REPUBLIKA E SHQIPËRISË

UNIVERSITETI POLITEKNIK I TIRANËS

FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT

DEPARTAMENTI I INXHINIERISË INFORMATIKE

EVA ÇIPI

Për marrjen e gradës

“ Doktor”

Në Inxhinieri Informatike

DISERTACION

SISTEMET MULTIAGJENTE SI ARKITEKTURA TË MODULUARA NË ZHVILLIMIN E APLIKACIONEVE SOFTUERE

Udhëheqës Shkencor

PROF. DR. BETIM ÇIÇO

TIRANË, 2013

SISTEMET MULTIAGJENTE SI ARKITEKTURA TË MODULUARA NË ZHVILLIMIN E APLIKACIONEVE SOFTUERE

Disertacioni

i paraqitur në Universitetin Politeknik të Tiranës

për marrjen e gradës

“ Doktor”

në

Inxhinieri Informatike

nga

Znj. Eva Çipi

2013

Disertacioni i shkruar nga

Znj. Eva Çipi

Master Shkencor, Universiteti Politeknik i Tiranës, Shqipëri, 2008

DIND, Universiteti Politeknik i Tiranës, Shqipëri, 1987

I aprovuar nga

_____ Kryetar i Komisionit të Disertacionit të Doktoraturës

_____ Anëtar i Komisionit të Disertacionit të Doktoraturës

_____ Anëtar i Komisionit të Disertacionit të Doktoraturës

_____ Anëtar i Komisionit të Disertacionit të Doktoraturës

_____ Anëtar i Komisionit të Disertacionit të Doktoraturës

I pranuar nga

_____Dekan , Fakulteti i Teknologjisë së Informacionit

Abstrakt

Zhvillimi dhe manaxhimi i aplikimeve të shpërndara në ditët e sotme është i vështirë. Ekzistenca e kompleksitetit në rritje që karakterizon familjen e madhe të sistemeve softuere vjen nga disa faktorë. Aktorët e përfshirë në këto sisteme, shpeshherë kanë kërkesa të cilësisë që janë kontradiktore. Sistemet janë subjekt i kushteve mjaft dinamike dhe ndryshimeve operative, ndërkohë që funksionalitetet në sisteme janë të lokalizuara, ndërsa është e vështirë ose gati e pamundur të arrihet kontrolli global.

Ky punim paraqet përpjekjet e bera për të ndërtuar një model për zhvillimin e sistemeve të tilla komplekse i cili integron sistemet e bazuar në agjentë si arkitektura softuare në rrjedhën e proçeseve të inxhinierisë softuere. Aspektet kryesore të procesit të modelimit janë zhvillimi i arkitekturës softuere, vetë-manaxhimi dhe kontrolli i decentralizuar. Zhvillimi i një arkitekture referenciale softuere detyron aktorët e përfshirë në proceset e zhvillimit të një sistemi softuere për t'u marrë në mënyrë të qartë me objektivat e cilësisë mbi bazën e kërkesave të ndryshme të sistemit. Vetë-manaxhimi mundëson një sistemi kompjuterik për t'u marrë në mënyrë autonome me dinamikën e ndryshimit të rrethanave në të cilat atij i duhet të veprojë. Kontrolli i decentralizuar është thelbësor për t'u përballuar me lokalizimin e pandarë të aktiviteteteve në një sistem ku kontrolli global nuk është një opsion dhe funksionaliteti i sistemit duhet të arrihet nëpërmjet bashkëpunimit të nënsistemeve.

Punimi në thelb është një model bazë për sistemet multiagjente që integron mjedisin si një abstraksion në nivelin e parë të projektimit me model agjenti duke ofruar mekanizma projektimi të gatshëm për të fituar sjellje adaptive të sistemit. Këto mekanizma mundësojnë që agjentët të mund të manaxhojnë situata dinamike në mjedis në mënyrë autonome. Sistemi multiagjent mund të përballet mjaft mire me procese zhvillimi në inxhinierinë e softit në të cilat ka agjentë që largohen nga sistemi si dhe agjentë të rinj që hyjnë dhe integrohen në sistem. Zakonisht kontrolli në një sistem multiagjent është i decentralizuar. Kjo do të thotë se agjentët janë implementuar në mënyrë që të bashkëpunojnë për të arritur funksionalitetin e përgjithshëm të sistemit.

Mbështetur mbi një bazë njohurish teorike nga artikuj të shumtë dhe literaturë shkencore mbi argumetin në fjalë si dhe nga përvojat tona në aplikacione të zhvilluara me sisteme multiagjente, ne kemi zhvilluar një sistem vetëmanaxhues duke përdorur një arkitekturë që i referohet arkitekturës së modeluar mbi stilin e dekompozimit bazuar në module komponentësh të përgjthshëm ekzistues për sistemet multiagjente.

Projektimi dhe zhvillimi i këtij sistemi ka kontribuar ndjeshëm në zhvillimin e një modeli bazë referencial duke diskutuar inkrementimin dhe riinxhinierizimin e kësaj

arkitekture me konkluzione pozitive për përdoueshmërinë e modelit në proceset e inxhinierisë së softit në aplikacione bazuar në zgjidhje me agjentë. Gjthashtu, zhvillimi i suksesshëm i aplikacionit në tre faza është sfidues dhe tregon se si sistemet multiagjente mund të integrohen si module softuare të zakonshëm në arkitekturën e sistemeve inteligjente në inxhinierinë softuare.

***Fjalë kyçe**: sistema multiagjente, vetëmanaxhim, projektim akitekturor, agjent, model bazë, dekompozim, mjedis, kontekst i vendosjes.*

PËRMBAJTJA

LISTA E FIGURAVE

LISTA E TABELAVE

MIRËNJOHJE

Dëshiroj të falenderoj të gjithë kolegët e mi që më kuptuan dhe më mbështetën gjatë punimit të këtij disertacioni. Mjediset e qendrës së kërkimit në të cilat u zhvillua ky punim, lehtësuan realizimin e eksperimenteve dhe provave të shpeshta duke bërë të mundur zhvillimin me sukses të aplikacionit që finalizoi disertacionin në përfundim.

Një falenderim i veçantë shkon për Profesor Betim Çiço që më drejtoi në këtë punim me përkushtim!

Eva Çipi

2013, Tiranë

KAPITULLI 1

Hyrje

Zhvillimi dhe manaxhimi i aplikacioneve të shpërndara softuere në ditët e sotme është i vështirë për shkak të kompleksitetit në rritje. Duke iu referuar kompleksitetit, ekzistojne tre arsye të rëndësishme mbi të cilat motivohet ky hulumtim. Një arsye është kërkesa në rritje për cilësinë e programeve[1]. Aktorë të ndryshëm të interesuar në zhvillimin e sistemeve softuere (përdorues, drejtues të projekteve, projektues, zhvillues apo administrator mirëmbajtësj) zakonisht kanë kërkesa të ndryshme mbi atributeve të cilësisë së sistemeve softuere siç janë performanca, fleksibiliteti, përdorueshmëria, etj). Përballja me këto kërkesa si dhe kufizime të llojeve të ndryshme të projekteve dhe ato të biznesit (buxhetet, koha, etj), zakonisht përbëjnë sfida të vështira të inxhinierisë softuere. Një arsye e dytë e rëndësishme pse zhvillimi dhe manaxhimi i aplikacioneve softuere vazhdon të jetë i vështirë, është ekzistenca e dinamikës së kushteve në të cilat veprojnë sot aplikacionet e shpërndara. Kushte të tilla imponojnë përballje me ngarkesa pune që vijnë për shkak të ndryshimit në mënyrë të paparashikueshme të kërkesave funksionale dhe jofunksionale, të disponueshmërisë së burimeve dhe shërbimeve, veçanërisht gjendjet e dështimeve të sistemit, variacionet në rrjetat me trafik, etj. Komunikimi wireless fut mundësi të mëdha për integrimin e vazhdueshëm të burimeve elektronike. Gjithashtu, ai shfaq kompleksitete shtesë për shkak të përhapjes së rrjetit dhe ndryshimeve të vazhdueshme topologjike në rrjet.[2] Përveç kompleksitetit që rrjedh nga atributet e cilësisë së kërkuar dhe kushteve dinamike operative, natyra e aplikacioneve të tilla bën që të jetë e vështirë të arrihet kontrolli global. Në këtë hulumtim ne propozojmë një model arkitekturë bazë për të zhvilluar këto aplikacione komplekse softuere. Kjo është e rëndësishme për manaxhimin e kompleksitetit dhe arritjen e atributeve të kërkuar të cilësisë së sistemit softuere. Modeli synon të arrihet vetë-manaxhimi i sistemeve softuere duke qenë në gjendje të manaxhojnë dinamikën dhe situatat e pararashikuara në mënyrë autonome.

Në qendër të modelit arkitekturor është një sistem multiagjent. Sistemet e bazuar në agjente sigurojnë një mënyrë për modelimin e vetë-manaxhimit të sistemeve të decentralizuara me qëllim që të arrihen objektivat e synuar në nivel lokal.

1.1. Aplikacionet e vetëmanaxhueshme

Teknologjia e informacionit po përballet me problemet komplekse në rritje të manaxhimit në aplikacionet e shpërndara dhe tashmë janë marrë iniciativa për t'u përballur dhe zgjidhur këto probleme. Këto dhe iniciativa të ngjashme njohin atributin

e vetëmanaxhimit, si alternativë mjaft komode për të përballuar me sukses problemet në rritje të projektimit arkitekturor të sistemeve softuere. Idea e përgjithshme e vetëmanaxhimit është aftësia e sistemeve softuere për t'u vetëmanaxhuar, sipas disa objektivave në nivel të lartë, të specifikuar nga njerëzit. Në [1] vetë-manaxhimi ndahet në katër fusha funksionale:

- Vetëkonfigurim: komponentet të përshtaten automatikisht në mjedise të ndryshme;
- Vetëshërim: komponentet automatikisht zbulojnë, diagnostikojnë dhe korrigjojnë gabimet.
- Vetëoptimizim: komponentet automatikisht monitorojnë dhe përshtasin burime për të siguruar funksionimin optimal në lidhje me kërkesat
- Vetëmbrojtje: komponentet parashikojnë, identifikojnë dhe mbrohen nga sulmet arbitrare.

Për të mundësuar vetëmanaxhimin, aplikacionet e shpërndara duhet të jenë të pajisura me teknika të përshtatshme dhe të jenë mbështetur nga infrastruktura e duhur. Në aplikacionet ekonomike, ato zënë një pozicion strategjik të rëndësishëm. Në një proces vetëmanaxhimi, sistemi duhet të marrë veprimet e duhura bazuar në situatën që ndjen në mjedis. Kjo kërkon funksionalitete shtesë për monitorimin, marrjen e vendimeve dhe ekzekutim. Rregullimet lokale duhet të jenë në përputhje me sistemin e gjerë të vetëmanaxhimit të politikave. Kjo kërkon koordinimin e sjelljes së sistemit brenda dhe jashtë mjedisit në aplikacione të ndryshme. Për të arritur këto sfida, në qendër të kërkimeve ose hulumtimeve kanë qenë një sërë teknikash, të tilla si teknikat për problemin e eksplorimit (probing)të diskutuar në [2], të mësuarit dhe teknikat e përshtatjes te [3], mekanizmat e tregut te [4], si dhe teknikat për të kontrolluar dhe shfrytëzuar sjellje emergjente [5]. Megjithatë, realizimi më i fundit i vetëmënaxhimit paraqitet në [6]. Në hulumtimin tonë, ne e konsiderojmë vetëmanaxhimin si aftësinë e një sistemi për të manaxhuar dinamikën dhe duke kryer ndryshime në mënyrë autonome. Me dinamikë dhe ndryshime i referohemi rrethanave të ndryshueshme të një sistemi që mund t'i paraqiten gjatë ekzekutimit të veprimit, të tilla si adresat e ndryshuara të punës, variacione në disponueshmërinë e burimeve dhe shërbimeve, si dhe nënsistemet që i shtohen ose largohen nga sistemi. Fokusi ynë në konceptin e vetëmanaxhimit është i lidhur ngushtë me vetë-optimizimin dhe vetëshërimin. Vetëkonfigurimi dhe vetëmbrojtja janë jashtë objektivit të këtij hulumtimi.

Kërkimi shkencor në sistemet multiagjente ka të bëjë me studimin, sjelljen, dhe ndërtimin e një koleksioni të agjentëve autonome që ndërveprojnë me njëri-tjetrin ose me mjedisin e tyre [7]. Në [8] përkufizohet një sistem multiagjent si një bashkësi veprimesh e entiteteve që zgjidhin probleme (agjentët), me qëllim që të

bashkëveprojnë për të zgjidhur problemet që janë përtej aftësive individuale ose njohurive të çdo bashkësie tjetër entitetesh.

Në përgjithësi, karakteristika të sistemeve multiagjente mund të përmendim:

- Çdo agjent ka informacion jo të plotë apo aftësi për zgjidhjen e problemit dhe, si rrjrdhim, ka një këndvështrim të kufizuar drejt problemit.
- Nuk ka një sistem të kontrollit global;
- Të dhënat janë të shpërndara, dhe
- Përpunimi i të dhënave dhe llogaritja është asinkrone.

Sistemet multiagjente robotike janë një familje e sistemeve multiagjente. Sistemi multiagjent përbëhet nga një mjedis i shpërndarë populluar me një grup agjentësh me qëllim që të bashkëpunojnë për të zgjidhur një problem të ndërlikuar në mënyrë të decentralizuar. Një agjent i vendosur ka një pozicion të qartë në mjedis dhe ka modelin e vet lokal për mjedisin, dmth çdo agjent është vendosur në një kontekst lokal dhe, në këtë mënyrë, ai mund të perceptojë dhe mund të veprojë ose të bashkëveprojë me agjentë të tjerë. Rezultatet e funksionalitetit të sistemit varen nga ndërveprimet e agjentëve në mjedis dhe jo nga aftësitë e tyre individuale. Një agjent robotik i konsideruar inteligjent përdor një sjellje të bazuar në mekanizmin e zgjedhjes së veprimit për të ekzekutuar veprimin e duhur. Zgjedhja e sjelljes e bazuar në veprim është e nxitur nga stimuj të perceptuar në mjedisin rrethues, si edhe stimuj të brendshëm. Të tillë agjentët punojnë në sajë të gjendjes së brendshme vendim-marrje për sa kohë kjo gjendje ka të bëjë me :

- informacione të përgjithshme statike të sistemit (p.sh. rregulla me prioritet të përcaktuar);
- informacion në lidhje me dinamikën e agjentit në kontekstin prezent të ndodhjes (p.sh. një marrëveshje e përkohshme për bashkëpunim me një agjent fqinjë), ose
- çështje të brendshme për agjentin (p.sh. një vlerë kufi që përdoret për të ndryshuar rolet).

Në sistemet multiagjente robotike, mjedisi është një pjesë thelbësore e sistemit. Mjedisi enkapsulon burime dhe mundëson agjentët për të patur akses në këto burime. Për më tepër, mjedisi mundëson agjentët për të bashkëvepruar me njëri tjetrin, si dhe shërben si ndërmjetës për agjentët për të ndarë informacionin dhe për të koordinuar aktivitetetin e tyre nëpërmjet shfaqjes së sjelljeve të ndryshme. Shembull i tillë është një *pheromone* artificial.

Pheromone është një strukturë dinamike në mjedis që krijohet dhe lëshohet për pika të caktuara të mjedisit duke dhënë informacion nëpërmjet shënimit të hapsirës, e cila avullon me kalimin e kohës. Agjentët mund të përdorin pheromonet dinamike për të përcaktuar shtigje kalimi drejt arritjes së objektivit. Për shkëmbimin e drejtpërdrejtë të informacionit nëpërmjet mjedisit, ndihmojne agjentët e kontrollit të cilët mundësojnë koordinimin. Kontrolli në një sistem multiagjent është i shpërndarë kështu që agjentët ndërmjetësues të kontrollit qëndrojnë midis agjentëve të tjerë dhe mjedisit.

Vetë-manaxhimi. Në një sistem multiagjent, vetë-manaxhimi është zakonisht i bazuar në aftësinë e agjentëve për të përshtatur sjelljen e tyre. Për shkak të efiçencës së përzgjedhjes së veprimit, agjentët mund t'u përgjigjen me shpejtësi ndryshimit të rrethanave. Përveç kësaj forme të fleksibilitetit të çastit, janë zhvilluar mekanizma specifike që lejojnë agjentët për të përshtatur sjelljet e tyre me kalimin e kohës. Mekanizmat për përshtatjen e sjelljeve mundësojnë agjentët për të kaluar në sjellje më të përshtatshme kur rrethanat ndryshojnë në mjedis. Mjedisi mund të luajë një rol aktiv në vetë-manaxhim. Një shembull i këtij percaktimi është një pheromon për rrugën e ndjekur që zhduket me kalimin e kohës, duke i penguar agjentët për të ndjekur rrugën e vjetëruar.

1.2. Sistemet multiagjente dhe arkitektura softuere

Hulumtimi ynë sjell një model bazuar mbi një arkitekturë bazë për proçesin e inxhinierisë softuere me sistemet multiagjent. Perspektiva jona mbi qëllimin kryesor lidhur me sistemet multiagjente është si vijon:

Një sistem multiagjent ofron një softuere për të zgjidhur një problem nëpërmjet strukturimit të sistemit në një numër entitetesh autonome që bashkëveprojne, të vendosur në një mjedis për të arritur kërkesat funksionale dhe cilësinë e sistemit. Kjo strukturë pikërisht përbën arkitekturën softuere. Në [9] përkufizohet arkitektura softuere si një strukturë apo struktura të sistemit, të përbëra nga elemente softuere, me karakteristika dhe atribute të dukshme të këtyre elementëve, si dhe marrëdhëniet ndërmjet tyre. Elemente softuere (ose në përgjithësi elementët arkitekturore) sigurojnë funksionalitetin nga pikëpamja cilësore e sistemit, ndërsa atributete të kërkuar të cilësisë si performanca, përdorshmëria, sigurohen kryesisht përmes strukturimit të arkitekturës softuere.

Si të tillë, sistemet multiagjente janë në thelb një familje ende e madhe e arkitekturave softuere. Bazuar në analizën e problemit që jep kërkesat funksionale dhe atributet e cilësisë së sistemit, nje zhvillues softuere mund ose nuk mund të zgjedh një zgjidhje të bazuar në sistem multiagjent. Kërkesa për cilësinë të tilla si fleksibiliteti, hapja dhe qëndrueshmëria mund të jenë argumente për projektuesin për të zgjedhur një arkitekturë softuere të sistemeve multiagjente. Kështu, ne konsiderojmë sistemet

multiagjente si një familje të modeleve të vlefshme për të zgjidhur problemet softuere në një spektër të madh të sistemeve teknologjike të mundshme që ekzistojnë për të dhënë zgjidhjen.

Elementët tipike arkitekturore të një arkitekture softuere të sistemit multiagjent janë agjentët, mjedisi, burimet, shërbimet, etj. Marrëdhëniet midis elementëve janë shumë të ndryshme, duke filluar nga mjedisi ndërmjetësues, në ndërveprimin mes agjentëve bashkëpunues, protokollet komplekse të komunikimit në një shoqëri të vetëinteresuar të agjentëve. Me pak fjalë, sistemet multiagjent jane një familje e pasur me arkitektura te ngjashme por me karakteristika specifike, të dobishëm për një larmi fushash të aplikacioneve me shkallë të ndryshme vështirësie.

Arkitektura bazë për sistemet multiagjente

Në [10] përkufizohet një arkitekturë bazë si një arkitekturë e përgjithësuar e sistemeve të ndryshme që i përkasin një fushe të përbashkët. Një arkitekturë e tille ofron një bazë për zhvillimin e arkitekturave të tjera softuere të klasave të sistemeve që ndajnë të njëjtën bazë arkitekturore së cilës i referohen. Në aktivitetin tonë kerkimor, ne kemi zhvilluar një model bazë si arkitekturë referimi për multiagjente. Arkitektura e sistemit përmbledh njohuritë dhe ekspertizën e fituar gjatë zhvillimit të aplikacioneve bazuar në modele arkitekturore. Një model arkitekturor përgjithëson dhe nxjerr në pah funksionet e përbashkëta, strukturat e aplikacioneve të ndryshme eksperimentale, që ne kemi studiuar dhe ndërtuar. Këto aplikacione ofrojnë nivele të ndryshme të kompleksitetit si dhe forma të ndryshme të dinamikës. Aplikacionet e realizuara si simulime përfshijnë realizimin e një prototipi sistemi bazuar në agjentë, të përfshirë në rastin e simulimit të sistemit të navigimit të mjetit lundrues, simulimi i navigimit në mjedis me pengesa dinamike dhe të paparashikuara, sistemi i autotransportit për sistemin e manaxhimit të një supermarketi, etj. Një modeli i krijuar arkitekturor përbën një pasuri për ripërdorim, mund të shërbejë si një model për zhvillimin e arkitekturave softuere për llojin e aplikimeve të shpërndara qe në synojmë në këtë kërkim. Gjithashtu, mund të shërbëje si mjet për të studiuar dhe përmirësuar ekspertizën mbi strukturat themelore dhe mekanizmat e vendosur në sistemet multiagjente.

Praktika aktuale në inxhinierinë softuere të orientuar nga agjenti. Kërkimi ynë ka për qëllim të integrojë sistemet multiagjente si mjete kryesore që japin zgjidhje të shpejta dhe efiçiente në inxhinierinë softuere, në eksperiencën aktuale konsiderohen sistemet multiagjent si një mënyrë rrënjësisht e re e inxhinierisë së softit. Ekzistojnë mjaft punime lidhur me këtë çështje që citojnë se :

“Ka një mospërputhje në mes të koncepteve themelore të përdorura nga zhvilluesit e orientuar nga objektet dhe të programeve të tjerë të paradigmave kryesore inxhinierike, si dhe këndvështrimit të orientuar nga agjenti. Teknikat

ekzistuese softuere të zhvillimit janë të papërshtatshme për të realizuar potencialin e agjentëve si një paradigmë e inxhinierisë softuere." [11]

"*Nëse modeli i orientuar nga agjenti konsiderohet si një paradigmë e inxhinierisë softuere (përcaktohet nga shkalla) në të cilën agjentët përfaqësohen, kjo jep një largim radikal nga koncepti i të menduarit të tanishëm në inxhinierinë softuere "*. [12]

"*Ne jemi në prag të një zhvendosjeje në një paradigmë revolucionare, të udhëhequr nga komuniteti i sistemeve multiagjente, dhe ka të ngjarë të ndryshojnë qëndrimet tona në modelimin softuere dhe inxhinierinë e softit." [13]*

"*Përpunimi i bazuar në agjentë mund të konsiderohet si një paradigmë të re me qëllim të përgjithshëm për zhvillimin e softuerit, i cili tenton në mënyrë rrënjësore të influencojë në rrugën në të cilën një sistem softuere është konceptuar dhe është zhvilluar." [14]*

Ky vizion ka çuar në zhvillimin e metodologjive të shumta të sistemeve multiagjente. Disa nga metodologjitë përqëndrohen në faza të veçanta të proçesit të zhvillimit të softuerit, p.sh. Gaia diskutuar në [15]. Metodologji të tjera mbulojnë në mënyrë të plotë gjithe ciklin e jetës së zhvillimit të softuerit. Një shembull është Tropos te [16]. Disa metodologji të tjera të propozuara pranojnë mekanizmat dhe praktikat nga inxhinieria softuere e zakonshme. Ne [17] autori fokusohet në mekanizmat e orientuar nga objekti. Ndërsa te [18] përdoren praktika të proçesit të unifikuar. Në [19] përdoren konstruktet e gjuhës së modelimit të unifikuar. Megjithatë, pothuajse të gjitha metodologjitë marrin një pozicion të pavarur në lidhje me praktikën e përgjithshme të inxhinierisë softuere. Inisiativa e rivlerësimit të sistemeve multiagjente si një paradigmë e re radikale për zhvillimin e softuerit në vend të vlerësimit të sistemeve multiagjente si model i ri në inxhinierinë softuere nga inxhinieria softuere e zakonshme, tregon që ne synojmë të konsiderojmë këto sisteme në thelb si arkitektura softuere, me një rol të qartë dhe të rëndësishëm në proçesin e zhvillimit softuere. Modeli arkitekturor në inxhinierinë softuere bazuar mbi sisteme multiagjente propozuar në këtë hulumtim, kontribuon në integrimin e sistemeve multiagjent në teknikat bazë të përdorura në inxhinierinë softuere.

Mekanizmat për adaptimin e sistemeve multiagjente

Për të mbështetur projektimin arkitekturor të aplikimeve vetë-manaxhuese me sisteme multiagjente, ne kemi zhvilluar një model të avancuar për sisteme multiagjente që shtrihet në fushën e aplikacioneve inteligjente. Ky model integron mjedisin si nivelin e parë të abstraksionit në një model të avancuar për agjentë që përbën mbështetje për zgjidhje të shpejta dhe të lehta duke parashikuar problematika të ndryshme që sjellin agjentët në lidhje me përshtatshmërinë.

Mjedisi si një abstraktion i nivelit të parë në sistemet multiagjente. Modeli ynë në këtë punë kërkimore, vë përpara mjedisin si një bllok ndërtimi të pavarur me gjendjet dhe proçeset që mund të shfrytëzohen gjatë projektimit të sistemit multiagjent. Përgjegjësi të rëndësishme të mjedisit janë:

- Mjedisi mundëson agjentët për të hyrë në burimet e jashtme në sistemin multiagjent, duke ofruar një mënyrë për agjentët për të ndarë informacionin, si dhe për të bashkëvepruar,
- Mjedisi mund të jetë një model ndërmjetës për burime informacioni dhe në mbështetje të ndërveprimit midis agjentëve.
- Mjedisi mund të mbajë dinamikën që ndodh në mënyrë të pavarur nga veprimtaria e agjentëve, siç është psh, kontrolli dhe sigurimi i një kujtese për përcaktimin e burimeve të jashtme të sistemit multiagjent.

Modeli i përgjithshëm për agjentët inteligjentë. Modelet ekzistuese të agjentëve janë kufizuar zakonisht në paraqitjen e mekanizmave për zgjedhjen e veprimit. Probleme të tjera të tilla si perceptimi dhe komunikimi nuk janë trajtuar, apo të integrohen në modelin e përzgjedhjes së veprimit në mënyrë te çfarëdoshme. Në kërkimin tonë, ne kemi zhvilluar një model të avancuar për agjentët i cili integron probleme të ndryshme të agjentëve dhe zhvillon mekanizma të ndryshme për të arritur përshtatjen në kushtet specifike të aplikacionit. Ne kemi zhvilluar një model për perceptim selektiv që mundëson një agjent për të perceptuar drejtpërdrejt në aspekte te veçanta mjedisin, sipas detyrave që realizon. Perceptimi selektiv i lejon një agjenti të përshtasë vëzhgimin e tij me ndryshimin e rrethanave. Kjo e permireson më mirë gjendjen e vetëdijes dhe ndihmon për të mbajtur nën kontroll përpunimin e të dhënave të perceptuara. Ne kemi konceptuar agjentë të pajisura me aftësi të qarta për ndërveprim social. Kjo mundëson agjentët për të shkëmbyer informacion të drejtpërdrejtë me njëri-tjetrin dhe për të ngritur bashkërenditje të veprimtarive individuale. Në veçanti, kemi zhvilluar sjellje përzgjedhëse të bazuar në mekanizmat e veprimit me nocionet e rolit dhe të angazhimit të agjentit. Një rol përfaqëson një pjesë koherente të funksionimit të një agjent në kontekstin e një agjensie. E konsiderojnë një agjensi si një grup të agjentëve që mund të luajnë një ose më shumë role dhe që punojnë së bashku. Një angazhim i vendosur përfaqëson një qëndrim social të një agjenti, dmth një marrëveshje të një agjenti për të luajtur një rol të veçantë në një agjensi. Rolet dhe angazhimet e vendosura mundësojnë agjentët për të krijuar bashkëpunime dhe të zhvillojnë role te ndryshme në bashkëpunime të tilla. Angazhimet në një bashkëpunim, zakonisht varen nga konteksti aktual nëse agjentët e përfshirë janë të vendosur në të. Ky model mundëson agjentët për të përshtatur sjelljet e tyre me rrethanat në ndryshim të vazhdueshëm në mjedis.

Ne kemi zhvilluar një model komunikimi bazuar në protokolle që mundëson agjentët të këmbejnë mesazhe në bazë të protokolleve të komunikimit., domethënë

mesazhe të përcaktuara mirë në sekuenca. Shkëmbimi i mesazheve është i lidhur zakonisht me agjentët eksplorues, ku informacioni i koduar në mesazhe është e lidhur me gjendjen e agjentit (besimi, qëllimi, planet, etj). Kjo perspektivë e supozuar në komunikim nuk është mjaft e përshtatshme me modelin e sistemeve multiagjente. Komunikimi i bazuar në protokolle vë në qendër të vëmendjes komunikimin në lidhje të ngushtë me mesazhet e shkëmbyera. Komunikimi i drejtpërdrejtë mundëson agjentët të shkëmbejnë informacion dhe të krijojnë bashkëpunim të ri, të reflektuar në angazhimet e ndërsjellta të vendosura në një kontekst mjedisi të caktuar.

1.3. Kontributi

Ky kërkim sjell një perspektivë të re në inxhinierinë e softit për sistemet multiagjente. Në mënyrë të veçantë, ky kërkim integron sistemet multiagjente si arkitektura softuere në proçesin e përgjithshëm të inxhinierisë së softit. Konkretisht mund të përmend:

- Sjellja e një modeli të avancuar për sistemet multiagjente në fushën e projektimit të këtyre sistemeve. Në këtë model, mjedisi vjen si abstraksion i nivelit të parë për aplikacionet me sistemet multiagjente. Gjithashtu modeli shtrin më tej konceptin e agjentëve softuere jo vetëm si mekanizma për zgjedhjen e veprimit por, deri në një model të integruar që përfshin detyra të ndryshme si perceptimi, sjellja bashkëpunuese si dhe komunikimi indirekt.
- Zhvillimi i një bashkësie mekanizmash për projektimin arkitekturor duke përfshirë infrastrukturën e mjedisit për perceptim, veprimin, komunikimin, ligjet që përcaktojnë veprimtarinë e agjentëve, si dhe dinamikën në një mjedis, (mjedisi virtual, perceptimi selektiv, mekanizmat e avancuar të zgjedhjes të veprimit me rolet përkatëse, komunikimi i bazuar në protokolle)[21].
- Projektimi i një arkitekture bazë që mbart modelin për sistemet multiagjente nëpërmjet stilit të dekompozimit të sistemit [21]. Arkitektura do të integrojë një bashkësi arkitekturash më të mira të marra nga aplikacione të ndryshme të cilat ne i kemi studiuar dhe ndërtuar paraprakisht. Kjo arkitekturë siguron për zhvilluesit një model bazë në aplikacione të ndryshme që kanë kërkesë vetë-manaxhimin. Gjithashtu ajo ofron një mjet për studimin dhe kërkimin në perspektiva të avancuara sistemet multiagjente. Për të paraqitur këtë idet, kemi zhvilluar një kornizë pune, e cila implementon arkitekturën si dhe e kemi përdorur atë duke zhvilluar aplikacione të vetëmanaxhueshme.
- Zhvillimi i sistemeve multiagjente në aplikacione industriale komplekse që përdorin vetëmanaxhimin për të arritur në mënyrë automatike

objektiva të caktuara. Në mënyrë konkrete, kemi aplikuar model multiagjent për të rritur funksionalitetet e sistemit duke kënaqur kërkesat më të rëndësishme të cilësisë së sistemit. Zhvillimi i këtij aplikacioni konsiston në këto aktivitete kryesore:

- Analiza e kërkesave funksionale të sistemit
- Vendosja e prioriteteve në kërkesat e cilësisë.
- Zhvillimi inkremental i arkitekturës softuere
- Vlerësimi i arkitekturës softuere
- Implementimi inkremental i aplikacionit
- Testimi i kërkesave kryesore të sistemit.

- Realizimi i një vlerësimi të disiplinuar të arkitekturës softuere në aplikimet industriale. Për vlerësimin e kërkesave dhe të krejt sistemit të aplikacionit ne kemi aplikuar metodën ATAM.[1]

1.4. Përmbajtja e kapitujve

Pas hyrjes , materiali i këtij punimi përmban 8 kapituj dhe konkluzionet.

