Fakultet elektrotehnike i računarstva

Unska 3, Zagreb

Pokretni programski agenti - seminar

Simulacija tramvajskog prometa pomoću pokretnih agenata

Alen Huskić

Ante Pavlinović

Tablica sadržaja

UVOD

Napretkom i razvojem tehnologije, agentske tehnologije su nam svakim danom sve bliže i bliže. Prednost agentskih tehnologija nad ostalim tehnologijama je da ta tehnologija omogućuje razvoj sustava u kojima se programski agenti na autonoman način međusobno koordiniraju tj. izmjenjuju podatke i druge informacije s ciljem obavljanja zadaća koje su im zadali korisnici sustava. Po sposobnosti samostalnog ispunjavanja ciljeva autonomni agenti se razlikuju od bilo kojeg drugoga tipa programskih komponenti. Da bi agenti se mogli međusobno koordinirati, djelovati samostalno, sa uspjehom izvršavati zadane ciljeve u njih mora biti ugrađena dovoljna inteligencija, a da bi ti agenti još mogli biti pokretljivi mora im se ugraditi pokretljivost tj. mora im se ugraditi mogućnost kretanja od čvora do čvora. Za sve gore navedene aktivnosti potrebna im je agentska platforma koja im to omogućuje.

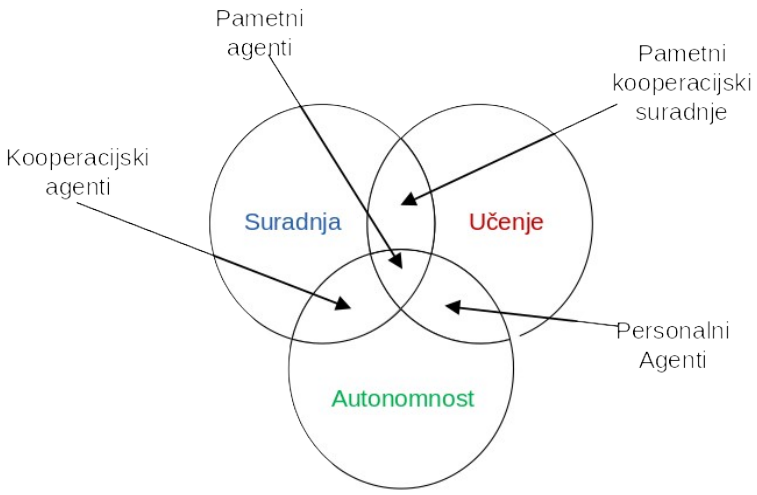
Danas je sve u pokretu, jedemo u pokretu, koristimo najrazličitiju opremu dok razgovaramo u pokretu (mobiteli, PDA, prijenosna računala) i sva ta oprema treba moći međusobno komunicirati. Komunikacija se odvija kroz nove generacije telekomunikacijskih mreža za koje je karakteristično objedinjavanje različitih vrsta postojećih mrežnih tehnologija i ostvarivanje novih konvergiranih telekomunikacijskih usluga.

Glavni problem koji je nastao zbog sve veće pokretljivosti korisnika i korištenja korisničke opreme različitih mogućnosti je problem pokretljivosti usluge. Problem uključuje pružanje usluga korisniku u pokretu te prilagođavanje usluga zahtjevima korisnika i mogućnostima korisničke opreme. Zbog svojih osobina, više-agentski sustavi s pokretnim agentima predstavljaju učinkovito rješenje za ostvarivanje potpune pokretljivosti korisnika i usluga u novim uvjetima u telekomunikacijskoj mreži.

1. Agenti

Pod pojmom agenti misli se na softverske agente. Definicija agenta je entitet u mreži koji samostalno obavlja skup poslova zadanih od strane korisnika, odnosno vlasnika. Koncept agenta može poslužiti za objašnjenje kompleksnog softverskog entiteta koji je donekle sposoban autonomno djelovati u okolini predstavljajući korisnika, vlasnika. Za razliku od objekata, koji su opisani pomoću metoda i atributa, agent se opisuje pomoću ponašanja.
Jedno od svojstva koje agent mora ispuniti jest samostalnost (autonomnost), sposobnost suradnje, inteligencija, pokretljivost, reaktivnost i proaktivnost.

