MOBILNOST PRI AGENTNO USMERJENEM RAZVOJU PROGRAMSKE OPREME

Bogdan Kronovšek, Dejan Lavbič, Marjan Krisper

Bogdan.Kronovsek@guest.arnes.si, {Dejan.Lavbic, Marjan.Krisper}@fri.uni-lj.si

POVZETEK

Paradigma mobilnosti programske kode in inteligentnih agentov sta se združili pri mobilnih agentih. Mobilni agenti so aktivne entitete, ki se selijo iz enega izvajalnega okolja v drugega, pri tem pa srečujejo druge agente, uporabljajo storitve in vire, ki so tam na voljo, medtem ko za poslovnega uporabnika izvršujejo določeno nalogo. Zaenkrat so še vedno bolj v domeni raziskovalnih krogov, kljub temu pa so obeti uporabe tega pristopa pri razvoju programske opreme veliki. Z razvojem kvalitetnih in zanesljivih platform, ki omogočajo implementacijo mobilnih agentov, se je zanimanje zanje zelo povečalo. Njihova uporabnost in prednosti pa se izkažejo predvsem pri večjih omrežnih sistemih in internetu. Študija primera razvoja mobilnega agenta, ki ga prispevek predstavlja, je s področja poslovnega obveščanja, združuje pa koncept mobilnih agentov in koncept mobilnega uporabnika.

ABSTRACT

Mobility of program code and intelligent agents has merged in paradigm of mobile agents. Mobile agents are active entities traversing between several containers while cooperating with other agents, using their services and resources. They work on behalf of business user and execute demanded tasks. While still mainly used in research environments they introduce feasible approaches with high expectations in software development. Interest for mobile agents has been growing rapidly as more and more execution platforms became available. Their applicability and advantages can be used especially in the field of network systems and internet. The article presents a prototype of mobile agents from the domain of business intelligence. The study conducted combines the mobile agent's concept and mobile user.

1 UVOD

V zadnjih nekaj letih je agentno usmerjen razvoj programske opreme deležen posebne pozornosti, saj ga nekateri priznavajo za naslednjo stopnjo v evoluciji razvoja programske opreme. Večagentni sistemi so sestavljeni iz več med seboj sodelujočih entitet, imenovanih agenti. Njihove glavne lastnosti in odlike so: avtonomnost (delovanje brez neposredne pomoči uporabnika), reaktivnost (odzivanje na spremembe okolja), proaktivnost (prevzem pobude v primernem trenutku) ter družbenost (sposobnost interakcije z drugimi agenti ali uporabniki). Poleg teh imajo nekateri tudi sposobnost mobilnosti oziroma selitve po omrežju. Paradigma mobilnosti je postala z razvojem migracije procesov, oddaljenega preračunavanja, porazdeljenih objektov in porazdeljenega procesiranja zelo zanimiva in uporabna. Mobilni agenti so programi, ki s seboj selijo izvajalno kodo, izvajalno stanje ter vse ostale podatke potrebne za njihovo avtonomno delovanje [18, 21]. Njihove prednosti so predvsem robustnost, zmanjševanje porabe lokalnih virov in zniževanje omrežnega prometa. Glede na lastnosti agentov in ob upoštevanju dejstva, da zahtevamo ljudje v razvijajoči se informacijski

družbi hiter, zanesljiv, poceni ter prijazen in fleksibilen dostop do različnih informacij, so postali mobilni agenti in večagentni sistemi na sploh zelo primerni za nudenje različnih storitev in osebne asistente uporabnikom. Seveda pa je ključno za približanje in širše sprejetje agentov kot človekovih osebnih asistentov njihova implementacija na mobilnih napravah, ki jih ljudje najpogosteje uporabljamo. Za razširitev agentov na področje mobilni naprav je potrebno splošno razširjeno ogrodje, ki omogoča njihovo izvajanje na performančno šibkejših napravah. Ta problem so razvijalci JADE (Java Agent Development Environment) platforme, ki je namenjena razvoju in implementaciji večagentnih sistemov rešili z dodatkom LEAP (Lightweight Extensible Agent Platform) in omogoča implementacijo agentov na različnih mobilnih napravah [6, 16].

V današnjem času se srečujemo z ogromno količino informacij v različnih omrežjih, najpomembneje pa je dobiti prave in kvalitetne informacije v pravem trenutku in na uporabniku čimbolj priročen način. Mobilni agent za poslovno obveščanje, ki ga v prispevku predstavljamo, je namenjen podpori odločanja dinamičnega poslovnega uporabnika. Cilj tega mobilnega agenta je, da uporabniku z izkoriščanjem prednosti, ki jih nudijo mobilni agenti, na čimbolj ekonomičen in za celoten sistem nemoteč način, pridobi podatke o novih izdelkih, ki so se v bližnji prihodnosti pojavili na svetovnem spletu in pri konkurenci. Primer mobilnega agenta združuje koncept inteligentnih agentov, mobilnih agentov in platforme JADE-LEAP, ki omogoča izvajanje agentov tudi na lahkih oz. mobilnih napravah.

Prispevek v 2. razdelku na kratko predstavi področje agentov in večagentnih sistemov, v 3. razdelku pa je podrobno predstavljena posebna vrsta agentov, ki so zmožni seljenja na različne lokacije. 4. razdelek opisuje primer mobilnega agenta za poslovno obveščanje. Na koncu sledi še zaključek in nadaljnje smernice za razvoj na področju mobilnih agentov.

2 AGENTI IN VEČAGENTNI SISTEMI

2.1 Kaj je agent?

Na področju informacijskih tehnologij ne obstaja standardizirana opredelitev pojma agent, saj ima zaradi široke uporabe v številnih krogih zelo različen pomen. Jasno je, da je agent nekaj več kot samo program, toda kje ležijo meje še ni povsem jasno. Ker se agenti uporabljajo na številnih področjih in v raznovrstne namene, se je pojavila cela vrsta opredelitev agenta, med katerimi nekatere definirajo pojem agenta preveč ohlapno, druge pa, predvsem s predpisovanjem tehnik umetne inteligence, preveč podrobno. Kljub zelo različnim pogledom na agente, njihove lastnosti, področja delovanja in inteligenco, je opredelitev agenta, ki jo je podal M. Wooldridge [25] še najbolj primerna in uporabljana: "Agent je računalniški sistem nameščen v določeno okolje, v katerem avtonomno deluje, da bi dosegel načrtovane cilje."

S podrobnim pregledom literature na tem področju lahko ugotovimo, da se pojavljata dve uporabi izraza agent – šibka notacija in močna notacija. Šibka notacija agenta predstavlja najmanjšo množico lastnosti s katerimi se strinja velika večina raziskovalcev, medtem ko je močnejša bolj polemična in še vedno stvar aktivnih raziskav.

Šibka notacija označuje agenta kot programski sistem z naslednjimi lastnostmi:

- **Avtonomnost:** agent je zmožen delovanja brez neposrednega posredovanja uporabnika ali drugih agentov. To pomeni, da je samostojen, ima nadzor nad svojimi akcijami in notranjim stanjem, kar pomeni, da se sam odloči ali se bo odzval in izpolnil zahteve drugih agentov ali ne [24].

- **Družbenost:** sposobnost interakcije agenta z drugimi agenti ali uporabniki za dosego svojih ciljev oziroma pomoči pri doseganju ciljev drugih agentov. Komuniciranje, sodelovanje in pogajanje agentov ponavadi poteka preko nekega skupnega komunikacijskega jezika [2, 8].
- **Reaktivnost:** sposobnost agenta, da zaznava svoje okolje (uporabnik preko grafičnega vmesnika, množica drugih agentov, internet ali pa kombinacija vsega tega) in se na njegove spremembe primerno odzove z namenom, da doseže načrtovane cilje.
- **Proaktivnost:** ciljno usmerjeno delovanje agenta, pri katerem je tudi sam sposoben prevzeti pobudo, če je to potrebno.

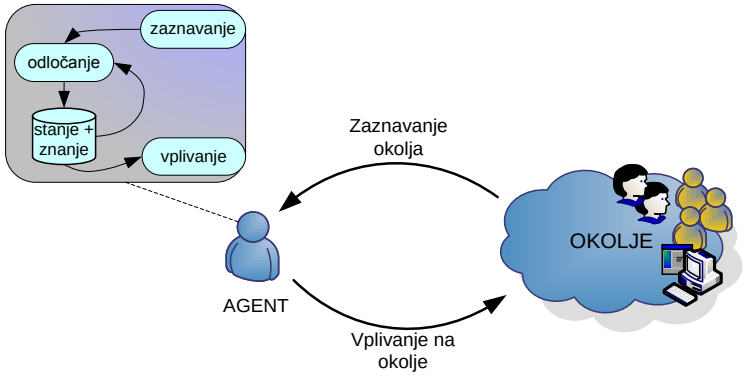
Močnejša notacija je razširitev šibkejše, ki agentom pripisuje bolj človeške lastnosti, kot so znanje, prepričanje, namen, dolžnost [22]. Nekateri raziskovalci s področja umetne inteligence [3] so šli še dlje in razmišljajo o čustvenih agentih. Potrebno je poudariti, da to ni le brezciljen antropomorfizem ter, da obstaja veliko argumentov za razvoj agentov s človeku podobnimi miselnimi stanji. Še posebno na področju vmesnikov med človeškim uporabnikom in računalnikom je veliko zanimanje za razvoj agentov s človeškimi lastnostmi in morebitno vizualizacijo oziroma uporabo animiranih obrazov.

