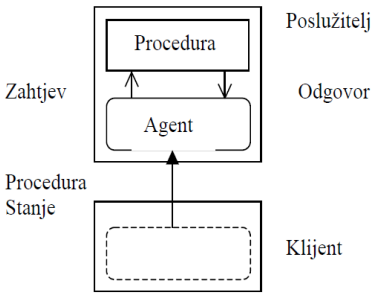
AGENT - entitet u mreži koji samostalno obavlja skup poslova zadanih od strane korisnika, agenti se dijele na ljudske, sklopovske i programske

SVOJSTVA AGENATA:

- Samostalnost -> sposobnost agenta da djeluje bez dodatnih intervencija korisnika ili drugih agenata, te da kontrolira vlastite akcije i unutarnje stanje ili stanja
- Sposobnost suradnje -> agenti su sposobni surađivati s drugim agentima, agenti koji surađuju nazivaju se *kooperacijski* agenti
- Inteligencija - > sposobnost učenja i odlučivanja na temelju podataka koje agent dobiva iz okoline
- Pokretljivost -> sposobnost kretanja agenta u mreži, prilikom kretanja agent zadržava trenutno stanje (vrijednost varijabli, programski kod, liniju koda od koje se počinje nakon premještanja)
- Reaktivnost -> sposobnost reagiranja na utjecaje iz okoline
- Proaktivnost -> usmjerenost cilju i poduzetnost bez stalnih kontakata s okolinom
- Fleksibilnost -> omogućava agentu prilagodbu ponašanja tijekom izvođena

POKRETNI PROGRAMSKI AGENTI – program koji može migrirati s čvora na čvor i pritom može obavljati poslove u korist korisnika ili vlasnika, poslove obavlja samostalno ili u interakciji s korisnikom, te komunicira s ostalim agentima u mreži, sa sobom prenosi kod, podatke i stanje izvođenja. On ujedinjava pristupe udaljenog programiranja. To je program koji se šalje od klijenta poslužitelju i ne vraća se odmah klijentu nego se kreće i zadržava stanje



Prednosti PPA su manje prometno opterećenje, samostalno izvršavanje, asinkrona komunikacija, smanjenje zahtjeva za korištenjem klijentskih resursa, konkurentnost izvođenja, Veća pouzdanost i fleksibilnost, distribuirano izvođenje.

Mane PPA su sigurnost, standardizacija, nekompatibilnost agentskih sustava, povećano opterećenje poslužitelja.

VRSTE AGENATA

- Informacijski – bave se informacijama, iz jednog ili više izvora. Njihova je zadaća dobivanje, skupljanje, filtriranje i odabir informacija. Ako informacije dolaze iz više nezavisnih izvora, potrebno ih je analizirati, obraditi te objediniti.
- Kooperacijski – rješavaju složene probleme grupnim radom, te sudjeluju u upravljanju komunikacijskih, mrežnih i programskih resursa

- Transakcijski – sudjeluju u poslovnim procesima, omogućuju obavljanje transakcija i nadgledaju elektroničko poslovanje

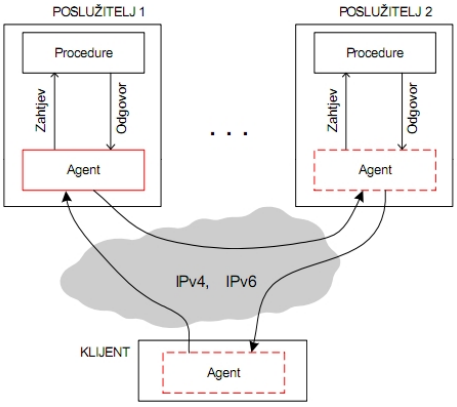
PODJELA AGENATA (po strukturi)

- Agenti sučelja - prate navike korisnika i prilagođavaju se, te pretpostavljaju događaje i akcije i olakšavaju korisniku upravljanje sustavom
- Reaktivni agenti - djelovanje ovisi o trenutnom stanju, ne sadrže logiku
- Inteligentni agenti - automatski izvršavaju postavljeni zadatak bez intervencije korisnika
- Hibridni agenti

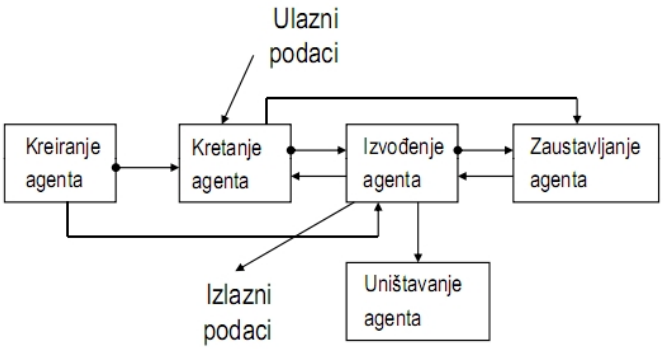
STRUKTURA POKRETNOG AGENTA

- pokretni agent se sastoji od:
  - Identifikatora vlasnika (ID)
  - Oznake čvora
  - Programskog dijela kojeg čini program (kod) izvođenja
  - Podatkovnog dijela kojeg čine podaci

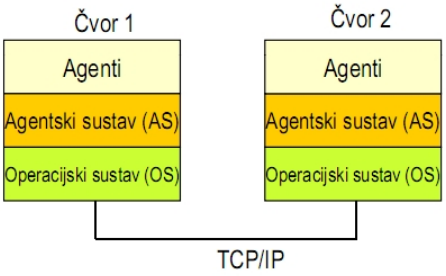
NAČIN RADA AGENTA - može se pokazati skicom:



ŽIVOTNI CIKLUS AGENTA – opisujemo ga slikom



AGENTSKI SUSTAV – agentski sustav je okružje u kojem se izvode agenti i u kojem se upravlja agentima, u njemu se agenti stvaraju pokreću i ukidaju. Raspodijeljeni agentski sustav prikazan na donjoj slici služi za programiranje raspodijeljenih aplikacija, programiranje složenih aplikacija, raspoređivanje opterećena, promatranje događaja….



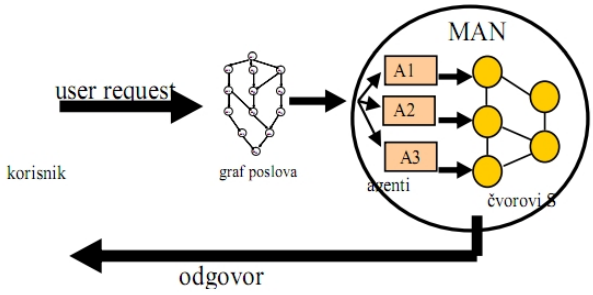
AGENTSKA PLATFORMA – sastoji se od agentskog sloja, sigurnosnog sloja i komunikacijskog sloja.

Agentski sloj brine se o upravljanju agentima, tj. o njihovom kreiranju, kretanju i izvođenju.