Osnovna klasifikacija agenata jest prema mobilnosti. Agenti mogu biti statični ili mobilni. Mobilni agenti imaju svojstvo kretanje po mreži. Agente možemo podijeliti kao deliberativne i reaktivne agente. Deliberativni ili kongitivni agenti imaju sposobnost logičkog odlučivanja. Takvi agenti imaju eksplicitni simbolički model okružja. Svaki agent ima svoju bazu podataka i postupke za obavljanje zadataka. Reaktivni agenti nemaju eksplicitni simbolički model okružja. Reaktivni agenti ne moraju biti nužno inteligentni da bi zajedno rješavali složeni problem. Takvi agenti prepoznaju određene situacije i reagiraju na njih.
Osnovnu klasifikaciju agenta možemo vidjeti na slici 1. Autonomnost se odnosi na svojstvo da je agent sposoban obavljati radnje bez ljudskog utjecaja . Također agent kao autonomni objekt djeluje samoinicijativno kao odgovor na zahtjeve okoline. Također agent kao autonomni entitet mora imati određenu pokretljivost, tj. svojstvo da se može kretati po mreži. Suradnja se odnosi na svojstvo da je agent sposoban surađivati s drugim agentima ili ljudima putem nekog komunikacijskog jezika. Zadnje svojstvo jest da je agent sposoban učiti putem reakcije/ interakcije s drugim sudionicima sustava.



Slika 1. Tipologija agenata

Treba napomenuti da granice između tipova agenata nisu strogo određene. Kooperacijski agent ima naglasak na suradnju i autonomnost, ali može imati i elemente učenja u sebi. Isto vrijedi i za personalne agente. Oni pretežito stavljaju naglasak na autonomnosti i učenju, ali elementi suradnje su i kod njih prisutni.

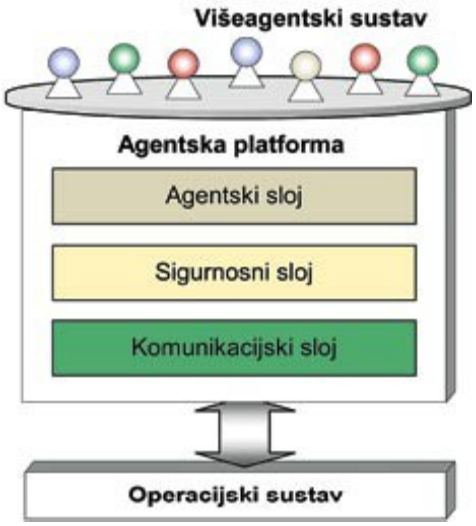
1.1 Agentska platforma

Kao što je već spomenuto u uvodu, da bi agenti mogli ispuniti zadatke koje su im dodijelili korisnici, moraju moći pristupiti i djelovati na nekom čvoru u mreži, znači da mora postojati okruženje (platforma) koja će prihvatiti programskog agenta i omogućiti mu izvođenje.

„Okružje, odnosno platforma upravlja životnim ciklusom agenata (tj., njihovim stvaranjem i uništavanjem) te omogućuje njihovu migraciju i komunikaciju. Agentska platforma nalazi se na čvoru u mreži. Skup povezanih agentskih platformi sačinjavaju agentsko okružje unutar kojega se agenti kreću i izvode operacije.

Tipična agentska platforma pruža mehanizme za:

- upravljanje agentima
- sigurnost (povjerljivost i integritet informacija)
- osiguravanje komunikacije među agentima.“[1]



Slika 2. Arhitektura agentske platforme

Arhitektura agentske platforme sadrži sljedeće slojeve:

- Agentski sloj- agentski sloj je odgovoran za upravljanje agentima (npr., njihovo kreiranje, kretanje, identifikaciju, itd.)
- Sigurnosni sloj- sigurnosni sloj osigurava mehanizme za zaštitu privatnosti informacija i štićenje od neovlaštenih aktivnosti u informacijsko-komunikacijskom sustavu. Sigurnosni mehanizmi su potrebni za ostvarivanje sigurne komunikacije i migracije agenata.
- Komunikacijski sloj- komunikacijski sloj se brine o isporuci poruka koje agenti razmjenjuju

Agentska platforma omogućuje realizaciju više-agentskoga sustava koji predstavlja raspodijeljenu aplikaciju zasnovanu na agentima.

