Projekt WSD
Raport

System wspomagający wybór najlepszego baru

Zespół 2.0
Szymon Bezpalko
Michał Korzeniewski
Michał Piotrak
Paweł Piotrowski
Dawid Zaniewski

Organizacja projektu

Role osób w projekcie

Ze względu na wczesny etap projektu, przypisane role do konkretnych osób są dość ogólne. Przy realizacji kolejnych części projektu, zostaną one opisane w sposób bardziej szczegółowy:

- Michał Korzeniewski - lider zespołu, specjalista ds. testowania
- Michał Piotrak - specjalista ds. architektury systemu, specjalista ds. implementacji
- Paweł Piotrowski - specjalista ds. architektury systemu, specjalista ds. implementacji
- Dawid Zaniewski - specjalista ds. implementacji, specjalista ds. testowania

Repozytorium kodu

Kod naszego systemu będzie można znaleźć w publicznym repozytorium pod podanym adresem: https://github.com/mickor78/WSD2019Z-bar-finder-JADE

Poprawki w dokumentacji

Zgodnie z zaleceniem odnośnie wyraźnego zaznaczania poprawek w dokumentacji dokonanych po ocenieniu danego etapu, postanowiliśmy oznaczać je czerwonym kolorem czcionki.

Część A

Identyfikacja i opis problemu

W ramach realizacji projektu, postanowiliśmy pochylić się nad problemem wyboru odpowiedniego lokalu do spożycia piwa dla danej osoby, biorąc pod uwagę jej indywidualne preferencje w najistotniejszych kwestiach dotyczących tego zagadnienia. Wbrew pozorom, rozważany problem nie jest wcale błahy, ponieważ wiąże się on z identyfikacją osobistych oczekiwań w stosunku do konkretnego baru, określeniem ich ważności dla nas oraz uzyskaniem niezbędnej wiedzy na temat, czy oceniany lokal zaspokaja nasze wcześniej sprecyzowane pragnienia.

W celu bardziej klarownego wytłumaczenia problematyki danej materii, posłużymy się konkretnym przykładem. Dana osoba chce napić się określonego piwa w miejscu znajdującym się niedaleko niej, które gwarantuje jej także możliwość spotkania się ze znajomymi w spokojnej atmosferze. Przypuśćmy, że posiada już jakąś wiedzę o istniejących lokalach w okolicy, jednak nadal może napotkać na dylematy dotyczące np.:

- kwestii dostępności wymarzonego piwa,
- liczby wolnych miejsc,
- hałasu w barze,
- wyboru pomiędzy lokalami, gdzie pierwszy oferuje w sprzedaży dane piwo, ale niestety nie gwarantuje zacisznej atmosfery. Za to drugi lokal nie posiada w swojej ofercie oczekiwanego piwa, ale może zaproponować coś podobnego oraz także pożądany spokój.

Jak widać, ze względu na brak wystarczającej ilości informacji, podmiot może mieć problem z podjęciem właściwej decyzji o wyborze odpowiadającego mu lokalu. Z tego względu postanowiliśmy zaprojektować system spełniający założenia architektury wieloagentowej, który wspomoże go przy tej czynności. System może się okazać pomocny dla wielu użytkowników, ze względu na narastającą popularność tzw. piw rzemieślniczych, z czym oczywiście wiąże się także rozwój lokali, serwujących tego typu napoje.

Rozwiązanie

Ogólny opis rozwiązania

W projektowanym systemie będzie można wyróżnić dwa rodzaje użytkowników - pierwsi z nich to będą potencjalni klienci lokali, drudzy będą odpowiadać konkretnym barom. Zainteresowany pójściem do baru będzie miał możliwość wyszukania w aplikacji mobilnej najlepszego dla niego miejsca do wypicia piwa poprzez określenie swoich preferencji. Ta czynność będzie polegała na wyspecyfikowaniu wartości / opcji oraz także stopnia ważności dla parametrów branych pod uwagę przy ocenie danego lokalu. Lista tych parametrów oraz przyjmowane przez nich wartości zostaną dokładnie przemyślane oraz ustalone, ale na pewno nie zabraknie tam takich czynników jak:

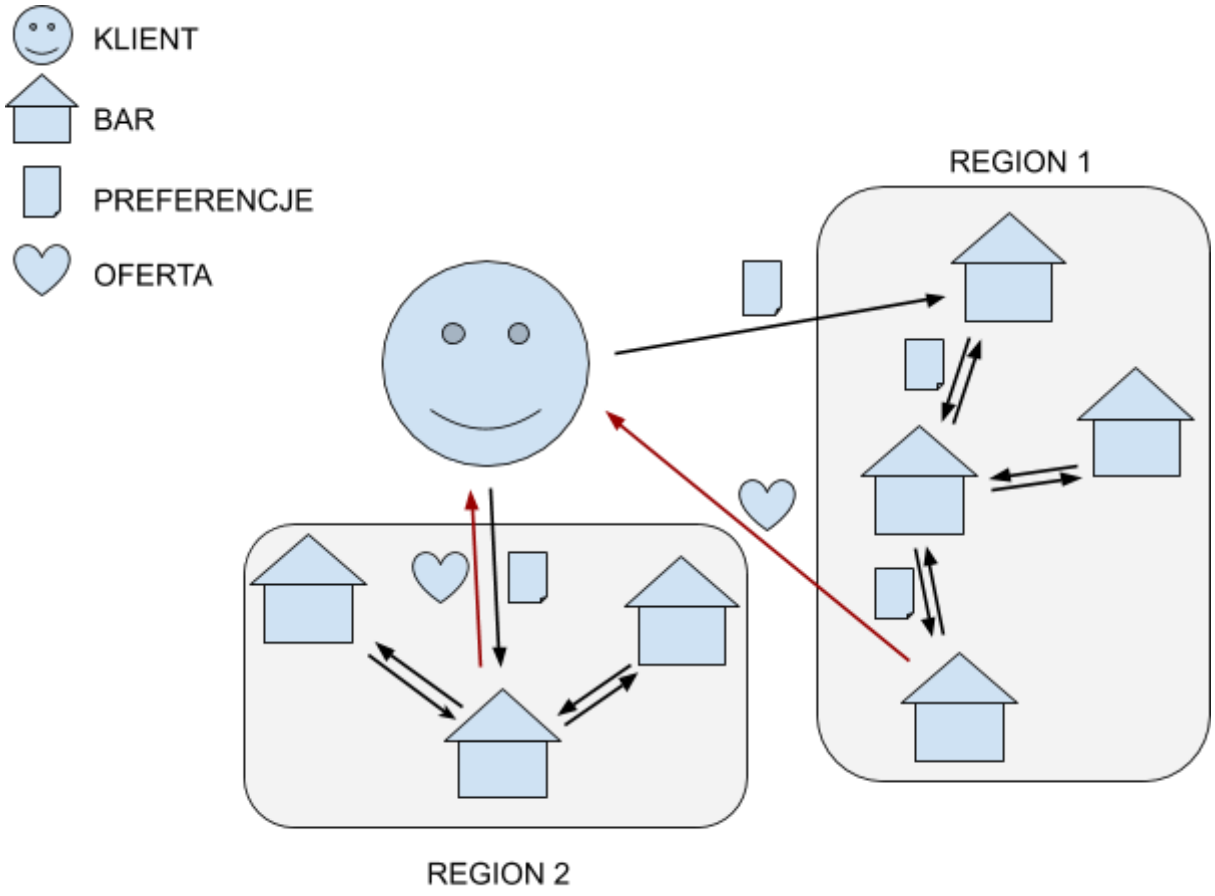
- odległość baru od aktualnej lokalizacji,
- dostępność podanego piwa,
- dostępność podanego stylu piwa,
- cena,
- hałas panujący w lokalu,
- dostępność miejsc.

Stopień ważności parametru w ocenie lokalu będzie można ustalić w skali procentowej, najprawdopodobniej poprzez odpowiednie przeciągniecie suwaka, znajdującego się przy danym czynniku.

Oprogramowanie po stronie baru będzie swoistym źródłem danych o jego ofercie, wyrażonej poprzez wymienione wcześniej parametry oceny lokalu. Aktualizację ich wartości będzie można dokonać w sposób manualny jak np. cenę danego piwa, ale nie wykluczamy tutaj również zastosowania zautomatyzowanych mechanizmów, które będą wyznaczały jakość niektórych czynników. W ten sposób bar może stać się złożonym systemem wieloagentowym, w którym dla przykładu występuje agent decybelomierz, służący za pomiar hałasu panującego w lokalu, czy agent monitoring, dostarczający obrazy z kamer, na podstawie których można wnioskować o liczbie wolnych miejsc. W realizacji tego pomysłu, funkcjonalność wspomnianych wcześniej agentów zostanie znacząco uproszczona i będzie polegała na zwracaniu jakiejś losowej wartości na żądanie pomiaru danego parametru.

Opisane wyżej oprogramowanie klienckie oraz baru należy utożsamiać z dwoma konkretnymi typami agentów, których od tej pory będziemy nazywać po prostu klientami lub barami. W następnym rozdziale zostanie wytłumaczona komunikacja między agentami, która ma na celu wyznaczenie miejsca, spełniającego w najlepszym stopniu wymagania danego klienta.

