilu użytkownika. Profil użytkownika i inne informacje osobiste udostępniane są Agentom Osobistym wyłącznie na żądanie. Dzięki temu Agent Osobisty jest lekki (nie przechowuje dodatkowych danych związanych z rozwojem profilu), nie związany z żadnymi trwałymi źródłami danych i – zgodnie z wymogami FIPA[4] – zachowuje mobilność, a rozwój i zmiana technik modelowania użytkownika (na przykład wprowadzenie technik kolaboratywnych) wymaga jedynie przebudowania odpowiednich zachowań AZP.

W trakcie pracy nad projektem powrócił także problem pogodzenia asynchronicznej natury komunikacji międzyagentowej z synchronicznej naturą protokołu HTTP (Kaczmarek 2005). W jaki sposób rozwiązywać sytuacje, gdy pomimo żądania użytkownika, system nie jest w stanie chwilowo w stanie odpowiedzieć na to żądanie. Przykładowo, w jaki sposób Agent Osobisty powinien reagować na prośbę użytkownika o dokonanie rekomendacji w sytuacji, gdy nie otrzymał jeszcze profilu użytkownika od AZP? Pytanie pozostaje otwarte, gdyż obecne rozwiązanie (wstrzymanie się z odpowiedzią do momentu otrzymania profilu) jest nie satysfakcjonujące i może irytować użytkownika.

[2] Maes, Pattie, Miller, Jim i Shneiderman, Ben. 1997. *Intelligent Software Agents vs. User-Controlled Direct Manipulation: A Debate*. `http://www.acm.org/sigchi/chi97/proceedings/panel/jrm.htm`.

[3] Szczegółowe rozwiązanie przedstawione jest w załączniku A.

[4] *FIPA Personal Travel Assistance Specification*: `http://www.fipa.org/specs/fipa00080/XC00080A.html#_Toc505481930`.

# Rozdział 7

# Wykaz źródeł

## 7.1 Wykaz źródeł publikowanych


Ali, D., Cobb, M., Fiedorowicz, I., Angryk, Rafał, Kołodziej, K., Paprzycki, Marcin i Rahimi, S. 2001. Development of a Travel Support System Based on Intelligent Agent Technology. *Strony 243–255 z: Proceedings of the PIONIER 2001 Conference*. Poznań: Wydawnictwo Politechniki Poznańskiej.

Alur, Deepak, Malks, Dan i Crupi, John. 2001. *Core J2EE Patterns: Best Practices and Design Strategies*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Andrzejewska, Olga. 2005. Rynek usług gastronomicznych. *Nowe Życie Gospodarcze*, **1**(385), 17–20.

Angryk, Rafał, Galant, Violetta, Gordon, Minor i Paprzycki, Marcin. 2002. Travel Support System – an Agent-Based Framework. *Strony 719–725 z: Proceedings of the International Conference on Internet Computing IC'2002*. Las Vegas, NV, USA: CSREA Press.

APNews. 1999. Wiadomość z dnia 24 stycznia 1999 roku. *Associated Press News Service*.

Armstrong, Robert, Freitag, Dayne, Joachims, Thorsten i Mitchell, Tom. 1995. WebWatcher: A Learning Apprentice for the World Wide Web. *Strony 6–12 z: AAAI Spring Symposium on Information Gathering*.

Babbie, Earl R. 2003. *Badania społeczne w praktyce*. Warszawa: Wydawnictwo Naukowe PWN.

Baczyńska, Maria i Kumanowska, Beata. 2004. *Expert systems for the Semantic Web (survey)*.

Bassara, Andrzej. 2004. «I weź tu dogadaj się» - Ontologie. *Gazeta IT*.

Berger, Helmut, Dittenbach, Michael i Merkl, Dieter. 2004. An adaptive information retrieval system based on associative networks. *Strony 27–36 z: CRPIT '04: Proceedings of the first Asian-Pacific conference on Conceptual modelling*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc.

Bieliková, Mária i Kurus, Jaroslav. 2005. Sharing User Models for Adaptive Hypermedia Applications. *Strony 506–511 z: ISDA 2005: 5th International Conference on Intelligent Systems Design and Applications*. Los Alamitos, CA, USA: IEEE Computer Society.

Billsus, Daniel i Pazzani, Michael J. 1999. A Hybrid User Model for News Classification. *Strony 99–108 z: Proceedings of User Modelling 1999*. Wiedeń, Austria i Nowy York, USA: Springer-Verlag.

Booch, Grady. 1994. *Object-oriented analysis and design with applications (2nd ed.)*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.

Brickley, Dan, Miller, Libby i Sharp, Kate. 2004. Semantic Web Standardization in Europe: SWAD-Europe. *ERCIM News*.

Brusilovsky, Peter. 1996. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, **6**(2-3), 87–129.

Buckley, Chris i Salton, Gerard. 1995. Optimization of relevance feedback weights. *Strony 351–357 z: SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM Press.

Burke, Robin. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, **12**(4), 331–370.

Caire, Giovanni. 2003 (grudzień). *Jade Tutorial: Jade Programming for Beginners*. TILab.

Caire, Giovanni. 2004 (listopad). *Jade Tutorial: Application-defined Content Languages and Ontologies*. TILab.

Carlson, William Lee i Thorne, Betty. 1997. *Applied statistical methods : for business, economics, and the social sciences*. Upper Saddle River, New Jeyrsey, USA: Prentice Hall.

Carroll, John M. i Rosson, Mary Beth. 1987. Paradox of the active user. *Chap. 5, strony 80–111 z:* Carroll, John M. (redaktor), *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. Bradford Books/MIT Press.

Chatterjee, Patrali, Hoffman, Donna L. i Novak, Thomas P. 2003. Modeling the Clickstream: Implications for Web-Based Advertising Efforts. *Marketing Science*, **22**(4), 520–541.

Chen, Liren i Sycara, Katia. 1998. WebMate: a personal agent for browsing and searching. *Strony 132–139 z: AGENTS '98: Proceedings of the second international conference on Autonomous agents*. New York, NY, USA: ACM Press.

Chen, Sherry Y. i Magoulas, George D. 2005. *Adaptable and Adaptive Hypermedia Systems*. IRM Press.

Chmiel, Krzysztof, Tomiak, Dominik, Gawinecki, Maciej, Karczmarek, Paweł, Szymczak, Michał i Paprzycki, Marcin. 2004. Testing the Efficiency of JADE Agent Platform. *Strony 49–56 z: ISPDC '04: Proceedings of the Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (ISPDC/HeteroPar'04)*. Washington, DC, USA: IEEE Computer Society.

Chou, Wu, Liu, Feng, Li, Li, Shan, Xueshan i Li, Jenny. 2003 (listopad). *Dialogue System and Web Convergence for Mobility and Wireless Access of Enterprise Applications*. Raport tech. Avaya Labs Research, New Jersey, USA.

Cohen, P. R. i Perrault, C. R. 1979. Elements of a Plan-based theory of speech acts. *Cognitive Science*, 177–212.

Cohen, P. R. i Perrault, C. R. 1998. Elements of a Plan-based theory of speech acts. *Business Week*, 68–75.

Cormen, Thomas H. 2004. *Wprowadzenie do algorytmów*. Warszawa: Wydawnictwa Naukowo-Techniczne.

Cunningham, P., Bergmann, R., Schmitt, S., Traphoener, R., Smyth, B. i AnUltaigh, P. 2001. WEBSELL: Intelligent Sales Assistants for the World Wide Web. *KI – Zeitschrift für Künstliche Intelligenz*.

Daconta, Michael C., Obrst, Leo J. i Smith, Kevin T. 2003. *The Semantic Web : A Guide to the Future of XML, Web Services, and Knowledge Management*. Indianapolis, Indiana, USA: Wiley Publishing, Inc.

Davies, John, Fensel, Dieter i van Harmelen, Frank. 2003. *Towards the Semantic Web: Ontology-Driven Knowledge Management*. John Wiley & Sons.

Fensel, Dieter. 2003. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Fink, Josef. 1996. A Flexible and Open Architecture for the User Modeling Shell System BGP-MS. *Strony 237–239 z: Proceedings of the Fifth International Conference on User Modeling (UM-96)*.

Fink, Josef i Kobsa, Alfred. 2002. User Modeling for Personalized City Tours. *Artif. Intell. Rev.*, **18**(1), 33–74.

Galant, Violetta i Paprzycki, Marcin. 2005. Agent nie tylko u Twoich drzwi (II). *Gazeta IT*, **37**(7).

Galant, Violetta, Gordon, Minor i Paprzycki, Marcin. 2002. Knowledge Management in an Internet Travel Support System. *Strony 97–104 z: Proceedings of ECON2002*. Wejcherowo: ACTEN.

Gawinecki, Maciej, Gordon, Minor, Kaczmarek, Paweł i Paprzycki, Marcin. 2003. The Problem of Agent-Client Communication on the Internet. *Parallel and Distributed Computing Practices*, **6**(1), 111–123.

Gawinecki, Maciej, Gordon, Minor, Paprzycki, Marcin, Szymczak, Michał, Vetulani, Zygmunt i Wright, Jimmy. 2005a. Enabling Semantic Referencing of Selected Travel Related Resources. *Strony 271–288 z:* Abramowicz, Witold (redaktor), *BIS 2005: Proceedings of the 8th International Conference on Business Information Systems*. Poznań: Poznań University of Economics Press.

Gawinecki, Maciej, Gordon, Minor, Nguyen, Ngoc Thanh, Paprzycki, Marcin i Szymczak, Michał. 2005b. RDF Demarcated Resources in an Agent Based Travel Support System. *Strony 271–288 z: Proceedings of the 17th Mountain Conference of the Polish Information Society*.

Genesereth, Michael J. i Nilsson, Nils J. 1986. *Logical Foundation of Artificial Intelligence*. Morgan Kaufmann, Palo Alto, CA, USA.

Gordon, Minor i Paprzycki, Marcin. 2005. Designing Agent Based Travel Support System. *Strony 207–214 z: Proceedings of the ISPDC 2005 Conference*. Los Alamitos, CA, USA: IEEE Computer Society Press.

Greer, J. i McCalla, G. 1993. Student modeling: the key to individualized knowledgebased instruction. *NATO ASI Series F*, **125**.

Harold, Elliotte Rusty. 2000. *XML. Księga eksperta*. Gliwice: Helion.

Hawkins, Delbert, Best, Roger i Coney, Kenneth. 1998. *Consumer Behavior: Building Marketing Strategy*. New York,NY, USA: Irwin/McGraw-Hill.

Hayes, Conor i Cunningham, Pádraig. 1999. *Smart Radio – a Proposal. Technical Report, TCD-CS-1999-24*. Raport tech. Trinity College, Computer Science, Dublin, Irlandia.

