Wydział Elektroniki i Technik Informacyjnych

Kierunek: Informatyka

Specjalność: Inteligentne systemy

Agentowe i aktorowe systemy decyzyjne

Zespół: Dobre Agenty

sem. 2021Z

Dokumentacja projektu

Niedobór narzędzi do nadzorowania rozwoju miasta w obszarze lokalnym przez mieszkańców

Imiona i nazwiska wykonawców projektu:

Krzysztof Łapan

Maciej Morawski

Karol Niemczycki

Jacek Prokopczuk

Kamil Wiłnicki

Link do repozytorium:

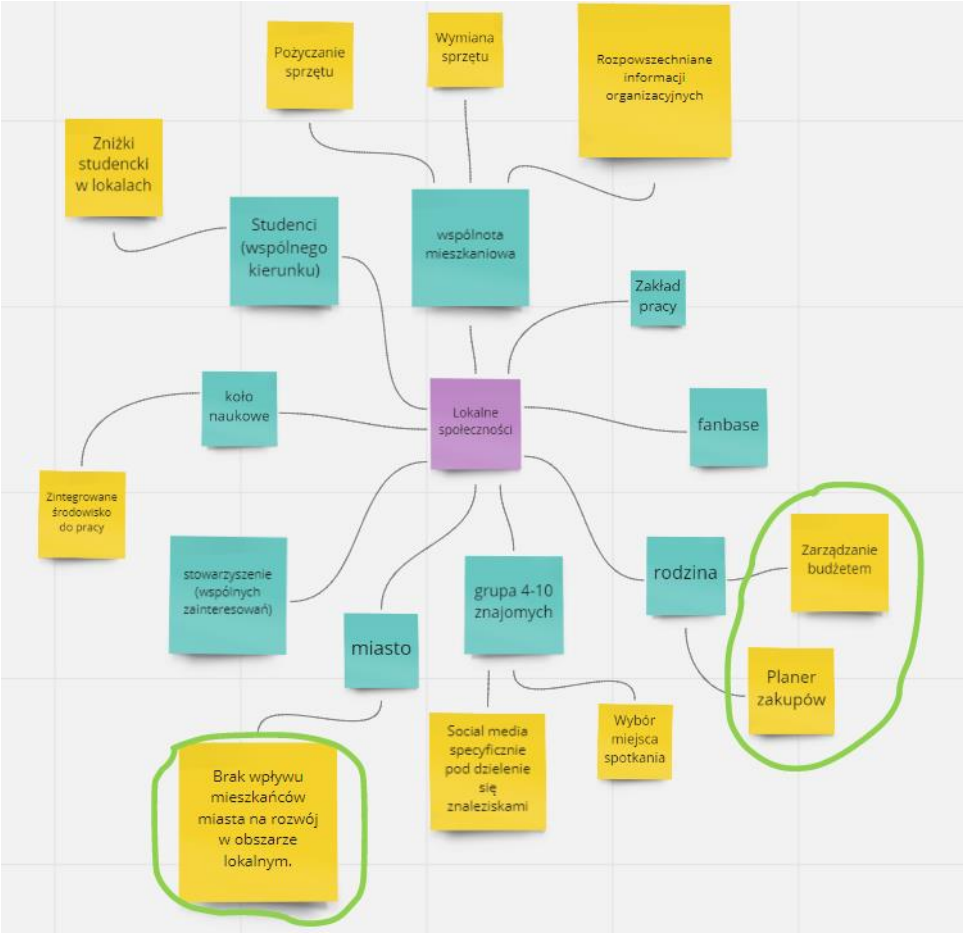
https://github.com/KarolNiem/AASD_projekt_DobreAgenty.git

Spis treści

A.1. Wstęp

Systemy wieloagentowe (MAS) to zagadnienie z zakresu informatyki określające systemy złożone z agentów, których zadaniem jest wzajemna komunikacja oraz współpraca w celu znalezienia rozwiązania postawionego przed nimi problemu. Każdy z agentów stanowi autonomiczną jednostkę zdolną do podejmowania samodzielnych decyzji oraz obserwacji środowiska, w którym została umieszczona. Komunikacja agentów ma zwykle charakter wzajemnych negocjacji i wymaga od agenta sformułowania i przekazania komunikatu. Charakterystyczną cechą systemów wieloagentowych jest ich różnorodność wynikająca z różnorodności dziedzin obliczeń, do których są stosowane. MAS wykorzystywane są na przykład w zagadnieniach związanych z wyszukiwaniem informacji w sieci internetowej czy poszukiwaniem rozwiązań zadań NP-trudnych.

W związku z szerokim obszarem zastosowań systemów wieloagentowych postanowiono wykorzystać je do rozwiązania jednego z problemów, z którymi borykają się lokalne społeczności. Punktem wyjściowym pracy była identyfikacja wspomnianych lokalnych społeczności. Następnie podjęto próbę przypisania każdej z nich problemów, z którymi ma na co dzień do czynienia. W wyniku wspólnej dyskusji opracowana została mapa myśli przedstawiona na Rysunku A.1.1 (zielonym markerem oznaczone zostały problemy, które szczególnie zwróciły naszą uwagę).



Rysunek A.1.1 Mapa myśli – poszukiwanie problemu do rozwiązania

A.2. Identyfikacja i opis problemu

A.2.1. Wybór problemu do rozwiązania

W kolejnym kroku analizowaliśmy oba problemy, które zwróciły naszą szczególną uwagę, aby wybrać jeden, nad którego rozwiązaniem będziemy pracowali. W pierwszej fazie wypisani zostali interesariusze zaznaczonych problemów wraz z ich interesami. Podsumowanie przedstawiono w Tabeli A.2.1.1.

Tabela A.2.1.1 Interesariusze problemów, które zwróciły szczególną uwagę autorów, oraz ich interesy

	A	B
Problem	Niedobór narzędzi do nadzorowania rozwoju miasta w obszarze lokalnym przez mieszkańców	Organizacja wydatków i inwestycji ze środków współdzielonych
Interesariusz - interes	• Obecni mieszkańcy – interes: zaspokojenie potrzeb, wsparcie w komunikacji z władzami; • Przyszli (potencjalni) mieszkańcy – interes: świadomy wybór nowego miejsca zamieszkania; • Deweloperzy – interes: większa świadomość potencjalnych klientów; • Urbaniści – interes: narzędzie wspomagające ich pracę, ale również kontrolujące aktualny stan zagospodarowania terenu.	• Rodzina – interes: zbilansowanie wydatków rodziny; • Grupa lokatorów/znajomych – interes: organizacja współdzielonych wydatków i wsparcie w realizacji wspólnych inwestycji; • Mały zakład pracy – interes: zwiększenie przychodu firmy; • Lokalne stowarzyszenia – interes: rozwój lokalnej kultury.

Po przeprowadzeniu głosowania jednogłośnie wybrany został Problem A ze względu na ciekawszą tematykę oraz opracowaną wstępną koncepcję rozwiązania jako systemu wieloagentowego.

A.2.2. Dokładniejszy opis problemu

Na przykładzie Warszawy zauważyliśmy, że istnieją kanały do komunikacji z władzami miasta poprzez możliwości zgłaszania pomysłów inwestycji przez mieszkańców (m.in. Budżet Obywatelski), jednak brak jest narzędzi do wyznaczenia wartości (kosztów oraz atrakcyjności) danej propozycji w sposób przystępny dla obywateli. Ocena inwestycji pozwoliłaby wspomóc obywateli w zgłaszaniu ważnych dla lokalnych społeczności potrzeb i wzbudzać w mieszkańcach poczucie wpływu na swoje otoczenie. Jednocześnie od razu uświadamiając obywateli, czy dane propozycje są możliwe do realizacji.