Në kapitullin e dytë, ne jemi përqëndruar në idenë e zhvillimin arkitekturor të një modeli bazë duke e vlerësuar këtë si një fazë mjaft të rëndësishme të ciklit të jetës në inxhinierinë e sistemeve softuere. Kemi trajtuar analizën dhe identifikimin e kërkesave funksionale dhe si dhe atributet e cilësisë si elementë të domosdoshëm në projektimin arkitekturor të një sistemi softure duke vëne theksin edhe në dokumentimin dhe vlerësimin kërkesave kryesore funksionale si një moment mjaft i rëndësishëm për të marrë vendime mbi modelin referencë që do të përdoret më tej në zhvillim

Në kapitullin e tretë kemi analizuar elementë të arkitekturave me agjente duke diskutuar mbi sistemet me një dhe shumë agjentë, si dhe duke sjellë në kujtesë arkitekturat më të njohura të agjentëve reaktivë. Jemi finalizuar në këtë kapitull me modelet hibride të agjentëve që mbështesin arkitekturat e agjentëve robotikë.

Në kapitullin e katër është sjellë koncepti i sistemeve stigmergjike për të karakterizuar sjelljen reaktive kolektive të sistemeve multiagjente. Kapitulli përmban një analizë të konceptit të mjedisit si element arkitekturor dhe rolit që ai luan me nivelet e veta të mbështetjes në projektimin arkitekturor. Kemi trajtuar mjedisin në

1 Kjo metode eshte zhvilluar nga instituti amerikan I inxhinierise software i universitetit Carnegie Mellon. Eshte nje nga modelet me te qenesishme te vleresimit te arkitekturave software tashme te vlefshme [23]

mënyrë të strukturuar, në kontekstin e sistemeve multiagjente me burimet e inkorporuara në të. Trajtohen në këtë kapitull mekanizmat e variancës si mekanizma të avancuar që realizojnë përshtatjen e modelit referencë në sistemet me agjentë robotikë. Për të shkuar më thellë në këtë koncept, kemi analizuar gjendjet e agjentit robotik në perceptimin përzgjedhës të mjedisit dhe veçoritë që bëjnë dallimin në aplikacione të ndryshme.

Kapitulli i peste mbulon teorikisht çështjen e projektimit arkitekturor me module të një sistemi bazuar në agjentë. Janë vlerësuar arritjet në plan historik arkitekturat ekzistuese. Kemi bërë një analizë të njohjes mbi karakteristikat dhe kërkesat e fushës së aplikacionit për të kaluar konkretisht në proçesin e zhvillimit të arkitekturës mbi një model integrues të sistemeve multiagjente. Pikërisht në këtë kapitull i referohemi dy elementëve kryesore duke analizuar modelimin e mjedisit dhe atë të agjentit

Kapitulli i gjashtë është një analizë e detajuar punës për ndërtimin e modelit referencë duke konceptuar dekompozimin në tre nivele. Strukturat e elementëve, lidhjet midis komponentëve përbërës të tyre, ndërfaqet si dhe burimet e të dhënave janë objekt i projktimit të modeli të sjellë duke mos lënë pasdore parashikimin e mekanizave të variancës. Janë përshkruar këndvështrime të ndryshme të këtij modeli dhe janë specifikuar mekanizma të ndryshëm në kushtet specifike për aplikimin e saj në ndërtimin e arkitekturave konkrete softuere.

Kapitulli i shtatë do të sjellë një aplikacion bazuar mbi agjentë në mjedisin e një sistemi të manaxhimit të informacionit. Duke nisur nga kërkesat e sistemit, do të diskutohet projektimi arkitekturor i aplikacionit, gjithashtu do të shpjegohet se si arkitektura softuere e këtij aplikacioni lidhet me modelin referencë dhe sesi ai ka kontribuar në zhvillimin e kësaj arkitekture. Do të shihet me vëmendje vlerësimi i arkitekturës softuere, si dhe do të diskutohen rezultatet e testeve të mbledhur nga implementimi i sistemit.

Kapitulli i tetë përqëndrohet në procesin e evolimit të sistemit me shërbime të tjera autonome duke synuar të nxjerrë në pah thjeshtësinë dhe komoditetin që sjellin arkitekturat e bazuara në agjentë në procesin e zhvillimit inkremental në sistemet softuere. Nënsistemi i yhvilluar shihet si një arkitekturë autonome bazuar mbi modelin referencë për arkitekturat robotike. Është vënë theksi në lehtësinë e shtimit të komponentlve në këto arkitektuara me elementë të lidhur dobët ku shtimi apo heqja e tyre nuk prek funksionalitetin e krejt sistemit. Kontrolli i funksionimit dhe realizimi i koordinimit vjen në një model hibrid që shfrytëzon avantazhet e arkitekturave me shtresa dhe atyre të integruara.

Kapitulli i nëntë përforcon idenë e përgjithësimit të modelit të zhvilluar nëpërmjet një aplikacioni tjetër me të njëjtën arkitekturë. Komponentët janë të njëjtë përveç atyre lidhës që përfshihen si elemetë të mekanizmave të variancës. Sistemi

ndryshon funksionalitetin në modalitet zbulues mjedisi. Agjenti përdor sjellje reaktive për të thelluar njohjen e botës ku vepron. Punimi realizohet me sukses edhe në mjedis real fizik

Në fund, kapitulli i dhjetë do të përmbajë konkluzionet e punimit. Do të përmblidhen kontributet e këtij punimi duke raportuar çfarë është arritur në planin teorik dhe eksperimental mbi aplikacionet bazuar ne sisteme multiagjente në praktikën industriale. Duke përfunduar , do të nënvizohet fakti për rrugë të tjera për kërkim në të ardhmen.

KAPITULLI 2

Projektimi Arkitekturor të Sistemit Softuere

Zhvillimi arkitekturor i modelit bazë

Që prej fillimit të viteve 90 të shekullit të kaluar, arkitekturat softuere kanë qenë subjekt i rritjes së interesit në fushën e kërkimit dhe të aplikimeve për rinxhinierinë softuere. Në [24] jepen tre arsyet më të rëndësishme pse arkitektura është e rëndësishme për sistemet softuere.

- Është një mjet komunikimi midis stakeholderave. Arkitektura softuere siguron një bazë për kuptimin e dyanshëm dhe pranimin për një sistem softuere të caktuar
- Është paraqitje e vendimeve të projektimit që ka influencën më të madhe në cilësitë e sistemit.
- Është e ripërdorshme dhe një abstraksion i transferueshëm i sistemit. Arkitektura softuere jep një model të mbikyrshëm se si një sistem është ndërtuar dhe se si ai punon. Ky model mund të transferohet tek sistemet e tjera me kërkesa të ngjashme dhe mund të promovojë një përdorim të gjerë të modelit arkitekturor.

Në këtë kapitull, do të fokusohem në zhvillimin arkitekturor duke u përqëndruar në atë të sistemeve multiagjente. Do të trajtojmë projektimin arkitekturor në ciklin e jetës të zhvillimit softuere. Më tej, do të diskutojmë shkurtimisht kërkesat e sistemit për të mundësuar projektimin arkitekturor. Kërkesat e ndryshme të sistemit grumbullohen në bazë të vendimeve arkitekturore bazuar mbi modelet arktitekturore siç janë stilet arkitekturore apo arkitekturat bazë referencë. Ne do të shpjegojmë se si një arkitekturë softuere dokumentohet dhe mbi të gjitha, se si mund të vlerësohet ajo. Pastaj shkurtimisht do të shpjegojmë se si arkitektura softuere shërben si referencë për një projektim të detajuar dhe implementim të shpejtë të sistemit.

2.1.1. Projektimi arkitekturor në ciklin e jetës së zhvillimit softuere

Për të kuptuar idenë e modelit tonë për zhvillimin softuere bazuar në sisteme multiagjente, le të vendosim projektimin arkitektruror në ciklin e jetës së zhvillimit softuere. Po përdorim ciklin e jetës sipas modelit evolucionar. [25] Në këtë model, projektimi arkitekturor vendoset në mes të aktiviteteve të zhvillimit. Idea kryesore është mbështetja e zhvillimit inkremental dhe vendosja e një *feedback*-u më të afërt

nga stakeholderat. Cikli i jetës konsiston në dy faza kryesore: zhvillimi i sistemit bazë dhe dorëzimi i produktit softuere final.

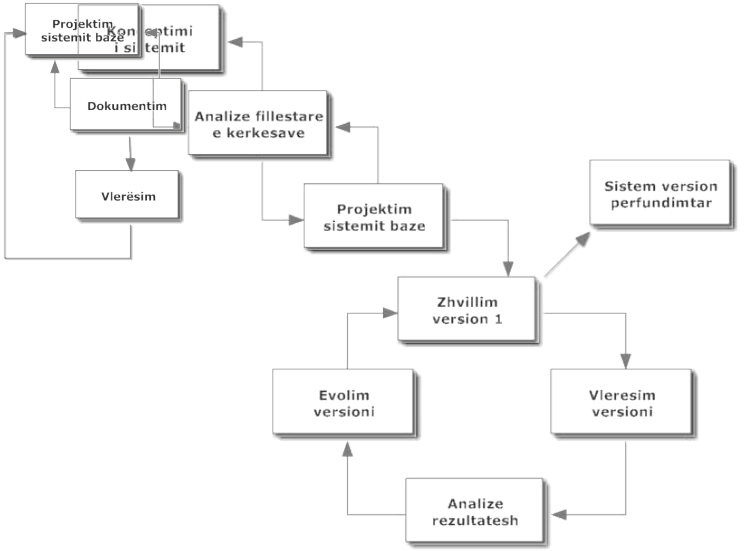

Figura 2. Projektimi arkitekturor në ciklin e jetës të zhvillimit softuere

Në fazën e parë ne zhvillojmë sistemin bazë duke përfshirë katër veprimtari kryesore:

- Përcaktimi i modelit të fushes
- Përformimi i analizës së kërkesave të sistemit
- Projektimi i arkitekturës softuere
- Zhvillimi i sistemit bazë

Analiza e kërkesave përfshin formulimin e kërkesave funksionale të sistemit sikurse edhe përzgjedhjen dhe vendosjen e prioriteteve të kërkesave të atributeve të cilësisë. Projektimi i arkitekturës softuere përfshin projektimin dhe dokumentimin e arkitekturës softuere si dhe vlerësimin e arkitekturës. Zhvillimi i sistemit bazë përfshin projektimin e detajuar, implementimin dhe testimin e sistemit. Proçesi i inxhinierisë së softit është një proçes iteraktiv ku sistemi bazë zhvillohet në mënyrë inkrementale duke përshkuar disa faza të proçesit të zhvillimit. Në figurën 2.1 paraqitet se si projektimi arkitekturor ndërthurret me analizën e kërkesave në njërën anë sikurse me zhvillimin e sistemit bazë në anën tjetër. Produkti i fazës së parë është modelimi i fushës, që do të thotë një listë e kërkesave të sistemit, një arkitekturë softuere dhe implementimi i sistemit bazë.

Në fazën e dytë, zhvillohen versione të sistemit në nivel nënsistemesh që plotësojnë të tjera kërkesa përpara se sistemi të dorëzohet në versionin final. Megjithatë duhet të nënvizohet fakti se, nuk ka një cikël kërcimi nga faza e dytë në të parën megjithëse në praktikë kjo mund të kërkojë disa rregullime përsa i takon këtij fakti.

2.1.2. Kërkesat funksionale dhe atributet e cilësisë së sistemeve softuere.

Projektimi arkitekturor mund të fillojë vetëm kur njhen kërkesat më kryesore të sistemit. Ky bllok kërkesash zakonisht quhet driver arkitekturor dhe përfshin kërkesat funksionale dhe ato të atributeve të cilësisë. Funksionaliteti është aftësia e sistemit për të kryer detyra për të cilat ai është konceptuar. Për të kryer një detyrë, elementët softuere duhet të jënë përcaktuar mirë duke patur përgjegjesinë plotësisht të saktësuar për të koordinuar me të tjerë elemente të sistemit për të ofruar funksionalitetin e kërkuar. Kërkesat funksionale të sistemit shprehen në mënyrë tipike si raste studimi.[26] Një rast studimi liston hapat e nevojshëm për të arritur një qëllim funksional për një aktor që përdor sistemin. Në punim, ne përdorim skenare që përshkruajnë ndërveprime midis pjesëve të sistemit. Një shembull është një skenar që përshkruan kërkesat e parashikimit të ndalimit të përplasjes së agjentëve robotike në kryqëzimë të patheve të lëvizjes së tyre. Funksionaliteti nuk varet nga struktura e sistemit. Në thelb, nëse funksionaliteti do të ishte e vetmja kërkesë, sistemi do të ekzistonte si një modul monolitik, pa një strukturë të brendshme.[27] Cilësia është metriku me të cilën në një sistem identifikohen kërkesat jofunksionale sipas kontekstit të funksionalitetit të kërkuar. Atributet e cilësisë janë veçori jofunksionale siç është performanca, përdorueshmëria, dhe ndryshueshmëria e tij. Arritja e atributeve të cilësisë mund të konsiderohet nëpërmjet proçesit të zhvillimit softuere. Megjithatë arkitektura softuere është kritike në realizimin e atributeve të cilësisë. Ajo siguron bazën për arritjen e cilësisë. Për të shprehur kërkesat e cilësisë, ne do të përdorim skenarët e atributeve të cilësisë të parë në [28]. Një skenar i tillë konsiston në tre pjesë kryesore:

1. **Stimulus**: një kusht i brendshëm ose i jashtëm, i gjeneruar që prek sistemin ose një pjesë të tij dhe ka nevojë të merret në konsideratë kur ai arrin tek sistemi(për shembull, kur një përdorues thërret një funksion, ose një mirëmbajtës bën një ndryshim, një komponent dështon, etj).
2. **Mjedisi** : kushtet nën të cilat stimuli ndodh.
3. **Përgjigje**: aktiviteti që kryet nën arkitekturën e sistemit kur arrin një stimul. Përgjigjja duhet të jetë e matshme në mënyrë që kërkesat të mund të testohen (për shembull ndryshimi kërkon një person në punë, gabimi shfaqet brenda 5 sekondave dhe sistemi kalon në modalitet shpëtimi, etj)

Skenarët janë mjaft të përdorshëm për të kuptuar cilësitë e sistemit, formulimi i skenarëve ndihmon stakeholderat të shprehin preferencat e tyre rreth sistemit në një mënyrë më të qartë. Gjithashtu, ato ndihmojnë edhe arkitektët të ndërmarrin vendime të drejtpërdrejta dhe janë një mjet kryesor në analizën dhe vlerësimin e sistemit. Në mënyrë ideale, skenarët mblidhen dhe renditen përpara projektimit arkitekturor. Megjithatë pa patur një numër ndërveprimesh nuk mund të mblidhen dhe të renditen kërkesat e sistemit. Në këtë punim ne kemi përdorur '*utility trees*' të diskutuar në [28] për të renditur skenarët e atributeve të cilësisë.

Projektimi arkitekturor

Të projektosh një arkitekturë softuere do të thotë të kalosh nga kërkesat e sistemit në vendime arkitekturore. Ky hap i rëndësishëm në inxhinierinë e softit mbështetet në njohjen dhe eksperiencat e projektuesve softuere. Në këtë pjesë do të shpjegojmë projektimin arkitekturor me një model bazë arkitekture. Gjithashtu do të shpjegojmë se si e dokumentojmë këtë arkitekturë, si dhe mekanizmat e ndryshëm për të aplikuar këtë model në projektimin e një arkitekture konkrete të një aplikacioni softuere.

2.1.3. Modelet arkitekturore dhe modeli bazë

Arritja e atributeve të cilësisë së sistemit bazohet në vendimet e projektimit. Të tilla vendime quhen taktika (*tactics*). Një '*tactic*' është një model arkitekturor i përgjithshëm, i cili përdoret për te arritur një cilësi të veçantë[29], për shembull '*rollback*' është një '*tactic*' për të manaxhuar dështimin me qëllim rritjen e disponueshmërisë së sistemit. Ose, një '*concurrency*' është nje '*tactic*' për të manaxhuar aksesin në burime, me qëllim përmirësimin e performancës së sistemit. Për të realizuar një ose më shumë '*tactics*', një arkitekt zgjedh një stil arkitekturor të vetin.[30]

Një stil arkitekturor përcaktohet edhe në [31] si : ***"përshkrim të elementëve arkitekturore dhe tipeve të marrdhënieve së bashku me një bashkësi kufizimesh në atë se si do të përdoren ata"*** Një stil arkitekturor është një model arkitekturor që shfaq atribute cilësie të veçanta. Shembuj të stileve të përgjithshme arkitekturore janë shtresat, '*pipe-and-filter*', '*blackboard*'. Arkitekturat referenciale [32] janë një hap përpara në ripërdorimin e praktikave më të mira në projektimin arkitekturor me një arkitekturë bazë. Një arkitekturë integron një bashkësi elementësh arkitekturore që kanë dhënë prova të vlerave të tyre për një familje të veçantë aplikacionesh. Kjo familje aplikacionesh karakterizohet nga funksionalite dhe kërkesa për atribute cilësie specifike. E rëndësishme për një arkitekturë reference është mekanizmi i variancës për të aplikuar këtë lloj arkitekture në një aplikacion specifik. Koncepti i arkitekturës referenciale është i ngjashëm me atë të një frameworku në teknologjinë e orientuar nga objekti apo si një arkitekture prodhimi seri të produkteve softuere.[33]

Një arkitekturë referenciale mund të përshkruhet nga paraqitje të ndryshme të cilat i përkasin aspekteve të ndryshme të arkitekturës softuere. Në [34] janë shpjeguar katër paraqitje të arkitektures softuere. Secila nga këto paraqitje jep një aspekt specifik arkitekturor i cili është mjaf i përdorshëm nga stakeholdera të ndryshem. Paraqitja logjike jep një përshkrim të shërbimeve që sistemi duhet të ofrojë tek përdoruesit e thjeshtë. Paraqitja e proçesit fikson aspekte të konkurrencës dhe sinkronizimit për projektimin. Paraqitja fizike përshkruan se si softuere integrohet në harduere duke reflektuar aspekte të shpërndarjes. Paraqitja e zhvillimit përshkruan organizimin e softuere dhe lidh module të softuere me zhvilluesin përkates. Një paraqitje finale paraqet se si elementët e katër paraqitjeve punojnë së bashku.[35]

Një perspektivë arkitekturore përkufizohet si: “një koleksion veprimtarish, taktikash dhe udhërrefyesish që përdoren për të siguruar që një sistem po ekzekuton një bashkësi të veçantë të veçorive të cilësisë që merren në konsideratë nëpërmjet një numri pamjesh arkitekturore të sistemit”. Ajo i drejton arkitektët në proçesin e analizës dhe modifikimit të arkitekturës softuere për të qenë të sigurtë që ajo do të shfaqë veçoritë përkatëse të cilësisë. Perspektiva arkitekturore përfshin: aksesueshmërinë, disponibilitetin, evolimin, ndërkombëtarizimin, performancën dhe shkallëzueshmërinë, sigurinë dhe përdorureshmërinë.

Në [36] përgjithësohet koncepti i paraqitjeve duke e lidhur me stilet arkitekturore. Futet koncepti i tipeve të paraqitjeve që përcaktojnë tipet e elementëve dhe tipet e marrdhënieve të përdorura për të përshkruar një arkitekturë sistemi nga një perspektivë e veçantë. Dallohen tre tipe paraqitjesh:

1. *Paraqitje në module:* paraqitjet në tipin e paraqitjes modul dokumentojnë një njësi kryesore sistemi për implementim.
2. *Paraqitje me komponentë dhe konektorë:* paraqitjet ketu dokumentojnë një njësi sistemi për ekzekutim
3. *Paraqitje të vendosjes:* paraqitjet këtu dokumentojnë marrdhëniet midis softuere të sistemit dhe zhvillimit të tij me mjedisin e ekzekutimit.

Një stil arkitekturor është një specializim i tipeve të paraqitjeve dhe reflekton një model arkitekturor që shfaq atribute specifike të cilësisë, të pavarura nga lloji i sistemit. Per shembull “stili i shtresëzuar” është një specializim i tipit të paraqitjes me shtresa. Ai përshkruan një pjesë të sistemit si një bashkësi shtresash ku çdo shtresë lejohet të përdorë shërbimet e shtresës më të poshtme. “Stili i proçeseve të komunikimit" është një tjetër shembull i paraqitjes me komponentë dhe konektorë. Ky lloj stili përshkruan një njësi konkurente siç janë proçeset dhe threadet, si dhe lidhjen midis njësive siç janë sinkronizimi dhe kontrolli. Si përfundim, një paraqitje është *një inst*ancë e një stili arkitekturor që përcaktohet me elemente specifike dhe marrdhënie në një sistem të dhënë specifik.

Në këtë punim ne do të përdorim këto koncepte për modelin tonë.

Mekanizmat e variancës. Më tej në modelin bazë referues futet edhe koncepti i mekanizmave të variancës për të shpjeguar se si ato përdoren për të ndërtuar një arkitekturë softuere për një aplikacion konkret. Një model i tillë referues mund të sigurojë mekanizma variance të ndryshme. Për shembull:

- *Largimi i një elementi:* element të veçantë të modelit bazë referues nuk janë të përbashkët për çdo aplikacion. Në këtë rast largohet elementi i paaplikueshëm.
- *Konkretizimi i një elementi abstrakt*: disa elemente janë përcaktuar si abstrakt dhe mund të konkretizohen sipas kërkesave të aplikacionit.
- *Përcaktimi i sjelljes*: sjellja e disa elementëve është përcaktuar në mënyrë abstrakte kështu që arkitekti duhet të konkretizojë sjelljen sipas kërkesave të aplikacionit.

Disa prej mekanizmave të variancës janë vështirë të aplikohen. Disa mund të kërkojnë mbështetje shtesë. Megjithatë dokumentimi i arkitekturës bazë referuese duhet të përshkruajë cilat mekanizma janë të aplikueshëm dhe se si ato mund të ekzekutohen.

2.1.4. Projektimi arkitekturor mbi modelin bazë referencial

Projektimi arkitekturor kërkon një model sistemi për të zhvilluar një arkitekturë softuere që arrin funksionalitetin e kërkuar dhe kënaq kërkesat e cilësisë. Në kërkimin tonë, ne kemi përdorur teknika nga metoda Attribute Driven Design për të projektuar arkitekturën për një sistem manaxhimi të informacionit i cili ka një arkitekturë bazë së cilës i referohemi. ADD është një metodë dekompozimi e cila bazohet në mënyrën se si të arrijmë qëllimet e cilësisë nëpërmjet modeleve tashmë të testuara dhe të provuara.

Zakonisht, çdo projektues nis nga sistemi si të ishte një kuti e zezë dhe më pas vendos dhe qartëson elementët arkitekturore, deri në çastin kur këta elementë janë të mjaftueshëm për të filluar projektimin e detajuar dhe implementimin. Në këtë pikë, arkitektura është vetëm një plan përshkrues për ndërtimin e sistemit që sjell idenë e vlerësimit nëse kënaq kërkesat funksionale dhe të cilësisë së sistemit (37).

Në secilën fazë të dekompozimit, projektuesi zgjedh një element arkitekturor të qartë dhe përcakton nëse realizohen dhe kombinohen me suksses objektivat e kërkesave funksionale në lidhje të ngushtë me atributet e cilësisë së sistemit.

Më tej, projektuesi zgjedh një model arkitekturor të përshtatshëm, dekompozon elementin arkitekturor dhe bashkangjit funksionalitetin tek nënelementët e sistemit. Nje model i tillë bazë do të shërbente si një guidë për të drejtuar

projektuesin në proçesin e mëtejshëm të proçesit të dekompozimit. Ky model siguron një bashkësi të integruar elementësh, të cilët projektohen për të dhënë zgjidhjet arkitekturore të duhura. Megjithatë është e rëndësishme të theksohet që modeli bazë asnjëherë nuk do të siguronte një guide ose tutorial për një projekt arkitekturor duke shmangur shpirtin krijues të projektuesit. Duke përdorur një model bazë, nuk zgjidhen krejtësisht problemet e veshtira që dalin në projektim, përfshirë këtu edhe përzgjedhjen e modeleve për kërkesa specifike të sistemit. Një model bazë arkitekturor ofron në fund të fundit një bashkësi elementesh arkitekturore të ripërdorshëm për të ndërtuar arkitektura softuere për aplikacione konkrete.

Si përfundim, për projektimin arkitekturor tëi një sistemi softuere me një model bazë, proçesi ADD mund të përdoret për të përcaktuar në mënyrë interaktive arkitekturën softuere. Ky model bazë mund të shërbejë si një guide e mirë në proçesin e dekompozimit. Më tej mund të themi se modelet arkitekturore të përbashkëta duhet të aplikohen për të përcaktuar dhe shtrirë elementët arkitekturore kur është e nevojshme, sipas kërkesave të sistemit.

Dokumentimi i arkitekturës softuere

Për të qenë sa më efektive, një arkitekturë softuere duhet të jetë e mirëorganizuar dhe nuk duhet të ketë situata të paqarta në komunikim me grupe të ndryshëm të personave të përfshirë me të. Kështu, një dokumentim i mirë i arkitekturës softuere është mjaft i rëndësishëm. Dokumentimi duhet të jetë i përgjithshëm aq sa të jetë i kuptueshëm dhe konkret në mënyrë të mjaftueshme për të udhëhequr projektuesin për të ndërtuar sistemin. Në (38) jepen këto përdorime të dokumentimit të arkitekturës:

1. *Arkitektura shërben si mjet edukimi*. Arkitektura softuere është një mjet mjaft i përdorshëm për të bërë një hyrje të sistemit tek persona të rinj si anëtarë të skuadrës së zhvillimit, analistë të jashtëm, etj.

2. *Komunikimi midis personave të përfshirë në sistem*. Arkitektura softuere përfaqson një abstraksion të përbashkët që shërben si një mjet navigimi në proçesin e komunikimit midis këtyre personave. Ajo formon një bazë për organizimin e projektit duke vendosur kufizime në projektim dhe në implementimin e sistemit. Ajo është një pikë nisjeje për veprimtaritë e mirëmbajtjes.

Dokumentimi i arkitekturës softuere konsiston në këndvështrime kryesore të kompletuara me informacion shtesë që aplikohen në këto këndveshtrime në mënyra të ndryshme. Paraqitjet mund të përshkruhen me një gjuhë përshkruese arkitekturore ADL (39). ADL mund të jetë formale ose gjysmë formale, gjuhë grafike ose kombinim i të dyjave. SEI (Instituti i inxhinierisë softuere) ka një faqe web që siguron linke për mjaft ADL. ADL-te janë të ndryshme , disa janë konceptuar vetëm

për modelimin, disa të tjera janë të lidhura ngushtë me një gjuhë programimi. Kjo është e rëndësishme të mbahet parasysh si veçori, sepse garanton që sistemi i implementuar është në përputhje të plotë me arkitekturën e përshkruar.

Mjaft projektues preferojnë UML si ADL. Për fat të keq, UML nuk ofron mbështetje direkte për shumë elemente arkitekturore siç janë modulet, semantikën e shtresave dhe ndërfaqeve. Qëkurse ADL siguron të gjitha lehtësirat për të kompletuar një dokument për paraqitje të ndryshme të tij, ne do të punojmë me një përshkrim hibrid që përfshin edhe konstrukte UML ku është e mundur. Secila diagramë ndërtohet me një çelës që shpjegon domethënien e simboleve të përdorura. Akoma më tej, ne do të përdorim shënim formal të thjeshtë në bashkësinë e teorisë për të specifikuar elemente të ndryshëm arkitekturore si dhe marrdhëniet midis tyre.

Se cilat paraqitje do të dokumentohen, kjo varet nga qëllimet e dokumentimit. Një arkitekturë softuere e konceptuar për planifikimin e projektit fillestar, përmban një tjetër bashkësi të paraqitjeve arkitekturore si një arkitekturë që specifikon njësitë për skuadrën e zhvillimit të sistemit softuere. Disa paraqitje të sistemit softuere japin informacion për elemente të ndryshme si dhe marrdhëniet midis tyre duke ekspozuar atribute të ndryshme të cilësisë.

Informacioni shtesë në dokumentimin e arkitekturës softuere mund të përfshijë informacion mbi historikun, template të paraqitjes, etj.

Vlerësimi i Arkitekturës Softuere

Një arkitekturë softuere është baza e sistemit softuere, ajo përfaqson bashkësinë më të hershme të vendimeve mbi projektimin (40). Këto vendime të hershme janë më të vështirat për t'u marrë për të qenë korrekt, më të vështirat për t'u ndryshuar më vonë dhe ka më shumë efekte me ndikim të gjerë. Arkitektura softuere jo vetëm që strukturon softuerin e sistemit, por dhe projektin në termat të strukturimit të skuadrës zhvilluese dhe të skedulimit të punës. Për shkak të këtij impakti të gjerë, është mjaft i rëndësishëm verifikimi i arkitekturës sa më shpejt që të jetë e mundur. Modifikimet në fazat e para të projektimit janë me kosto më të ulët dhe më lehtë për t'u realizuar. Mungesa e vlerësimit do të kërkonte ndryshime me kosto të lartë ose çon deri në rezultate të sistemit me cilësi të ulët.

Vlerësimi arkitekturor do të thotë ekzaminim i arkitekturës softuere për të përcaktuar nëse ajo kënaq kërkesat e sistemit, në mënyrë të veçantë kërkesat për atributet e cilësisë së sistemit. Një vlerësim i tillë fokusohet në atributet më të rëndësishëm që do të thotë atribute që janë më të rëndësishme për stakeholderat e sistemit dhe të tilla që kanë impaktin më të gjerë të arkitekturës softuere. Vlerësimi arkitekturor në mënyrë tipike është efektiv atëhere kur arkitektura është specifikuar përpara implementimit. Kjo lejon që të shtohen pjesë të humbura, të korrigjohen

vendime apo të detajohen pjesë specifike të arkitekturës përpara se kostot për këto ndryshime të bëhen mjaft të larta.

Në kërkimin tonë ne kemi përdorur Architecture Tradeoff Analysis Method për vlerësimin e arkitekturës së sistemit softuere. ATAM është zhvilluar në Institutin e Inxhinierisë Softuere dhe përbën një nga modelet më të provuara dhe të testuara për vlerësimin e vlefshmërisë së arkitekturës softuere kohët e fundit.

Qëllimi kryesor i ATAM është të përcaktojë risqet në lidhje me kërkesat për atributet e cilësisë. ATAM është një metodë vlerësimi që:

1. Përdoret nga stakeholderat për të përcaktuar kërkesat për atributet më të rëndësishmetë cilësisë Përdoret nga projektuesi për t'u fokusuar në pjesët më të rëndësishme të arkitekturës
2. Përdor modele arkitekturore për të përcaktuar se cilat janë problemet potencialisht më të mëdha në fazën e projektimit.

Ekzistojnë dy grupe të mëdha njerëzish të përfshirë në ATAM:

- Skuadra e vlerësimit, e cila udhëheq vlerësimin dhe realizon analizën
- Stakeholderat, të cilët kanë një interes të veçantë në arkitekturën softuere që i nënshtrohet vlerësimit, të tillë si manaxheri i projektit, zhvilluesit, klientët ose përdoruesit, etj.

Një proçes vlerësimi duke përdorur ATAM prodhon rezultatet e mëposhtme:

- Një listë të kërkesave për atribute të cilësisë në trajtën e një peme të përdorimit të atributeve të cilësisë me më prioritet.
- Një hartë të modeleve arkitekturore sipas atributeve të cilësisë. Analiza e arkitektures paraqet se si arrin ose deshton ne arritjen e kerkesave per atribute te cilesise.
- Rreziqet dhe jorreziqet. Rreziqet jane vendime arkitekturore problematike, ndersa jorreziqet jane vendime arkitekturore te mira.
- Pikat e ndjeshme dhe tyradeoff. Nje pike e ndjeshme eshte nje vendim arkitekturor i cili eshte kritik per arritjen e nje atributi cilesie te veçantë, ndersa nje pike tradeoff eshte nje vendim arkitekturor qe prek me shume se nje atribut, eshte nje pike e ndjeshme per me teper se nje atribut.

Nje dokument mjaft i rendesishem ne metoden ATAM eshte pema e perdorimit te atributeve te cilesise ose shkurt '*utility tree*'.