Poleg naštetih lastnosti se pri šibki notaciji najpogosteje omenjajo še učenje/prilagajanje, kognitivnost, fleksibilnost in mobilnost. Pomembnost in vrsta posameznih lastnosti je seveda zelo odvisna od perspektive, področja in namena uporabe agenta, kljub temu pa za večino strokovnjakov predstavlja ta seznam tisto, kar naj bi vplivalo na opredelitev agenta. V okviru že naštetih lastnosti, kvalitet in taksonomij, se v širšem pogledu na predstavitev inteligence v agentih srečamo z dvema posebnima vrstama inteligentnih agentov – reaktivni in kognitivni agenti.

Reaktivni agenti so preprosti agenti, ki zaznavajo svoje okolje in v njem delujejo, sami po sebi pa niso »inteligentni«. Njihove naloge in sposobnosti so omejene zgolj na zaznavanje okolja in odzivanje na spremembe, saj nimajo simbolične predstavitve okolja, znanj, prepričanj in drugih mehanizmov umetne inteligence. Njihova inteligenca se kaže predvsem v združevanju v večje skupine ali večagentne sisteme (razdelek 2.2), kjer je inteligenca porazdeljena med posameznimi reaktivnim agenti, ki zaznavajo in delujejo na okolje ter ostale agente, navzven pa delujejo kot nekakšen globalen inteligenten sistem.

Kognitivni agenti so za razliko od reaktivnih »inteligentnejši«, saj so sposobni planiranja svojega načina delovanja, zapomniti si že izvedene akcije in se pogajati z drugimi agenti. Ti agenti vsebujejo tudi dodaten simbolični nivo, v katerem so predstavljeni miselni pojmi, kot so znanje, namen, prepričanje in zadolžitve, to je način, kjer prenesemo človeško inteligenco in perspektivo sveta v večagentne sisteme [14, 22]. Zaradi narave teh agentov oziroma njihovega inteligentnega delovanja se v okviru večagentnih sistemov navadno pojavljajo v manjšem številu.

Agent vedno deluje v svojem okolje, ki ga zaznava in glede na svoje trenutno stanje ter zaznave, izvede akcijo s katero vpliva na okolje z namenom, da doseže zadane cilje (glej Slika 1). Pri svojem cikličnem delovanju *zaznavanje okolja-odločanje-vplivanje na okolje*, je ključnega pomena izbrati pravo akcijo, izmed vseh, ki so mu na voljo, da bo z njo vplival na okolje in tako čim lažje izvedel svojo trenutno nalogo in na koncu tudi dosegel svoj cilj.



Slika 1: Agent in njegovo okolje

Okolja so seveda lahko zelo različna, zato je od njihovih lastnosti v določeni meri odvisno tudi delovaje agenta. Russel in Norvig [20] sta predlagala naslednjo opredelitev oziroma lastnosti okolja:

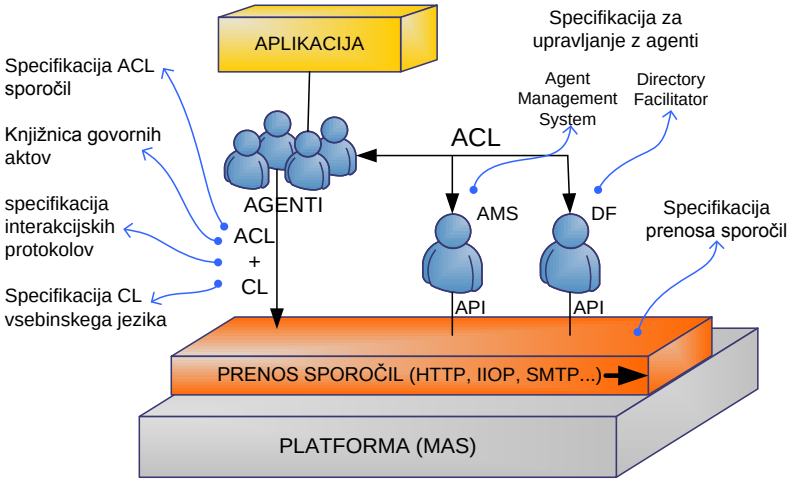
- **Dostopnost – nedostopnost**: v primeru, da je agentovo okolje dostopno, lahko agent pridobi popolne in natančne informacije o celotnem okolju. V nasprotnem primeru je okolje nedostopno (večina resničnih okolij, na primer internet).
- **Determinističo – nedeterminističo**: determinističo okolje je popolnoma predvidljivo, saj je njegovo naslednje stanje v celoti določeno s trenutnim stanjem in akcijo, ki jo agent izvede.
- **Dinamično – statično**: dinamično okolje je tisto, ki se lahko med agentovim načrtovanjem akcije spreminja, zato ga mora agent nenehno preverjati, npr. internet. Pri statičnem okolju pa je vsaka sprememba odvisna le od agentovih akcij.
- **Diskretnost – zveznost**: diskretno okolje ima točno določeno in končno število možnih akcij in stanj, npr. šah. V nasprotnem primeru zvezno okolje nima teh omejitev.

2.2 Večagentni sistemi

Podobno kot človek, ki je družbeno bitje, tudi agent ne more, oziroma je težko pričakovati, da bo obstajal in deloval sam zase, brez interakcije z drugimi agenti in uporabniki. Že pri definiciji agenta igra družbenost (glej razdelek 2.1) pomembno vlogo, ki pa je z razširjanjem tehnologije večagentnih sistemov (MAS – Multi-Agent Systems) postala še pomembnejša. Vendar pa le zbrati več agentov v skupino še ne pomeni nujno večagentnega sistema. Ko govorimo o večagentnih sistemih moramo upoštevati še problematiko organizacije agentov in dinamiko sistema. Večagentni sistem je vsak računalniški sistem sestavljen iz več agentov, ki ima naslednje značilnosti: vsak agent ima nepopolne informacije ali sposobnosti za reševanje problema in ima tako omejen pogled na sistem; sistem nima globalnega nadzora, podatki so porazdeljeni, procesiranje pa je asinhrono [24]. Ker so agenti avtonomne entitete, ki imajo vsak svoje cilje, se velikokrat seveda zgodi, da so ti cilji konfliktni. Za uspešno delovanje večagentnega sistema je tako potrebno, da so agenti sposobni medsebojne interakcije, kar pomeni, da so sposobni sodelovati drug z drugim ter so sposobni koordinacije in pogajanj v primeru konfliktnih interesov. Zato je MAS tudi sistem, v katerem agenti med seboj sodelujejo, največkrat z izmenjavo sporočil po izbrani infrastrukturi računalniškega omrežja

[24, 25]. Seveda pa mora za dobro delovanje MAS sistem skrbeti nekdo, ki predstavlja avtoriteto in upravlja s celotnim sistemom, kot sta AMS in DF agenta (glej razdelek 4.1)

Zaradi vse večjega povezovanja različnih večagentnih sistemov, se je pojavila potreba po združljivost ter uvedbi norm in standardov na tem področju. Prav tako se je pokazala potreba po centralni organizaciji, ki bi določala in skrbela za standarde na tem področju.



Slika 2: FIPA kompatibilna platforma večagentnega sistema [9]

FIPA (The Foundation for Intelligent Physical Agents) je organizacija, ki skrbi za standarde na področju agentnih in večagentnih tehnologij ter se ukvarja z združljivostjo s standardi drugih tehnologij [9]. Ustanovljena je bila leta 1996, od junija 2005 naprej pa je članica IEEE Computer Society in je edino resnejše združenje, ki se ukvarja s standardi na tem področju. Definirali so že več poglavitnih specifikacij agentov, med njimi tudi standard za jezik komunikacije agentov FIPA ACL (FIPA Agent Communication Language). Njena osrednja področja raziskovanja so upravljanje z življenjskim ciklom agenta, prenos sporočil, struktura sporočil, interakcijski protokoli, ontologije in varnost [26]. Standardi in specifikacije določajo predvsem pravila, logiko in smernice pri razvoju platform ter agentov in so z vidika programskih jezikov in platform popolnoma neodvisni. Razvijalec mora za razvoj in implementacijo agentov tako poznati predvsem logiko MAS sistemov, strukturo standardnih sporočil, ontologij ter protokolov in seveda katerega od programskih jezikov, ki jih podpira razvojno orodje oziroma platforma, ki ima implementirane FIPA standarde v jedru in svojih knjižnicah (glej Slika 2).