A.3. Propozycje rozwiązań

Kolejnym krokiem po określeniu problemu trapiącego lokalne społeczności było wytypowanie sposobów, w jakie można go rozwiązać. W tym celu została użyta technika “burzy mózgów”, w której każdy uczestnik zespołu wymienił wszystkie rozwiązania, które przyszły mu do głowy. Ważnym elementem tej techniki jest anonimowość propozycji oraz zakaz komentowania i krytykowania podanych pomysłów. Zachęca to do wymyślania śmiałych, a nawet absurdalnych pomysłów, z których można następnie wybrać fragmenty, które zostaną wykorzystane do sformułowania finalnego rozwiązania. Zapisane pomysły pogrupowaliśmy na bloki tematyczne, które podane są poniżej.

Inicjatywa obywatelska - platforma do wywierania wpływu na otoczenie przez mieszkańców:

- Zgłaszanie przez mieszkańców problemów na mapie;
- Adaptacja pomysłów mieszkańców do formularza budżetu obywatelskiego;
- Zgłaszanie zagrożeń na osiedlu;
- Realizacja w postaci aplikacji, w której mieszkańcy mogą mieć możliwość wystosowania petycji do władz miasta;
- Realizacja w postaci aplikacji, w której mieszkańcy mogą brać udział w głosowaniach/ankietach na temat rozwoju.

Wsparcie komunikacji - interakcja pomiędzy mieszkańcami a lokalnym samorządem; kanał komunikacji pomiędzy nimi:

- Platforma do zgłaszania problemów i pomysłów (rubryka na tablicy ogłoszeń przed blokiem);
- Automatyczne wysyłanie zgłoszeń na 19115 (lub inne Centrum Kontaktu);
- Instytucja przyjmuje powiadomienia od mieszkańców;
- Kooperacyjne planowanie przestrzeni lub przebudowy aktualnej przestrzeni;
- Władze miasta lub inwestorzy mogą mieć możliwość zgłaszania swoich pomysłów, które następnie podlegają ocenie pod względem użyteczności;
- Każde miejsce w lokalnej miejscowości może podlegać ocenie w zakresie tego, czy jest ono odpowiednie pod dany typ inwestycji.

Analiza atrakcyjności - wgląd mieszkańców na obecny stan infrastruktury w danej okolicy:

- Mapa dostępności komunikacyjnej;
- Mapa zieleni miejskiej (włącznie z prywatnymi, jeśli ktoś chce się pochwalić);
- Każde miejsce w lokalnej miejscowości może podlegać ocenie w zakresie tego, czy jest ono odpowiednie pod dany typ inwestycji.

Analiza potrzeb - ewaluacja potrzebności inwestycji na podstawie danych o lokalnej społeczności:

- Kalkulacja skali popytu usług według liczby mieszkańców, gęstości zaludnienia i podaży według pojemności obiektów;
- Analiza wieku mieszkańców, która miałaby wpływ na infrastrukturę.

Implementacja aplikacji - techniczna realizacja systemu, dostępne funkcje aplikacji:

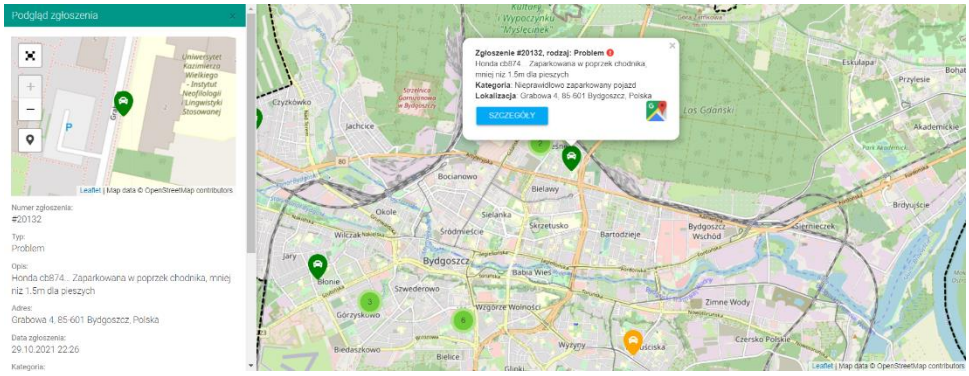
- Aplikacja mobilna;
- Każda propozycja/zgłoszenie to agent;
- Agenci wymieniają się informacjami o istniejącej infrastrukturze;
- Agenci szukają chętnych po lokalizacji i już popartych akcjach;
- Dostępna tablica z akcjami/propozycjami.

A.4. Analiza źródeł

W kolejnym kroku nasz problem poddaliśmy analizie w kontekście istniejących rozwiązań. Niestety, najczęściej rozrost miast nie idzie w parze z odpowiednim zarządzaniem przestrzenią publiczną oraz negatywnie wpływa na przepływ informacji między lokalnymi społecznościami a włodarzami. Brak

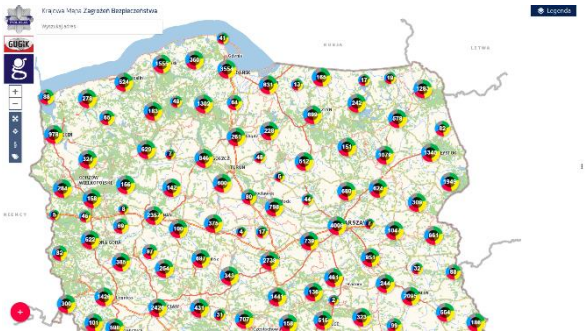
odpowiedniej komunikacji prowadzi do niezadowolenia mieszkańców oraz do pogłębienia braku ufności do osób zarządzających miastem.

Dzięki nowym technologiom takowy brak komunikacji może być częściowo bądź całkowicie wyeliminowany. Najczęściej dokonywane jest to poprzez odpowiednie aplikacje i programy. Na rynku istnieje wiele rozwiązań, które umożliwiają organom miasta szybko reagować na zgłaszane problemy. Przykładem takiego rozwiązania jest na przykład portal działający na terenie miasta Bydgoszcz (Rysunek A.4.1). Mieszkańcy w bardzo prosty sposób mogą nanieść na mapę informację o rzeczach, które ich zaniepokoiły bądź zagrażają niebezpieczeństwu innym osobom. Strona wydaje się być używana i bez przerwy aktualizowana. Każdego dnia można znaleźć na niej nowe wpisy, które weryfikowane są przez administrację portalu.



Rysunek A.4.1 Bydgoski portal do zamieszczania zgłoszeń przez mieszkańców.

Podobne rozwiązanie zastosowano na Krajowej Mapie Zagrożeń Bezpieczeństwa (Rysunek A.4.2). Strona obejmuje tym razem już cały obszar Polski. Niestety strona jest mało znana, a wpisy dokonane kilka lat temu przez użytkowników są nadal widoczne na mapie. Wydaje się, że ta mapa zagrożeń nie cieszy się tak dużą popularnością jak bydgoski portal, który spełnia w założeniu to same zadanie.



Rysunek A.4.2 Krajowa Mapa Zagrożeń Bezpieczeństwa.