Opis komunikacji pomiędzy agentami



Powyższy rysunek ma na celu zobrazować przebieg komunikacji pomiędzy agentami w systemie. Po uruchomieniu aplikacji, sprecyzowaniu wymagań oraz określeniu maksymalnej odległości, w promieniu której powinien znajdować się bar, przez klienta zostanie rozesłany komunikat do pewnej ilości barów (minimalizacja prawdopodobieństwa wystąpienia błędu komunikacji). Następnie bary rozpoczną rozsyłanie tego komunikatu do pozostałych agentów tego samego typu oraz przeprowadzą między sobą negocjacje, w celu zidentyfikowania najbardziej dopasowanych reprezentantów. Podczas tych negocjacji będą brane pod uwagę między innymi poszczególne preferencje użytkownika czy opinie o danym barze, aby proces ten nie sprowadzał się do porównania tylko jednego czynnika. Aby zagwarantować realny czas znalezienia idealnego baru, zostanie narzucone pewne ograniczenie czasowe, po przekroczeniu którego wszystkie oferty, które nie przegrały negocjacji z innymi agentami zostaną przesłane do aplikacji klienckiej.

Część B

Projekt naszego systemu wieloagentowego postanowiliśmy zrealizować przy pomocy metodologii GAIA. O takim wyborze zdecydowała głównie przystępność metody, brak narzuconych z góry wymagań dotyczących sposobu realizacji oraz możliwość ułatwienia implementacji systemu w środowisku JADE, którego wykorzystanie zostanie rozważone w kolejnej części raportu.

Faza analizy

Identyfikacja ról

Customer - klient poszukujący baru najlepiej dopasowanego do jego preferencji:

- rola odpowiadająca za kontakt z klientem: **CustomerPreferencesManager**.
- rola czuwająca nad komunikacją z barami: **CustomerHandler**.

Bar - lokal zarejestrowany w systemie:

- rola odpowiedzialna za odbiór preferencji klienta i komunikację z innymi barami: **BarOfferManager**.
- rola zarządzająca parametrami baru: **BarParametersManager**.
- rola reprezentująca bar z aktualnie najlepszą ofertą: **BestOfferHolder**.

Bar controllers - zbiór obiektów przy pomocy których ustalane są wartości danych parametrów baru. Mogą być to specjalne mechanizmy w postaci np. sensorów znajdujących się w barze, czy po prostu osoba, określająca wartość czynnika w sposób manualny:

- rola odpowiedzialna za pomiary hałasu w barze: **LoudnessController**.
- rola czuwająca nad zasobami: **ResourcesController**.
- rola reprezentująca system do określania liczby wolnych miejsc: **SeatsController**.

Identyfikacja aktywności / protokołów dla danych ról

- **CustomerPreferencesManager** - ustala preferencje klienta.
- **CustomerHandler** - wysyła preferencje do poszczególnych barów, w celu wyszukania najlepszego i zbiera dla klienta oferty barów.
- **BarOfferManager** - odbiera listę preferencji klienta, przedstawia swoją ofertę i negocjuję z **BestOfferHolder**.
- **BarParametersManager** - ustala parametry baru na podstawie komunikacji z innymi agentami, znajdującymi się wewnątrz baru.
- **BestOfferHolder** - trzyma najlepszą ofertę (która jeszcze nie przegrała w dotychczas przeprowadzonych negocjacjach przez **BestOfferHolder**), negocjuje z innymi barami, zwraca najlepszą ofertę klientowi.
- **LoudnessController** - nadzoruje poziom hałasu w barze.
- **ResourcesController** - nadzoruje zasoby baru, czyli piwo - zarówno pod względem ilościowym, jak i ceny.
- **SeatsController** - nadzoruje liczbę wolnych miejsc w barze.

Model ról

CustomerPreferencesManager

Aktywności:

- SetPreferences - ustawia nowe wartości preferencji klienta.

Protokoły:

- GivePreferences - dostarcza preferencje klienta.

Role Schema: CustomerPreferencesManager
Description: Zadaniem tej roli jest ustalenie preferencji klienta, które posłużą potem do wybrania najlepszego dla niego baru.
Protocols and activities: SetPreferences, GivePreferences
Permissions: **changes** customerPreferences
Responsibilities: Liveness: CustomerPreferencesManager = ((SetPreferences . GivePreferences) | (GivePreferences))+ Safety: true

CustomerHandler

Aktywności:

Protokoły:

- AwaitCall - oczekuje na żądanie klienta.
- SendPreferences - wysyła preferencje klienta do barów.
- AwaitOffers - oczekuje na oferty przesyłane przez bary.
- InformCustomer - informuje klienta o dostarczonych ofertach.

Role Schema: CustomerHandler
Description: Zadaniem tej roli jest przekazanie preferencji do barów i zebranie w ustalonym czasie najlepszych ofert.
Protocols and activities: AwaitCall, SendPreferences, AwaitOffers, InformCustomer
Permissions: **reads** customerPreferences customerDetails offers
Responsibilities: Liveness: CustomerHandler = (AwaitCall . GenerateOffers)$^{\omega}$ GenerateOffers = (SendPreferences . AwaitOffers . InformCustomer) Safety: infoAvailable(customerPreferences, customerDetails, offers)

BarOfferManager

Aktywności:

- CompareOffers - porównuje zaproponowane oferty pod względem dopasowania do dostarczonych preferencji klienta.

Protokoły:

- GetBarParameters - pobiera parametry określające bar.
- AwaitNegotiations - oczekuje na rozpoczęcie negocjacji.
- GetOpponentOffer - pobiera ofertę rywalizującego z nim baru do przeprowadzenia negocjacji.
- SendCounteroffer - wysyła kontrofertę.
- AdmitDefeat - wysłanie komunikatu o braku możliwości przebicia oferty drugiego baru.

Role Schema: BarOfferManager
Description: Zadaniem tej roli jest negocjacja przejęcia tytułu najlepszej oferty (roli BestOfferHolder). Negocjacje odbywają się na podstawie preferencji klienta, oferty reprezentowanego baru oraz aktualnie najlepszej oferty.
Protocols and activities: GetBarParameters, AwaitNegotiations, GetOpponentOffer, CompareOffers, SendCounteroffer, AdmitDefeat
Permissions: **reads** customerPreferences ownOffer bestOffer
Responsibilities: Liveness: BarOfferManager = (GetBarParamaters . AwaitNegotiations . Negotiate)* Negotiate = (GetOpponentOffer . CompareOffers . (SendCounteroffer | [AdmitDefeat]))+ Safety: infoAvailable(ownOffer)

BarParametersManager

Aktywności:

Protokoły:

- GetLoudnessLevel - pobiera informację o poziomie hałasu w barze.
- GetSeatsNumber - pobiera informację o liczbie wolnych miejsc.
- GetResourcesInfo - pobiera informację o stanie zasobów.
- ProduceOffer - generuje ofertę baru na podstawie jego parametrów.

Role Schema: BarParameteresManager
Description: Zadaniem tej roli jest ustalenie parametrów baru i wystawienie na ich podstawie oferty.
Protocols and activities: GetLoudnessLevel, GetSeatsNumber, GetResourcesInfo, ProduceOffer
Permissions: **reads** loudnessLevel resourcesInfo seatsNumber **generates** offer
Responsibilities: Liveness: BarParametersManager = (GetLoudnessLevel . GetSeatsNumber . GetResourcesInfo . ProduceOffer) Safety: infoAvailable(loudnessLevel, resourcesInfo, seatsNumber)

BestOfferHolder

Aktywności:

- CompareOffers - porównuje zaproponowane oferty pod względem dopasowania do dostarczonych preferencji klienta.

Protokoły:

- ProvideBestOffer - dostarcza dotychczas najlepszą ofertę do klienta.
- SetNewBestOffer - ustanawia daną ofertę za aktualnie najlepszą.
- SendOffer - wysyła swoją ofertę do innego baru w celu rozpoczęcia negocjacji.
- AwaitCounteroffer - oczekuje na kontrofertę.
- AdmitDefeat - wysłanie komunikatu o braku możliwości przebicia oferty drugiego baru.

Role Schema: BestOfferHolder
Description: Zadaniem tej roli jest trzymanie najlepszej aktualnie oferty oraz zwrócenie jej w ustalonym czasie CustomerHandler.
Protocols and activities: ProvideBestOffer, SetNewBestOffer, SendOffer, CompareOffers, AwaitCounteroffer, AdmitDefeat
Permissions: **reads** otherOffer bestOffer # own bar offer customerDetails # for ProvideBestOffer customerPreferences
Responsibilities: Liveness: BestOfferHolder = (Negotiate+ . (SetNewBestOffer | ProvideBestOffer)) Negotiate = ((SendOffer . AwaitCounteroffer . CompareOffers)+ . [AdmitDefeat]) Safety: infoAvailable(bestOffer, customerDetails)

LoudnessController

Aktywności:

Protokoły:

- LoudnessLevelRequest - żąda dokonania pomiaru poziomu hałasu w barze.
- LoudnessLevelResponse - dostarcza informację o poziomie hałasu.

Role Schema: LoudnessController
Description: Zadaniem tej roli jest nadzorowanie poziomu hałasu w lokalu.
Protocols and activities: LoudnessLevelRequest, LoudnessLevelResponse
Permissions: **generates** loudnessLevel
Responsibilities: Liveness: LoudnessController = (LoudnessLevelRequest . LoudnessLevelResponse) Safety: true

ResourcesController

Aktywności:

Protokoły:

- ResourcesInfoRequest - żąda informacji o stanie zasobów.
- ResourcesInfoResponse - dostarcza informację o stanie zasobów.

