

Politechnika Warszawska

W Y D Z I A Ł E L E K T R O N I K I
I T E C H N I K I N F O R M A C Y J N Y C H



Wieloagentowe Systemy Decyzyjne

SmartPark

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

WARSZAWA 2019

Spis treści

1. Opis problemu	4
2. Słowny opis koncepcji systemu	5
3. Architektura systemu	7
4. Projekt systemu wieloagentowego	9
4.1. Identyfikacja ról	9
4.2. Model ról	10
4.2.1. Parking Manager	10
4.2.2. Car Tracker	11
4.2.3. Parking Mapper	12
4.2.4. Client	13
4.2.5. Approacher	14
4.3. Model interakcji	15
4.4. Definicje protokołów	16
4.4.1. Map Parkings	16
4.4.2. PlaceReservation	17
4.4.3. TrackReservation	18
4.4.4. TrackCar	18
4.4.5. ReservationCancellation	18
4.5. Model agentów	19
4.6. Model usług	20
4.7. Model znajomości	21
5. Opis implementacji systemu	22
5.1. Framework	22
5.2. Sposób implementacji agentów	22
5.3. Sposób implementacji komunikatów	24
5.3.1. Zastosowane performatywne	24
5.3.2. Protokoły komunikacyjne	24
5.3.3. Zastosowane języki treści	24
5.4. Wykorzystane standardy	24
5.5. Algorytmy	24
5.6. Wizualizacja	26
5.7. Napotkane problemy i zmiany w kolejnej iteracji	27
Spis rysunków	28
Spis tabel	28

1. Opis problemu

Coraz większym problemem rozwijających się aglomeracji staje się nadmierne zaspłyczenie 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

Propozowanym przez nas rozwiązaniem problemu omówionego w punkcie pierwszym jest 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 wybranego **pobliskiego parkingu po wcześniejszym zgłoszeniu swojej lokalizacji**.

System po zgłoszeniu chęci zaparkowania proponuje pięć pobliskich parkingów z wolnymi miejscami, spośród których kierowca może wybrać ten do którego chce się udać. Kluczowe w systemie jest pozwolenie kierowcy na znalezienie jak najbliższego odpowiadającego mu parkingu z przynajmniej jednym wolnym miejscem. Celem parkingu jest natomiast pozbycie się pustych miejsc, czyli jego zapełnienie.

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 **na dany parking** wsamochodowi. Każdy parking agreguje konkretną liczbę miejsc postojowych na przykład na odcinku konkretnej 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. Gdy kierowca zadeklaruje chęć zaparkowania, po czym jest nawigowany do określonego parkingu, powoduje to zwiększenie się jego zajętości. Parking oczekuje na kierowcę przez pewien ustalony czas, nie przyjmując kolejnych deklarujących chęć pojazdów, jeżeli nie ma w swoim obrębie wolnych miejsc. Dzięki takiemu rozwiązaniu mamy pewność, że po dojechaniu na parking znajdziemy puste miejsce. Gdy kierowca podjadzie 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 parking docelowo powinien być oznaczony np. przy wykorzystaniu Open Street Map jako region do którego prowadzony jest kierowca.

2. Słowny opis koncepcji systemu

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. System wyszukuje parki z przynajmniej jednym wolnym miejscem w okolicy użytkownika komunikując się z pobliskimi parkingami.

3. System proponuje parki, które użytkownik może zaakceptować.

4. Użytkownik wybiera w aplikacji jeden proponowanych przez system parking.

5. System nawiguje kierowcę do parkingu.

6. Kierowca stawia się na dowolnym miejscu parkingowym w obrębie parkingu

Scenariusz 2 - użytkownik oddala się od parkingu

1-5. Jak w scenariuszu głównym

6. Kierowca nie kieruje się na parking (jego pozycja oddala się).

7. Parking sprawdza czy proces postępuje

8. Kierowca nadal się oddala

9. Parking dokonuje zwolnienia miejsca parkingowego

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 - użytkownik rezygnuje z rezerwacji

1-5. Jak w scenariuszu głównym

6. Kierowca rezygnuje z miejsca parkingowego

7. Parking dokonuje zwolnienia miejsca parkingowego

Scenariusz 5 - 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

Celem nadzorowanego systemu jest maksymalne zapełnienie parkingów.

System opiera się na dwóch rodzajach agentów

- kierowców samochodów z zainstalowaną aplikacją
- parkingów agregujących miejsca parkingowe

Kierowcy dążą do znalezienia parkingu jak najbliżej aktualnego miejsca. Parkingi dążą do maksymalnego zapełnienia się. System wysyła kierowcy propozycję miejsca parkingowego na parkingu, który znajduje się najbliżej jego aktualnego położenia i który posiada wolne miejsca. Po zaakceptowaniu przesłanej propozycji następuje zarezerwowanie miejsca postojowego dla klienta na podstawie jego unikalnego identyfikatora.

System wymaga, aby samochód cyklicznie wysyłał zapytania do parkingów otrzymując w odpowiedzi ich położenie.

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.

Parking nadzoruje, czy kierowca nie oddala się od niego po zgłoszeniu chęci zaparkowania przez dłuższy czas lub nie wysłał zgłoszenia dotyczącego anulowania rezerwacji. W przypadku wystąpienia jednego z tych zdarzeń parking odwołuje rezerwację zwalniając przydzielone dla samochodu miejsce.

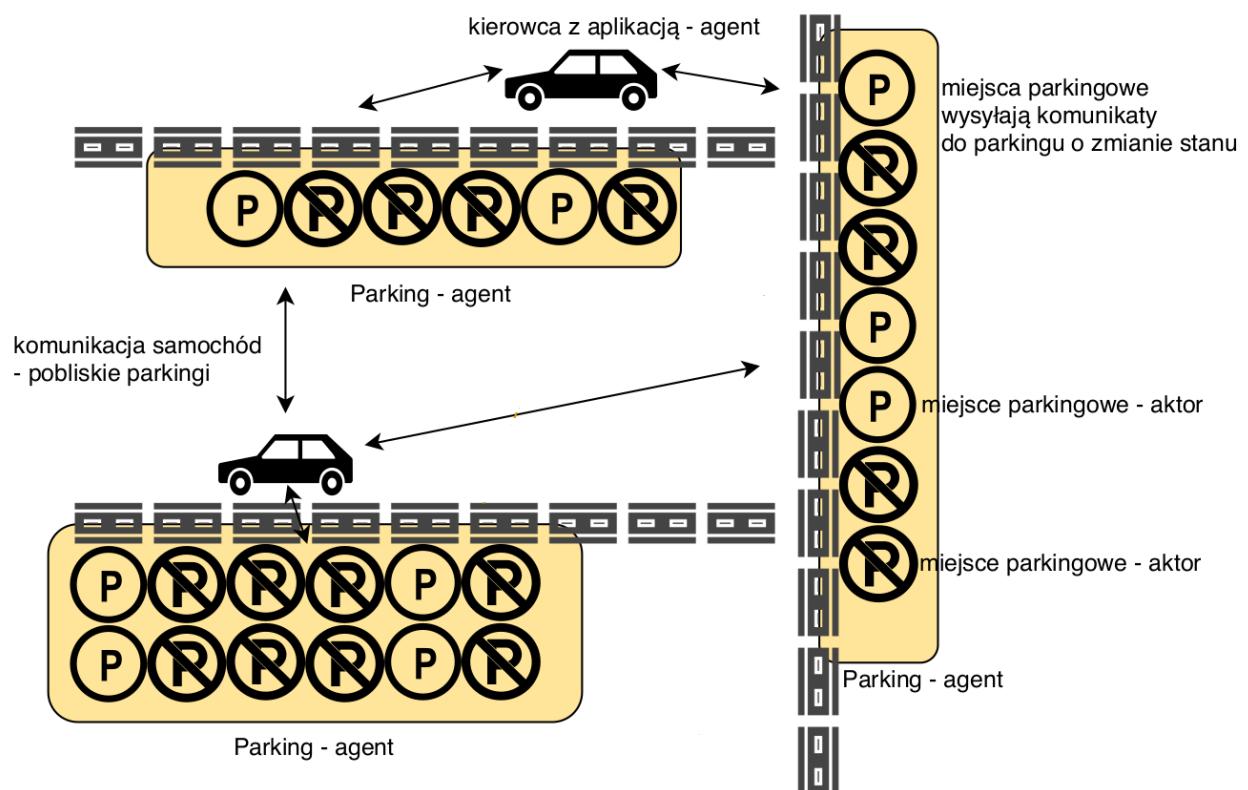
Miejsca parkingowe są natomiast aktorami będącymi nieaktywnymi przez większość czasu. Ich zadanie polega jedynie na wysyłaniu komunikatów do parkingu, do którego przynależą. Jest to komunikat o zajęciu i zwolnieniu miejsca przez kierowcę. Na tej podstawie parking wie ile samochodów znajduje się na nim,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

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

Naszimi propozycjami rozwoju systemu w kolejnych etapach jest:

- dodanie nadzoru nad czujnikami - sygnalizowanie awarii i prowadzenie napraw
- udoskonalenie algorytmu monitorującego zachowanie kierowcy tak, aby uwzględnił on korki, objazdy, wypadki itd.