Oprócz możliwości zgłaszania zagrożeń, istnieją portale, które wspierają komunikację między samymi mieszkańcami małych społeczności. Przykładem może być mycoop.com, który ogranicza się do społeczności mieszkającej w jednym budynku. Okazuje się, że człowiek może stać się bardziej anonimowy mieszkając z innymi w jednym budynku (na przykład w bloku), niż mieszkając w wolnostojącym domu na terenach wiejskich, gdzie więzy społeczne są znacznie bardziej zacieśnione.

Portal umożliwia szybką komunikację między mieszkańcami, którzy mogą udostępniać istotne informacje między sobą. Takie rozwiązanie pozwala na szybkie reagowanie (np. w obliczu jakiegoś niebezpieczeństwa jak zalanie mieszkania). Przy czym, korzystając z aplikacji, nie jest konieczne

udostępnianie takich danych jak numer telefonu. Dzięki temu można nadal zachować swoja anonimowość i być na bieżąco z problemami innych mieszkańców.

Powyższe przykłady są niezwykle złożonymi aplikacjami, które skutecznie pomagają w codziennym funkcjonowaniu. Podobne złożone problemy zostały przedstawione jako problemy w ujęciu agentowym. Przykładem jest artykuł ukazujący, w jaki sposób można rozbić ideę Smart City na mniejsze części składowe, które będą stanowione przez pojedynczych agentów. W artykule zaproponowano sposób komunikacji między agentami oraz sposób ich organizacji. Ze względu na złożoność przedstawianego projektu, wymienione zasady oraz porady wpasowują się w nasze potrzeby.

A.5. Opis koncepcji systemu

W celu doprecyzowania opracowywanego rozwiązania, spośród bloków tematycznych wymienionych w Rozdziale A.3 zostały wybrane dwa, które będą stanowić trzon pomysłu: inicjatywa obywatelska oraz wsparcie w komunikacji. Analiza atrakcyjności inwestycji będzie narzędziem wspierającym główny cel, jakim jest ułatwienie mieszkańcom danego obszaru sygnalizowania ich potrzeb instytucjom odpowiedzialnym za rozwój przestrzeni publicznej.

Zgodnie z założeniami projektu rozwiązanie będzie realizowane w formie systemu wieloagentowego. Głównym procesem, za który będą odpowiedzialne agenty, jest przetwarzanie zgłoszeń od mieszkańców z propozycjami nowych inwestycji.

System zostanie zaimplementowany w formie aplikacji mobilnej bądź serwisu webowego.

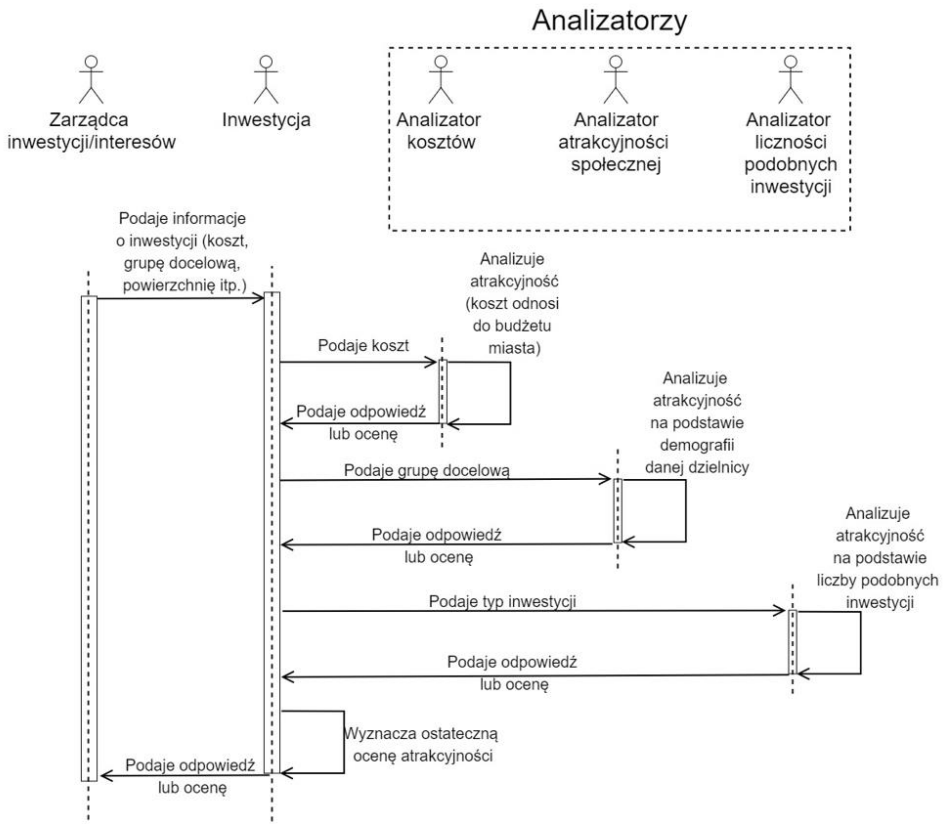
We wstępnej koncepcji architektury systemu zaproponowaliśmy utworzenie trzech grup agentów: *Zarządca*, *Inwestycje*, *Analizatorzy*. Pierwszy z nich byłby pojedynczym agentem z grupy *Zarządca*, który kontaktowałby się z mieszkańcem zgłaszającym inwestycję. Agent ten odbiera podanie od użytkownika i na podstawie otrzymanej informacji określa parametry tej inwestycji (m.in koszt, grupę docelową, powierzchnię, lokalizację) i przekazuje je do agenta z grupy *Inwestycje*, którego w tym momencie tworzy. Następnie agent z grupy *Inwestycje* przekazuje niezbędne informacje do agentów z grupy *Analizatorzy*, którzy zwracaliby odpowiedź (ewentualne odrzucenie) lub ocenę zgłoszonej inwestycji. Po każdej otrzymanej odpowiedzi agent *Inwestycji* sprawdza, czy dalsza analiza ma sens. Jeśli inwestycja zostanie odrzucona, to kontaktuje się z agentem *Zarządca* i przekazuje tę informację.

Zaproponowani przez nas analizatorzy to:

- *Analizator kosztów* – otrzymywałby od agenta *Inwestycji* wysokość kosztów i porównywałby je z punktem odniesienia, np. budżetem miejskim. Na podstawie swojej analizy odsyłałby odpowiedź, czy dana inwestycja mieści się w ogóle w budżecie, oraz liczbową ocenę kosztów;
- *Analizator atrakcyjności społecznej* – otrzymywałby od agenta *Inwestycji* informację o grupie docelowej, której dotyczy inwestycja i odnosiłby tę informację do struktury ludności na danym obszarze. Na podstawie swojej analizy odsyłałby odpowiedź, jak bardzo dana inwestycja byłaby przydatna poprzez liczbową ocenę atrakcyjności;
- *Analizator liczności podobnych inwestycji* – otrzymywałby od agenta *Inwestycji* informację o jej typie i zliczałby występowanie podobnych obiektów na danym terenie. Na podstawie swojej analizy odsyłałby odpowiedź, czy obszar oddziaływania inwestycji nie pokrywa się ze zbyt dużą liczbą istniejących obiektów danego typu oraz liczbową ocenę atrakcyjności.

Po otrzymaniu informacji od wszystkich analizatorów agent z grupy *Inwestycje* przekazywałby ostateczną ocenę (rangę danę inwestycji określającą jak ważny jest problem oraz jak bardzo jest rzeczywisty do wykonania) do agenta *Zarządca*, który przekazywałby ją do władz miasta.