[1] Hrvoje Gjurić: „Java Agenti“, seminarski rad

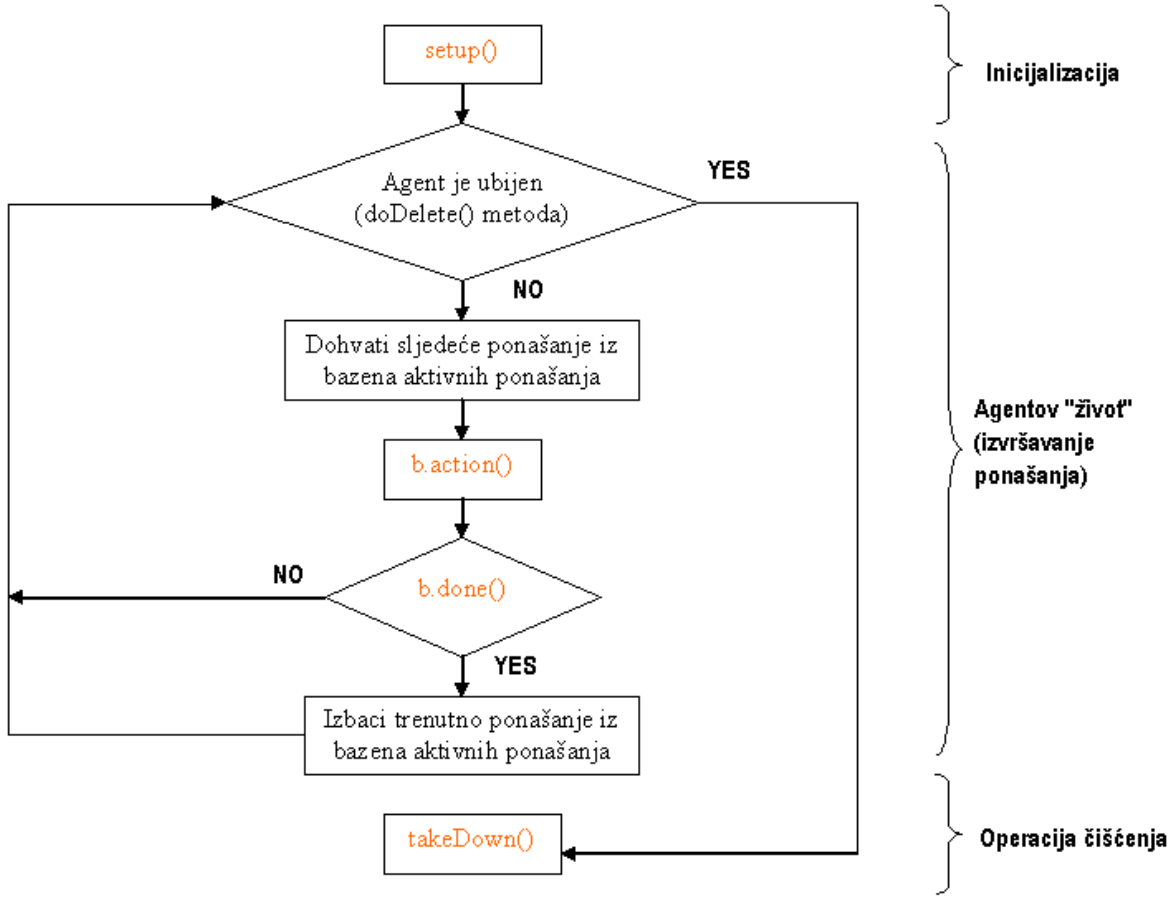
2. JADE (Java Agent Development framework)

JADE (Java Agent Development Framework) je softverska razvojna okolina za razvoj agentskih aplikacija po FIPA specifikacijama za kooperativne inteligentne višeagentske sustave, koju je razvio TILAB. Inteligencija, inicijativa, informacija, resursi i kontrola se mogu cjelovito distribuirati na mobilne terminale kao i na računala u fiksnoj mreži. JADE se može shvatiti ka agentski middleware koji implementira Agentsku Platformu i razvojnu okolinu. Platforma može evoluirati dinamički sa agentima, koji se pojavljuju i nestaju u sistemu prema potrebama i zahtjevima aplikacijske okoline. Zadaća platforme je da rješava probleme vezane uz transport poruka, dekodiranje i parsiranje podataka te se brine o životnom vijeku agenta. Komunikacija između agenata, bez obzira da li oni rade u bežičnoj ili žičanoj mreži, je potpuno simetrična tako da svaki agent može inicirati komunikaciju i odgovoriti. Svaka instanca JADE-a se zove kontejner (jer sadrži agente). Set svih kontejnera se zove platforma i osigurava homogeni sloj koji od agenata (i od programera aplikacije isto) krije kompleksnost i raznolikost dijelova ispod (hardver, operacijski sustav, vrsta mreže, Java Virtual Machine).