3. Architektura systemu



Rysunek 1. Architektura systemu.

4. Projekt systemu wieloagentowego

System SmartPark został zaprojektowany zgodnie z wytycznymi metodologii GAIA.

4.1. Identyfikacja ról

Role agenta CarAgent:

1. Client
 - zgłasza chęć zaparkowania w pobliżu udostępnionej lokalizacji
 - akceptuje lub odrzuca propozycje wskazanego miejsca
2. Parking mapper
 - mapuje istniejące parkingi
3. Approacher
 - zbliża się do parkingu i wysyła komunikat z aktualną lokalizacją
 - może zrezygnować z zarezerwowanego miejsca

Role agenta ParkingAgent:

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. Projekt systemu wieloagentowego

4.2. Model ról

4.2.1. Parking Manager

Aktywności:

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

Protokoły:

- MapParkings
- PlaceReservations
- FreePlace

Tabela 1. Rola Parking Manager.

Role schema: Parking Manager
Description: <ul style="list-style-type: none">• Zwraca informacje o lokalizacji parkingu• Kontroluje dostępność miejsc na parkingu• Przydziela miejsce parkingowe na danym parkingu, jeśli takie jest dostępne
Protocols and Activities: <u>CheckPlacesAvailability</u> , <u>MakeReservation</u> , <u>CancelReservation</u> , MapParkings(SendCoordinates), PlaceReservations(SendAvailablePlaceInfo), FreePlace(ConfirmFreedPlace)
Permissions: reads: sensors_data generates: parking_location, free_place_count
Responsibilities: Liveness: PARKING MANAGER = [MapParkings]. (<u>CheckPlacesAvailability</u> . [SendAvailablePlaceInfo]. [<u>MakeReservation</u>]. [<u>CancelReservation</u>]. ConfirmFreedPlace)* MAPPARKINGS = SendCoordinates Safety: true

4.2.2. Car Tracker

Aktywności:

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

Protokoły:

- TrackCar
- TrackReservation
- ReservationCancellation
- FreePlace

Tabela 2. Rola Car Tracker.

Role schema: Car Tracker
Description: <ul style="list-style-type: none"> • 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: <u>ShouldMaintainReservation</u> , TrackReservation(ConfirmReservationInfo) TrackCar(SubscribeForClientLocation), ReservationCancellation(ConfirmCancellation), FreePlace(RequestSetPlaceFree)
Permissions: reads: approacher_info, car_location, reservation_cancellation, IsApproacher generates: ApproacherIsNotComing, ApproacherCancelledReservation
Responsibilities: Liveness: CAR TRACKER = TrackReservation. TrackCar. [ReservationCancellation]. FreePlace TRACKRESERVATION = ConfirmReservationInfo TRACKCAR = SubscribeForClientLocation. <u>ShouldMaintainReservation</u> RESERVATIONCANCELLATION = ConfirmCancellation FREEPLACE = RequestSetPlaceFree Safety: IsApproacher = True

4. Projekt systemu wieloagentowego

4.2.3. Parking Mapper

Aktywności:

- CreateParkingDict - zapamiętuje lokalizacje opowiadające parkingom
- RunUpdate - włącza skanowanie parkingów w celu pobrania lokalizacji

Protokoły:

- MapParking

Tabela 3. Rola Parking Mapper.

Role schema: ParkingMapper
Description: <ul style="list-style-type: none">• mapuje istniejące parkingi
Protocols and Activities: RunUpdate, MapParking(UpdateListOfParkings)
Permissions: reads: parking_location generates: parking_list
Responsibilities: Liveness: PARKINGMAPPER = (RunUpdate.MapParking.CreateParkingDict)ω MAPPARKING = UpdateListOfParkings Safety: true

4.2.4. Client

Aktywności:

- GetMyLocation - lokalizuje się
- ComputeParkingList - wybiera kilka parkingów w najbliższej odległości z wolnymi miejscami postojowymi, z którymi będzie się komunikować

Protokoły:

- PlaceReservation
- TrackReservation

Tabela 4. Rola Client.

Role schema: Client
Description: <ul style="list-style-type: none"> • zgłasza chęć zaparkowania w pobliżu udostępnionej lokalizacji • wybiera jeden z zaproponowanych parkingów lub odrzuca propozycje
Protocols and Activities: GetMyLocation, PlaceReservation(CallForParkingOffers, AcceptParkingOffer), TrackReservation(SendReservationInfo), ComputeParkingList
Permissions: reads: parking_list generates: IsApproacher, car_location, selected_parking_list
Responsibilities: Liveness: CLIENT = (GetMyLocation, ComputeParkingList, CallForParkingOffers, [AcceptParkingOffer], TrackReservation)* TRACKRESERVATION = SendReservationInfo Safety: isApproacher = false

4. Projekt systemu wieloagentowego

4.2.5. Approacher

Aktywności:

- ShouldParkingApproaching - podejmowanie decyzji czy chce zmierzać do parkingu i ewentualna jazda w jego stronę

Protokoły:

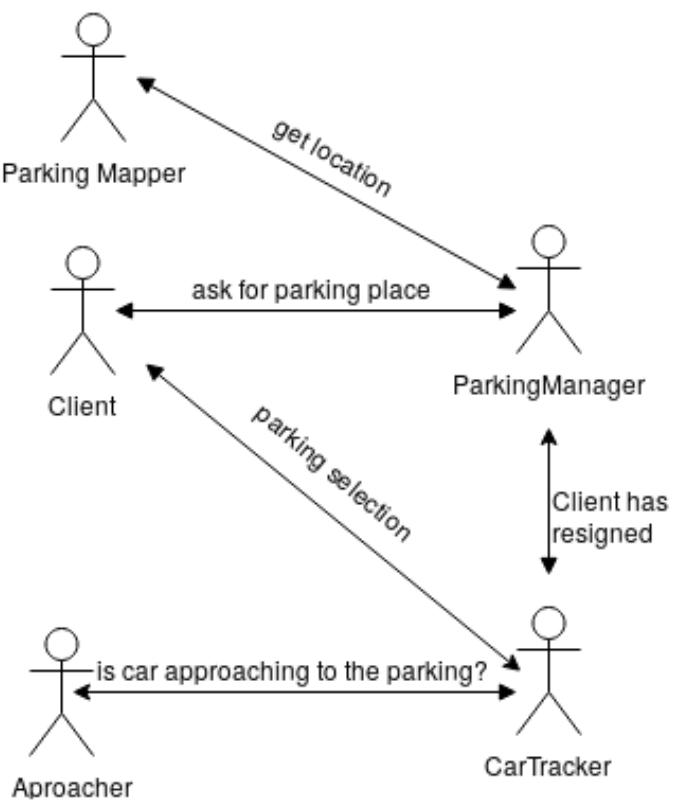
- TrackCar
- Reservation Cancellation
- FreePlace

Tabela 5. Rola Approacher.

Role schema: Approacher
Description: <ul style="list-style-type: none">• ma zarezerwowane miejsce na parkingu i wysyła informację o swoim aktualnym położeniu (ciągle)• może zrezygnować z zarezerwowanego miejsca
Protocols and Activities: TrackCar(SendApproacherLocation), Reservation Cancellation(CancelClientReservation)
Permissions: reads: IsApproacher generates: reservation_cancellation, car_location
Responsibilities: Liveness: APPROACHER = TrackCar TRACKCAR = SendReservationInfo. (SendApproacherLocation ReservationCancellation) 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. Model interakcji.