Role Schema: ResourcesController
Description: Zadaniem tej roli jest nadzorowanie ilości poszczególnych piw dostępnych w lokalu oraz ich cen.
Protocols and activities: ResourcesInfoRequest, ResourcesInfoResponse
Permissions: **generates** resourcesInfo
Responsibilities: Liveness: ResourcesController = (ResourcesInfoRequest . ResourcesInfoResponse) Safety: true

SeatsController

Aktywności:

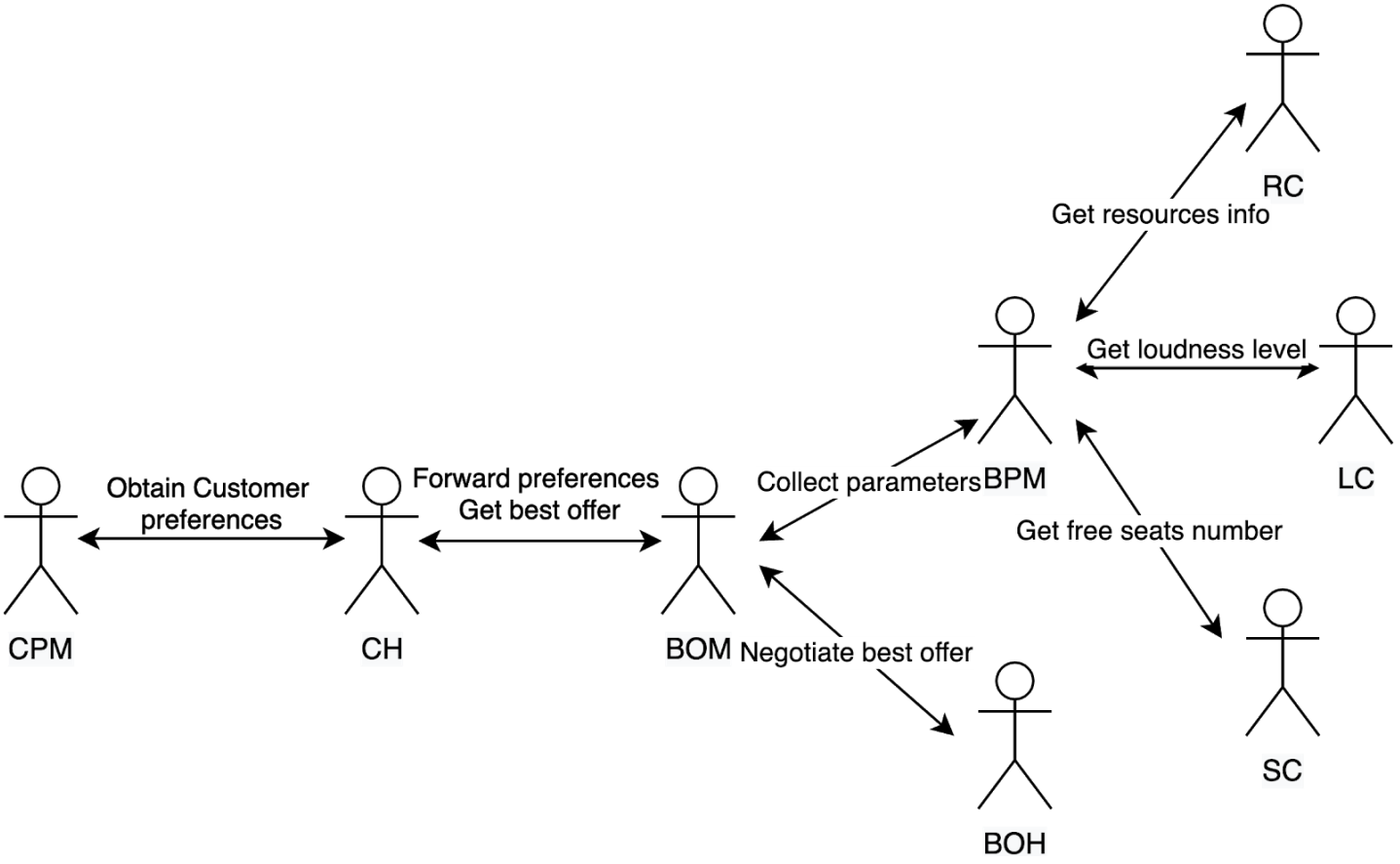
Protokoły:

- SeatsNumberRequest - żąda informacji o liczbie wolnych miejsc w barze.
- SeatsNumberResponse - dostarcza informację o liczbie wolnych miejsc.

Role Schema: SeatsController
Description: Zadaniem tej roli jest nadzorowanie liczby miejsc w lokalu.
Protocols and activities: SeatsNumberRequest, SeatsNumberResponse
Permissions: **generates** seatsNumber
Responsibilities: Liveness: SeatsController = (SeatsNumberRequest . SeatsNumberResponse) Safety: true

Model interakcji

Skróty nazw ról użyte do modelu interakcji:
CPM - CustomerPreferencesManager
CH - CustomerHandler
BOH - BestOfferHolder
BOM - BarOfferManager
BPM - BarParametersManager
SC - SeatsController
LC - LoudnessController
RC - ResourcesController



Diagramy protokołów:

GivePreferences		
CPM	CH	
Przekazanie ustalonych preferencji klienta.		*customerPreferences*

Uwaga do protokołu SendPreferences:
Po tym, jak BOM otrzymuje preferencje klienta od CH, zostaje mu przyznana rola BOH.

SendPreferences		
CH	BOM	
Przekazanie preferencji oraz danych klienta.		*customerPreferences, customerDetails*

SendOffer		
BOH	BOM	
Przekazanie aktualnie najlepszej oferty do kolejnego baru, wraz z preferencjami klienta.		*bestOffer, customerPreferences*

SendCounteroffer		
BOM	BOH	*bestOffer, ownOffer, customerPreferences*
Porównanie ofert i w przypadku posiadania lepszej oferty, wysłanie jej do konkuretna.		*ownOffer*

AdmitDefeat		
BOH	BOM	*bestOffer, otherOffer, customerPreferences*
Porównanie ofert i wysłanie komunikatu o braku możliwości przebicia oferty.		

Uwaga do protokołu SetNewBestOffer:
Bar odpowiedzialny za tę operację po jej wykonaniu utraci rolę BOH. Za to zostanie ona przyznana drugiemu baru, który jest odbiorcą tej komunikacji.

SetNewBestOffer		
BOH	BOM	
Ustanawia ofertę rywalizującego baru za najlepszą.		*customerDetails*

ProvideBestOffer		
BOH	CH	*customerDetails*
Przekazanie informacji o najlepszej ofercie.		*bestOffer*

ProduceOffer		
BPM	BOM	*loudnessLevel* *resourcesInfo* *seatsNumber*
Przekazanie wartości parametrów, w przypadku wielu czujników zbierających dane tego samego parametru uśrednienie i ostatecznie utworzenie oferty.		*offer*

SeatsNumberRequest		
BPM	SC	
Prośba o informację o ilości wolnych miejsc		

SeatsNumberResponse		
SC	BPM	
Przekazanie informacji o ilości wolnych miejsc		*seatsNumber*

LoudnessLevelRequest	
BPM	LC
Prośba o informację o poziom natężenia dźwięku	

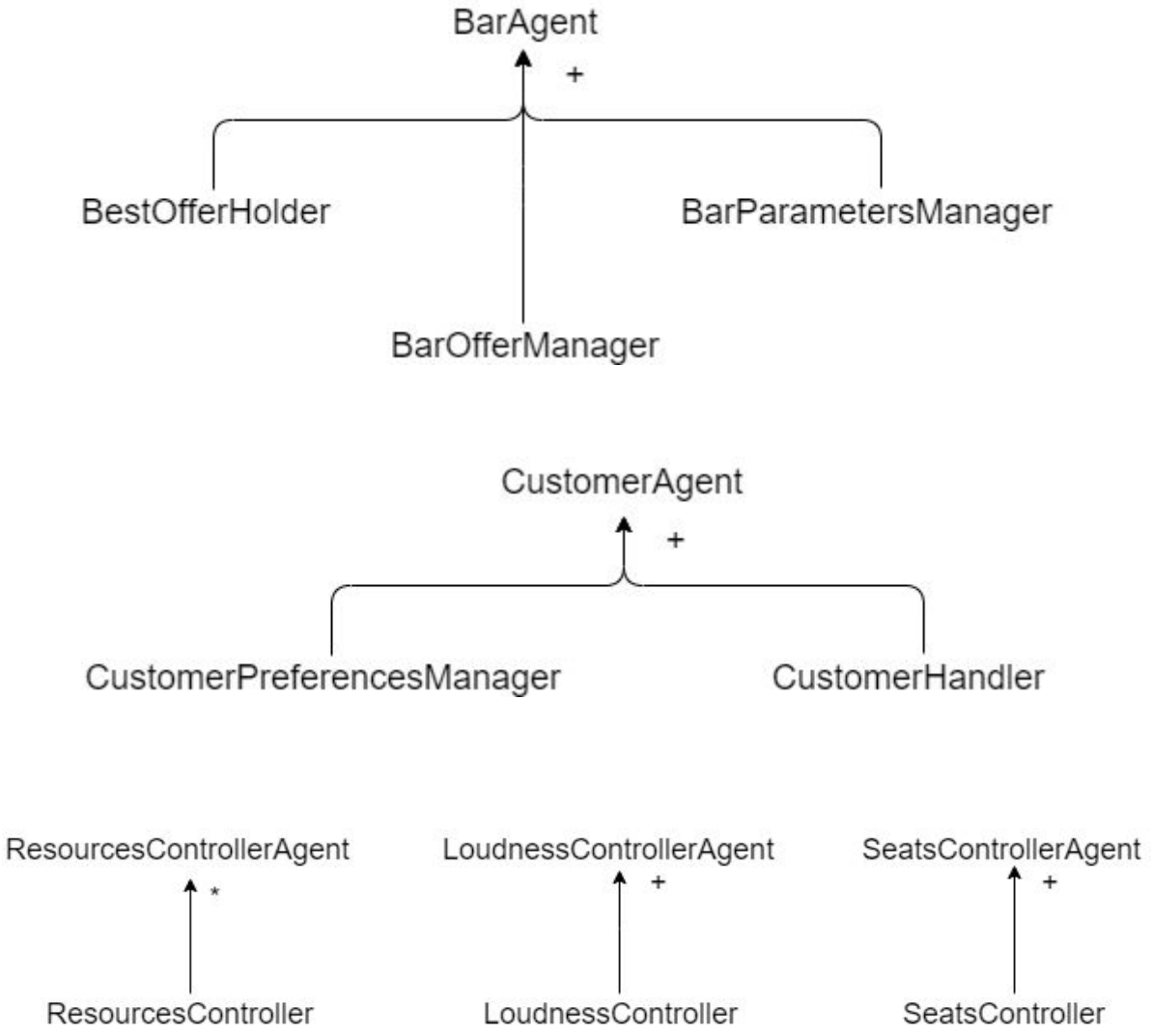
LoudnessLevelResponse	
LC	BPM
Przekazanie informacji o poziomie natężenia dźwięku	*loudnessLevel*