„JADE je cijeli razvijen u Javi i baziran na pogonskim principima:

- uzajamno djelovanje – JADE je suglasan sa FIPA (The Foundation of Intelligent Physical Agents) specifikacijama. Kao posljedica, JADE agenti mogu uzajamno djelovati sa drugim agentima, pod uvjetom da zadovoljavaju iste standarde.
- uniformnost i prenosivost – JADE osigurava homogeni set APIja (Application Programming Interface) koji nisu ovisni o mreži na kojoj se nalazi i Java veziji.
- jednostavnost uporabe – kompleksnost middleware-a je sakriveno iz jednostavnog i intuitivnog seta APIja.
- plaćaj-kako-ideš filozofija – programeri ne moraju koristiti sve mogućnosti middleware-a. Mogućnosti koje nisu korištene ne zahtijevaju programere da znaju nešto o njima.“[2]

[2] Hrvoje Gjurić: „Java Agenti“, seminarski rad



Slika 3. Dijagram toka agentske niti

Unutar JADE agentske platforme nalazi se Java klase koje su nužne za razvijanje programskih agenta i okolina koja osigurava osnovne usluge (komunikacija, migracija, upravljanje životnim ciklusom agenta) .

3. Studijski primjer

3.1 Zahtjevi i problemi prilikom izrade seminara.

Nakon uvodnog opisa agenata, agentskih platformi (općenito), te JADE agentske platforme (na kojoj smo razvijali naše agente), sada ćemo u kratkim crtama opisati naš zadatak odnosno zahtjeve koji su pred nas postavljeni i nekoliko glavnih problema sa kojima smo se susretali.

Naš zadatak je bio da se napravi simulacija tramvajskog prometa i da se napravi usluga koja omogućuje korisniku da sazna kada tramvaj dolazi na određenu stanicu. Tokom izrade zadatka pojavili su se neki problemi, nisu to bili problemi tipa da smo mi nešto napravili pa da to ne radi nego su problemi bili više idejnog tipa koje smo ja i kolega zajedničkim snagama riješili.

Neki važniji problemi su:

- Kretanje tramvaja
- Vrijeme putovanja
- Zapis rute

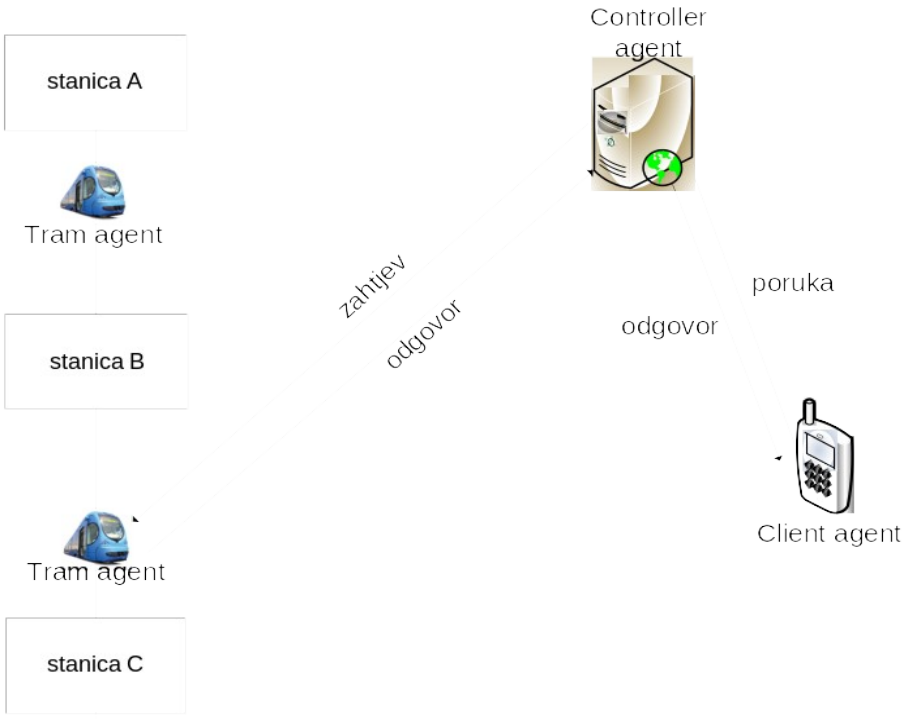
Kretanje tramvaja – od svih važnijih problema ovaj smo najlakše riješili, svaki tramvaj ima svoga agenta, kretanje tramvaja između stanica je jednostavno migracija agenta od jednog do drugog kontejnera. Kontejneri su u ovom slučaju predstavljali stanice.