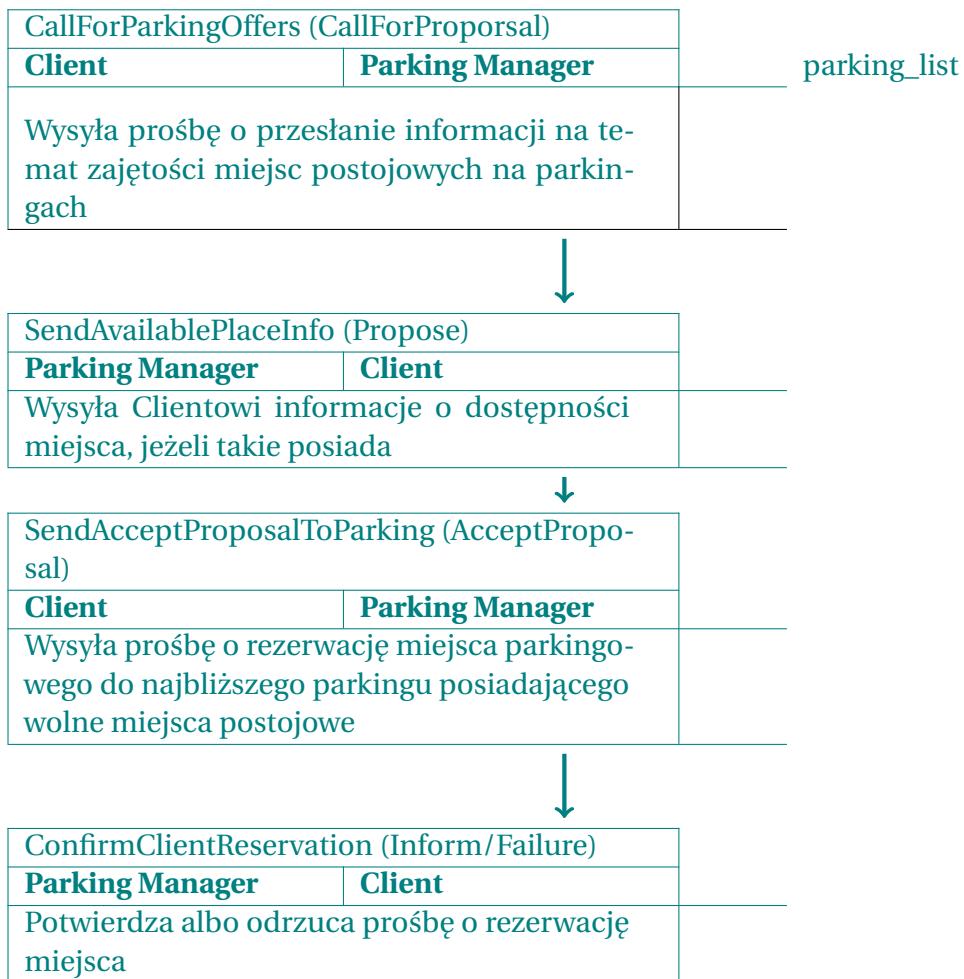
4. Projekt systemu wieloagentowego

4.4. Definicje protokołów

4.4.1. Map Parkings

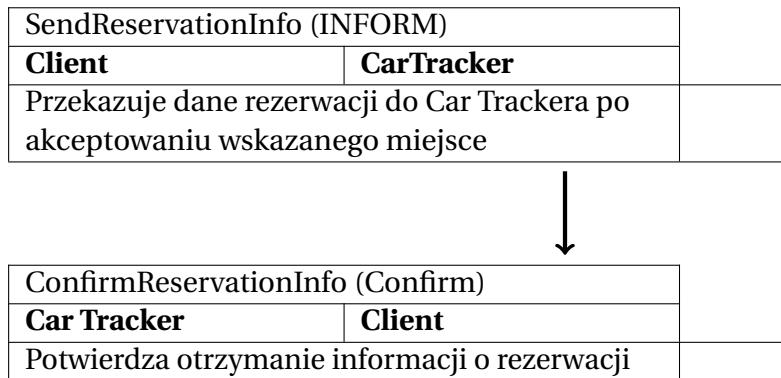


4.4.2. PlaceReservation

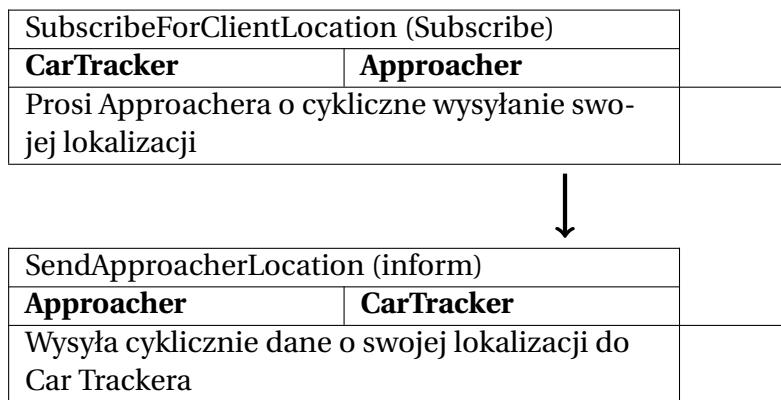


4. Projekt systemu wieloagentowego

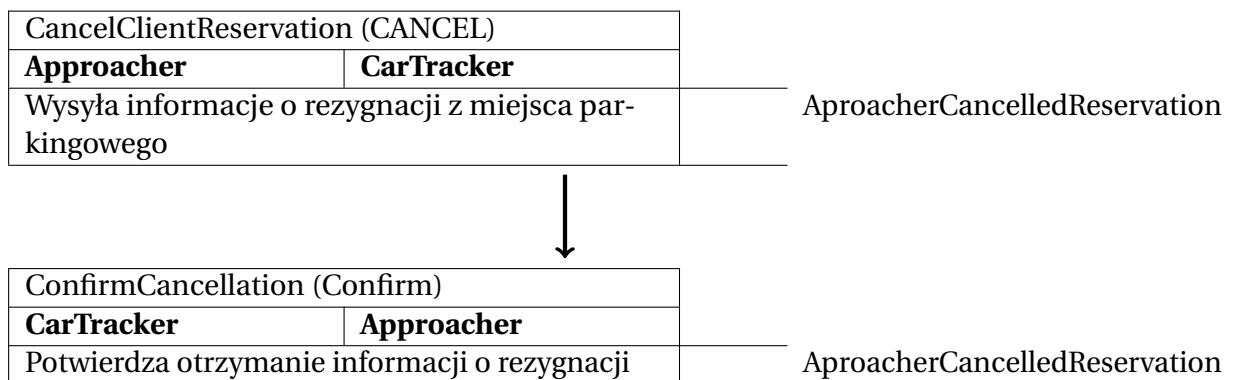
4.4.3. TrackReservation



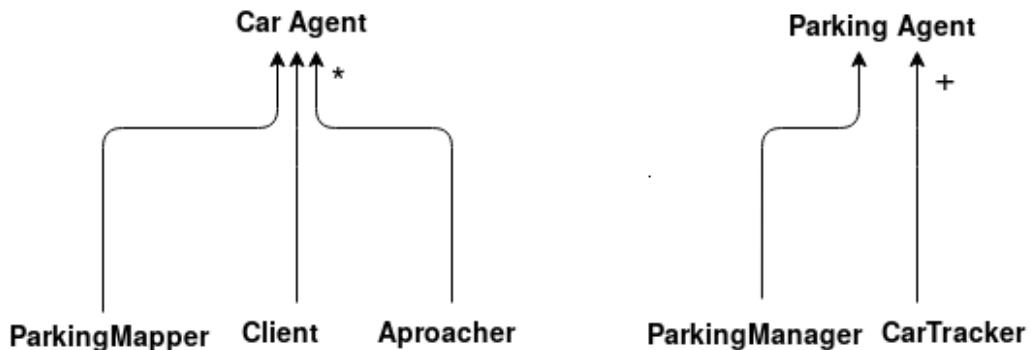
4.4.4. TrackCar



4.4.5. ReservationCancellation



4.5. Model agentów



Rysunek 3. Model agentów.

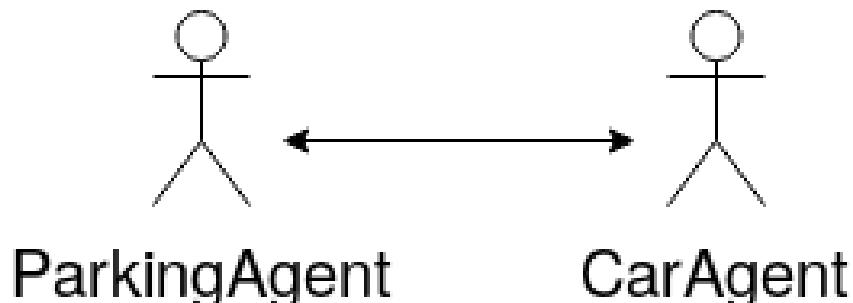
Na podstawie powyższych schematów (Rys. 3) modeli agentów można stwierdzić, że oba rodzaje agentów występujących w systemie są złożone w kontekście zaimplementowanych w nich ról. **ParkingAgent** posiada nieco większe skomplikowanie z uwagi na bardziej odpowiedzialną rolę jaką pełni w systemie, ponieważ 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 zinstancjami **ParkingAgent'a** bez drugiego rodzaju agentów występujących w systemie.