ResourcesInfoRequest	
BPM	RC
Prośba o informację o zasobach baru	

ResourcesInfoResponse	
RC	BPM
Przekazanie informacji o zasobach baru	*resourcesInfo*

Faza projektowania

Model agentów



Wydzielone zostały następujące agenty:

BarAgent - agent zbierający informacje opisujące parametry baru i negocjujący z innymi agentami, w celu wybrania baru najlepiej dopasowanego do preferencji klienta. W przypadku zwycięstwa w negocjacjach przekazuje zwycięską ofertę. Przewidziano wielu agentów, po jednym na bar / lokal.

CustomerAgent - agent pobierający wskazanych przez użytkownika wartości i priorytetów parametrów, przekazujące zapytania do barów, oraz wskazujący użytkownikowi wybrany bar. Przewidzianych zostało wielu agentów, po jednym na użytkownika końcowego.

ResourcesControllerAgent - agent pobierający dane o stanie zasobów w barze (ilość i cena poszczególnych piw) i przekazujący pomiary do BarAgent. Przewidziano wielu

agentów, po jednym na każdy czujnik, czujnik nie jest jednak obligatoryjny (informację, czy cenę danego piwa może wprowadzić ręcznie barman).

LoudnessControllerAgent - agent pobierający dane z decybelomierza i przekazujący pomiary do BarAgent. Przewidziano wielu agentów, po jednym na każdy czujnik.

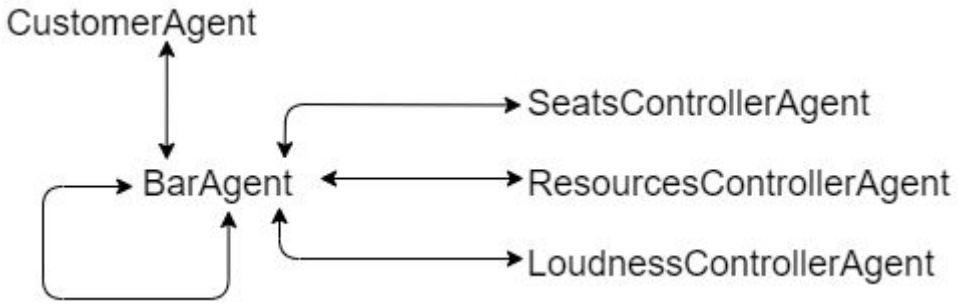
SeatsControllerAgent - agent pobierający dane z odpowiedniego systemu do analizy ilości wolnych miejsc i przekazujący pomiary do BarAgent. Przewidziano wielu agentów, po jednym na każdy element systemu.

Model usług

Usługa	Wejścia	Wyjścia	Warunki Wstępne	Warunki Końcowe
Obtain Customer Preferences	*customerDetails*	*customerPreferences*	customerPreferences !=NULL customerDetails !=NULL	true
Get Best Offers	*customerDetails customerPreferences*	*bestOffer*	customerPreferences !=NULL customerDetails, offers !=NULL	true
Negotiate Best Offer	*customerPreferences ownOffer bestOffer*	*bestOffer*	customerPreferences !=NULL	true
Monitor Parameters and Produce Offer	*loudnessLevel resourcesInfo seatsNumber*	*offer*	loudnessLevel !=NULL, resourcesInfo !=NULL, seatsNumber !=NULL	true
Monitor Resources		*resourcesInfo*	*true*	*true*
Monitor Loudnesss		*loudnessLevel*	*true*	*true*
Monitor Free Seats		*seatsNumber*	*true*	*true*

Model znajomości

Relacje pomiędzy agentami są przedstawione na poniższym diagramie:



Część C

Implementacja

Wykorzystane technologie

Implementacja systemu została wykonana w technologii Java z użyciem frameworka JADE w wersji 4.5.0. Dane konfiguracyjne systemu (lista barów i ich parametrów, preferencje klientów, definicje regionów) są do niego dostarczane w postaci plików w formacie JSON. W celu ułatwienia budowy końcowego produktu wykorzystane zostało narzędzie Maven, dla którego konfiguracja, zawierająca między innymi zależności oraz opisująca dokładnie szczegóły projektu zawarta jest w pliku *pom.xml*.

Głównymi powodami wyboru JADE były:

- implementacja w powszechnie znanym języku programowania Java,
- dobre wsparcie i dokumentacja, ze względu na wysoką popularność tego frameworka,
- zgodność ze standardami FIPA.

Opis implementacji agentów

Zgodnie z przygotowanym projektem, w systemie zostały wydzielone następujące agenty:

- **BarAgent** - reprezentuje instytucję baru. Bierze udział w negocjacjach z innymi barami, które polegają na prezentacji swojej oferty w kontekście dostarczonych preferencji klienta. Jest także odpowiedzialny za komunikację z wewnętrznymi agentami baru, np. z ResourcesControllerAgent, w celu aktualizacji swojego stanu, co jest równoważne z modyfikacją oferty baru.
- **CustomerAgent** - jest utożsamiany z potencjalnym klientem baru, który specyfikuje swoje preferencje odnośnie miejsca, w którym chciałby spożyć mniej czy bardziej określone piwo. Przesyła ustalone preferencje do barów znajdujących się w najbliższych klientowi okolicach, a następnie oczekuje na oferty.
- **ResourcesControllerAgent** - dostarcza informacje o stanie zasobów baru, czyli o dostępnych piwach w barze oraz ich ilościach.
- **LoudnessControllerAgent** - sprawdza poziom hałasu panującego w lokalu.
- **SeatsControllerAgent** - aktualizuje liczbę wolnych miejsc w barze.

BarAgent realizuje dwa główne zachowania, które zostały wcześniej zaprojektowane, czyli **BarOfferManager** i **BestOfferHolder**. Na zachowanie BarOfferManager składa się kilka czynności, które dotyczą m.in. oczekiwania na preferencje danego klienta (klasa **AwaitPreferences**), nasłuchiwania propozycji negocjacji ze strony innego baru (klasa **AwaitNegotiations**) czy już samych negocjacji (klasa **Negotiations**). Przy pomocy zachowania BarOfferManager odbywa się również komunikacja z wewnętrznymi agentami baru, która polega na cyklicznym wysyłaniu do nich żądania dotyczącego pobrania informacji

o aktualnej wartości danego parametru baru (klasy **GetBarParameters** i **GetControllerAgentResponse**).

Agent posiadający zachowanie BestOfferHolder jest utożsamiany z barem dostarczającym taką ofertę dla otrzymanych preferencji klienta, która w bezpośrednich porównaniach z ofertami innych barów okazywała się dotychczas lepsza. Inaczej mówiąc, agent z zachowaniem BestOfferHolder reprezentuje bar, który nie przegrał negocjacji dotyczących preferencji konkretnego klienta z innym barem w swoim regionie. Poprzez termin region określamy z góry przyjęty obszar, do którego należy dany bar. Warto wspomnieć, że bary mogą jedynie negocjować z konkurentami należącymi do tego samego regionu. Dzięki takiemu podejściu, tylko jeden bar w regionie w danym momencie może posiadać zachowanie BestOfferHolder. Jedynie taki lokal może inicjować negocjacje z innymi barami (klasa **StartNegotations**), które poprzedza wyszukanie potencjalnych rywali w regionie (klasa **SearchCompetitors**). Więcej o samym sposobie prowadzenia negocjacji zostanie wspomniane w następnym rozdziale, lecz za ich realizację po stronie zachowania BestOfferHolder odpowiadają klasy **NegotiationsGetOffer**, **NegotiationsProcessOffers** i **NegotiationsSendResponse**. W przypadku, gdy dany bar nie znalazł w swoim regionie lepszych ofert, przesyła on swoją własną do danego klienta (klasa **ProvideBestOffer**).

Odnośnie zadań wykonywanych przez agenta klienta (**CustomerAgent**), należy wspomnieć o wysyłaniu zdefiniowanych preferencji do barów, znajdujących się w najbliższych mu regionach (klasa **FindBar**) oraz nasłuchiwaniu przesyłanych mu ofert (**AwaitOffers**). W tym miejscu, należy wspomnieć, że w momencie otrzymania preferencji klienta przez BarAgent, dodawane mu jest wcześniej opisywane zachowanie **BestOfferHolder**.

Klasy BarAgent i CustomerAgent rozszerzają funkcjonalność klasy **BarFinderAgent**, która jest podstawowym bytem, zapewniającym rejestrację danego agenta w systemie. Jest to realizowane przy pomocy podstawowego agenta dla środowiska JADE - **DF agent**. Dzięki jego zastosowaniu, agent klienta może wysyłać swoje preferencje do pobliskich lokali oraz bary mogą wyszukiwać kolejnych konkurentów w regionie.