Hayes, Conor i Cunningham, Pádraig. 2000 (grudzień). Smart Radio: Building Music Radio on the Fly. *Z: Proceedings of Expert Systems 2000 (ES2000)*.

Helic, Denis. 2001 (lipiec). *Aspects of Semantic Data Modeling in Hypermedia Systems*. Praca doktorska, Institute for Information Processing and Computer Supported New Media (IICM), Graz University of Technology.

Hendler, James A. 1999. Is There an Intelligent Agent in Your Future? *Nature Webmatters*.

Herlocker, Jonathan L., Konstan, Joseph A. i Riedl, John. 2000. Explaining collaborative filtering recommendations. *Strony 241–250 z: CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*. Nowy York, NY, USA: ACM Press.

Holte, Robert C. i Yan, John Ng Yuen. 1996. Inferring What a User Is Not Interested in. *Strony 159–171 z: AI '96: Proceedings of the 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*. London, UK: Springer-Verlag.

Horrocks, Ian i Sattler, Ulrike. 2002 (lipiec). Description Logics Tutorial. *Z: European Conference on Artificial Intelligence*.

Jennings, Nicholas R. 1999. Agent-oriented software engineering. *Strony 4–10 z: Proceedings of the 12th international conference on Industrial and engineering applications of artificial intelligence and expert systems : multiple approaches to intelligent systems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Joachims, Thorsten, Freitag, Dayne i Mitchell, Tom M. 1997. Web Watcher: A Tour Guide for the World Wide Web. *Strony 770–777 z: Proceedings of IJCAI'97 (1)*.

Joerding, Tanja, Michel, Stefan i Popella, Matthias. 1998. Tellim – ein System für adaptive Multimediale Produktpräsentationen im World Wide Web. *Strony 29–40 z:* Timm, U. J. i Róssel, M. (redaktorzy), *ABIS-98 – 6. Workshop Adaptivität und Personalised hypermedia presentation techniques 151 Benutzermodellierung in interaktiven Softwaresystemen*.

Kaczmarek, Paweł. 2005. *Multimodalna komunikacja agentów programowych z użytkownikiem*. Praca magisterska, Wydział Matematyki i Informatyki, Uniwersytet Adama Mickiewicza, Poznań.

Kaczmarek, Paweł, Paprzycki, Marcin, Gawinecki, Maciej i Vetulani, Zygmunt. 2005. Interakcja Użytkownik - Agentowy System Wspomagania Podróży. *Z: Proceedings of the 17th Mountain Conference of the Polish Information Society*.

Kay, Judy. 1999. Ontologies for reusable and scrutable student models. *Strony 727–777 z:* Mizoguchi, Riichiro (redaktor), *AIED Workshop W2: Workshop on Ontologies for Intelligent Educational Systems*.

Kobsa, Alfred. 1990. User modeling in Dialog systems: potentials and hazards. *AI Soc.*, **4**(3), 214–231.

Kobsa, Alfred. 2001. Generic User Modeling Systems. *User Modeling and User-Adapted Interaction*, **11**(1-2), 49–63.

Kobsa, Alfred. 2002. Personalized hypermedia and international privacy. *Commun. ACM*, **45**(5), 64–67.

Kobsa, Alfred i Pohl, Wolfgang. 1995. The User Modeling Shell System BGP-MS. *User Model. User-Adapt. Interact.*, **4**(2), 59–106.

Kobsa, Alfred, Koychev, Ivan i Schwab, Ingo. 2000. Learning about Users from Observation. *Strony 102–106 z: Adaptive User Interfaces: Papers from the 2000 AAAI Spring Symposium*. Stanford, CA, USA: AAAI Press.

Kobsa, Alfred, Koenemann, Jürgen i Pohl, Wolfgang. 2001. Personalised hypermedia presentation techniques for improving online customer relationships. *Knowl. Eng. Rev.*, **16**(2), 111–155.

Koenig, Michael E. D. 1990. Linking library users: a culture change in librarianship. *American Libraries*, **21**(9), 844–849.

Konstan, Joseph A., Miller, Bradley N., Maltz, David, Herlocker, Jonathan L., Gordon, Lee R. i Riedl, John. 1997. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, **40**(3), 77–87.

Koychev, Ivan. 2000. Gradual Forgetting for Adaptation to Concept Drift. *Strony 101–106 z: Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning*.

Krukowski, Marek. 2005. Ku przyszłości internetu – sieć semantyczna. *Software Developer's Journal*, 24–28.

Krulwich, B. 1997. Lifestyle Finder: Intelligent User Profiling Using Large-Scale Demographic Data. *Artificial Intelligence Magazine*, **18**(2), 37–45.

Lang, K. 1995. NewsWeeder: Learning to Filter News. *Strony 331––339 z: Proceedings of the 12th International Conference on Machine Learning*.

Lieberman, Henry. 1995. Letizia: An Agent That Assists Web Browsing. *Strony 924–929 z:* Mellish, Chris S. (redaktor), *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*. Montreal, Quebec, Canada: Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

Luhn, H.P. 1961. Selective Dissemination of New Scientific Information with the Aid of Electronic Processing Equipment. *American Documentation*, **12**(2), 131–138.

Maes, Pattie. 1994. Agents that reduce work and information overload. *Commun. ACM*, **37**(7), 30–40.

Malone, Thomas W., Grant, Kenneth R., Turbak, Franklyn A., Brobst, Stephen A. i Cohen, Michael D. 1987. Intelligent information-sharing systems. *Commun. ACM*, **30**(5), 390–402.

McCauley, C., Stitt, C.L. i Segal, M. 1980. Stereotyping: from prejudice to prediction. *Psycho Bulletin*, **87**, 195—-208.

Mitchell, Thomas M. 1997. *Machine Learning*. McGraw-Hill Higher Education.

Mladenic, Dunja. 1996 (październik). *Personal WebWatcher: Implementation and Design. Technical Report IJS-DP-7472*. Raport tech. School of Computer Science, Carnegie-Mellon University, Pittsburgh, USA.

Mobasher, Bamshad, Cooley, Robert i Srivastava, Jaideep. 2000. Automatic personalization based on Web usage mining. *Commun. ACM*, **43**(8), 142–151.

Montaner, Miquel, López, Beatriz i de la Rosa, Josep Lluís. 2002. Developing trust in recommender agents. *Strony 304–305 z: AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press.

Montaner, Miquel, López, Beatriz i de la Rosa, Josep Lluís. 2003. A Taxonomy of Recommender Agents on the Internet. *Artif. Intell. Rev.*, **19**(4), 285–330.

Mooers, Calvin. 1951. Zatocoding applied to mechanical organization of knowledge. *American Documentation*, 20–32.

Morita, Masahiro i Shinoda, Yoichi. 1994. Information Filtering Based on User Behaviour Analysis and Best Match Text Retrieval. *Strony 272–281 z:* Croft, W. Bruce i van Rijsbergen, C. J. (redaktorzy), *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. ACM/Springer.

Nichols, D. 1998. Implicit rating and filtering. *Strony 31–36 z: Proceedings of 5th DELOS Workshop on Filtering and Collaborative Filtering*. ERCIM.

Nistor, Cristina Elena, Oprea, Robert, Paprzycki, Marcin i Parakh, Garima. 2002. The Role of a Psychologist in E-commerce Personalization. *Strony 227–231 z: Proceedings of the 3rd European E-COMM-LINE 2002 Conference*.

Norman, Donald A. 1994. How might people interact with agents. *Commun. ACM*, **37**(7), 68–71.

North, Simon i Hermans, Paul. 2000. *XML dla każdego*. Gliwice: Helion.

Nowak, Marcin. 2004. Pajęczyna II. *CHIP*, **7**, 116–119.

Oard, Douglas W. 1997. The State of the Art in Text Filtering. *User Modeling and User-Adapted Interaction*, **7**(3), 141–178.

Oktaba, Wiktor. 1969. *Słownik polsko-rosyjsko-angielski statystyki matematycznej i teorii doświadczenia*. Warszawa: Państwowe Wydawnictwo Naukowe.

Orwant, Jon. 1995. Heterogeneous learning in the Doppelgänger user modeling system. *User Modeling and User-Adapted Interaction*, **4**(2), 107—-130.

Ovum. 1994. *Intelligent Agents: the New Revolution in Software (Report)*. Raport tech. Ovum, Londyn, Wielka Brytania.

Padgham, Lin i Winikoff, Michael. 2002. Prometheus: a methodology for developing intelligent agents. *Strony 37–38 z: AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press.

Paprzycki, Marcin. 2003. Agenci programowi jako metodologia tworzenia oprogramowania. *KKIO V*. Problemy i metody inżynierii oprogramowania.

Pazzani, Michael i Billsus, Daniel. 1997. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Mach. Learn.*, **27**(3), 313–331.

Pazzani, Michael J. 1999. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artif. Intell. Rev.*, **13**(5-6), 393–408.

Pazzani, Michael J., Muramatsu, Jack i Billsus, Daniel. 1996. Syskill Webert: Identifying Interesting Web Sites. *Strony 54–61 z: AAAI/IAAI, Vol. 1*.

Perepletchikov, Mikhail. 2004 (październik). *Tools and Processes gor Agent-oriented, Requirements Engineering*. Praca magisterska, School of Computer Science and Information Technology, RMIT University, Melbourne, Australia.

Periklis, A. 2002 (marzec). *Data Clustering Techniques*. Qualifying Oral Examination Paper, University of Toronto, Toronto, Kanada.

Pohl, Wolfgang i Nick, Achim. 1999 (maj). Machine learning and knowledge-based user modeling in the LaboUr approach. *Strony 179–188 z:* Kay, Judy (redaktor), *UM99 User Modeling: Proceedings of the Seventh International Conference Springer-Verlag*.

Powers, Shelley. 2003. *Practical RDF*. Sebastopol, CA, USA: O'Reilly & Associates, Inc.

Próchnicka, Maria. 2000. Metody i techniki modelowania użytkownika w inteligentnych systemach informacyjnych. *Z: II Krajowa Konferencja MISSI 2000 (Multimedialne i Sieciowe Systemy Informacyjne*.

Razmerita, Liana, Angehrn, Albert A. i Maedche, Alexander. 2003. Ontology-Based User Modeling for Knowledge Management Systems. *Strony 213–217 z: User Modeling*.

Reed, Steve K. i Friedman, M. P. 1973. Perceptual vs. conceptual categorization. *Memory & Cognition*, **1**, 157—163.

Resnick, Paul, Iacovou, Neophytos, Suchak, Mitesh, Bergstrom, Peter i Riedl, John. 1994. GroupLens: an open architecture for collaborative filtering of netnews. *Strony 175–186 z: CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*. New York, NY, USA: ACM Press.

Rich, Elaine A. 1979a. *Building and exploiting user models*. Praca doktorska.

Rich, Elaine A. 1979b. User modeling via stereotypes. *Cognitive Science*, **3**, 329–354.