4.6. 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 propozowanych 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	selected_parking_list	reservation_info	parking_list =/= NULL	reservation_info =/= NULL
Śledzenie położenia samochodu	car_location reservation_info	ApproacherIsNotComing	reservation_info =/= NULL	Approacher IsNotComing = True OR IsApproacher = False
Odwoływanie zarezerwowanego miejsca	reservation_info	ApproacherCancelled-Reservation	reservation_info =/= NULL	ApproacherCancelled-Reservation = True

4.7. 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 przypadku agenta ParkingAgent następuje komunikacja pomiędzy rolami, w które wciela się ta sama instancja tego rodzaju agenta.

5. Opis implementacji systemu

5.1. Framework

Framework Narzędziem użytym w realizacji projektu jest JADE 4.5.0 (Java Agent Development Framework). Jest to popularny framework przeznaczony do implementowania systemów wieloagentowych w pełni zaimplementowany w języku Java. Wyбралиśmy JADE, ponieważ jak można przeczytać w dokumentacji:

- w JADE wdrożony został pełny model komunikacji FIPA, a jego komponenty zostały wyraźnie wyróżnione i w pełni zintegrowane: protokoły interakcji, koperta, ACL, języki treści, schematy kodowania, ontologie i protokoły transportowe.
- architektura komunikacji oferuje elastyczne i wydajne przesyłanie wiadomości, gdzie JADE tworzy i zarządza kolejką przychodzących wiadomości ACL. Agenci mogą uzyskać dostęp do swojej kolejki za pomocą kombinacji kilku trybów: blokowania, odpytywania, limitu czasu i dopasowywania wzorców.
- Dodatkowo przydatną funkcjonalnością jest możliwość sterowania konfiguracją za pomocą interfejsu GUI.

5.2. Sposób implementacji agentów

Utworzyliśmy dwie klasy agentów CarAgent oraz ParkingAgent. Każdy agent dziedziczy funkcjonalność po klasie jade.core.Agent. Agenty wykorzystują klasę jade.core.behaviours. Behaviour do implementacji aktywności oraz protokołów zdefiniowanych dla ról.

CarAgent - Agent samochód, który może się przemieszczać. Dla celów testowych przyjęliśmy, iż na początku swojego istnienia agent ten losuje swoją pozycję początkową na mapie i dodaje zachowania UpdateListOfParkings, CallForParkingOffers, SendReservationInfo oraz ReservationCancellation, ListenForLocationCancelSubscriptionFromCarTracker.

UpdateListOfParkings - służy do wysyłania zapytania do wszystkich parkingów z prośbą o informacje o ich lokalizacji i uaktualnienia ich położenia. Protokół ten jest aktywowany na początku istnienia agenta samochodu. Agent zapisuje położenia wszystkich parkingów w HashMapie, gdzie kluczem jest AID parkingu, a wartością jest lokalizacja parkingu. Lokalizacje wszystkich parkingów są niezbędne dla CarAgent, ponieważ chce on wiedzieć, który parking dysponujący wolnym miejscem znajduje się najbliżej.

CallForParkingOffers - służy do zapoczątkowania konwersacji z najbliższymi parkin-gami w celu zarezerwowania miejsca. Agent samochód wysyła wiadomość typu CFP i otrzymuje od parkingów posiadających wolne miejsce odpowiedź typu PROPOSE lub REFUSE. Następnie typowany jest najbliższy parking z wolnym miejscem i podejmowana jest próba dokonania rezerwacji miejsca (ACCEPT PROPOSAL). Jeżeli CarAgent otrzyma odpowiedź pozytywną (INFORM) od ParkingAgent, to dany parking jest zapisany w pa-

mięci jako cel podróży, jeżeli jednak otrzymamy odpowiedź negatywną (FAILURE) to ponownie rozpoczynamy protokół CallForParkingOffers.

SendReservationInfo - służy do wysłania wiadomości od CarAgent (INFORM) do ParkingAgent o rezerwacji zrobionej dla konkretnego agenta. Po otrzymaniu informacji ParkingAgent wysyła potwierdzenie CONFIRM.

CancelClientReservation - służy do wysłania wiadomości o rezygnacji z zarezerwowanego miejsca (CANCEL) do ParkingAgent, będącego do tej pory celem podróży agenta-samochodu (CarAgent). Po otrzymaniu takiego żądania ParkingAgent zwalnia miejsce postojowe, a następnie wysyła potwierdzenie wykonania akcji (CONFIRM).

ListenForLocationCancelSubscriptionFromCarTracker - służy do nasłuchiwanego CarTracker chce przestać subskrybować wiadomości, które są wysyłane w protokole SendReservationInfo.

ParkingAgent - agent parking posiadający miejsca parkingowe udostępniane pojazdom. Głównym zadaniem agenta jest oferowanie wolnych miejsc parkingowych oraz śledzenie aktualnej pozycji samochodów posiadających rezerwację. Agent ten wykorzystuje zachowania:

SendCoordinates - służy do wysłania informacji o swojej pozycji. Zachowanie to jest cykliczne i wykonywane po otrzymaniu wiadomości od agenta samochodu typu INFORM_REF.

SendAvailablePlaceInfo - zachowanie to jest cykliczne, wykonywane po przyjęciu wiadomości zawierającej performatywę CFP od CarAgent. Jeżeli agent posiada wolne miejsca parkingowe to wysyła wiadomość typu PROPOSE w odpowiedzi. W przypadku kiedy agent nie posiada wolnego miejsca następuje wysłanie odpowiedzi zawierającej performatywę REFUSE.

ConfirmReservation - zachowanie cykliczne; po otrzymaniu od CarAgent wiadomości zawierającej performatywę ACCEPT_PROPOSAL, wyrażającej chęć zarezerwowania miejsca na danym parkingu, ParkingAgent zmienia stan miejsca z "wolny" na "zajęty" i odsyła wiadomość o pozytywnym przebiegu procesu (INFORM). Jeśli dane miejsce zostanie przed otrzymaniem wiadomości zawierającej ACCEPT_PROPOSAL zajęte przez innego CarAgent, zostaje wysłana wiadomość informująca o niepowodzeniu rezerwacji (FAILURE).

ConfirmCancellation - zachowanie cykliczne, po otrzymaniu wiadomości od CarAgent wyrażającej żądanie usunięcia rezerwacji miejsca parkingowego (CANCEL) zmienia stan miejsca z "zajęty" na "wolny", a następnie wysyła odpowiedź o powodzeniu akcji (CONFIRM).

getReservationInfo - zachowanie cykliczne; służy do odbierania wiadomości od CarAgent o dokonaniu rezerwacji dla tego CarAgenta na parkingu, którym jest adresat wiadomości - ParkingAgent. Po otrzymaniu informacji ParkingAgent przesyła potwierdzenie CONFIRM.

5. Opis implementacji systemu

5.3. Sposób implementacji komunikatów

5.3.1. Zastosowane performatywy

1. INFORM_REF
2. INFORM
3. FAILURE
4. CFP
5. PROPOSE
6. ACCEPT_PROPOSAL
7. CANCEL
8. CONFIRM
9. REFUSE

5.3.2. Protokoły komunikacyjne

W trakcie dotychczasowej implementacji został użyty tylko jeden typowy protokół komunikacyjny zdefiniowany przez FIPA i jest nim Subscribe Interaction Protocol, który został zastosowany do śledzenia Car Agents przez rolę CarTracker realizowaną przez Parking Agent. Pozostała wymiana komunikatów pomiędzy agentami została zebrana w indywidualnie zdefiniowanych dla naszego systemu protokołach stworzonych poprzez rozszerzenie klasy Behaviours, CyclicBehaviours oraz TickerBehaviours.

5.3.3. Zastosowane języki treści

Zastosowanym językiem treści jest FIPA Semantic Language (SL), który jest standar-dowo zaimplementowany we framework'u JADE.

5.4. Wykorzystane standardy

Implementowany system jest zgodny ze standardami FIPA, ze względu na wykorzysta-nie framework'a JADE.