Sigurnosni sloj sadržava razne sigurnosne mehanizme među kojima su sigurna komunikacija, sigurna migracija, vatrozid, digitalni potpis, zaštita podataka šifriranjem….

Komunikacijski sloj je zadužen za komunikaciju s ostalim agentima i prijenos agenata. U ovom se sloju odvija slanje i primanje podataka te se nalazi protokol za poruke i definicija njihovog formata.

VIŠEAGENTSKI SUSTAV - agentski sustav u kojem se posao koji treba obaviti može raspodijeliti na više agenata. Takvi sustavi omogućavaju brže postizanje cilja, smanjenu komunikaciju s okolinom, skalabilnost, fleksibilnost i modularnost te robustnost. Veći broj agenata zahtjeva suradnju i komunikaciju među njima, pokretljivost, koordinaciju (zbog ostvarivanja zajedničkog cilja) te organizaciju (u grupe ili timove).

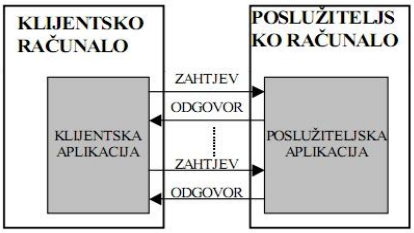


Višeagentski sustavi se dijele na:

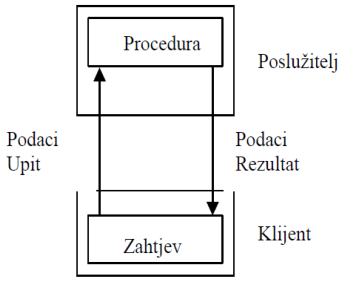
- Heterogeni sustav s nekomunicirajućim agentima -> različiti agenti
- Heterogeni sustav s komunicirajućim agentima -> složeni sustavi
- Homogeni sustav s nekomunicirajućim agentima -> isti agenti sa zajedničkim ili vlastitim ciljem, različite pobude izazivaju različito ponašanje
- Homogeni sustav s komunicirajućim agentima ->isti agenti sa zajedničkim ciljem

PROGRAMSKI MODELI AGENATA:

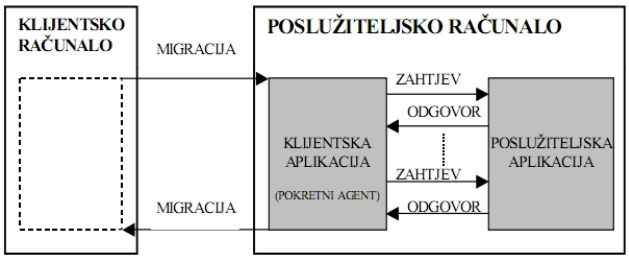
- Klijent-poslužitelj -> udaljena interakcija u kojoj klijent šalje zahtjev kojeg poslužitelj obrađuje i šalje klijentu odgovor, šalje se veliki broj poruka kroz mrežu. U ovu arhitekturu pripada *poziv udaljenih procedura*



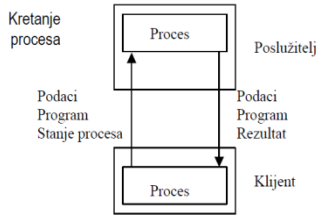
- Poziv udaljene procedure - Omogućava s jednog računala poziv procedure na drugom računalu. Klijent šalje zahtjev poslužitelju za obradom. Komunikacija unutar ove arhitekture je sinkrona.



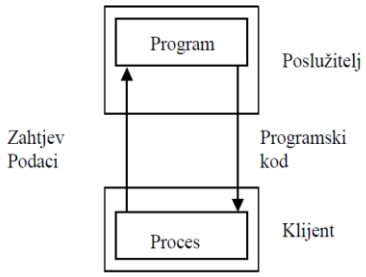
- Udaljeno programiranje -> nadogradnja klijent-poslužitelj arhitekture gdje klijentska aplikacija migrira na poslužitelja i time se udaljena interakcija zamjenjuje lokalnom. U ovu arhitekturu pripadaju *udaljeno izvršavanje* i *kod na zahtjev*



- Udaljeno izvršavanje -> ne poziva udaljenu proceduru nego klijent šalje proceduru koja se treba izvesti, šalje se kod procedure poslužitelju koji izvodi proceduru i vraća odgovor



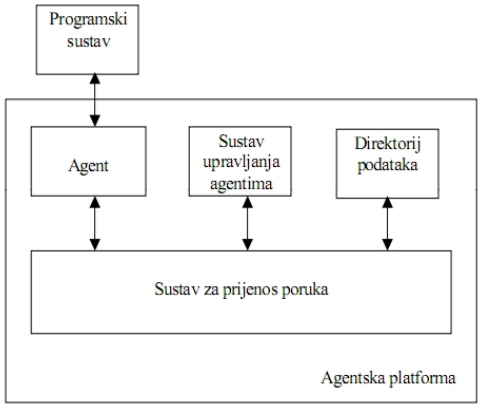
- Kod na zahtjev -> predstavlja raspodijeljeni repozitorij programa, klijent ne sadrži unaprijed izvršivi kod nego šalje zahtjev poslužitelju koji mu vraća kod programa kojeg klijent kada primi izvršava i nakon izvršavanja briše



IEEE FIPA - neprofitna organizacija koja standardizira upravljanje agentima, komunikaciju (porukama ACL), imenovanje, sigurnost, pokretljivost, interakciju, jezik sadržaja, prijenos poruka itd.

Njihova specifikacija upravljanja agentima obuhvaća registraciju agenata, izmjenu poruka, životni ciklus te agentsku platformu.

Agentska platforma, prema specifikacijama FIPA-e, izgleda ovako:



Komponente koje su dio platforme su:

- AMS (sustav upravljanja agentima),
- DF (direktorij podataka) i
- MTS (sustav za prijenos poruka).

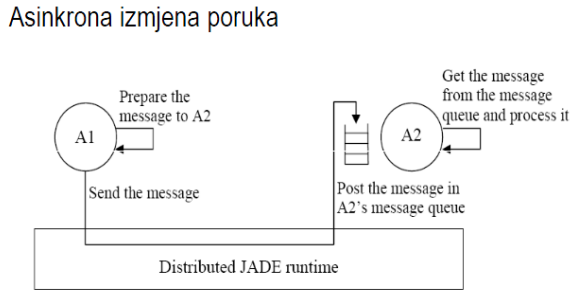
AMS predstavlja sustav za upravljanje agentima i obvezna je komponenta agentske platforme. Ova komponenta upravlja životnim ciklusom agenata, kontrolira pristup, kreira, briše, identificira i migrira agente. U AMS se agenti mogu prijaviti, odjaviti, promijeniti njihovi podaci te ih pretraživati.

DF predstavlja direktorij i također je obavezna komponenta agentske platforme. U direktoriju se nalaze liste agenata s podacima o njima samima i informacijama o uslugama koje pružaju.

MTS je sustav za prijenos poruka i omogućuje komunikaciju s drugim agentima na drugim platformama pomoću poruka ACL putem komunikacijskog kanala ACC. Kanalom ACC se ostvaruje interakcija agenata, a koristi se protokol MTS za prijenos poruka.