Opisaną wstępną koncepcję architektury systemu przedstawiono na Rysunku A.5.1.



Rysunek A.5.1 Zaproponowana wstępna koncepcja architektury systemu (komunikacji pomiędzy agentami)

B.1. Wybór metodologii do projektowania systemu

Po wstępnej analizie problemu, źródeł oraz opracowaniu wstępnej koncepcji systemu postanowiono dokonać wyboru metodologii, przy użyciu której zaprojektowana zostanie docelowa struktura systemu. W wyniku wspólnej dyskusji oraz rozważenia wad i zalet każdej ze stosowanych najczęściej metod zdecydowano, że system zostanie zaprojektowany zgodnie z wytycznymi metodologii Gaia. Podstawą do rozważań był schemat przedstawiony na rysunku A.5.1. Na jego podstawie można określić role stanowiące rdzeń działania systemu, co jest jedną z głównych zasad metody Gaia. Ponadto w oparciu o niego można wnioskować o zależnościach i interakcjach między rolami.

W wyniku dyskusji ustalono także podstawowe funkcjonalności systemu agentowego:

- Możliwość zgłaszania przez użytkowników pomysłów inwestycji;
- Ocena atrakcyjności inwestycji w kontekście kosztów;
- Ocena atrakcyjności inwestycji w kontekście demografii;
- Ocena atrakcyjności inwestycji w kontekście liczby podobnych inwestycji;
- Wysłanie informacji zwrotnej użytkownikowi;
- Pomysł jest odnotowywany w bazie pomysłów wraz z przypisaną do niego oceną.

Funkcjonalności systemu można będzie również rozwinąć o:

* Automatyczne przesyłanie pomysłów o ocenie spełniającej postawione kryteria do urzędu miasta.

Przy okazji rysunek A.5.1. może być traktowany jako opis procesu obsługi obywatela oraz wstępny model środowiska. Przedstawione są na nim podstawowe zasoby systemu oraz relacje między nimi. Widoczne zależności wchodzą w skład dynamiki środowiska. Agenty wchodząc w interakcje między sobą dokonują różnego rodzaju operacji, takich jak odczytywanie, zmiana lub generowanie współdzielonych zasobów obliczeniowych.

B.2. Identyfikacja ról systemu

Po rozważeniu problemu w kontekście modelowania systemu przy użyciu metodologii Gaia postanowiono podjąć dalsze kroki. Ze względu na to, że metodologia Gaia zakłada pominięcie fazy formułowania wymagań, w kolejnym kroku dokonano identyfikacji ról występujących w systemie. Proces ten przeprowadzony został w oparciu o składowe rysunku A.5.1.

W wyniku identyfikacji udało się rozróżnić następujące role:

- Zarządca inwestycji/interesów:
    - rola zapewniająca kontakt z klientem: **CustomerHandler**
- Inwestycja:
    - rola czuwająca nad procesem komunikacji z analizatorami: **CommManager**
    - rola wyznaczająca ogólną ocenę inwestycji: **GlobalEvaluator**
- Analizator kosztów:
    - rola sprawdzenia, czy inwestycja mieści się w budżecie: **BudgetChecker**
    - rola oceny atrakcyjności ze względu na koszt: **CostEvaluator**

- Analizator atrakcyjności społecznej (demografii):
    - rola oceny atrakcyjności na podstawie demografii: **AgeStructEvaluator**
- Analizator liczności podobnych inwestycji:
    - rola oceny atrakcyjności ze względu na podobne inwestycje: **UsabilityEvaluator**
- Klient: **Customer**

B.3. Aktywności i protokoły ról

Każda ze zidentyfikowanych ról ma przypisane aktywności i protokoły, które wykorzystuje do wykonania swoich zadań. Aktywności oznaczają działania, które agent może podjąć bez interakcji z innymi agentami, protokoły opisują natomiast sposób i cel komunikacji z innymi. Poniżej znajduje się opis obu tych atrybutów dla każdej ze zdefiniowanych ról.

CustomerHandler: otrzymuje od Klienta informacje o pomyśle, generuje ofertę pomysłu (podsumowanie propozycji inwestycji) i przekazuje ją do Inwestycji, odbiera dane od Inwestycji o ocenie pomysłu oraz przekazuje ją dalej (wysyła do Klienta oraz zapisuje w bazie danych).

- Protokoły**:**
    - ReceiveIdea,
    - SendToCommManager,
    - ReceiveFromGlobalEvaluator,
    - InformCustomer.
- Aktywności**:**
    - GenerateOffer,
    - SaveToDB.

CommManager: otrzymuje ofertę pomysłu i przesyła ją dalej do analizatorów oraz odbiera odpowiedzi od analizatorów, tworzy ich podsumowanie i przekazuje do globalnego ewaluatora.

- Protokoły**:**
    - ReceiveOffer,
    - SendToCostEvaluator,
    - SendToAgeStructEvaluator,
    - SendToUsabilityEvaluator,
    - SendToBudgetChecker,
    - ReceiveFromCostEvaluator,
    - ReceiveFromAgeStructEvaluator,
    - ReceiveFromUsabilityEvaluator,
    - ReceiveFromBudgetChecker,
    - SendToGlobalEvaluator.
- Aktywności**:**
    - GenerateEvaluationSummary.

GlobalEvaluator: otrzymuje podsumowanie ocen od CommManagera, wyznacza ocenę globalną pomysłu i przekazuje ją zarządcy.

- **Protokoły:**
  - ReceiveSummaryFromCommManager,
  - SendEvaluationToCustomerHandler.
- **Aktywności:**
  - EvaluateGlobally.

BudgetChecker: otrzymuje ofertę pomysłu od CommManagera, wyznacza ocenę pomysłu poprzez sprawdzenie, czy inwestycja mieści się w budżecie, i przekazuje ją z powrotem do CommManagera.

- **Protokoły:**
  - ReceiveFromCommManager,
  - SendDecision.
- **Aktywności:**
  - EvaluateBudget.

CostEvaluator: otrzymuje ofertę pomysłu od CommManagera, wyznacza ocenę pomysłu ze względu na koszt i przekazuje ją z powrotem do CommManagera.

- **Protokoły:**
  - ReceiveFromCommManager,
  - SendEvaluation.
- **Aktywności:**
  - EvaluateCost.

AgeStructEvaluator: otrzymuje ofertę pomysłu od CommManagera, wyznacza ocenę pomysłu ze względu na strukturę wieku i przekazuje ją z powrotem do CommManagera.

- **Protokoły:**
  - ReceiveFromCommManager,
  - SendEvaluation.
- **Aktywności:**
  - EvaluateAgeStructure.

UsabilityEvaluator: otrzymuje ofertę od CommManagera, wyznacza ocenę pomysłu ze względu na liczbę podobnych inwestycji i przekazuje ją z powrotem do CommManagera.

- **Protokoły:**
  - ReceiveFromCommManager,
  - SendEvaluation.
- **Aktywności:**
  - EvaluateUsability.

Customer: wysyła informacje o pomyśle i otrzymuje jego ocenę.