Vrijeme putovanja – najteži problem od svih, problem se sastojao u tome kako procijeniti koliko je tramvaju potrebno da prijeđe put između dvije stanice, odnosno da se korisniku vrati točno vrijeme do dolaska na stanicu kada korisnik pošalje upit, a ne da se vraća fiksno vrijeme. Npr. potrebno vrijeme tramvaju da prijeđe put između dvije stanice je 10s, i u slučaju da je tramvaj na pola puta, vrijeme koje tramvaj treba vratiti jest 5s.

Zapis rute – na koji će način tramvaji doći do podataka o svojoj ruti kretanja tj. na kojim stanicama moraju kupiti putnike. Prva verzija našeg sustava je bila da se rute budu u kodu, ali to nije dobar načina za zapisivanje rute tramvaja jer svaka promjena rute iziskivala bi promjene u kodu, te smo se u drugoj verziji našeg sustava odlučili za zapisivanje ruta u „xml“ datoteku, te se tramvaju prilikom pokretanja predaje ta datoteka iz koje on pročita stanice po kojima se mora kretati.

3.2 Arhitektura sustava

Da bi isprobali mogućnosti koje nudi JADE odlučili smo napraviti simulaciju tramvajskog sustava. Napravit ćemo više agentski sustav koji će nam pomoći simulirali tramvajski sustav. Glavni cilj ovog studijskog primjera jest omogućiti krajnjem korisniku da sazna kada tramvaj dolazi na određenu tramvajsku stanicu. Korisnik svoj zahtjev šalje tramvajskom središtu i čeka odgovor.



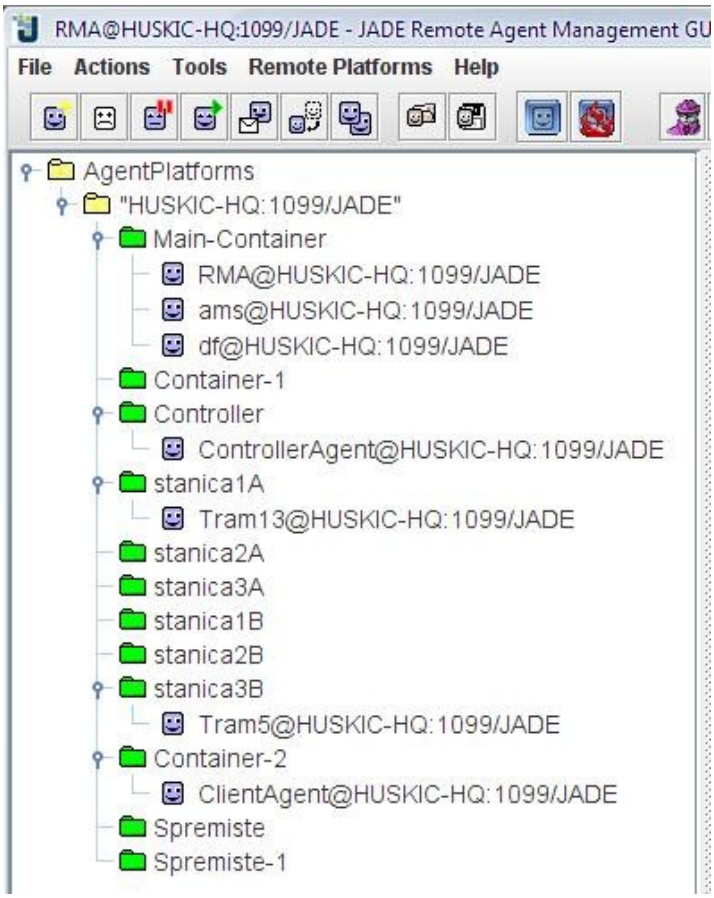
Slika 4. Arhitektura više-agentskog sustava