ACC – komunikacijski kanal koji služi za interakciju agenata. On prosljeđuje i usmjerava poruke između agenata. Prenosi MTS poruke.

KOMUNIKACIJA U JADE-u – komunikacija je asinkrona. Agent koji šalje poruku priprema i šalje poruku koja se stavlja u red poruka agenta koji prima poruku. Taj agent dohvaća poruke iz reda poruka jednu po jednu redoslijedom kako su pristigle u njegov red poruka.



IDENTIFIKACIJA AGENATA – za identifikaciju se koristi:

- AID (*Agent identifier*) ->domaća platforma na kojoj je agent kreiran zove se HAP (*Home Agent Platform*) sadrži AMS odgovoran za stvaranje agenta i Global Unique Identifier koji se sastoji od (<name>@<hostname>:<port>/<target>):
  - Name -> jedinstveno ime agenta unutar HAP
  - Hostname -> IP adresa na kojem se nalazi
  - Port-> broj priključne točke na kojoj se osluškuje
  - Target -> identifikacija agenta koji prima poruku
- Agent-id – jedinstvena identifikacija agenta
- Agent-type – tip agenta koji određuje koje funkcije agent može izvoditi
- Personal-key – veza agenta s korisnikom

IMENOVANJE AGENATA – agente možemo imenovati na 3 načina:

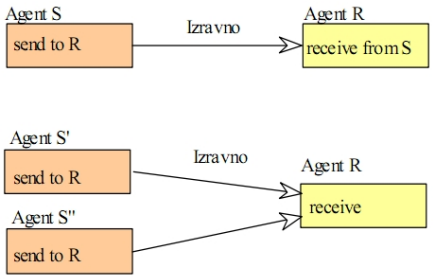
- Lokacijski ovisno imenovanje -> temelji se na imenu čvora (poslužitelja) i broju pristupne točke (porta), a ime se mijenja ovisno o promjeni lokacije
- Lokacijski transparentno imenovanje -> ovija trenutnu lokaciju udaljenih entiteta korištenjem lokalnog poslužitelja/posrednika (proxy) te zahtijeva sposobnost pridruživanja simboličkih imena trenutnoj lokaciji imenovanog entiteta
- Lokacijski neovisno imenovanje -> bez obzira na promjenu lokacije (čvora) ime entiteta je uvijek isto, koristi uslugu imenika

Imenovanje agenata može biti i:

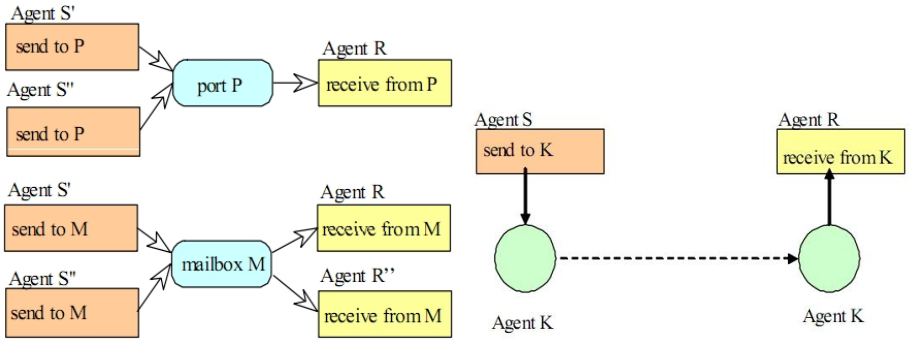
- Izravno
  - Simetrično -> agent imenuje onog drugog
  - Asimetrično -> agent prima poruku od bilo kojeg agenta

- Posredno
    - Agent posrednik
    - Vrata (port) -> više agenata komuniciraju s jednim agentom
    - Poštanski sandučić (mailbox) -> više agenata komunicira s više agenata

Izravnim imenovanjem se adresira točno određeni agent koji će primati poruke. Simetričnim imenovanjem se poruka šalje 1:1, a asimetričnim jedan agent prima poruku od više njih (n:1).

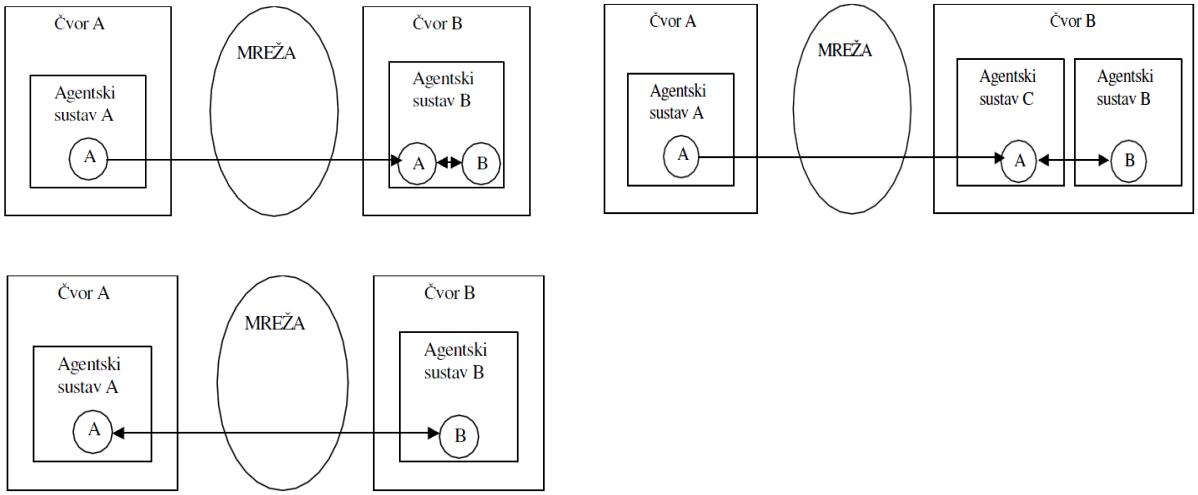


Posredno imenovanje koristi posrednika. U tu svrhu može koristiti dodatni agent preko kojeg će poruke stizati, te vrata preko kojih će se poruka dostavljati ili poštanski sandučić s kojeg poruke može uzimati više agenata.