Rotaru, Mihai. 2005 (maj). *Recommendation Systems*. Praca doktorska, Computer Science Department, University of Pittsburgh. Comprehensive Exam.

Sakagami, Hidekazu, Kamba, Tomonari i Koseki, Yoshiyuki. 1997. Learning Personal Preferences on Online newspaper articles for User Behaviors. *Strony 291–300 z: Proceedings of 6th Int. World Wide Web Conference*.

Sala, Józef. 2004. *Marketing w gastronomii (wydanie I)*. Warszawa: Polskie Wydawnictwo Ekonomiczne.

Salton, Gerard i Buckley, Chris. 1997. Improving retrieval performance by relevance feedback. 355–364.

Saracevic, Tefko, Spink, Amanda i Wu, Mei-Mei. 1997. Users and Intermediaries in Information Retrieval: What Are They Talking About? *Strony 43–54 z:* Jameson, Anthony, Paris, Cécile i Tasso, Carlo (redaktorzy), *User Modeling: Proceedings of the Sixth International Conference, UM97*.

Schafer, J. Ben, Konstan, Joseph i Riedi, John. 1999. Recommender systems in e-commerce. *Strony 158–166 z: EC '99: Proceedings of the 1st ACM conference on Electronic commerce*. New York, NY, USA: ACM Press.

Schwab, Ingo i Pohl, Wolfgang. 1999. Learning Information Interest from Positive Examples. *Z: Workshop "Machine Learning for User Modeling", Seventh International Conference on User Modeling.*

Sokulski, Jacek. 2002. Sieć semantyczna. *Software 2.0*, **96**(12), 63–67.

Somlo, Gabriel L. i Howe, Adele E. 2003. Using web helper agent profiles in query generation. *Strony 812–818 z: AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems.* New York, NY, USA: ACM Press.

Tusiewicz, Marcin. 2003. *System wieloagentowy: teoria, projekt, implementacja oraz przykłady zastosowań.* Praca magisterska, Instytut Informatyki, Wydział Matematyki, Fizyki i Informatyki, Uniwersytet Jagielloński, Kraków.

Weihberg, Marek. 2003. *Personalizacja interfesju w hipermedialnych witrynach wykorzystywanych w promocji.* Praca magisterska, Politechnika Wrocławska, Zakład Systemów Informacyjnych Wydziału Informatyki i Zarządzania, Wrocław.

White, Ryen, Jose, Joemon M. i Ruthven, Ian. 2001. Comparing Explicit and Implicit Feedback Techniques for Web Retrieval: TREC-10 Interactive Track Report. *Z: TREC.*

White, Ryen, Ruthven, Ian i Jose, Joemon M. 2002. The Use of Implicit Evidence for Relevance Feedback in Web Retrieval. *Strony 93–109 z: Proceedings of the 24th BCS-IRSG European Colloquium on IR Research.* London, UK: Springer-Verlag.

Wlekły, Grzegorz. 2004. Narzędzia do tworzenia ontologii. *Gazeta IT*, **2**.

Wooldridge, M. 1997. Agent-based software engineering. *IEE Proceedings Software Engineering*, **144**(1), 26–37.

Yan, T. i Garcia-Molina, H. 1995. SIFT—A tool for Wide-Area Information Dissemination. *Strony 177–186 z: Proc. 1995 USENIX Technical Conference.*

Zambonelli, Franco, Jennings, Nicholas R., Omicini, Andrea i Wooldridge, Michael. 2000. Agent-Oriented Software Engineering for Internet Applications. *Strony 326–346 z:* Omicini, A., Zambonelli, F., Klusch, M. i Tolksdorf, R. (redaktorzy), *Coordination of Internet Agents: Models, Technologies, and Applications.* Springer-Verlag: Heidelberg, Niemcy.

## 7.2 Wykaz źródeł internetowych

ACL. 2002. *FIPA Agent Communication Specifications.* `http://www.fipa.org/repository/aclspecs.html`.

Amazon. 2005. *Amazon.* `http://www.amazon.com`.

AUML. 2005. *Agent UML Web Site.* `http://www.auml.org/`.

Berners-Lee, Tim. 1998. *RDF Anonymous nodes and quantification.* `http://www.w3.org/DesignIssues/Anonymous.html`.

Berners-Lee, Tim. 2000. *Semantic Web – XML2000 – slide «Architecture»*. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html.

Cervenka, Radovan. 2003 (kwiecień). *Prometheus Design Tool*. http://www.auml.org/auml/documents/Prometheus030402.doc.

ChefMoz. 2005. *ChefMoz dining guide*. http://chefdmoz.org/.

Cocoon. 2005. *The Apache Cocoon Project*. http://cocoon.apache.org/.

CSS. 2005. *Cascading Style Sheets (CSS)*. http://www.w3.org/Style/CSS/.

CWM. 2000. *CWM – Close World Machine*. http://www.w3.org/2000/10/swap/doc/cwm.html.

DAO. 2002. *Core J2EE Patterns - Data Access Object*. http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html.

FIPA. 2005. *FIPA Specifications*. http://www.fipa.org/specifications/index.html.

FIPA Ontology Service. 2005. *FIPA Ontology Service Specification*. http://www.fipa.org/specs/fipa00086/XC00086C.html.

FIPA Personal Travel Assistance Specification. 2000. *FIPA Personal Travel Assistance Specification*. http://www.fipa.org/specs/fipa00080/XC00080A.html#_Toc505481930.

FIPA Request. 2005. *FIPA Request Interaction Protocol Specification*. http://www.fipa.org/specs/fipa00026/SC00026H.pdf.

Google. 2005. *Google*. http://www.google.com/.

Grobman, Gary M. 1990. *Stereotypes and Prejudices*. http://www.remember.org//guide//History.root.stereotypes.html.

Gruber, Tom. 1993. *What is an Ontology?* http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

HTML. 2005. *W3C HTML Home Page*. http://www.w3.org/MarkUp/.

HTTP. 2000. *HTTP - Hypertext Transfer Protocol*. http://www.w3.org/Protocols/.

IsaViz. 2004. *IsaViz: A Visual Authoring Tool for RDF*. http://www.w3.org/2001/11/IsaViz/.

JADE. 2005. *Jade - Java Agent DEvelopment Framework*. http://jade.tilab.com/. an Open Source platform for peer-to-peer agent based applications.

Jena. 2005. *Jena – A Semantic Web Framework for Java*. http://jena.sourceforge.net/.

Komputronik. 2005. *Komputronik*. http://www.komputronik.pl/.

Maes, Pattie, Miller, Jim i Shneiderman, Ben. 1997. *Intelligent Software Agents vs. User-Controlled Direct Manipulation: A Debate*. http://www.acm.org/sigchi/chi97/proceedings/panel/jrm.htm.

MSN. 2005. *MSN.com*. http://www.msn.com/.

MultiWordNet. 2005. *MultiWordNet*. http://multiwordnet.itc.it/.

MyYahoo. 2005. *My Yahoo!* http://my.yahoo.com/.

N3. 2005. *Notation 3 – Ideas about Web architecture*. http://www.w3.org/DesignIssues/Notation3.html.

Nielsen, Jakob. 1996. *Top Ten Mistakes in Web Design*. http://www.useit.com/alertbox/9605.html.

ODP. 2002. *Open Directory Project*. http://dmoz.org/.

Paradowski, Michał. 2005. *Europe's Top Restaurants 2005*. http://katalogi.gastrona.pl/art/article_3617.php.

PDT. 2005. *Prometheus Design Tool - application site*. http://www.cs.rmit.edu.au/agents/pdt/.

PointcastNetwork. 1997. *Pointcast Network*. http://www.pointcast.com/.

Pretschner, Alexander i Gauch, Susan. 1999. *Personalization on the Web*. http://citeseer.nj.nec.com/pretschner99personalization.html.

Protege. 2005. *Protégé*. http://protege.stanford.edu/.

Prywatność 1. 1998. http://www.thestandard.com/article/display/0,1151,235,00.html.

Prywatność 2. 1998. *Reasons for Not Registering*. http://www.cc.gatech.edu/gvu/user_surveys/survey-1998-10/graphs/privacy/q48.htm.

Quinn, Lois M. i Pawasarat, John. 2001 (czerwiec). *Confronting Anti-Urban Marketing Stereotypes: A Milwaukee Economic Development Challenge*.

Raccoon. 2005. *Raccoon*. http://rx4rdf.liminalzone.org/Racoon.

Ramachandran, Vijay. 2002 (styczeń). *Design Patterns for Building Flexible and Maintainable J2EE Applications*. http://java.sun.com/developer/technicalArticles/J2EE/despat/.

RDF. 2005. *Resource Description Framework (RDF)*. http://www.w3.org/RDF/.

RDF Primer. 2005. *RDF Primer*. http://www.w3.org/TR/rdf-primer.

RDFS. 2004. *RDF Vocabulary Description Language 1.0: RDF Schema*. http://www.w3.org/TR/rdf-schema/.

RDF/XML. 2005. *RDF/XML Syntax Specification*. http://www.w3.org/TR/rdf-syntax-grammar.

RDQL. 2004. *RDQL - A Query Language for RDF*. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/.

Restaurant Spending. 2003. *Restaurant Spending*. http://www.restaurant.org/research/consumer/spending.cfm.

Reynolds, Dave. 2004. *e-Person - Personal Information Infrastructure*. http://www.hpl.hp.com/semweb/e-person.htm. 8 stycznia 2004.

Richter, Agnieszka. 2001. *Historia Internetu*. http://www.kailastudio.com.pl/design/htm/article/historia.htm.

RSS. 2005. *RSS (protocol*. http://en.wikipedia.org/wiki/RSS_(protocol).

SW Activity. 2001. *Semantic Web Activity Statement*. http://www.w3.org/2001/sw/Activity.

SWESE. 2005. Workshop on Semantic Web Enabled Software Engineering (SWESE). *Z: 4th International Semantic Web Conference (ISWC 2005)*.

Transfer Object. 2002. *Core J2EE Patterns - Transfer Object*. http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html.

URL. 1994. *Uniform Resource Locators (URL)*. http://www.ietf.org/rfc/rfc1738.txt.

Wikipedia. 2005. *Wikipedia: Ontologia. Kontekst informatyczny (w oparciu o sieci semantyczne)*. http://pl.wikipedia.org/wiki/Ontologia.

Wooldridge, Michael i Jennings, Nicholas R. 1995. *Intelligent Agents: Theory and Practice*. HTTP://www.doc.mmu.ac.uk/STAFF/mike/ker95/ker95-html.h (Hypertext version of Knowledge Engineering Review paper).

WordNet. 2005. *WordNet, a lexical database for the English language*. http://wordnet.princeton.edu/.

XML. 2005. *Extensible Markup Language (XML)*. http://www.w3.org/XML/.