Ky dokument është një listë e qëllimeve për atributet e cilësisë me prioritet, të formuluara si skenare. Një metodë '***utility tree***' shpreh se cilat janë qëllimet më të rëndësishme e cilësisë për sistemin. Një shembull i një '***utility tree***' paraqitet në

figurën 2.2. Nyjet kryesore të pemës paraqesin cilësinë në përgjithësi të sistemit. Atributet e cilësisë në nivel të lartë formojnë nivelin e dytë të pemës. Secili atribut cilësie detajohet më tej në nivelin e tretë. Në fund, gjethet e nyjeve të pemës janë skenarët e atributeve të cilësisë. Secili skenar ka të shënuar prioritetin në mënyrë relative me skenarët e tjerë. L do të jetë i larte, M i mesëm dhe U i ulët. Vendosja e prioritetit bëhet në dy permasa. Shënimi i parë i referohet rëndësisë së skenarit për suksesin e sistemit. Ndërsa skenari i dytë i referohet vështirësisë për të arritur skenarin. Pema shpreh se cilat cilësi të sistemit janë të rëndësishme dhe kjo shërben si një guidë për vlerësuesit për të zgjedhur modelet arkitekturore që kënaqin skenarët më të rëndësishëm të sistemit. Është e qartë që skenarët me prioritete (L,L) dhe (L,M) janë kandidatët e parë për analizë gjatë vlerësimit të arkitektures.

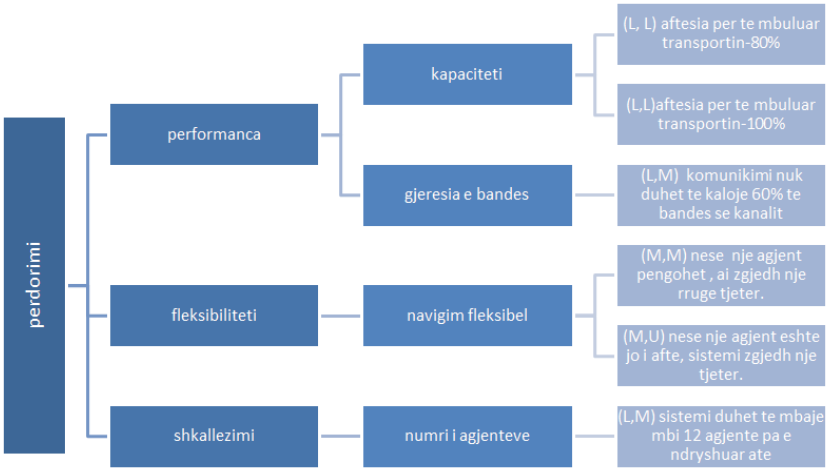

Figura **2**. **Shembull i metodës 'Utility Tree'**

Vlerësimi i arkitekturës softuere me ATAM konsiston në tre faza kryesore:

1. Prezantimet. Faza e parë konsiston në tre hapa kryesore: drejtuesi i vlerësimit fillon duke dhënë një shikim të përgjithshëm të metodës së vlerësimit. Më pas, manaxheri i projektit përshkruan qëllimet e biznesit që duhen të arrihen nga projekti; në fund projektuesi jep një pamje të përgjithshme të arkitekturës softuere.

2. Hulumtimi dhe analiza. Faza e dytë gjithashtu konsiston në tre hapa. Në fillim, projektuesi identifikon modelet arkitekturore të aplikueshme në arkitekturën softuere. Në hapin e dytë gjenerohet një dokument '*utility tree*' e atributeve të cilësisë. Atributet e cilësisë për sistemin shfaqen nga stakeholderat dhe specifikohen si skenarë. Kjo listë bëhet në bazë të prioriteteve të përcaktuara. Në fund, modelet arkitekturore me skenarë

me prioritet të lartë analizohen duke prodhuar një listë të rreziqeve dhe pikave të ndjeshme. Analiza mund të zbulojë modele arkitekturore të tjera shtesë.

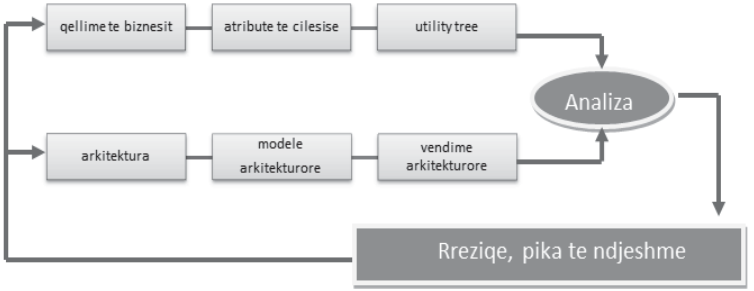

Figura **2**. **Diagrama konceptuale e metodës së vlerësimit**

3. Raportimi i rezultateve. Në fazën finale, informacioni i mbledhur gjatë ATAM it prezantohet tek stakeholderat, të mbledhur së bashku në një aktivitet.

Një diagramë e metodës ATAM paraqitet në figurën 2.3. Diagrama paraqet se si ATAM shfaq rreziqet arkitekturore me të cilat përballet arkitektura softuere dhe arritjet e mundshme të qëllimeve të biznesit si organizatë nëpërmjet një procesi analize.kompleks.

Përfundime

Në këtë kapitull ne diskutuam për zhvillimin arkitekturor të modelit bazë softuere, se si ka evoluar konceptimi i arkitekturës si një mjet komunikimi midis stakeholderave apo si një paraqitje e vendimeve të projektimit që ka ndikimin më të madh në cilësitë e sistemit duke arritur në përfundimin se:

Arkitektura softuere jep një model të mbikyrshëm se si një sistem është ndërtuar dhe se si ai punon. Ky model mund të transferohet tek sistemet e tjera me kërkesa të ngjashme dhe mund të nxisë një përdorim të gjerë të modelit arkitekturor.

Nje arkitekture softuere përcakton pikat më të vështira në projektim dhe implementim, përshkruan se si implementimi duhet të ndahet në elemente dhe, se si këto elemente duhet të ndërveprojnë me njeri tjetrin për të arritur objektivat e sistemit. Një arkitekturë nuk përcakton një implementim softi, domethënë se disa vendime projektimi mund të mbeten të hapura për t'u vendosur nga projektuesit dhe zhvilluesit. Shembuj mund të përmend algoritma specifike, struktura të dhënash të brendshme të moduleve, etj.

Në këtë kapitull, u fokusuam në rëndësinë e zhvillimit të një modeli bazë arkitekturor duke u përqëndruar në atë të sistemeve multiagjente. Trajtuam projektimin arkitekturor në kuadrin e ciklit të jetës të zhvillimit softuere. U diskutuan shkurtimisht kërkesat e sistemit për të mundësuar projektimin arkitekturor. Kërkesat e ndryshme të sistemit grumbullohen në bazë të vendimeve arkitekturore, bazuar mbi modelet arktitekturore siç jane stilet arkitekturore apo arkitekturat bazë referencë.

Dokumentimi i arkitekturës softuere dhe vlerësimi i saj u trajtuan në fund të kapitullit duke theksuar diskutuar avantazhet e metodës së përdorur ATAM në vlerësimin e arkitekturave të tilla në përshtatje të dokumentit të kërkesave të sistemit softuere.

KAPITULLI 3

Agjentët dhe Arkitekturat e Tyre

Sistemet softuere bazuar mbi agjentë

Mjaft kërkues janë fokusuar në problemet bazë të sistemeve të ndërtuar me agjentë duke arsyetuar kryesisht në modelet simbolike të brëndshme dhe planifikimin e veprimit. Këto modele bëheshin të papërdorshëm dhe të pamajftueshëm për sisteme ku agjentët duhet të vepronin në një mjedis dinamik dhe të paparashikueshëm. Në mjaft punime janë propozuar arkitektura të reja për ndërtimin e agjentëve. Ndërkohë që modeli vendimmarrës vë theksin mbi njohjen dhe zgjedhjen racionale, thelbi i arkitekturave të reja është më i drejtëpërdrejtë duke realizuar nga perceptimi në ndërveprim dinamik me mjedisin, për të arritur në modularizimin e sjelljes së përcaktuar nga veprime elementare. Kjo lloj perspektive do të përbënte fillimin e ndërtimit të një vizioni të ri mbi arkitekturën dhe zhvillimin e agjentëve. Parime të tillë kanë qënë në bazë të shumë modeleve të sistemeve agjentë dhe aplikacioneve të zhvilluara me sukses në praktikë.

Në këtë kapitull do të shpjegohet një model i avancuar për një sistem multiagjent të vendosur në qëndër të kërkimit tonë, duke paraprirë me një vështrim historik mbi sistemet multiagjentë dhe më tej do të qëndrojmë mbi modelin tonë i cili do të ilustrojë konceptin mbi arkitekturat e moduluara bazuar në agjentë për zhvillimin e një paradigmë të re në inxhinierinë softuere në projektimin dhe zhvillimin e sistemeve softuere bazuar mbi agjentët softuere.

Vështrim i përgjithshëm mbi agjentët

Në këtë paragraf kemi bërë një shikim historik mbi evolucionin e sistemeve bazuar mbi agjentë si dhe ndikimin e mjedisit në zhvillimin e modeleve që përballin projektimin e sistemeve të tilla.

3.1.1. Sistemet me një agjent

Fillimisht, fokusi i hulumtimit të agjencisë vendosur ishte në sistemet me një agjent të vetëm. Në këtë seksion, ne do të japim një pasqyrë të zhvillimit të mëvonshëm të agjentëve në arkitekturën përkatëse si dhe do të diskutojmë për rolin e mjedisit në këtë evolucion.

Agjentët robotikë reaktive

Në mes të viteve 1980, studiuesit u përballën me problemin ndërtimit të robotëve autonome që nëpërmjet sensorëve, janë të aftë gjenerojnë sjellje të sigurta, në një mjedis të paparashikuar,dhe në një botë në ndryshim [41]. Përpjekjet për të ndërtuar robotë të tillë me teknika tradicionale të inteligjencës artificiale, kanë treguar mangësi të tilla si dështime në misione, reagime jo në kohë reale [42]. Përveç kësaj, këto sisteme paraqesin disa probleme teorike, të tilla si problemi i kufizimit dhe mungesës së arsyetimit në kohë reale [43]. Kjo solli që një numër i studiuesve të arrinin në përfundimin se arsyetimi me modele simbolike të brëndshme dhe planifikimi i rendit të veprimeve për të arritur qëllimet është i parealizueshëm për agjentët robotikë. Në mjaft raste sjell situata të objektivave konfliktuale në mjedise komplekse.. Ky përfundim çoi në zhvillimin e një qasjeje rrënjësisht të re për të ndërtuar agjentët autonome. Karakteristikat kryesore të kësaj qasjeje janë përshkruar në [44]:

'Situatedness'. Agjentët të vendosur në një botë të njohur për ta, por në mënyrë direrkte ndikon në sjelljen e tyre.

'Embodiment'. Agjentët janë prezentë dhe kanë eksperiencat e tyre në këtë botë. Ata janë në ndërveprim direkt me të.

'Intelligence'. Agjentët kontrollohen për të qënë inteligjentë. Burimi i kësaj inteligjence nuk është i kufizuar për agjentët brënda botës ku ata veprojnë. Ajo vjen e rritet nga bashkëveprimi i agjentëve me mjedisin.

'Emergence'. Inteligjenca e sistemit vjen jo vetëm nga ndërveprimi i sistemit me mjedisin por edhe nga ndërveprimet midis komponentëve të tij.

Problemi kryesor në sistemet multiagjente është se të arsyetuarit dhe planifikimi i veprimeve bëhet i pamjaftueshëm për agjentët të cilët duhet të veprojnë në situata dinamike të mjedisit të paparashikuar. Kërkues të ndryshëm kanë propozuar në mënyrë radikale arkitektura të reja për ndërtimin e agjentëve. Ndërsa modelet deliberative theksojnë njohje eksplicite dhe zgjedhje racionale, arkitekturat e reja mbështeten në perceptim dhe veprim, modularizim të sjelljes dhe ndërveprim dinamik me mjedisin. Kjo perspektivë, e cila në përgjithësi i referohet sistemeve multiagjente shërben si bazë për ndërtimin e mjaft arkitekturave me agjentë për t'u aplikuar me sukses në praktikë. Në këtë kapitull, ne do të shpjegojmë modelin e zhvilluar në këtë kërkim.

3.1.1.1.Rreth arkitekturave të agjentëve

Arkitekturat e agjentëve janë mekanizmi kryesor që theksojnë komponentët autonome që mbështesin sjelljen efektive në mjedise dinamike të hapura të botës reale. Në fakt përpjekjet e para në fushën e përpunimit të bazuar në agjentë u

fokusuan në zhvillimin e arkitekturave të agjentëve inteligjentë dhe, qysh atëhere, u përcaktuan stilet më të fundit të arkitekturave.

Le të rreshtojmë disa modele arkitekturash duke filluar nga arkitekturat reaktive (në mënyrë të varfër) që veprojnë sipas modelit *stimulus – response*, *a*rkitektura Subsumtion dhe pastaj në arkitekturat deliberative që arsyetojnë sipas veprimeve të tyre siç janë arkitekturat BDI (belief desire intention) deri në arkitekturat ekstreme. Ka një ndarje të shtjelluar të arkitekturave të agjentëve në katër grupe kryesore të paraqitur në [45]. Po përmendim

Arkitekturat 'Logic based'.Këto arkitektura lindën nga teknikat e sistemeve bazuar në njohuritë në të cilat një mjedis simbolikisht prezantohet dhe manipulohet duke përdorur mekanizmat e arsyetimit.

Figura 3. Një arkitekturë Subsumtion për një agjent të thjeshtë

Arkitekturat 'Reactive'. Këto arkitektura implementojnë vendimmarrjen (*decision making)* si një hartim direkt nga gjendja në veprim dhe bazohen në mekanizmin '*stimulus response*' të trigeruar nga të dhënat e sensorit. Arkitektura realizon sjelljen inteligjente të gjeneruar jo nga paraqitje eksplicite dhe arsyetimi abstrakt, por është produkt i cilësive emergjente të disa sistemeve komplekse. Ne figurën 3.1 paraqitet arkitektura në fjalë. Kjo arkitekturë përcakton shtresa të gjendjeve të fundme të makinës që lidhen me sensorë që transmetojnë informacion në kohë reale.

Një shtresë lidh direkt perceptimin me veprimin, që do të thotë një gjendje e entitetit, e shtuar me shumë elemente në kohë. Këto shtresa formojnë një hierarki të sjelljes në të cilën më e ulta ka më pak kontroll se më e larta. Kështu vendimmarrja '*decision making*' arrihet nëpërmjet sjelljeve të drejtuara nga qëllimi. Agjentët mësojnë nga eksperiencat e tyre. Agjentët, të dizenjuar sipas arkitekturës subsumtion, perceptojnë kushtet dhe veprojnë, por nuk planifikojnë asgjë. Avantazhi i kësaj arkitekture është performanca më e mirë (përgjigjen shpejt por nuk arsyetojnë më

mirë) në mjedise dinamike ashtu sikurse ata shpesh janë më të thjeshtë në projektim sesa agjentët e ndërtuar sipas arkitekturës '*logic-based*'. Disavantazhi kryesor i agjentëve reaktivë është pikërisht se ata nuk zhvillojnë modele të rezultateve të mjedisit të tyre dhe bllokimi i gjendjes së agjentëve e bën të pamundur projektimin. **BDI** – janë arkitekturat më popullore. Ato kanë rrënjët e tyre në filozofi dhe ofrojnë një teori logjike që përcakton tre gjendjet (*belief* , *desire* dhe *intention)* duke përdorur modelet logjike. Mjaft sisteme agjentësh janë bazuar mbi këtë arkitekturë si JAM te[46], JACK tek [47], dMARS dhe JADEX. Më e njohura është PRS (procedurial reasoning system).[48] Kjo arkitekturë bazohet në katër struktura të dhënash kyçe siç duket në figurën 3.2.

Arkitektura të **shtresëzuara.** Janë arkitektura hibride që lejojnë sjellje reaktive dhe deliberative së bashku për agjentët e projektuar sipas kësaj arkitekture. Figura 3.3 paraqet një arkitekturë të tillë. Ka dy tipe të rrjedhës së kontrollit brënda arkitekturës së shtresëzuar:

Figura 3. Arkitektura PRS

- **Shtresëzimi horizontal-** këtu shtresat lidhen direkt me inputin dhe outputin sensorial (shih figurën) dhe çdo shtresë është duke vepruar si një agjent. Avantazhi i kësaj arkitekture është thjeshtësia e projektimit: nëse agjenti ka nevojë për n tipe të sjelljes, atëherë arkitektura do të kërkojë n shtresa[50]. Megjithatë, meqë një çdo shtresë është një agjent, veprimet e tyre mund të jenë të paqëndrueshme.

Figura 3. Shtresëzimi horizontal

- **Shtresëzimi vertikal**– eliminon mundësinë e paqëndrueshmerisë në hyrjet sensoriale paralele dhe rezultatet e rrjedhura prej tyre[51]. Siç shihet në figurën 3.4, kjo arkitekturë mund të ndahet në arkitektura kontrolli të tipit:një hapësh dhe dy hapësh

Në modelin e parë, kontrolli nis nga shtresa fillestare që merr të dhëna nga sensorët, deri tek shtresa finale që gjeneron veprime. Në modelin e dytë, të dhënat ngjiten dhe kontrollojnë kur zbresin. Avantazhi më kryesor i arkitekturës së shtresëzuar vertikalisht është që ndërveprimi midis shtresave zvogëlohet. Megjithatë, risku mbetet në varësinë e të gjitha shtresave dhe kjo sjell mungesë tolerance në rast dështimesh. Nëse një shtresë dështon, atëherë gjithë sistemi ka dështuar.

Figura 3. Shtresëzimi vertikal një hapësh dhe dy hapësh

Arkitektura të tjera që mbështesin vendimmarrjen në kohë reale, duke iu referuar agjentëve bazuar në sjellje, mund të përmendim Free Flow Architecture[52], Distributed Architecture for mobile Navigation [53]. Në këto arkitektura, sjellje të ndryshme gjenerojnë outpute nga votimi. Arbitrimi i sjelljes realizohet duke vlerësuar numrin më të madh të votave për një veprim. Kjo sjell që të ekzekutohet pikërisht ky veprim.

Për të ilustruar idenë e agjentëve bazuar në sjellje, le të shohim arkitekturën e rrjetit të agjentëve ANA[54]. Kjo arkitekturë e ilustruar në figurën 3.5, kombinon parimet e reaktivitetit të agjentëve si, dekompozimi në detyra dhe funksionalitete emergjente me arritjen e qëllimit në kohën e ekzekutimit të veprimit. Një ANA është një arkitekturë që zgjedh veprimin e një agjenti. Kjo arkitekturë konsiston në një bashkësi modulesh me kompetenca të veçanta, të lidhur në një rrjet së bashku me një mekanizëm që zgjedh një modul për ekzekutim. Çdo modul paraqet një sjellje të veçantë të agjentit. Çdo modul kompetent ka një listë të parakushteve që duhet të bëhen të vërteta përpara se moduli të bëhet i ekzekutueshëm. Akoma më tej, çdo modul ka një nivel veprimi. Kur një nivel aktiviteti për një modul kompetence arrin një pikë të caktuar, ai mund të zgjidhet për ekzekutim.

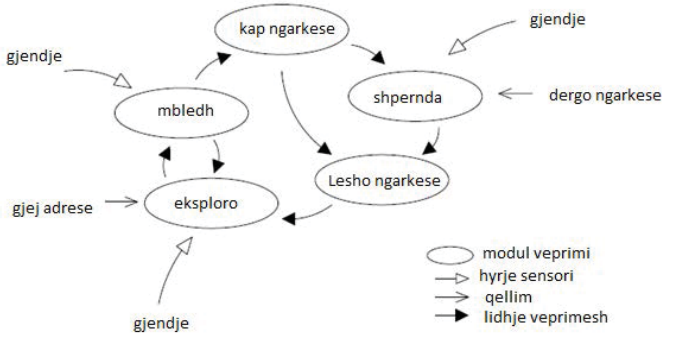

Figura 3. Arkitektura ANA

3.1.2. Modelet eksplicite të botës dhe arkitekturat hibride të agjentëve

Që në fillim të zhvillimit të sistemeve bazuar në agjente, ka patur një ka pasur një diskutim të vazhdueshëm në lidhje me shfrytëzimin e modeleve të botës së brëndshme në arkitekturat me agjentë. Në [57] është argumentuar kundër nevojës për çfarëdo lloj modeli të botës apo në çdo lloj niveli njohës në përgjithësi. Hulumtues të tjerë treguan se si njohuritë mund të hartohen me implementime jo-simbolike. Në [58] theksohet se agjentët autonome, pa modelet e brendshme, gjithmonë do të jenë shumë të kufizuar. Këtu propozohet që të përdoren prezantime analogjike në vënd të përfaqësimeve simbolike. Një përfaqësim simbolik i botës është një përshkrim i nxjerrë nga bota që perceptohet në disa gjuhë, p.sh, një grup i fjalive të shprehura në logjikën e rendit të parë. Një përfaqësim analogjik, nga ana tjetër, është një model i botës. Shembuj mund të jenë një hartë, një foto dhe një diagram. Në [59], njohja paraqitëse është mjaft e rëndësishme për orientimin dhe navigimin e robotëve. Njohja e botës, e fituar në mënyrë dinamike, mund të shfrytëzohen për të rritur fleksibilitetin dhe efektivitetin në navigimin reaktiv.

I lidhur me çështjen e modeleve eksplicite të botës është pozicioni i planifikimit. Në [60], përpunohet ideja mbi përdorimin e planeve në vendim-marrjen e agjentëve. Autorët fokusohen mbi planet duke i vlerësuar si:

- **Planet si një burim njohjeje të agjentit**. Në funksion të planeve si një burim njohjeje, agjentët i përdorin ato për të gjetur një pikë referimi ndër të tjera në vazhdimësi për të rivendosur se çfarë të bëjë më tej.
- **Planet për veprim**. Në funksion të planeve të veprimit, agjentët ekzekutojnë planet për të arritur objektivat. Pra, një plan është një recetë e veprimeve të mëtejshme për të arritur një qëllim

Kjo analizë e bërë hedh themelet për studimin mbi planifikimin reaktiv të diskutuar në [61]. Në [62], autorët paraqesin idenë e arkitekturës hibrid. **Një arkitekturë hibride** kombinon një nënsistem deliberativ me një nënsistem bazuar në sjellje. Nënsistemi deliberativ lejon njohjen prezantuese që do të përdoret për planifikimin e qëllimeve para ekzekutimit, ndërsa nën-sistemi bazuar në sjellje mban përgjegjësinë, fuqinë dhe fleksibilitetin e sistemeve të konsideruar thjesht reaktivë. Gjatë viteve në vazhdim, u zhvilluan mjaft arkitektura hibride bazuar në sjellje. Sot, është modeli më i zakonshëm në fushën e robotikës, dhënë dhe diskutuar tek [63]. Një element kyç në arkitekturat hibride është ndërfaqja ndërmjet reaktivitetit dhe shtresës deliberative pasi ajo lidh reagimin e shpejtë me planifikimin e gjatë.

Një model i përgjithshëm për të balancuar reaktivitetin me planifikimin është futja e një shtrese të qartë të tretë që luan rolin e koordinatorit ndërmjet shtresës reaktive dhe deliberative. Megjithatë në përgjithësi, koordinimi i reaktivitetit dhe

vëzhgimit deliberativ ende nuk është kuptuar mirë dhe është subjekt i mjaft hulumtimeve.

3.1.3. Përfundime të kapitullit

Në këtë kapitull trajtuam disa çështje kryesore në lidhje me sistemet e bazuar në agjentë robotikë. Duke filluar nga parimet fillestare të reaktivitetit, janë zhvilluar një numër i madh arkitekturash bazuar në agjentë. Tashmë janë identifikuar tre klasa modelesh:

- Agjentë robotikë që synojnë ndërveprim dinamik me mjedisin. Pjesa harduere e agjentit robotik realizon direkt perceptimin në veprim, duke aftësuar reaktivitetin në kohë reale.
- Agjentë bazuar në sjellje që vënë në dukje nevojën për zgjedhje veprimi dinamik dhe fleksibël duke synuar përshtatjen me mjedise komplekse. Arkitekturat që mbështesin agjentë bazuar në sjellje, përballen me arbitrimin e sjelljeve të ekzekutuara paralelisht dhe lejojnë ndryshime të qëllimeve në mënyrë dinamike gjatë kohës së ekzekutimit të veprimeve që performojnë sjellje.
- Agjentë hibrid shpjegojnë njohjen prezantuese rreth mjedisit. Arkitekturat për agjentë hibridë integrojnë arsyetimin e brëndshëm për mjedisin dhe planifikimin me reaktivitetin ndaj stimulimeve duke synuar kombinimin e avantazheve të planifikimit me përgjigjet e shpejta.

Këto modele kanë dy cilësi të përbashkëta. Fokusi shtrihet në arkitekturën e një agjenti të veçantë. Arkitekturat ndryshojnë në mënyrën si ato zgjidhin problemin e përzgjedhjes së veprimit. Këto arkitektura nuk mbështesin ndërveprimin social.

Modelet theksojnë rëndësinë e dinamikës mjedisore. Megjithatë, vetë mjedisi konsiderohet si botë e jashtme, që do të thotë mjedisi nuk është konsideruar pjesë e modelit të arkitekturës së sistemit bazuar në agjentë.

KAPITULLI 4

Sjellja e Sistemeve Multiagjente

Nga sjellja reaktive kolektive në sistemet multiagjente

Që prej viteve 1990, hulumtues të agjensisë robotike kanë studiuar sistemet ku multiagjentët punojnë së bashku për të realizuar funksionalitetin e sistemit. Në këto sisteme, agjentët shfrytëzojnë mjedisin për të ndarë informacionin dhe koordinojnë sjelljet e tyre. Le të shohim disa modele që janë zhvilluar bazuar në këtë koncept.

4.1.1. Sjellja Reaktive Kolektive

Në [64], është analizuar sjellja midis agjentëve në një agjensi robotike. Sjellja agregate e një sistemi multiagjent shfaqet nga ndërveprimi i agjentëve, ku secili ndjek një bashkësi të rregullave të sjelljeve të thjeshta elementare. Këto teknika me robotët reale janë paraqitur në[65], duke treguar se si një bashkësi robotësh prodhojnë një sjellje për paketë sjellje. Çdo robot pajiset me një bashkësi të sjelljeve të thjeshta elementare nga të cilat ai zgjedh sjelljen më të përshtatshme sipas kontekstit të mjedisit prezent. Psh, një zgjedhje do të ishte sipas pozicionit të tij aktual në lidhje me robotët e tjerë.

Në[66] është diskutuar një tjetër formë e koordinimit reaktiv, duke përdorur fushat vektor për të kontrolluar uljen dhe lëvizjen të një grupi aeroplanësh në një simulim. Në këtë model, çdo agjent udhëhiqet nga një fushë potenciale që ai ndërton, bazuar nga qëllimet dhe pengesat duke përfshirë këtu edhe agjentët e tjerë në mënyrë respektive.

Sistemet Agjente Stigmergjike

Në [67], termi '*stigmergy*' trajtohet si koncept që tregon se si entitete individuale ndërveprojnë në mënyrë jodirekte nëpërmjet një mjedisi të përbashkët: Një individ ndryshon mjedisin dhe të tjerët i përgjigjen këtij ndryshimi duke rifreskuar atë në njohjen e tyre. Në [68] paraqitet një skenar se si robotët eksplorues përmirësojnë kërkimin e objektivit duke shënuar mjedisin. Kur një robot gjen objekte burimi target, ai vendos një shenjë në mjedis duke lejuar që të tjerë robotë të gjejnë lehtë objektet burim duke qenë të informuar më parë. Për të qenë në mjedis dinamik, këto shënime zhduken gradualisht me kalimin e kohës. Ky mekanizëm i koordinimit jodirekt nëpërmjet mjedisit, kombinon përforcimin e kontrollit pozitiv nga agjenti me mirëmbajtjen e njohjes në gjëndjen e saj reale. '*Stigmergy*' është parë nga shumë kërkues. Kështu, në[69] ky koncept trajtohet si një mundësi që lejon parime të

sistemeve agjente natyrale të mund të përdoren dhe të aplikohen në sistemet agjente artificiale.

Sistemet multiagjente robotike

Në mënyrë të veçantë, në sistemet multiagjente robotike vihet theksi në rëndësinë e arkitekturës për agjentët dhe mjedisin. Aktiviteti i agjentit ndahet nga aktiviteti i mjedisit. Veprimet e agjentit janë subjekt i ligjeve që përfaqësojnë kufizime specifike të fushës ku ata veprojnë. Karakteristika të rëndësishme të sistemeve multiagjente mund të listojmë:

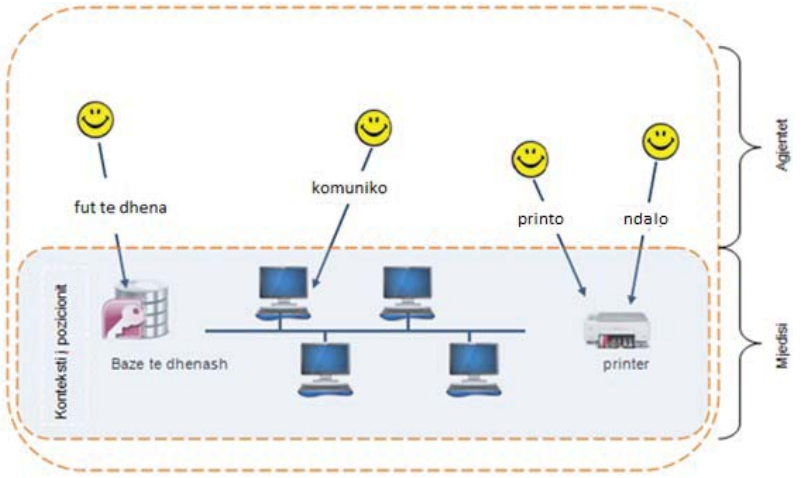

Figura 4. Konteksti i pozicionit të një mjedisi për një sistem me agjentë

4.1.2. Nivelet e mbështetjes të siguruara nga mjedisi

- Agjentët dhe mjedisi janë pjesë eksplicite e sistemeve me përgjegjësi specifike.
- Funksionaliteti i sistemit vjen nga ndërveprime jo të drejtpërdrejta të agjentëve nëpërmjet mjedisit.

Megjithëse sistemet multiagjente robotike janë aplikuar me sukses në praktikë, përsëri mjaft çështje mbeten të hapura për kërkues të ndryshëm të fushës së agjentëve.

Në këtë punim, arkitektura e aplikacionit është një model i gatshëm që përfshin funksionalitetet bazë duke siguruar një zhvillim dhe rritje të sistemit multiagjent mbi module ekzistuese.

Mjedisi i agjentëve

Inxhinieria e sistemeve multiagjente e konsideron mjedisin si infrastrukturën e agjentëve. Le të shpjegojmë se përse mjaft funksionalitete të sistemit multiagjent i dedikohen mjedisit dhe njohjes së tij. Më tej do të japim një përkufizim të mjedisit si një komponent kryesor në arkitekturën për arritjen e abstraksionit.Figura 9 paraqet një shembull i strukturës së mjedisit për një agjensi. Mjedisi fokusohet si kontekst i poyicionimit të veprimit të agjensisë.

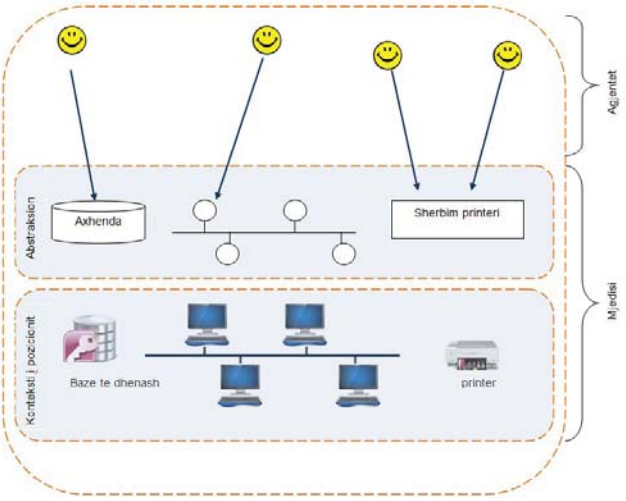

Figura 4. Nivelet e abstraksionit në një sistem bazuar në agjentë

Agjentët mund të përdorin tre nivele mbështetes, të siguruar nga mjedisi për të arritur objektivat e tyre.