- FIPA ACL (Agent Communication Language)
- FIPA SL (Semantic Language) - język semantyczny dla komunikacji ACL

5.5. Algorytmy

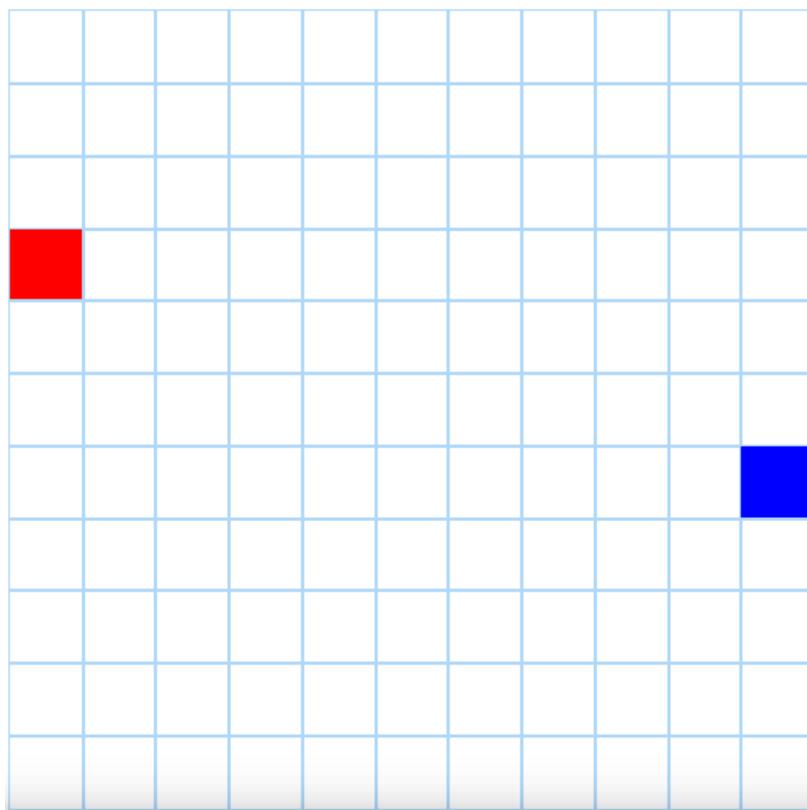
Algorytmy wykorzystane w systemie zostały pokrótko opisane w sposobie implemen-tacji agentów. Na uwagę zasługuje sposób implementacji środowiska oraz sposób porusza-nia się po nim agentów. Stan naszego systemu jest zdefiniowany w postaci mapy 2D, gdzie każda z kratek odpowiada współrzędnej w środowisku agentów. Do każdej współrzędnej mogą być przypisane trzy wartości: czy współrzędna jest ulicą czy parkingiem. Samochody są losowo inicjalizowane na współrzędnych odpowiadających ulicom. Parking docelowy jest znajdowany poprzez obliczanie ścieżek do wszystkich parkingów, a następnie wy-branie najkrótszej trasy. Samochód porusza się w jego kierunku po drodze obliczonej

w algorytmie wybierania ścieżki (opisanym poniżej). Końcowo samochód wjeżdża na współrzedną z parkingiem. Dopuszcza się możliwość, że dwa samochody znajdują się na tej samej współrzędnej. W aktualnej wersji systemu wszystkie pola mapy są ulicami.

Algorytm wyznaczania ścieżki został zaimplementowany przy pomocy algorytmu przeszukiwania wszerz (BFS). Algorytm ten w formie iteracyjnej wykorzystuje kolejkę do określenia, które kolejne punkty powinniśmy przebadać. Rozpoczynając od punktu startowego dodajemy wszystkie sąsiednie, nieodwiedzone wcześniej, punkty do kolejki i powtarzamy ten proces dla wszystkich punktów w kolejce. Jeżeli dotrzymy do punktu docelowego to zwracamy listę, która jest ścieżką od miejsca początkowego do docelowego. Nie jest to najoptymalniejszy algorytm (u nas, $O(i \cdot j)$ time | $O(i \cdot j)$ space, gdzie i - szerokość mapy, j - wysokość mapy), jednak został wybrany ze względu na łatwość implementacji w realizowanym systemie.

5.6. Wizualizacja

Postanowiliśmy zvisualizować działanie naszego projektu poprzez wyświetlanie informacji o aktualnym stanie systemu w konsoli oraz pokazywaniu położenia agentów na mapie 2D. Agent aktualizując swoje położenie wysyła informację do serwera, który przechowuje dane o bieżącym położeniu każdego agenta. Następnie serwer przekazuje aktualne informacje do aplikacji React, gdzie wyświetlana jest mapa. Backend i Frontend połączone są ze sobą poprzez websocket, więc wszystkie aktualizacje są wykonywane w czasie rzeczywistym. Ze względu na taki stos technologiczny możliwe są opóźnienia pomiędzy prezentowanym stanem systemu na mapie, a w konsoli. Przykładowa mapa jest widoczna na rysunku 5, gdzie kratka: biała oznacza drogę, czerwona parking, a niebieska samochód. Przykładowe wizualizacje naszego systemu są dostępne w postaci gifów na repozytorium naszego projektu w README.



Rysunek 5. Przykład mapy.

5.7. Napotkane problemy i zmiany w kolejnej iteracji

- Żaden członek zespołu nie jest biegły w Javie, więc dodatkową trudnością było szybkie zapoznanie się z samym językiem, a nie tylko nowym frameworkm. Z tego powodu, nie mogliśmy się wyłącznie skupić na najlepszym rozwiązaniu i implementacji problemu.
- Podczas implementacji równoległego oczekiwania na wiadomości mieliśmy problem z przechwytywaniem wiadomości przez niepowołane do tego zachowania. Problem ten rozwiązaliśmy wykorzystując zaimplementowane w JADE szablony wiadomości (jade.lang.acl.MessageTemplate). Za pomocą metod MatchPerformative, MatchConversationID i wyfiltrowaliśmy przychodzące wiadomości skierowane do poszczególnych zachowań.

6. Zmiany, ulepszenia i uzupełnienie informacji względem poprzedniego etapu C

Implementacja systemu

Implementacja systemu została uzupełniona o zachowania, które pominięto w etapie C.

Car Agent

`ReceiveCancelReservationByIncreaseDistance` – odbiera wiadomość CANCEL od CarTrackera o zamknięciu rezerwacji z powodu oddalania się od wybranego parkingu. CarAgent traci wtedy status approachera.

ParkingAgent

`CancelReservationIncreaseDistance` – oczekuje na wiadomość CANCEL od CarTrackera czy CarAgent oddala się od parkingu. Dokonuje anulowania rezerwacji, jeśli CarTracker wyliczył sobie, że samochód się oddala.

`TrackCar` – śledzi położenie samochodu. Zaprzestaje śledzenia, gdy samochód pomyślnie zaparkował i przesyła wiadomość CANCEL o zaprzestaniu subskrybowania lokalizacji. Przesyła wiadomość CANCEL o anulowaniu rezerwacji, jeśli samochód oddala się od parkingu.

Budowa i odczytywanie komunikatów

Komunikaty budowane były w bardzo prosty i przejrzysty sposób. ID konwersacji wskazuje jakiej wiadomości można się spodziewać po danym komunikacie (np. ‘update-location’), a sama zawartość komunikatu zawiera najczęściej jedną informację (np. zestaw współrzędnych samochodu). Dzięki temu odczytywanie komunikatów również nie jest skomplikowane i opiera się na odbieraniu wiadomości z właściwym ID konwersacji, a następnie pobraniu informacji zawartej w treści. Warto zaznaczyć, że w naszym systemie bardziej skomplikowane komunikaty nie miałyby po prostu zastosowania.

Wizualizacja

Wizualizacja systemu wieloagentowego została wzbogacona nie tylko od strony estetycznej, ale przede wszystkim udało się nam przybliżyć prawdziwe warunki miejskie – samochody mogą poruszać się po wyznaczonych polach (ulicach), podczas gdy pozostałe pola zajmują budynki, parki, itd. Każdy parking wyświetla informację o aktualnej liczbie dostępnych miejsc. Informacja ta aktualizowana jest dynamicznie podczas symulacji. Nowa wersja wizualizacji systemu pokazana jest w podrozdziale *Testy*.

7. Braki

Algorytm wyliczania anulowania rezerwacji przez CarTrackera został zaimplementowany tylko w podstawowej wersji: jeśli choć raz CarTracker zauważa oddalenie się od parkingu, rezerwacja jest anulowana. **W kontekście projektu nie jest to duża niedogodność.** Na naszej siatce algorytm ten działa bez zarzutu, ponieważ wszystkie ulice są do siebie prostopadłe/równoległe i obliczenia są bardzo proste. Jedynie w rzeczywistości powinny być uwzględniane objazdy, drogi jednokierunkowe, podejrzanie długiego czasu postoju w jednym miejscu itd.

Nie zdążyliśmy zaimplementować funkcjonalności wyjazdu z parkingu. W związku z tym jedynie zachowania zwalniające zarezerwowane miejsce parkingowe to zamknięcie rezerwacji przez CarAgenta lub anulowanie rezerwacji z powodu oddalania się samochodu od parkingu.

Nie wszystkie wymieniane komunikaty udało się utrzymać w standardzie FIPA.