Do grupy agentów wewnętrznych baru należą: **LoudnessControllerAgent**, **ResourcesControllerAgent** i **SeatsControllerAgent**. Ich działanie opiera się na oczekiwaniu na żądanie dotyczące udostępnienia informacji o aktualnej wartości danego parametru, którego pomiarem się zajmują, oraz jej wysłaniem. Implementacja tych agentów ma jedynie symulować funkcjonalność złożonych systemów odpowiedzialnych za pomiar np. hałasu czy ilość danego piwa w beczce. Dla przykładu, ResourcesControllerAgent w sposób losowy może modyfikować ilość określonego piwa, przy czym faworyzowane jest zmniejszanie tejże ilości o wartość imitującą zakup 1 piwa 0,5l przez klienta. Co jakiś czas powinna zostać zasymulowana również dostawa piwa, w przypadku, gdy jest już go po prostu mało albo w ogóle. W podobnym charakterze zostały zaimplementowane inne wewnętrzne agenty baru.

Opis implementacji komunikacji między agentami

Ze względu na złożoność problemu, na początku tego rozdziału skupimy się na opisie protokołu negocjacji pomiędzy rywalizującymi barami. Przy jego projektowaniu staraliśmy się brać inspirację z wzorców zalecanych przez FIPA. Dany protokół zostanie przedstawiony jako lista następujących po sobie kroków (BOH - BestOfferHolder, BOM - BarOfferManager):

1. BarAgent z zachowaniem BOH wysyła komunikat z performatywą CFP do wyszukanych wcześniej potencjalnych rywali. W przesyłanej wiadomości zawiera preferencje klienta.
2. BarAgent z zachowaniem BOM otrzymuje komunikat z performatywą CFP. Odbiera preferencje klienta, dla których oblicza wartość swojej oferty (w przypadku, gdy w swojej ofercie ma jakieś piwa - inaczej odrzuca propozycję negocjacji, wysyłając komunikat z performatywą REFUSE). Po dokonaniu obliczeń, wysyła dany wynik do wyzywającego z performatywą PROPOSE i oczekuję na odpowiedź.
3. BarAgent z zachowaniem BOH odbiera komunikat od danego baru.
    a. Jeśli przeciwnik odrzucił propozycję negocjacji, agent dodaje go do listy pokonanych i oczekuje dalej na oferty pozostałych barów, które wyzwał.
    b. Jednak, jeśli konkurent przyjął wyzwanie i przesłał wartość swojej oferty, to inicjator negocjacji zapamiętuje jego wynik i oczekuje na oferty innych barów.
4. Gdy BarAgent z zachowaniem BOH doczekał się odpowiedzi od wszystkich konkurentów, to wybiera z nich ofertę najbardziej wartościową i porównuje ją ze swoim wynikiem.
    a. Jeśli przegrał, przekazuje zwycięzcy swoją rolę właściciela dotychczas najlepszej oferty, wysyłając komunikat z performatywą ACCEPT_PROPOSAL, wraz z listą barów, których oferty okazały się gorsze od wyniku wygranego. Innym za to rozsyła wiadomość z performatywą REJECT_PROPOSAL. Następnie oczekuje na potwierdzenie od wszystkich konkurentów.
    b. Jeśli wygrał, to wszystkim agentom, z którymi prowadził negocjacje rozsyła komunikaty z performatywą REJECT_PROPOSAL, sam zaś zwraca klientowi własną ofertę jako najlepszą w danym regionie.
5. BarAgent z zachowaniem BOM dostaje odpowiedź na swoją ofertę.
    a. W przypadku wiadomości z performatywą ACCEPT_PROPOSAL, do danego agenta dodawane jest zachowanie BOH, wraz z przypisaniem dostarczonej listy przegranych barów. Jest to realizowane w celu uniknięcia zbędnych negocjacji. Do wspomnianej listy jest oczywiście dodawany również ostatni właściciel najlepszej oferty. Na koniec, agent odsyła wiadomość z performatywą INFORM o zawartości “DONE”, w celu potwierdzenia zakończenia negocjacji.
    b. Jeśli otrzymana wiadomość posiada performatywę REJECT_PROPOSAL, agent wysyła potwierdzenie zakończenia negocjacji w postaci komunikatu z performatywą INFORM o zawartości “DONE”.
6. BarAgent z zachowaniem BOH dostaje potwierdzenie o zakończeniu negocjacji ze strony rywala i także je kończy.

Warto w tym miejscu dodać, że do przesyłania preferencji klienta został wykorzystany mechanizm ontologii, udostępniony w środowisku JADE. Klasa **PreferencesOntology** opisuje strukturę preferencji, która jest zrozumiała dla agentów posługujących się daną ontologią.

Odnośnie komunikacji agent baru - agent klienta przebiega ona w następujący sposób (BOH - BestOfferHolder, BOM - BarOfferManager):

1. CustomerAgent rozsyła zdefiniowane preferencje do barów znajdujących się w najbliższych regionach od lokalizacji klienta. Wysyłane komunikaty są oznaczone performatywą REQUEST.
2. BarAgent z zachowaniem BOM otrzymuje komunikat z performatywą REQUEST. Pobiera preferencje klienta i dodaje zachowanie BOH, co jest równoważne z rozpoczęciem procesu negocjacji z innymi barami.
3. Gdy dany BarAgent okaże się najlepszy w kontekście określonych preferencji, wysyła komunikat ze swoją ofertą do klienta. Wiadomość zawiera performatywę INFORM.
4. CustomerAgent odbiera komunikat z proponowaną ofertą.

Ostatnim rodzajem interakcji zachodzącej w systemie, jest wymiana wiadomości pomiędzy agentem baru a agentem wewnętrznym baru. W tym przypadku, schemat postępowania wygląda tak:

1. BarAgent wysyła zapytanie o aktualną wartość danego parametru do wyspecjalizowanego agenta wewnętrznego baru, który zajmuje się pomiarem wartości konkretnego atrybutu. Dany komunikat jest oznaczony performatywą QUERY_REF.
2. Agent wewnętrzny baru otrzymuje żądanie z performatywą QUERY_REF. Dokonuje odpowiedniego pomiaru.
    a. W przypadku sukcesu operacji, wspomniany agent przekazuje uzyskany rezultat w postaci wiadomości z performatywą INFORM.
    b. Jeśli wystąpił błąd, wysyłany jest komunikat z performatywą FAILURE.
3. BarAgent dostaje odpowiedź na swoje zapytanie. Jeśli pomiar przebiegł prawidłowo, następuje aktualizacja stanu baru, która polega na przypisaniu nowej wartości określonego parametru.

Opis funkcji oceny oferty baru

Funkcja oceny oferty baru (realizowana przez klasę **ParametersScore**) zgodnie z założeniami zależy od 6 parametrów baru (odległość baru od aktualnej lokalizacji klienta, dostępność podanego piwa, dostępność podanego stylu piwa, cena, hałas panujący w lokalu, dostępność miejsc). Każdy z parametrów jest przeliczany przez jego wagę ustaloną przez klienta.

Lokalizacja jest wartościowana zgodnie z zasadą:

Odległość lokalu od klienta	**Wynik**

<= 1000 m	Funkcja liniowa, zakres wartości <1;0.5>
<= 5000 m	0.5
> 5000 m	0

Podobnie obecność poszukiwanego piwa w lokalu jest punktowana zależnie od jego ilości (ilość odniesienia została ustalona na 10l):

Ilość piwa	**Wynik**
< 30% ilości odniesienia = 3	0
od 30% do 100% ilości odniesienia <3;10>	Funkcja liniowa, zakres wartości <0;5>
> 100% ilości odniesienia = 10	5

W przypadku gdy konkretne piwo nie zostało przez klienta określone lub gdy bar nie posiada poszukiwanego piwa, punktowany jest styl. Punktowanie za styl jest analogiczne do tego za piwo. Istotna różnicą jest waga stanowiąca 20% tej od piwa. Za każde piwo danego stylu przyznawane są punkty, ale maksymalna suma to 3. Punktowane jest również posiadanie w ofercie piw o stylach podobnych do poszukiwanego. Określenie podobieństwa jest oparte na danych z: https://www.wiki.piwo.org/Zestawienie_styl%C3%B3w_piw_(tabela). Punkty za podobne style są przyznawane analogicznie do punktów za styl, z wagą stanowiącą 30%. Maksymalna liczba punktów to 1.

Punkty za cenę są uzależnione od ilości piw z ceną poniżej pożądanej i są przyznawane następująco:

Liczba piw wystarczająco tanich	**Wynik**
> 3	1
<1;3>	0.8
1	0.5
0	0

Punkty za hałas są są przyznawane zgodnie z następującą strategią:

Oczekiwania klienta	**Poziom hałasu w barze**			
	QUIET	**MEDIUM**	**NOISE**	**UNKNOWN**
QUIET	1	0.5	0	0

MEDIUM	0.5	1	0.5	0.5
NOISE	0	0.5	1	0

Punkty z wolne miejsca są przyznawane następująco:

Aktualna liczba wolnych miejsc	Wynik
> 200% pożądanej liczby miejsc	1
> 100% pożądanej liczby miejsc	0.5
< 100% pożądanej liczby miejsc	0

Punkty za każdy z parametrów przeliczane są przez wagi określone przez klienta. Uzyskany wynik jest wykorzystywany w negocjacjach pomiędzy agentami.

Przykładowe logi z działania systemu i ich opisy

Przesłanie preferencji klienta, negocjacje między barami (wraz z przekazaniem roli BestOfferHolder) oraz zwrócenie najlepszej oferty.