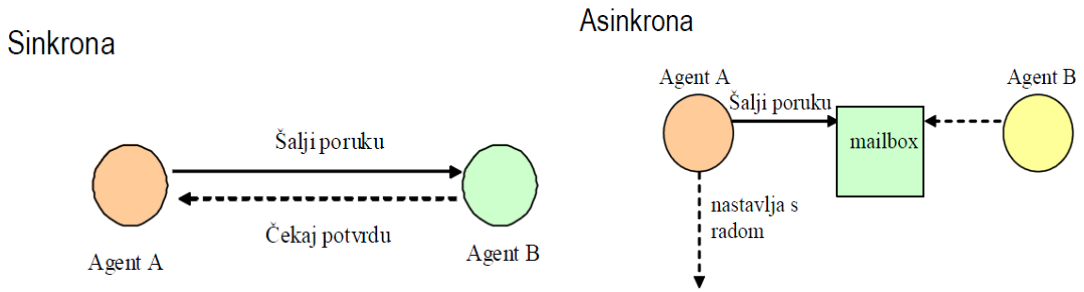


POKRETLJIVOST AGENTA – pokretljivost agenta označava sposobnost kretanja agenta, dijeli se na jaku i slabu pokretljivost. Jaka pokretljivost označava prenošenje stanja i programskog koda prilikom migriranja agenta, uključuje prenošenje i izvršavanje prenesenog koda. Slaba pokretljivost označava ne prenošenje programskog koda nego prenošenje reference na programski kod. Kretanje agenta može biti proaktivno i reaktivno. Proaktivno kretanje je kretanje koje agent određuje sam. Reaktivno kretanje je kretanje koje određuje okolina (korisnik ili neki drugi agent).

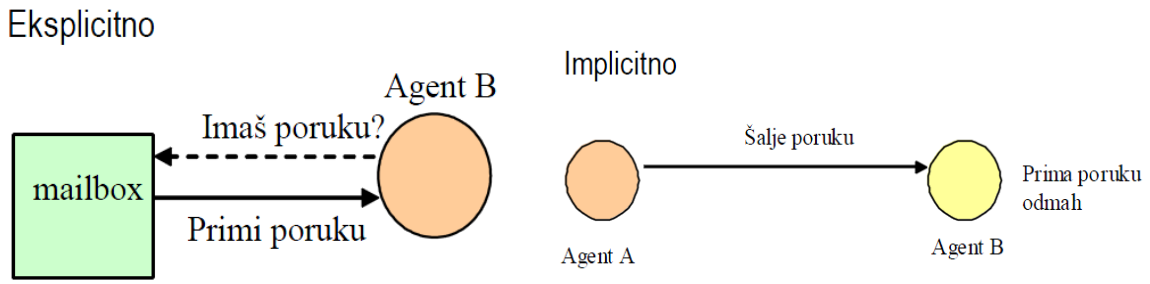
MASIF (Mobile Agent System Interoperability Facility) – koristi se za međudjelovanje različitih platformi. Agenti mogu imati isti agentski profil, podržavati zajednički profil ili ne podržavati zajednički profil:



IZMJENA PORUKA – može biti sinkrona i asinkrona. Sinkrona izmjena poruke podrazumijeva čekanje odgovora dok se asinkrona izmjena poruka oslanja na ostavljanje poruke (obično u poštanskom sandučiću) i normalnom nastavku rada pošiljatelja.



PRIJAM PORUKE – prijam poruke može biti eksplicitni (posredan) i implicitni (izravan). Eksplicitni prijem poruke zahtjeva upit o postojanju poruke (obično poštanskom sandučiću pa je pogodan kod asinkrone izmjene poruka) dok je implicitni prijem poruke standardni izravni prijem poruke od nekog pošiljatelja.



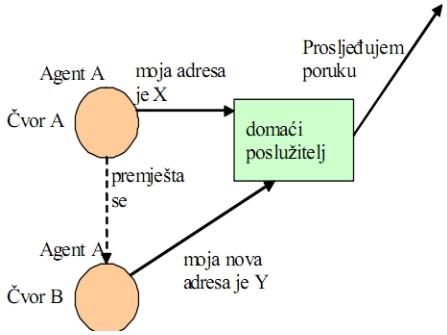
ADRESIRANJE AGENTA – adresiranje može biti:

- Asimetrično izravno -> lokalna komunikacija

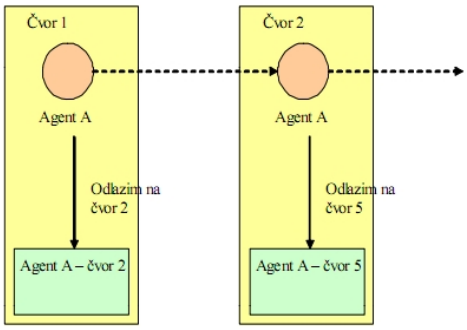
- Posredno -> udaljena komunikacija
- Sinkrono ili asinkrono -> ovisno o namjeni

PRAĆENJE PUTA – odvija se na tri načina:

- Domaći poslužitelj -> sadrži podatke o trenutnim adresama agenata. Oslanja se na agentovo dojavljivanje svoje trenutne lokacije svom domaćem poslužitelju. Sve poruke, namijenjene udaljenom agentu, šalju se njegovom domaćem poslužitelju koji ih onda prosljeđuje agentu. Prednost ovog pristupa je u tome što je lako izvediv, ali su nedostaci veliki mrežni promet (svako seljenje dojavljivanje lokacije), nedostizanje poruke pri brzom kretanju agenta te trokutasto usmjeravanje.

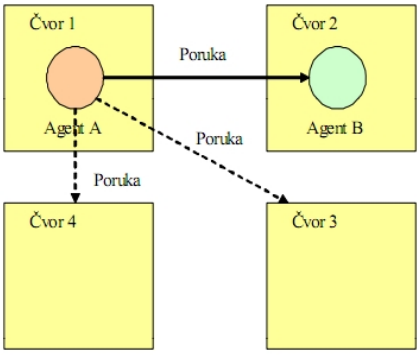


- Praćenje traga -> zasniva se na agentovom zapisivanju svoje sljedeće lokacije na trenutni čvor. Tako se može, praćenjem čvor po čvor doći do agenta. Prednost je što više nema prijenosa poruka o lokaciji i što je pristup decentraliziran. Nedostatak je to što se poruka mora kretati od čvora do čvora i što se agent ne može pratiti ako se prebrzo kreće.



-

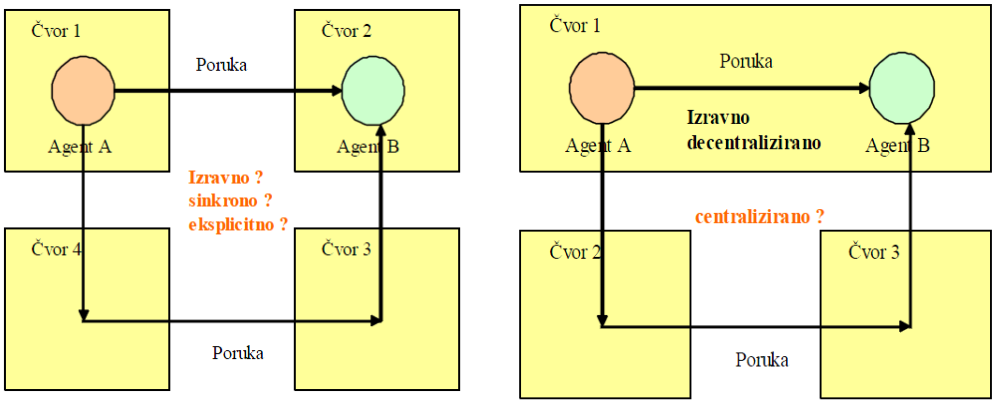
- Višeodredišni pristup -> Višeodredišni pristup se temelji na razašiljanju višeodredišne poruke pri čemu se mora znati uže područje kretanja agenta kojem se šalje poruka. Prednost je što se poruka brzo dostavlja i što se može obavještavati više agenata. Nedostatak je opterećenje mreže porukama.