Odnosząc się do uwag w recenzji Pana dr Piotr Pałka - możliwe, że nie wszystkie dziedziczenia w zachowaniach agentów zostały dobrane w sposób optymalny. Priorytetem było dla nas stworzenie działającego systemu w możliwie najmniej skomplikowany sposób, jeśli chodzi o kod. Po przeanalizowaniu zachowań są one jednak w naszym odczuciu poprawne, co zdają się potwierdzać pozytywne wyniki testów.

8. Testy

W celu sprawdzenia poprawności działania systemu wykonano szereg testów funkcjonalnych. Przeprowadzenie testów ułatwiło wcześniejsze wprowadzenie wizualizacji opisanej w poprzednich punktach tego sprawozdania. W celu przeprowadzenia testów dokonano również parametryzacji wybranych zachowań agentów tj. Odwoływanie rezerwacji oraz oddalania się od miejsca rezerwacji.

Uwaga techniczna: Niestety screeny na pierwszy rzut oka mogą być niewczytelne. Robiliśmy o w naszej mocy, aby jak najlepiej przestawić symulację, jednak tak dużego rozmiaru plansza nie ma szans wyglądać bardziej przejrzystie w dokumencie. Na szczęście przy uważnym przyjrzeniu się widać, że aplikacja przechodziła pozytywnie nasze testy.

W repozytorium dostępne są również nagrania bardziej zaawansowanych z zawartych w tym rozdziale testów.

Test 1. Samochód rezerwuje miejsce i dojeżdża na parking

Cel: Sprawdzenie czy system spełnia swoje główne zadanie dla pojedynczego samochodu i pojedynczego parkingu: umożliwia rezerwację miejsc na najbliższym parkingu.

Agenty: 1 CAR 1 PARKING

Konfiguracja: -agents "car1:CarAgent; parking1:ParkingAgent"

Przebieg:

Początkowo na Parkingu 1 znajdują się 3 wolne miejsca. Po pojawienniu się samochodu 1 i dokonaniu przez niego rezerwacji liczba wolnych miejsc zostaje zmniejszona do 2:



Samochód 1 zaczyna poruszać się najkrótszą drogą w kierunku Parkingu 1:



Samochód 1 wjeździ na Parking 1 i znika z wizualizacji. Liczba wolnych miejsc pozostaje równa 2, co jest poprawnym zachowaniem systemu:



Logi z konsoli potwierdzające przebieg testu:

```

Hello car1@192.168.1.16:1099/JADE is ready.
Hello parking1@192.168.1.16:1099/JADE is ready.
[1, 1]
car1@192.168.1.16:1099/JADE      Sent INFORM_REF to parking agents. Waiting for replies...
parking1@192.168.1.16:1099/JADE  Sent reply with location to car agent.
car1@192.168.1.16:1099/JADE      Received locations from all parking agents.
car1@192.168.1.16:1099/JADE      Location of parking1@192.168.1.16:1099/JADE is [1, 1]
car1@192.168.1.16:1099/JADE      Sent CFP to parking agents. Waiting for replies...
parking1@192.168.1.16:1099/JADE  Sent reply with information about availability
car1@192.168.1.16:1099/JADE      Location parking1@192.168.1.16:1099/JADE [1, 1] is available for reservation.
car1@192.168.1.16:1099/JADE      Best Location: [1, 1] and shortestDistance is 33
parking1@192.168.1.16:1099/JADE  Sent reply with information about reservation.
car1@192.168.1.16:1099/JADE      Place reserved at: parking1@192.168.1.16:1099/JADE
car1@192.168.1.16:1099/JADE      Sent reservation info to parking1@192.168.1.16:1099/JADE
parking1@192.168.1.16:1099/JADE  Got reservation info from car1@192.168.1.16:1099/JADE
parking1@192.168.1.16:1099/JADE  I am tracking now car1@192.168.1.16:1099/JADE
car1@192.168.1.16:1099/JADE      parking1@192.168.1.16:1099/JADE has subscribed for info about my location
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [17, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [16, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [15, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [14, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [13, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [12, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [11, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [10, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [8, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [7, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [6, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [5, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [4, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [3, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [2, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [1, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 15]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 14]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 13]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 12]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 11]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 10]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 9]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 8]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 7]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 6]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 5]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 4]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 3]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 2]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [0, 1]
parking1@192.168.1.16:1099/JADE  Current location of car1@192.168.1.16:1099/JADE is [1, 1]
car1@192.168.1.16:1099/JADE      Received Cancel Subscription info from + parking1@192.168.1.16:1099/JADE. I stopped sending info...

```

Wynik: Pozytywny

Test 2. Parking jest w stanie obsłużyć kilka samochodów

Cel: Sprawdzenie czy parking jest w stanie śledzić więcej niż 1 samochód, który zarezerwował miejsce na tym parkingu.

Agenty: 2 CAR 1 PARKING

Konfiguracja: -agents "car1:CarAgent; car2:CarAgent; parking1:ParkingAgent"

Przebieg:

Początkowo na parkingu 1 znajdują się 3 wolne miejsca. Po pojawienniu się dwóch samochodów, które dokonały rezerwacji na parkingu, liczba wolnych miejsc spadła do 1:



Samochód, który był bliżej parkingu, już zaparkował. Drugi samochód wciąż zmierza w stronę parkingu:



Po pewnym czasie również drugi samochód zaparkował na zarezerwowanym miejscu:



Logi z konsoli potwierdzające przebieg tego testu ze względu na ich długość znajdują się w repozytorium projektu pod scieżką TESTS/test2.txt

Wynik: pozytywny

Test 3. Parking ogranicza liczbę obsługiwanych w danym momencie samochodów do liczby wolnych miejsc

Cel: Sprawdzenie czy parking spróbuje obsłużyć wszystkich zgłaszających się samochodów mimo, że ma mniej wolnych miejsc niż jest oczekujących samochodów.

Agenty: 4 CAR 1 PARKING

Konfiguracja: -agents "car1:CarAgent; car2:CarAgent; car3:CarAgent; car4:CarAgent; parking1:ParkingAgent "

Przebieg:

Na początku dostępne są 2 miejsca parkingowe na Parkingu1 oraz jest 4 oczekujących CarAgentów.



Tylko 2 CarAgentów zrobiło rezerwacje na miejsca parkingowe. Tylko oni zmienią swoje pozycje i zaczną zmierzać w stronę parkingu:



Agenci którzy zmierzali w stronę parkingu pomyślnie zaparkowali. Pozostałych 2 agentów pozostało na swoich miejscach:



Logi z konsoli potwierdzające przebieg tego testu ze względu na ich długość znajdują się w repozytorium projektu pod ścieżką TESTS/test3_console_log.txt

Wynik: pozytywny

Test 4. System kieruje samochody do najbliższego wolnego parkingu

Cel: Sprawdzenie czy system faktycznie wskazuje najbliższy parking z wolnym miejscem.

Agenty: 2 CAR 2 PARKING

Konfiguracja: -agents "car1:CarAgent; parking1:ParkingAgent; parking2: ParkingAgent "

Przebieg:

Na planszy stoi jeden samochód na polu [20,2]. Istnieją dwa parkingi – pierwszy kilka pól dalej, dugi na przeciwnym końcu planszy. Dostępne są również dwa miejsca parkingowe – po jednym na każdym parkingu. Widać, że CarAgent zarezerwował miejsce na bliższym parkingu (wyświetlana liczba wolnych miejsc spadła do 0). Na dalszym parkingu miejsce wciąż pozostaje wolne:



CarAgent zmierza do parkingu, na którym zarezerwował miejsce.