Uruchomione agenty:

- CustomerAgent - 1
- BarAgent - 2

```
customer_Jan_Krafciarz (customer) - receiver found: bar_Jabberwocky

customer_Jan_Krafciarz (customer) - sends preferences to bars.

bar_Jabberwocky (BOM) - receives from customer_Jan_Krafciarz:
Preferences parameter: name - localization, value - 52.229876;21.015919, importance
- 0.80
Preferences parameter: name - beer_style, value - AMERICAN_PALE_ALE, importance -
0.40

bar_Jabberwocky (BOH) - score for customer customer_Jan_Krafciarz: 0.800000.

bar_Jabberwocky (BOH) - sends CFP messages to 1 competitors.

bar_Kufle_i_kapsle (BOM, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) -
receives from bar_Jabberwocky:
Preferences parameter: name - localization, value - 52.229876;21.015919, importance
- 0.80
Preferences parameter: name - beer_style, value - AMERICAN_PALE_ALE, importance -
0.40

bar_Kufle_i_kapsle (BOM, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) -
score for preferences: 1.142114.

bar_Kufle_i_kapsle (BOM, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) -
sends proposal to bar_Jabberwocky.

bar_Jabberwocky (BOH, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) -
receives proposal (score: 1.142114) from bar_Kufle_i_kapsle.
```

```
bar_Jabberwocky (BOH, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) - is
defeated by bar_Kufle_i_kapsle.

bar_Jabberwocky (BOH, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) - sends
proposal acceptation to bar_Kufle_i_kapsle.

bar_Kufle_i_kapsle (BOM, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) -
receives proposal acceptation from bar_Jabberwocky.

bar_Kufle_i_kapsle (BOM, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) -
sends negotiations end confirmation to bar_Jabberwocky.

bar_Jabberwocky (BOH, conversationId: 1e69963e-663f-4c03-bf4e-5f66ff4e63d5) -
receives negotiations end confirmation  from bar_Kufle_i_kapsle.

bar_Kufle_i_kapsle (BOH) - sends offer to customer customer_Jan_Krafciarz.

customer_Jan_Krafciarz (customer) - receives offer: bar_Kufle_i_kapsle - 1.142114.
```

Aktualizacja parametrów baru

Uruchomione agenty:

- BarAgent - 1
- LoudnessControllerAgent - 1
- SeatsControllerAgent - 1
- ResourcesControllerAgent - 1

```
bar_Jabberwocky (BOM) - sends query to bar_Jabberwocky_seats_controller_agent.

bar_Jabberwocky (BOM) - sends query to bar_Jabberwocky_loudness_controller_agent.

bar_Jabberwocky (BOM) - sends query to bar_Jabberwocky_resources_controller_agent.

bar_Jabberwocky (BOM) - receives from bar_Jabberwocky_loudness_controller_agent:
QUIET

bar_Jabberwocky (BOM) - receives from bar_Jabberwocky_seats_controller_agent: 19

bar_Jabberwocky (BOM) - receives from bar_Jabberwocky_resources_controller_agent:
Beer: Name - Miękkie lądowanie, BreweryName - Nook, Style - GERMAN_PILSNER, Price -
16.00, Quantity - 30.00
Beer: Name - Slow & Trve, BreweryName - Zagovor Brewery, Style - ENGLISH_IPA, Price
- 30.00, Quantity - 60.00
```

Rysunek 1. Agenty widoczne w JADE Remote Agent Management GUI

Wprowadzone zmiany

- Negocjacje zostały przekształcone w przekazywanie między barami całościowych ocen ich ofert, a nie poszczególnych parametrów. W ten sposób uprościliśmy komunikację między rywalizującymi barami oraz postaraliśmy się wzorować na zaleceniach sugerowanych przez FIPA.
- Nie zostało zrealizowane ograniczenie czasowe dla poszukiwania oferty, nakładane od strony klienta. Zmiana została wprowadzona w celu uproszczenia implementacji. Problem jest złożony i nie zdołaliśmy rozwiązać go w ograniczonym czasie.
- Także ograniczenie czasowe dotyczące oczekiwania danego baru z zachowaniem BestOfferHolder na kolejne oferty ze strony konkurentów nie zostało zrealizowane z podobnych pobudek.
- Do określenia oceny baru związanej z ceną piw nie został uwzględniony pożądany styl piwa lub konkretne piwo. Są brane pod uwagę wszystkie piwa spełniające jedynie warunek odnoszący się do ceny piwa. Zmiana została wprowadzona w celu uproszczenia algorytmu.

Część D

Integracja systemu

Podsumowanie przygotowanego rozwiązania

W ramach realizacji projektu stworzyliśmy wieloagentowy system wspomagający decyzję wybrania lokalu do spożycia piwa. Analizując możliwe oczekiwania klientów barów, opracowaliśmy koncepcję systemu, który na podstawie zadanych kryteriów wybiera spośród dostępnych miejsc kilka najbardziej odpowiadających preferencjom klienta. System, posiadając aktualne informacje np. o zatłoczeniu barów, dostępności określonych piw, umożliwia klientom bardziej świadomy wybór lokalu.

Kwestie techniczne dotyczące opracowanego rozwiązania zostały w dość szczegółowy sposób opisane w poprzedniej części raportu.

Zidentyfikowane braki w systemie

Ze względu na ograniczony czas realizacji projektu, w przygotowanym rozwiązaniu znajdują się pewne luki, które powinny zostać docelowo uzupełnione. Poniżej została umieszczona lista braków / defektów, które udało się nam ustalić samodzielnie albo poprzez analizę dostarczanych recenzji innych studentów:

1. Brak ograniczeń czasowych na wykonanie poszczególnych etapów procesu wyszukiwania najlepszych barów. Precyzując, tego typu obostrzenie powinno zostać przyszykowane po stronie klienta, jak i także dla baru, który oczekuje na kontroferty swoich potencjalnych oponentów.
2. Uproszczona realizacja funkcjonalności oceny dostarczonej oferty. W głównej mierze chodzi tutaj o tendencję do nie rozważania filtrów nałożonych na inne atrybuty w trakcie dokonywania oceny danego parametru. Prowadzi to do m.in. takich sytuacji, że w ocenie oferty baru pod względem cen piw bierzemy pod uwagę wszystkie dostępne piwa, a nie tylko te, które spełniają restrykcję dotyczącą stylu czy nazwy.
3. Lokalizowanie barów i klientów poprzez bezpośrednie podawanie współrzędnych. Powinno zostać to zastąpione poprzez wykorzystanie mechanizmu geokodowania.
4. Brak bardziej złożonej obsługi sytuacji wyjątkowych w niektórych miejscach systemu (np. reakcja na otrzymanie komunikatu z performatywą NOT_UNDERSTOOD).
5. Nieprzygotowanie graficznego interfejsu użytkownika, w szczególności dla klienta.
6. Brak przemyśleń dotyczących integracji systemu z konkretną bazą danych.
7. Ewentualne rozważenie modyfikacji działania agentów wewnętrznych barów na bardziej proaktywne. W takim przypadku, określone agenty wychodziłyby same z inicjatywą informowania centralnego agenta baru o zmianach wartości parametrów takich jak poziom hałasu, liczba wolnych miejsc czy dostępność oferowanych piw.

Testy

Testy systemu zostały przeprowadzone dla kilku wariantów danych wejściowych (pliki z danymi testowymi znajdują się w repozytorium). Przy opisach kolejnych testów są podawane nazwy plików, z których były pobierane dane odnoszące się do preferencji klientów oraz ofert barów. Jeśli chodzi o definicje regionów to dla każdego przypadku testowego były one uzyskiwane z pliku *regions.json*.

Test 1

Opis
Badanie ma na celu sprawdzenie działanie systemu w przypadkach, gdy klient w swoich preferencjach ustalił jeden dominujący parametr nad innymi w kwestii ważności.
Wybrane zestawy danych: *test1_bars.json*, *test1_customers.json*.

Klient	Oczekiwany wynik	Komentarz	Rzeczywisty wynik	Komentarz
Jan_Krafciarz1	Cuda na kiju	Wysoki priorytet na odległość powinien doprowadzić do wybrania najbliższego baru.	Cuda na kiju	Zgodnie z oczekiwaniami.
Jan_Krafciarz2	Piwiarnia Warki	Wysoki priorytet na styl piwa powinien doprowadzić do wybrania baru posiadającego w ofercie piwo o podanym stylu.	Piwiarnia Warki	Zgodnie z oczekiwaniami.

Podsumowanie
System poprawnie wyszukał najbardziej pasujące oferty do preferencji wskazanych klientów.

Test 2

Opis
Przygotowany test sprawdza mechanizm odnajdywania podobnych stylów piwa.
Wybrane zestawy danych: *test2_bars.json*, *test2_customers.json*.

Klient	Oczekiwany wynik	Komentarz	Rzeczywisty wynik	Komentarz
Jan_Krafciarz1	Piwiarnia Warki	Wysoki priorytet na styl piwa, żaden z barów nie posiada tego stylu w ofercie, jednak bar Piwiarnia Warki oferuje podobny styl Vienna Lager.	Piwiarnia Warki	Zgodnie z oczekiwaniami.

Podsumowanie
System poprawnie wyszukał najbardziej pasującą ofertę do preferencji wskazanego klienta.

Test 3

Opis
W tym teście mamy do czynienia z dwoma regionami. Klientowi powinny zostać zaoferowane najlepsze bary z każdego regionu.
Wybrane zestawy danych: *test3_bars.json*, *test3_customers.json*.

Klient	Oczekiwany wynik	Komentarz	Rzeczywisty wynik	Komentarz
Jan_Krafciarz1	Jabberwocky	Wysoki priorytet na konkretne piwo, a w regionie Śródmieście jedynie bar Jabberwocky ma to piwo w ofercie.	Jabberwocky	Zgodnie z oczekiwaniami.
	Pub Bemol	Wysoki priorytet na konkretne piwo, a w regionie Bemowo jedynie bar Pub Bemol ma to piwo w ofercie.	Pub Bemol	Zgodnie z oczekiwaniami.