XSL. 2005. *Extensible Stylesheet Language Family (XSL)*. http://www.w3.org/Style/XSL/.

Zagat. 2005. *Przewodnik Zagat Survey – online*. http://www.zagat.com/.

# Skorowidz

# Dodatek A

# Identyfikacja użytkownika i obsługa sesji

HTTP jest protokołem bezstanowym, w związku z czym, standardowo, serwer nie potrafi identyfikować serii połączeń z jednym użytkownikiem[1]. Są jednak w systemie scenariusze wielokrokowe (np. rejestracja), które wymagają takiej identyfikacji. Rozwiązaniem jest mechanizm sesji. *Sesją użytkownika* nazwiemy każdą serię połączeń między tym użytkownikiem a systemem. Każda sesja posiada unikalny identyfikator (SID). Dzięki temu system potrafi zapamiętać dane zebrane w trakcie interakcji z użytkownikiem na przestrzeni kilku kroków.

Istotnym jest, aby SID był przekazywany zarówno w żądaniach od użytkownika do systemu, jak w drodze powrotnej. W tym celu wykorzystany został klasyczny mechanizm obsługi sesji[2]:

1. SID przekazywany jest, obok innych parametrów żądania użytkownika takich nazwa operacji do wykonania czy typ medium, za pomocą metody GET protokołu HTTP[3].

2. Natomiast w drodze powrotnej SID jest dodawany jako parametr do każdego odnośnika i każdego formularza w zwracanym dokumencie.

Za wyekstrahowanie SID z nagłówka HTTP (część 1-sza mechanizmu) odpowiedzialny stał się APr. Do realizacji części 2-ej w sposób naturalny został wyznaczony ATW. Ponieważ SID musi być przekazywany między APr a ATW, dokonaliśmy rozszerzenia *SystemOntology* (ontologii wiadomości, które krążą w systemie), a dokładniej jej konceptu *MessageInfo*, o dodatkowe pole *SessionID*.

W związku z przyjętym mechanizmem sesji rodzą się pytania:

1. W jaki sposób generować SID?

2. W jaki sposób przechowywać dane związane ze scenariuszami wielokrokowymi?

[1] Ta część systemu wymagała przeprogramowania szkieletu systemu i brał w niej udział Paweł Kaczmarek. Jest on również współautorem tego dodatku.

[2] Patrz paragraf 2.4.1.

[3] Rozwiązanie z użyciem ciasteczek jest niemożliwe przy poczynionych założeniach o „lekkości" medium klienta systemu (patrz paragraf 3.1.3).

W celu rozwiązania tych problemów stworzyliśmy *Agenta Obsługującego Sesję*[4] (AOS). Niżej prezentujemy listę ról w nim zawartych.

**Rola: Obsługa sesji**

Każde żądanie użytkownika, które trafia do APr, jest przekazywane do AOS. Każde żądanie, które nie zawiera SID, rozpoczyna sesję. W tym celu AOS generuje nowy SID na podstawie trzech zmiennych: numeru IP maszyny użytkownika[5] (otrzymanego od APr), bieżącego czasu i pewnej liczby losowej[6]. Stworzony SID dołączany jest do bieżącego żądania.

Ponieważ AOS jest odpowiedzialny za scenariusze wielokrokowe, takie jak rejestracja, udostępnia swoim zachowaniom tymczasową tablicę do zapamiętywania na czas sesji danych związanych z takim scenariuszem.

**Rola: Logowanie zachowań i udostępnianie ich bazy**

Z jednej strony mamy AOS, który gra rolę „strażnika" w systemie: obserwuje wszystkie żądania użytkownika. Z drugiej strony proces uczenia wymaga uprzedniego zebrania informacji zwrotnej. W rezultacie obowiązek ten spadł na AOS, który zapisuje w modelu bazy danych (`events-model`) wszystkie zachowania określone w paragrafie 5.1.3. Jednocześnie AOS udostępnia zachowania pozwalające innym agentom na dostęp do tak zgromadzonych zachowań.

**Rola: Identyfikacja użytkownika (zarządzanie użytkownikami)**

Ani numer IP użytkownika, ani SID z nim związany nie pozwalają na określenie tożsamości użytkownika. Dopiero dołączenie *identyfikatora użytkownika* (UID) do przesyłanych komunikatów – podobnie jak to czynimy z SID – pozwala rozwiązać ten problem. Użytkownik posiada UID tylko wtedy, jeśli zarejestrował się w systemie. Jednak dopiero po zalogowaniu może nawiązać dialog z AO i zażądać informacji spersonalizowanej.

**Rejestracja nowego użytkownika** Rejestracja pełni dwie funkcje. Po pierwsze – nowy użytkownik dodawany jest do listy zarejestrowanych użytkowników (`users-model`), którzy mogą korzystać z faktycznych możliwości systemu (czyli wyszukiwania restauracji). Po drugie – służy stworzeniu profilu użytkownika na podstawie podanych danych[7]. Diagram A.1 prezentuje dokładnie proces rejestracji.

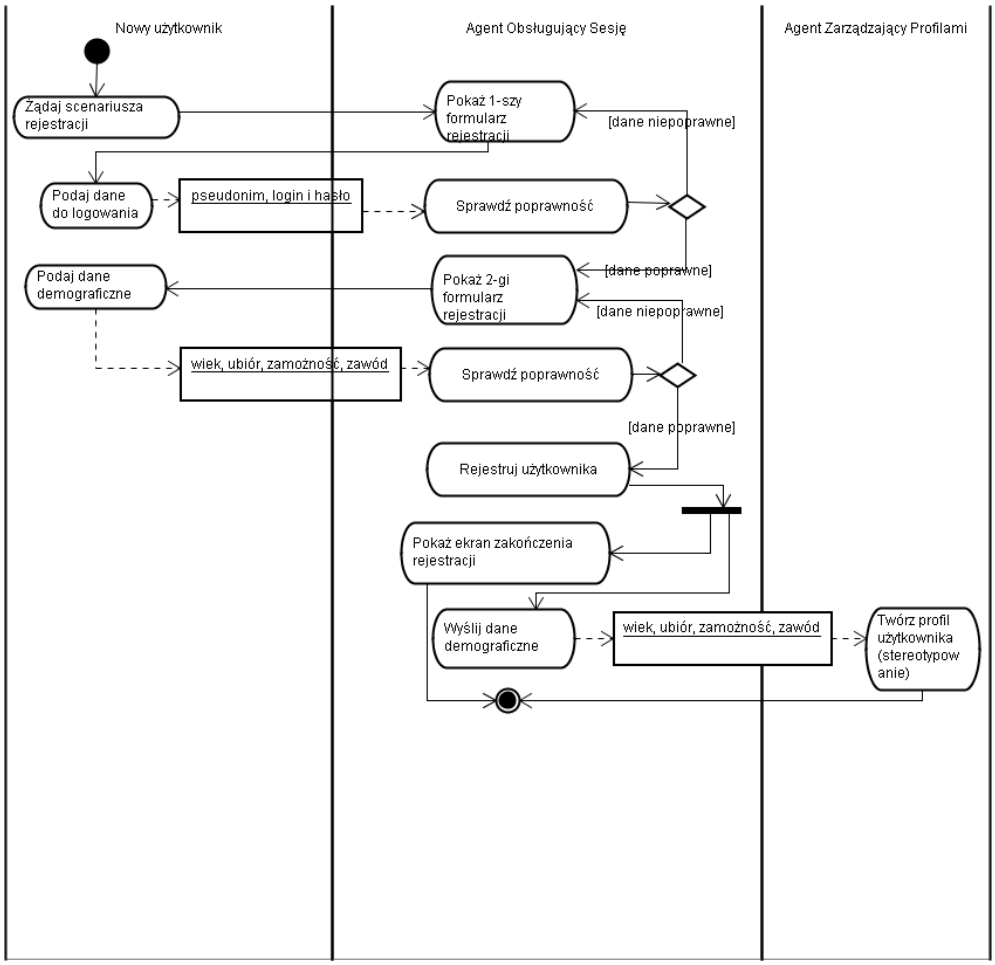
**Logowanie i wylogowanie** Każdy zarejestrowany w systemie użytkownik ma możliwość zalogowania się do systemu. W momencie zalogowania się, AOS tworzy AO i żąda od niego wysłania do użytkownika ekranu powitalnego. Od tej pory, aż do momentu wylogowania wszelkie żądania użytkownika będą obsługiwane przez jego AO. Proces logowania w sposób szczegółowy został opisany na rysunku A.2

[4] Ang. *Session Handling Agenta* (SHA).

[5] Ang. *Internet Protocol*.

[6] Dokładniej, wykorzystywany jest uproszczony algorytm zaadaptowany z języka PHP: szyfrujący SID: `SID = szyfrujDES(IP_klienta . czas . liczba_losowa)`.

[7] Paragraf 5.1.2 opisuje dokładnie sam algorytm inicjalizacji profilu.



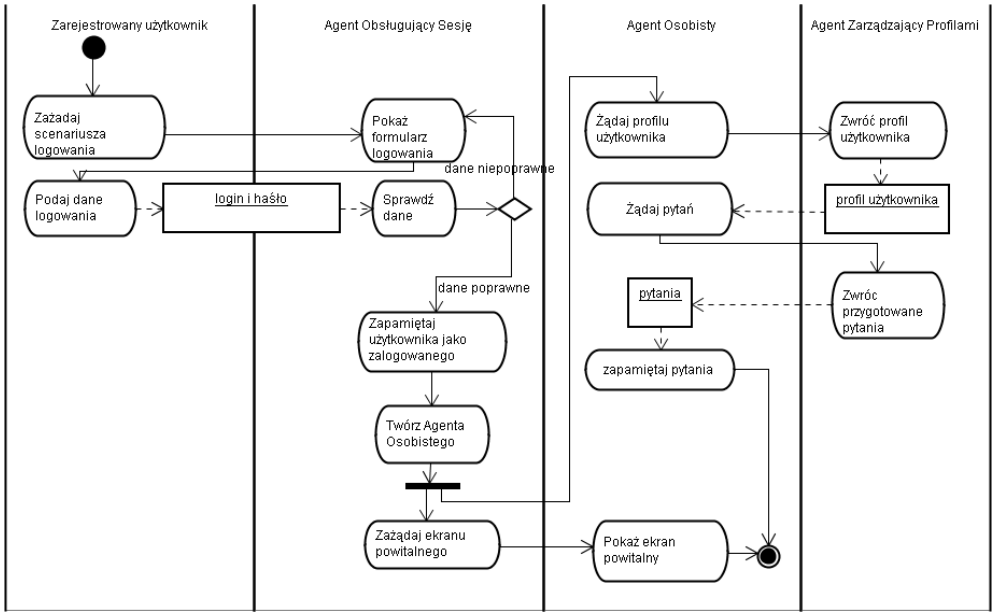

Rysunek A.1: Diagram aktywności dla scenariusza rejestracji.