Komunikacija predstavlja bistveno interakcijo med agenti v MAS. Z intenzivno medsebojno izmenjavo informacij lahko agenti bolje spoznavajo svet okoli sebe in agente s katerimi lahko sodelujejo pri doseganju lastnih ali pa skupnih ciljev. Na tem mestu je potrebno omeniti še pomembno teorijo na področju komunikacije med agenti – to je teorija govornih aktov, ki pravi, da lahko izjavo primerjamo z izvajanjem akcij; če pomeni delovati spreminjanje stanja določene stvari v svetu, je govoriti podobno spreminjanju miselnega stanja sogovornika [12]. Ta teorija je imela velik vpliv na številne jezike komunikacije razvite za področje MAS, saj predstavlja tri glavne zahteve:

- Opredelitev taksonomije različnih govornih aktov.
- Definirati formalno izrazoslovje.
- Upravljanje z govornimi akti ter preučiti povezavo med posameznimi govornimi akti in njihovo semantiko.

Zaradi potrebe po standardizaciji komunikacije med agenti, so bili razviti komunikacijski jeziki ACL. Njihov namen je poenostaviti komunikacijo, učinkovito opisati komunikacijske akte, tako iz semantičnega kot tudi iz sintaktičnega vidika, ter zagotoviti enostavno izmenjavo znanja in drugih miselnih pojmov. Da se lahko agenti med seboj uspešno sporazumevajo, tako v smislu semantike kot tudi pragmatike, vsebuje ACL naslednje vidike (primer v Tabela 1):

- **Sintaksa** določa kako bodo simboli jezika strukturirani.
- **Pragmatika** določa uporabo in interpretacijo simbolov.
- **Ontologija** določa skupen slovar besed, s katerim bo predstavljena vsebina jezika.

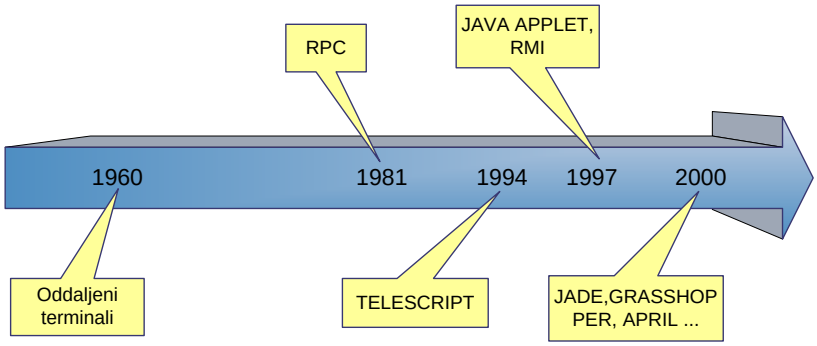
```
(inform
  :sender     agent1
  :reciver    agent2
  :content    (cena  izdelek2  230)
  :language                    sl
  :ontology              hpl-auction
)
```

Tabela 1: Primer FIPA-ACL sporočila

Pred ustanovitvijo organizacije FIPA in standardizacije jezika FIPA-ACL, je bil edini poskus standardizacije jezik KQML, ki se je kasneje razvil v več med seboj nezdružljivih različic. Potrebno pa je poudariti, da niti FIPA-ACL, niti KQML ne določata implementacije, ki bi vnaprej predpisovala določen programski jezik ali računalniško okolje. Edina zahteva je, da mora biti implementacija skladna s specifikacijami jezika.

3 MOBILNOST IN MOBILNI AGENTI

Eden od načinov delitve agentov je tudi delitev agentov po njihovih lastnostih oziroma po njihovi dejavnosti, npr. filtrirni agenti, posredovalni agenti, nakupovalni agenti, osebni pomočniki itd. Znotraj te delitve obstaja velika skupina agentov, ki jih združuje lastnost seljenja in navigacije. Imenujejo se **mobilni agenti** in so agenti, ki so sposobni prenesti sebe, svoj program in stanje čez omrežje in nadaljevati z izvajanjem na oddaljeni lokaciji [10, 11, 13, 23].



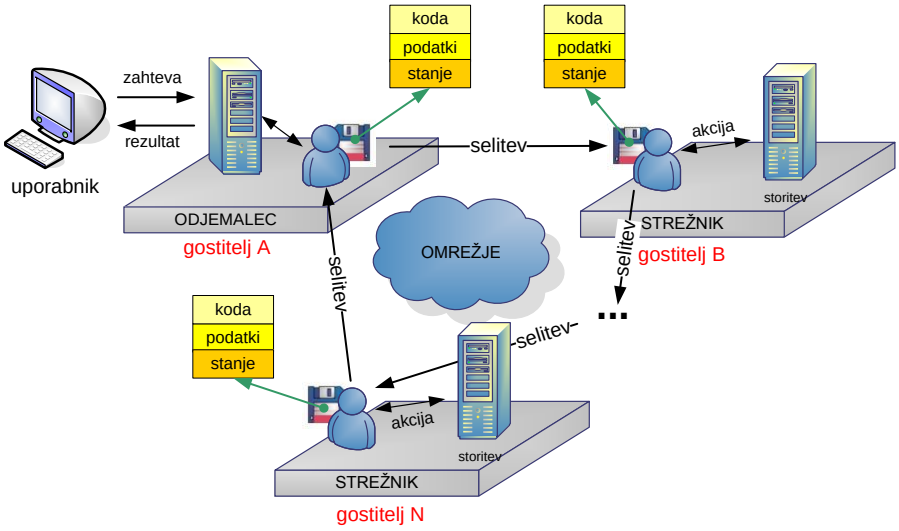
Slika 3: Časovni pregled koncepta mobilnosti

Koncept mobilne kode oziroma programa sploh ni nov, saj njegovi zametki segajo v leto 1960 (glej Slika 3), ko so za vnos programov v centralni računalnik uporabljali oddaljene terminale [10]. Največji premik na tem področju pa je leta 1994 naredilo podjetje General Magic, ko je predstavilo idejo mobilnega agenta in razvilo skriptni jezik Telescript, ki bi naj uresničil vizijo mobilnega agenta [24]. Z razvojem interneta in programskega jezika Java pa je

zanimanje za pionirski Telescript počasi zbledelo. Internetni brskalniki, s podporo Java virtualnega stroja, so sedaj omogočili prenos in izvajanje majhnih programov na daljavo, imenovanih aplet (ang. applet). Z omenjenimi začetki je tako tehnologija mobilnih agentov napredovala do te mere, da omogoča popolno implementacijo koncepta mobilnosti (prenos programa in stanja na drugo lokacijo) in koncepta agenta (avtonomnost, učenje, inteligenca itd.). Trenutno obstaja kar nekaj platform oziroma okolij za implementacijo MAS in mobilnih agentov po standardih FIPA [7]:

- **FIPA-OS** (mobilnost je še prototip)
- **GRASSHOPPER** (standardi: MASSIF, FIPA; šibka mobilnost in simulacija močne mobilnosti)
- **TRYLLIAN ADK** (standardi: FIPA, SOAP; močna mobilnost)
- **JADE / LEAP** (standardi: FIPA; močna mobilnost)

Običajna MAS aplikacija vključuje več agentov, ki komunicirajo in sodelujejo med seboj, kjer vsak igra svojo vlogo. Aplikacije mobilnih agentov pa dodatno vsebujejo še število izvajalnih okolij oziroma oddaljenih lokacij, ki omogočajo agentom njihovo izvajanje in nudijo določene storitve ter vire (glej Slika 4). Ta izvajalna okolja predstavljajo logične lokacije, ki se lahko nahajajo na istem računalniku ali pa so dejansko razpršene po oddaljenih računalnikih in so med seboj povezani. Dejstvo, kjer se neka lokacija dejansko nahaja, je za delovanje agenta popolnoma nepomembno.



Slika 4: Primer sistema mobilnih agentov

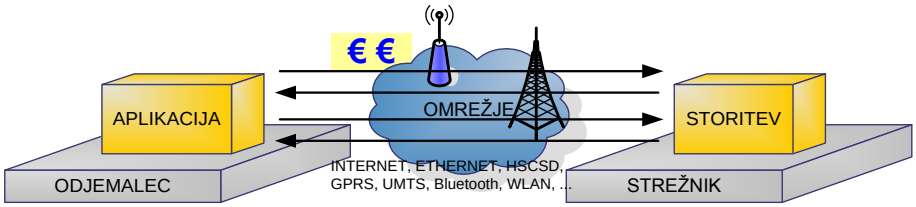
V tem kontekstu so mobilni agenti aktivne entitete, ki se selijo iz enega sistema na drugega, pri tem pa srečujejo druge agente, uporabljajo storitve in vire, ki so tam na voljo, medtem ko za poslovnega uporabnika oz. naročnika storitve izvršujejo določeno nalogo. Na drugi strani pa viri predstavljajo neavtonomne entitete, kot so datoteke, objekti, podatkovne baze, programi in zunanje aplikacije, ki jih lahko uporabljajo in souporabljajo med seboj različni agenti.