Niveli basik. Në këtë nivel, mjedisi aftëson agjentët të aksesojnë në kontekstin e pozicionimit apo të vendosjes së aplikacionit. Me këtë frazë ne i referohemi një bashkësie komponentësh harduere dhe softuere, si dhe burime të jashtme me të cilat sistemi multiagjent ndërvepron (sensorë dhe aktuatorë, printer, rrjet, bazë të dhënash, një shërbim web, etj). Sigurimi i këtij aksesi për agjentët është një funksionalitet thelbesor i mjedisit në çdo sistem agjent. Ai paraqet pjesën më elementare në një mjedis për një sistem agjentësh. Figura 4.1 përshkruan skenarë shembulli se si agjentët aksesojnë direkt në kontekstin e vendndodhjes. Agjenti në të majtë fut vlera në tabelën e bazës se të dhënave. Ndërsa agjenti në qendër komunikon në një portë të veçantë për të kontaktuar një agjent tjetër. Dy agjentët në të djathtë aksesojnë për të ndarë përdorimin e një printeri.

Nga këto shembuj bëhet e qartë se aksesimi i drejtpërdrejtë në kontekstin e pozicionit dhe vendosjes detyron agjentët që të merren me detajet të nivelit të ulët të rrjetit, burimet, dhe kështu me radhë. Sidomos në kontekstet dinamike edhe të paparashikueshme të vendosjes si rrjeta ad-hoc, detyra e agjentëve bëhet më e rëndë dhe më e vështirë.

Niveli i abstraksionit. Ky nivel lidh nga ana konceptuale abstraksionin e agjentit me nivelin e pozicionimit dhe te vendosjes. Niveli i abstraksionit siguron një ndërfaqe të vetën për agjentët, duke fshehur detaje të nivelit të ulët të rrjetit, burime të jashtme ose nënsisteme të tjera nga sistemi me agjentë. Agjentët mund të hyjnë në abstraksionet e burimeve për të cilat ata janë të interesuar. Proçedura e ruajtjes në axhendën e bazës së të dhënave lejon agjentët për të bashkëvepruar me bazën e të dhënave në një nivel më të lartë të abstraksionit. Në shembull, agjenti shton një takim në rend të ditës. Niveli i abstraksionit merr përsipër që të transformojë komandat e agjentëve në instruksione SQL për të bashkëvepruar me bazën e të dhënave aktuale. Abstraksioni si nivel në rrjet për figurën 4.2 shërbimi print siguron një abstraksion të printerit që lejon agjentët të instruktojnë shërbimin e printimit me qëllim printimin e një dokumenti në vend të dërgimit të një rradhe të gjatë bitësh në portën e daljes. Në përgjithësi niveli i abstraksionit të një mjedisi është i përbashkët në një sistem me agjente sidomos në një kontekst dinamik dhe të paparashikuar të pozicionimit.

Niveli ndërmjetësues-ndërveprues. Në vazhdim të nivelit të abstraksionit, një mjedis mund të sigurojë një mjedis ndërmjetës ndërveprues për të mbështetur ndërveprimin në një mjedis të ndërmjetëm. Ky nivel ofron mbështetje në këto drejtime:

- Të rregullojë aksesin në burime të përbashkëta
- Të ndërmjetësoje ndërveprimin midis agjentëve

Agjenti në qëndër të figurës 4.3 bashkëvepron me një infrastrukturë pheromone që është vendosur në krye të një topologjie rrjeti. Agjentët në anën e djathtë kontrollojnë aktuatoret. Niveli abstraksion ofron një hartë objektesh të mjedisit fizik. Niveli i ndërmjetësimit dhe ndërveprimit shton shenja në hartë për të rregulluar veprimet e mëtejshme të agjentëve të tjerë. Disa nga shenjat mund të manipulohen nga agjentët. Shenja të tjera mund të menaxhohen nga mjedisi. Për shembull, një shenjë që tregon koston për të arritur në një destinacion të veçantë mund të përshtatet në përputhje me trafikun prezent. Me një nivel bashkëveprimi dhe ndërmjetësimi, mjedisi bëhet një subjekt aktiv në sistem. Mjedisi rregullon aktivitetin e veçantë në sistem. Në mënyrë tipike, mjedisi mbështet pjesën e pavarur të aktiviteteve të agjentëve. Shembuj të aktiviteteve të tilla janë grumbullimi, difusion dhe avullimi i feromoneve dixhitale[70] dhe mbështetje për koordinim bazuar në përdorimin e shënjuesve[71].

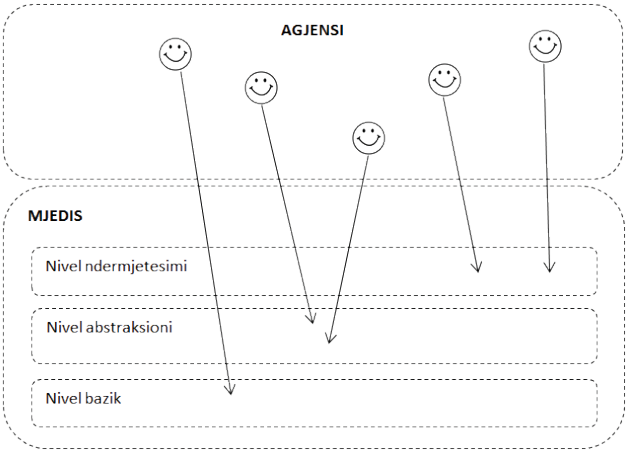

Figura 4. Niveli i ndërmjetësimit dhe ndërveprimit në sistemin e bazuar mbi agjentë softuere

Shtresa e ndërmjetësimit dhe ndërveprimit mundëson agjentët për të shfrytëzuar mjedisin për të koordinuar sjelljet e tyre.

Refleksioni. Të tre nivelet paraqesin shkallë të ndryshme të funksionalitetit të siguruara nga mjedisi të cilat agjentët mund t'i përdorin për të arritur qëllimet e tyre. Në përgjithësi ne e konsiderojmë mjedisin si një infrastrukturë për koordinimin e agjentëve. Kjo infrastrukturë siguron një zgjidhje të ripërdorshme që mund të shfrytëzohet në shumë aplikacione të tjera. Një infrastrukturë në menyre tipike perfaqëson një bashkësi përgjegjësish në sistem. Për një aplikacion të veçantë të gjitha përgjegjësitë që nuk mund të manaxhohen nga infrastruktura, mbeten të mbulohen nga agjentët duke kaluar në konceptimin e agjentëve komplekse. Gjithashtu, mund të themi se infrastrukturat janë në mënyrë tipike të lidhura me një lloj modeli koordinimi të veçantë siç janë shënimet, fushat , etj. Megjithatë, një zgjidhje mund të merret si një zgjidhje hibride duke integruar modele të ndryshme sipas kërkesave të sistemit me të cilin kemi të bëjmë. Tashmë infrastrukturat e sistemeve multiagjente nuk zhvillohen me këtë lloj konceptimi të integrimit. Ato fokusohen kryesisht në një bashkësi të përgjegjësive të sistemit. Infrastrukturat e komunikimit sigurojnë mbështetje për transferimin e mesazheve, koordinimin jodirekt, etj. Në punim ne do të konsiderojmë mjedisin si një abstraksion të përfaqsuar në një bllok që mund të shfrytëzohet në mënyrë krijuese në aplikacionet bazuar në agjentë.

Përcaktime të mjedisit si niveli i parë i abstraksionit

Në mjaft punime bëhet fjalë për një program mjedisi të përgjithshëm.[72]. Ky program bën të mundur që agjentët të perceptojnë dhe të marrin sërish veprimet e tyre. Më tej, programi risjell gjëndjen e mjedisit bazuar në veprimet e agjentëve dhe

mundësisht, proçese të tjera në mjedise që nuk janë marrë parasysh nga agjentët. Programi i mjedisit ilustron marrdhëniet bazë midis agjentëve dhe mjedisit të tyre.

Në [73] specifikohen karakteristikat e mjedisit të cilat janë të zbatueshme në fusha aplikimesh me agjentë. Kështu , mund të rendisim:

- Në çdo çast kohe ka mjaft rrugë të ndryshme në të cilat mjedisi ndryshon.
- Në çdo çast kohe ekzistojnë mjaft veprime të mundshme që agjenti mund të performojë.
- Objektiva të ndryshme mund të mos jenë të arritshme në të njëjtën kohë.
- Veprimet që arrijnë objektivat në mënyrën më të mirë varen nga gjëndja e mjedisit.
- Mjedisi mund të ndjehet vetëm në mënyrë lokale
- Numri i përllogaritjeve dhe veprimeve që kryhen, varet nga numri i herëve në të cilat mjedisi ndryshon.

Në një tjetër punim [74] përcaktohet mjedisi si një hapësire në të cilën objektet duke përfshirë edhe agjentët janë të vendosur dhe kanë një pozicion të caktuar në një çast kohe të dhënë. Objektet janë të lidhur me njëri tjetrin dhe agjentët janë të aftë të perceptojnë objektet si dhe të manipulojnë ato në hapësirën e dhënë. Veprimet e agjentëve janë subjekt i ligjeve të universitetit që përcaktojnë efektet e veprimeve në mjedis. Funksioni mbartës që ka mjedisi thekson ndarjen e veprimeve të agjentëve në mjedis.

Në [75] flitet për katër blloqe ndërtimi në sistemet me agjentë.

- Agjentë si njësi përpunuese
- Ndërveprime si elementë të komunikimit midis entiteteve
- Organizata si elementë të bashkësive strukturore me sisteme bazuar në agjentë
- Mjedisi si element i varur nga fusha, për strukturimin e ndërveprimeve të jashtme midis entiteteve.

Kjo perspektivë thekson cilësitë e strukturimit të elementeve të jashtëm të sistemit bazuar në agjentë. Në [76] mjedisi përcaktohet si një binom *(gjendje, proçess*)**.** *Gjendje* është një bashkësi vlerash që percaktojnë mjedisin totalisht duke përfshirë edhe agjentët dhe objektet brënda mjedisit. Ndërsa proçesi tregon që mjedisi vetë është një entitet aktiv. Mjedisi ka proçesin e vet që ndryshon gjendje, pavarësisht nga veprimet e agjentëve të vendosur në të. Qëllimi kryesor i proçesit është të implementojë dinamikën në mjedis. Pra, kemi të bëjmë me një natyrë aktive të mjedisit.

Në një përcaktim tjetër në [77], mjedisi konsiderohet si kusht nën të cilin një entitet (agjent) ekziston. Autorët bëjnë dallimin midis mjedisit fizik dhe mjedisit të komunikimit.

1. Mjedisi fizik: siguron ligjet, rregullat që drejtojnë ekzistencën fizike të agjentëve dhe objekteve.
2. Mjedisi i komunikimit siguron :
    a. Parimet dhe proçeset që drejtojnë dhe mbështesin këmbimin e ideve, njohjes dhe informacionit
    b. Funksionet dhe strukturat që përdoren për të kembyer komunikimin si rolet, grupet si dhe protokollet e ndërveprimit midis roleve dhe grupeve.

Në përkufizimin e përgjthshëm, e konsiderojmë mjedisin si:

Përfundim: ***Mjedisi është një nivel i parë i abstraksionit që siguron kushtet e nevojshme për ekzistencën e agjentëve dhe luan rolin e ndërmjetësuesit në ndërveprimin midis agjentëve dhe në aksesimin e burimeve.***

Si të mund ta shohim këtë përcaktim të ri? Duke e konsideruar mjedisin si një bllok ndërtues në sistemet multiagjente, ai enkapsulon përgjegj;sitë e veta ndaj agjentëve që ekzistojnë në të. Mjedisi është një pjesë themelore e sistemeve multiagjentë. Mjedisi është gjithë bota ku agjentët jetojnë , ndërveprojnë dhe ku mund të vërehen dhe të vlerësohen efektet e ketij bashkëveprimi. Akoma më tej, mjedisi enkapsulon dhe siguron akses të netitet e jashtme dhe burimet dhe aftëson agjentët që të ndërveprojnë duke luajtur rolin e ndërmjetësuesit. Mjedisi është materia ku njëkohësisht intermediohet për ndërveprim midis agjentetëve dhe akses në burime. Kështu burimet ndahen midis agjentëve duke manaxhuar nëpeërmejt koordinimit. Duke e njohur mjedisin si niveli i parë i abstraksionit, e shohim si një element arkitekturor të rëndësishëm në aplikacionet me multiagjentë. Mjedisi përcaktohet në arkitektura si një bashkësi e personalizueshme e përgjegjësive sipas kërkesave të aplikacionit. Duke vendosur përgjegjësitë midis agjentëve, mjedisi ndihmon mjaft në manaxhimin e kompleksitetit të aplikacionit duke përmirësuar dhe ndarë më mirë kufizimet në sistemet multiagjente.

Funksionalitetet e mjedisit

Le të shohim shkurt një seri të funksioneve kryesore që shoqërojnë një mjedis në një sistem multiagjent.

4.1.3. Strukturat e mjedisit në sistemet multiagjentë.

Mjedisi siguron një hapësirë të përbashkët virtuale apo fizike për agjentët dhe burimet. Agjentët dhe burimet janë të lidhur në mënyrë dinamike me njëri tjetrin.

Është detyrë e mjedisit të përcaktojë rregullat me të cilat duhet të kompletohen marrëdhëniet midis entiteteve në këtë mjedis. Kjo e bën këtë mjedis si një entitet strukturues për një sistem multiagjent. Në përgjithësi, mund të dallojmë këto forma strukturimi.

- Struktura fizike që i referohet një strukture hapësinore apo topologjie[78].
- Struktura e komunikimit i referohet infrastrukturës për transferimin e mesazhit ose të komunikimit jodirekt[78].
- Struktura sociale që i referohet strukturës organizative të mjedisit në terma të roleve, grupeve, shoqërive, etj[78].

Strukturimi është një funksionalitet themelor i mjedisit. Strukturat e mjedisit mund te jenë të përcaktuara apo të detyrueshme nga kufizime të fushës se aplikacionit por gjithashtu ato mund të zgjidhen me kujdes në bazë të arkitekturës së sistemit multiagjent. Në një mjedsi të shpërndarë, strukturat fizike dhe të komunikimit janë pjesë e kontekstit të zhvillimit dhe duhet të mbështeten nga një nivel i caktuar i abstraksionit. Ndërsa struktura sociale sigurohet në nivelin e ndërveprimit dhe të ndërmjeteësimit sipas skemës që kemi parë në figurën më sipër.

4.1.4. Mjedisi me burime të inkorporuara.

Një funksionalitet i rëndësishëm i mjedisit është të inkorporojë burime. Burimet janë të vendosura zakonisht në një strukturë fizike. Mjedisi duhet të sigurojë mbështetje në nivelin e abstraksionit duke mbajtur detaje të vogla të burimeve dhe shërbimeve për agjentët.

- **Mjedisi është mbështetës i dinamikës.** Përveç, veprimeve të agjentëve, mjedisi mund të ketë proçeset e veta, të pavarura nga agjentët. Një shembull i aktivitetit të mjedisit është një fushë vetë manaxhuese në një rrjet. Kur topologjia e rrjetit fizik ndryshon, mjedisi duhet të sigurojë qëdrueshmërinë e fushave, të ofrojë konceptin e pamjes që ofron një aplikim për përditësimi për agjentët në një rrjet dinamik. Mjedisi mundet gjithashtu, të sigurojë mbështetje për gjendjen e mirëmbajtjes së agjentëve siç janë target që përdoren në koordinim. Varur nga natyra e dinamikës, mirëmbajtja e tij mund të sigurohet në nivelin e abstraksionit(për shembul ruajtja e të dhënave nga sensorët) ose në nivelin e ndërmjetësimit dhe të ndërveprimit(për shembull regjistrimi e shënimeve për qëllime të koordinimit).
- **Mjedisi është lokalisht i dukshëm dhe i kontrollueshëm nga agjëntët.** Agjentët duhet të jenë të aftë të inspektojnë struktura të ndryshme të mjedisit sikurse, edhe burimet dhe nëse është e mundur gjendje të jashtme të agjentëve të tjere. Vëzhgimi i një strukturë është zakonisht i kufizuar në

këndvështrimin e pozicionimit të agjentit në të(në kontekstin hapsinor, të komunikimit dhe social). Agjenti duhet të jetë i aftë të vëzhgojë mjedisin në nivelin e duhur te abstraksionit.

- **Mjedisi është lokalisht i aksesueshëm nga agjentët.** Agjenti duhet të jete në gjëndje të aksesojë struktura të ndryshme të mjedisit sikurse burime dhe gjëndje të jashtme të agjentëve të tjerë nëse është e mundur. Aksesimi në strukturat hapësinore kufizohet deri në kontekstin e mbështetjes për lëvizshmërinë, metrikët, etj. Ndërsa aksesimi në infrastrukturën e komunikimit i referohet sigurimë i komunikimit direkt ose jodirekt. Për strukturat sociale, aksesimi ka të bëjë me organizime, anëtarësime, etj. Në përgjithësi, burimet mund të perceptohen, të modifikohen, të gjenerohen apo të konsumohen nga agjentët. Fakti se cili agjent do të aksesojë në një burim të veçantë mund të varet nga disa faktorë siç është natyra e burimit, kapacitetet e agjentit, marrëdhëniet ekzistuese midis burimeve dhe agjentëve, etj. Agjentët duhet të jenë në gjendje të aksesojnë mjedisin në një nivel të vetin dhe të përshtatshëm të abstraksionit. Suporti për akses në kontekstin social është vendosur në nivelin ndërmjetësimit dhe të ndërveprimit.
- **Mjedisi është përcaktues i rregullave për sistemet multiagjentë.** Mjedisi mund të përcaktojë tipe të ndryshme të rregullave në të gjitha entitetet në një sistem multiagjentë. Këto rregulla i referohen kufizimeve të vendosura nga fusha e aplikacionit ose ligjeve të vendosura nga projektuesi. Rregullat mund të forcojnë aksesimin në burime specifike për agjentë të veçantë ose të përcaktjnë rezultatet e marra nga ndërveprimet mes agjentëve. Rregullat i referohen aksesimit të përbashkët në burime dhe kufizimeve në ndërveprimin mes agjentëve. Zakonisht këto rregulla vendosen në nivelin e ndërveprim-ndërmjetesimit.

Mekanizmat e përshtatjes për agjentët robotikë

Motivimi për modelin e trajtuar është prezantimi i një arkitekture për një mbështetje më të mirë në proçesin e projektimit në inxhinierinë e agjentëve si softuere. Qëllimi kryesor është të sigurojë një model të veçantë për perceptimin dhe komunikimin si dhe të integroje këto modele me zgjedhjen e veprimit. Gjithashtu tjetër qëllim është sigurimi i mekanizmave që aftësojnë agjentët të përshtasin sjelljen e tyre me ndryshimet e kushteve të mjedisit. Le të shohim në fillim gjendjen e agjentëve dhe të diskutojmë mbi perceptimin selektiv, selektimin e avancuar të veprimit bazuar ne sjellje si dhe komunikimin bazuar në protokolle.

4.1.5. Gjendjet e agjentit robotik

Në të kundërt me agjentët e bazuar në njohje, agjentët robotike nuk mund te ndërtojne një model të brendshëm të mjedisit. Por ata stimulojnë të shfrytëzojnë mjedisin vetë si një burim informacioni. Agjentë të tillë përdorin njohjen e botës të drejtojnë vendimet e tyre me parimin **"bëhet këtu dhe në këtë çast".** Një agjent robotik nuk mban gjurmë të kujtesës hipotetike për të ardhmen e gjendjes apo të hetojë për implikime të ndryshime të gjendjes apo planit të veprimeve. është e reëndësishme që ne të bëjmë dallimin midis gjendjes së përbashkët dhe gjendjes së brendshme.

- **Gjendje e përbashkët** i referohet gjendjes së përbashkët midis agjentëve. Shembuj të tillë janë gjendje që i referohen elementeve të vërejtur në mjedis dhe gjendja e ardhur nga të dhëna që ndryshojnë nëpërmjet mesazheve.
    - **Gjendje e pergjithshme statike**: kur gjendja e agjentit nuk ndryshon me kalimin e kohës.
    - **Gjendje dinamike**: kur gjendja e agjentit ndryshon me kalimin e kohës në kuptimin e kontekstit prezent të agjentit
- **Gjendje e brendshme** i referohet gjëndjes së agjentit që nuk është e përbashkët me agjentët e tjerë të sistemit. Kjo gjendje mund të jetë statike ose dinamike. Shembuj të gjendjes së brendshme janë parametra të ndryshëm të mekanizmit të përzgjedhjes së veprimit bazuar mbi sjellje. Një agjent mund të përdor këtë gjëndje për të përshtatur sjelljen e tij në kohë. [79]

4.1.6. Perceptimi selektiv i mjedisit

Perceptim do të quajmë aftësinë e një agjenti për të ndjerë fqinjët e tij duke rezultuar në një perceptim të mjedisit. Një perceptim përshkruan mjedisin e sensuar në një formë shprehjesh qe mund të kuptohet nga agjenti. Megjithatë perceptimi është mjaft prezent në çdo sistem multiagjent prandaj dhe modelimi i tij është një çështje e veçantë kur flitet per mjedisin softuere. Problemi kryesor është marrëdhënia perceptim –veprim. Modeli i perceptimit selektiv aftëson një agjent të drejtojë perceptimin e vet në aspekte të ndjeshme të mjedisit në përputhje me detyrën prezente**.** Kjo bën të mundur dhe lehtëson mjaft një njohje të situatës dhe mbajtjen e të dhënave të marra nga perceptimi në përpunim të vazhdueshëm për të arritur kontrollin e gjëndjes.Për të drejtuar perceptimin e vet një agjent zgjedh një bashkësi të fokusimesh. Do të thotë që çdo fokus i seciles bashkësi të zgjedhur karakterizohet nga një përceptueshmëri e veçantë. Selektimi i fokusit aftëson një agjent të drejtojë perceptimin e vet, lejon agjentin të ndjejë mjedisin veteë për informacion specifik.

Paraqitja konsiston në një strukturë të dhënash që mbajnë vlera të fushave në një rang të përcaktuar të sensimit. Për të interpretuar një paraqitje, agjentët përdorin përshkrimet.

Një **përshkrim** aftëson një agjent për të nxjerrë një perceptim nga një paraqitje. Një perceptim (percept) konsiston në elemente të dhënash që përshkruajnë mjedisin e sensuar në një formë që mund të kuptohet nga nga një makineri e brendshme e agjentit. Akoma me tej, perceptimi selektiv aftëson një agjent të zgjedhe një set filtrash. Këto lejojnë agjentin të selektojë vetëm ato elemente të dhënash të një perceptimi që përputhen me kriterin e selektimit specifik. Çdo filtër vendos kushte në një perceptim që përcakton se cilët elemente të dhënash kalojnë filtrimin.

4.1.7. Përmbledhje e kapitullit

Në këtë kapitull trajtuam sistemet multiagjentë si sisteme ku entitete të pavarur ndajnë informacionin në një mjedis të përbashkët dhe koordinojnë sjelljet e tyre. Le të shohim disa modele që janë zhvilluar. Mbi këtë objektiv bazë trajtuam sjelljen reaktive kolektive midis një bashkësie agjentësh. Arritëm në përfundimin se sjellja agregate e një sistemi multiagjent shfaqet nga ndërveprimi i agjentëve multiple ku secili ndjek një bashkësi të rregullave të sjelljeve të thjeshta elementare.

Në këtë kapitull sollëm termin *stigmergy* si koncept që tregon se si entitete individuale ndërveprojnë në mënyrë indirekte nëpërmjet një mjedisi të përbashkët: një individ ndryshon mjedisin dhe të tjerët i përgjigjen këtij ndryshimi duke ndryshuar atë në rradhën e tyre.

Folëm ne menyrë të veçantë për sistemet multiagjentë robotike ku u vu theksi në rëndësinë e arkitekturës për agjentët dhe mjedisin. U përcaktuan karakteristika të rëndësishme të sistemeve multiagjente në lidhje me mjedisin.

Sfidat kryesore që mundëm të trajtonim ishin:

- Shtrirja e arkitekturave të agjentëve në zhvillimin e mekanizmave të përzgjedhjes së veprimit në lidhje të ngushtë me perceptimin dhe komunikimin.
- Aftesimi i agjenteve per te patur nje nderveprim social dhe per te luajtur role te ndryshme ne bashkepunime te ndryshme.
- Promovimi i mjedisit si nje abstraksion i shkalles se pare qe mund te perdoret ne menyre krijuese ne aplikacionet bazuar ne multiagjente.
- Zhvillimi i nje metodologjie bazuar në parime të caktuara në proçesin e inxhinierisë për sistemet multiagjente.

Në këtë kapitull, ne arritëm në përfundimin se arkitektura multiagjente është një model i gatshëm që përfshin funksionalitetet bazë të aplikacionit softuere, duke siguruar një zhvillim dhe inkrementim të sistemit multiagjent mbi module ekzistuese.

KAPITULLI 5

Arkitektura Bazë e Sistemeve Multiagjente

Projektimi Arkitekturor i moduluar i një sistemi bazuar në agjentë

Në këtë kapitull do të trajtojmë se si aplikohet një arkitekturë e moduluar në projektimin e sistemeve multiagjentë. Aplikimi i kesaj arkitekture i ndihmon mjaft projektuesit në aplikacione të ngjashme, në mënyrë që të zhvillojnë me shpejtësi sisteme të tilla me funksionalitete të ngjashme vetë manaxhuese. Po japim një arkitekturë të trajtuar në planin teorik, të ndarë në dy pjesë.

- Së pari, do të sjellim një model të integruar për sistemet multiagjentë që sintetizon funksinalitetet e agjentit dhe të mjedisit te tij.
- Së dyti, do të paraqitet mënyra e dokumentimit të kësaj arkitekture në disa kënd vështrime. Gjatë zhvillimit, objektivi kryesor do të jetë paraqitja e fizibilitetit të modelit nga pikëpamja e projektimit.

Në kapitullin e mësipërm, ne paraqitëm rolin e mjedisit në sistemet multiagjente dhe diskutuam mbi funksionalitete të rëndësishme që i takojnë mjedisit. Ne paraqitëm mekanizma të përshtatshmërisë për agjentët robotikë. Qëllimi kryesor i një arkitekture të sistemeve multiagjente është të mbështesë projektimin arkitekturor për aplikacione të vetëmanaxhueshme me agjentë. Motivimet kryesore mbi këto arkitektura janë:

- **Integrimi i mekanizmave të ndryshme.** Për të ndërtuar një aplikacion konkret këto mekanizma duhet të punojnë së bashku**.** Arkitektura e projektuar përcakton se si funksionalitete të mekanizmave të ndryshëm vendosen në elementë softuere të agjentëve dhe të mjedisit dhe se si këto elementë ndërveprojnë me njëri tjetrin.
- **Udhërrëfyes për projektimin arkitekturor.** Arkitektura përgjithëson funksione të përbashkëta apo struktura të marra nga aplikacione eksperimentale. Ajo siguron një projekt të ripërdorshëm që lehtëson projektimin e arkitekturave të reja.
- **Përforcim i njohjes dhe ekspertizë.** Këtu është përfshirë një njohje e gjerë dhe ekspertizë e nevojshme që ne kemi fituar gjatë këtij kërkimi.

Proçesi i zhvillimit të arkitekturës

Arkitektura integron funksionalitete të ndryshme të agjentit dhe mjedisit. Mbi këtë arkitekturë, dekompozimi siguron implementimin dhe shtimin e funksionaliteteve duke e berë sistemin të ripërdorshëm. Dokumentimi i arkitekturës nis me një hyrje që përshkruan modelin e integruar të sistemeve multiagjente. Modeli sintetizon funksionalitete të agjentit dhe mjedisit që mbulohen nga akitektura.

Komponentët kryesorë në modulet e arkitekturës janë tre:

- Të dhëna të përbashkëta
- Komponentët bashkëpunues
- Proçeset e komunikimit

Dekompozimi paraqitet në bllokskema të organizuara në bllokskema për nivele të ndryshme dekompozimi. Një bllokskemë e tillë është një përmbledhje informacioni e arkitekturës që përmban qëllimin dhe përshkrim të shkurtër të elementëve dhe marrdhënieve mes tyre. Çdo paketë paraqitjeje konsiston në prezantimin kryesor dhe informacionin shtesë mbështetës. Paraqitjet kryesore tregojnë elementet dhe marrdhëniet e tyre në një paketë. Ato jane grafikë me nje legjendë që shpjegon domethënien e simboleve të përdorur**.**

5.1. Modeli integrues i sistemeve multiagjente

Modeli sintetizon mjedisin dhe agjentët në funksionet e tyre duke e ndarë modelin në dy pjesë: modeli Mjedis dhe modeli Agjent. Le të shohim me rradhë keto dy elemente.

5.1.1. Modeli Mjedis

Mjedisi si model konsiston në nje bashkësi modulesh me rrjedhje të dhënash midis moduleve[80]. Modulet paraqesin funksionalitetet kryesore të mjedisit të treguar në modelin Mjedis. Modeli ka dy module kryesore:

- **Konteksti i vendosjes**. Me këtë term ne i referohemi një harduer i dhënë ose softuer dhe burime të jashtme me të cilat bashkëvepron sistemi multiagjent.(sensor, aktuator, një databazë, një shërbim web, etj).
- **Mjedisi i aplikacionit**: Me këtë term ne i referohemi pjesës së mjedisit që duhet të projektohet për aplikacionin që do të thotë mbi kontekstin e vendosjes. Një shembull i një sistemi multiagjent është sistemi peer to peer. Ky aplikacion zhvillohet në krye të një rrjeti nyjesh me file dhe burime të mundshme. Mjedisi i aplikacionit aftëson agjentët të

aksesojnë në burime të jashtme si dhe mund të sigurojë një infrastrukturë koordinimi të sjelljeve të tyre.

Duke qëndruar në mjedisin e aplikacionit, pjesa e mjedisit është mjaft e rëndësishme të projektohet për një aplikacion bazuar në agjentë softuerë. Modeli i mjedisit të aplikacionit mbulon funksionalitete të ndryshme të mjedisit duke vënë theksin tek dekompozimi si përcaktues i mënyrës se si agjentët ndërveprojnë me mjedisin. Një agjent mund të ndjejë dhe të perceptojë mjedisin me një bashkësi fokusimesh për të marrë paraqitjen e botës rreth tij. Një agjent mund të ketë një influencë në mjedis duke tentuar të modifikojë detyrat e tij në mjedis, si dhe ai mund të këmbejë mesazhe me agjentë të tjerë. Këto ndikime janë të lidhura dhe i dedikohen manipulimit direkt të gjendjes së detyrave në një mjedis duke përjashtuar këmbimin e mesazheve. Ndërveprimi komunikues ndodh në një mënyrë sekuenciale dhe realizon koordinimin e veprimeve midis agjentevë duke aftësuar agjentët të zgjidhin konflikte, të kërkojnë shërbime te njëri tjetri, të marrin vendime për një kooperim të ardhshëm.

Duke e konsideruar perceptimin, veprimin dhe komunikimin si nje mënyrë e veçantë e aksesimit në mjedis, le të shohim modulet e ndryshme funksionale të mjedisit të aplikacionit.

Ruajtja e gjendjes. Ka nje rol kryesor në mjedisin e aplikacionit pasi siguron funksionalitetin e moduleve të tjerë për të aksesuar dhe për të rifreskuar gjendjen e mjedisit të aplikacionit. Gjendja e mjedisit të aplikacionit përfshin zakonisht një abstraksion të kontekstit të zhvillimit dhe një gjendje shtesë të mundshme. Shembuj të një gjendjeje të lidhur me kontekstin e zhvillimit janë paraqitja e topologjisë lokale të nje rrjeti si dhe të dhëna të ardhura nga një bashkësi sensorësh. Gjendja e mjedisit mund të perfshijë të dhëna specifike të agjentëve si identitet të agjentëve, pozicionim me koordinata të tyre, etj.

Gjenerator paraqitjeje. Siguron funksionalitet të agjentëve në perceptimin e mjedisit në menyrë selektive. Funksionet kryesore të gjeneratorit të paraqitjes janë:

- **Kufizim perceptimi**
- **Mbledhje të gjendjeve**
- **Gjenerim paraqitjeje**

Për të ndjerë mjedisin, agjenti thërret një kërkesë perceptimi me një bashkësi fokusimesh. Perceptimi i agjentit është subjekt i ligjeve të percetimit(p-ligje). Korniza e perceptimit aplikon ligjet e perceptimit tek bashkësia e fokusimeve, duke përcaktuar çfarë është në gjendje agjenti të perceptojë. Për shembull një projektues fut kufizime në largesinë e perceptimit për të mos patur një rritje të panevojshme të informacionit mbi mjedisin.