- **Protokoły:**
  - SendIdea,
  - ReceiveEvaluation
- **Aktywności:**
  - Brak

B.4. Pozwolenia ról

Dodatkowymi atrybutami roli są pozwolenia, jakimi ta rola dysponuje. Określają one prawa, jakie posiada rola, które pozwalają wykonać powierzone jej zadania. Pozwolenia dzielą się na prawo do odczytu (*reads*), zapisu (*changes*) oraz tworzenia (*generates*). Dodatkowo możliwe jest zdefiniowanie pozwolenia na operowanie danymi dostarczonymi od innego agenta (*supplied*).

Poniżej znajdują się pozwolenia przydzielone poszczególnym rolom.

CustomerHandler:

- **reads supplied** idea // informacje o pomyśle,
- **reads supplied** evaluation // ocena inwestycji,
- **reads supplied** customerDetails // dane klienta,
- **generates** offer // wygenerowana oferta.

CommManager:

- **reads supplied** offer // otrzymana oferta,
- **reads supplied** costEvaluation // ocena ze względu na koszt,
- **reads supplied** ageStructEvaluation // ocena ze względu na strukturę wieku,
- **reads supplied** usabilityEvaluation // ocena ze względu na liczbę podobnych inwestycji,
- **reads supplied** budgetEvaluation // ocena, czy oferta mieści się w budżecie,
- **generates** evaluationSummary // podsumowanie (suma) ocen od różnych ewaluatorów.

GlobalEvaluator:

- **reads supplied** evaluationSummary // podsumowanie ocen od różnych ewaluatorów,
- **generates** evaluation // ogólna ocena inwestycji.

BudgetChecker:

- **reads supplied** offer // oferta,
- **generates** budgetEvaluation // // ocena, czy oferta mieści się w budżecie.

CostEvaluator:

- **reads supplied** offer // oferta,
- **generates** costEvaluation // ocena ze względu na koszt.

AgeStructEvaluator:

- **reads supplied** offer // oferta,
- **generates** ageStructEvaluation // ocena ze względu na strukturę wieku.

UsabilityEvaluator:

- **reads supplied** offer // oferta,
- **generates** usabilityEvaluation // ocena ze względu na liczbę podobnych inwestycji.

Customer:

- **reads supplied** evaluation // ocena inwestycji,
- **generates** idea // informacje o pomyśle,
- **generates** customerDetails // dane klienta.

B.5. Żywotne obowiązki

Kolejnym elementem opisywanym w modelu Gaia są obowiązki każdej z ról. Definiują one funkcjonalności, jakimi cechuje się dana rola. Jednym z rodzajów obowiązków są tzw. obowiązki żywotne. Określają one ciąg aktywności i protokołów, które rola wykonuje tak długo, jak działa poprawnie („żyje"). Gaia specyfikuje obowiązki żywotne jako "*liveness expressions*", których składnia jest podobna do wyrażeń regularnych. Dla każdej z wcześniej zdefiniowanych ról widoczne są przypisane im obowiązki żywotne:

CustomerHandler = (ReceiveIdea . GenerateOffer . SendToCommManager . ReceiveFromGlobalEvaluator . SaveToDB . InformCustomer)$^{\omega}$

CommManager = (ReceiveOffer . CommunicateWithEvaluators . GenerateEvaluationSummary . SendToGlobalEvaluator)

CommunicateWithEvaluators = ((SendToCostEvaluator || SendToAgeStructEvaluator || SendToUsabilityEvaluator || SendToBudgetChecker) . (ReceiveFromCostEvaluator || ReceiveFromAgeStructEvaluator || ReceiveFromUsabilityEvaluator || ReceiveFromBudgetChecker)) // obowiązek pomocniczy, grupująca protokoły służące do komunikacji z różnymi ewaluatorami

GlobalEvaluator = (ReceiveSummaryFromCommManager . EvaluateGlobally . SendEvaluationToCustomerHandler)

BudgetChecker = (ReceiveFromCommManager . EvaluateBudget . SendDecision)

CostEvaluator = (ReceiveFromCommManager . EvaluateCost . SendEvaluation)

AgeStructEvaluator = (ReceiveFromCommManager . EvaluateAgeStructure. SendEvaluation)

UsabilityEvaluator = (ReceiveFromCommManager . EvaluateUsability . SendEvaluation)

Customer = (SendIdea . ReceiveEvaluation)+ // sekwencja wysłania pomysłu i otrzymania odpowiedzi musi zostać wykonana przynajmniej raz, aby agent realizujący rolę Customer miał rację bytu

B.6. Obowiązki bezpieczeństwa

W modelu Gaia określenie obowiązków żywotnych bywa często niewystarczające. Opis struktury systemu powinien zawierać także tak zwane obowiązki bezpieczeństwa, które wykluczać będą występowanie pewnych niepożądanych warunków. Są one wyrażane w formie predykatów, w skład których wchodzą określone dla danej roli pozwolenia. Obowiązki bezpieczeństwa określone dla każdej z ról systemu przedstawiają się następująco:

- **CustomerHandler:**
  - **True** – rola o nieskończonym czasie trwania.
- **CommManager**:
  - **infoAvailable(customerDetails)** – jeśli agent ma dostęp do danych osoby zgłaszającej może działać poprawnie.
- **GlobalEvaluator:**
  - **budgetEvaluation = false ⇒ evaluation = NULL** – jeśli inwestycja przekracza budżet wówczas nie zwracamy ewaluacji.
- **CostEvaluator:**
  - **True**
- **AgeStructEvaluator:**
  - **True**
- **UsabilityEvaluator:**
  - **True**
- **BudgetChecker:**
  - **True**
- **Customer:**
  - **True**

B.7. Model ról

Na podstawie określonych ról oraz zdefiniowanych dla nich aktywności, protokołów, pozwoleń i obowiązków, można sporządzić schematy ról. Przedstawiają się one następująco:

- **CustomerHandler**

Schemat roli: CustomerHandler
Opis: otrzymuje od Klienta informacje o pomyśle, generuje ofertę pomysłu (podsumowanie propozycji inwestycji) i przekazuje ją do Inwestycji, odbiera dane od Inwestycji o ocenie pomysłu oraz przekazuje ją dalej (wysyła do Klienta oraz zapisuje w bazie danych).
Protokoły i aktywności: ReceiveIdea, SendToCommManager, ReceiveFromGlobalEvaluator, InformCustomer, GenerateOffer, SaveToDB

Pozwolenia:

- **reads supplied** idea
- **reads supplied** evaluation
- **reads supplied** customerDetails
- **generates** offer

Obowiązki:

- Żywotne:

  CustomerHandler = (ReceiveIdea . GenerateOffer . SendToCommManager . ReceiveFromGlobalEvaluator . SaveToDB . InformCustomer)$^{\omega}$

- Bezpieczeństwa:

  True

- **CommManager**

Schemat roli: CommManager
Opis: otrzymuje ofertę pomysłu i przesyła ją dalej do analizatorów oraz odbiera odpowiedzi od analizatorów, tworzy ich podsumowanie i przekazuje do globalnego ewaluatora.
Protokoły i aktywności: ReceiveOffer, SendToCostEvaluator, SendToAgeStructEvaluator, SendToUsabilityEvaluator, SendToBudgetChecker, ReceiveFromCostEvaluator, ReceiveFromAgeStructEvaluator, ReceiveFromUsabilityEvaluator, ReceiveFromBudgetChecker, SendToGlobalEvaluator, GenerateEvaluationSummary.
Pozwolenia: • **reads supplied** offer • **reads supplied** costEvaluation • **reads supplied** ageStructEvaluation • **reads supplied** usabilityEvaluation • **reads supplied** budgetEvaluation • **generates** evaluationSummary
Obowiązki: • Żywotne: **CommManager** = (ReceiveOffer . CommunicateWithEvaluators . GenerateEvaluationSummary . SendToGlobalEvaluator) **CommunicateWithEvaluators** = ((SendToCostEvaluator || SendToAgeStructEvaluator || SendToUsabilityEvaluator || SendToBudgetChecker) . (ReceiveFromCostEvaluator || ReceiveFromAgeStructEvaluator || ReceiveFromUsabilityEvaluator|| ReceiveFromBudgetChecker)) • Bezpieczeństwa: **infoAvailable(customerDetails)**