Sistem mobilnih agentov je programska komponenta, ki deluje kot neke vrste pristanišče in agentom nudi lokalne vire. Zadolžen je za izvajanje agentovega programa ali kode v zaščitenem okolju [19]. Poleg tega mora agentom omogočiti tudi druge lastnosti, kot so robustnost, varnost in druge operacije, ki omogočajo komunikacijo, seljenje in dostop do lokalnih virov. V tem kontekstu predstavljajo izvajalni prostori logične lokacije, kjer se agenti

izvajajo, srečujejo in komunicirajo z drugimi agenti ter uporabljajo lokalne vire. V splošnem so agenti, sistemi, lokacije in viri identificirani z unikatnimi imeni ali elektronskimi naslovi. Eden ali več sistemov mobilnih agentov lahko obstaja v vozlišču, ki predstavlja strojno infrastrukturo na kateri se izvajajo tako sistem mobilnih agentov, kot tudi agenti sami. Tipični predstavnik vozlišča je osebni računalnik, nekateri sistemi, kot je npr. JADE-LEAP pa omogočajo implementacijo vozlišč celo na prenosnih napravah, kot so dlančniki in mobilni telefoni.

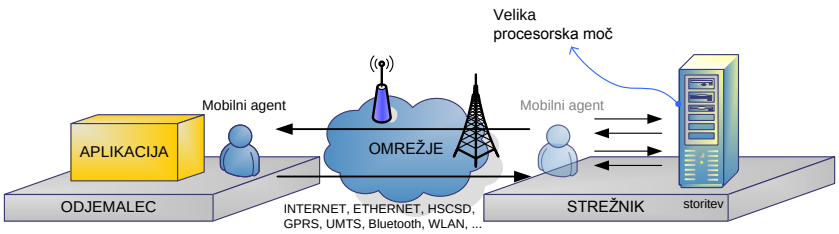
3.1 Prednosti mobilnih agentov

Paradigma mobilnih agentov se je razvila iz dveh predhodnikov, paradigme odjemalec-strežnik (ang. Client-server, glej Slika 5) in paradigme oddaljenega preračunavanja (ang. Remote Evaluation – REV) [11]. Porazdeljene aplikacije so tradicionalno temeljile na paradigmi odjemalec-strežnik, pri kateri odjemalec in strežnik vzpostavita komunikacijo s pomočjo sporočil ali z oddaljenim klicem procedure (ang. Remote Procedure Call – RPC). Ta komunikacijska oblika je sinhrona, kar pomeni, da se odjemalec, ko pošlje zahtevo strežniku, postavi na čakanje dokler ne dobi odgovora. V primeru izpada omrežja ali v primeru da strežnik ne odgovori, bo odjemalec čakal neskončno dolgo.



Slika 5: Klasična aplikacija odjemalec-strežnik

V začetku devetdesetih je bila predlagana alternativa REV-u. Pri tej paradigmi odjemalec namesto klica oddaljene procedure pošlje kodo svoje procedure in prosi strežnik, da jo izvede ter vrne rezultat. Pri RPC se podatki prenašajo v obe smeri med strežnikom in odjemalcem, pri REV pa se koda pošlje od odjemalca k strežniku, nazaj pa se vrnejo le podatki. V tem kontekstu je mobilni agent program (ovita koda, podatki in izvajalna vsebina ali stanje), ki ga odjemalec pošlje strežniku. Za razliko od REV in RPC, agentu ni potrebno vrniti rezultatov odjemalcu, lahko se celo preseli na nek drug strežnik, pošlje informacijo odjemalcu ali pa se celo preseli k njemu, če je to primerno oz. potrebno. Na mobilnega agenta lahko gledamo kot zadnjo stopnjo evolucije abstrakcije mobilnosti, kot je mobilna koda, objekti in procesi [16, 23]. Preneseno stanje kode, kot je aplet, vsebuje samo kodo, stanje mobilnega objekta vsebuje kodo in podatke, stanje mobilnega procesa pa dodatno vsebuje še stanje izvajalne niti. Stanje mobilnega agenta zajema njegovo kodo, podatke, izvajalno nit in pooblastilo svojega lastnika. Glede na vzorce selitve, se mobilni agenti razlikujejo od apletov (prenesenih iz strežnika na odjemalec) in servletov (prenesenih iz odjemalca na strežnik) v tem, da lahko opravijo več skokov oziroma selitev ter se odcepijo od odjemalca in delujejo avtonomno. Mobilni agenti avtonomno obiskujejo različne lokacije brez interakcij s svojo domačo oziroma izvirno lokacijo [15, 27].



Slika 6: Izvajanje računsko zahtevnih nalog na oddaljenem in procesorsko močnejšem računalniku

V primerjavi z drugimi pristopi porazdeljenih sistemov ima paradigma mobilnih agentov kar nekaj prednosti [5, 10, 11, 13, 23] (glej Slika 6):

- **Zmanjšanje stroškov komunikacije.** S povečanjem prometa pri podatkovnih komunikacijah je samo še vprašanje časa kdaj bo potrebno plačevati promet glede na preneseno količino. Z uporabo mobilnih agentov se zmanjša količina prenesenih podatkov in s tem tudi stroški komunikacije.
- **Zmanjšana oziroma uravnotežena poraba lokalnih sredstev.** Zmožnost procesiranja informacij na lokalnih računalnikih je omejena, zato uporaba mobilnih agentov pozitivno vpliva na porabo lokalni virov in jih v večjih sistemih celo uravnoteži. Tukaj gre za porazdeljeno procesiranje ali pa procesiranje na lokaciji z veliko procesorsko močjo.
- **Razbremenitev računalniških omrežij.** Ker mobilni agenti komunicirajo z oddaljenimi strežniki lokalno, se zaradi tega zmanjša količina prenesenih podatkov po omrežju.
- **Asinhrono izvajanje.** Mobilni agenti lahko izvajajo zadane naloge tudi takrat, ko uporabnik ni priključen na omrežje.
- **Dinamično prilagajanje na spremembe.** Mobilni agenti lahko komunicirajo s svojim okoljem in pridobivajo informacije na podlagi katerih potem sprejemajo odločitve.
- **Robustnost in odpornost na izpade.** V primeru izpada komunikacije lahko agent počaka na oddaljeni lokaciji do ponovne vzpostavitve. Zaradi mobilnosti imajo to lastnost, da lahko shranijo svoje stanje, zato v primeru sesutja sistema shranijo svoje stanje na lokalni disk in po vnovični vzpostavitvi nadaljujejo z izvajanjem.

Mobilni agenti se izkažejo kot zelo inovativen pristop pri različnih problemskih domenah, kot so elektronske trgovine, porazdeljeno pridobivanje informacij, upravljanje z delovnim tokom in sodelovanje, osebni pomočniki, oddaljeni nadzor in nastavitve raznih naprav, sodobne telekomunikacijske storitve itd. Te aplikacije jasno kažejo na prednosti uporabe mobilnih agentov, saj zmanjšujejo obremenjenost omrežja, premagujejo latentnost omrežja, se izvajajo avtonomno in asinhrono, so neobčutljivi na napake in izpade ter zaznavajo svoje izvajalsko okolje in se na njegove spremembe dinamično odzivajo [1, 12].

3.2 Mobilnost programske kode in izvajalnega stanja

Mobilnost agenta vključuje prenos kode, podatkov in po možnosti tudi izvajalnega stanja na lokacijo, kjer se nahajajo drugi agenti ali viri, ki jih agent želi uporabljati.

Z vidika izvajalnega stanja obstajata dva pristopa:

- **Močna mobilnost** (ali s popolnim stanjem). V tem primeru ima sistem mobilnih agentov zmožnost, da dovoli prenos tako kode kot tudi agentovega izvajalnega stanja na drugo lokacijo. Agent lahko vsebuje selitveni ukaz kjerkoli v kodi, brez posebne zahteve za obnovo stanja. Pri izvedbi ukaza za seljenje se agentovo trenutno stanje in koda preneseta na ciljno lokacijo. Po prihodu pa agent nadaljuje z izvajanjem ukaza, ki sledi selitvenemu.

- **Šibka mobilnost** (ali z nepopolnim stanjem). V tem primeru ima sistem mobilnih agentov zmožnost, da dovoli le prenos kode na drugo lokacijo. Koda lahko sicer vsebuje določene inicializacijske podatke, vendar pri tej selitvi ni vključeno seljenje izvajalnega stanja, zato se po agentovem prihodu njegovo izvajanje ponovno inicializira.

Na drugi strani omogočajo mehanizmi prenosa agentove kode po sistemu mobilnih agentov in jo dinamično povežejo z določenim agentom ali pa se uporabi kot segment kode za novega agenta.

Ti mehanizmi se delijo glede na **smer prenosa**, saj lahko agent kodo prejme ali pa pošlje, **način prenosa**, ki je določen s tem kako se kodni zaključek prenese, ali se prenese ali ne ali se prenese na enkrat ali pa postopoma po potrebi. Pomembni lastnosti, ki določata mehanizme prenosa sta tudi **vključenost sinhronizacije**, ali se zahteve izvajajo sinhrono ali asinhrono slednje pomeni, da se agentove zahteve začasno ne ustavijo, čeprav prejšnja koda še ni izvedena do konca ter **čas v katerem se koda dejansko izvede**, ki je lahko takojšnje ali pa odloženo.

Na področju mobilnih agentov je **varnost** osrednja tema pogovorov in eden glavnih vzrokov zakaj se mobilni agenti ne uporabljajo bolj množično. Glavna težava je v tem, da če želimo, da agent po prihodu na novo lokacijo izvede nekaj koristnega, mora imeti dostop do lokalnih virov te lokacije. Toda nuditi uporabo teh virov pomeni potencialno nevarnost. Nevarnosti se pojavljajo v treh glavnih smereh, **s strani agenta do gostitelja**, **s strani agenta do drugega agenta** in **s strani gostitelja do agenta**.