Tramvajsko središte kada zaprimi zahtjev od korisnika analizira korisnikov zahtjev i na temelju rezultata analize šalje upit prema određenim tramvajima. Kada svi traženi tramvaji jave svoju lokaciju tramvajsko središte obrađuje i sortira podatke i uređene podatke šalje natrag korisniku. Prije nego što se tramvaj pusti u promet on se mora registrirati u tramvajskom središtu.

Arhitektura više-agentskog sustava prikazana je na slici 4. Postoje tri vrste agenata:

- *Client* agent - ovaj agent predstavlja korisnika. Ima grafičko sučelje koje služi da korisnik pošalje svoj zahtjev i prikazuje informacije koje se dobije od tramvajskog središta. Prema zamisli ovog studijskog primjera ovaj agent bi se izvodio na korisničkom mobitelu, ali zbog nedostatka potrebne opreme Client agent će se izvoditi na računalu.
- *Controller* agent – predstavlja tramvajsku centralu. Ona služi za primanje korisničkog zahtjeva, analiziranje korisničkog zahtjeva, kontaktiranje tramvaj agenata, te se na tu centralu svi tramvaji prije početka vožnje prvo moraju registrirati.
- *Tram* agent – predstavlja tramvaj u prometu. Ima zadaću odgovarati na upite *Controller* agenta i kretati se po stanicama. Redoslijed stanica i vrijeme potrebno da se dođe do druge stanice je zapisano u xml datoteci koju *Tram* agent čita u svojoj *setup* metodi. Primjer jednog *Tram* agenta može se vidjeti u dodatku A.

Primijetimo na slici 5 smještaj agenata po kontejnerima. U *Main* kontejneru su smješteni osnovni agenti za upravljanje i podršku po FIPA standardu. U *Controller* kontejneru se nalazi *Controller* agent. Također primijetimo sve kontejnere pod nazivom *stanica***.



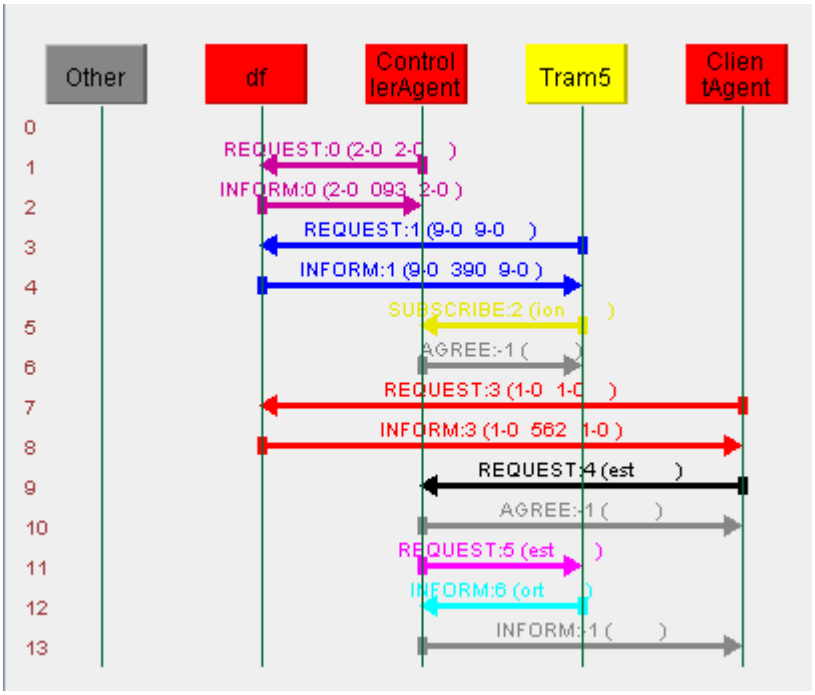
Slika 5. Distribucija agenata u kontejnerima

To su kontejneri koji reprezentiraju tramvajske stanice po kojima tramvaji putuju. Na slici y vidimo da se u kontejnerima stanica1A i stanica3B se nalaze agenti, a u ostalima ne. U kontejneru pod nazivom *Container-2* nalazi se korisnikov agent, dok kontejneri s nazivom *Spremiste* označavaju početnu točku kretanja pojedinog tramvaja.