- **GlobalEvaluator**

Schemat roli: GlobalEvaluator
Opis: otrzymuje podsumowanie ocen od CommManagera, wyznacza ocenę globalną pomysłu i przekazuje ją zarządcy.
Protokoły i aktywności: ReceiveSummaryFromCommManager, SendEvaluationToCustomerHandler, EvaluateGlobally
Pozwolenia: • **reads supplied** evaluationSummary • **generates** evaluation
Obowiązki: • Żywotne: **GlobalEvaluator** = (ReceiveSummaryFromCommManager . EvaluateGlobally . SendEvaluationToCustomerHandler) • Bezpieczeństwa: **budgetEvaluation = false ⇒ evaluation = NULL**

- **BudgetChecker**

Schemat roli: BudgetChecker
Opis: otrzymuje ofertę pomysłu od CommManagera, wyznacza ocenę pomysłu poprzez sprawdzenie, czy inwestycja mieści się w budżecie, i przekazuje ją z powrotem do CommManagera.
Protokoły i aktywności: ReceiveFromCommManager, SendEvaluation, EvaluateBudget
Pozwolenia: • **reads supplied** offer • **generates** budgetEvaluation
Obowiązki: • Żywotne: **BudgetChecker** = (ReceiveFromCommManager . EvaluateBudget . SendDecision) • Bezpieczeństwa: **True**

- CostEvaluator

Schemat roli: CostEvaluator
Opis: otrzymuje ofertę pomysłu od CommManagera, wyznacza ocenę pomysłu ze względu na koszt i przekazuje ją z powrotem do CommManagera.

Protokoły i aktywności: ReceiveFromCommManager, SendEvaluation, EvaluateCost
Pozwolenia: • reads supplied offer • generates costEvaluation
Obowiązki: • Żywotne: CostEvaluator = (ReceiveFromCommManager . EvaluateCost . SendEvaluation) • Bezpieczeństwa: True

- **AgeStructEvaluator**

Schemat roli: AgeStructEvaluator
Opis: otrzymuje ofertę pomysłu od CommManagera, wyznacza ocenę pomysłu ze względu na strukturę wieku i przekazuje ją z powrotem do CommManagera.
Protokoły i aktywności: ReceiveFromCommManager, SendEvaluation, EvaluateAgeStructure
Pozwolenia: • **reads supplied** offer • **generates** ageStructEvaluation
Obowiązki: • Żywotne: **AgeStructEvaluator** = (ReceiveFromCommManager . EvaluateAgeStructure. SendEvaluation) • Bezpieczeństwa: **True**

- **UsabilityEvaluator**

Schemat roli: UsabilityEvaluator
Opis: otrzymuje ofertę pomysłu od CommManagera, wyznacza ocenę pomysłu ze względu na liczbę podobnych inwestycji i przekazuje ją z powrotem do CommManagera.
Protokoły i aktywności: ReceiveFromCommManager, SendEvaluation, EvaluateUsability
Pozwolenia: • **reads supplied** offer • **generates** usabilityEvaluation
Obowiązki: • Żywotne: **UsabilityEvaluator** = (ReceiveFromCommManager . EvaluateUsability . SendEvaluation)

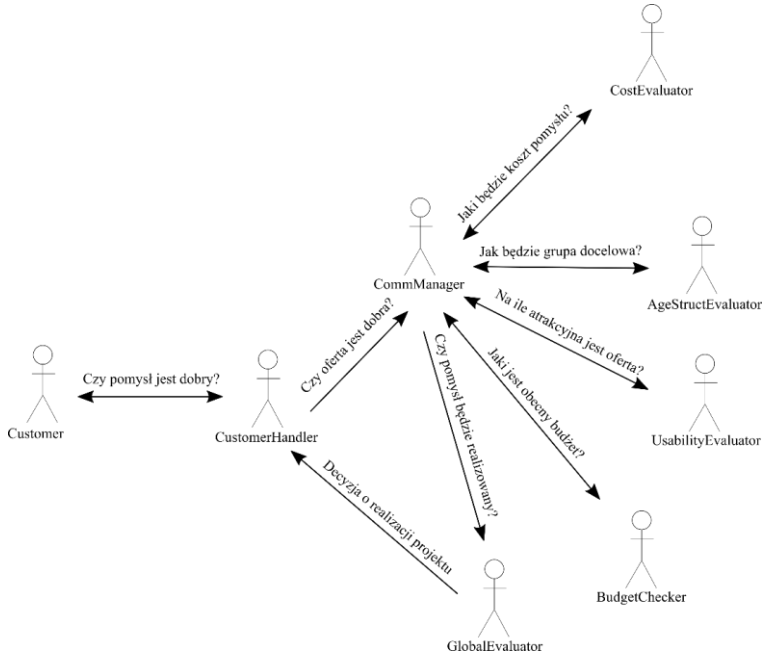
• Bezpieczeństwa: **True**

- **Customer**

Schemat roli: Customer
Opis: wysyła informacje o pomyśle i otrzymuje jego ocenę.
Protokoły i aktywności: SendIdea, ReceiveEvaluation
Pozwolenia: • **reads supplied** evaluation • **generates** idea • **generates** customerDetails
Obowiązki: • Żywotne: **Customer** = (SendIdea . ReceiveEvaluation)+ • Bezpieczeństwa: **True**

B.8. Model interakcji

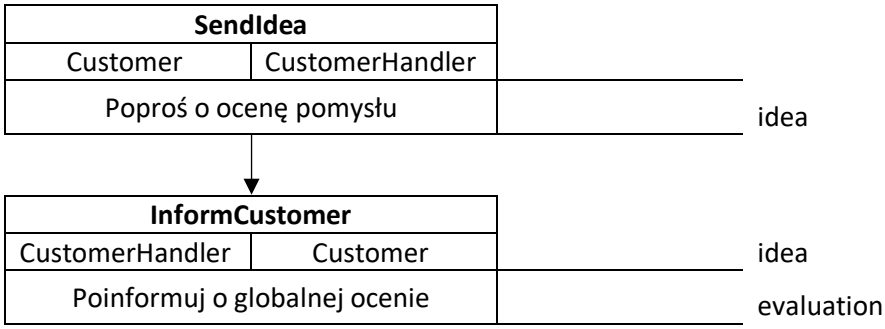
Po sporządzeniu schematów ról można określić zależności i relacje pomiędzy nimi. Właściwe ich określenie jest kluczowym zagadnieniem w kontekście projektowania systemu wieloagentowego. W metodologii Gaia zależności i relacje pomiędzy określonymi rolami prezentuje się poprzez model interakcji (którego schemat został przedstawiony na Rys. B.8.1).