Možne zlorabe gostitelja s strani agenta so predvsem izbris sistemskih datotek, kraja in nezaželen dostop do informacij, zloraba identitet drugih agentov za dostop do gostitelja ter vohunjenje in spremljanje gostiteljevega delovanja. Gostitelj ima za zaščito na razpolago tri rešitve, digitalni podpis s katerim agent izkaže istovetnost, izvajanje sumljive kode v bolj zaščitenem in omejenem okolju (črna skrinjica) ter sprotno ocenjevanje in napovedovanje morebitne zlorabe.

Agent pa seveda lahko pomeni nevarnost tudi drugimi agentom, saj lahko pridobi občutljive informacije, ki lahko škodijo ostalim agentom, ki se izvajajo v istem vsebniku (virtualnem stroju). Rešitev tega problema je predvsem v zahtevi po identifikaciji agenta za izvedbo določene operacije.

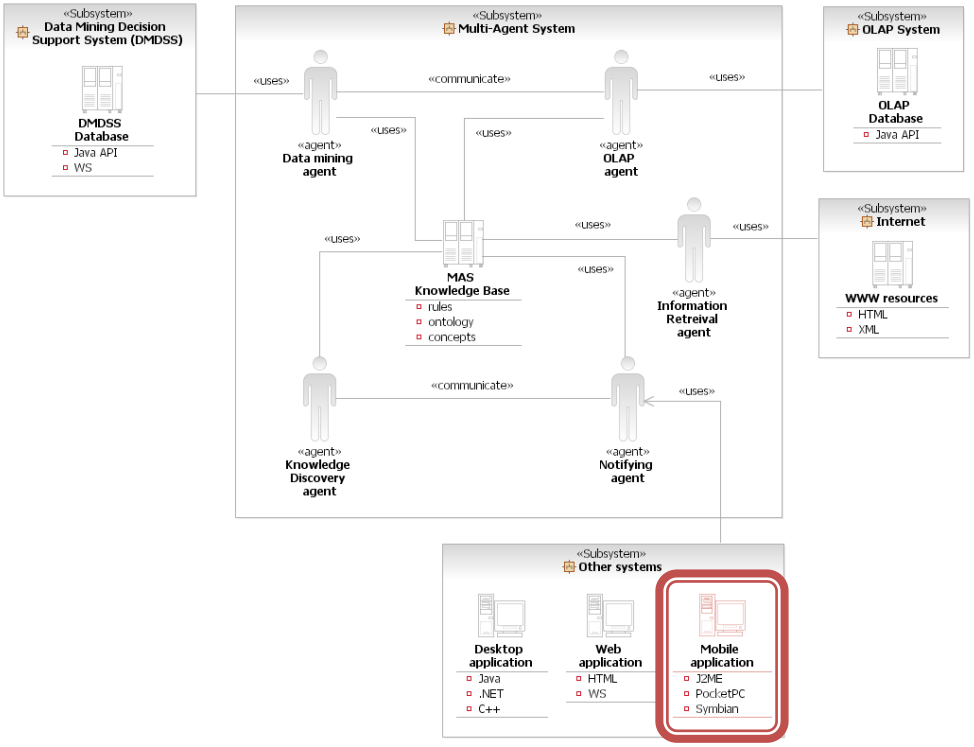
Gleda na to, da gostitelj nudi izvajalno okolje agentu ima nad njim popolni nadzor, to pa pomeni, da lahko v primeru zlorabe spremeni agentovo kodo, prepreči in blokira komunikacijo in seljenje agenta, lahko se zamaskira in predstavi kot nek drug zaupanja vreden gostitelj, dostopa do zaupnih podatkov, ki so namenjeni drugim agentom.

Omejitev gostiteljeve moči nad agenti pomeni predvsem kodiranje funkcij, ki jih gostitelj lahko izvede brez dostopa do njene kode, agentovo sprotno ocenjevanje svojega stanja in napovedovanje pričakovane spremembe stanja, snemanje agentove poti za kasnejši pregled.

Kombinacija prej naštetih nevarnosti pa pomeni grožnjo, ki jo lahko predstavlja gostitelj drugemu gostitelju.

4 PRIMER MOBILNEGA AGENTA ZA POSLOVNO OBVEŠČANJE

Mobilni agent za poslovno obveščanje je v našem primeru namenjen podpori odločanju, kjer je pomembno spremljanje ponudbe novih izdelkov in novosti na svetovnem trgu ter pri domači konkurenci. Je del večagentnega sistema, za svoje delovanje pa izkorišča skupno bazo znanja. Kot je prikazano na shemi večagentnega sistema (glej Slika 7) igra vlogo agenta za obveščanje in uporablja tako podatke iz internega poslovanja podjetja, kot podatke, ki jih najde sam in drugi agenti za luščenje informacij na svetovnem spletu.



Slika 7: Arhitektura celotnega večagentnega sistema, katerega del je mobilni agent za poslovno obveščanje

Glavni cilj mobilnega agenta je pridobitev informacij o ponudbi podobnih izdelkov, ki jih ciljna organizacija prav tako ponuja in so se pojavili na svetovnem spletu ter jih organizacija še nima v svoji ponudbi. Agent tudi preveri ceno in morebitno ponudbo katerega izmed teh izdelkov pri konkurenčnih domačih podjetjih. Na koncu posreduje pridobljene informacije odgovorni osebi za sprejemanje odločitev s področja ponudbe izdelkov iz sorodnega prodajnega programa.

Pri svoji dejavnosti uporablja zmožnost seljenja na performančno učinkovitejše in glede virov unikatne lokacije ter delovanje na mobilnih in performančno šibkih napravah kot je npr. dlančnik.

Upravičenost mobilnega agenta se v tem primeru kaže v:

- Razbremenitvi brezžičnega ali mobilnega omrežja, ki je drago in počasno, saj se agent svoje naloge opravlja v stacionarnem omrežju.
- Hitro dostopanje do virov, ki jih ima samo določena lokacija:
    - Velika procesorska moč – lokalno procesiranje, saj agent izvaja svoje naloge lokalno z lokalnimi resursi in ne z oddaljenimi operacijami preko omrežja.

- Dostop, zbiranje in filtriranje podatkov – lokalni dostop do podatkovne baze, agent poišče lokacijo z ustrezno podatkovno bazo in izvede potrebne operacije nad podatki.
- Hiter dostop do interneta preko široko-pasovne povezave, saj agent poišče najmanj obremenjeno lokacijo z najhitrejšim dostopom.

- Podprta mobilnost uporabnika.

Avtonomnost kot glavna lastnost agentov še posebno pride do izraza pri mobilnih agentih, v omenjenm primeru se kaže predvsem v tem, da agent popolnoma avtonomno, brez pomoči uporabnika potuje po omrežju in išče lokacijo z iskano podatkovno bazo ter lokacijo z najhitrejšim dostopom do interneta.

Ker je agent mobilen lahko za večina nalog poskrbi sam zato družbenost kot pomembna lastnost v tem primeru seveda ne pride tako do izraza, kljub temu pa komunicira z vsaj enim agentom, ki skrbi za prikaz grafičnega vmesnika in komunikacijo z uporabnikom.

Reaktivnost agenta se kaže predvsem v njegovem zaznavanju okolja oziroma testiranju lokacij, ki so mu na voljo in na podlagi teh ugotovitev načrtovati nadaljnje akcije za dosego cilja oziroma izvršitve naloge.

Okolje v katerem agent deluje je nedostopno, nedeterministično, statično in zvezno.

4.1 JADE platforma

Platforma JADE je namenjena implementaciji večagentnih sistemov in je v celoti razvita v programskem jeziku Java. Je v fazi nenehnega razvoja, saj jo poskušajo razvijalci v največji meri uskladiti s specifikacijami FIPA [9, 17]. Celoten komunikacijski model FIPA je v osnovi implementiran v platformi JADE, z vsemi svojimi integriranimi komponentami kot so interakcijski protokol, ovojnica, ACL, vsebinski jeziki, kodirne sheme in transportni protokol. Platforma je lahko porazdeljena na različnih napravah, katere ne potrebujejo enakega operacijskega sistema, konfiguracija pa se lahko spremlja preko oddaljenega grafičnega vmesnika. Konfiguracija se lahko spreminja celo med delovanjem s selitvijo agentov iz ene naprave na drugo oziroma iz ene fizične lokacije na drugo, kamor in kadar je to potrebno.

Vsaka delujoča instanca JADE izvajalnega okolja se imenuje **vsebnik** in lahko vsebuje poljubno število agentov. Množica aktivnih vsebnikov pa se imenuje **platforma**. Poseben vsebnik – **glavni vsebnik** mora biti aktiven v vsaki platformi, saj predstavlja upravljalno središče, v katerega se registrirajo vsi ostali vsebniki takoj ob zagonu. Zato je to tudi prvi vsebnik, ki se zažene v platformi, medtem ko so mu vsi ostali vsebniki podrejeni in potrebujejo njegov naslov (gostitelja in številko vrat), na katerega se lahko registrirajo. Vsak JADE agent ima unikatno ime s katerim se identificira v platformi. S poznavanjem imen drugih agentov lahko transparentno komunicira z njimi ne glede na to, kje se fizično nahajajo oziroma brez poznavanja njihove dejanske lokacije.