W momencie wylogowania uruchamiany jest algorytm przygotowywania pytań[8] na następną sesję. Po jego zakończeniu AO wysyła gotowe pytania do AZP (aby ten zapisał je w modelu `questions-model`) i dokonuje usunięcie samego siebie z systemu.

### Rola: Brokering żądań

Zauważmy, że AOS obsługuje dwa rodzaje żądań: (a) *zewnętrzne*, przychodzące od użytkownika (za pośrednictwem APr) i (b) *wewnętrzne*, pochodzące od innych agentów w systemie (przykładowo: prośba o podesłanie fragmentu zalogowanej *Historii*.) W celu rozróżnienia tych żądań AO sprawdza, czy zachowanie odpowiedzialne za jego realizację implementuje interfejs `InClientSessionHandler`. Jeśli tak, JEST to znak, że żądanie pochodzi od użytkownika i należy sprawdzić, czy wymaga ono od użytkownika bycia zalogowanym i ewentualnie wymusza rozpoczęcie takiego procesu logowania. Wszystkie żądania pochodzące od użytkownika zalogowanego (na przykład żądanie znalezienia restauracji) przekazywane są do realizacji do AO. Do zadań nie wymagających logowania, czyli tych, za które odpowiedzialne jest odpowiednie zachowanie AOS, należy na przykład obsługa procesu rejestracji.

[8]Patrz paragraf 5.1.3




Rysunek A.2: Diagram aktywności dla scenariusza logowania.

Jeśli żądanie pochodzi od innych agentów w systemie, wtedy jego realizacją zajmuje się odpowiednie zachowanie w AOS.

# Dodatek B

# Ontologie występujące w systemie

Ontologie stanowią kluczowy element opisywanego systemu. Każda z ontologii składa się z miniontologii, reprezentowanych przez jeden lub więcej plików. Poniżej prezentuję najważniejsze z nich dla lepszego zrozumienia działania systemu.

## B.1 Ontologia restauracji

Ontologia restauracji definiuje sposób opisywania lokali gastronomicznych w zgromadzonej bazie danych restauracji. Składają się na nią następujące miniontologie:

- *ontologia lokacji*:
  `location/AttractionCategoryCode.n3`, wprowadza rozmaite typy miejsc, gdzie ludzie spędzają wolny czas,
  `location/IndexPointCode.n3`, wprowadza możliwe obiekty potrzebne do określenia względnego położenia lokacji,
  `location/Location.n3`, wprowadza pojęcie klasy lokacji,
  `location/LocationCategoryCode.n3`, wprowadza pojęcie kategorii lokacji (miasto, góra itp.)

- *ontologia pieniądza*:
  `money/FuzzyPriceCode.n3`, wprowadza sześciostopniową skalę wysokości ceny,
  `money/MeanOfPayment.n3`, opisuje honorowane sposoby płatności (gotówka, karta kredytowa itp.),

- *ontologia właściwa restauracji*:
  `restaurant/AccessibilityCode.n3`, uwzględnia możliwości dostępu do lokalu przez osoby niepełnosprawne,
  `restaurant/AlcoholCode.n3`, określa alkohol dostępny w lokalu,
  `restaurant/CuisineCode.n3`, definiuje możliwe serwowane kuchnie,
  `restaurant/DinerReview.n3`, wprowadza klasę recenzji o restauracji,
  `restaurant/DressCode.n3`, definiuje rodzaj obowiązkowego stroju (np. strój formalny)
  `restaurant/FeatureCode.n3`, określa szczególne cechy lokalu, takie jak np. możliwość skorzystania z Internetu w lokalu,

`restaurant/ParkingCode.n3`, określa rodzaj parkingu towarzyszącego restauracji

`restaurant/ReservationCode.n3`, definiuje możliwości rezerwacji,

`restaurant/Restaurant.n3`, wprowadza definicję klasy restauracji,

`restaurant/RestaurantCategoryCode.n3`, określa typ restauracji (przykładowo kafeteria),

`restaurant/RestaurantServiceInfo.n3`, określa dodatkowe usługi, które może prowadzić restauracja, np. organizacja przyjęć albo przyrządzanie potraw na specjalnie zamówienie.

`restaurant/SmokingCode.n3`, definiuje dopuszczalność palenia w lokalu.

Plik `restaurant/Restaurant.n3` definiuje klasę restauracji (`Restaurant`), dziedziczącej po klasie lokacji (`Location`) i jej atrybuty, takie jak serwowana kuchnia czy ułatwienia dla niepełnosprawnych.

```
1  # author: Michal Szymczak, wildmike@tlen.pl
2  # author: Maciej Gawinecki, mg@bydnet.pl
3
4  @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
5  @prefix dc: <http://purl.org/dc/elements/1.0/#>.
6  @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
7  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
8
9  @prefix : <http://www.agentlab.net/schemas/restaurant#>.
10 @prefix mon: <http://www.agentlab.net/schemas/money#>.
11 @prefix loc: <http://www.agentlab.net/schemas/location#>.
12
13 :Restaurant rdfs:subClassOf loc:Location.
14
15 # previously: r:id (in chefmoz)
16 :id              a rdf:Property;
17                  rdfs:range xsd:string;
18                  rdfs:label "ID of restaurant";
19                  rdfs:comment "ID of restaurant";
20                  rdfs:domain :Restaurant.
21
22 :title           a rdf:Property;
23                  rdfs:label "restaurant name";
24                  rdfs:comment "Restaurant name written using Dublin
                        Core Title element";
25                  rdfs:range xsd:string;
26                  rdfs:domain :Restaurant;
27                  = dc:title.
28
29
30 ################################################
31 # BEGIN: extensions of Location ontology
32 ################################################
33
34 :deliveryPhone   a rdf:Property;
35                  rdfs:subPropertyOf loc:phone;
```

```
36                  rdfs:comment "A restaurant's delivery phone number
                        . Defined only if there is a separate phone
                        number for delivery. Same format as Phone.";
37                  rdfs:domain :Restaurant.
38
39  ###############################################
40  # END: extensions of Location ontology
41  ###############################################
42
43  :URL            a rdf:Property;
44                  rdfs:label "Restaurant's web page";
45                  rdfs:comment "A restaurant's main web page.";
46                  rdfs:domain :Restaurant;
47                  rdfs:range xsd:anyURI.
48
49  :deliveryURL    a rdf:Property;
50                  rdfs:label "delivery web page";
51                  rdfs:comment "An URL where the user may order food
                        from this restaurant online.";
52                  rdfs:domain :Restaurant;
53                  rdfs:range xsd:anyURI.
54
55  :reservationURL a rdf:Property;
56                  rdfs:label "reservation web page";
57                  rdfs:comment "An URL where the user may make
                        reservations for this restaurant.";
58                  rdfs:domain :Restaurant;
59                  rdfs:range xsd:anyURI.
60
61  :menuURL        a rdf:Property;
62                  rdfs:label "menu web page";
63                  rdfs:comment "The URL of the restaurant's online
                        menu.";
64                  rdfs:domain :Restaurant;
65                  rdfs:range xsd:anyURI.
66
67  :cuisine        a rdf:Property;
68                  rdfs:label "cuisine";
69                  rdfs:comment "The type of food a restaurant serves
                        . We repeat this field up to three times.";
70                  rdfs:domain :Restaurant;
71                  rdfs:range :CuisineCode.
72
73  :dinnerPrice    a rdf:Property;
74                  rdfs:label "price";
75                  rdfs:range mon:FuzzyPriceCode;
76                  rdfs:comment "The cost of an average dinner at
                        this restaurant, including entree, non-
                        alcoholic drink, and half an appetizer or
                        dessert. If the restaurant does not serve
                        dinner, we use the closest meal.";
77                  rdfs:domain :Restaurant.
78
```

```
79  :breakfastPrice a rdf:Property;
80                  rdfs:label "breakfast price";
81                  rdfs:range mon:FuzzyPriceCode;
82                  rdfs:comment "Breakfast Price.";
83                  rdfs:domain :Restaurant.
84
85  :lunchPrice     a rdf:Property;
86                  rdfs:label "lunch price";
87                  rdfs:range mon:FuzzyPriceCode;
88                  rdfs:comment "Lunch Price.";
89                  rdfs:domain :Restaurant.
90
91  :hours          a rdf:Property;
92                  rdfs:label "hours";
93                  rdfs:comment "A string describing the hours the
                        restaurant is open.";
94                  rdfs:range xsd:string;
95                  rdfs:domain :Restaurant.
96
97  :parsedHours    a rdf:Property;
98                  rdfs:label "parsing hours";
99                  rdfs:comment "Our best guess at parsing the hours
                        .";
100                 rdfs:range xsd:string;
101                 rdfs:domain :Restaurant.
102
103 :accepts        a rdf:Property;
104                 rdfs:label "accepts";
105                 rdfs:comment "Payment method accepted by this
                        restaurant. Expect several of these for each
                        restaurant. Comment: All restaurants accept
                        cash, so we don't list it.";
106                 rdfs:range mon:MeanOfPayment;
107                 rdfs:domain :Restaurant.
108
109 :alcohol        a rdf:Property;
110                 rdfs:label "alcohol";
111                 rdfs:comment "A string describing the alcohol
                        service.";
112                 rdfs:range :AlcoholCode;
113                 rdfs:domain :Restaurant.
114
115 :smoking        a rdf:Property;
116                 rdfs:label "smoking-friendlyness";
117                 rdfs:comment "A string describing how smoking-
                        friendly the restaurant is.";
118                 rdfs:range :SmokingCode;
119                 rdfs:domain :Restaurant.
120
121 :dress          a rdf:Property;
122                 rdfs:label "acceptable dress";
123                 rdfs:comment "A string describing acceptable dress
                         for the restaurant.";
```

```
124                 rdfs:range :DressCode;
125                 rdfs:domain :Restaurant.
126
127 :feature         a rdf:Property;
128                 rdfs:label "restaurant feature";
129                 rdfs:comment "A string describing a feature of the
                        restaurant. One of these tags exists for each
                        of the restaurant's features.";
130                 rdfs:range :FeatureCode;
131                 rdfs:domain :Restaurant.
132
133 :accessibility  a rdf:Property;
134                 rdfs:label "accessibility";
135                 rdfs:comment "String describing how handicapped-
                        accessible the restauarant is.";
136                 rdfs:range :AccessibilityCode;
137                 rdfs:domain :Restaurant.
138
139 :accessibilityNotes a rdf:Property;
140                 rdfs:label "accessibility Notes";
141                 rdfs:comment "Details on a restaurant's
                        handicapped accessibility.";
142                 rdfs:range xsd:string;
143                 rdfs:domain :Restaurant.
144
145 :parking        a rdf:Property;
146                 rdfs:label "parking";
147                 rdfs:comment "A string describing parking options.
                        One of these tags per parking option.";
148                 rdfs:range :ParkingCode;
149                 rdfs:domain :Restaurant.
150
151 :reservations   a rdf:Property;
152                 rdfs:label "reservations";
153                 rdfs:comment "String describing whether
                        reservations are accepted.";
154                 rdfs:range :ReservationCode;
155                 rdfs:domain :Restaurant.
156
157 :capacity       a rdf:Property;
158                 rdfs:label "capacity";
159                 rdfs:comment "Maximum number of people the
                        restaurant can hold.";
160                 rdfs:range xsd:nonNegativeInteger;
161                 rdfs:domain :Restaurant.
162
163 :largestParty   a rdf:Property;
164                 rdfs:label "largest party";
165                 rdfs:comment "Largest group size that the
                        restaurant can comfortably seat. Reservations
                        in advance may be necessary.";
166                 rdfs:range xsd:nonNegativeInteger;
167                 rdfs:domain :Restaurant.
```

```
168
169 :clientele       a rdf:Property;
170                  rdfs:label "clientele";
171                  rdfs:comment "The type of people who usually
                        frequent this restaurant.";
172                  rdfs:range xsd:string;
173                  rdfs:domain :Restaurant.
174
175 :hasDinerReview a rdf:Property;
176                  rdfs:label "diner review";
177                  rdfs:comment "Reference to Diner Review.";
178                  rdfs:range :DinerReview;
179                  rdfs:domain :Restaurant.
180
181 :restaurantCategory      a rdf:Property;
182                  rdfs:label "category of a restaurant";
183                  rdfs:comment "Category describes general features
                        of a restaurant.";
184                  rdfs:range :RestaurantCategoryCode;
185                  rdfs:domain :Restaurant.
186
187 :restaurantService   a rdf:Property;
188                  rdfs:label "restaurant service information";
189                  rdfs:comment "Restaurant Service Information.
                        Could be similar to restaurant category.";
190                  rdfs:range :RestaurantServiceInfo;
191                  rdfs:domain :Restaurant.
```