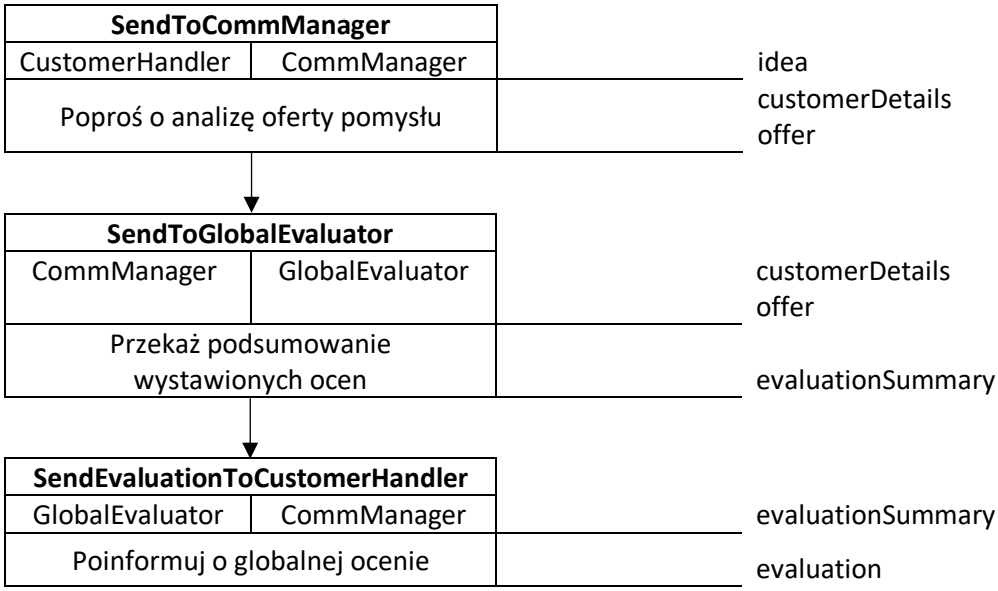
Rysunek B.8.1 Schemat modelu interakcji

Następnie zależności i relacje pomiędzy rolami opisano dokładniej poprzez diagramy protokołów.

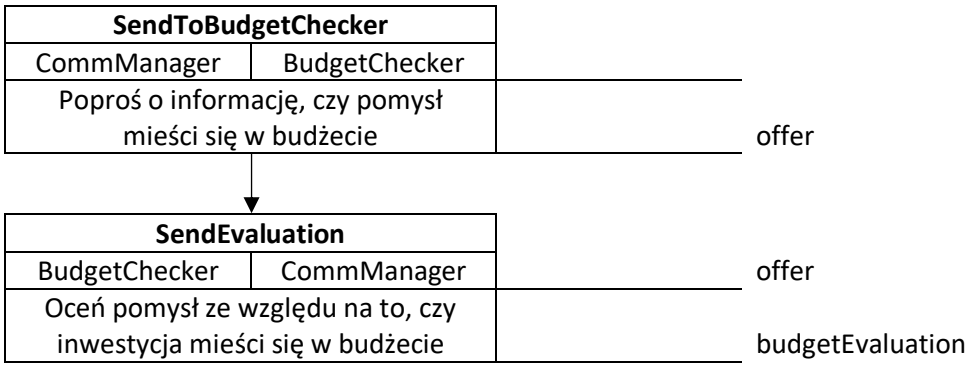
- **Wysłanie przez klienta pomysłu do oceny**



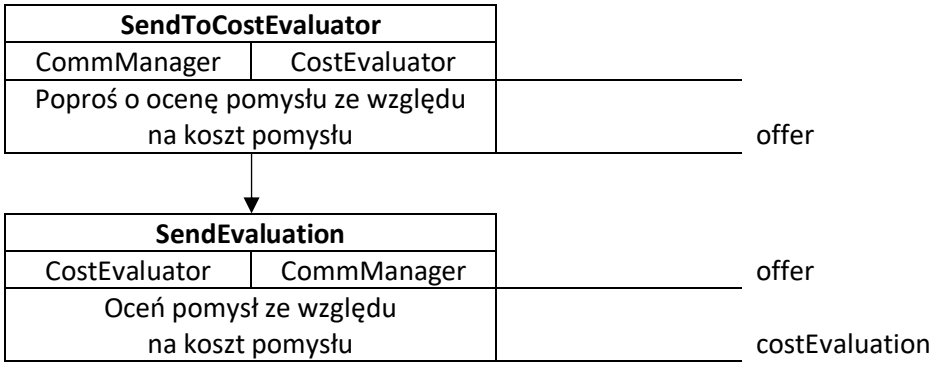
- **Wysłanie oferty pomysłu do oceny**



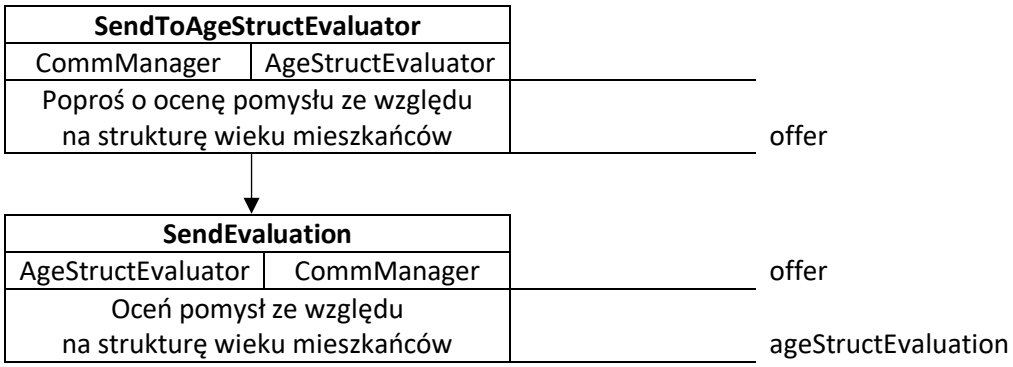
- **Sprawdzenie, czy oferta mieści się w budżecie**