Glavni vsebnik ima poleg zmožnosti registriranja drugih vsebnikov tudi dva posebna agenta, ki se samodejno aktivirata ob zagonu glavnega vsebnika in predstavljata nekakšno upravljalno in informacijsko središče platforme. **AMS agent** ima dve večji vlogi in sicer zagotavljanje unikatnega poimenovanja agentov ter predstavlja in izvaja avtoriteto v okviru platforme. To na primer pomeni, da lahko od AMS agenta zahtevamo kreiranje ali uničenje agenta v oddaljenem vsebniku. **DF agent** izvaja imeniško storitev rumenih strani, ki omogoča ostalim

agentom vpis svojih storitev in iskanje storitev ostalih agentov v globalnem katalogu. Iskanje in vpis storitev se izvaja preko ACL sporočil z ustreznim vsebinskim jezikom in ustrezno ontologijo po specifikaciji FIPA.

4.2 Realizacija mobilnosti

JADE nam omogoča razvoj in implementacijo mobilnih agentov, ki se lahko selijo ali kopirajo oz. klonirajo na različne računalnike v omrežju. Selitev ali kloniranje je samo prehodno stanje v agentovem življenjskem ciklu. Za njuno uspešno izvedbo pa se mora agent zavedati svoje lokacije in lokacije na katero bi se rad preselil. Selitev agenta pomeni pošiljanje agentove kode in stanja skozi mrežni kanal, ki se lahko sproži kadarkoli med agentovim delovanjem pri tem pa mora uspešno prestati serializacijski in deserializacijski proces. Najprej se agent ustavi in naredi posnetek iz katerega se na ciljni lokaciji naredi nov agent, ki čaka na aktivacijo. Nato se original na izvorni lokaciji ustavi, njegovi viri pa sprostijo. Nov agent se prijavi v platformo z identiteto originala, zasede vse potrebne vire in se aktivira. Nekateri agentovi viri se pri tem procesu preselijo z agentom na drugo lokacijo, medtem ko se drugi sprostijo in se na cilju ponovno zasedejo (npr. grafični vmesnik). Kloniranje poteka podobno, le da se original ne uniči, nov agent pa se prijavi s svojo novo identiteto.

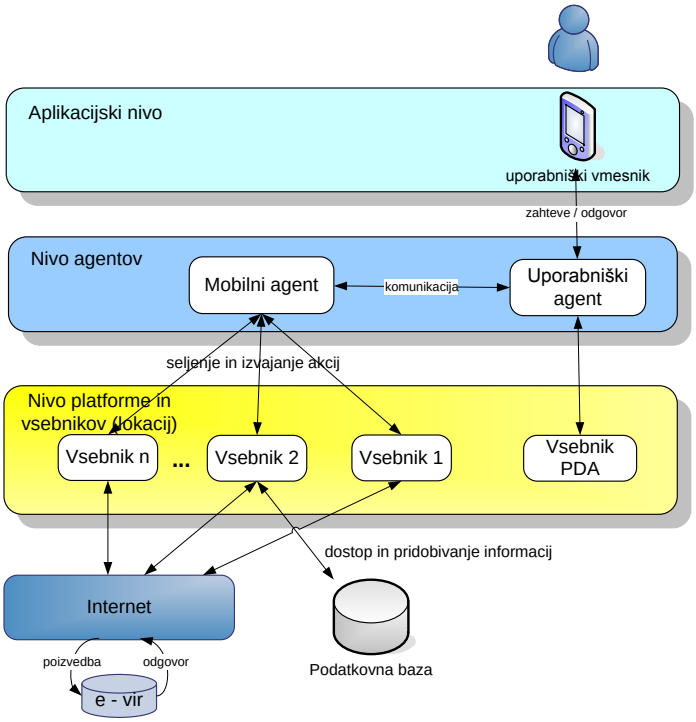
Kot posledica razvoja omrežij (GPRS, UMTS, WLAN) in vse močnejših prenosnih napravah kot so dlančniki in mobilni telefoni, so brezžična in stacionarna omrežja vse bolj povezana in integrirana. Zaradi tega so se pojavile potrebe po porazdeljenih aplikacijah, ki se izvajajo delno v stacionarnem omrežju in delno na prenosni napravi. Zaradi specifičnosti in omejenosti prenosnih naprav, je bil v okviru projekta LEAP-IST v marcu 2005 za platformo JADE razvit dodatek **LEAP**. V kombinaciji z JADE modificiranim izvajalnim okoljem se predstavlja kot JADE-LEAP. Uporablja se lahko na širokem izboru naprav, od strežnikov pa do mobilnih telefonov, ki podpirajo Javo. Za dosego takšne fleksibilnosti je bilo potrebno JADE-LEAP izvajalno okolje oblikovati v tri različne smeri, glede na tri izvajalna okolja Java, ki se uporabljajo na teh napravah.

- **J2SE**, kjer je izvajanje JADE-LEAP na osebnih računalnikih in strežnikih v stacionarnem omrežju ter podprto Javo 1.4.x.
- **PJAVA**, kjer je izvajanje JADE-LEAP na prenosnih napravah, ki podpirajo J2ME CDC ali PersonalJava, to je pa večina današnjih dlančnikov.
- **MIDP**, kjer je izvajanje JADE-LEAP na prenosnih napravah, ki podpirajo sam MIDP 1.0 ali pa MIDP 2.0, kar je večina današnjih Java podprtih mobilnih telefonov.

Zaradi širokega nabora naprav in velike raznolikosti med njimi tako po procesorski moči kot tudi po velikosti delovnega pomnilnika in drugih sistemskih virov, sta bila uvedena dva zagonska načina [4, 6, 16]:

- **Stand-alone** oz. samostojni zagon, je normalen zagon pri katerem se celoten vsebnik izvaja na napravi/gostitelju, ki je ta zagon izvedla.
- **Split** oz. deljen zagon je zagon pri katerem se vsebnik razdeli na uporabniški – lažji del in zaledni sistem – težji del vsebnika. Uporabniški del vsebnika se dejansko izvaja na napravi/gostitelju, kjer je bil zagon JADE-LEAP izvajalnega okolja aktiviran, medtem ko se zaledni sistem izvaja na oddaljenem strežniku. Oba dela vsebnika sta skupaj povezana preko trajne povezave.

4.3 Arhitektura in delovanje



Slika 8: Arhitektura sistema mobilnega agenta

Arhitekturo sistema mobilnega agenta lahko predstavimo kot med seboj povezani aplikacijski nivo, nivo agentov, nivo platforme in vsebnikov ter dostop do različnih virov informacij (glej Slika 8):

- **Aplikacijski nivo**

  Ta nivo predstavlja vmesnik med agenti in uporabnikom, saj vsebuje grafični uporabniški vmesnik preko katerega uporabnik ukazuje agentom, spremlja njihovo izvajanje in pregleduje rezultate. Glede na to, da je pri večagentnih sistemih najpomembnejše čim bolj inteligentno in avtonomno delovanje s čim manj interakcijami s strani uporabnika ter velikost, ki jo omejuje zaslon dlančnika, je grafični vmesnik sistema mobilnega agenta dokaj preprost. Uporabniku omogoča določitev števila zadetkov za prikaz, zagon mobilnega agenta ter pregledovanje rezultatov.

- **Nivo agentov**

  Osrednji del tega nivoja so agenti preko katerih se izvajajo vse akcije. Sestavljata ga statični uporabniški agent, ki sprejema uporabnikove zahteve in jih preko mobilnega agenta izvede, ter mobilni agent, ki predstavlja srce tega sistema. Slednji izvede vse potrebne akcije in selitve v druge vsebnike za dosego cilja oziroma pridobitve ustreznih informacij.

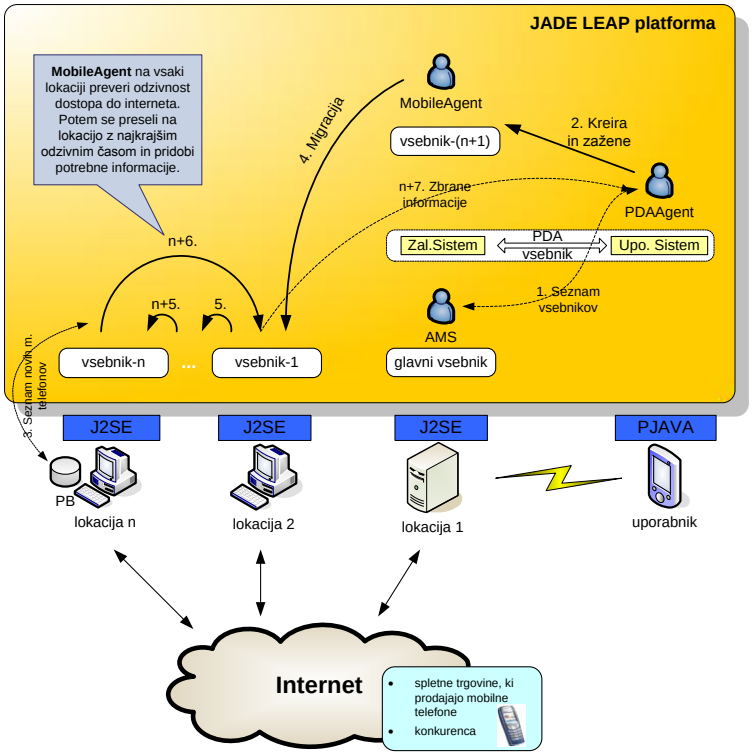
- **Nivo platforme in vsebnikov**

  Vsebniki predstavljajo posamezne izvajalne lokacije v katere se lahko mobilni agent seli, deluje in izkorišča njihove vire, v tem primeru dostop do interneta in do ustrezne podatkovne baze.