Ndërveprimi. Ka të bëjë me veprimet e agjentëve në mjedis. Për të modeluar veprime, përdorim modelin influencë-riveprim[81]. Ky model bën dallimin midis influencave të cilat janë prodhuar nga agjentët dhe që synojnë të modifikojnë kursin e ngjarjeve në mjedis, si dhe veprimet reaktive, të cilat prodhohen nga mjedisi dhe që rezultojnë nga ndryshimet e gjendjes së mjedisit.

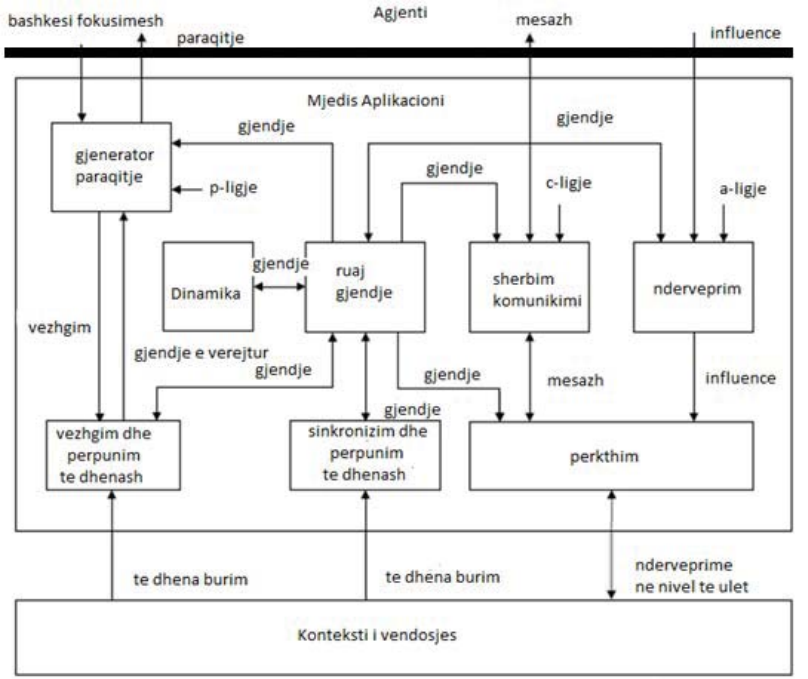

Figura 5. Modeli Mjedis

Funksionet bazë të modulit të ndërveprimit janë:

- **Mbledhës influencash**. Mbledh influencat të sjella nga agjentët. Këto influenca mund t'i ndajmë në influenca që tentojnë të modifikojne gjendjen e mjedisit të aplikacionit dhe influenca që tentojnë të ndryshojnë gjendjen e kontekstit të zhvillimit të aplikacionit. Shembull është një agjent që tenton të shkruaj në një bazë të dhënash. Influencat janë subjekt i ligjeve të veprimit (a-ligje).
- **Kufizues veprimi.** Ligjet e veprimit vendosin kufizime mbi influencat e sjella nga agjentët. Kufizimi i veprimeve aplikon një bashkësi ligjesh të veprimit në influencat e agjentëve.

- **Veprues.** Aplikon me efiçencë influencat e sjella nga agjentët. Për ato influenca që lidhen me mjedisin e aplikacionit, reaktori përllogarit reaksionin e influencave duke rezultuar në një rifreskim të gjendjes së mjedisit të aplikacionit. Ndërsa influencat që lidhen me kontekstin e zhvillimit të aplikacionit, kalohen te moduli perkthim që mbulon influencat në nivel të lartë të agjentëve në veprimet me nivelin e ulët të tyre në kontekstin e vendosjes.

Shërbimi i komunikimit. Merret me këmbimin e mesazheve në sistemin me agjentë. Funksionalitetet kryesore të shërbimit të komunikimit janë ndërtimi i mesazhit dhe shpërndarja e mesazhit. I pari rregullon këmbimin e mesazheve midis agjentëve sipas gjendjes prezente të mjedisit të aplikacionit dhe një bashkësie të ligjeve të aplikueshme. Ligjet e komunikimit vendosin rregulla dhe kufizime në këmbimin e mesazheve. Për shembul vendosin rregulla për vendosjen e prioriteteve dhe trajtimin në bazë të hierarkisë. Ky shërbim kalon mesazhet në modulin e përkthimit i cili trajton mesazhet të përshkruar në një ACL (**agjent communication language**)[82] në primitiva të kontekstit. Përkthimi konverton mesazhet e ardhura nga konteksti në përshkrime mesazhi të nivelit të lartë dhe i kalon ato në pjesën e shpërndarjes së mesazheve duke i nisur ato në agjentët përkatës.

Sinkronizimi dhe përpunimi i të dhënave. Kontrollon pjesë specifike të kontekstit të zhvillimit dhe mban paraqitjen koresponduese të gjendjes së mjedsit të aplikacionit të rifreskuar. Shembull është topologjia e nje rrjeti dinamik ku ndryshimet reflektohen në një abstraksion rrjeti të mbajtur në mjedisin e aplikacionit**.** Moduli i sinkronizimit zakonisht parapërpunon të dhënat të ardhura nga konteksti i aplikacionit, përpara se ato të kalojnë në modulin e mbajtjes së gjendjes.

Dinamika. Mban ngjarjet në mjedisin e aplikacionit që ndodhin në mënyrë të pavarur nga agjentët dhe konteksti i zhvillimit të aplikacionit**.** Mund të vendoset një kufizim në kohën e mbajtjes së një grupi ngjarjesh, sipas interesave të agjentit**.**

5.1.2. Modeli i Agjentit

Le të shohim tani modelin e agjentit në këndveshtrimin e funksionaliteteve të tij.

Modeli konsiston në katër komponentë bazë si dhe lidhjet midis tyre[83]. Ne figurën 5.2 paraqitet bllokskema e modelit të agjentit ku modulet paraqesin funksionalitetet bazë të agjentit. Katër modulet sigurojnë funksionalitetet e kërkuara nga agjenti për mekanizmat e përshtatjes përfshirë këtu perceptitimin selektiv, përzgjedhjen e veprimit me rolet, angazhimet për vendndodhjen si dhe komunikimin e bazuar në protokolle.

Le të shohim secilin modul**.**

Integrimi i njohjes. Siguron funksionin e të aksesuarit dhe rifreskimit të njohjes prezente të agjentit. Moduli i perceptimit merr kërkesa perceptimi nga modulet Vendim marrje dhe Komunikim për të rifreskuar njohjen e agjentit rreth mjedisit. Këto dy module përdorin njohjen prezente të agjentit për të marrë vendimet e duhura. Akoma më tej, këto module shfrytezojnë njohjen prezente si një mjet për koordinim. Gjate proçesit të bashkëpunimit, moduli Komunikim ndryshon njohjen prezente të agjentit të varur nga Ndërveprimi komunikues. Kur vendoset një komunikim, moduli Komunikim aktivizon një tjetër pritshmëri që lidhet me zgjedhjen e veprimit të agjentit në modulin VendimMarrje. Kjo vazhdon deri sa pritshmëria çaktivizohet dhe bashëkpunimi merr fund. Dallohen dy raste:

- Pritshmëria çaktivizohet për shkak të ndryshimeve të perceptuara në mjedis
- Fundi i pritshmërisë komunikohet në mënyrë eksplicite midis agjentëve bashkëpunues.

Shembull i koordinimit midis moduleve VendimMarrje dhe Komunikim është një agjent që kërkon nje tjetër agjent për informacion të nevojshem për të kompletuar detyrën.

Perceptimi siguron funksionin e agjentit për perceptim selektiv. Tre janë funksionet bazë të modulit:

- **Sensimi.** Aftëson agjentin të ndjejë mjedisin me një bashkësi fokusimesh. Fokusimet bëjnë që agjenti të sensojë mjedisin për tipe specifike të mjedisit.
- **Përkthimi .** Sensimi përfundon në një paraqitje. Një paraqitje është një strukturë të dhënash që paraqet elemente në mjedis. Përkthimi bën të mundur krijimin e nje harte të perceptimeve të bëra.
- **Filtrimi** përdor një bashkësi filtrash për të zgjedhur ato objekte të dhënash të një perceptimi që përputhen me kriterin e selektimit të vendosur në filtrat. Perceptimi i filtruar përdoret nga moduli i integrimit të njohjes, për të rifreskuar njohjen prezente të agjentit.

VendimMarrje siguron funksionin e agjentit për të selektuar dhe krijuar influenca në mjedis. Ky modul ka dy funksione bazë:

- Selektimin e influencave
- Ekzekutimin e detyrave

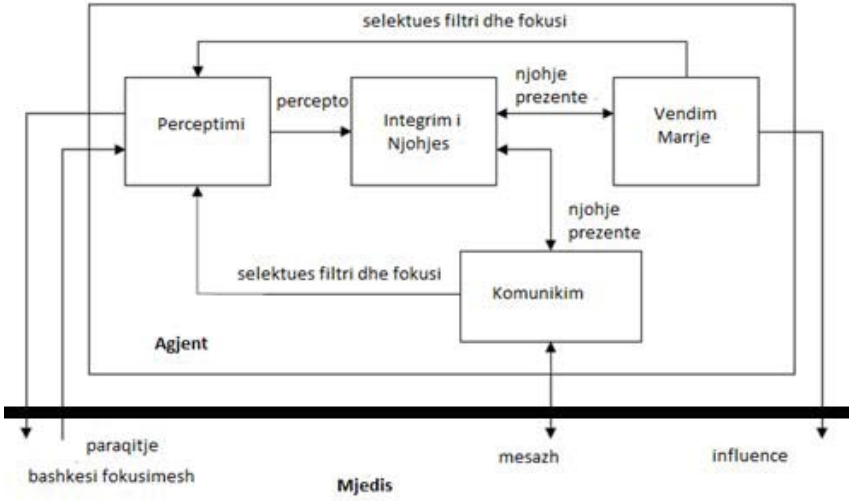

Figura 5. Modeli Agjent

Për të selektuar influencat e përshtatshme, një agjent robotik përdor mekanizëm selektimi të veprimit bazuar në sjellje, me rolet dhe pritshmërite e veta. Ekzekutimi siguron funksionin për të sjellë influencat e zgjedhura në mjedis.

Komunikimi siguron për agjentin funksionin e këmbimit të mesazheve me agjentë të tjerë sipas protokolleve të komunikimit, të përcaktuar qartë. Komunikimi konsiston në tre funksione bazë:

- **Dekodim mesazhi**. Ruan mesazhet që hyjnë në një buffer dhe i dekodon ato një nga një. Nxjerr informacionin nga mesazhet sipas ACL, të dhënat e dekoduara të mesazhit përshkruajnë informacionin e nxjerrë në një format të kuptueshëm nga agjenti.
- **Komunikim**. Ka nje funksion të dyfishtë, atë të interpretimit të informacionit të dekoduar dhe riveprimin në menyrë të përshtatshme, si dhe atë të nisjes dhe vazhdimit të bashkëbisedimit kur është e nevojshme, në bazë të kushteve prezente. Për të arritur këtë, komunikimi përdor një kujtesë të protokolleve të komunikimit, njohjen prezente të agjentit dhe një ontologji. Kjo e fundit përcakton një fjalor fjalësh që agjentët përdorin për të paraqitur konceptet e fushës dhe marrdhëniet midis koncepteve .
- **Enkodim mesazhi.** Aftëson agjentët të enkodojnë sërish një mesazh për ta kaluar atë më pas në sistemin e shpërndarjes së mesazhit. Moduli ka edhe një buffer që parashikon dhe zgjidh problemin e vonesave.

KAPITULLI 6

Projektimi Arkitekturor i Sistemeve Vetëmanaxhuese

Dekompozimi në module

Paraqitja e dekompozimit në module do të jap një sistem të shpërbërë në njësi të manaxhueshme softuere. Elementi përbërës i një dekompozimi quhet modul.

Në inxhinierinë e soft-it, një **modul** është një njësi softuere që siguron funksionalitetin e një sistemi. Kriteri bazë për dekompozimin është arritja e atributeve të cilësisë. Modulet në një dekompozim arkitekturor përfshijnë një përshkrim të ndërfaqeve të moduleve që dokumenton sesi moduli kombinon me modulet e tjerë. Përshkrimi i ndërfaqes jep dy ndërfaqe:

- Ndërfaqe shërbimi që specifikon se çfarë funksioni agjenti u ofron agjentëve të tjerë.
- Ndërfaqe e kërkuar specifikon çfarë funksioni ka nevojë moduli nga modulet e tjerë.

Modeli arkitekturor që sjellim në dekompozim përbëhet nga tre shkallë dekompozimi kryesore. Le të shohim nivelin më të lartë të dekompozimit për një sistem bazuar në agjent.

6.1.1. Niveli i parë i dekompozimit të sistemit multiagjent

Në këtë shkallë të parë, dekompozimi nënsistemet kryesore do të jenë dy, sikurse është paraqitur në figurën 14.

Agjenti. Agjenti është një njësi autonome brenda sistemit që kryen dhe zgjidh detyra. Agjenti enkapsulon gjendjen e vet dhe kontrollon sjelljet e veta. Përgjegjësia e një agjenti është të arrijë objektivat për të cilat është projektuar. Agjentët janë vendosur në një mjedis të cilin mund ta perceptojnë dhe në të cilin ata ndërveprojnë dhe veprojnë. Agjentët janë të aftë të përshtasin sjelljen e tyre sipas kushteve të ndryshueshme të mjedisit.

Mjedisi i aplikacionit. Është mjeti apo ndërmjetësi që lejon agjentët të ndajnë informacionin dhe të koordinojnë sjelljen e tyre. Përgjegjësitë bazë të mjedisit të aplikacionit janë:

- Të sigurojë akses tek entitetet e jashtme dhe burimet.
- Të aftësojë agjentët të perceptojnë dhe të manipulojnë fqinjët dhe të ndërveprojnë me ta

- Të ndërmjetësojë veprimtaritë e agjentëve dhe si ndërmediator mjedisi jo vetëm aftëson perceptimin, veprimin dhe ndërveprimin por edhe i detyron ato.

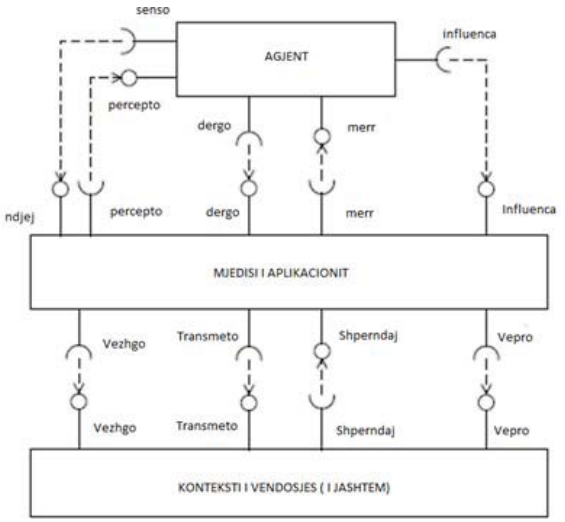

Figura 6. Dekompozimi në nivel të parë dhe ndërfaqet midis nënsistemeve

Mjedisi i aplikacionit është pjesë e mjedisit që duhet të projektohet për një aplikacion bazuar në sisteme multiagjente. Entitete të jashtme dhe burime me të cilat sistemi multiagjent ndërvepron janë pjesë e kontekstit të zhvillimit. Struktura e brendshme e kontekstit të zhvillimit dhe funksionimi i saj nuk është marrë parasysh në këtë punim.

6.1.2. Ndërfaqësimi midis moduleve

Në figurën 6.1 janë paraqitur edhe ndërfaqet e agjentit, e mjedisit të aplikacionit dhe kontekstit të zhvillimit. Ndërfaqja *Senso* aftëson agjentët të ndjejnë mjedisin , *Dërgo* aftëson një agjent të dërgojë mesazhe tek agjent të tjerë dhe *Influence* aftëson agjentët të sjellin influenca në mjedis. Këto ndërfaqe sigurohen nga mjedisi i aplikacionit.

Mjedisi i aplikacionit kërkon ndërfaqen *Percepto* për të kaluar në paraqitje të agjentit dhe ndërfaqe *Marr* për të shpërndarë mesazhe. Akoma më tej mjedisi i aplikacionit kërkon ndërfaqen *Vëzhgo* nga konteksti i vendosjes për të vërejtur burime të veçanta bazuar në kërkesat për perceptim të agjentëve. *Transmeto* për të derguar mesazhe dhe *Vepro* për të modifikuar gjendjen e burimeve të jashtme.

Ndërsa në fund, konteksti i zhvillimit kërkon ndërfaqen *Shpërnda* nga mjedisi i aplikacionit për të shpërndarë mesazhet e ardhura tek agjentët.

Mekanizmat e variancës

Ekzistojnë tre mekanizma variance për këtë paraqitje.

Përcaktimi i tipit të agjentit. Kemi parasysh që në një sistem multiagjent të aplikuar duhet të kemi parasysh që integrohen agjentë të tipeve të ndryshëm. Kështu agjentë të të njëjtit tip natyrisht që kanë kapacitete të njëjtë dhe vendosen në sisteme të tilla për të realizuar pothuaj të njëjtat objektiva. Si rrjedhim, secili tip agjenti do të ketë një arkitekturë specifike në strukturën e vet. Përsa i takon ndryshimeve në strukturën e brendshme, këtë aspekt do ta hasim më poshtë, në trajtime të mëtejshme.

Përcaktimi i ontologjisë së fushës. Ontologjia përcakton terminologjinë për fushën e aplikacionit. Përcaktimi i ontologjisë përfshin specifikimin e koncepteve të fushave të ndryshme si dhe marrëdheniet midis tyre. Ontologjia e fushës shërben për përcaktimin e një bazë njohjeje të agjentëve dhe të gjendjes së tyre në mjedisin e aplikacionit.

Përcaktimi i primitivave të ndërveprimit për kontekstin e vendosjes. Për të aftësuar një sistem multiagjent që të ndërveprojë me këtë kontekst, primitivat e ndërveprimit me kontekstin e vendosjes duhet të konkretizohen sipas aplikacionit ku zhvillohen. Dallohen tre tipa të primitivave të ndërveprimit:

- Primitivat e vëzhgimit aftësojne vëzhgimin e burimeve të kontekstit të hapjes. Ky lloj primitivi tregon se cili burim vëzhgohet dhe çfarë tip informacioni duhet mbledhur.
- Primitivat e veprimit aftësojnë aksesimin në burimet e kontekstit. Ky lloj primitivi tregon burimin target si dhe tipin e veprimit.
- Primitivat e komunikimit aftësojnë transmetimin e mesazheve në nivel të ulët nëpërmjet mjedisit të hapjes. Një mesazh i formatuar në nivel të ulët është një strukturë të dhënash që paraqet një mesazh të këmbyer midis një dërguesi ose shumë adresash dhe që mund të transmetohet nëpërmjet konteksit të hapjes.

Projektimi logjik

Ka disa parime mbi të cilat mbështetet dekompozimi i sistemeve multiagjente.

Kontrolli i decentralizuar. Në këto sisteme, kontrolli ndahet midis agjentëve që veprojnë në të njëjtin mjedis. Kontrolli i decentralizuar është thelbësor për të përballuar lokalizimin e pandarë të aktivitetit të sistemit, e cila është një karakteristikë e aplikacioneve të synuara të arkitekturës multiagjente.

Vetë-manaxhimi. Në një sistem multiagjent, vetë-manaxhimi është i bazuar kryesisht në aftësinë e agjentëve për të përshtatur sjelljen e tyre. Vetë-manaxhimi i mundëson një sistemi të manaxhojë kushtet dinamike dhe ndryshimet në veprimet autonome, e cila është dhe kjo një kërkesë e rëndësishme e aplikacioneve të synuara të arkitekturës multiagjente.

Megjithatë, arkitektura e decentralizuar e një sistemi multiagjent nënkupton një numër shmangjesh dhe kufizimesh:

- Kontrolli i decentralizuar zakonisht kërkon më shumë komunikim. Performanca e sistemit mund të ndikohet nga lidhjet e komunikimit mes agjentëve.
- Ka një konkurrencë mes performancës së sistemit dhe fleksibilitetit të saj për të trajtuar situata problematike. Një sistem, që është projektuar për të përballuar më shumë probleme, në përgjithësi ka nevojë për redundancë, zakonisht në dëm të performancës, dhe anasjelltas.
- Marrje e vendimeve nga agjentët është bazuar vetëm në informacion lokal, i cili mund të çojë në sjellje jo optimale të sistemit.

Këto kontradikta dhe kufizime duhet të mbahen mend në të gjithë hartimin dhe zhvillimin e një sistemi multiagjent. Vëmendje të veçantë duhet të ketë komunikimi i cili mund të imponojë një fenomen të qafës së shishes.

Dekompozimi modular në nivelin Agjent

Konsiderohet fillimisht nënsistemi Agjent i cili shihet në tre module:

1. Perceptimi
2. Vendim marrje
3. Komunikimi

Modulet mbulojnë funksionalitetin e moduleve me të njëjtin emër në modulin e integruar të dhënë më sipër. Le të diskutojmë për secilin modul.

Perceptimi. Është përgjegjës për të mbledhur informacion në kohën e ekszekutimit nga mjedisi i aplikimit dhe konteksti i vendosjes. Moduli i perceptimit mbështet perceptimin selektiv [84]. Perceptimi selektiv bën që një agjent të drejtojë perceptimin e vetë sipas detyrës prezente të tij. Fokusimet lejojnë që agjenti të ndjejë mjedisin vetëm për tipe specifike të informacionit. Ndjeshmëria është e pranishme në një paraqitje të mjedisit të perceptuar. Një paraqitje është një strukturë të dhënash që paraqet elemente burime në një mjedis. Moduli i perceptimit mapon këtë paraqitje të një perceptim që do të thotë përshkrim i mjedisit të ndjesuar në një formë të elementëve të dhëna që mund të perdoren për të rifreskuar njohjen prezente të

agjentit. Bashkësia e selektuar e filtrave akoma më tej zvogëlon perceptimin sipas një kriteri të specifikuar nga filtrat.

Vendim Marrje. Ky modul është përgjegjës për selektimin e veprimit. Modeli i veprimit i arkitekturës bazë bazohet në modelin influence – reaction të dhënë në [85]. Ky model dallon influencat që janë prodhuar nga agjentët dhe tentojnë të ndryshojnë rrjedhën e ngjarjeve në një mjedis si dhe reagimet që rezultojnë në ndryshimet e gjendjes së mjedisit. Përgjegjësia e modulit Vendim marrje është të selektojë influenca për të realizuar detyrat e agjentit dhe të thërrasë influenca në mjedis. Për të aftësuar agjentët të vendosin bashkëpunime, mekanizma selektimi bazuar në sjellje janë integruar me rolin përkatës. Një rol paraqet një pjesë koherente e funksionalitetit të agjentit në kontekstin e një agjensie[86]. Një angazhim i agjentit ka të bëjë me dhënien e preferencës veprimeve me një rol të veçantë në një gjendje të tillë. Agjentët në menyrë tipike angazhohen me njëri-tjetrin në një bashkëpunim por një agjent mund të kryejë veprime mbi veten e vet kur një e tyre kryesore është realizuar dhe kompletuar. Rolet dhe punët kanë një emër të mirënjohur i cili është pjesë e ontologjisë së domenit si dhe ndahet mes agjentëve të sistemit. Ndarja e këtyre emrave aftëson agjentët të vendosin bashkëpunime nëpërmjet këmbimit të mesazheve.

Komunikimi. Është përgjegjës për ndërveprime komunikimi me agjentët e tjerë. Moduli i komunikimit përpunon mesazhe të ardhura dhe prodhon mesazhe daljeje sipas protokolleve të komunikimit të përcaktuara mirë dhe qartë[87]. Një protokoll komunikimi specifikon një bashkësi sekuencash të mundshme të mesazheve. Një bashkëbisedim i referohet një ndërveprimi komunikativ në vazhdim. Një bashkëbisedim inicializohet nga një mesazh fillestar i protokollit të komunikimit. Në secilën fazë në një bashkëbisedim , ka një numër të kufizuar mesazhesh që mund të këmbehen. Gjendjet fundore përcaktojnë kur një bashkëbisedim shkon drejt fundit.

Një informacion i këmbyer nëpërmjet një mesazhi enkodohet nëpërmjet një gjuhe të përbashkët komunikimi. Kjo gjuhë përcakton formatin e mesazheve siç janë fushat që e përbëjnë atë. Një mesazh përfshin një fushë me një identifikues unik të bashkëbisedimit në vazhdim, më tej janë fushat që mbajnë identifikimin e dërguesit, fusha me fjalën që identifikon një veprim të caktuar, fusha me përmbajtjen e mesazhit.

Përshkrimi i Ndërfaqes

Përshkrimi i ndërfaqes specifikon se si modulet e një agjenti përdoren së bashku. Figura 6.2 paraqet projektimin arkitekuror të ndërfaqjes së modulit Agjent. Ndërfaqja *Kërkesë* realizon ndërfaqe të modulit të percepimit duke aftësuar Vendim marrje dhe Komunikim të kërkojnë një perceptim të mjedisit. Të ndjesh mjedisin sipas nevojave prezente të tyre , Vendim marrje dhe komunikimi kalojnë në një fokus dhe selektor filtri te moduli i perceptimit. Selektori specifikon një bashkësi

fokusimesh dhe filtrash të cilat i përdorë moduli i perceptimit për të ndjerë mjedisin në mënyrë selektive.

Ndërfaqet *Percepto* dhe *Marr* delegojnë për përpunim tek moduli i perceptimit dhe i komunikimit në mënyrë respektive. Moduli i perceptimit kërkon ndërfaqen *Senso* që i delegohet ndërfaqes *Senso*. Në mënyrë të ngjashme, ndërfaqja *Dërgo* e modulit të komunikimit si dhe ndërfaqja *Influencë* për modulin Vendi marrje i delegohen ndërfaqeve të kërkuara me të njëjtin emër.

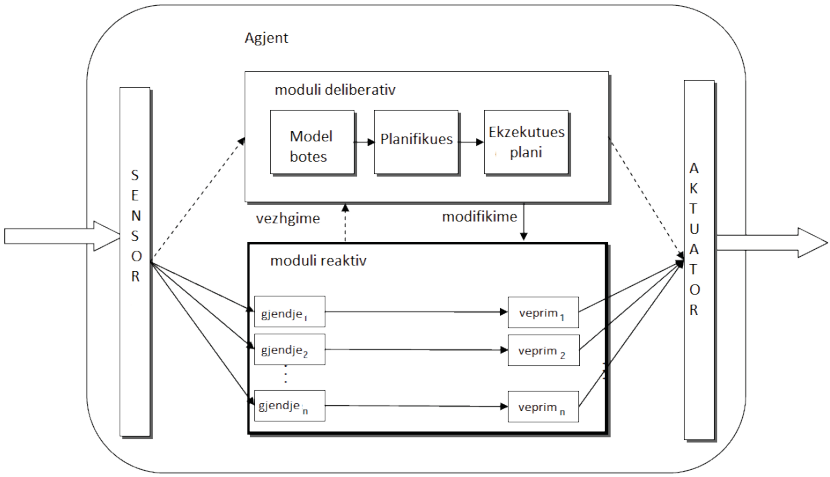

Figura 6. Projektimi arkitekuror i modulit Agjent

Mekanizmat e variancës.

Kjo paraqitje siguron mekanizma të variancës si më poshtë:

Largim i modulit të komunikimit. Për agjentët që nuk komunikojnë me këmbim mesazhe, moduli i komunikimit mund të hiqet. Shembull i tillë është një sistem multiagjent ku agjentët nuk komunikojnë pasi manipulojnë vetëm shenjat në mjedisin ku veprojnë.

Përcaktimi i fokusimeve dhe selektorëve të fokusit. Fokusimi aftëson agjentët të ndjejnë mjedisin në mënyrë selektive. Përcaktimi i fokusit në një agjent përfshin specifikimin atyre të dhënave për çdo target fokusi së bashku me veçoritë e secilit focus. Përcaktimi i selektorëve të fokusit përfshin specifikimin e kombinimeve të ndryshme të fokusimeve që mund të përdoren për të ndjerë mjedisin.

Përcaktimi i paraqitjeve. Në këto prezantime rezulton ndjerja e mjedisit. Paraqitjet përcaktohen nga struktura të dhënash që përfaqësojnë elemente dhe burime në mjedis. Përkufizimi i paraqitjeve duhet të përmbushë ontologjinë për domenin e përcaktuar.

Përcaktimi i filtrave dhe selektorëve të filtrave. Agjentët i përdorin për të filtruar të dhënat e përceptuara. Përkufizimi i filtrave në një sistem me agjent përfshin specifikimin e të dhënave për çdo filtër si dhe veçoritë specifike të secilit filtër. Përkufizimi i selektorëve të filtrave përfshin specifikimin e kombinimeve të ndryshme të filtrave që mund të përdoren për të filtruar perceptimet.

Përcaktimi i influencave. Influencat aftësojnë agjentët të modifikojnë gjendjen e punëve në mjedis. Përcaktimi i influencave përfshin specifikimin e një veprimi që sigurohet nga mjedisi i aplikacionit dhe që mund të thirret nga agjentët.

Përcaktimi i roleve dhe detyrave. Secili rol në një sistem me agjent përcaktohet nga një emër i vetëm dhe një përshkrim i semantikës së rolit në terma të influencave që mund të zgjidhen në këtë rol si dhe marrëdheniet e rolit me rolet e tjerë në këtë sistem agjentësh. Çdo detyrë gjithashtu përcaktohet qartë nga një emër sikurse dhe përshkrimi i semantikës së detyrave në sistemin e agjentëve.

Përcaktimi i gjuhës së komunikimit dhe ontologjisë. Gjuha e komunikimit përcakton formatin e mesazheve. Në përshkrim përfshihen specifikime të identiteteve të agjentëve dhe të komunikimeve , formate të përmbajtjes së mesazheve, etj. Ontologjia për komunikimin është një pjesë që përcakton fjalorin e fjalëve që përfaqësojnë konceptet e domenit.

Modeli Logjik

Çdo modul në dekompozim enkapsulon një funksionalitet të veçantë të agjentit. Duke minimizuar mbivendosjen e funksionaliteteve midis moduleve , ne mund të fokusohemi në një aspekt të veçantë të funksionalitetit të një agjenti. Duke përcaktuar funksionalitete të ndryshme të agjentit për të ndarë module arrihet në një projektim të qartë. Kjo ndihmon për të futur ndryshime dhe për të rifreskuar një modul pa prekur të tjerët si dhe kjo bën që të mbështetet ripërdorueshmëria.

Perceptimi selektiv. Aftëson një agjent për t'u fokusuar mbi aspekte të rëndësishme në një mjedis sipas detyrës së tij. Kur veprimet selektuese dhe mesazhet e komunikimit me agjentët e tjerë , marrja e vendimeve dhe komunikimi kërkojnë perceptime për të rifreskuar njohjen e agjentit rreth mjedisit. Duke selektuar një set të përshtatshëm filtrash , agjenti drejton vëmendjen tek aspektet e interesit të tij. , dhe përshtat vëmendjen e tij kur kushtet e veprimtarisë ndryshojnë.

Koordinimi midis vendim-marrje dhe komunikim. Sjellja e përgjithshme e agjentit është rezultat i koordinimit të dy moduleve :

1. Vendim-marrje
2. Komunikim

Vendim marrje është përgjegjës për zgjedhjen e influencave për të vepruar në një mjedis. Ndërsa komunikimi është përgjegjës për ndërveprime komunikative me agjentët e tjerë. Megjithatë të dy modulet nuk veprojnë në mënyrë të pavarur. Këto module koordinojnë për të kompletuar detyrat në mënyrë eficente. Për shembull agjentët mund të dërgojnë mesazhe tek njëri-tjetri me kërkesa për informacion që u lejon atyre të veprojnë në mënyrë më të qëllimshme. Vendim marrje dhe komunikimi koordinojnë gjatë proçesit të bashkëpunimit. Bashkëpunimet vendosen zakonisht nëpërmjet shkëmbimit të mesazheve duke përcaktuar angazhim kur arrihet një marrëveshje. Ky angazhim do të provokojë Vendim marrjen drejt veprimeve të tjera derisa ai të çaktivizohet dhe bashkëveprimi përfundon.