CarAgent pomyślnie zaparkował:



Logi z konsoli potwierdzające przebieg testu:

```

Hello parking1@192.168.1.31:1099/JADE is ready.
Hello parking2@192.168.1.31:1099/JADE is ready.
Hello car1@192.168.1.31:1099/JADE is ready.
[11, 6]
[1, 39]
car1@192.168.1.31:1099/JADE Sent INFORM_REF to parking agents. Waiting for replies...
parking2@192.168.1.31:1099/JADE Sent reply with location to car agent.
parking1@192.168.1.31:1099/JADE Sent reply with location to car agent.
car1@192.168.1.31:1099/JADE Received locations from all parking agents.
car1@192.168.1.31:1099/JADE Location of parking1@192.168.1.31:1099/JADE is [11, 6]
car1@192.168.1.31:1099/JADE Location of parking2@192.168.1.31:1099/JADE is [1, 39]
car1@192.168.1.31:1099/JADE Sent CFP to parking agents. Waiting for replies...
parking1@192.168.1.31:1099/JADE Sent reply with information about availability
parking2@192.168.1.31:1099/JADE Sent reply with information about availability
car1@192.168.1.31:1099/JADE Location parking1@192.168.1.31:1099/JADE [11, 6] is available for reservation.
car1@192.168.1.31:1099/JADE Location parking2@192.168.1.31:1099/JADE [1, 39] is available for reservation.
car1@192.168.1.31:1099/JADE Best Location: [11, 6] and shortestDistance is 14
parking1@192.168.1.31:1099/JADE Sent reply with information about reservation.
car1@192.168.1.31:1099/JADE Place reserved at: parking1@192.168.1.31:1099/JADE
parking1@192.168.1.31:1099/JADE Sent reservation info to parking1@192.168.1.31:1099/JADE
parking1@192.168.1.31:1099/JADE Got reservation info from car1@192.168.1.31:1099/JADE
I am tracking now car1@192.168.1.31:1099/JADE
parking1@192.168.1.31:1099/JADE has subscribed for info about my location
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [20, 2]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [20, 3]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [20, 4]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [20, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [19, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [17, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [16, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [15, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [14, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [13, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [12, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [11, 5]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [11, 6]
parking1@192.168.1.31:1099/JADE Current location of car1@192.168.1.31:1099/JADE is [11, 6]
car1@192.168.1.31:1099/JADE Received Cancel Subscription info from + parking1@192.168.1.31:1099/JADE. I stopped sending info..
|

```

Wynik: pozytywny

Test 5. System obsługuje odwołanie rezerwacji przez CarAgent

Cel: sprawdzenie czy system pozwala na odwołanie rezerwacji przez samochód i czy parking reaguje na to w sposób prawidłowy.

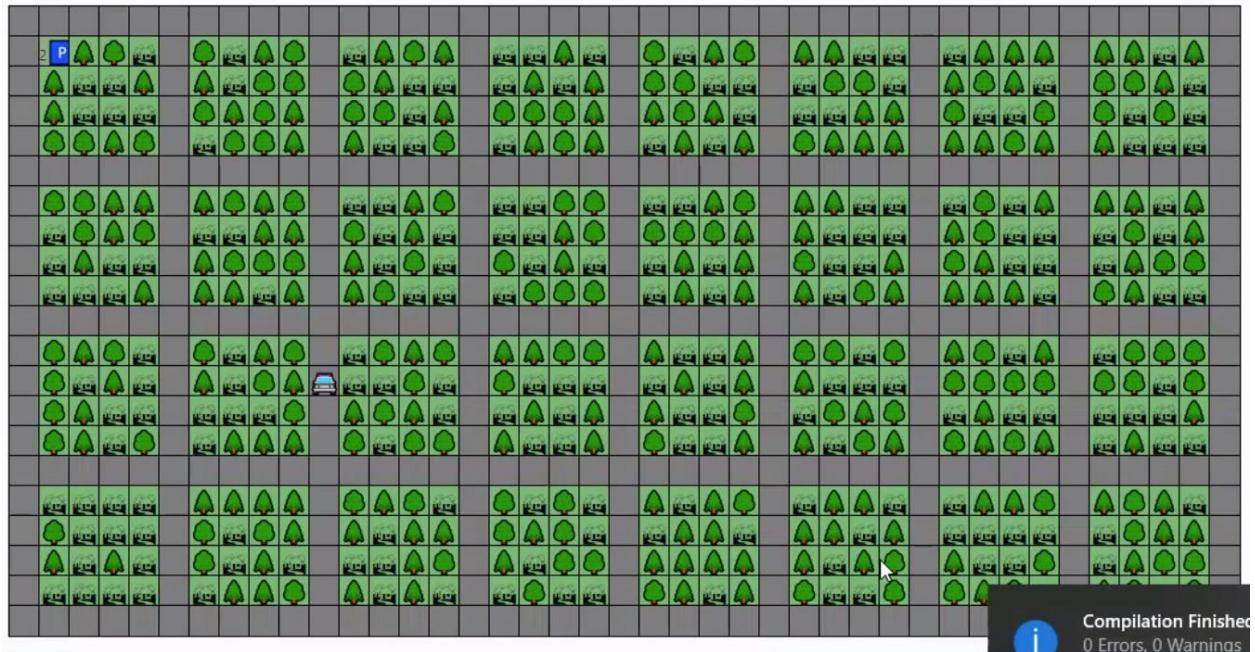
Agenty: 1 CAR 1 PARKING

Konfiguracja: -agents "car1:CarAgent; parking1:ParkingAgent"

Przebieg:

W repozytorium zawarte jest nagranie ekranu z przebiegu tego testu w TEST/test5.mp4

Samochód dokonuje rezerwacji miejsca na Parkingu i porusza się w jego kierunku. Parking zmniejsza liczbę dostępnych miejsc z 3 do 2:



W połowie drogi samochód postanawia odwołać rezerwację. Parking trzymuję informacje o odwołaniu rezerwacji, przestaje śledzić samochód i zwiększa liczbę wolnych miejsc o 1:



Logi z konsoli potwierdzające przebieg testu:

```
Hello car1@192.168.1.16:1099/JADE is ready.  
Hello parking1@192.168.1.16:1099/JADE is ready.  
[1, 1]  
car1@192.168.1.16:1099/JADE Sent INFORM_REF to parking agents. Waiting for replies...  
parking1@192.168.1.16:1099/JADE Sent reply with location to car agent.  
car1@192.168.1.16:1099/JADE Received locations from all parking agents.  
car1@192.168.1.16:1099/JADE Location of parking1@192.168.1.16:1099/JADE is [1, 1]  
car1@192.168.1.16:1099/JADE Sent CFP to parking agents. Waiting for replies...  
parking1@192.168.1.16:1099/JADE Sent reply with information about availability  
car1@192.168.1.16:1099/JADE Location parking1@192.168.1.16:1099/JADE [1, 1] is available for reservation.  
car1@192.168.1.16:1099/JADE Best Location: [1, 1] and shortestDistance is 26  
parking1@192.168.1.16:1099/JADE Sent reply with information about reservation.  
car1@192.168.1.16:1099/JADE Place reserved at: parking1@192.168.1.16:1099/JADE  
car1@192.168.1.16:1099/JADE Sent reservation info to parking1@192.168.1.16:1099/JADE  
parking1@192.168.1.16:1099/JADE Got reservation info from car1@192.168.1.16:1099/JADE  
parking1@192.168.1.16:1099/JADE I am tracking now car1@192.168.1.16:1099/JADE  
car1@192.168.1.16:1099/JADE parking1@192.168.1.16:1099/JADE has subscribed for info about my location  
- - - - -  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [15, 10]  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [14, 10]  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [13, 10]  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [12, 10]  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [11, 10]  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [10, 10]  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [9, 10]  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [8, 10]  
parking1@192.168.1.16:1099/JADE Current location of car1@192.168.1.16:1099/JADE is [7, 10]  
car1@192.168.1.16:1099/JADE Sent CANCEL to target parking.  
parking1@192.168.1.16:1099/JADE Send info about cancelling the reservation. This place is available for further customers  
car1@192.168.1.16:1099/JADE Reservation cancelled.
```

Wynik testu: pozytywny

Test 6. System odwołuje rezerwacje, gdy samochód oddala się od zarezerwowanego miejsca

Cel: Sprawdzenie czy system automatycznie odwoła rezerwację i zwiększy ilość wolnych miejsc, gdy samochód zacznie oddalać się od zarezerwowanego miejsca.

Agenty: 1 CAR 1 PARKING

Konfiguracja: -agents "car1:CarAgent; parking1:ParkingAgent"

Przebieg:

W repozytorium zawarte jest nagranie ekranu z przebiegu tego testu w TEST/test6.mp4

Początkowo liczba wolnych miejsc w Parkingu równa jest 3. Samochód dokonuje rezerwacji i liczba miejsc spada do 2:



Samochód porusza się w stronę Parkingu:



Samochód zaczął oddalać się od Parkingu. Z tego powodu parking odwołał rezerwację i zwiększył liczbę swoich wolnych miejsc o 1:



Logi z konsoli potwierdzające przebieg testu:

```
Hello car1@192.168.1.16:1099/JADE is ready.  
Hello parking1@192.168.1.16:1099/JADE is ready.  
[1, 1]  
car1@192.168.1.16:1099/JADE      Sent INFORM_REF to parking agents. Waiting for replies...  
parking1@192.168.1.16:1099/JADE  Sent reply with location to car agent.  
car1@192.168.1.16:1099/JADE      Received locations from all parking agents.  
car1@192.168.1.16:1099/JADE      Location of parking1@192.168.1.16:1099/JADE is [1, 1]  
car1@192.168.1.16:1099/JADE      Sent CFP to parking agents. Waiting for replies...  
parking1@192.168.1.16:1099/JADE  Sent reply with information about availability  
car1@192.168.1.16:1099/JADE      Location parking1@192.168.1.16:1099/JADE [1, 1] is available for reservation.  
car1@192.168.1.16:1099/JADE      Best Location: [1, 1] and shortestDistance is 42  
parking1@192.168.1.16:1099/JADE  Sent reply with information about reservation.  
car1@192.168.1.16:1099/JADE      Place reserved at:parking1@192.168.1.16:1099/JADE  
car1@192.168.1.16:1099/JADE      Sent reservation info to parking1@192.168.1.16:1099/JADE  
parking1@192.168.1.16:1099/JADE  Got reservation info from car1@192.168.1.16:1099/JADE  
I am tracking now car1@192.168.1.16:1099/JADE  
parking1@192.168.1.16:1099/JADEhas subscribed for info about my location  
Current location of car1@192.168.1.16:1099/JADE is [10, 31]  
Current location of car1@192.168.1.16:1099/JADE is [10, 30]  
Current location of car1@192.168.1.16:1099/JADE is [9, 30]  
Current location of car1@192.168.1.16:1099/JADE is [8, 30]  
Current location of car1@192.168.1.16:1099/JADE is [7, 30]  
Current location of car1@192.168.1.16:1099/JADE is [6, 30]  
Current location of car1@192.168.1.16:1099/JADE is [7, 30]  
Current location of car1@192.168.1.16:1099/JADE is [8, 30]  
Car agent is going far away. I am cancelling reservation.  
Reservation cancelled by car tracker.  
Send info about cancelling the reservation. Car increased distance.
```

Wynik testu: pozytywny

Podsumowanie

Numer testu	Opis	Wynik
1	Samochód rezerwuje miejsce i dojeżdża na parking	😊
2	Parking jest w stanie obsłużyć kilka samochodów	😊
3	Parking ogranicza liczbę obsługiwanych samochodów do liczby wolnych miejsc	😊
4	System kieruje samochody do najbliższego wolnego parkingu	😊
5	System obsługuje odwołanie rezerwacji	😊
6	System odwołuje rezerwacje, gdy samochód oddala się od zarezerwowanego miejsca	😊

8. Studia przypadku

Studium przypadku 1 Uniknięcie dojazdu do zajętego parkingu

Nagranie tego przypadku dostępne jest w repozytorium pod ścieżką:

[CASESTUDIES/CASESTUDY1.mp4](#)

Dostępne są dwa parkingi mające po 3 miejsca, na których chce zaparkować 5 samochodów. Samochody dokonują rezerwacji. Po upływie chwili pierwszy samochód dojechał na obszar swojego parkingu (po lewej), pozostałe 4 samochody nadal kierowały się w kierunku zarezerwowanych miejsc:



Następnie jeden z samochodów dotarł do centralnego parkingu, na którym miał zarezerwowane miejsce i tam zaparkował. W ruchu pozostały 3 samochody mające swoje zarezerwowane miejsca:



Dwa z trzech aut kierują się na parking centralny:



Dwa auta zaparkowały na parkingu centralnym, gdzie miały zarezerwowane miejsca i parking się zapełnił. Trzecie auto kieruje się na lewy parking, gdzie ma ze rezerwowane miejsce I w ten sposób unika dojazdu na w pełni zajęty parking. Gdyby nie obecność systemu kierowca ostatniego auta najpierw skierowałby się na najbliższy parking, a po stwierdzeniu braku wolnych miejsc musiałby wracać I jechać na kolejny parking. To studium przypadku pokazuje, że nasz system pozwala uniknąć tego typu irytuujących rozczarowań i zapewnia oszczędność czasu.

Studium przypadku 2 Dynamiczne wskazywanie wolnego miejsca

Nagranie tego przypadku dostępne jest w repozytorium pod ścieżką:

CASESTUDIES/ CaseStudy2_no_noise.mp4.

Studium przypadku nr 2 ma na celu pokazanie, że nasza aplikacja pozwala na dynamiczne wskazywanie wolnego miejsca. W momencie pojawienia się nowego wolnego miejsca, system od razu obsługuje oczekującego klienta. Nasz system pozwala nie tylko na uniknięcie zbędnej jazdy do zapełnionego parkingu, ale też od razu informuje, gdy miejsce się zwolni.

Dostępne jest tylko 1 miejsce na Parkingu 1 –pole [1,1] oraz występują dwa oczekujące samochody CarAgent1 na polu [20,0] i CarAgent2 na polu [20,2]. Car Agent1 jest pierwszy i to on dokonuje rezerwacji wolnego miejsca. Niestety nie zdążyliśmy nagrać samego początku symulacji, gdzie cyfra obok parkingu pokazuje dokładnie jedno wolne miejsce:



CarAgent 1 zaczyna jechać w stronę parkingu, Car Agent 2 pozostaje na swoim miejscu i oczekuje, ponieważ nie ma rezerwacji:



Jednak po pewnym czasie Car Agent 1 zaczyna się oddalać od parkingu (cofać). CarTracker anuluje jego rezerwację. W tym momencie CarAgent 2 przejmuje rezerwację na miejsce, które się zwolniło i zmierza w kierunku parkingu:



CarAgent2 za chwilę zaparkuje, a CarAgent1 dalej zmierza w nieznanym nam kierunku:



CarAgent2 pomyślnie zaparkował, CarAgent1 odjeżdża sobie w przeciwnym kierunku:



Można prześledzić logi zwracane w konsoli, potwierdzające przebieg case study:

1. CarAgent1 był pierwszy z zdobyciu rezerwacji. CarAgent2 otrzymał odmowę.

```
Hello car2@192.168.1.31:1099/JADE is ready.  
Hello parking1@192.168.1.31:1099/JADE is ready.  
Hello car1@192.168.1.31:1099/JADE is ready.  
[1, 1]  
car1@192.168.1.31:1099/JADE Sent INFORM_REF to parking agents. Waiting for replies...  
car2@192.168.1.31:1099/JADE Sent INFORM_REF to parking agents. Waiting for replies...  
parking1@192.168.1.31:1099/JADE Sent reply with location to car agent.  
parking1@192.168.1.31:1099/JADE Sent reply with location to car agent.  
car1@192.168.1.31:1099/JADE Received locations from all parking agents.  
car2@192.168.1.31:1099/JADE Received locations from all parking agents.  
car2@192.168.1.31:1099/JADE Location of parking1@192.168.1.31:1099/JADE is [1, 1]  
car1@192.168.1.31:1099/JADE Location of parking1@192.168.1.31:1099/JADE is [1, 1]  
car1@192.168.1.31:1099/JADE Sent CFP to parking agents. Waiting for replies...  
car2@192.168.1.31:1099/JADE Sent CFP to parking agents. Waiting for replies...  
parking1@192.168.1.31:1099/JADE Sent reply with information about availability  
parking1@192.168.1.31:1099/JADE Sent reply with information about availability  
car1@192.168.1.31:1099/JADE Location parking1@192.168.1.31:1099/JADE [1, 1] is available for reservation.  
car2@192.168.1.31:1099/JADE Location parking1@192.168.1.31:1099/JADE [1, 1] is available for reservation.  
car1@192.168.1.31:1099/JADE Best Location: [1, 1] and shortestDistance is 21  
car2@192.168.1.31:1099/JADE Best Location: [1, 1] and shortestDistance is 22  
parking1@192.168.1.31:1099/JADE Sent reply with information about reservation.  
parking1@192.168.1.31:1099/JADE Sent reply with information about reservation.  
car2@192.168.1.31:1099/JADE Failure. Parking is occupied. ✓  
car1@192.168.1.31:1099/JADE Place reserved at:parking1@192.168.1.31:1099/JADE ✓  
car1@192.168.1.31:1099/JADE Sent reservation info to parking1@192.168.1.31:1099/JADE  
parking1@192.168.1.31:1099/JADE Got reservation info from car1@192.168.1.31:1099/JADE ✓  
parking1@192.168.1.31:1099/JADE I am tracking now car1@192.168.1.31:1099/JADE  
car1@192.168.1.31:1099/JADE parking1@192.168.1.31:1099/JADE has subscribed for info about my location
```

2. Śledzenie CarAgent1 i rozsyłanie CFP przez CarAgent2:

- CarTracker wyłapuje, że CarAgent1 się oddala, zamiast przybliżać. Anuluje rezerwację i umożliwia zrobienie rezerwacji przez CarAgent2:

4. Rozpoczęcie śledzenia CarAgent2 i odnotowanie jego pomyślnego zaparkowania:

Spis rysunków

1. Architektura systemu.	8
2. Model interakcji.	15
3. Model agentów.	19
4. Model znajomości.	21
5. Przykład mapy.	26

Spis tabel

1. Rola Parking Manager.	10
2. Rola Car Tracker.	11
3. Rola Parking Mapper.	12
4. Rola Client.	13
5. Rola Approacher.	14
6. Model usług.	20