VRSTE AGENTSKE KOMUNIKACIJE – agentska komunikacija može biti:

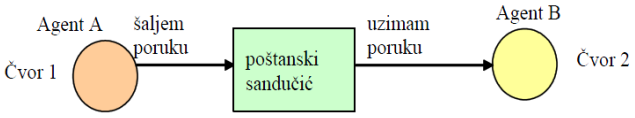
- Udaljena
- Lokalna
- Posredna

Udaljena i lokalna komunikacija su izravne.

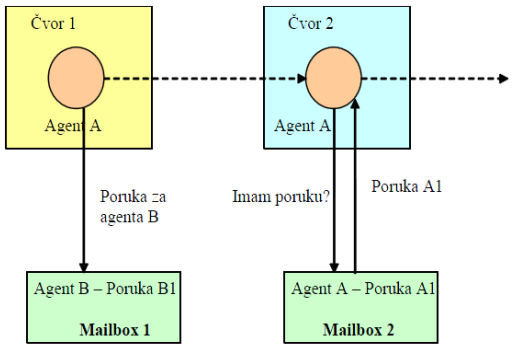


Posredna komunikacija može biti:

- Poštanski sandučić -> dijeli se na
    - Centralizirani poštanski sandučić -> uobičajeni sandučić u kojeg agenti ostavljaju i iz kojeg čitaju poruke. Prednost je što je takva komunikacija vremenski neovisna, a nedostatak je dugo trajanje komunikacije.

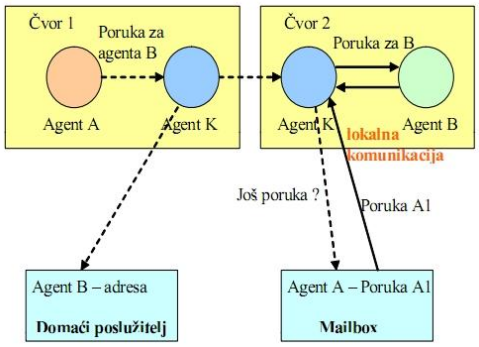


    - Poštanski sandučić na svakom čvoru -> pri dolasku na svaki novi čvor agent iščitava poruke koje su poslane. Problem se javlja ako je čvor nedostupan ili ako je sandučić postao pun jer agent ne dolazi na čvor.



- Pokretljivi poštanski sandučić -> sandučići se kreću zajedno s agentom. Loše rješenje jer je problem praćenje puta. Drugi način je dinamičko određivanje premještanja gdje sandučić dolazi na čvor na kojem je agent kada treba poruku, problem je praćenje sandučića

- Treći (komunikacijski) agent -> komunikacijski agent brine o komunikaciji, koja se odvija s više agenata. Komunikacija je asinkrona. Udaljena komunikacija se pomoću njega svodi na lokalnu. Prednost su brzina i pregovaranje.



Jezik ACL definira vrstu poruke, tj. njenu strukturu i značenje poruke. Tim je jezikom definiran skup unaprijed definiranih vrsta poruka i protokola. Poruke koje se šalju su poruke ACL.
Struktura ACL poruke je takva da poruka počinje s vrstom komunikacijske akcije te se nastavlja u izraz sadržaja poruke. Taj sadržaj poruke predstavljen je nizom parova oblika ":parametar vrijednost" (npr. :sender agent1).
Službena standardizirana poruka ACL (standardizirana od strane FIPA-e) sadrži sljedeće dijelove:

- tip komunikacijskog djelovanja (tj. vrstu poruke, njenu svrhu u komunikaciji, a to mogu biti zahtjev, potvrda, informacija, greška itd.),
- sudionici komunikacije (pošiljatelj, primatelj i povratna adresa),
- sadržaj poruke,
- opis sadržaja poruke (jezik poruke, način kodiranja i ontologiju) i
- kontrola komunikacije (protokol, identifikator komunikacije, izraz raspoznavanja, prethodno korišteni izraz raspoznavanja i vremensko ograničenje odgovora).

Primjer: poruka inform
(inform

:sender (agent-identifier :name mali-ivica)
:receive (agent-identifier :name zavod-za-telekomunikacije)
:content "slajdovi(umiremo, sad)"
:language Prosti:))

FIPA ACL PROTOKOLI – dijele se na:

- FIPA-request interaction protokol -> protokol slanja zahtjeva između dva agenta radi izvršenja neke akcije te primanje odgovora, ako je to moguće,
- FIPA-query interaction protokol -> definira slanje upita drugom agentu
- FIPA-request-when interaction protokol -> definira slanje zahtjeva drugom agentu radi izvođenja neke akcije uz određeni uvjet
- FIPA-contract-net interaction protokol -> protokol u kojem jedan agent uzima ulogu upravljanja s jednim ili više agenata

ARHITEKTURE AGENATA:

- Reaktivni agenti -> izlazna akcija nastaje na temelju ulaza
- Logički agenti -> temelje se na logičkom odlučivanju
- Uvjerenje-želja-namjera agenti (BDI) -> vrše manipulaciju sa strukturama podataka koje predstavljanju uvjerenja, želje i namjere agenata
- Slojeviti agenti -> zaključuje se na različitim razinama apstrakcije

Reaktivni agenti reagiraju na podatke iz okoline. Prednost im je jednostavnost i mogućnost praćenja računanja. Veliki je nedostatak što kod njih nema učenja. Sve svoje odluke donose na temelju ugrađenih ponašanja po principu automata stanja.

Logički agenti rade logičkom dedukcijom i dokazivanjem teorema. Nedostatak im je rad u stvarnom vremenu i pretpostavka da se okolina ne mijenja tijekom odlučivanja.

Arhitektura uvjerenje-želja-namjera temelji se na praktičnom zaključivanju. Odlučuje se koje ciljeve treba postići i na koji način. Faze zaključivanja su promišljanje i svrhovito zaključivanje. Svrhovito zaključivanje proizlazi iz namjera, a namjere ograničavaju buduće promišljanje. Uvjerenja agenta predstavljaju znanje koje on ima. Želje predstavljaju skup ciljeva koje želi postići (zadaci), a namjera je želja koju agent pokušava trenutno provoditi. Agent pokušava ostvariti namjeru. Ako je ne ostvari prelazi na sljedeću želju i tako radi sve dok ne prođe sve želje.

Slojevita arhitektura pretpostavlja rastavljanje problema na dijelove. Svaki dio predstavlja određeno ponašanje, tj. sloj. S obzirom na razmjenu kontrolnih podataka moguće je napraviti horizontalnu ili vertikalnu podjelu. Kod horizontalne podjele svaki sloj je predstavljen jednim agentom i ima svoj ulaz i izlaz.