- **Dostop do različnih virov informacij**

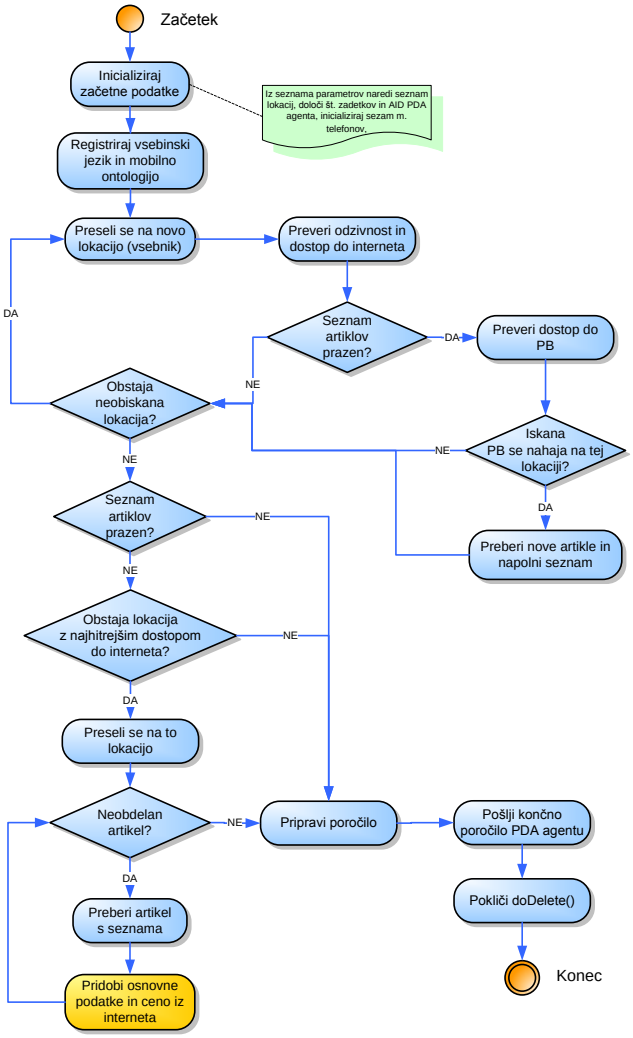
Glavna predstavnika sta internet za pregled potrebnih podatkov, povezanih s poslovanjem organizacije (stanje ponudbe, konkurenca ipd.) ter podatkovna baza skupnega večagentnega sistema. Slednja hrani podatke in statistične preglede o prodaji ter ponudbi izdelkov, s katerimi organizacija posluje. Za ta sistem zanimivi podatki so predvsem novi izdelki iz iste kategorije, ki so se pojavili na svetovnem trgu in jih organizacija še nima v svoji ponudbi. Za pridobitev teh informacij skrbi poseben agent v okviru sistema MAS.

Ob predpostavki, da je JADE LEAP platforma aktivna in dosegljiva preko brezžičnega omrežja, se samo delovanje tega sistema prične z uporabnikovo aktivacijo PDA agenta na dlančniku. Pri tem se ustvari deljeni vsebnik, katerega lažji del oz. uporabniški sistem se nahaja na dlančniku, težji del oz. zaledni sistem pa na računalniku z glavnim vsebnikom. Po aktivaciji in v fazi inicializacije zaprosi PDA agent preko ACL sporočila AMS agenta za seznam vseh dosegljivih vsebnikov (glej Slika 9).



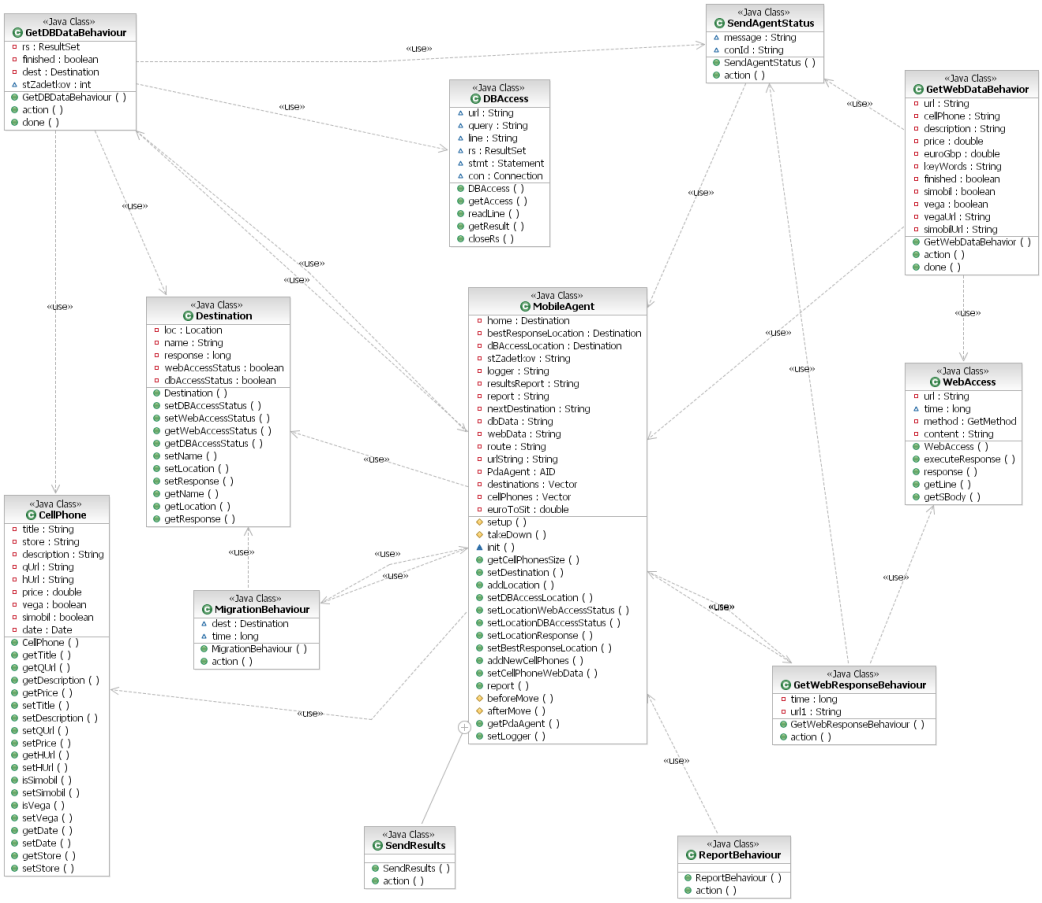
Slika 9: Delovanje sistema mobilnega agenta

Uporabnik lahko določi maksimalno število novih izdelkov, ki naj jih agent poišče in prikaže. PDA agent kreira nov samostojen vsebnik z mobilnim agentom, ki mu preko parametrov posreduje vse potrebne informacije za svoje. Mobilni agent se nato poda na pot, pri čemer se seli iz lokacije na lokacijo s seznama, pri tem pa preverja odzivnost in dostop do interneta. Hkrati preverja tudi prisotnost in dostop do ustrezne podatkovne baze. V primeru njene prisotnosti, agent prebere potrebne podatke (poslovna poročila, opozorila ipd.) in je na nadaljnjih lokacijah ne išče več. Po uspešnem testiranju vseh lokacij se mobilni agent preseli na tisto z najkrajšim odzivnim časom. Tam pridobi za vsak izdelek s seznama opis, tehnične lastnosti, ceno ter preveri morebitno ponudbo pri domači konkurenci. Po pridobitvi iskanih podatkov pripravi končno poročilo in ga pošlje PDA agentu, sam pa se ugasne (glej Slika 10).