Podsumowanie
System poprawnie wyszukał najbardziej pasujące oferty do preferencji wskazanego klienta.

Test 4

Opis
Dany test ma na celu sprawdzić, czy zgodnie z przyjętym założeniem, bar z małą ilością poszukiwanego przez klienta piwa jest gorzej oceniany.
Wybrane zestawy danych: *test4_bars.json*, *test4_customers.json*.

Klient	Oczekiwany wynik	Komentarz	Rzeczywisty wynik	Komentarz
Jan_Krafciarz1	Cuda na kiju	Wysoki priorytet na konkretne piwo - jest tylko jeden bar, który ma to piwo w ofercie, jednak ma go za mało i nie jest nawet brane pod uwagę. Wygra najbliższy bar - drugi priorytet w rankingu ważności to odległość do baru.	Cuda na kiju	Zgodnie z oczekiwaniami.

Podsumowanie
System poprawnie wyszukał najbardziej pasującą ofertę do preferencji wskazanego klienta. Ignorowanie piwa o pomijalnej ilości działa poprawnie.

Test 5

Opis
Ten przypadek jest analogiczny do poprzedniego. Różnią się tylko ilością pożądanego przez klienta piwa, która nie jest już pomijalna.
Wybrane zestawy danych: *test5_bars.json*, *test5_customers.json*.

Klient	Oczekiwany wynik	Komentarz	Rzeczywisty wynik	Komentarz
Jan_Krafciarz1	Jabberwocky	Wysoki priorytet na konkretne piwo, tylko bar Jabberwocky ma go w ofercie.	Jabberwocky	Zgodnie z oczekiwaniami.

Podsumowanie
System poprawnie wyszukał najbardziej pasującą ofertę do preferencji wskazanego klienta.

Test 6

Opis
Test 6 jest uproszczoną symulacją działania systemu w “naturalnych” warunkach. Utworzono większy niż w poprzednich przypadkach zestaw danych wejściowych (pięć klientów, dwanaście barów w dwóch regionach). Część parametrów jest generowanych losowo - symulacja zmienności parametru (hałas, ilość wolnych miejsc). Test ma sprawdzić działanie systemu przy większej ilości klientów oraz lokali, ale dla specjalnie spreparowanych danych, dla których można przewidzieć wynik programu.
Wybrane zestawy danych: *test6_bars.json*, *test6_customers.json*.

Klient	Oczekiwany wynik	Komentarz	Rzeczywisty wynik	Komentarz
Jan_Krafciarz	Kufle_i_ kapsle	Brak punktów za odległość, bo powyżej wartości granicznej, wygra bar z konkretnym piwem w ofercie.	Kufle_i_kapsle	Zgodnie z oczekiwaniami.
	Beerhub	Brak punktów za odległość, bo powyżej wartości granicznej, wygra bar z konkretnym piwem w ofercie.	Beerhub / Piwodawca	Kilka testów, wynikiem jest jeden z dwóch barów oferujących piwo Bard.
Tomasz_ Chmielik	Piwiarnia Warki	Ma najwięcej tanich piw, a klient ma duży priorytet na cenę.	Piwiarnia Warki	Zgodnie z oczekiwaniami.
	Beerhub	Ma najwięcej tanich piw, a klient ma duży priorytet na cenę.	Beerhub	Zgodnie z oczekiwaniami.
Grzegorz_ Chmielik	Jabberwocky	Ma w ofercie piwo w podobnym stylu do pożądanego i jedno piwo w oczekiwanej cenie.	Jabberwocky	Zgodnie z oczekiwaniami
	NaWyki Piwne	Wysoka ocena dzięki dobrej lokalizacji, dodatkowo punkty	NaWyki Piwne	Zgodnie z oczekiwaniami.

		za ceny piw.		
Grzegorz_ Chmielik2	Jabberwocky	Wysoka ocena za piwo Miękkie lądowanie.	Jabberwocky	Zgodnie z oczekiwaniami.
	Żaden	Żaden bar na Bemowie nie wydaje się być pożądany. Preferowane piwo i styl przez klienta nie faworyzuje żadnego z barów, a dodatkowo znajdują się one za daleko, żeby dostały punkty za lokalizację.	Beerhub / NaWyki Piwne	Wynik losowy (score = 0).
Grzegorz_ Chmielik3	Trudno przewidywać wynik dla tego klienta, gdyż wysoko wyceniane są parametry zmienne (głośność i liczba wolnych miejsc). Konkretne piwo nie jest oferowane w żadnym barze, zaś styl American Pale Ale jest oferowany w większości lokali.		Piwiarnia Warki / Kufle_ i_kapsle	Wynik mocno zależny od zmiennych parametrów.
			Pub Bemol / Beerhub	Wynik mocno zależny od zmiennych parametrów.

Podsumowanie
System poprawnie wyszukiwał najbardziej pasujące oferty do preferencji wskazanych klientów. Podczas wielokrotnego wykonywania tego przypadku testowego, nigdy nie doszło do żadnej sytuacji wyjątkowej, powodującej awarię systemu.

Case studies

Opis wykorzystanych danych

Działanie systemu będzie poddane analizie dla przypadku jednego klienta oraz zbioru barów znajdujących się w dwóch regionach. Preferencje klienta podano poniżej:

Jan Krafciarz		
Parametr	**Wartość**	**Priorytet**
Lokalizacja	52.226718, 21.013022	0.50
Styl piwa	German Pilsner	0.70

Natężenie dźwięku	Głośno	0.30
Cena piwa	17.00	0.80
Wolne miejsca	10	0.20

Nadane wartości priorytetu poszczególnych parametrów wskazują na to, że preferencje klienta skierowane są głównie w stronę stylu piwa oraz jego ceny. Lokalizacja ma dla niego mniejsze znaczenie, zaś natężenie dźwięku i liczba wolnych miejsc zupełnie drugorzędne.

Bary zdefiniowano w dwóch regionach: Śródmieście i Bemowo. Dane zostały tak przygotowane, aby były atrakcyjne dla klienta w kontekście różnych czynników. W tabeli przedstawiono zdefiniowany zestaw lokali:

Nazwa Baru	Lokalizacja	Nazwa Piwa	Browar	Styl piwa	Cena piwa	Ilość piwa	
Jabberwocky	**Region**	Aparat	Jabberwocky	American Pale Ale	14.0	50.5	**Liczba wolnych miejsc**
	Śródmieście	Mosad	Zagovor Brewery	German Pilsner	14.0	50.5	30
	Współrzędne	Miękkie lądowanie	Nook	German Pilsner	16.0	30.5	**Odległość od klienta**
	52.229876, 21.015919	Slow & Trve	Zagovor Brewery	English IPA	30.0	10.0	403,5 m
Kufle i kapsle	**Region**	Ogrodnik	Profesja	Saison	15.0	10.5	**Liczba wolnych miejsc**
	Śródmieście	Bard	Profesja	American Pale Ale	14.0	36.5	50
	Współrzędne						**Odległość od klienta**
	52.229304, 21.014750						312 m
Piwiarnia Warki	**Region**	Ogrodnik	Profesja	Saison	15.0	10.5	**Liczba wolnych miejsc**
	Śródmieście	Bard	Profesja	American Pale Ale	14.0	36.5	80
	Współrzędne	Piwo3	Profesja	German Pilsner	17.0	39.5	**Odległość od klienta**

	52.224591, 21.013905	Piwo4	Browar2	Vienna Lager	12.0	25.4	246 m
Pub Bemol	**Region**	Aparat	Jabberwocky	American Pale Ale	13.0	50.5	**Liczba wolnych miejsc**
	Bemowo	Piwo4	Browar2	Vienna Lager	12.0	17.4	450
	Współrzędne	Slow & Trve	Zagovor Brewery	English IPA	5.0	1.0	**Odległość od klienta**
	52.250731; 20.918327						6970 m
NaWyki Piwne	**Region**	Piwo6	Browar1	German Pilsner	24.0	5.5	**Liczba wolnych miejsc**
	Bemowo	Ogrodnik	Profesja	Saison	15.0	10.5	10
	Współrzędne	Slow & Trve	Zagovor Brewery	English IPA	30.0	1.0	**Odległość od klienta**
	52.231064; 20.902742						7600 m
Piwny Kolektyw	**Region**	Piwo6	Browar1	German Pilsner	10.0	20.5	**Liczba wolnych miejsc**
	Bemowo	Piwo4	Browar2	Vienna Lager	15.0	10.5	7
	Współrzędne	Slow & Trve	Zagovor Brewery	English IPA	30.0	1.0	**Odległość od klienta**
	52.231064, 20.902742						10003 m

Opis przebiegu działania systemu

System uruchomiono z powyżej opisanym zestawem danych. Poskutkowało to utworzeniem agentów związanych z konkretnymi barami (łącznie 24 różnych agentów) oraz agenta utożsamianego z klientem.