## B.2 Ontologia profilu

Każdy profil, zarówno użytkownika jak i stereotypu posiada wspólny atrybut: `OpinionsSet` (zbiór opinii), zdefiniowany w *ontologii opinii*. Natomiast to, co odróżnia profil użytkownika od profilu stereotypu to sposób opisywania danych. Elementy danych w profilu użytkownika są jednego z typów prostych: `Interval` (typ przedziałowy), `Ordinal` (typ porządkowy), `Nominal` (typ nominalny). Natomiast elementy danych w profilu stereotypu określone są przez typy złożone, zbudowane na bazie typów prostych `IntervalSet`, `OrdinalSet`, `NominalSet` – odpowiednio. Celowość takiego rozgraniczenia typów wynika z mechanizmu stereotypizacji[1].

Typy proste i złożone zostały zdefiniowane w *ontologii miar danych*. Na ich bazie powstały typy wykorzystywane w dalszych profilach, a opisywane w *ontologii danych profilu*.

### B.2.1 Ontologia właściwa profilu

Ontologia właściwa profilu (`profile/Profile.n3`) opisuje wspólne elementy profilu użytkownika i profilu stereotypu.

```
1 # author: Maciej Gawinecki, mg@bydnet.pl
```

[1] Patrz paragraf **??**

```
2
3  @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
4  @prefix dc: <http://purl.org/dc/elements/1.0/#>.
5  @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
6  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
7
8  @prefix : <http://www.agentlab.net/schemas/system#>.
9
10 :Profile          a rdfs:Class.
11
12 :hasOpinions      a rdf:Property;
13                   rdfs:range :OpinionsSet;
14                   rdfs:domain :Profile.
```

### B.2.2 Ontologia opinii

Ontologia opinii (`profile/OpinionsSet.n3`) wprowadza definicję klasy opinii (`Opinion`).

```
1  # author: Maciej Gawinecki, mg@bydnet.pl
2
3  @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
4  @prefix dc: <http://purl.org/dc/elements/1.0/#>.
5  @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
6  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
7
8  @prefix : <http://www.agentlab.net/schemas/system#>.
9
10 :OpinionsSet       a rdfs:Class.
11
12 :wasChanged        a rdf:Property;
13                   rdfs:range xsd:date;
14                   rdfs:label "when this profile was changed
                         lastly";
15                   rdfs:domain :OpinionsSet.
16
17 # Below: depends whether belongs to user profile or stereotype.
18
19 :hasStereotypeID  a rdf:Property;
20                   rdfs:range xsd:integer;
21                   rdfs:label "ID of the stereotype this profile
                         belongs to";
22                   rdfs:domain :OpinionsSet.
23
24 :hasUserID        a rdf:Property;
25                   rdfs:range xsd:integer;
26                   rdfs:label "ID of the user this profile belongs
                          to";
27                   rdfs:domain :OpinionsSet.
28
29 ##########
30
31 :containsOpinion  a rdf:Property;
32                   rdfs:range :Opinion;
```

```
33                    rdfs:label "the user has opinion";
34                    rdfs:domain :OpinionsSet.
35
36 :Opinion           a rdfs:Class.
37
38 :Classification    a rdfs:Class.
39
40 :Interesting       a :Classification;
41                    rdfs:label "interesting".
42
43 :NotInteresting    a :Classification;
44                    rdfs:label "not interesting".
45
46 :NotClassified     a :Classification;
47                    rdfs:label "not classified".
48
49 :hasClassification a rdf:Property;
50                    rdfs:range :Classification;
51                    rdfs:label "classification of opinion";
52                    rdfs:domain :Opinion.
53
54 :about             a rdf:Property;
55                    rdfs:range rdfs:Class;
56                    rdfs:label "the concept this opinion is about";
57                    rdfs:domain :Opinion.
58
59 :atIndividualProbability  a rdf:Property;
60                    rdfs:range xsd:decimal;
61                    rdfs:label "individual probability";
62                    rdfs:domain :Opinion.
63
64 :atNormalizedProbability  a rdf:Property;
65                    rdfs:range xsd:decimal;
66                    rdfs:label "normalized probability";
67                    rdfs:domain :Opinion.
68
69 :atInferredProbability    a rdf:Property;
70                    rdfs:range xsd:decimal;
71                    rdfs:label "infferred from domain probability";
72                    rdfs:domain :Opinion.
```

### B.2.3 Ontologia miar danych

Ontologia miar danych (`profile/Measures.n3`) wprowadza typy proste i złożone opisane wyżej.

```
1 # author: Maciej Gawinecki, mg@bydnet.pl
2
3 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
4 @prefix dc: <http://purl.org/dc/elements/1.0/#>.
5 @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
6 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
7
```

```
@prefix : <http://www.agentlab.net/schemas/system#>.

:Nominal            a rdfs:class.

:contains           a rdf:Property;
                    rdfs:range (:Nominal :Ordinal);
                    rdfs:domain (:NominalSet :OrdinalSet).

:Ordinal            a rdfs:class.

:OrdinalSet         rdfs:subclassOf :Set.

:hasRank            a rdf:Property;
                    rdfs:range xsd:integer;
                    rdfs:comment "rank of instance of Ordinal
                       dautum";
                    rdfs:domain :Ordinal.

:Interval           rdfs:subclassOf xsd:integer.

:IntervalSet         rdfs:subclassOf :Set.

:hasLeftBound       a rdf:Property;
                    rdfs:range xsd:integer;
                    rdfs:domain :IntervalSet.

:hasRightBound      a rdf:Property;
                    rdfs:range xsd:integer;
                    rdfs:domain :IntervalSet.

:Set                a rdfs:class.

:NominalSet         rdfs:subclassOf :Set.
```

### B.2.4 Ontologia danych profilu

Ontologia danych profilu (`profile/ProfileData.n3`) opisuje elementy danych profilu:

- użytkownika: wiek (`Age` typu `Interval`), zawód (`Profession` typu `Nominal`), zamożność (`Wealth` typu `Ordinal`) i sposób ubioru (`Dress` typu `Ordinal`).
- stereotypu: przedział wiekowy (`AgeSet` typu `IntervalSet`), zbiór zawodów (`ProfessionSet` typu `NominalSet`), zbiór zamożności (`WealthSet` typu `OrdinalSet`) i zbiór sposobów ubioru (`DressSet` typu `OrdinalSet`).

Jednocześnie ontologia ta wprowadza instancje poszczególnych typów.

```
# author: Maciej Gawinecki, mg@bydnet.pl

@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix dc: <http://purl.org/dc/elements/1.0/#>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
```

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

@prefix : <http://www.agentlab.net/schemas/system#>.


:Age                rdfs:subclassOf :Interval.

:Wealth             rdfs:subclassOf :Ordinal.

:Profession         rdfs:subclassOf :Nominal.

:Dress              rdfs:subclassOf :Ordinal.

:AgeSet             rdfs:subclassOf :IntervalSet.

:WealthSet          rdfs:subclassOf :OrdinalSet.

:ProfessionSet      rdfs:subclassOf :NominalSet.

:DressSet           rdfs:subclassOf :OrdinalSet.

# Wealth instances

:NotRich                    a :Wealth;
                            :hasRank 1.

:AverageRich                a :Wealth;
                            :hasRank 2.

:Rich                        a :Wealth;
                             :hasRank 3.

:VeryRich                    a :Wealth;
                             :hasRank 4.

# Profession instances

:StudentPupil                a :Profession.

:PensionerAnnuitant          a :Profession.

:ScientistTeacher            a :Profession.

:UnemployedJobSeeker         a :Profession.

:Handworker                  a :Profession.

:AdvertisingMarketingWorker  a :Profession.

:ServicesTradeWorker         a :Profession.

:SpecialistFreeLancer        a :Profession.

```

```
59 :ManagerDirector              a :Profession.
60
61 :OtherProfession              a :Profession.
62
63 # Dress instances
64
65 :SportyDress                  a :Dress;
66                                :hasRank 1.
67
68 :NaturalDress                 a :Dress;
69                                :hasRank 2.
70
71 :ElegantDress                 a :Dress;
72                                :hasRank 3.
73
74 :OtherDress                   a :Dress;
75                                :hasRank 4.
```

## B.3 Ontologia profilu użytkownika

Ontologia profilu użytkownika korzysta z ontologii profilu wyprowadzając klasę *profilu użytkownika* (`UserProfile`) z ogólnej klasy *profilu* (`Profile`). Składa się z dwóch miniontologii: *ontologii właściwej profilu użytkownika* i *ontologii danych profilu użytkownika*.

### B.3.1 Ontologia właściwa profilu użytkownika

Zdefiniowana w `profile/UserProfile.n3`.

```
1  # author: Maciej Gawinecki, mg@bydnet.pl
2
3  @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
4  @prefix dc: <http://purl.org/dc/elements/1.0/#>.
5  @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
6  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
7
8  @prefix : <http://www.agentlab.net/schemas/system#>.
9
10 :UserProfile     rdfs:subclassOf :Profile.
11
12 :hasUserID       a rdf:Property;
13          rdfs:range xsd:integer;
14          rdfs:domain :UserProfile.
15
16 :hasUserProfileData    a rdf:Property;
17          rdfs:range :UserProfileData;
18          rdfs:domain :UserProfile.
```

### B.3.2 Ontologia danych profilu użytkownika

Zdefiniowana w `profile/UserProfileData.n3`. Korzysta z *ontologii danych profilu*.

```
# author: Maciej Gawinecki, mg@bydnet.pl

@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix dc: <http://purl.org/dc/elements/1.0/#>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

@prefix : <http://www.agentlab.net/schemas/system#>.

:UserProfileData     a rdfs:Class.

:wasBorn          a rdf:Property;
                  rdfs:range xsd:Date;
                  rdfs:domain :UserProfileData.

:hasAge           a rdf:Property;
                  rdfs:range :Age;
                  rdfs:domain :UserProfileData.

:hasWealth        a rdf:Property;
                  rdfs:range :Wealth;
                  rdfs:domain :UserProfileData.

:hasProfession    a rdf:Property;
                  rdfs:range :Profession;
                  rdfs:domain :UserProfileData.

:hasDress         a rdf:Property;
                  rdfs:range :Dress;
                  rdfs:domain :UserProfileData.
```

## B.4 Ontologia profilu stereotypu

Ontologia profilu stereotypu korzysta z ontologii profilu wyprowadzając klasę *profilu stereotypu* (`StereotypeProfile`) z ogólnej klasy *profilu* (`Profile`). Składa się z dwóch miniontologii: *ontologii właściwej profilu stereotypu* i *ontologii danych profilu stereotypu*.

### B.4.1 Ontologia właściwa profilu stereotypu

Zdefiniowana w `profile/StereotypeProfile.n3`.

```
# author: Maciej Gawinecki, mg@bydnet.pl

@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix dc: <http://purl.org/dc/elements/1.0/#>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
```

```
7
8  @prefix : <http://www.agentlab.net/schemas/system#>.
9
10 :StereotypeProfile     rdfs:subclassOf :Profile.
11
12 :stereotypeID      a rdf:Property;
13         rdfs:range xsd:integer;
14         rdfs:domain :StereotypeProfile.
15
16 :hasStereotypeData     a rdf:Property;
17         rdfs:range :StereotypeData;
18         rdfs:domain :StereotypeProfile.
```

### B.4.2 Ontologia danych stereotypu

Zdefiniowana w `profile/StereotypeProfileData.n3`. Korzysta z *ontologii danych profilu*.

```
1  # author: Maciej Gawinecki, mg@bydnet.pl
2
3  @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
4  @prefix dc: <http://purl.org/dc/elements/1.0/#>.
5  @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
6  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
7
8  @prefix : <http://www.agentlab.net/schemas/system#>.
9
10 :StereotypeProfileData a rdfs:class.
11
12 :hasName             a rdf:Property;
13                       rdfs:range xsd:string;
14                       rdfs:domain :StereotypeProfileData.
15
16 :hasAgeSet           a rdf:Property;
17                       rdfs:range :AgeSet;
18                       rdfs:domain :StereotypeProfileData.
19
20 :hasWealthSet        a rdf:Property;
21                       rdfs:range :WealthSet;
22                       rdfs:domain :StereotypeProfileData.
23
24 :hasProfessionSet    a rdf:Property;
25                       rdfs:range :ProfessionSet;
26                       rdfs:domain :StereotypeProfileData.
27
28 :hasDressSet         a rdf:Property;
29                       rdfs:range :DressSet;
30                       rdfs:domain :StereotypeProfileData.
```

## B.5 Ontologia zdarzenia

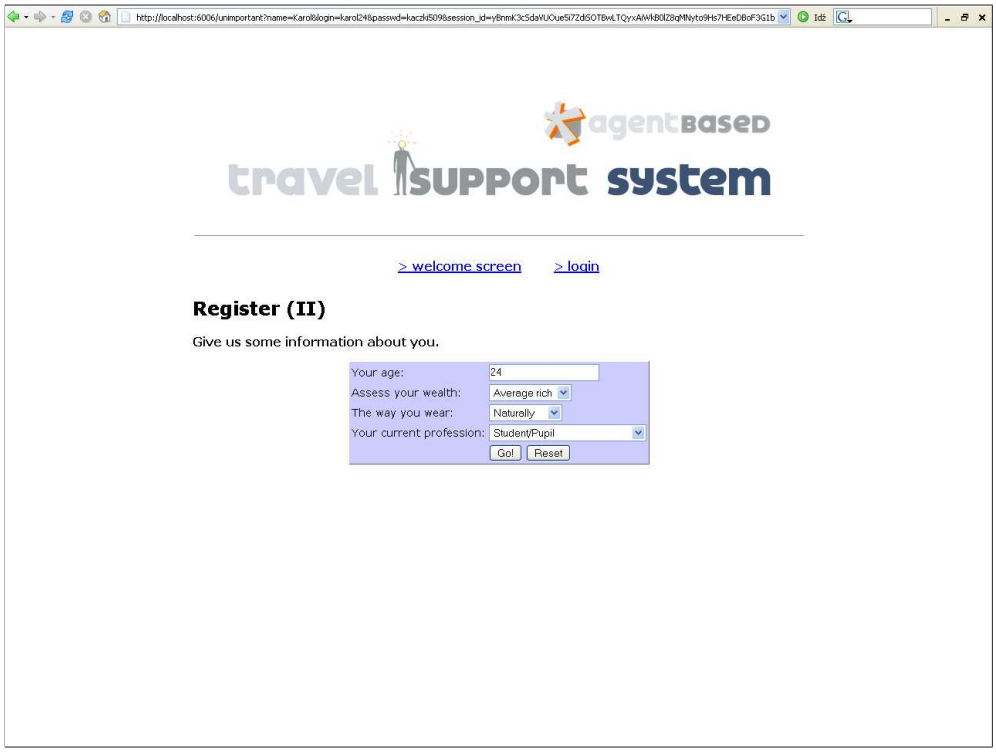
```
1  # author: Maciej Gawinecki, mg@bydnet.pl
2
3  @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
4  @prefix dc: <http://purl.org/dc/elements/1.0/#>.
5  @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
6  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
7
8  @prefix : <http://www.agentlab.net/schemas/system#>.
9
10 :Event a rdfs:Class.
11
12 :hasEventID       a rdf:Property;
13                   rdfs:range xsd:integer;
14                   rdfs:label "ID of the event";
15                   rdfs:comment "ID of the event";
16                   rdfs:domain :Event.
17
18 :hasUserID        a rdf:Property;
19                   rdfs:range xsd:integer;
20                   rdfs:label "ID of actor of the event";
21                   rdfs:comment "ID of actor of the event";
22                   rdfs:domain :Event.
23
24 :hasSessionID     a rdf:Property;
25                   rdfs:range xsd:integer;
26                   rdfs:label "ID of session, when of the event
                        occured";
27                   rdfs:comment "ID of session, when of the event
                        occured";
28                   rdfs:domain :Event.
29
30 # TODO: find range describing date/time from any known ontology
31 :when          a rdf:Property;
32                   rdfs:range xsd:string;
33                   rdfs:label "when the event happened";
34                   rdfs:comment "when the event happened";
35                   rdfs:domain :Event.
36
37 :hasExtendedBehaviour   a rdf:Property;
38                   rdfs:range :ExtendedBehaviour;
39                   rdfs:label "type of user's behavoiur";
40                   rdfs:comment "type of user's bevaiour";
41                   rdfs:domain :Event.
42
43 :ExtendedBehaviour a rdfs:Class.
44
45 :hasBehaviour    a  rdf:Property;
46                   rdfs:range :Behaviour;
47                   rdfs:domain :ExtendedBehaviour.
48
49 :hasContext    a rdf:Property;
50                   rdfs:range :Context;
```

```
51                  rdfs:domain :ExtendedBehaviour.
52
53 :Behaviour a rdfs:Class.
54
55 :ClickForRestaurantDetailsBehaviour    a :Behaviour.
56 :QueryForRestaurantBehaviour           a :Behaviour.
57 :RateRestaurantPositiveBehaviour       a :Behaviour.
58 :EntrySearchingBehaviour               a :Behaviour.
59 :ExitSearchingBehaviour                a :Behaviour.
60 :NotLoggedBehaviour                    a :Behaviour.
61
62 :Context a rdfs:Class.
63
64 :hasParameter  a rdf:Property;
65                 rdfs:range :Parameter;
66                 rdfs:domain :Context.
67
68 :Parameter a rdfs:Class.
69
70 :hasName a rdf:Property;
71                 rdfs:range :ParameterName;
72                 rdfs:domain :Parameter.
73
74 :ParameterName = xsd:string.
75
76 :hasTargetURI a :parameterName.
77
78 :hasValue a rdf:Property;
79                 rdfs:range :ParameterValue;
80                 rdfs:domain :Parameter.
81
82 :ParameterValue = xsd:string.
```

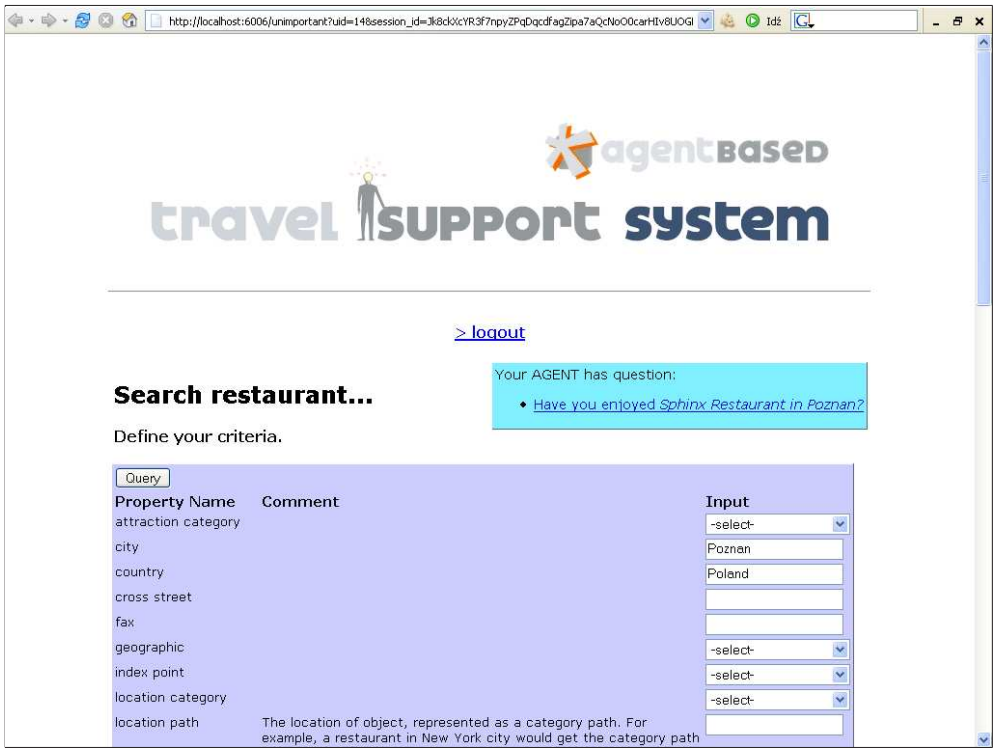
# Dodatek C

# Interfejs powstałej aplikacji

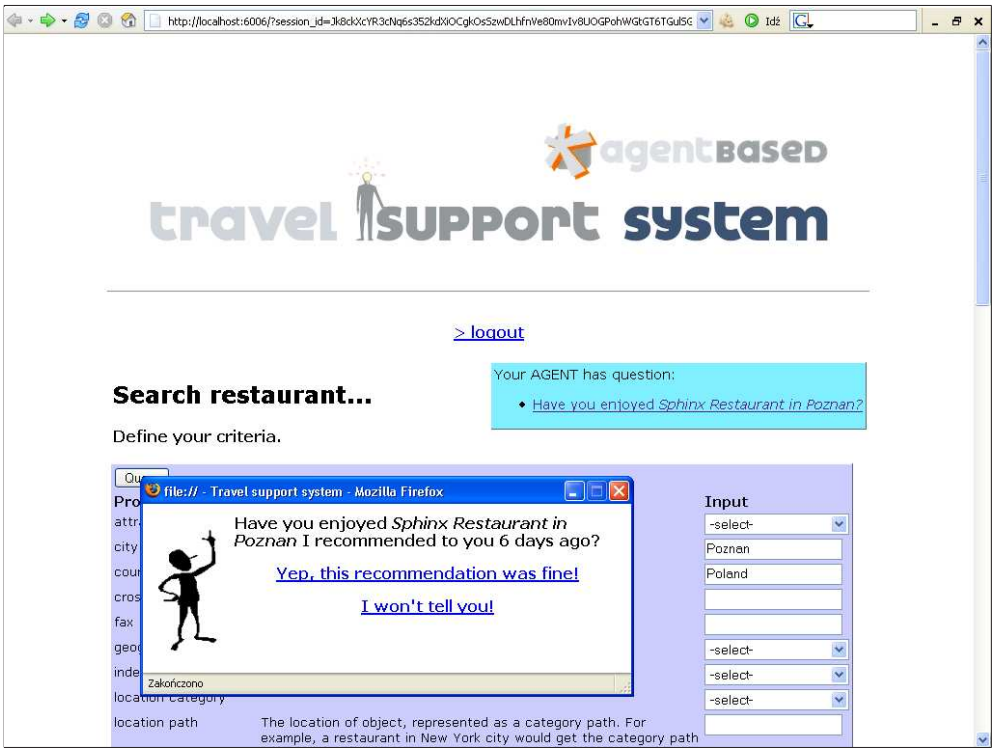

Rysunek C.1: Ekran rejestracji nowego użytkownika w systemie (I).




Rysunek C.2: Ekran rejestracji nowego użytkownika w systemie (II).




Rysunek C.3: Ekran wyszukiwania restauracji. W niebieskim prostokącie znajduje się pytanie Agenta Osobistego.




Rysunek C.4: Ekran głosowania na restaurację. Użytkownik ma możliwość oddania pozytywnego głosu lub zaniechania głosowania (równoznaczne z zamknięciem okna).

# Dodatek D

# Summary (in English)

User modeling is a well-known technique for delivering personalized and relevant information to the user. In this work I discuss how it can be utilized in the context of the Semantic Web. More precisely, I propose whole process of construction and management of user models (profiles) for an agent-based travel support system. Since travel ontology is the centerpiece of the system it is naturally to represent user profiles as sets of statements about the "world of travel". This solution seems reasonable, allowing to preserve existing user profiles when the domain ontology is still in development phase. Probability-like measure applied to each statement models strengths of user preferences. These probabilities are derived from observations of implicit and explicit user feedback. Experience derived from existing recommendation systems proves that both types of feedbacks are crucial for efficiency of recommending process. However, the WWW interface brings limitations to observing user behaviour, especially when talking about possibility of negative feedback. Therefore, I adapt existing techniques of univariance significance analysis based on frequency of occurrence of only positive user feedback. This information is used to compute probabilities of interest during learning phase. Separately, delivering of personalized information is realized by exploitation of established user profile. It is the responsibility of the Personal Agent. More precisely, Personal Agent returns list of recommended travel objects (i.e. restaurants) in a response to the user query. Co-existence of two proactive entities in the system: the user (requesting a recommendation) and the Personal Agent (asking for opinion about recommended travel objects) results in the problem of mixed initiative. Since user comfort is crucial, an agent cannot break user-directed scenario by taking control over the dialog. Therefore, I propose adaptation of push-and-pull technique into context of HTTP protocol. I describe also a novel algorithm for stereotyping to address problem of introducing new user to the system (cold start problem). More precisely, the algorithm matches user with one of constructed stereotypes on the base of demographic data. Reasonable stereotypes must be developed for efficiency of stereotyping and thus I point the need of engaging psychologists and sociologists in stereotypes' development, since stereotypes people create are subjective and depend on the population context. During preparing the work I also noticed how lack of data of recommended objects (coming from Chefmoz service, Semantic Web in general) can negatively influence user modeling process; especially when utilizing content-based techniques for information filtering. Finally, I present practical application of the user modeling process in an travel support system, represented according to Prometheus methodology of agent-oriented programming.

# Dodatek E

# Słownik angielsko-polski terminów

| *Termin angielski* | *Polski odpowiednik* |
|---|---|
| action | akcja |
| agent behaviour | zachowanie agenta |
| Agent Communication Language (ACL) | Język Komunikacji Agentów |
| Agent Managing System (AMS) | System Zarządzania Agentem |
| Agent Platform (AP) | platforma agentowa |
| agent role | rola agenta |
| agent-oriented programming | programowanie zorientowane agentowo |
| assertional knowledge | wiedza faktyczna |
| browse | szperać |
| cold start problem | problem zimnego startu |
| collaborative recommending | rekomendowanie kolaboratywne |
| communication act | akt komunikacji |
| Content Collection Subsystem | Podsystem Gromadzenia Informacji |
| Content Delivery Subsystem | Podsystem Dostarczania Informacji |
| Content Management Subsystem | Podsystem Zarządzania Informacją |
| Content Storage | Repozytorium Informacji |
| content-based recommending | rekomendowanie oparte na treści |
| Controller | Kontroler |
| cookie | ciasteczko |
| demographic recommending | rekomendowanie demograficzne |
| Directory Facilitator (DF) | Usługa Katalogowa |
| domain knowledge ontology | ontologia dziedziny wiedzy |
| explicit feedback | reakcja wprost, bezpośrednia informacja zwrotna |
| feasibility precondtition | wstępny warunek wywołania |
| feature-based recommending | rekomendowanie oparte na atrybutach |
| hyperlink | hiperodnośnik |
| implicit feedback | reakcja nie-wprost, pośrednia informacja zwrotna |
| individual probability | prawdopodobieństwo indywidualne |

| | |
|---|---|
| information filtering (IF) | filtrowanie informacji |
| information retrieval (IR) | wyszukiwanie informacji |
| interaction protocol | protokół interkacji |
| interferred probability | prawdpodobieństwo wywnioskowane |
| interval type | typ przedziałowy |
| literal | literał, ciąg znaków |
| memory-based user profile | profil użytkownika oparty na pamięci |
| model-based user profile | profil użytkownika oparty na modelu |
| Model-View-Controller (MVC) | Model-Widok-Kontroler (MWK) |
| multi-agent system (MAS) | system wieloagentowy |
| nominal type | typ nominalny |
| normalized probablity | prawdopodobieństwo znormalizowane |
| object | dopełnienie |
| object-oriented programming (OOP) | programowanie obiektowe |
| ordinal type | typ porządkowy |
| percept | sygnał |
| personal agent | agent osobisty |
| Personal Agent (PA) | Agent Osobisty (AO) |
| personalization | personalizacja |
| predicate | predykat, orzeczenie |
| Profile Managing Agent (PMA) | Agent Zarządzający Profilem (AZP) |
| property | właściwość |
| Proxy Agent (PrA) | Agent Proxy (APr) |
| rational effect | efekt racjonalny |
| recommending system | system rekomendujący |
| relevance feedback | informacja zwrotna, reakcja zwrotna, informacja relewantna |
| resource | zasób (internetowy) |
| Resource Description Framework (RDF) | Szkielet Opisu Zasobów, Ramowy Opis Zasobów, Metoda Opisu Zasobów |
| Restaurant Service Agent (RSA) | Agent Serwisu Restauracyjnego (ASR) |
| second-generation web | sieć drugiego pokolenia |
| selective dissemination of information | selektywne rozpowszechnianie informacji |
| Semantic Web | Semantyczna Sieć WWW, Semantyczny Internet |
| Session Handling Agent (SHA) | Agent Obsługujący Sesję (AOS) |
| session identifier (session ID) | identyfikator sesji |
| software agent | agent programowy |
| sparsity problem | problem rozrzedzenia |
| stakeholder | udziałowie |
| statement | zdanie |
| stereotype | stereotyp |
| stereotyping | stereotypowanie |
| subject | podmiot |
| terminological knowledge | wiedza terminologiczna |
| travel support system | system wspomagania podróży |

| | |
|---|---|
| triple | trójka, relacja trójargumentowa |
| univariant signifance analysis | jednowymiarowa analiza istotności |
| user identifier (user ID) | identyfikator użytkownika |
| user model | model użytkownika |
| user modelling (UM) | modelowanie użytkownika |
| user profile | profil użytkownika |
| User profile adaptation | adaptacja profilu użytkownika |
| user profile exploitation | eksploatacja profilu użytkownika |
| user profile initialization | inicjalizacja profilu użytkownika |
| user profile learning | uczenie profilu użytkownika |
| user profile maintanance | rozwój profilu użytkownika |
| user profile representation | reprezentacja profilu użytkownika |
| Verified Content Providers | Zweryfikowani Dostawcy Informacji |
| View Transforming Agent (VTA) | Agent Transformujący Widok (ATW) |
| push-and-pull (technique) | technika „wypchnij i ściągnij” |
| mixed initiative | współdzielona inicjatywa |
| Agent Communication Channel | Kanał Komunikacyjny Agenta |

# Spis tabel

# Spis rysunków