Ekzistojnë dy avantazhe të veçantë në projektimin e komunikimit të decentralizuar nga kryerja e veprimeve:

1. Lejon funksione të ekzekutohen në paralel.
2. Lejon funksionet të ekzekutohen me ritëm të ndryshëm.

Në mjaft aplikacione, dërgimi i mesazheve dhe ekzekutimi i veprimeve ndodh në kohë të ndryshme. Një shembull tipik janë robotët, por ka dhe aplikacione të tjerë. Kjo ndarje i aftëson agjentët të rikonsiderojnë koordinimin e sjelljes së tyre, ndërkohë që ata realizojnë veprime duke përmirësuar përshtatshmërinë dhe efiçencën.

Dekompozimi modular i mjedisit të aplikacionit

6.1.3. Modulet përbërëse dhe veçoritë kryesore të tyre

Mjedisi i aplikacionit ndahet në 7 module kryesore. Modulet mbulojnë funksionalitetin e moduleve me të njëjtin emër në modelin e integruar për mjedisin e aplikacionit. Le të shohim me vëmendje secilin modul përbërës.

Pamja e dekompozimit në module për mjedisin e aplikacionit do të përbehet nga:

1. **Gjeneratori i paraqitjes**. Siguron funksionalitetin e perceptimit të mjedisit nga agjenti. Kur një agjent ndjen mjedisin, gjeneratori e prezantimit përdorë gjendjen ekzistuese të mjedisit të aplikacionit dhe gjendjen e mundshme të mbledhur nga konteksti i vendndodhjes për të prodhuar një paraqitje për agjentin. Perceptimi i agjentit është subjekt i ligjeve të perceptimit që siguron një domethënie të perceptimit të kufizuar. Një ligj perceptimi përcakton kufizime në atë që çfarë mund të sensojë një agjent nga një mjedis me një bashkësi fokusimesh.
2. **Vëzhgim dhe përpunim të dhënash**. Siguron funksionalitetin për të vëzhguar kontekstin e vendndodhjes. Ky modul përkthen kërkesat e vëzhgimit në primitiva vëzhgimi që mund të përdoren për të mbledhur të

dhënat e kerkuara nga konteksti i vendndodhjes. Këto të dhëna të mbledhura nga burime të kontekstit i kthehen kërkuesit për shembull gjeneratorit të paraqitjes. Për më tepër se dërgimit të të dhënave të papërpunuara marrë nga burimet në kontekstin e vendosjes,vëzhgimi i të dhënave dhe përpunimi i tyre si modul mund të sigurojë funksione shtesë për parapërpunim të të dhënave, integrim të të dhënave nga sensorët, etj. -

3. **Ndërveprim**. Është përgjegjës për të trajtuar ndikimet e agjentëve në mjedis. Këto ndikime mund të ndahen në dy kategori kryesore:
    a. Ndikime që tentojnë të modifikojnë gjendjen e mjedisit të aplikacionit.
    b. Ndikime që tentojnë të modifikojnë gjendjen e burimeve të konteksit të vendosjes.

Ndikimet e agjentëve janë subjekt i ligjeve të veprimit që paraqesin kufizime specifike në ndikimet e agjentëve. Për ndikime që lidhen me mjedisin e aplikacionit, moduli i ndërveprimit llogaritë reaksionin e ndikimeve nga rifreskimi i gjendjes së mjedisit të aplikacionit. Ndikimet e lidhura me kontekstin e vendndodhjes kalojnë në një modul përkthimi që konverton ndikimet nga agjentët në primitiva veprimi të nivelit të ulët në kontekstin e vendndodhjes.

4. **Shërbime komunikimi**. Është përgjegjës për mbledhjen e mesazheve, siguron infrastrukturën e nevojshme për ruajtjen e mesazheve si dhe shpërndarjen e tyre tek agjenti i duhur. Ky modul rregullon këmbimin e mesazheve midis agjentëve sipas një seti ligjesh të komunikimit të aplikueshme. Ligjet e komunikimit vendosin kufizime në mesazhet ose detyrojnë ligje dhe rregulla specifike në këmbimin e tyre. Për të transmetuar mesazhet, shërbimi i komunikimit përdor sistemin e transferimit të tyre të siguruar nga konteksti i ndodhjes. Moduli i shërbimit të komunikimit përdor modulin e përkthimit për të konvertuar përshkrimet e mesazheve në nivel të lartë në primitiva komunikimi të nivelit të ulët për kontekstin e vendndodhjes.

5. **Përkthim**. Realizon një urë lidhëse midis ndikimeve dhe përshkrimeve të mesazheve të përdorur nga agjentët dhe veprimeve përkatëse dhe primitivave të komunikimit të kontekstit të vendndodhjes. Përkthimi siguron një funksionalitet të dyfishtë. Ai përkthen ndikimet në primitiva të nivelit të ulët të veprimeve me kontekstin e vendndodhjes.Përkthen mesazhe ACL në mesazhe të formatuara në nivel të ulët që mund të transmetohen nëpërmjet kontekstit të ndodhjes.

6. **Sinkronizim dhe përpunim të dhënash.** Monitoron pjesë specifike të kontekstit të ndodhjes dhe mban të rifreskuar paraqitjen koresponduese

në gjendjen e mjedisit të aplikacionit. Moduli konverton të dhënat nga burimet , të vëzhguara në kontekstin e ndodhjes në një format që mund të përdoret për të rifreskuar gjendjen e mjedisit të aplikacionit.

7. **Dinamika.** Është përgjegjëse për proçeset e mirëmbajtjes në mjedisin e aplikacionit që ndodhin në mënyrë të pavarur nga agjentët dhe konteksti i ndodhjes. Moduli i dinamikës akseson në mënyrë direkte gjendjen e mjedisit dhe mban gjendjen sipas përcaktimit specifik të aplikacionit.

6.1.4. Përshkrimi i ndërfaqes

Përshkrimi i ndërfaqes tregohet në figurën 6.3 dhe paraqet se si modulet e mjedisit të aplikacionit përdoren së bashku duke ndërvepruar edhe me bazën e të dhënave të mjedisit të aplikacionit.

Ndërfaqja *Senso* e mjedisit të aplikimit dërgon kërkesat e perceptimit te ndërfaqja *Senso* e gjeneratorit të perceptimit. Për të vëzhguar burimet në kontekstin e vendosjes, ndërfaqja *Mbledh* e gjeneratorit të përceptimit varet nga ndërfaqja *Mbledh* e modulit Vëzhgim dhe Përpunim të dhënash. Ndërfaqja e kërkuar *Vëzhgo* e modulit Vëzhgo dhe përpunim të dhënash i referohet ndërfaqes *Vëzhgo* të mjedisit të aplikimit. Të dhënat që rezultojnë nga vëzhgimi i burimeve në kontekstin e aplikimit përpunohen nga moduli *Vëzhgim* dhe përpunim të dhënash. Ndërfaqja *gjenero–vëzhgim* dhe përpunim të dhënash përdorë një prezantim për agjentin që ka bërë kërkesën bazuar në të dhënat e përpunuara. *Vëzhgo* është **e** vetmja ndërfaqe e kërkuar nga muduli *Sinkronizim dhe përpunim të dhënash* për funksionimin e mjedisit të aplikacionit. Përpunimi i kësaj ndërfaqe i drejtohet ndërfaqes *Vëzhgo* të mjedisit të aplikacionit. Ndërfaqja *Dërgo* e mjedisit të aplikacionit aftëson agjentët të dërgojnë mesazhe tek njëri-tjetri. Mjedisi i

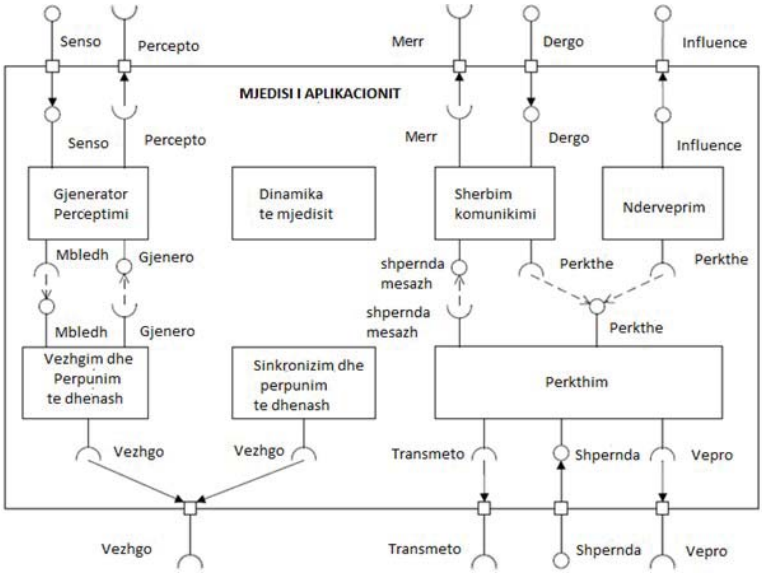

Figura 6. Ndërfaqet e moduleve të mjedisit të aplikacionit.

aplikacionit i drejton këtë ndërfaqe tek ndërfaqja *Dërgo* e shërbimit të komunikimit. Për të konvertuar mesazhet në një nivel më të ulët formati për transmetim nëpërmjet kontekstit të aplikacionit, ndërfaqja *Përkthim* e shërbimit të komunikimit i drejtohet ndërfaqes *Përkthim* të siguruar nga moduli i përkthimit.

Ndërfaqja *Transmeto* e modulit të përkthimit drejton transmetimin e mesazheve te nderfaqja *Transmeto* e mjedisit të aplikacionit. Mjedisi i aplikacionit ofron ndërfaqe *Shpërnda* për të dorëzuar mesazhet hyrëse.

Ndërfaqja *Shpërnda* e mjedisit të aplikacionit drejton mesazhet hyrëse te ndërfaqja *Shpërnda* të modulit të përkthimit. Përkthimi konverton mesazhet në një format të përshtatshëm për agjentët dhe përdorë ndërfaqen *Shpërnda* mes të shërbimit të komunikimit për të dorezuar mesazhe. Ndërfaqja Marr e shërbimit të komunikimit drejton dhënien e mesazheve për te ndërfaqja e mjedisit të aplikimit që kalon mesazhet për te adresat.

Ndërfaqja *Influence* e mjedisit të aplikacionit mundëson agjentët të kërkojnë ndikimet në mjedis. Për ndikimet që tentojnë të modifikojnë gjendjen e burimeve në kontekstin e vendosjes, ndërfaqja e modulit të kërkuar të ndërveprimit *Përkthe* varet nga ndërfaqe *Përkthe*, e siguruar nga moduli i përkthimit. Kjo ndërfaqe ofron funksionalitet për të kthyer ndikimet në nivel të ulët të primitivës në kontekstin e vendosjes. Ndërfaqja *Vepro* i delegon modulit të përkthimit veprimet në burimet e jashtme në ndërfaqe *Vepro* të mjedisit të aplikimit që thërret veprimet në kontekstin e vendosjes.

Mekanizmat e variancës

Ky nënsistem siguron këto mekanizma variance.

Lëshimi i vëzhgimit, sinkronizimit dhe përkthimit. Për aplikimet që nuk ndërveprojnë me burime të jashtme , vëzhgimi, sinkronizimi dhe përkthimi si module mund të hiqen. Në këto aplikime , mjedisi është tërësisht virtual.

Lëshimi i shërbimit të komunikimit. Për sistemet agjente në të cilat agjentët nuk komunikojnë nëpërmjet këmbimit të mesazheve, moduli komunikimit mund të hiqet.

Lëshimi i dinamikave. Për aplikimet me sisteme me multiagjent me mjedis aplikimi që nuk duhet të mbajnë dinamika të pavarura nga agjentët, moduli i dinamikave mund të hiqet.

Përkufizimi i vëzhgimeve. Vëzhgimet aftësojnë sistemet multiagjente për të mbledhur të dhëna nga burimet në kontekstin e vendosjes. Përkufizimi i një vëzhgimi përfshin specifikimin e atyre të dhënave për t'u vëzhguar në kontekstin e ndodhjes së bashku me veçoritë shtesë të vëzhgimeve.

6.2. Projektimi logjik

Dekompozimi i mjedisit të aplikacionit mund të konsiderohet në dy dimensione: Në atë horizontal: dekompozim bazuar në menyrat e veçantë me të cilat agjentët mund të aksesojnë mjedisin.

Në atë vertikal: dekompozim bazuar në dallimin midis ndërveprimeve në nivel të lartë midis agjentëve dhe mjedisit të aplikacionit, si dhe ndërveprimeve në nivel të ulët midis mjedisit të aplikimit dhe kontekstit të vendndodhjes. Dekompozimi tregohet në figurën 6.4.

Dekompozimi horizontal i mjedisit të aplikimit konsiston në tre kolona që korespondojnë me mënyrat e ndryshme me të cilat agjentët mund të aksesojnë mjedisin: perceptimin, komunikimin, dhe veprimin. Një agjent mund të ndjejë mjedisin për të marrë paraqitje të botës rreth tij, ai mund të këmbejë mesazhe me agjentët e tjerë, dhe një agjent mund të sjellë një influencë në mjedis duke tentuar të modifikojë gjendjen në mjedis. Përveç së influencave të ardhura nga agjentët, në marrim në konsideratë veprimtari që ndodhin pavaresisht nga agjentët dhe që modifikojnë gjendjen e mjedisit të aplikacionit si pjesë e kolonës së veprimit.

Dekompozimi vertikal i mjedisit të aplikacionit konsiston në dy rreshta. Në rreshtin e sipërm kemi aksesin e agjentëve në mjedisin e aplikacionit dhe përfshin gjeneratorin e paraqitjeve , shërbime të komunikimit si dhe ndërveprim dhe dinamika. Specifikimi i aktiviteteve dhe konceptet në rreshtin e sipërm janë të njëjta si ato të përdorura për agjentët. Rreshti i sipërm përcakton një seri ligjesh që detyrojnë

veprimtaritë e agjentëve në mjedis. Rreshti i poshtëm merret me ndërveprimin e mjedisit të aplikacionit e kontekstin e vendndodhjes dhe konsiston në vëzhgimin dhe sinkronizimin me përpunimin e të dhënave dhe përkthimin. Funksionaliteti lidhur me ndërveprimet në nivel të ulët të mjedisit të aplikimit përfshijnë:

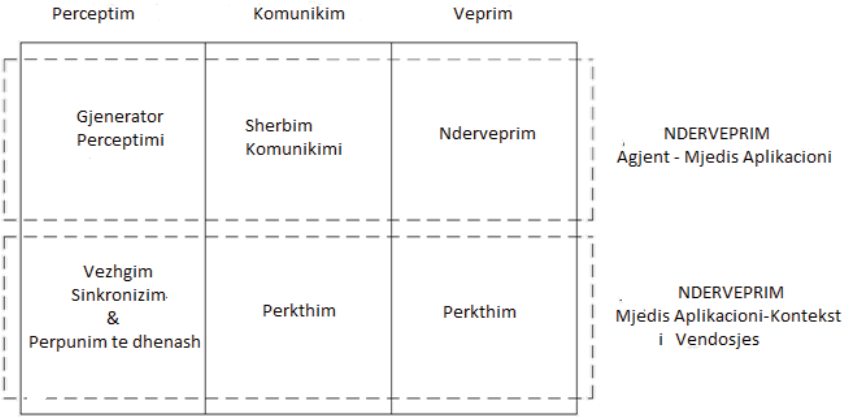

Figura 6. Dekompozimi i Mjedisi të aplikacionit

- Mbështetje për bashkëbisedim të nivelit të lartë të aktivitetit lidhur me agjentët në ndërveprime të nivelit të ulët lidhur me konteksitn e vendndodhjes dhe anasjelltas
- Mbështetje për parapërpunim të të dhënave për t'u transferuar në një nivel më të lartë paraqitjeje të përdorshme nga agjentët.

Dekompozimi dy dimensional i mjedisit të aplikimit jep një modularitet fleksibël që mund të përshtatet në një familje të gjerë të fushave të aplikimeve me agjentë. Për ato aplikime që nuk ndërveprojnë me një kontekst të jashtëm, shtresa e fundit mund të hiqet për dekompozimin vertikal. Për ato aplikime në të cilat agjentët agjentët ndërveprojnë me shënime në mjedis, por që nuk komunikojnë nëpërmjet këmbimit të mesazheve, kolona në dekompozimin horizontal që i korespondon transferimit të mesazheve(shërbimit të komunikimt dhe komunikimit) mund të hiqet.

Çdo modul i mjedisit të aplikimit vendoset në një kolonë dhe rresht të veçantë dhe ka një funksionalitet të veçantë(p.sh. moduli i përkthimit përfshin dy qeliza që sigurojnë funksionalitetin për përkthimin e ndikimeve dhe mesazheve. Duke minimizuar bllokimin e funksionalitetit midis moduleve, kjo ndihmon një projektues të fokusohet në një aspekt të veçantë të funksionalitetit të mjedisit të aplikimit. Kjo e bënë arkitektutrën të ripërdorshme si dhe të mirëpresë ndryshime dhe prurje të një moduli të ri pa ndikuar te të tjerët.

Të dhënat e përbashkëta dhe komponentët e tyre

Paraqitja e të dhënave të përbashkëta tregon se si një sistem multi agjent është strukturuar si një bashkësi akserorësh të të dhënave që lexojnë dhe shkruajnë të dhëna në baza të dhënash të përbashkëta. Elementët kryesore në këtë paraqitje janë:

- Aksesorët e të dhënave
- Bazat e të dhënave
- Konektorët midis tyre

Aksesorët e të dhënave janë komponentë runtime që përformojnë përllogaritje që kërkojnë të dhëna nga një ose më shumë baza të dhënash. Bazat e të dhënave ndërmjetësojnë ndërveprime midis aksesuesve të të dhënave. Një bazë ë tillë të dhënash mund të sigurojë një mekanizëm trigerimi për të sinjalizuar te konsumatorët e të dhënave për ardhjen e të dhënave për të cilat ata janë të interesuar. Pas leximit dhe shkrimit të të dhënave, një bazë të dhënash mund të sigurojë mbështetje shtesë, siç është konkurrenca, etj.

Marrdhënia e pamjes së të dhënave të përbashkëta përcakton se cilët aksesues të të dhënave janë të lidhur dhe me cilën bazë të dhënash**.** Aksesorët e të dhënave u ngjiten konektorëve që janë të lidhur me një bazë të dhënash.

Arkitektura jonë sjell dy paraqitje : së pari zhvillojmë në detaje një paraqitje në detaje të pamjes së agjentit për të dhënat e përbashkëta. Dhe së dyti shikohet arkitektura e mjedisit të aplikacionit.

6.2.1. Të dhënat e përbashkëta për modulin Agjent – komponentët bazë

Elementët bazë dhe veçoritë e tyre. Paraqitja e parë e arkitekturës është dhënë në figurën 6.5. Aksesorët e të dhënave të agjentit janë Perceptim, Vendimmarrje dhe Komunikim. Këto komponente janë instanca ekzekutuese për modulet respektive. Aksesorët ndajnë të njëjtën bazë të njohjes, e cila siguron funksionalitetin e modulit të integrimit të njohjes për modelin e agjentit. Baza e të dhënave të njohjes presente përmban të dhëna që ndahen nga aksesuesit e tyre. Të dhënat e kësaj baze i referohen gjendjes së perceptuar në mjedis, gjendjes lidhur me rolin e agjentit dhe angazhimeve apo gjendjeve të mundshme të brendshme. Komponenti i perceptimit merret me njohjen e agjentit të mjedisit përrreth. Për te rifreskuar njohjen e agjentit, komponentët e komunikimit dhe Vendim marrje së bashku këmbejnë rolet në perceptim për të ndjerë ndryshimet në mjedis.

Në figurën 6.5 më poshtë paraqitet ndërlidhja midis bazës së njohjes prezente dhe komponentëve të brendshëm të agjentit. Këto ndërveprime quhen konektorë mbledhës[13]. Ai lidh një ndërfaqe komponenti me më shume ndërfaqe komponentësh të kërkuar. Baza e njohjes prezente paraqet dy ndërfaqe:

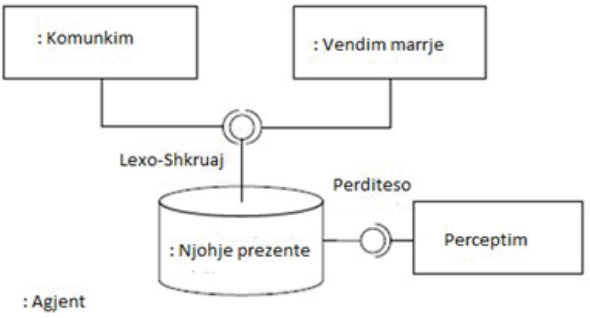

Figura 6. Dekompozimi i modulit agjent

6.2.2. Përshkrimi i ndërfaqes

Përditëso aftëson komponentin perception për të rifreskuar njohjen e agjentit sipas informacionit të ardhur nga sensimi i mjedisit. Ndërsa ndërfaqja Lexo-Shkruaj aftëson komponentët Komunikim dhe Vendim marrje për të aksesuar dhe ndryshuar njohjen prezente të agjentit.

6.2.3. Mekanizmat e variancës

Përcaktimi i njohjes prezente. Përfshin përkufizimin e gjendjes së agjentit dhe specifikimin e bazës së njohjes. Përcaktimi i gjendjes së agjentit duhet të përmbush përcaktimet e sistemeve multiagjente në aplikacione. Specifikimi i njohjes në bazë të dhënash përfshin aspekte të ndryshme si konkurrenca, rregulla të qëndrueshmerisë së të dhënave, mbështetje të tranaksioneve, etj.

6.2.4. Projektimi logjik

Mënyra e përdorimit të të dhënave të përbashkëta ndan edhe komponentë të agjentit. Çiftimi në nivele të ulta përmirëson ndryshueshmerinë që do të thotë ndryshime në një element nuk prekin elementë të tjerë ose ndryshimet kanë efekt lokal. Gjithashtu ripërdorimi është një tjetër avantazh që merret nga ky modelim. Në këtë model elementët të çiftuar dobët kanë përgjegjësi të ndara gjë që i bën ata të kuptohen dhe të njihen më mirë. Elementë të tillë nuk kërkojnë njohje të detajuar rreth strukturës së brendshme dhe veprimeve të elementëve të tjerë. Përsa i takon aksesit në bazën e të dhënave, ky lloj modeli kërkon përpjekje speciale për të sinkronizuar aksesin në të dhënat e përbashkëta që këta elementë ndajnë.

Së bashku komunikimi dhe decision making delegojnë kërkesa perceptimi tek komponenti i perceptimit. Ky i fundit rifreskon njohjen e agjentit me informacionin e

derivuar nga të perceptuarit e mjedisit. Baza e të dhënave të njohjes prezente bën të vlefshme informacionin e rifreskuar për komponentët e komunikimit dhe decision making. Duke ndarë njohjen , të dy komponentët mund të përdorin të dhënat prezente për të marrë vendime. Gjithashtu kjo mënyrë organizimi e të dhënave lejon që të dy këta komponentë të ndajnë të dhëna dhe të komunikojnë në mënyrë indirekte duke lejuar që ato të veprojnë në paralel duke përmirësuar eficencën dhe adaptueshmërinë.

Një alternativë është modeli ku çdo komponent enkapsulon gjendjen e vet dhe siguron ndërfaqe nëpërmjet të cilës elementët e tjerë kanë akses në një informacion të veçantë. Megjithatë që kurse mjaft gjendje të agjentit mund të rrisin varesitë midis komponentëve , këto gjendje duhet të sinkronizohen.

Të dhënat e përbashkëta: mjedisi i aplikacionit

Elementët dhe veçoritë e tyre. Mjedisi i aplikimit konsiston në disa aksesues të dhënash që i bashkangjiten dy bazave të të dhënave: Gjendje dhe Ligje. Aksesorët e të dhënave janë instanca runtime të moduleve korresponduese. Baza Gjendje siguron funksionalitetin e modulit Mirëmbajtje gjendjeje të modelit të mjedisit. Ndërsa baza Ligje enkapsulon ligje të ndryshme (perceptim, komunikim, veprim) të modelit të mjedisit.

Për të përformuar funksionalitetet e tyre, duhet të lexojnë gjendjen e mjedisit. Gjeneratori i paraqitjes, shërbimi i komunikimit dhe komponentët e përkthimit kanë nevojë vetëm të lexojnë gjendjen e bazës Gjenje për të kryer funksionalitetet e tyre.

Baza Ligje mban ligje të ndryshme të vendosura për aplikacionin. Kjo bazë ndahet në tre nën baza:

1. Ligje të perceptimit (p-ligje)
2. Ligje të veprimit(a-ligje)
3. Ligje të komunikimit(c-ligje)

Secila nga këto baza është bashkangjitur komponentit përgjegjës për funksionalitetin korrespondues.

6.2.5. Përshkrimi i ndërfaqes

Në figurën 6.6 paraqiten ndërveprimet midis bazave Gjendje dhe komponentëve të brendshëm të mjedisit të aplikacionit. Baza Gjendje paraqet dy ndërfaqe. Ndërfaqja Lexo aftëson komponentët e bashkangjitur të lexojnë gjendjen e bazës. Ndërsa Lexo-Shkruaj aftëson komponentët të aksesojnë dhe të modifikojnë gjendjen e mjedisit të aplikimit.

Baza Ligje paraqet tre ndërfaqe për të lexuar tipe të ndryshme ligjesh. : Lexo-AL, Lexo-PL, Lexo-CL. Këto lejojnë konsultim të komponentëve të ndërfaqësuar.

6.2.6. Mekanizmat e variancës

Në këtë modul sigurohet një mekanizëm variance:

Përcaktimi i gjendjes. Përfshin përcaktimin e gjendjes prezente të mjedisit të aplikacionit dhe specifikimin e bazës së gjendjeve. Përcaktimi i gjendjeve duhet të përmbush modelin për aplikacionet me agjentë. Specifikimi i bazës së gjendjeve përfshin aspekte të ndryshme siç është konkurrenca , specifikimi i mekanizmave për trajtimin e ngjarjeve të veçanta që njoftojnë konsumatorët e këtyre të dhënave si dhe suporti për transaksione të mundshme. Interpretimi i këtyre aspekteve varet nga kërkesat specifike të fushës së aplikimit.

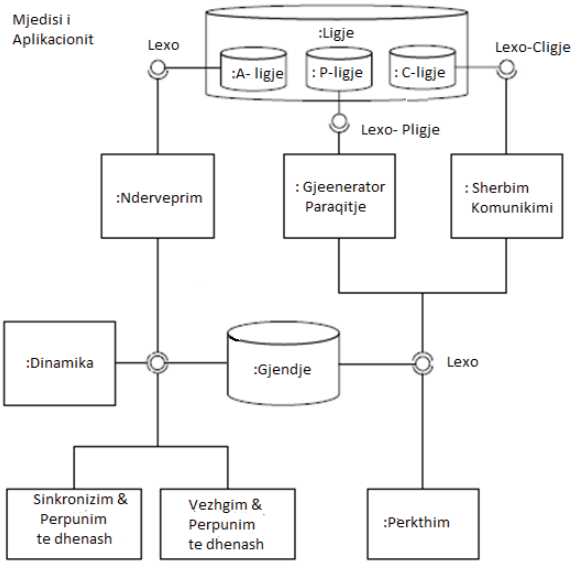

Figura 6. Të dhënat e përbashkëta për modulin Mjedis

6.2.7. Modeli logjik

Motivimet për aplikimin e modelit të të dhënave në arkitekturën e mjedisit të aplikacionit janë të ngjashme si ato të arkitekturës së agjentit. Modeli i të dhënave të përbashkëta rezulton në lidhje të dobët midis elementëve të ndryshëm duke përmirësuar ndryshueshmërinë dhe ripërdorueshmërinë.

Baza e gjendjeve aftëson komponentë të ndryshëm të mjedisit të aplikacionit të ndajë të dhëna gjendjeje dhe të komunikojë në menyrë indirekte. Kjo ndalon dublikimin e të dhënave dhe lejon që komponentë të ndryshëm të veprojnë në paralel.

Baza e të dhënave për Ligjet enkapsulon ligje të ndryshme që do të përdoren nga komponentë të ndryshëm.

Komponentët dhe ndërlidhësit e tyre

Arkitektura e komponentëve bashkëpunues paraqet sistemin multiagjent si një bashkësi të komponentëve bashkëveprues ekzekutues të cilët përdorin një bashkësi të dhënash të përbashkëta për të realizuar funksionalitetet e kërkuara të sistemit. Diagrama e komponentëve bashkëpunues paraqet sesi këta komponentë realizojnë funksionalitete të ndryshme në një sistem multiagjent.

Elementët kryesorë të një diagrame të tillë janë:

- Komponentë ekzekutues. Arrijnë një pjesë të funksionalitetit të sistemit. Janë instanca të moduleve të paraqitura në dekompozimin e sistemit.
- Bazat e të dhënave. Aftësojnë komponentët e mësipërm të ndajnë të dhëna. Ato i korrespondojnë bazave të të dhënave përkatësisht sipas modelit të dekompozimit për bazat e të dhënave të përbashkëta.
- Konektorët për komponent –baza të dhënash. Lidhin këto dy elemente duke përcaktuar se cilët komponentë mund të shkruajnë apo të lexojnë të
- dhëna në bazën e të dhënave të sistemit.
- Konektorët komponent- komponent. Komponentët që bashkëveprojnë kërkojnë funksionalitete nga njeri tjetri dhe sigurojnë këtë për komponentët e tjerë. Aftësojnë komponentët ekzekutive të kërkojnë nga njëri tjetri të performojnë funksionalitete të veçanta.

Komponentët bashkëpunues në arkitekturë janë një mjet i mirë për të kuptuar sjelljen e një sistemi multiagjent. Diagrama paraqet rrjedhjen e të dhënave midis komponentëve ekzekutues dhe ndërveprimet me bazat e të dhënave si dhe ajo specifikon funksionalitetet e secilit komponent në termat e hyrjes dhe daljes së të dhënave. Arkitektura jep tre pamje të komponentëve bashkëpunues. Në paraqitjen e parë përshruhen komponentët bashkëpunues të perceptimit. Më tej paraqitja tjetër jep komponentët bashkëpunues të ndërveprimit dhe në fund paraqitja e komunikimit që përshkruan bashkëpunimin midis komponentëve të komunikimit dhe shërbimit të komunikimit.

Komponentët bashkëpunues: Perceptimi dhe Gjeneratori i paraqitjes

Elementët dhe veçoritë e tyre. Kjo diagramë paraqet një numër të komponentëve bashkëpunues që realizojnë perceptimin. Elemetët kryesore të kësaj diagrame janë komponentët e perceptimit , gjeneratori i paraqitjes dhe baza e të dhënave lokale.

Për të shpjeguar bashkepunimin midis komponentëve të ndryshëm, ne do të përdorim rrjedhën logjike të veprimeve që identifikohen nga çasti kur agjenti merr inisiativën të ndjejë mjedisin derisa perceptimi është në gjendje të rifreskojë njohjen

prezente të agjentit. Sensim merr atributet e një selektuesi të fokusit dhe selekton një fokus korrespondues për të prodhuar një kërkesë perceptimi. Kufizim_perceptimi merr një bashkësi kërkesash perceptimi dhe gjeneron sipas ligjeve të perceptimit një qellim perceptimi. Ky i fundit përcakton qëllimin e kërkesës së perceptimit sipas detyrimeve të vendosura nga ligjet e perceptimit.

Mbledhës_gjendje mbledh gjendjen e vërejtur për qëllimin e perceptimit për një gjendje prezente të mjedisit. Në veçanti, Mbledhës_gjendje zgjedh nënbashkësi të elementëve të gjendjeve të mjedisit të aplikacionit për qëllimin e perceptimit dhe prodhon një vëzhgim për të mbledhur të dhëna nga konteksti i vendosjes me qëllimin e perceptimit. Gjenerator i paraqitjes merr gjendjen e vëzhguar nga mjedisi i aplikacionit me gjendjen e vëzhguar nga konteksti i vendosjes dhe prodhon një paraqitje.

Interpretim përdor bashkësinë e përshkrimeve të agjentit për të interpretuar paraqitjen. Interpretim rezulton në një perceptim të agjentit. Më tej janë filtrat në *Filtrim* dhe *Selektim*, filtri që filtrojnë perceptimin për të rifreskuar njohjen prezente të agjentit.