Po inicjalizacji, agent *customer_Jan_Krafciarz* przesyła do dwóch losowych barów żądanie znalezienia lokalu, którego oferta pasuje najlepiej do jego preferencji. Oczywiście, wybrane bary znajdują się w innych regionach. W tym przypadku zostały wyłonione bary Piwiarnia z regionu Śródmieście i NaWyki Piwne z regionu Bemowo.

```
================================================
Hello World! My name is customer_Jan_Krafciarz
================================================
================================================
customer_Jan_Krafciarz (customer) - receiver found: bar_Piwiarnia
Warki
================================================
================================================
customer_Jan_Krafciarz (customer) - receiver found: bar_NaWyki
Piwne
================================================
================================================
customer_Jan_Krafciarz (customer) - sends preferences to bars.
================================================
```

Agenty barów otrzymują preferencje klienta i następuje wyliczenie oceny ich ofert. Przyznawana jest im tym samym rola posiadacza najlepszej oferty (zachowanie *BestOfferHolder*).

```
================================================
================================================
bar_Piwiarnia Warki (BOM) - receives from customer_Jan_Krafciarz:
Preferences parameter: name - localization, value -
52.226718;21.013022, importance - 0,50
Preferences parameter: name - beer_style, value - GERMAN_PILSNER,
importance - 0,70
Preferences parameter: name - bar_loudness_level, value - NOISE,
importance - 0,30
Preferences parameter: name - beer_price, value - 17.0, importance
- 0,80
Preferences parameter: name - bar_free_seats, value - 10,
importance - 0,20
================================================
================================================
bar_NaWyki Piwne (BOM) - receives from customer_Jan_Krafciarz:
Preferences parameter: name - localization, value -
52.226718;21.013022, importance - 0,50
Preferences parameter: name - beer_style, value - GERMAN_PILSNER,
importance - 0,70
Preferences parameter: name - bar_loudness_level, value - NOISE,
importance - 0,30
Preferences parameter: name - beer_price, value - 17.0, importance
- 0,80
Preferences parameter: name - bar_free_seats, value - 10,
importance - 0,20
================================================
```

```
================================================
bar_NaWyki Piwne (BOH) - score for customer customer_Jan_Krafciarz:
0,950000.
================================================
================================================
bar_Piwiarnia Warki (BOH) - score for customer
customer_Jan_Krafciarz: 1,775978.
================================================
```

Szczegółowy opis przebiegu procesu negocjacji dla regionu Śródmieście

W regionie Śródmieście bar Piwiarnia Warki znajduje dwóch konkurentów i przesyła im preferencje klienta. Oba bary także obliczają swoją ocenę dla otrzymanych preferencji. Warto w tym miejscu zaznaczyć, że do grona konkurentów danego lokalu nie muszą zaliczać się wszystkie pozostałe bary w określonym regionie. Zbiór oponentów jest generowany w sposób losowy i w tym przypadku zdarzyło się akurat tak, że do opisywanego zbioru zostały dodane wszystkie inne lokale w regionie.

```
================================================
bar_Piwiarnia Warki (BOH) - sends CFP messages to 2 competitors.
================================================
================================================
bar_Kufle_i_kapsle (BOM, conversationId:
dacf43b9-6b83-4aec-bfd6-9bc7da6f54f2) - receives from bar_Piwiarnia
Warki:
Preferences parameter: name - localization, value -
52.226718;21.013022, importance - 0,50
Preferences parameter: name - beer_style, value - GERMAN_PILSNER,
importance - 0,70
Preferences parameter: name - bar_loudness_level, value - NOISE,
importance - 0,30
Preferences parameter: name - beer_price, value - 17.0, importance
- 0,80
Preferences parameter: name - bar_free_seats, value - 10,
importance - 0,20
================================================
================================================
bar_Kufle_i_kapsle (BOM, conversationId:
dacf43b9-6b83-4aec-bfd6-9bc7da6f54f2) - score for preferences:
1,053539.
================================================
================================================
bar_Jabberwocky (BOM, conversationId:
eda6d6aa-01bd-425f-8463-419fe1ad4825) - receives from bar_Piwiarnia
Warki:
```

```
Preferences parameter: name - localization, value -
52.226718;21.013022, importance - 0,50
Preferences parameter: name - beer_style, value - GERMAN_PILSNER,
importance - 0,70
Preferences parameter: name - bar_loudness_level, value - NOISE,
importance - 0,30
Preferences parameter: name - beer_price, value - 17.0, importance
- 0,80
Preferences parameter: name - bar_free_seats, value - 10,
importance - 0,20
================================================
================================================
bar_Jabberwocky (BOM, conversationId:
eda6d6aa-01bd-425f-8463-419fe1ad4825) - score for preferences:
2,420866.
================================================
```

Następnie konkurujące lokale z barem Piwiarnia Warka wysyłają do niego wyliczony wynik oceny. Agent *bar_Piwiarnia Warki* otrzymuje przesłane rezultaty oraz porównuje je ze swoją oceną.

```
================================================
bar_Jabberwocky (BOM, conversationId:
eda6d6aa-01bd-425f-8463-419fe1ad4825) - sends proposal to
bar_Piwiarnia Warki.
================================================
================================================
bar_Kufle_i_kapsle (BOM, conversationId:
dacf43b9-6b83-4aec-bfd6-9bc7da6f54f2) - sends proposal to
bar_Piwiarnia Warki.
================================================
================================================
bar_Piwiarnia Warki (BOH, conversationId:
dacf43b9-6b83-4aec-bfd6-9bc7da6f54f2) - receives proposal (score:
1,053539) from bar_Kufle_i_kapsle.
================================================
================================================
bar_Piwiarnia Warki (BOH, conversationId:
eda6d6aa-01bd-425f-8463-419fe1ad4825) - receives proposal (score:
2,420866) from bar_Jabberwocky.
================================================
```

Okazuje się, że bar Jabberwocky posiada aktualnie najlepszą ofertę i to on zostaje posiadaczem najlepszej oferty.

```
================================================
bar_Piwiarnia Warki (BOH, conversationId:
eda6d6aa-01bd-425f-8463-419fe1ad4825) - is defeated by
bar_Jabberwocky.
================================================
================================================
bar_Piwiarnia Warki (BOH, conversationId:
eda6d6aa-01bd-425f-8463-419fe1ad4825) - sends proposal acceptation
to bar_Jabberwocky.
================================================
================================================
bar_Jabberwocky (BOM, conversationId:
eda6d6aa-01bd-425f-8463-419fe1ad4825) - receives proposal
acceptation from bar_Piwiarnia Warki.
================================================
================================================
bar_Piwiarnia Warki (BOH, conversationId:
dacf43b9-6b83-4aec-bfd6-9bc7da6f54f2) - wins against
bar_Kufle_i_kapsle.
================================================
================================================
bar_Piwiarnia Warki (BOH, conversationId:
dacf43b9-6b83-4aec-bfd6-9bc7da6f54f2) - sends proposal rejection to
bar_Kufle_i_kapsle.
================================================
```

Na końcu agenty potwierdzają zakończenie negocjacji. Szczegóły odnośnie realizowanego procesu negocjacji można znaleźć w rozdziale *“Opis implementacji komunikacji między agentami”*, który znajduje się w części C raportu.

```
================================================
bar_Kufle_i_kapsle (BOM, conversationId:
dacf43b9-6b83-4aec-bfd6-9bc7da6f54f2) - sends negotiations end
confirmation to bar_Piwiarnia Warki.
================================================
================================================
bar_Jabberwocky (BOM, conversationId:
eda6d6aa-01bd-425f-8463-419fe1ad4825) - sends negotiations end
confirmation to bar_Piwiarnia Warki.
================================================
================================================
bar_Piwiarnia Warki (BOH, conversationId:
dacf43b9-6b83-4aec-bfd6-9bc7da6f54f2) - receives negotiations end
confirmation  from bar_Kufle_i_kapsle.
```

```
================================================
```

Najlepszy lokal z regionu Śródmieście - bar Jabberwocky przesyła swoją ofertę klientowi.

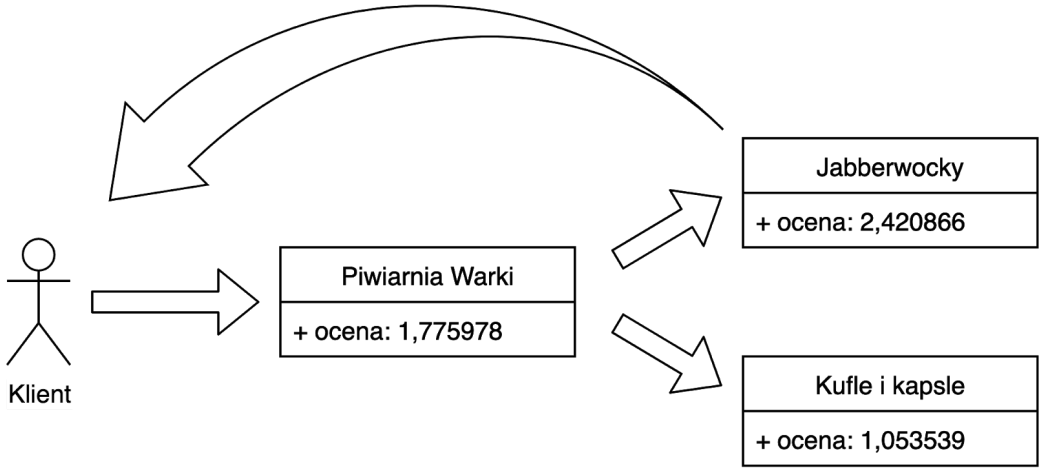
```
================================================
bar_Jabberwocky (BOH) - sends offer to customer
customer_Jan_Krafciarz.
================================================
================================================
customer_Jan_Krafciarz (customer) - receives offer: bar_Jabberwocky
- 2,420866.
================================================
```

W analogiczny sposób przebiega wyłonienie najlepszego baru w kontekście przekazanych preferencji klienta dla regionu Bemowo.

Wyniki negocjacji

W obu regionach wyłoniły się najlepsze oferty barów: Jabberwocky oraz Piwny Kolektyw. Oferty te charakteryzują się posiadaniem najtańszego piwa typu German Pilsner w swoim regionie. Wygrywają one z innymi ofertami, posiadającym piwo pożądanego stylu, pomimo największych odległości od klienta. Dzieje się tak dlatego, że cena i styl piwa są dla klienta najbardziej interesujące.

Uproszczony schemat wyniku negocjacji w regionie Śródmieście:



Uproszczony schemat wyniku negocjacji w regionie Bemowo: