Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Wieloagentowe Systemy Decyzyjne

SmartPark
Część A: Identyfikacja problemu.

Jan Dubiński | 271165
Joanna Kaleta | 271181
Aleksander Ogonowski | 267381
Maria Oniszuczuk | 271199
Kornel Szymczyk | 267778

WARSZAWA 2019

repozytorium git: https://github.com/Jot-De/WSD/

Spis treści

1. Opis problemu

Coraz większym problemem rozwijających się aglomeracji staje się nadmierne zagęszczenie ruchu samochodowego, a co się z tym wiąże - trudność w znalezieniu miejsc parkingowych. Brak łatwo dostępnej informacji o wolnych miejscach do parkowania zarówno na pobliskich parkingach, jak i w przestrzeni publicznej prowadzi do irytacji kierowców oraz marnowania ich czasu. Jedynym sposobem aby przekonać się, że wybrany parking jest w pełni obłożony, jest pojawienie się tam osobiście. Kierowcy w poszukiwaniu miejsc parkingowych krążą po ulicach dodatkowo potęgując korki uliczne. Stając w miejscach publicznych, w których parkowanie jest niedozwolone, dodatkowo utrudniają ruch pieszym i pozostałym kierowcom, jednocześnie narażając się na nieprzyjemne konsekwencje finansowe. Naszą propozycją mającą ułatwić parkowanie i zredukować ruch uliczny jest system wieloagentowy.

2. Słowny opis koncepcji systemu

Proponowanym przez nas rozwiązaniem problemu omówionego w punkcie pierwszym jest SmartPark - system wieloagentowy ułatwiający znalezienie miejsca parkingowego, który pozwoli na znaczną redukcję liczby krążących w jego poszukiwaniu samochodów.

System będzie składać się z sieci parkingów oraz aplikacji mobilnej dla kierowców pojazdów. Przy wykorzystaniu aplikacji, użytkownik będzie mógł zrealizować wyszukanie oraz nawigację do jak najbliższego parkingu po wcześniejszym określeniu preferencji dotyczącej jego typu (płatny/bezpłatny). System zapewni również równomierne rozłożenie zajętości parkingów poprzez odpowiednie propozycje. Zrównoważony powinien zostać cel kierowcy jakim jest znalezienie najbliższego parkingu i jak najbardziej równomierne zapełnianie wszystkich parkingów. Kluczowe jest tutaj unikanie sytuacji, gdy jeden parking został zapełniony,a pozostałe będą puste.

Każde miejsce parkingowe powinno zostać wyposażone w czujniki ultradźwiękowe wykrywające zajętość miejsca oraz kamerę identyfikującą konkretny pojazd. W przypadku zwolnienia miejsca powinna podnosić się blokada uniemożliwiająca wjazd niezgłoszonemu samochodowi. Każdy parking agreguje konkretną liczbę miejsc postojowych na przykład na odcinku określonej ulicy i posiada informacje o ich zajętości. Każde miejsce parkingowe po zwolnieniu wysyła adekwatną informację do parkingu, do którego przynależy, dzięki czemu może on zgłosić chęć przyjęcia następnego kierowcy. Również gdy kierowca zgłosi chęć parkowania i kierowany jest do określonego parkingu, zwiększa się wtedy zajętość i parking oczekuje na kierowcę przez pewien ustalony czas, nie przyjmując kolejnych kierowców, jeśli brak jest wolnych miejsc lub parkingi wypełniają się nierównomiernie. Dzięki temu mamy pewność, że po dojechaniu na parking znajdziemy wolne miejsce. Gdy kierowca podjedzie do miejsca na przydzielonym parkingu i zostanie poprawnie zidentyfikowany przez kamerę poprzez numer rejestracyjny, blokada opuszcza się, kierowca parkuje na miejscu postojowym, a miejsce informuje parking o jego prawidłowym zajęciu, zwiększając zajętość jego aż do chwili opuszczenia miejsca przez pojazd. Przydzielony parkingu docelowo powinien być oznaczony np korzystając z Open Street Map jako region do którego prowadzony jest kierowca.

System zakłada współistnienie dwóch typów parkingów : płatnych i bezpłatnych wyszukiwanych zgodnie z preferencjami kierowcy, jeśli jest to aktualnie możliwe. Można założyć że parkingi płatne powstaną w centach miast, ale przez konieczność uiszczenia opłaty ich obłożenie będzie równoważne z parkingami darmowymi, znajdującymi się w mniej dogodnych miejscach.

Podczas korzystania z aplikacji można wydzielić kilka typowych scenariuszy:

Scenariusz 1 - główny - wyszukanie miejsca parkingowego:

1. Użytkownik deklaruje w aplikacji chęć znalezienia miejsca parkingowego.
2. Użytkownik wybiera typ parkingu (płatny/bezpłatny).
3. System wyszukuje parking z przynajmniej jednym wolnym miejscem w okolicy

użytkownika komunikując się z pobliskimi parkingami.

4. System proponuje parking, które użytkownik może zaakceptować.
5. Użytkownik wybiera w aplikacji zasugerowane przez system parking.
6. System nawiguje kierowcę do parkingu.
7. kierowca stawia się na dowolnym miejscu parkingowym w obrębie parkingu

Scenariusz 2 - alternatywny do scenariusza 1 - wybór innego typu parkingu:

1-3. Jak w scenariuszu głównym.

4. Aplikacja wyświetla komunikat o braku wolnych miejsc parkingowych na preferowanym typie parkingu oraz propozycję najbliższego wolnego parkingu o innym typie.
5. Użytkownik wybiera w aplikacji zasugerowany przez system inny parking.
6. System nawiguje kierowcę do parkingu.
7. Kierowca stawia się na dowolnym miejscu parkingowym w obrębie parkingu

Scenariusz 3 - alternatywny do scenariusza 1 - użytkownik odrzuca proponowane parking

1-4. Jak w scenariuszu 2
5. Użytkownik nie wybiera zasugerowanego przez system parkingu.
6. Kierowca oddala się o pewną odległość
7. Powrót do punktu 1 scenariusza głównego..

Scenariusz 4 - brak wolnych miejsc parkingowych

1-3. Jak w scenariuszu głównym

4. Aplikacja wyświetla komunikat o braku wolnych miejsc parkingowych na każdym typie parkingu.
5. Kierowca oddala się o pewną odległość
6. Powrót do punktu 1 scenariusza głównego.

3. Architektura systemu

System opiera się na dwóch rodzajach agentów kierowców samochodów z zainstalowaną aplikacją parkingów agregujących miejsca parkingowe

Celem kierowców jest znalezienie parkingu jak najbliżej aktualnego miejsca, najlepiej bezpłatnego.

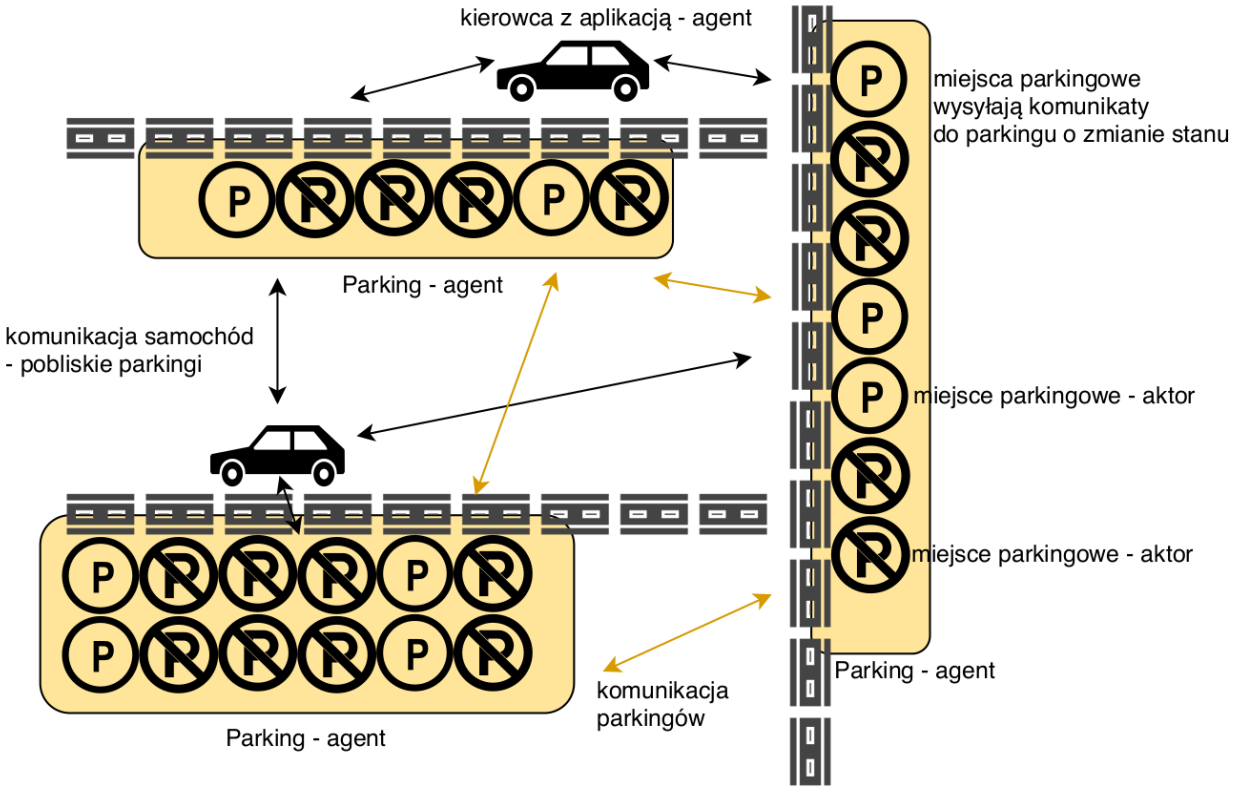
Celem parkingów jest natomiast niedopuszczenie do zapełnienia i odpowiednie balansowanie wypełnienia pobliskich parkingów poprzez wzajemną komunikację i uzgadnianie, który parking powinien aktualnie przyjąć pojazd.