STRUKTURE ORGANIZIRANJA AGENATA:

- Potpuna -> svaki agent zna sposobnosti ostalih
- Centralizirana -> jedan agent zna sposobnosti ostalih i distribuira informacije ostalima

- Raspodijeljena -> agenti znaju svoje sposobnosti i na koji način dobiti informacije o ostalima

MODELI KOORDINACIJE AGENATA:

- Kooperativnost -> agenti se podupiru
- Kompetitivnost -> agenti se natječu
- Osnovni modeli:
    - Mravinjak :
        - Pojedinačan rad agenata - agenti dobivaju male jednostavne zadaće i imaju male mogućnosti, ne komuniciraju, nisu organizirani i ne snalaze se u prostoru
        - Skupni rad agenata – agenti su dobro organizirani, zajedno pronalaze optimalan put
    - Nadređeni-podređeni -> nadređeni agent upravlja podređenim agentima. Poslovi se izvode paralelno i postoji stalna komunikacija između podređenih i nadređenih.
    - Timovi agenata -> agenti mogu imati različite uloge. Postoje dvije teorije kod ovog modela:
        - Teorija združenih namjera – temelji se na arhitekturi uvjerenje-želja-namjena
        - Teorija dijeljenog plana – postoji dogovor o zajedničkom planu gdje su definirana pravila ponašanja te individualne i zajedničke namjere

Primjene agenata

Područja primjene agenata:

- Osobni agenti -> prikupljanje informacija i djelovanje u ime vlasnika
- Višeagentski sustavi za odlučivanje -> npr. pregovatanje u telekomunikacijskoj vezi
- Višeagentski simulacijski sustavi -> računalne mreže, igre, upravljanje prometom…

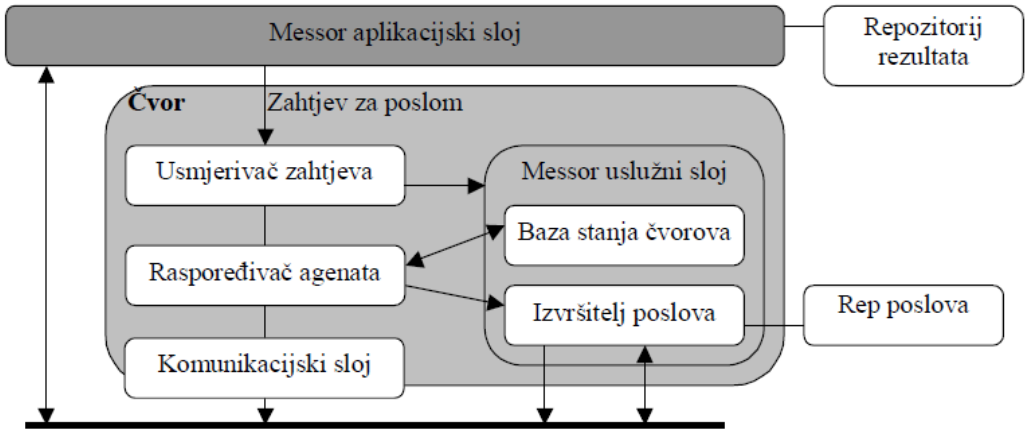
Agenti u uravnoteženju opterećenja -> autonomni programski entiteti s mogućnošću samostalnog kretanja koji u ime vlasnika obavljaju poslove prikupljanja informacija, podešavanja rada stacionarnih agenata i izvršavanje dinamičkih metoda za uravnoteženje. Prednosti korištenja ovih agenata su lokalna obrada podataka, omogućavanje komunikacije za sinkronizaciju i kontrolu, osiguranje prenosivosti između platformi i zamjena agenta višeagentskim sustavom ako je to potrebno.

Sustavi inspirirani ekonomskim tržištem -> vrše kontrolu korištenjem elektroničke gotovine gdje preopterećeni čvorovi s visokim cijenama tjeraju korisnike.

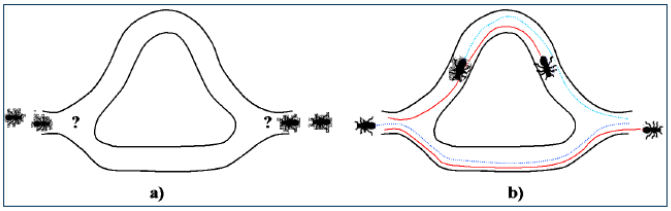
- *Challenger* -> višeagentski sustav za raspodijeljenu kontrolu korištenja resursa koji djeluje po principu licitacije. Njegova prilagodljivost je riješena s mogućnošću učenja prema vrijednostima kašnjenja u mreži i procjeni vremena potrebnog agentu da izvrši posao.

Sustavi inspirirani biološkim sustavima -> sustavi sa svojstvima samoorganizacije prilagodljivosti i elastičnosti gdje su elementi jednostavni, a funkcioniranje skupine složeno.

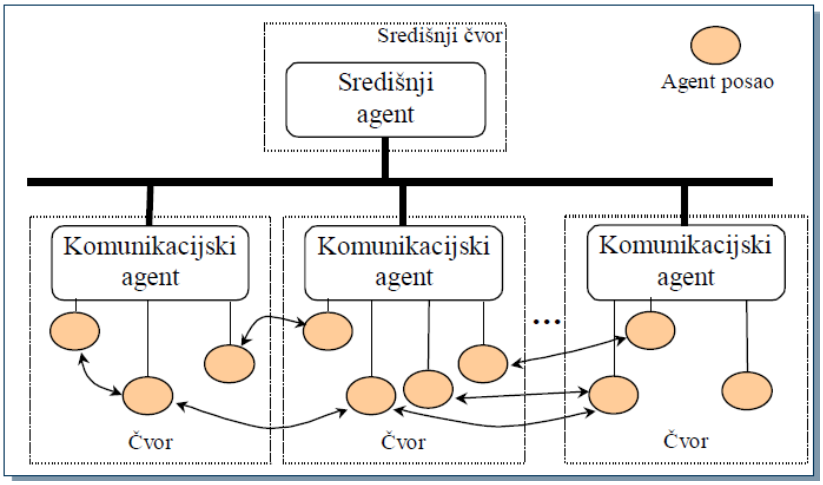
- *Messor* -> višeagentski sustav gdje agenti pokušavaju raširiti posao na sve čvorove, oni lutaju mrežom dok ne nađu veće količine posla



- *AntNet* -> usmjeravanje agenata korištenjem višeagentskog sustava gdje je ponašanje prirodnog sustava pretočeno u računalni. Postoje F (Forward) i B (Backward) agenti.



Višeagentski sustavi za uravnoteženje -> sadrže središnji čvor (Agent) koji je zadužen za uravnoteženje agentskog posla u mreži između ostalih agenata.



Programiranje agenata u Erlangu

- Postoje dva zahtjeva za izvedbu inteligentnih agenata:
  - Izvedba ponašanja agenata (JADE)
  - Izvedba inteligencije agenata (JADE+JESS)
- Erlang je programski jezik koji zadovoljava oba uvjeta, on je funkcionalni jezik prenosiv na različite platforme, ima dobre performanse, podržava konkurentnost, raspodijeljenost i kompletnost
- eXAT -> Erlang eXperimental Agent Tool koji se sastoji od:
  - ERES modula -> modul za procesiranje pravila namijenjen za razvoj ekspertnih modula i inteligentnih agenata koji sadrži bazu znanja sa skupom spoznaja/činjenica. Pravila se pišu kao uvjeti funkcija, a pri pozivu se vrši provjera da li se uzorak nalazi u bazi.
  - ACL modula -> služi za kodiranje i dekodiranje ACL poruka, dekodiranje sadržaja poruke i transport poruka. Elementi poruke se uspoređuju s atomom, specijalnim simbolom _(označava bilo što) ili lambda funkcijom za kompleksnije usporedbe.
  - Modula za izvršenje agenata -> svaki agent sadrži ponašanje (automat stanja) i skup parametara u obliku {ime, vrijednost}. Sustav omogućuje prijelaze iz stanja u stanje pod utjecajem događaja, a programer mora definirati akcije za svaki par {stanje, događaj}.

Objekti u eXAT-u -> omogućuju pisanje Erlang objektno-orijentiranih programa, kombiniraju objektno-orijentiranu tehnologiju s funkcionalnim programiranjem, omogućuju definiranje klasa i virtualno

nasljeđivanje, omogućuju pisanje metoda sa višestrukim klauzulama i čuvarima, omogućuju redefiniranje ulazafunkcije i dodavanje novog. Neke od ovih funkconalnosti ne postoje u drugim objektno-orijentiranim jezicima (Java, C++,….)

Pokretni agenti na malim uređajima

Kod pokretljivosti u agentskim sustavima razlikujemo:

- Prijenos podataka -> serijalizacija agenata. Postoje standardna serijalizacija agenata u Javi i prilagođena serijalizacija.
- Prijenos klasa -> vrši se preko učitavača klasa (ClassLoader) gdje se klase učitavaju s adrese opisane URL-om. Učitavači klasa se dijele na:
    - System Class Loader -> prvi se poziva i vrši učitavanje klasa iz *classpatha*
    - Extension Class Loader -> poziva se delegiranjem od System Class Loadera i vrši učitavanje klasa iz direktorija koji su navedeni u varijabli okoline *java.ext.dirs*
    - Bootstrap Class Loader -> poziva se delegiranjem od Extension Class Loadera i učitava jezgru i *native* klase (pisane u C-u).

Algoritam učitavanja klasa:

1. Pozvana je metoda loadClass(String name)
2. Provjera je li klasa već učitana
    a. Ako JESTE učitana vrati ju
    b. Ako NIJE učitana delegira se pronalaženje klase, ako delegat nije našao klasu proba ju sam učitati

Mali uređaji se dijele na:

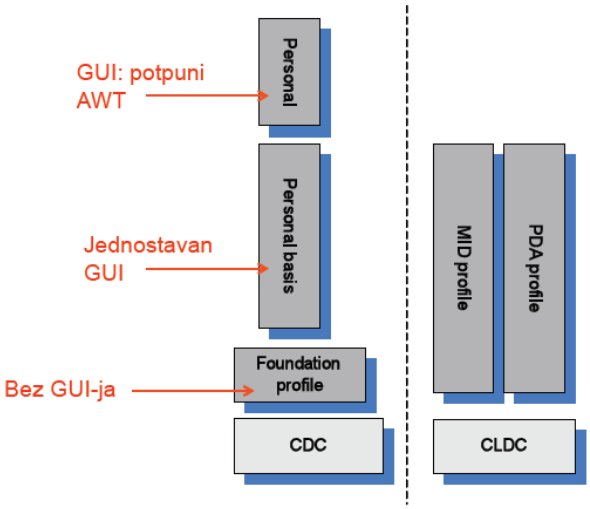
- Pokretne telefone -> imaju specifični operacijski sustav proizvođača i zato nisu programirljivi. Jedini način programiranja za takve uređaje je pomoću Java ME-a
- Pametne telefone -> imaju poznati operacijski sustav i programirljivi su
- Ručna računala -> imaju poznati operacijski sustav i programirljivi su

Rad u odspojenosti mreže -> način rada koji korisniku omogućava ostvarivanej usluge bez stalne veze s komunikacijskom mrežom. Pouzdana veza je potrebna samo pri slanju zahtjeva i primanju rezultata. Modeli za podršku ovakvog rada su:

- Korištenje posredničkih poslužitelja (proxy-a) i posebnih protokola
- Korištenje agentskih tehnologija

Java Micro Edition

- Skup specifikacija koje definiraju reducirane verzije Java SE platforme
- Platforma male moći s ograničenjem memorije i/ili procesorske moći, te brojem ulazno-izlaznih jedinica
- Nije moguće proizvesti jedinstven programski proizvod koji bi se prilagodio svima
- Programska okolina podijeljena na:
    - Konfiguracije -> specifikacija koja definira programsku okolinu za skup uređaja sa zajedničkim skupom svojstava (tipom dostupne mrežne konekcije, veličinom memorije, jačine procesora). Konfiguracija predstavlja minimalnu platformu za ciljane uređaje koja se sastoji od JVM-a i osnovnih biblioteka funkcija koje čine osnovu za razvoj aplikacija namijenjenih određenoj skupini uređaja. Dijele se na:
        - CDC -> veća konfiguracija, 32-bitni procesor, 2MB memorije, namijenejna gruši jačih Java ME uređaja
        - CLDC -> veličinom manja konfiguracija, 16 ili 32-bitni procesor, 160-512kb memorije, namijenjena grupi Java ME uređaja koji često gube vezu s mrežom imaju ograničenu memoriju, jednostavne ulazno-izlazne jedinice i spori procesori
    - Profile



    - Dodatne pakete

Višeagentske platforme za male uređaje –> dijele se na:

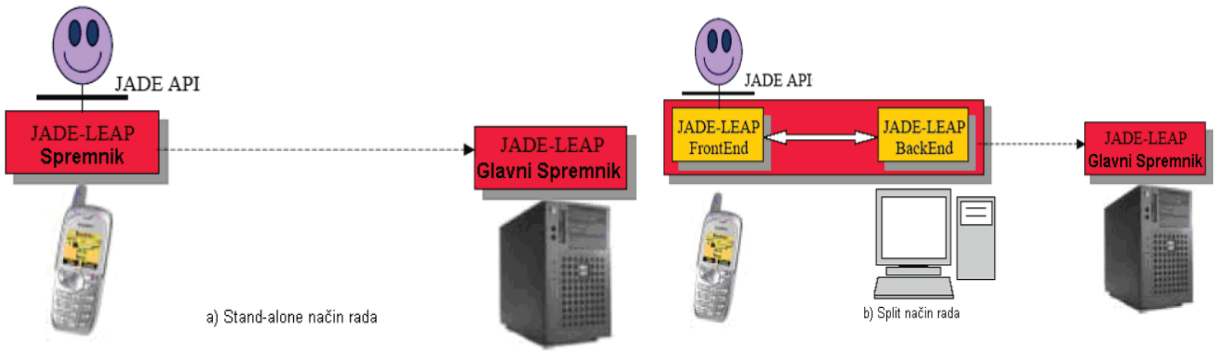
- Portalne platforme -> agenti se izvode na računalu, mali uređaji se koriste samo kao sučelje prema korisniku
- Zamjenske platforme -> izvode se dijelom na malom uređaju, a dijelom na drugim računalima
- Ugrađene platforme -> u cijelosti se izvode na malim računalima zajedno s jednim ili više agenata.