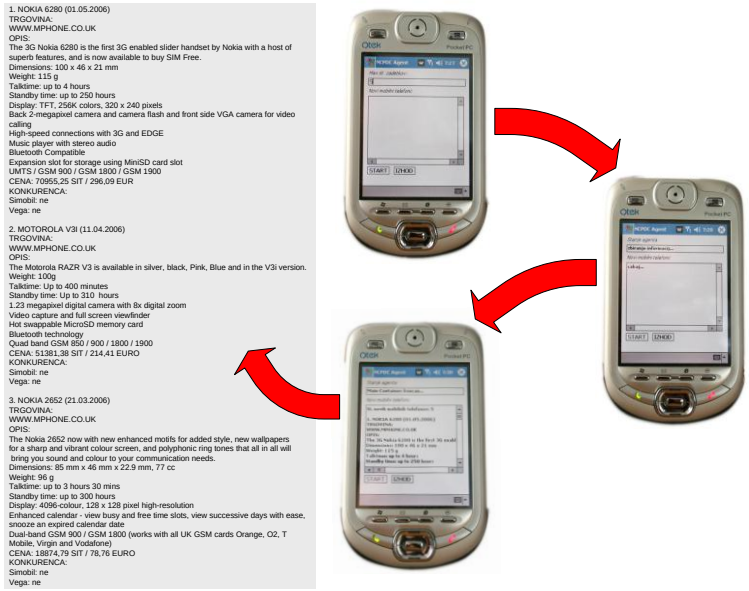
Slika 10: Diagram poteka mobilnega agenta

Razredni diagram prikazuje statično strukturo mobilnega agenta. Glavni razred predstavlja razred *MobileAgent* in skrbi za koordinacijo, komuniciranje ter izvajanje akcij (glej Slika 11).



Slika 11: Razredni diagram mobilnega agenta

Uporabnik lahko v glavnem polju pregleduje končno poročilo, ki mu ga po prejetju prikaže PDA agent (glej Slika 12).



Slika 12: Primer enostavnega grafičnega vmesnika

Razviti mobilni agent tako predstavlja pomemben element v večagentnem sistemu v katerem deluje in opravlja vlogo agenta za obveščanje. Njegov glavni cilj je pravočasno obveščanje uporabnikov, ki niso vedno dosegljivi na standarden način – elektronska pošta, aplikacije

nameščene na delovnih postajah, spletni portali itd. Poudarek pri razvoju ni bil na grafičnem vmesniku, ampak učinkovitosti mobilnega agenta pri iskanju informacij po lokalnih virih in svetovnem spletu.

5 SKLEP

Večagentni sistemi in še posebej mobilni agenti so razvijajoče se področje oziroma pristop razvoja programske opreme in so nekakšna evolucija objektnega pristopa. Zaradi tega razloga je ta pristop še vedno v stopnji razvoja in sprejemanja standardov, njegova uporaba pa predvsem stvar raziskovalnih krogov. Seveda je bil z ustanovitvijo osrednje organizacije FIPA, ki bdi nad razvojem področja večagentnih sistemov in sprejema standarde, narejen velik korak naprej. Z uspešno implementacijo teh standardov (komunikacija med agenti, upravljanje z agenti, ontologije itd.) je postala platforma JADE zelo zanimivo in uporabno orodje za implementacijo večagentnih sistemov in mobilnih agentov. Dodatek LEAP je vso to uporabnost razširil še na področje mobilnih naprav, ki so procesorsko in pomnilniško šibkejše.

Predvsem omenjene lastnosti ter zelo dobra podpora razvijalcev platforme in ostalih uporabnikov, ki jo nudijo preko spletnih strani in spletnih forumov, so bili ključni dejavniki za izbor platforme JADE kot osnove pri razvoju mobilnega agenta.

Največja težava pri razvoju je bila vsekakor ta, da je bilo pri prehodu iz stopnje testiranja na računalniku na stopnjo testiranja na dlančniku ugotovljeno, da na njem ni mogoča implementacija vsebnika s samostojnim načinom zagona. To pa pomeni, da je direktna implementacija mobilnega agenta, ki bi se selil in vrnil z rezultati nemogoča.

Potrebna je bila sprememba koncepta in uporaba deljenega vsebnika, ki ne podpira mobilnih agentov ter vmesnega PDA agenta za upravljanje z grafičnim vmesnikom in posredovanje informacij med uporabnikom in mobilnim agentom.

Glavne možnosti za nadaljnje raziskave so vsekakor predlogi metodologij in modeliranja, ki podpirajo mobilne agente, določitve standardov in meritev, ki bi opravičevale in predlagale uporabo mobilnih agentov, razvoj mehanizmov varnosti, tako za agente kot lokacije, oddaljena vedenja itd.

6 SEZNAM UPORABLJENIH VIROV

[1] **Aderounmu, G.A.**, Performance comparison of remote procedure calling and mobile agent approach to control and data transfer in distributed computing environment. *Journal of Network and Computer Applications*, 2004. **27**(2): str. 113-129.

[2] **Anumba, C.J., et al.**, Negotiation within a multi-agent system for the collaborative design of light industrial buildings. *Advances in Engineering Software*, 2003. **34**(7): str. 389-401.

[3] **Bates, J.**, The Role of Emotion in Believable Agents. *Communications of the Acm*, 1994. **37**(7): str. 122-125.

[4] **Bellifemine, F., G. Caire, in T. Trucco**. JADE Programmer's guide. Posodobljeno 21.8.2006; dostopno na: http://jade.tilab.com/doc/programmersguide.pdf.

[5] **Bredin, J., et al.**, Computational markets to regulate mobile-agent systems. *Autonomous Agents and Multi-Agent Systems*, 2003. **6**(3): str. 235-263.

[6] **Caire, G. in F. Pieri**. LEAP User guide. Posodobljeno 20.1.2006; dostopno na: http://jade.tilab.com/doc/LEAPUserGuide.pdf.

[7] **Eid, M., et al.**, Trends in mobile agent applications. *Journal of Research and Practice in Information Technology*, 2005. **37**(4): str. 323-351.

[8] **Genesereth, M.R. in S.P. Ketchpel**, Software Agents. *Communications of the Acm*, 1994. **37**(7): str. 48-&.

[9] **Greenwood, D.** FIPA - The Foundation for Intelligent, Physical Agents. Posodobljeno 10.5.2004; dostopno na: http://jade.tilab.com/papers/JADETutorialIEEE/JADETutorial_FIPA.pdf.

[10] **Jones, K.** Mobile Agents in Distributed Environments: Principles and Paradigms. Posodobljeno 20.1.2005; dostopno na: http://www.cse.dmu.ac.uk/~kij/Research/presentations/MobileAgentsP&P/MobileAgentsP&P.pdf.

[11] **Karnik, N.M. in A.R. Tripathi**, Design issues in mobile-agent programming systems. *Ieee Concurrency*, 1998. **6**(3): str. 52.

[12] **Lange, D.B. in O. Mitsuru**, Programming and Deploying Java Mobile Agents Aglets. 1st ed. 1998, Boston, USA: Addison-Wesley Longman Publishing.

[13] **Lavbič, D.**, Uporaba inteligentnih agentov, v *Information Systems Laboratory*. 2004, University of Ljubljana: Ljubljana, Slovenia.

[14] **Lavbič, D., R. Rupnik, in M. Krisper**. Poslovna pravila v večagentnih sistemih. v zborniku '*8. mednarodna multikonferenca Informacijska družba IS 2005'*. 2005. Ljubljana, Slovenia.

[15] **Magedanz, T. in A. Karmouch**, Mobile software agents for telecommunication applications. *Computer Communications*, 2000. **23**(8): str. 705-707.

[16] **Moreno, A., A. Valls, in A. Viejo**. Using JADE-LEAP to implement agents in mobile devices. Posodobljeno 2005; dostopno na: http://jade.tilab.com/papers/EXP/02Moreno.pdf.

[17] **Nikraz, M., G. Caire, in P.A. Bahri**, A methodology for the development of multi-agent systems using the JADE platform. *Computer Systems Science and Engineering*, 2006. **21**(2): str. 99-116.

[18] **Nwana, H.S.**, Software agents: An overview. *Knowledge Engineering Review*, 1996. **11**(3): str. 205-244.

[19] **Papaioannou, T. in J. Edwards**, Building agile systems with mobile code. *Autonomous Agents and Multi-Agent Systems*, 2001. **4**(4): str. 293-310.

[20] **Russell, S.J. in P. Norvig**, Artificial Intelligence: Modern Approach. 2nd ed. 2002, New York, USA: Prentice Hall.

[21] **Saleh, K. in C. El-Morr**, M-UML: an extension to UML for the modeling of mobile agent-based software systems. *Information and Software Technology*, 2004. **46**(4): str. 219-227.

[22] **Shoham, Y.**, Agent-Oriented Programming. *Artificial Intelligence*, 1993. **60**(1): str. 51-92.

[23] **White, J.E.**, Mobile agents, v *Software agents*. 1997, MIT Press: Cambridge, USA.

[24] **Wooldridge, M.**, An Introduction to MultiAgent Systems. 2002, Chichester, England: John Wiley & Sons.

[25] **Wooldridge, M. in N.R. Jennings**, Intelligent Agents - Theory and Practice. *Knowledge Engineering Review*, 1995. **10**(2): str. 115-152.

[26] **Xu, H.P., Z.G. Zhang, in S.M. Shatz**, A security based model for mobile agent software systems. *International Journal of Software Engineering and Knowledge Engineering*, 2005. **15**(4): str. 719-746.

[27] **Zerfiridis, K.G. in H.D. Karatza**, Brute force web search for wireless devices using mobile agents. *Journal of Systems and Software*, 2004. **69**(1-2): str. 195-206.