3.3 Zahtjev za tramvajem

U ovom poglavlju ćemo opisati kako sistem radi i kako se odvija komunikacija između *Controller* agenta, *Tram* agenta i *Client* agenta. Da bi omogućili uslugu zahtjeva za tramvajem potrebno je pokrenuti *Controller* agenta i *Tram* agente. *Controller* agent pri svom pokretanju prima kao argument *xml* datoteku u kojoj se nalazi popis svih stanica. Ovo je za uspješan rad usluge, kasnije će biti objašnjeno i zašto. Potom *Controller* mora registrirati svoju uslugu kod *DF*-a (*engl. Directory Facilitator*). Kada se uspješno registrira usluga moguće je pokrenuti proizvoljan broj *Tram* agenata koji prvo traže uslugu od *DF*-a pomoću *REQUEST* poruke i dobivaju natrag popis agenata koji pružaju traženu uslugu. Sada se *Tram* agenti mogu registrirati kod *Controller* agenta. Tramvaji se registriraju koristeći *SUBSCRIBE* poruke. U *SUSCRIBE* poruci agent mora poslati svoje podatke, a to uključuje svoju rutu, vrijeme putovanja između stanica, broj tramvaja i svoj identifikacijski broj.

Controller agent provjerava da li stanice po kojoj se kreće tramvaj postoje i ako su svi podaci valjani šalje se *AGREE* poruka natrag *Tram* agentu koja signalizira agentu da je prošao provjeru i da se može početi kretati svojoj predviđenom rutom. *Controller* agent u sebi sadrži popis svih stanica i za svaku stanicu ima naznačeno koji sve tramvaji prolaze stanicom. Također ima spremljene sve podatke o svakom tramvaju. Ako bi tramvaj imao stanicu koja se ne nalazi kod *Controller*-a onda registracija tramvaja ne bih uspjela i tramvaj bi dobio *REFUSE* poruku. Tramvaji se kreću po poznatim stanicama i nije im dozvoljeno uvoditi nove stanice.



Slika 6. Redoslijed komunikacije

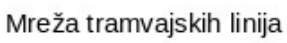
Prilikom pokretanja *Client* agenta, *Client* agent prvo mora potražiti uslugu u *DF*-u. Kada dobije uslugu *Client* agent putem svojeg grafičkog sučelja upisuje ime svoje stanice i to ime se prenosi u *REQUEST* poruci *Controller* agentu. *Controller* agent provjerava ispravnost korisničkog zahtjeva i šalje *AGREE* poruku ako je korisnikov zahtjev ispravno napisan. Korisnikov agent čeka odgovor od *Controller* Agenta. Dok korisnik čeka odgovor *Controller* u svojem popisu stanica traži korisničku stanicu. Kada nađe takvu stanicu pogleda koji sve tramvaji idu tom stanicom i svima šalje *REQUEST* poruku u kojoj navodi da mu se da lokacija tramvaja. Tramvaj u svojoj *INFORM* poruci odgovara sa imenom svoje prošle stanice i vremenom koji je proveo na putu prema sljedećoj stanici. *Controller* agent čeka dok ne dobije odgovore od svih tramvaja, potom izračunava vrijeme do korisničke stanice i sortira rezultate po rastućim vremenima. U svojoj *INFORM* poruci prema *Client* agentu *Controller* šalje izračunate podatke.