JADE-LEAP (*Lightweight Extensible Agent Platform*) –> omogućuje pokretanje JADE-ovih agenata na ručnim računalima. Izvršiva je na malim uređajima poput PDa i pokretnih telefona, podupire žične i bežične komunikacije, proširiva je u veličini i funkcionalnosti. Za JADE-LEAP je razvijen poseban protokol za odspojeni rad JICP (JADE Inter Container Protocol). Ugrađen je u JADE 4.0. LEAP je primjenjiv na širokom spektru uređaja, od poslužitelja do mobilnih telefona koji podržavaju Javu. Vrste JADE-LEAPa ovisno o tipu Javinog okruženja:

- Java SE -> osobna računala i poslužitelji u nepokretnoj mreži, jedan kontejner (MainContainer) je privilegiran i načazi se na domaćinu na kojem je pokrenut RMI poslužitelj, agenti u različitim kontejnerima koriste RMI protokol za komunikaciju
- PJava -> za ručna računala koja podržavaju Personal Java, ne omogućava izvršavanje JADE-ovih administratorskih alata ni agenata Sniffer i Introspector
- MIDP -> za mobilne telefone koji podržavaju MIDP1.0 i više, za njega vrijede ista ograničenja kao i kod PJave, ne podržava pokretljivost agenata niti njihovo kloniranje
- JADE-Android Addon -> podržava pokretne telefone Android OS, za njega vrijede ista ograničenja kao i kod PJave, ne podržava pokretljivost agenata niti njihovo kloniranje

Izvršavanje JADE-LEAPa na ručnim uređajima dijeli se na:

- Samostalni rad (*Stand-alone*) -> kompletni spremnik se izvršava na malom uređaju
- Podijeljeni rad (*Split*) -> spremnik se dijeli na dva dijela:
    - FrontEnd – na malom uređaju
    - BackEnd – na stolnom računalu ili poslužitelju



Semantički agenti

Semantički web -> web zasnovan na automatizaciji obrade informacija na webu. Sastoji se od:

- Semantičkog znanja -> konceptualizacija (prikaz informacija pomoću koncepata) koja omogućuje logičko zaključivanje o informaciji
- Ontologije -> formalna specifikacija konceptualizacije. Ontologija je reprezentacija znanja kojom su opisane veze između činjenica i koncepata, a jezici za njihovo definiranje se zasnivaju na logičkim formalizmima (npr. opisnoj logici). Primjenom ontologije osigurava se nedvosmisleno tumačenje informacija na webu

RDF (*Resouce Description Framework*) -> osnovni podatkovni model za resurse weba i veze među njima. Resursi se opisuju trojkama (subjekt, predikat, objekt), a sintaksa je definirana XML-om.

RDFS (*RDF Schema*) -> opisni jezik za definiranje rječnika pojmova koji se odnose na svojstva i klase resursa. RDFS omogućuje klasificiranje pojmova u hijerarhije čime se otvara mogućnost stjecanja znanja putem logičkog zaključivanja.

OWL (*Web Ontology Language*) -> opisni jezik za definiranje ontologija zasnovan na opisnoj logici (podskup predikatne logike). Izražajnost opisne logike određena je brojem konstrukata i tipovima dozvoljenih aksioma. Postoje OWL Lite, OWL DL, OWL Full.

Baze znanja zasnovane na opisnoj logici sadrže:

- Ontologije (T-Box) -> opisane jezikom OWL
- Izjave o činejnicama (A-Box) -> mrežno strukturirane i referencirane na ontologije.

Baze znanja omogućuju postavljanje semantičkih upita gdje je znanje zasnovano na uspoređivanju uzoraka. Baza znanja dostupna kao otvoreni kod je *Sesame*.

Jezici za semantičke upite:

- SPARQL (*Simple Protocol and RDF Query language*) -> protokol i jezik za postavljanje semantičkih upita specificiran od W3C-a
- RDQL(*RDF Data Query Language*) -> jezik za postavljanje semantičkih upita kroz programsko sučelje Jena
- SeRQL(*Sesame RDF Query Language*) -> jezik za postavljanje upita u bazu znanja Sesame

Semantički agenti ->agenti koji sadrže baze znanja i programske komponente koje im omogućuju logičko zaključivanje. Oni komuniciraju terminologijom koja je definirana ontologijom. Korištenjem ontologija i jezika za semantičke upite semantički agenti mogu zaključivati o:

- Hijerarhijskim vezama među klasama
- Ekvivalentnosti instanci
- Pripadnosti instance klasi

Procesni model usluge u ontologiji OWL-S -> opisuje uslugu weba kao kompoziciju procesa i razlikuje 3 tipa procesa:

- Elementarni
- Osnovni
- Složeni -> organizirani na osnovu određene strukture kontrole toka koja se može specificirati kontrolnim konstruktima *Sequence*

Ovaj model omogućuje korisnicima postavljanje upita koji uzimaju u obzir način na koji se ulazni parametri transformiraju u izlazne parametre usluge weba.

Semantičko uspoređivanje usluga weba -> davatelji usluga oglašavaju uslugu opisanu pomoću ontologije OWL-S, a korisnik šalje upit za uslugom. Semantički agent uspoređuje korisnički upit s oglasima davatelja usluge i odabire odgovarajuću uslugu primjenom algoritma za uspoređivanje. Semantičkim uspoređivanjem utvrđuje se stupanj podudarnosti usluga weba na osnovu semantičke ekvivalentnosti upita i oglasa.

Programske komponente za semantičko uspoređivanje:

- Mehanizam za zaključivanje -> omogućuje zaključivanje temeljeno na opisnoj logici. Transformira datoteke opisane u OWL-u u oblik pogodan za komparator.
- Mehanizam zasnovan na pravilima -> omogućuje semantičkom agentu obradu trojki u RDF-u prema pravilima koja definiraju aksiome jezika OWL i evaluacijom tih pravila na osnovu činjenica iniciraju se akcije specificirane pravilim. Ovaj mehanizam sadrži radnu memoriju, bazu pravila i mehanizam za evaluaciju pravila, te omogućuje deklarativno definiranje činjenica i pravila. Jess je mahanizam zasnovan na pravilima za programsku podršku Javi
- Komparator -> semantički uspoređuje datoteke napisane u OWL-u primjenom algoritma za uspoređivanje i određuje stupanj podudarnosti usluge weba.