Mekanizmat e variancës

Ky model ofron dy mekanizma të variancës:

Përcaktimi i përshkrimeve. Përshkrimet mundësojnë një agjent për të interpretuar paraqitje që rrjedhin nga vëzhgimi i mjedisit. Një përshkrim mund të përkufizohet si një template që specifikon një model të veçantë të një paraqitjeje. Duke interpretuar një paraqitje, themi se ajo vjen për të kërkuar për përputhje mes një template përshkrimi dhe paraqitje të ekzaminuar. Çdo perputhje jep të dhënat e një perceptimi që është përdorur për të rinovuar gjendjen e agjentit, ndoshta pas disa filtrime të mëparshme.

Përcaktimi i Ligjeve të perceptimit. Ligjet perceptimi imponojnë specifika të aplikimit të detyrueshme nga perceptimi i agjentëve mbi mjedisin. Çdo ligj perceptimi përcakton kufizime në atë që mund të ndjehet nga gjendja aktuale e mjedisit për një fokus të veçantë. Kufizimet e imponuara nga një ligj perceptimi mund të përcaktohet në lidhje me gjendjen aktuale të mjedisit. Për shembull, kufizimet në vëzhgimin e nyjeve lokale në një rrjet mund të jetë përcaktuar në funksion të trafikut aktual në rrjet.

Projektimi logjik

Grup i integruar i komponentëve të perceptimit dhe gjeneratori i paraqitjes ofrojnë funksionalitet për perceptimin selektiv në sistemin multiagjent. Funksionaliteti i përgjithshem rezulton nga bashkëpunimi i komponentëve të ndryshëm në këtë bashkëpunim, çdo komponent ofron një funksionalitet të qartë,

ndërsa çiftimi midis komponentit është mbajtur i dobet. Koncepte të tilla si: fokusimet, përshkrimet, filtra, dhe ligjet janë të nivelit të parë në arkitekturën e sistemit. Kjo ndihmon për të përmirësuar ndryshueshmërinë dhe ripërdorueshmërinë e arkitekturës.

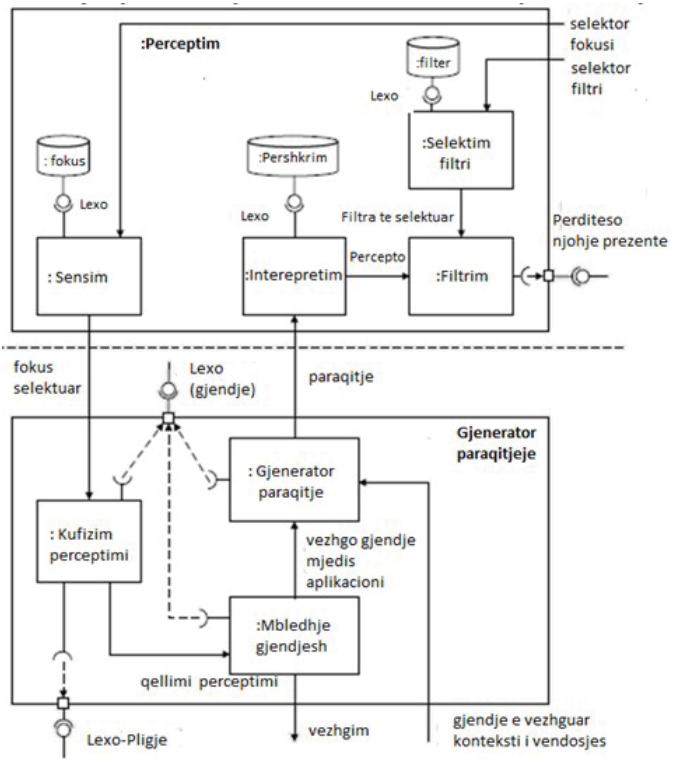

Figura 6. Komponentët bashkëpunues dhe ndërfaqet e komunikimit midis tyre

Perceptimi selektiv lejon një agjent për të përshtatur perceptimin e saj në përputhje me detyrat e tij aktuale. Arkitektura mbështet përshtatjen e ligjeve sipas perceptimit në rrethanat me ndryshim të mjedisit. Të dyja këto cilësi kontribuojnë tek fleksibiliteti i sistemit.

Komponentët bashkëpunues dhe konektorë:dekompozimi në nivel të dytë

6.2.8. Vendimmarrje dhe Ndërveprim

Elementët dhe veçoritë e tyre. Në këtë nivel, dekompozimi paraqet komponentët bashkëpunues që realizojnë funksionalitetin për veprim. Elementët në këtë paraqitje janë komponentët e vendimmarrjes dhe ndërveprimit dhe dy bazave të të dhënave lokale. Për të diskutuar mbi këto elementë, do të marrim parasysh

aktivitetet e njëpasnjëshme që ndodhin nga momenti që një agjent nis një vendimmarrje derisa efektet e influencës sa janë zgjedhur janë përfunduar.

Përditësim_njohje mundëson agjentin për të rinovuar njohuritë e tij aktuale të mjedisit. Përditësim_njohje **merr njohjen prezente të agjentit dhe bashkësine e stimujve aktuale dhe** prodhon një përzgjedhës fokusi dhe filtër. Ne kemi dalluar mes dy lloje të stimujve:

- **Stimuj të jashtëm**. Një stimul i jashtëm i referohet një faktor në mjedisin që drejton vendimmarrjen e agjentit.
- **Stimuj të brendshëm**. Një stimul i brendshëm është një faktor i brendshëm që prek vendimmarrjen e agjentit.

Selektuesi i fokusit dhe filtrit është kaluar në modulin e perceptimit i cili senson mjedisin për të prodhuar një perceptim dhe që rifreskon njohjen e agjentit. Selektim_veprimi merr stimuj prezentë dhe njohjen e agjentit për të zgjedhur një operator dhe përditëson grupin e stimujve prezente. Një operator është një paraqitje e brendshme e përdorur nga agjenti për të përfaqësuar veprimin e zgjedhur. Vendimmarrja konverton një operator në influencë që thirret në mjedis. Për të zgjedhur operatorët e përshtatshëm, komponenti i zgjedhjes së veprimit komponenti enkapsulon një mekanizëm perzgjedhës të veprimit bazuar në zgjedhje.

Në përgjithësi, një mekanizëm i tillë bazohet në një bashkësi modulesh të sjelljes. Çdo modul sjelljeje është një modul relativisht e thjeshtë llogaritjeje që lidh fortë sensimin me veprimin.

Një skemë arbitrazhi apo ndërmjetësimi kontrollon se cilat module që prodhojnë sjellje kanë kontrollin dhe zgjedh veprimin e ardhshëm të agjentit Për të mundësuar agjentët për të vendosur bashkëpunime, mekanizmat e përzgjedhjes së veprimit bazuar në sjellje janë zgjeruar me nocionin e rolit dhe të angazhimit. Modulet e sjelljes që përfaqësojnë një pjesë të funksionimit koherent në kontekstin e një organizate janë të përcaktuara si një rol. Një rol është përkufizuar si:

R:(emër, stimuli, operatore, zgjidh)

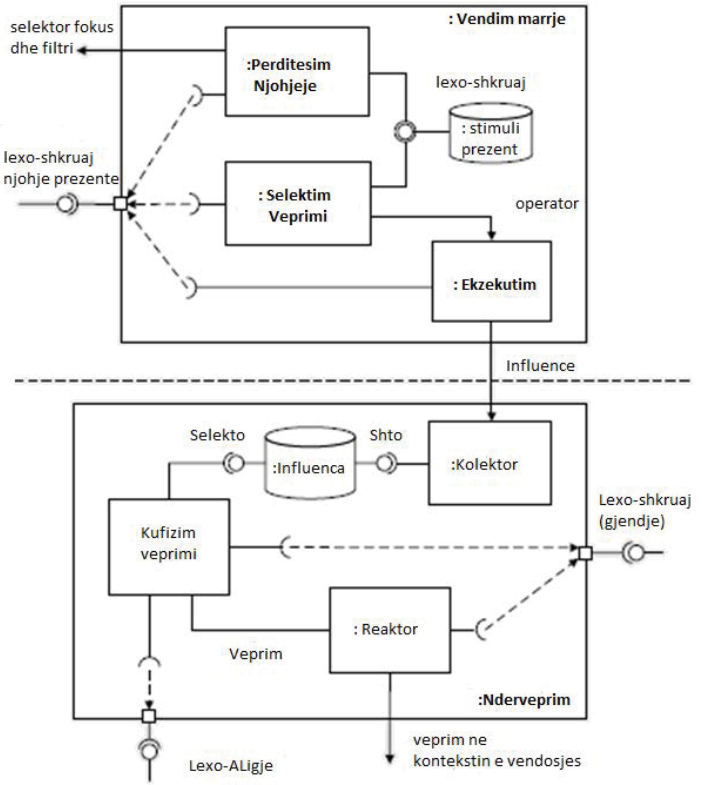

Figura 6. Bllokskema për komponentët Vendimmarrje dhe Ndërveprim

Rolet kanë një emër të mirënjohur që është i përbashkët mes agjentëve në sistem. Funksioni zgjedh dhe harton grupin e stimujve për një grup të operatorëve dhe pre-faktorëve, një për secilin operator në bashkësinë e operatorëve që mund të përzgjidhen nga roli. Një pre-faktor përcakton preferencën relative për zgjedhjen e operatorit shoqërues. Operatori dhe pre-faktori janë përdorur nga ana e skemës së arbitrazhit për të përcaktuar se cili rol ka kontrollin dhe se cili operator është zgjedhur për ekzekutim.

Një angazhim i vendosur është përcaktuar si:

C: (emri, rel-set, konteksti, akt-kusht, çakt-çkusht, gjendje, harte_role)

Sikurse rolet, angazhimet e vendosura kanë një emër të mirënjohur. Shprehimisht emërtimet e roleve dhe angazhimeve mundëson agjentët për të ngritur bashkëpunime, të reflektuara në angazhimet eventuale. Bashkësia e marrëdhënieve rel-set (bashkësi reflekse) përmban identitetin e agjentit përkatës në angazhimin e ndodhur. Konteksti përshkruan vetitë kontekstuale të angazhimit të vendosur, të tilla si përshkrimet e objekteve në mjedisin lokal. *Akt-kusht* dhe çakt-*kusht* janë kushtet e

aktivizimit dhe çaktivizimit që përcaktojnë statusin e angazhimit të ndodhur. Kur gjendja e aktivizimit bëhet i vërtetë, angazhimi aktivizohet. Sjellja e agjentit do përcaktohet sa e njëanshme sipas përcaktimit të hartes se roleve(harte_*role*). Rolemap specifikon peshën relative të faktorëve preferenciale të operatorëve të roleve të ndryshme. Në formën e tij më të thjeshtë, rolemap ngushton zgjedhjen e agjentit të veprimit të operatorëve në një rol të veçantë. Sa më shpejt që të bëhet gjendja e çaktivizimit të vërtetë, angazhimi i vendosur është çaktivizuar dhe nuk do të ndikojë në sjelljen e agjentit.

Ekzekutimi merr operatorin e përzgjedhur dhe njohjen aktuale të agjentit dhe prodhon një ndikim që thirret në mjedis. Ekzekutimi decouples përfaqeson aktivitetin e brendshëm të agjentit nga ndikimet që janë në dispozicion për agjentin për të hyrë dhe për të modifikuar gjendjen e mjedisit.

Kolektor mbledh ndikimet të thirrura nga agjentët, dhe shton ndikimet në grupin e influencave të mbetura në pritje në sistemin me agjentë. Kufizim_veprimi zgjedh një ndikim nga grup i ndikimeve në pritje dhe konverton ndikimin në një veprim. Një veprim është një përfaqësim i influences së zgjedhur brenda mjedisit të aplikimit. Përzgjedhja e ndikimit bazohet në një politikë të përzgjedhjes së ndikimit që specifikon reshtimin e ndikimeve, duke marrë parasysh gjendjen aktuale të mjedisit. Kufizim_veprimi zbaton një sërë ligjesh të veprimit për veprimin e përzgjedhur, duke e tanishme gjendjen e mjedisit. Ligjet e veprimit vendosin kufizime për llojin e manipulimeve që agjentët mund të kryejnë në mjedis.

Reaktor zbaton veprimin në gjendjen aktuale të aplikimit të mjedisit dhe prodhon një veprim për të vepruar në kontekstin e vendosjes nëse është e aplikueshme.

Ekzekutimi i veprimit ndryshon gjendjen e mjedisit të aplikimit dhe prodhon një veprim në kontekstin e vendosjes që kalohet për modulin e perkthimit.

6.2.9. Modeli logjik

Komponentët bashkëpunues të vendimmarrjes dhe ndërveprimit sigurojnë funksionalitetin për ekzekutimin e veprimit në sistemin me agjent. Në këtë bashkëpunim, çdo komponent ofron një funksionalitet të qartë, ndërsa lidhja midis komponentëve është mbajtur e dobët. Kjo ndihmon për të përmirësuar modifikueshmërinë dhe ripërdorueshmërinë e sistemit. Zgjedhja e veprimit bazuar në sjellje u mundëson agjentëve të sillen në përputhje me situatën në mjedis, dhe për të ndryshuar sjelljen e tyre me ndryshimin e rrethanave të mjedisit. Konceptimi për rolin dhe angazhimin mundëson agjentët për të ngritur bashkëpunime. Kohëzgjatja e një angazhimi mund të rregullohet në bazë të kushteve në kontekstin lokal në të cilin agjentë janë vendosur. Kjo qasje shkon sipas parimeve të përgjithshme të vendosjes dhe pozicionimit (*situatedness*) dhe përmirëson fleksibilietin dhe transparencën. Një

agjent përshtat sjelljen e tij kur kushtet e mjedisit ndryshojnë ose kur agjentët hyjnë ose largohen nga qëllimi i ndërveprimit.

Ligjet e veprimit dhe politika e përzgjedhjes se influencave siguron një mjet për të rregulluar veprimet e agjentëve në sistem. Të dy ligjet e veprimit dhe përzgjedhjes mund të përcaktohen mbi bazën e kushteve prezente të mjedisit duke përmirësuar fleksibilitetin e mjedisit.

Komponentët bashkëpunues: shërbimi i komunikimit dhe komunikimi

6.2.10. Elementët dhe veçoritë e tyre

Kjo bllokskemë tregon se si një numër i komponentëve bashkëpunues realizojnë funksionalitetin për komunikim. Elementët në këtë bllokskemë janë komponentët e komunikimit, shërbimeve dhe bazat e të dhënave lokale. Ne do të diskutojmë komunikimin si komponent që merret me ndërveprimet komunikuese të agjentëve. Komponenti i komunikimit përpunon mesazhet që vijnë, dhe prodhon mesazhe që dalin sipas protokolleve të mirëpërcaktuar. Ndërveprimet komunikuese mund të modifikojnë gjendjen e agjentit duke ndikuar zgjedhjen e influencave të agjentit. Një shembull tipik është aktivizimi ose çaktivizimi i angazhimeve të agjentit. Një protokoll kounikimi konsiston në një seri hapash të protokollit. Një hap është një seri kusht-efekt(*condition*, *effect*). Kushti është një shprehje booleane që përcakton nëse është i zbatueshëm hapi. Shprehjet mund të merren me llogarine e njohjes aktuale të agjentit, bashkësia e bisedave që përmban historinë e agjentit për ndërveprimet e vazhdueshme, dhe të dhënat e një mesazhi të marrë të mundshëm.

Ne dallojmë tre lloje të hapave të protokollit:

- **Fillim_bisedë**,. Një hap fillimi bisede fillon një bisedë të re në bazë të një protokolli të veçantë. Një agjent nis një bisedë të bazuar në njohjen e tij aktuale, ndoshta duke marrë parasysh të dhënat e deshifruara nga mesazhi marrë nga një tjetër agjent që filloi bashkëveprimin.
- **Vazhdim_bisedë** Një vazhdim bisede kryen një hap në një bisedë të vazhdueshme. Një vazhdim bisede mund të merret me një mesazh të marra direkt pa iu përgjigjur, menjëherë mund të reagojnë me një mesazh përgjigje, ose mund të kap një bisedë pas një pushimi.
    - **Përfundim_bisedë** Së fundi, një përfundim bisede përfundon një bisedë të vazhdueshme. Përfundimi i një bisede mund të jetë i detyruar nga rrethanat që ndryshojnë në mjedis ose mund të rezultojë drejtpërdrejt nga një hap paraprak i bisedës.

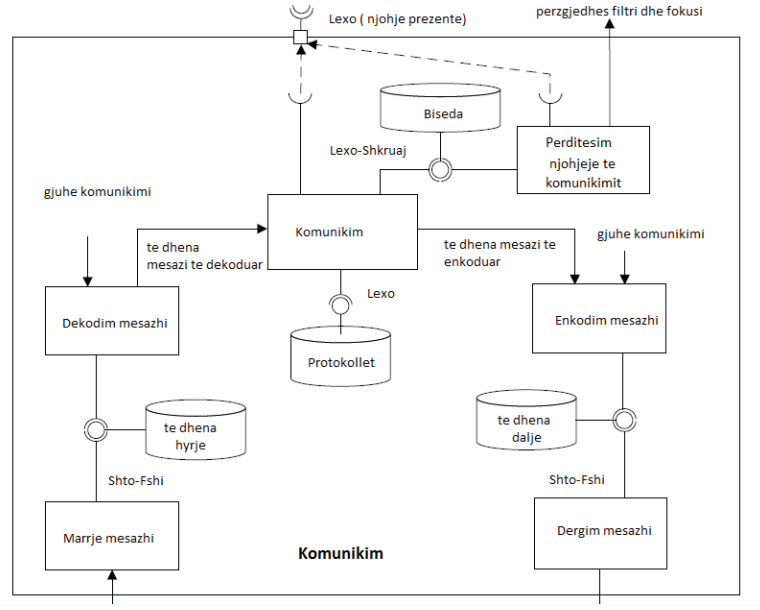

Figura 6. Bllokskema për komponent komunikim dhe shërbim komunikimi

Përditësim _njohje mundëson agjent për të rinovuar njohuritë e tij aktuale sipas bisedave të tij të vazhdueshme. Komponenti i azhornimit të njohurive merr njohjen aktuale të agjentit dhe grupin e bisedave dhe prodhon një zgjedhës fokusi dhe filtëri që është kaluar në modulin e perceptimit që senson mjedisin, prodhon një perceptim të ri dhe përditëson njohjen e agjentit.

Enkodim_mesazhi kodon të dhënat e një mesazhi në mesazhet dhe vendos mesazhet në bufferin e daljes së agjentit. Enkodim_mesazh bazohet në gjuhën e komunikimit që është e përbashkët mes agjentëve në sistem.

Dergo_mesazh përzgjedh një mesazh nga grupi i mesazheve të mbetura pezull ne bufferin e daljes dhe e kalon atë në shërbimin e komunikimit.

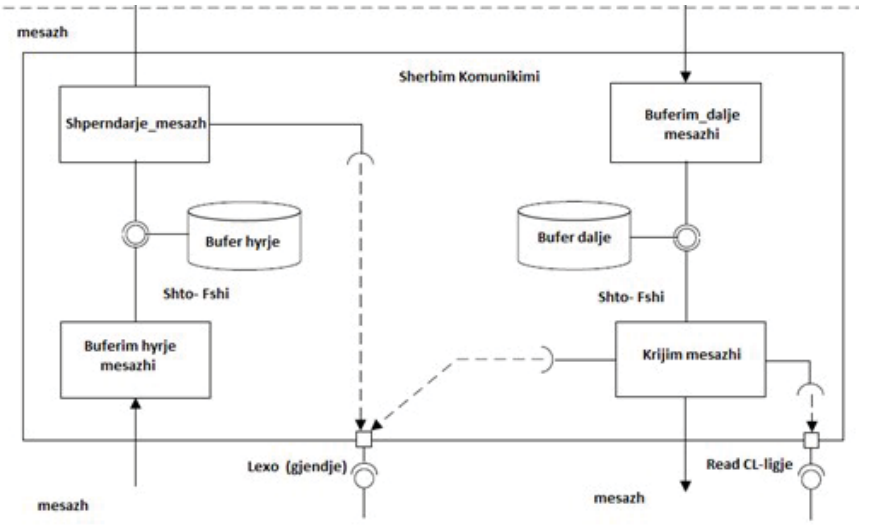

Figura 6. Bllokskema për komponenten Shërbim_komunikimi

Bufer_dalje_mesazh mbledh mesazhet e dërguara nga agjentët dhe i vë ato në bufferin e daljes të shërbimit të komunikimit. Postimi përzgjedh një mesazh nga bufferi i daljes duke dhënë një politikë të përzgjedhjes të mesazhit, dhe aplikon një sërë ligjesh të komunikimit për mesazhin e zgjedhur duke marrë parasysh gjendjen aktuale të mjedisit. Për të përcaktuar rendin në të cilin janë dërguar mesazhet, politika e zgjedhjes merr parasysh gjendjen aktuale të mjedisit. Mailing kalon mesazhet në komponentin e përkthimit që merret me konvertimin e mesazheve për transmetim.

Buferim hyrje mesazhi mbledh mesazhet hyrëse dhe i vë ato në buferin e hyrjes të mjedisit. *Shpërndarje mesazhi* jep mesazhet e dhëna të buferit të hyrjes te agjentët e duhur.

Marrje mesazhi pranon mesazhe dhe i vë ato në kutinë e agjentit. Së fundi, Dekodim mesazhi përzgjedh një mesazh nga kutia e agjentit dhe dekodon mesazhin në bazë të gjuhës se dhënë tëkomunikimit. Të dhënat e deshifruara të mesazhit të zgjedhur kalojnë në komponentin e komunikimit që do të perpunojë ato.

6.2.11. Mekanizmat e variances

Ka tre mekaniza variance në këtë modul.

Përcaktimi i protokollit të komunikimit. Kjo përfshin specfikimin e :

- Kushtet për të cilat agjenti nis një protokoll.
- Kushtet për të cilat agjenti vazhdon ndërveprimin në faza të ndryshme të protokollit

- Kushtet për të përfunduar ndërveprimin.

Për çdo situatë, protokolli duhet të përcaktojë veprimin konkret që agjenti duhet të ekzekutojë. Për çdo situatë, protokolli duhet të përcaktojë veprimet konkrete që duhet të ekzekutohen. Veprimet përfshijnë përpunimin e mesazheve të marra, përbërjen e mesazheve të reja dhe përditësimin e njohurive të tanishme.[90] Një aspekt i rëndësishëm i këtij të fundit është aktivizimi ose çaktivizimi i angazhimeve të ndodhura.

Përcaktimi i rregullit përzgjedhës të mesazhit. Një përzgjedhje mesazhi imponon një politikë specifike të rreshtimit të aplikimit në transmetimin e mesazheve të pazgjidhura nga bufferi dalës i mjedisit të aplikimit. Një politikë e thjeshtë është FIFO. Forma të avancuara të rreshtimit mund të përcaktohen bazuar në kombinimin e kritereve, të tilla si rreshtim bazuar në llojin e mesazheve dhe llojin e agjentëve që kanë dërguar mesazhe. Përveç kësaj, politika mund të marrë në konsideratë informacione të tilla si ngarkesa aktuale të rrjetit për të zgjedhur mesazhet. Në parim, një politikë zgjedhjeje duhet të përcaktohet për bufferin e hyrjes të shërbimit të komunikimit gjithashtu. Megjithatë, qëkurse rrjedha e mesazheve mes agjentëve është e rregulluar tashmë nga politika e përzgjedhjes se mesazheve e bufferit të daljes, mund të përdoret një politikë e përzgjedhjes statike si FIFO për përzgjedhjen e mesazheve nga bufferi i hyrjes. Edhe pse kjo mund të shkaktojë sjellje jooptimale në situata të veçanta, në përgjithësi shmang analizat komplekse dhe të gjata në kohë të koleksionev të mesazheve të bufferizuara.

Përcaktimi i ligjeve të komunikimit. Ligjet e komunikimit imponojnë kufizime specifike të aplikimit për ndërveprime të agjentëve komunikues në mjedis. Një ligj komunikimi përcakton kufizimet për dërgimin e mesazheve. Kufizimet e imponuara nga një ligj komunikimi mund të përcaktohen në lidhje me gjendjen aktuale të mjedisit. Për shembull, shperndarja e një mesazhi të transmetimit në një rrjet mund të kufizohet ne adresa që janë të vendosura brenda një zone të caktuar fizike perreth dërguesit.

6.2.12. Projektimi logjk

Komponentët bashkëpunues të komunikimit dhe të shërbimit të komunikimit sigurojnë funksionalitetin për këmbimin e mesazheve në sistemin agjent. Komunikimi i drejtpërdrejtë i lejon agjentët për të shkëmbyer informacion dhe për të ngritur bashkëpunime. Koordinimi nëpërmjet shkëmbimit të mesazheve plotësohet me koordinimin nëpërmjet elementëve përcaktues të mjedisit (p.sh. piksel të ngjyrosur). Komponentët e ndryshëm në bashkëveprim përcaktojnë përgjegjësi të qarta dhe bashkimi mes komponentëve është mbajtur i nivelit të ulët. Komunikimi i përcaktuar në drejtim të protokolleve të vë fokusin e komunikimit në marrëdhëniet midis mesazheve. Në çdo hap të ndërveprimit komunikues, janë kushtet që përcaktojnë

sjelljen e agjentit në nje bisedë. Kushtet nuk varen vetëm nga statusi i bisedave të vazhdueshme dhe përmbajtja e mesazheve të marra, por edhe në kushtet aktuale në mjedis të reflektuara në njohuritë aktuale të agjentit dhe në veçanti mbi statusin e angazhimeve të agjentit. Kjo kontribuon në fleksibilitetin e sjelljes së agjentit. Shërbimit të komunikimit vetëm i siguron funksionet bazë për shkëmbimin e mesazhit. Komunikimi i drejtpërdrejtë në një sistem multiagjent duhet të përputhet me parimin e lokalizimit, dmth komunikimi duhet të shërbejë si një mjet për agjentët për të shkëmbyer informacion dhe për të ngritur bashkëpunime me agjentë në zonën e tyre. Nëse është e nevojshme, shërbimi i komunikimit mund të sigurohet me shërbime shtesë, të tilla si një shërbim faqe e verdhë (faqe për të mundësuar agjentët për të gjetur ofruesit e shërbimeve të caktuara). Por, pasi këto shërbime janë të rralla për sistemet me agjent robotike, ata nuk janë të përfshira në arkitekturën e paraqitur. Politika e zgjedhjes së mesazhit dhe ligjet e komunikimit sigurojnë mekanizma për tërregulluar rrjedhën e mesazheve në sistemin agjent. Të dyja politikat dhe ligjet janë vendosur sipas kushteve të ndryshueshme në mjedis, duke kontribuar në arritjen e fleksibilitetit të sistemit.

6.2.13. Proçeset per elementet komunikuese

Bllokskema e proçeseve të komunikimit paraqet sistemin multiagjent si një bashkësi të elementëve njëkohësisht në ekzekutim dhe ndërveprimet midis tyre. Elementet janë njësi, baza të dhënash, dhe lidhës. Njësitë janë një abstraksion i më shumë elementeve softuere konkrete të tilla si detyrë, proçes, dhe thread. Lidhësit mundësojnë shkëmbimin e të dhënave ndërmjet njësive konkurruese dhe kontrollin e tyre si fillim, stop, sinkronizimi, etj. Marrëdhënia në këtë bllokskemë tregon se cilët lidhës janë lidhur me njësitë dhe bazën e të dhënave [91] Bllokskema e proçeseve të komunikimit shpjegon cilat pjesë të sistemit veprojnë paralelisht dhe për këtë arsye është një artifakt i rëndësishëm për të kuptuar se si punon sistemi dhe për të analizuar performancën e tij. Për më tepër, bllokskema është e rëndësishme për të vendosur se cilat komponentë duhet të caktohen për secilin proçes. Në fakt, ne paraqesim proçeset komunikuese si një numër i komponentëve kryesore dhe i veshim me një grup të njësive ekzekutuese njëkohësisht si dhe ndërveprimet e tyre. Arkitektura ofron një bllokskemë komponentësh dhe komunikimin midis tyre. Në të paraqiten proçeset kryesore të përfshira në perceptim, ndërveprim, dhe komunikim në sistemin multiagjent.

6.2.14. Proçeset: Perceptim, Ndërveprim, Komunikim

Elementet dhe veçorite e tyre. Bllokskema që tregon proçeset e komunikimit dhe lidhjeve paraqitet në figurën 6.21. Kjo pamje tregon proçeset kryesore, të dhënat e agjentit dhe mjedisin e aplikimit. Ne bëjmë një dallim në mes të proçeseve aktive që ekzekutohen në mënyrë autonome, dhe proçeseve reaktive që janë

shkaktuar nga proçeset e tjera që kryejnë një detyrë të veçantë. Diskutimi i elementeve në këtë pamje është i ndarë në katër pjesë. Ne shohim proçeset e komunikimit të perceptimit, ndërveprimit dhe komunikimit, si dhe proçeset e pavarura të mjedisit të aplikacionit.

Perceptimi. Proçesi i perceptimit të agjentit është një proçes reaktiv që mund të aktivizohet nga proçesi vendimmarrjes dhe proçesi i komunikimit. Pasi aktivizohet, proçesi i perceptimit kërkon proçesin e gjeneratorit të paraqitjes për të gjeneruar një paraqitje. Proçesi i gjeneratorit të paraqitjes mbledh gjendjen e kërkuar nga baza e të dhenave të gjendjes të mjedisit të aplikacionit dhe ai kërkon proçesin e vëzhgimit për të mbledhur të dhëna shtesë nga konteksti i aplikimit. Mbledhja e gjendjeve është subjekt i ligjeve të perceptimit. Proçesi i vëzhgimit kthen të dhënat e vërejtura në proçesin e gjeneratorit të paraqitjes sapo ato të vihen në dispozicion. Më pas, gjeneratori i paraqitjes

integron gjendjen e perceptuar dhe gjeneron një paraqitje që kthehet te proçesi i perceptimit të agjentit. Proçesi i perceptimit konverton përfaqësimin në një perceptim që përdor për të rinovuar njohjen e agjentit.

Së fundi, proçesi kërkues mund të lexojë gjendjen e përditësuar të agjentit. Baza e të dhënave, Depoja e tanishme e njohjes mund të sigurojë një mekanizëm njoftimi për të informuar proçesin e vendimmarrjes dhe komunikimit kur është përfunduar një përditësim gjendjeje.

Ndërveprimi. Proçesi i vendimmarrjës është një proçes aktiv i agjentit që zgjedh dhe thërret ndikimet në mjedis. Proçesi i bashkëveprimit mbledh ndikimet njëkohësisht të thirrura dhe i konverton ato në veprime. Ekzekutimi i veprimeve është subjekt i ligjeve të veprimit të sistemit. Operacionet që tentojnë për të ndryshuar gjendjen e mjedisit të aplikimit janë ekzekutuar nga proçesi i ndërveprimit, operacionet që tentojnë për të ndryshuar gjendjen e kontekstit të vendosjes janë përcjellë te proçesi i perkthimit. Ky proçes përkthimi konverton operacionet në ndërveprime të nivelit të ulët në kontekstin e vendosjes.