Implementirani studijski primjer dobro prikazuje osnovnu ideju usluge traženja informacije o dolasku tramvaja na određenu stanicu pomoću agenata. Međutim ima još puno mjesta za napredak studijskog modela. Kao budući rad može se izvesti složenija simulacija tramvajskog prometa. To znači uvesti mogućnost zastoja tramvaja, da na stanici smije biti neki određeni broj tramvaja, a svi ostali tramvaji koji dolaze na stanicu moraju provjeri da li mogu pristupiti stanici što na kraju utječe na vrijeme dolaska na određenu stanicu. Simulirati raskrižja, te one dionice pruge koje tramvaji dijele s ostalim sudionicima prometa što ponovno u određenom dobu dana (kada većina ljudi ide na posao ili se vraća s posla) bitno utječe na brzinu tramvaja i njegovo vrijeme dolaska do sljedeće stanice. Kod samog

ponašanja tramvaja kao poboljšanje bi trebalo uvesti da se vrijeme čekanja da putnici izađu ili uđu ne utječe na vrijeme između stanica, tj. da vrijeme kreće tek onda kada tramvaj stvarno krene. Moguće poboljšanje bi bilo i napraviti server na kojem bih bio pokrenuta tramvajska središnjica koje bi onda primao upite. Moguće bi bilo i prebaciti korisnika na stvarni mobitel. Postoje već pokretni agenti koji se mogu staviti na mobitele ili PDA (JADE LEAP). Znači implementirati agenta na mobitelu koji bi bežično komunicirao prema tramvajskoj središnjici, možda i implementirati GUI kod klijenta koji će pokazivati gdje se točno tramvaj nalazi.

4. Testiranje sustava

Nakon uspješne faze razvoja sustava, uslijedilo je testiranje sustava. Napravili smo četiri testa na sustavu, te na osnovu rezultata testiranja zaključili da sustav ispravno radi. Testiranje sustava će se provesti prema modelu prikazanim na slici 7. Dodatne informacije o tramvajima br.5 i br.13 pogledajte u Dodatku A.



Slika 7. Testni model

Vrijeme potrebno tramvaju da dođe do druge stranice biti će uvijek isto, ali to vrijeme može biti specifično za bilo koji put između dvije stanice. Mi ćemo upotrebljavati isto vrijeme zbog jednostavnosti.

Kada se pokrene simulacija, otvara se korisničko sučelje i korisnik upisom podataka unutar tekst polja (stanica i broj tramvaj – opcionalna mogućnost) i pritiskom na „Send message“, šalje zahtjev da mu se dostavi vrijeme dolaska tramvaja za traženu stanicu, ako su podatci ispravni tj. stanica postoji u sustavu i traženi tramvaj vozi tom stanicom (ako je upisan broj tramvaja prilikom slanja zahtjeva),

korisniku stiže poruka sa vremenima dolaska tramvaja, ako su podatci neispravni dobiva se poruka o pogrešci.

1. Unos ispravnog naziva stanice i nepostojećeg broja tramvaja (voze samo tramvaji 5 i 13)

Slika 8. Test1

2. Unos ispravnog naziva stanice (Stanicom3A voze tramvaji 5 i 13- pogledati testne podatke)

Slika 9. Test2

3. Unos ispravnog naziva stanice i unos neispravnog broja tramvaja (uneseni tramvaj ne prolazi zadanom stanicom- pogledati testne podatke)

Slika 10. Test3

4. Unos ispravnog naziva stanice i broja tramvaja

Slika 11. Test4

5. Literatura

[1] http://www.carnet.hr
[2] http://jade.tilab.com/
[3] Hrvoje Gjurić : „Java Agenti“ seminarski rad
[4] Lovrek, Ignac: Pokretni programski agenti, bilješke s predavanja predmeta *Telematičke usluge*, 2006, FER/ZZT

DODATAK A

Podaci o tramvaju br.13 iz testnog primjera

```
<tram>
 <tram_info number="13" first_station="stanica3A">
  <station current="stanica1A" next="stanica3A" travelling_time="10" />
  <station current="stanica3A" next="stanica3B" travelling_time="10" />
  <station current="stanica3B" next="stanica1B" travelling_time="10" />
  <station current="stanica1B" next="stanica1A" travelling_time="10" />
   </tram_info>
 </tram>
```

Podaci o tramvaju br.5 iz testnog primjera

```
<tram>
  <tram_info number="5" first_station="stanica1A">
  <station current="stanica1A" next="stanica2A" travelling_time="10" />
  <station current="stanica2A" next="stanica3A" travelling_time="10" />
  <station current="stanica3A" next="stanica3B" travelling_time="10" />
  <station current="stanica3B" next="stanica2B" travelling_time="10" />
  <station current="stanica2B" next="stanica1B" travelling_time="10" />
  <station current="stanica1B" next="stanica1A" travelling_time="10" />
  </tram_info>
</tram>
```