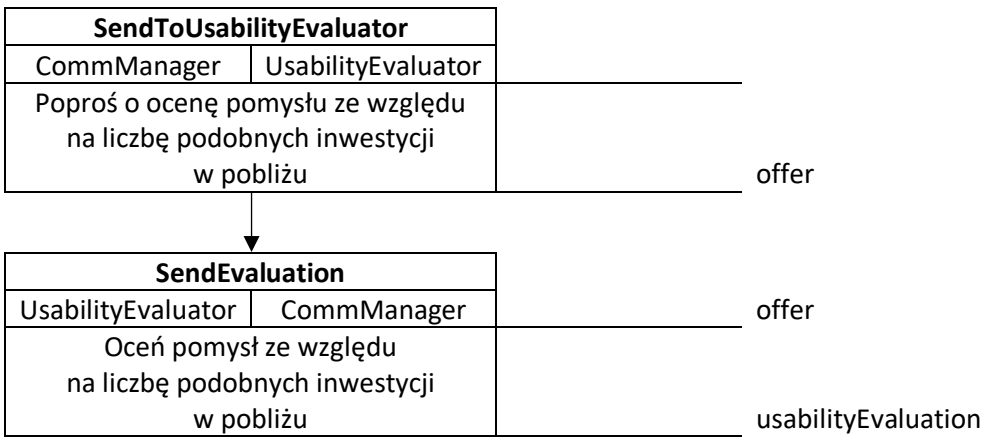
- **Ocena kosztów pomysłu**



- **Ocena pomysłu ze względu na strukturę wieku mieszkańców**

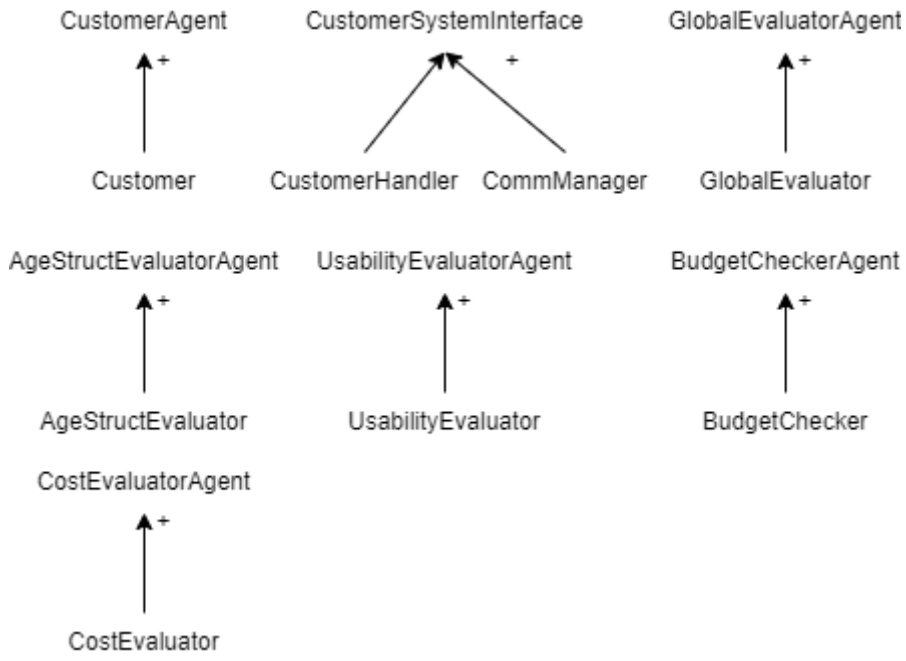


- **Ocena pomysłu ze względu na liczbę podobnych inwestycji w pobliżu**



B.9. Model agentów

Kolejnym etapem po sporządzeniu modelu interakcji i opisaniu zależności pomiędzy rolami poprzez diagramy protokołów było określenie pierwszego elementu projektu systemu wieloagentowego - modelu agentów. Ponieważ pojedynczy agent jest bytem obliczeniowym, który może odgrywać różne role, zaprojektowano agenta CustomerSystemInterface poprzez złożenie blisko powiązanych ról: CustomerHandler oraz CommManager. Inne agenty określone zostały na podstawie pozostałych ról (w relacji 1:1).
Model agentów przedstawiono na Rys. B.9.1.



Rys. B.9.1 Schemat modelu agentów

B.10. **Model usług**

Kolejnym elementem projektu systemu wieloagentowego jest model usług, który służy do identyfikacji usług skojarzonych z każdym agentem. Model usług jest pochodną zdefiniowanych protokołów, aktywności, odpowiedzialności i pozwoleń, gdzie:

- wejście i wyjście to pochodne protokołów,
- warunki wstępne i warunki końcowe to pochodne reguł bezpieczeństwa oraz reguł organizacyjnych.

Model usług przedstawiono w Tabeli B.10.1.

Tabela B.10.1.: Model usług

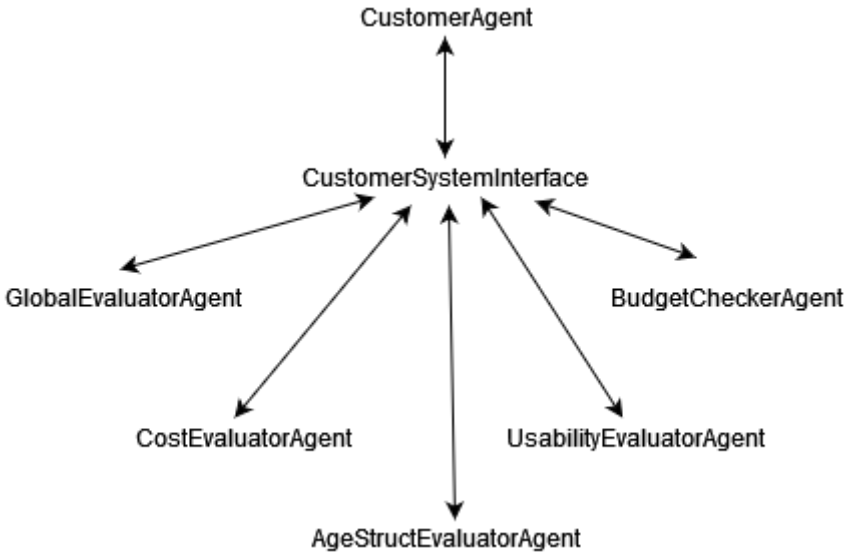
Usługa	Wejścia	Wyjścia	Warunki wstępne	Warunki końcowe
Przekaż informacje o pomyśle od klienta do oceny	idea customerDetails	offer	idea ≠ NULL customerDetails ≠ NULL offer = NULL	offer ≠ NULL

Sprawdź zgodność pomysłu z budżetem	offer	budgetEvaluation	offer ≠ NULL budgetEvaluation = NULL	budgetEvaluation ∈ (true, false)
Przeprowadź analizę zgłoszonego pomysłu według przyjętych kryteriów	offer	costEvaluation ageStructEvaluation usabilityEvaluation	offer ≠ NULL CostEvaluation = NULL ageStructEvaluation = NULL usabilityEvaluation = NULL	costEvaluation ≠ NULL ageStructEvaluation ≠ NULL usabilityEvaluation ≠ NULL
Dokonaj całościowej oceny pomysłu	costEvaluation ageStructEvaluation usabilityEvaluation	evaluation	costEvaluation ≠ NULL ageStructEvaluation ≠ NULL usabilityEvaluation ≠ NULL evaluation = NULL evaluationSummary = NULL	evaluationSummary ≠ NULL evaluation ≠ NULL
Przekaż klientowi informacje o ocenie pomysłu	evaluation	-	true	true

B.11. Model znajomości

Ostatnim elementem projektu systemu wieloagentowego jest model znajomości, który opisuje, jaka komunikacja zachodzi między każdą parą agentów. W tym modelu nie jest określane, jakie dokładnie komunikaty są wymieniane pomiędzy agentami, ani kiedy – przedstawiony jest tutaj wyłącznie sam fakt komunikacji.

Model znajomości przedstawiony został na Rys. B.11.1.



Rys. B.11.1 Schemat modelu znajomości

Na schemacie modelu znajomości (Rys. B.11.1) można zauważyć, że CustomerSystemInterface może stanowić tzw. “wąskie gardło” w projektowanym systemie. Na takie rozwiązanie zdecydowano się

jednak z myślą o skalowalności projektu. Przy rozwoju projektu będzie bowiem możliwe dodawanie agentów analizujących pomysł według dodatkowych kryteriów bez ingerencji w dotychczasowy kanał komunikacji z klientem oraz innymi ewaluatorami.

B.12. Projekt systemu

W wyniku przeprowadzonej analizy, polegającej na określeniu modelu ról (schematy w Rozdziale B.7) oraz modelu interakcji (Rys. B.8.1 oraz diagramy protokołów w Rozdziale B.8), wykonany został projekt systemu wieloagentowego, który określony jest poprzez:

- model agentów (Rys. B.9.1);
- model usług (Tabela. B.10.1);
- model znajomości (Rys. B.11.1).