System uzgadniania miejsc nie może dopuścić do sytuacji gdy przy jednoczesnym zgłoszeniu chęci parkowania przez dwa lub więcej samochody zostaną przydzielone dwa samochody na jedno wolne miejsce parkingowe.

Miejsca parkingowe są natomiast aktorami będącymi nieaktywnymi przez znaczną większość czasu. Ich zadanie polega jedynie na wysyłaniu komunikatów do parkingu, do którego przynależą. Jest to komunikat o zajęciu i zwolnieniu miejsca przez kierowcę. Na tej podstawie parking wie ile samochodów znajduje się na nim, a ile może przyjąć. Od liczby samochodów, które parking może przyjąć należy również samochody, które zgłosiły chęć parkowania, ale jeszcze nie dojechały. Nigdy jednak nie zostaje przyjęte więcej samochodów niż liczba wolnych miejsc.

Parkingi komunikują się z kierowcami i między sobą prowadząc do optymalnego rozłożenia samochodów.

Kierowcy komunikują się z parkingami zgłaszając chęć parkowania lub odrzucając sugerowany parking. Generalnie nie ma potrzeby, aby samochody prowadziły komunikację między sobą.



Rysunek 1. Architektura systemu.

4. Projekt systemu wieloagentowego

4.1. Identyfikacja ról

1. Client
    - zgłasza chęć zaparkowania w pobliżu udostępnionej lokalizacji
    - akceptuje lub odrzuca propozycje wskazanego miejsca
2. Parking mapper
    - mapuje istniejące parkingi
3. Parking Approacher
    - zbliża się do parkingu i wysyła komunikat z aktualną lokalizacją
    - może zrezygnować
4. Parking Manager
    - kontrola dostępności miejsc na parkingu (stanu wewnętrznego),
    - przydziela miejsce parkingowe na parkingu, jeśli takie jest dostępne
5. Car Tracker
    - monitoruje samochód który zarezerwował miejsce (żeby zamknąć rezerwacje jeśli nie będzie się zbliżał lub nie przyjedzie przez 10 min)

4.2. Model ról

4.2.1. Parking Manager

Aktywności:

- CheckPlacesAvaliablility - sprawdzanie liczby wolnych miejsc na parkingu
- MakeReservation - wykonanie rezerwacji
- CancelReservation - zwolnienie rezerwację

Protokoły:

- MapParkings
- PlaceReservations
- FreePlace

Tabela 1. Rola Parking Manager.

Role schema: Parking Manager
Description: • Zwraca informacje o lokalizacji parkingu • Kontroluje dostępność miejsc na parkingu • Przydziela miejsce parkingowe na danym parkingu, jeśli takie jest dostępne
Protocols and Activities: CheckPlacesAvailability, MakeReservation, CancelReservation, MapParkings(SendCoordinates), PlaceReservations(SendAvailablePlaceInfo), FreePlace(ConfirmFreedPlace)
Permissions: reads: generates: parking_location, free_place_count
Responsibilities: Liveness: PARKING MANAGER = [MapParkings]. (CheckPlacesAvailability. [SendAvailablePlaceInfo]. [MakeReservation]. [CancelReservation]. ConfirmFreedPlace)* MAPPARKINGS = SendCoordinates Safety: true

4.2.2. Car Tracker

Aktywności:

- ShouldMaintainReservation - podejmowanie decyzji o utrzymaniu rezerwacji (gdy Approacher się oddala/nie pojawia, podniesienie flagi ApproacherIsNotComing)

Protokoły:

- TrackCarWithReservation
- ReservationCancellation
- FreePlace

Tabela 2. Rola Car Tracker.

Role schema: Car Tracker
Description: • monitoruje samochód, który zarezerwował miejsce i podejmuje decyzję, czy należy rezerwacje zamknąć (jeśli np. samochód nie przybywa w ciągu kilku minut lub oddala się przez dłuższy czas)
Protocols and Activities: ShouldMaintainReservation, TrackCarWithReservation(SubscribeForClientLocation), ReservationCancelation(ConfirmCancellation), FreePlace(RequestSetPlaceFree)
Permissions: reads: approacher_info, car_location, reservation_cancellation, IsApproacher generates: ApproacherIsNotComing, ApproacherCancelledReservation
Responsibilities: Liveness: AR TRACKER = TrackCarWithReservation.[ReservationCancellation].FreePlace TRACKCARWITHRESERVATION = SubscribeForClientLocation.ShouldMaintainReservation RESERVATIONCANCELLATION = ConfirmCancellation FREEPLACE = RequestSetPlaceFree Safety: IsApproacher = True

4.2.3. Parking Mapper

Aktywności:

- ComputeParkingList
- RunUpdate

Protokoły:

- MapParking

Tabela 3. Rola Parking Mapper.

Role schema: ParkingMapper
Description: • mapuje istniejące parkingi
Protocols and Activities: RunUpdate, MapParking(UpdateParkingsCoordiantes)
Permissions: reads: parking_location generates: parking_list
Responsibilities: Liveness: PARKINGMAPPER = (RunUpdate.MapParking.ComputeParkingList) MAPPARKING = UpdateParkingsCooridnates Safety: adsa

4.2.4. Client

Aktywności:

- GetMyLocation - lokalizuje się
- ComputeParkingList - wybiera kilka parkingów z wolnymi miejscami w najbliższej

Protokoły:

- PlaceReservation

Tabela 4. Rola Client.

Role schema: Client
Description: • zgłasza chęć zaparkowania w pobliżu udostępnionej lokalizacji • wybiera jeden z zaproponowanych parkingów lub odrzuca propozycje
Protocols and Activities: GetMyLocation, PlaceReservation(CallForParkingOffers, AcceptParkingOffer)
Permissions: reads: parking_list generates: IsApproacher, car_location, selected_parking_list
Responsibilities: Liveness: Client = (GetMyLocation.CallForParkingOffers.[AcceptParkingOffer])* Safety: isApproacher = false

4.2.5. Approacher

Aktywności: brak

Protokoły:

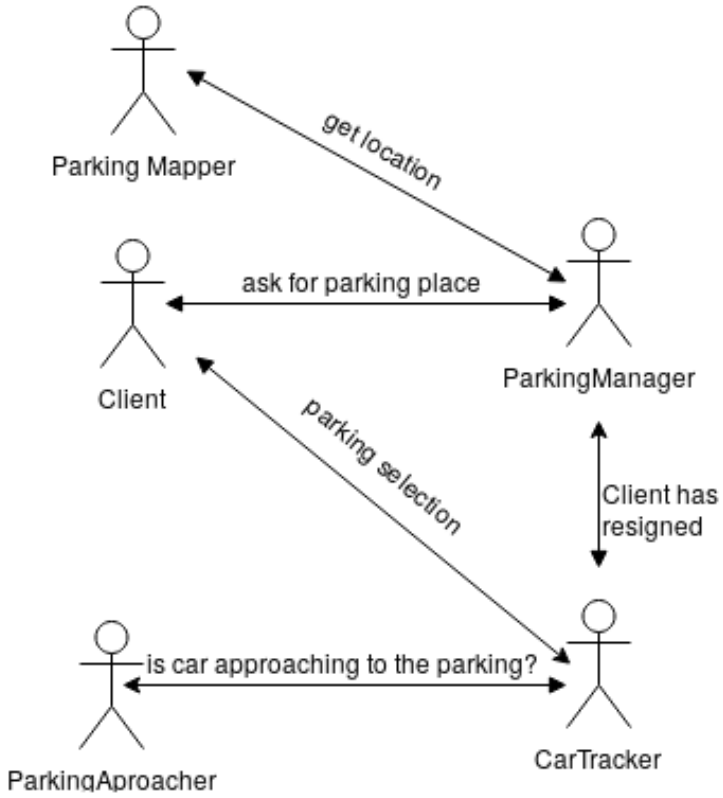
- TrackCarWithReservation
- Reservation Cancellation
- FreePlace

Tabela 5. Rola Approacher.

Role schema: Approacher
Description: • ma zarezerwowane miejsce na parkingu i wysyła informację o swoim aktualnym położeniu (ciągle) • może zrezygnować z zarezerwowanego miejsca
Protocols and Activities: TrackCarWithReservation(SendReservationInfo, SendApproacherLocation), Reservation Cancellation(CancelClientReservation)
Permissions: reads: IsApproacher generates: reservation_cancellation, car_location
Responsibilities: Liveness: APPROACHER = TrackCarWithReservation.[ReservationCancellation] TRACKCARWITHRESERVATION = SendReservationInfo.(SendApproacherLocation) RESERVATIONCANCELLATION = CancelClientReservation Safety: isApproacher = true

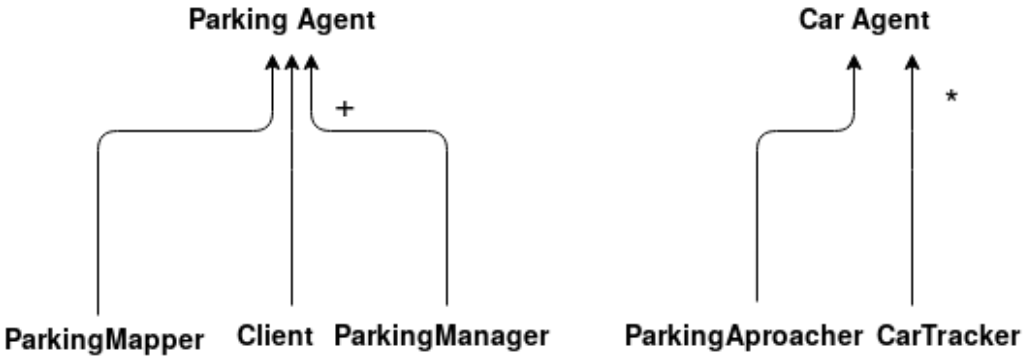
4.3. Model interakcji

Na rysunku 2 przedstawiającym model interakcji widać, że nie posiada on żadnej roli centralnej komunikującej się bezpośrednio ze wszystkimi innymi rolami. System posiada znaczne rozproszenie ról, biorąc pod uwagę występowanie tylko dwóch typów agentów wcielających się w różne role. Wszystkie agenty wchodzą między sobą w interakcje. Każdy z agentów powinien zostać więc poprawnie zaimplementowany w celu utrzymania systemu.



Rysunek 2. Schemat interakcji.

4.4. Model agentów



Rysunek 3. Model agentów.

Na podstawie powyższych schematów (3) modeli agentów można stwierdzić, że oba rodzaje agentów występujących w systemie są złożone w kontekście zaimplementowanych w nich ról. ParkingAgent posiada nieco większe skomplikowanie z uwagi na bardziej odpowiedzialną rolę jaką pełni w systemie, ponieważ to agenty-parkingi zarządzają agentami-pojazdami jednocześnie spełniając ich prośby. System nie może istnieć bez żadnej instancji ParkingAgent, ponieważ CarAgent w roli klienta nie mógłby wchodzić w interakcje i w rezultacie zaparkować. Natomiast można wyobrazić sobie istnienie systemu z instancjami ParkingAgent'a bez drugiego rodzaju agentów występujących w systemie.

4.4.1. Map Parkings

UpdateParkingsCordinates (INFORM REF)	
ParkingMapper	**ParkingManager**
Wysyła zapytania do wszystkich parkingów i uaktualnia ich położenie z prośbą o informacje o ich lokalizacji	

SendCoordinates (INFORM)		
ParkingManager	**ParkingMapper**	
wysyła koordynaty parkingu		parking_location

4.4.2. PlaceReservation

CallForParkingOffers (CallForProporsal)		
Client	**Parking Manager**	parking_list
Wysyła prośbę o rezerwację miejsca postojowego do parkingów z wolnymi miejscami w określonej odległości		

SendAvailablePlaceInfo (Propose)	
Parking Manager	**Client**
wysyła Clientowi informacje o dostępności miejsca, jeżeli takie posiada	

AcceptParkingOffer (AcceptProposal)	
Client	**Parking Manager**
otrzymuje odpowiedź Clienta czy rezerwuje wskazane miejsce	

4.4.3. TrackCarWithReservation

SendReservationInfo (INFORM)		
Approacher	**CarTracker**	parking_list
przekazuje dane rezerwacji do Car Trackera po akceptowaniu wskazanego miejsce		

↓

SubscribeForClientLocation (Subscribe)		
CarTracker	**Approacher**	
prosi Approachera o cykliczne wysyłanie swojej lokalizacji		

↓

SendApproacherLocation (inform)		
Approacher	**CarTracker**	
wysyła cyklicznie dane o swojej lokalizacji do Car Trackera		ApproacherIsNotComing

4.4.4. ReservationCancellation

CancelClientReservation (CANCEL)		
Approacher	**CarTracker**	parking_list
wysyła informacje o rezygnacji z miejsca parkingowego		AproacherCancelledReservation

↓

ConfirmCancellation (Confirm)		
CarTracker	**Approacher**	
potwierdza otrzymanie informacji o rezygnacji		AproacherCancelledReservation

4.4.5. FreePlace

RequestSetPlaceFree (Request)		
CarTracker	**Parking Controller**	parking_list
wysyła żądanie usunięcia rezerwacji miejsca do Parking Controllera		

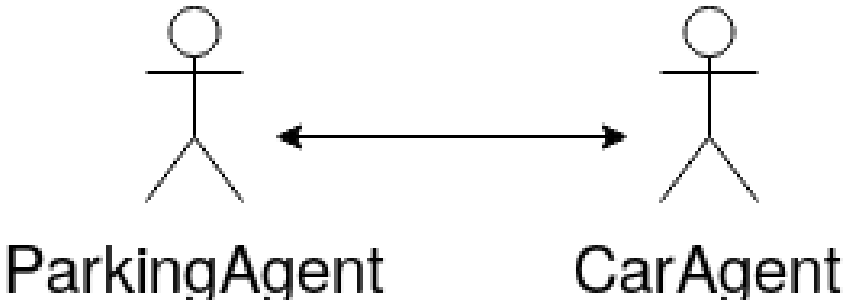
ConfirmFreedPlace (Confirm)		
Parking Controller	**CarTracker**	
potwierdza otrzymanie informacji o rezygnacji		AproacherCancelledReservation

4.5. Model usług

Tabela 6. Model usług.

Usługa	Wejścia	Wyjścia	Warunki wstępne	Warunki końcowe
Tworzenie listy lokalizacji istniejących parkingów	-	parking_list	true	parking_list =/= NULL
Tworzenie listy proponowanych parkingów z wolnymi miejscami w okolicy	car_location	selected_parking_list	parking_list =/= NULL, car_location =/= NULL	length(sort(selected_parking_list)) == length(distance(car_location, parking_location) > set_distance) and length(sort(selected_parking_list)) <=5 (wybór do 5 najbliższych parkingów z okolicy lub mniej jeśli ich liczba< 5)
Znalezienie wolnego miejsca	car_location	reservation_info	parking_list =/= NULL	reservation_info =/= NULL
Śledzenie położenia samochodu	car_location reservation_info	ApproacherIsNotComing	reservation_info =/= NULL	ApproacherIsNotComing = True OR IsApproacher = False
Odwoływanie zarezerwowanego miejsca	reservation_info	ApproacherCancelledReservation	reservation_info =/= NULL	ApproacherCancelledReservetation = True

4.6. Model znajomości



Rysunek 4. Model znajomości.

Model znajomości agentów w systemie (Rys. 4) jest nieskomplikowany z uwagi na małą liczbę agentów w nim występujących. Wszystkie rodzaje agentów komunikują się między sobą. Nadmienić należy że architektura systemu została zaprojektowana w ten sposób, że nie wymaga komunikacji pomiędzy różnymi instancjami tego samego rodzaju agenta. W ParkingAgent następuje komunikacja pomiędzy rolami, w które wciela się ta sama instancja tego rodzaju genta.

Spis rysunków

Spis tabel