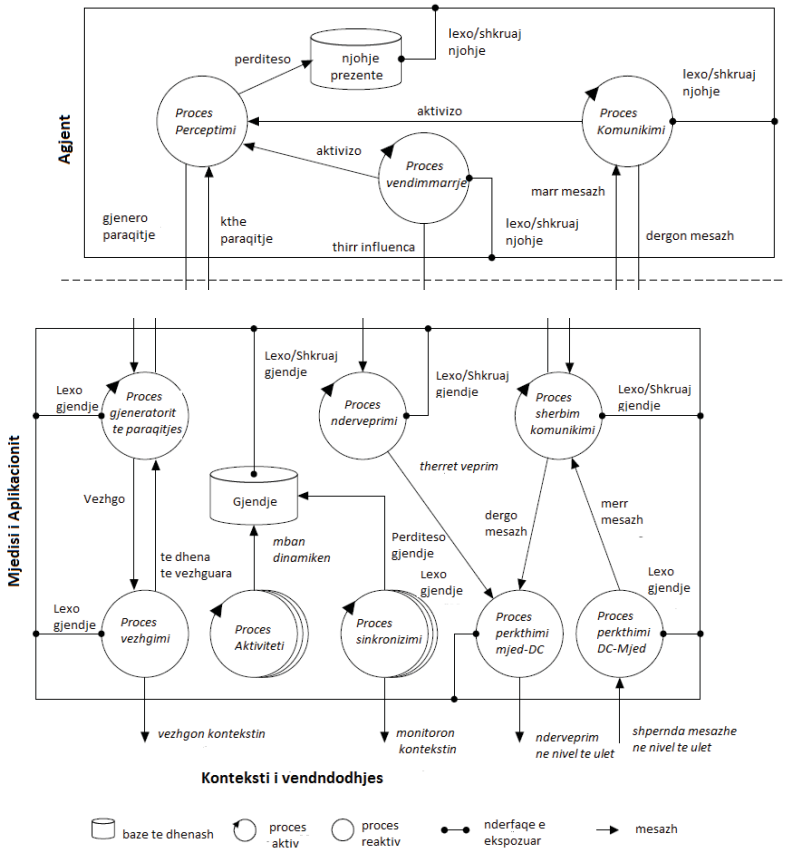

Figura 6. Diagramë e proçeseve në komunikimin e tre shtresave

Komunikimi. Proçesi i komunikimit është një proçes aktiv që merret me ndërveprime komunikuese të agjentit. Mesazhet e përbëra sapo kanë kaluar në proçesin e shërbimit të komunikimit që aplikon ligjet e komunikimit dhe më pas kalon mesazhet në proçesin e përkthimit. Ky proçes i fundit konverton mesazhet në ndërveprime të nivelit të ulët, me kontekstin e vendosjes. Për më tepër, proçesi i perkthimit mbledh mesazhe të nivelit të ulet nga konteksti i vendosjes, konverton mesazhet në një format të kuptueshëm për agjentët, dhe përcjell përpara mesazhet në proçesin e shërbimit të komunikimit. Proçesi i komunikimit jep mesazhe në proçesin e komunikimit të agjentit të përshtatshëm.

Proçeset e pavarur në mjedisin e aplikimit. Proçeset e sinkronizimi janë proçese aktive që monitorojnë pjesë të veçanta të aplikimit në kontekstin e vendosjes dhe mbajnë te përditësuar gjendjen përkatëse të mjedisit të aplikimit. Aktivitetet e vazhdueshme janë proçese aktive që mbajnë dinamikat e veçanta të aplikimit në mjedisin e aplikimit.

6.2.15. Mekanizmat e variancës

Ka dy mekanizma te ndryshimit në këtë diagrame.

Përkufizimi i sinkronizimit të gjëndjes me kontekstin e vendosjes. Pjesët e kontekstit të vendosjes për të cilat ka një paraqitje per t'u mbajtur në mjedisin e aplikimit, duhet të jenë të përcaktuar. Konteksti i vendosjes mund të sigurojë një mekanizëm njoftimi për të informuar proçeset sinkronizimit për ndryshime, apo proçeseve mund të bëjnë sondazhe në kontekstin e vendosjes sipas skemave të caktuara kohore.

Përkufizimi i aktiviteteve të vazhdueshme. Për çdo aktivitet të vazhdueshëm në mjedisin e aplikimit duhet të përcaktohet një proçes aktiv. Proçeset e aktiviteteve në vazhdim mund të kandidojë mënyrë të pavarur, ose ata mund të monitorojë dhe të reagojë ndaj ngjarjeve të veçanta në mjedisin e aplikimit.

6.2.16. Projektimi logjik

Agjentët janë të pajisur me dy proçeset aktive, një për vendimmarrje dhe një për komunikim. Kjo qasje i lejon këto proçese për të kandiduar në mënyrë paralele, duke përmirësuar eficencën. Komunikimi midis proçeseve ndodh në mënyrë indirekte nëpërmjet bazës së të dhënave të njohjes aktuale. Proçesi i perceptimit është reaktiv, agjenti vetëm ndjen mjedisin, kur kërkohet për marrjen e vendimeve dhe ndërveprimin komunikues. Si i tillë, proçesi i perceptimi aktivizohet vetëm kur është e nevojshme. Mjedisi i aplikacionit sigurohet me proçese të veçanta për të mbledhur dhe perpunuar kërkesat e proçesit të perceptimit, të trajtojë në influencat, dhe të sigurojë transferimin e mesazheve. Proçesi i vëzhgimi është reaktiv, ai mbledh të dhëna nga konteksti i vendosjes kur kërkohet nga gjeneratori i paraqitjes. Proçeset e përkthimit janë gjithashtu reaktive, ato ofrojnë shërbimet e tyre nën komandën e proçeseve të tjera. Së fundi, proçeset e sinkronizimi dhe aktivitetet në vazhdim janë proçese aktive që veprojnë të pavarura nga proçeset e tjera në sistem. Proçeset e sinkronizimit monitorojnë dinamika të veçanta në kontekstin e vendosjes dhe mbajnë paraqitjet përkatëse të përditësuara në gjendjen e mjedisit të aplikimit. Aktivitetet e vazhdueshme paraqesin dinamikën në mjedisin e aplikimit që ndodhin të pavarura nga agjentët dhe konteksti i vendosjes. Këto proçese janë përgjegjës për ruajtjen e gjendjes së mjedisit të aplikimit sipas dinamikës së vazhdueshme. Proçeset aktive përfaqësojnë aktivitet të vazhdueshëm në sistem. Duke i lënë proçeset aktive të kandidojë në mënyrë paralele, aktivitete të ndryshme në sistem mund të trajtohen njëkohësisht duke përmirësuar eficencën. Proçeset reaktive, në anën tjetër, janë aktivizuar vetëm dhe kur është e nevojshme të perdorin burimet.

Fusha aplikimi të arkitekturës bazuar ne agjente

Ne kemi zhvilluar më tej projektime bazuar arkitekturën e sistemit bazuar në agjentë. Ajo mbështet zhvillimin e sistemeve agjente vendosur në një mjedis softuerë, si dhe sisteme në një mjedis fizik. Mbështetja ka të beje me perceptimin selektiv, protokolle të bazuar në komunikim, sjellje të bazuar në vendimmarrje, si dhe dinamika që ndodhin të pavarur nga aktivitetet e agjentëve. Bashkëveprimi në mjedis është zgjeruar me mbështetje për veprime të njëkohshme. Veprimet e njëkohshme janë veprime që ndodhin së bashku dhe që mund të ketë një efekt të kombinuar në mjedisin e aplikimit [92]. Një shembull i veprimeve të njëkohshme në mjedis simulimi janë dy agjentë që transportojnë ngarkesa në drejtime të ndryshme. Arkitektura ofron mbështetje për ligjet që mundësojnë projektuesin për të vënë kufizime mbi llojet e ndryshme të aktiviteteve të agjentëve në mjedis, por nuk ofron mbështetje për shpërndarjen e një mjedisi softuerë dhe ndërveprimin me kontekstin e vendosjes. Zhvillimi i arkitekturës është një përvojë e vlefshme. Ajo ka përmirësuar kuptimin e përgjithshëm të aspekteve të rëndësishme të sistemeve të tilla që ndodhin si gjëndja e mjedisit të aplikimit, përfaqësimi i njohjes së agjentëve, dhe *threading*. Ne gjithashtu mësuam se të projektosh nga arkitektura bazë, kërkohet shumë përpjekje dhe ekspertizë projektuesi.

Përmbledhje e kapitullit

Në këtë kapitull, ne kemi prezantuar një arkitekturë për sistemet multiagjent. Qëllimi i përgjithshëm i arkitekturës është për të mbështetur projektimin arkitekturor të vetë-manaxhimit të aplikimeve. Kontributet konkrete janë:

- Arkitektura përcakton se si mekanizma të ndryshme të pershtatjes për sistemet multiagjent janë të integruara në ketë arkitekturë,
- Arkitektura ofron një plan për projektim , ajo lehtëson projektimin e sistemeve të reja softuerë që ndajnë bazë të përbashkët të dhënash,
- Arkitektura mbështet njohuritë dhe ekspertizën e fituar në kërkimin tonë, duke studiuar dhe mësuar mbi perspektivën e avancuar të sistemeve multiagjent që ne kemi zhvilluar në kërkimin tonë.

Ne kemi filluar kapitullin me një përmbledhje të funksioneve kryesore të një sistemi multiagjent. Pastaj ne prezantuam arkitekturën për sistemet multiagjent. Arkitektura paraqet hartën e funksionaliteteve të një sistemi multiagjent mbi një dekompozim të sistemit, elementet softuerë si dhe marrëdhëniet ndërmjet elementeve. Ne kemi paraqitur arkitekturën me anë të katër këndvështrimeve që e përshkruajnë këtë arkitekturë nga perspektiva të ndryshme. Një pamje fokusohet në një pjesë të veçantë të arkitekturës. Ne kemi dhënë një prezantim kryesor të çdo paraqitjeje duke shpjeguar vetitë e elementeve arkitekturore. Përveç kësaj, çdo pamje është e pajisur

me një numër të mekanizmave variancës dhe një arsyetim të projektimit. Mekanizmat e variancës përshkruajnë se si një pamje mund të aplikohet për të ndërtuar arkitektura konkrete softuerë. Modeli logjik shpjegon zgjedhjet kryesore të projektimit të paketës dhe atributet e cilësisë që lidhen me pamje të ndryshme. Për të demonstruar mundësitë e arkitekturës, kemi përmendur dy fusha aplikimi që zbatojnë modulet kryesore të arkitekturës në dy mjedise aplikimi të ndryshme. Arkitektura shërben si një model për zhvillimin e arkitekturave konkrete softuerë. Ajo integron një grup të modeleve arkitekturore që projektuesit mund të nxjerrin gjatë projektimit. Megjithatë, arkitektura nuk është një referencë e gatshme për të gjithë sistemet bazuar në agjentë. Ajo ofron këndvështrim për një sërë zgjidhjesh të ripërdorshme arkitekturore për ndërtimin e arkitekturave softuerë për aplikimet konkrete. Megjithatë, duke aplikuar arkitekturën tonë, kjo nuk e shpëton projektuesin nga çështje të vështira arkitekturore, duke përfshirë zgjedhjen e metodave plotësuese për t'u marrë me kërkesat specifike të sistemit. Ne e konsiderojmë këtë arkitekturë si një referencë për projektimin arkitekturor me komponente të ripërdorshëm për ndërtimin e arkitekturave softuere për aplikimet konkrete. Megjithatë, ky grup nuk është i kompletuar dhe duhet të plotësohet me qasje të tjera arkitekturore.

APLIKACIONI

ZHVILLIMI DHE EVOLIMI I SISTEMIT VETËMANAXHUES BAZUAR NE AGJENTË SOFTUERE

KAPITULLI 7

Sistemet Vetëmanaxhuese Bazuar në Agjentë

Qëllimi i aplikacionit

Aplikacioni synon të sjell një arkitekturë të moduluar bazuar në agjent duke u shtrirë në modele agjentësh të ndryshëm që mbartin funksionalitete specifike. Gjithashtu, aplikacioni synon të krijojë një bazë njohjeje dhe eksperiencë në zhvillimin e sistemeve vetëmanaxhuese softuere. Ai paraqet modelin e zhvillimit të aplikacioneve softuere me komponentë të ripërdorshëm bazuar mbi arkitetkura të njëjta duke ulur kostot e proçeseve të inxhinierisë së softit .

Aplikacioni gjatë zhvillimit të tij u përball me sfida të tilla si shtrirja e arkitekturave të agjentëve në zhvillimin e mekanizmave të përzgjedhjes së veprimit në lidhje të ngushtë me perceptimin dhe komunikimin. Aftësimi i agjentëve për të patur një ndërveprim me mjedisin dhe për të luajtur role të ndryshme në bashkëpunime të ndryshme ishte një tjetër sfidë në realizimin e qëllimit të aplikacionit duke mos lënë mënjanë promovimin e mjedisit si një abstraksion i shkallës së parë që mund të përdoret në mënyrë krijuese në aplikacionet bazuar në multiagjent.

Zhvillimi i një metodologjie bazuar në parime të caktuara në proçesin e inxhinierisë për sistemet multiagjent sjell në këtë punim arkitekturën e aplikacionit si një model i gatshëm që përfshin funksionalitete bazë duke siguruar një zhvillim dhe inkrementim të sistemit multiagjent mbi një model të gatshëm.

Përdoruesit janë të interesuar në vetëproçesim të sasive të mëdha të të dhënave për qëllime të caktuara. **Aplikacionet autonome në manaxhimin e njohjes jane mjaftë të kërkuara në prezencë të sasive të mëdha të informacionit.** Këto aplikacione janë të përdorshme në sisteme vetëmanaxhimi të informacionit, sisteme kontrolli të zonave, sisteme të analizave të mjediseve të ndryshme, etj, të cilat mbështeten mbi baza të dhënash me përmasa të konsiderueshme.

Sistemet multiagjent dhe arkitekturat softuere

Në inxhinierinë e kompjuterave, një agjent është një sistem kompjuterik softuere ose harduere që ka pas veti:

- **Autonomia**: agjentë të veprojë pa ndërhyrje.
- **Aftësia sociale**: agjentët ndërveprojnë me njëri-tjetrin.

- **Reaktiviteti:** Agjentët e perceptojnë mjedisin e tyre dhe veprojnë sipas tij, pra duke iu përgjigjur ndryshimeve të mjedisit
- **Pro-*activeness***: Agjentët shfaqin sjellje të drejtuar nga qëllimi dhe marrin iniciativën për të arritur këto qëllime.

Sipas [93], në përgjithësi një agjent është dikush apo diçka që "vepron në emër të". Një sistem multiagjent është një sistem i projektuar dhe i zbatuar si disa agjent që bashkëveprojnë, dmth që agjentët bashkëpunojnë dhe negociojnë të koordinuar. Në sistemet multiagjente, çdo agjent ka informacion të paplotë mbi mjedisin ku jeton. Në këtë mënyrë mungon kontrolli global i sistemit. Të dhënat në këtë rastë janë të decentralizuara dhe llogaritja duhet të jetë asinkrone.

Elementët kryesore të sistemeve multi-agjente

Agjentët ndjejnë mjedisin e tyre lokal përmes stimujve. Ata kanë një gjendje të brendshme që është përcaktuar nga një grup i vlerave që karakterizojnë një shembull të veçantë të jetës së tyre. Agjentët kanë besim në lidhje me veten e tyre, agjentë të tjerë dhe mjedisin. Ata zotërojnë një grup të sjelljeve, të shkaktuar nga stimuj dhe gjendja e tyre e brendshme. Agjentët janë në gjendje të komunikojnë informacion me agjentë të tjerë në rrethana të veçanta. Struktura e sistemit është shumë dinamike, agjent të rinj mund të vijnë në lojë, të tjerët mund të pushojnë së ekzistuari, disa mund të ndryshojnë rolet, ndërsa kanalet e komunikimit në mes të agjentëve ri-konfigurohen me kalimin e kohës.

Një sistem i tillë strukturohet si një sistem elementësh që ndërveprojnë për të arritur objektivat e kërkesave të sistemit. Një arkitekturë bazuar në agjent përfshin:

- **Elementë Softuere** (elementë arkitekturale) që sigurojnë funksionalitetin e sistemit
- **Atribute të cilësisë** (performancë, përdorueshmëri, etj.) që arrihen nëpërmjet strukturës së arkitekturës softuere.
- **Për sistemet multiagjent elementët tipike arkitekturore janë** agjent, mjedis, burime, shërbime.

Për të modeluar një sistem bazuar në agjent specifikohet qëllimi i sistemit dhe më pas problemi për të cilin sistemi jep një zgjidhje**.** Sistemi analizohet duke identifikuar komponentët dhe marrëdheniet midis tyre si dhe burimet e të dhënave. Në fund aplikohet modeli që çon në eksperimentimin e një serie algoritmash duke ndryshuar në mënyrë sistematike parametrat dhe supozimet për të arritur rezultatin e dëshiruar. Elementi kryesor që mbart funksionalitetin e sistemit është agjenti. Për të ndërtuar një agjent, ekzistojnë metodologji të ndryshme. Ajo më e përdorshme është zhvillimi inkremental në një sërë hapash në inxhinierinë e proçeseve të soft-it.

Fillimisht identifikohet tipi i agjentit me atributet e tyre. Përcaktohet mjedisi ku do të jetojnë agjentët dhe me të cilin do të ndërveprojnë. Specifikohen se cilat atribute të agjentëve do të rifreskohen në përgjigje të ndërveprimeve agjent–agjent ose agjent–mjedis . Shtohen metoda që kontrollojnë se cili agjent ndërvepron dhe me kë ndërvepron, kur dhe se si ai ndërvepron gjatë simulimit. Hapi i fundit përfshin implementimin e modelit të agjentit në një softuere.

Avantazhet janë të dukshme në performimin e sjelljeve të thjeshta për agjentin. Arkitektura e agjentit është një projekt i ripërdorshëm, e cila mund të implementohet në një tjetër aplikacion që mbështet detyrën mbi këtë model. Abstraksioni i përfituar është i pavarur nga fusha e aplikacionit . Akoma më tej , në dekompozimin e trajtuar gjenden një bashkësi komponentësh softuere tipi generic për paraqitjen e njohjes, dhe këto komponentë softuere janë të përbashkët të gjithë agjentët.

7.1 Rast studimi: Një sistem vetëmanaxhimi i informacionit bazuar mbi agjentë softuere

Për këtë rast studimi, ne përdorim arkitekturën bazuar në agjent dhe tentojmë të përshtasim atë me një mjedis tregu. Kjo arkitekturë përdor disa lloje agjentësh, duke filluar nga agjent të thjeshtë siç janë agjentët e informacionit të përcaktuar mirë për të vepruar dhe për të bërë veprime të veçanta të manaxhimit të informacionit. E veçanta e kësaj arkitekture është modulariteti: që do të thotë që, ne mund të shtojmë agjentë të tjerë që zgjerojnë funsionalitetet e sistemit. Ata nxjerrin dhe ofrojnë njohuri në kohë reale që mund të përdoret për të marrë vendime të shpejta dhe ky është një nga avantazhet më të rëndësishëm të sistemit. Sistemet inteligjente dhe sidomos sistemet të bazuar në agjent mund të ofrojnë mjetet e nevojshme për ruajtjen dhe ekspertizën e informacionit në një sistem të manaxhimit të bazës së të dhënave. [94]

Cilat janë problemet që zgjidh arkitektura e sistemit bazuar në agjentë bazuar në hipotezat e ngritura kur diskutuam teorikisht në kapitujt e parë?

- **Problemi 1:** Si mund të manaxhohet kompleksiteti i lartë në proçeset e inxhinierisë së soft-it? Cili është modeli i përshtatshëm për të zgjidhur këtë situatë?
- **Hipotezë 1**: Një model i bazuar në agjent i cili rrit abstraksionin dhe fsheh kompleksitetin. Problemi mund të dekompozohet në një numër komponentësh më të vegjël dhe më të thjeshtë. Këto komponentë janë më të lehtë në zhvillim. Secili komponent specializohet në zgjidhjen e nënproblemeve specifike.
- **Problem 2:** A ka një model zhvillimi për sistemet vetëmanaxhuese?

- **Hipotezë 2**: Stili i dekompozimit në një arkiteturë bazuar në sisteme multiagjent , model avantazhues për vetëmanaxhimin, ku agjentët përfaqësojnë një metodë unike për ta bëre një sistem modular.
- **Problem 3:** Kosto e lartë zhvillimi për sisteme komplekse me autokontroll?
- **Hipotezë 3:** Ripërdorimi i arkitekturave multiagjent me komponentë autonomë ose lidhur dobët është një zgjidhje efikase në proçesin e inxhinierisë softuere, sepse kemi të njëjtën arkitekturë multiagjent në sisteme të ndryshme aplikimi, dhe ku mjedisi i aplikacionit bën diferencën si komponent.

Metodologjia: simulimi

Simulimi si metodë e modelimit të proçeseve të inxhinierisë së soft-it është përdorur për të parashikuar proçeset softuere dhe për të bërë një analizë bazuar në parametra reale që shfaqin rezultate reale.

Modeli përdor një analizë specifikimi top-down duke mbështetur dekompozimin në rrjedhen e proçeseve për të lehtësuar përmirësimin dhe optimizimin e inxhinierisë së soft-it. Simulimi na lejon të realizojmë kontrollin e zhvillimit të aplikacionit në gjithë ciklin e jetës së tij. Kompleksiteti i sistemit në tërësinë e kërkesave mund të reduktohet duke përdorur simulimin duke ulur mjaft kostot e zhvillimit si dhe rrit cilësinë dhe besueshmërinë e produktit të marrë.

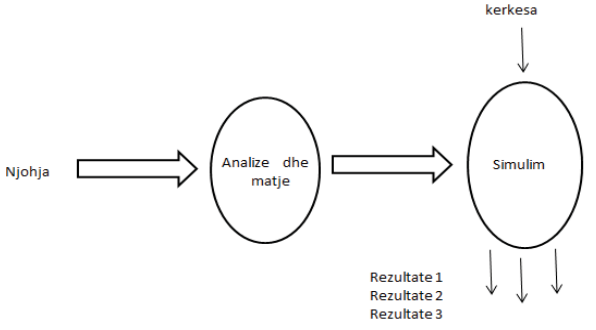

Figura 7. Modeli i simulimit si metodologji zhvillimi

Le të përcaktojmë për çdo problem të deduktuar simulimin përkatës:

- **Simulim 1:** Agjensi me katër agjent informacioni. Vetëmanxhim i një baze të dhënash në një mjedis supermarket për të arritur vendimmarrje në kohë reale.

- **Simulim 2:** Funksionalitete të reja të sistemit. Automatizim transporti me dy agjent robotike për uljen e kostove të manaxhimit të organizatës dhe rritjen e performancës së shërbimit.
- **Simulim 3:** Agjent robotik si zbulues dhe hartues i një mjedisi të panjohur i paaksesueshëm nga operatori njeri. Aplikacion i ri bazuar në të njëjtin model arkitekture duke përcaktuar mekanizmat e variancës për këtë aplikacion.

Simulimi 1: Sistem vetëmanaxhimi i bazuar në agjente informacioni

Aplikacioni synon në fazën e parë të zhvilloje kontekstin e vendosjes së sistemit vetëmanaxhues duke trajtuar marrëdhëniet mes agjentëve në sistemet e manaxhimit të të dhënave si dhe teknikat që bëjnë të mundur këtë bashkëpunim. Ne i konsiderojmë këto marrdhënie shumë të dobishme, sepse ne besojmë se agjentët e informacionit e bëjnë punën e tyre shumë më të shpejtë dhe shumë më mirë se çdo objekt tjetër. Gjatë studimit disa pyetje interesante dalin në lidhje me hulumtimet aktuale: A mund të gjejmë një model të mirë që mund të përdoret gjerësisht në aplikime të bazës së të dhënave? Mund të shtojmë shërbime të reja duke shtuar agjent të rinj pa kompromentuar përpunimin dhe kohën kur realizohen detyrat e agjentëve të tjerë? A mundemi të zhvillojmë në procesin gjetjes se zgjidhjeve më të mira një model të ri duke kombinuar agjentët dhe sistemet e manaxhimit të bazës së të dhënave?

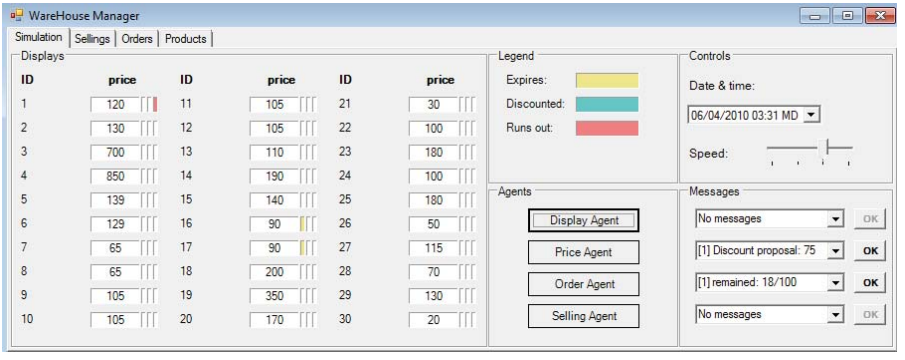

Figura 7. Ndërfaqe e simulimit me katër agjent informacioni

Në dritën e këtyre pyetjeve, ne fillojmë të zhvillojë një aplikacion që simulon një mjedis të biznesit në një vendosje marketi. Në figurën 7.2 është paraqitur kjo ndërfaqja e aplikacionit, të zhvilluar në fazën e parë. Ne do të vëzhgojmë përformancën e sistemit duke vëzhguar sjelljen e agjentëve. Mjedisi është një komponent softuere i mbrojtur nga agjentët dhe detaje të botës reale dhe siguron ndërfaqen për veprim, perceptim dhe komunikim me agjentët. Modelimi i një arkitekture softuere është një hap thelbësor për zhvillimin e sistemeve komplekse,

duke përfshirë sistemet multiagjent. Zgjidhja ideale është një zinxhir vlerash logjike me komponente të ndryshme të fokusuara në ofrimin e shërbimeve të nevojshme për trajtimin në kohë të informacionit të ndryshueshëm. Figura 7.3 paraqet konceptimin logjik të përgjithshëm të sistemit të vetëmanaxhimit bazuar në agjente të ndryshëm.

Në ditët e sotme, inxhinierë softuere duhet të perballen me disa probleme në zgjidhjen apo vendimmarrjen si dhe të gjejnë mënyra të reja, të shpejta dhe të lehta në fushën e zhvillimit të proçeseve softuere. Sasia e informacioni vazhdon të rritet dhe lista e kërkesave akoma me tepër. Ne kemi nevojë për strategji të reja në projekte inxhinierike. Nëse ne njohim atributet e entiteteve, ne mund të paraplanifikojmë duke përdorur njohjen ekzistuese të zhvendosjes me hapa.[96] Përdorimi i eksperiencës së kaluar dhe hartimi i planit të veprimit, na lejon të fitojmë kohë dhe të shmangim gabimet.

Në këtë aplikacion, ne kemi përdorur si agjentë softuere të sistemit të manaxhimit të bazës së të dhënave komponentët që lejojnë një sistem të tillë të mund të jetë konfiguruar dhe zgjeruar për të mbështetur kërkesa të reja Arkitektura softuere është një arkitekturë hibrid. Arkitektura bazuar në agjent është një zgjidhje në rritje të efikasitetit në proçesin e inxhinierise softuere. Sistemi paraqet një pasqyrë në proçesin e zhvillimit të sistemeve multiagjente në mënyrë që të promovojë agjentët inteligjentë si arkitektura të moduluara për modelimin e sjelljeve të thjeshta racionale në një gamë të gjerë të aplikacioneve softuere. Së pari, ne do të diskutojnë në lidhje me zhvillimin e proçeseve softuere dhe teknikat inxhinierike të dekompozimit të arkitekturës së sistemit.

Në sistemin tonë, elementët kryesor arkitekturor janë:

- **Mjedisi:** Një bazë të dhënash për një sistem manaxhimi të informacionit në një *warehouse*.
- **Agjensia:** Katër agjent informacioni të mirëpërcaktuar për të kryer veprime specifike.
- **Arkitektura e bazuar në agjent:** Kombinim i arkitekturës *Layered* dhe *Build in* për të marrë avantazhet e secilit model. Çdo agjent nxjerr dhe ofron njohje në kohë reale për të marrë vendime.

Agjensia realizon ekspertizën e bazës së të dhënave për sistemin e manaxhimit të informacionit për një *warehouse.*

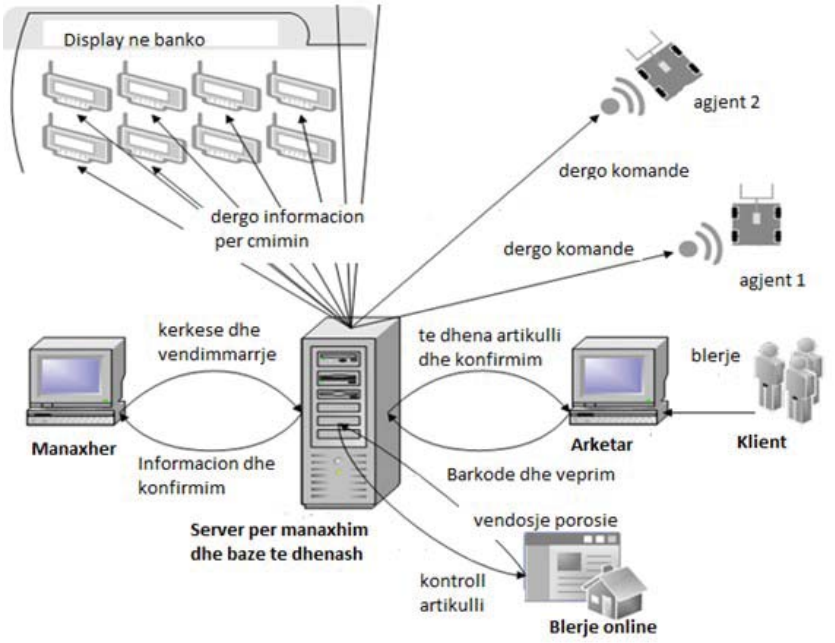

Figura 7. Bllokskema e përgjithshme e Sistemit të vetëmanaxhimit bazuar në agjent.

7.1.1. Mjedisi i aplikacionit

Zhvillimi dhe strukturimi i të dhënave është një proçes i vazhdueshëm, evolon në të njëjtën kohë me organizatën. Duke marrë parasysh këtë aspekt, projektimi dhe manaxhimi i tyre mund të çojë në humbje të informacionit të nevojshëm për vendimet të ardhshme strategjike dhe avantazh konkurrues të organizatës.

Aplikacioni i bazuar mbi këtë bazë të dhënash paraqet një aktivitet kompleks, duke përfshirë dy faza kryesore.

Në fazën e parë, të konfigurimit të sistemit, modeli konceptual i bazës së të dhënave është ndërtuar në përputhje me kërkesat e përdoruesve. Pastaj jane krijuar burimet e të dhënave, si dhe mënyra e nxjerrjes dhe ngarkimit të të dhënave (të dhënat e blerjes). Së fundi, teknologjia e magazinimit është zgjedhur dhe është vendosur në mënyrë që të dhënat do të duhet të arrihen dhe të jenë të aksesueshme. Zhvillimi i bazës së të dhënave duhet të marrë në konsideratë kërkesat e përdoruesve në lidhje me raportimin dhe analizimin. Përndryshe, kjo bazë të dhënash do të bëhet një "burg të dhënash", i cili fsheh informacione të rëndësishme. Për ta bërë këtë: janë zgjedhur të paktën katër proçese të biznesit për modelimin, për secilin proçes të

biznesit do të zhvillohet një agjent informacioni që manaxhon informacionin në këtë bazë duke aksesuar me të drejta leximi apo shkrimi modifikimi ose fshirjeje.

Megjithatë, nuk mund të mos të ketë probleme nëse kjo bazë ndërtohet pa ekspertizën e manaxherit për të ngritur një bazë të dhënash të qëndrueshme dhe funksionale si mjedis ku do të operojnë agjent autonomë. Ky projektim që ne e kemi përdorur në nisje të sistemit të manaxhimit , si zgjidhje mund të refuzohet nga linja e biznesit nëse ky biznes mund të dojë të përfshijë këtë prototip. Si qasje përshtatet me vizionin e [56] Baza e të dhënave është ndërtuar në Microsoft Access 2003 sipas modelit logjik të bazës së të dhënave për sistemin e manaxhimit të marketit. Tabelat e të dhënave dhe marrëdhëniet midis tyre jepen në figurën7.4.

7.1.2. Agjentët dhe sistemet e manaxhimit të bazave të të dhënave

Integrimi i dy teknologjive rrit mjaft kompleksitetin e sistemeve. Është e domosdoshme të zhvillojmë një arkitekturë që është fokusuar në gjetjen e një niveli të lartë të abstraksionit që fsheh kompleksitetin, pa pasoja të drejtpërdrejta. Megjithatë, një pikë e rëndësishme duhet përmendur se një pengesë të rëndësishme për integrimin e të dy teknologjive është mungesa e metodologjive për ndertimin e planeve dhe rregullave. Mjetet më të fuqishme për trajtimin dhe zhvillimin e programeve janë **modulariteti** dhe **abstraksioni**. Agjentët përfaqësojnë një mjet të fuqishëm për të bërë sistemet modulare. Nëse një fushë të caktuar problemi është veçanërisht kompleks, i madh ose i paparashikueshëm, atëherë mënyra e vetme që mund të arsyetohet është të adresohet te zhvillimi i një numri komponentesh modulare të cilat janë të specializuara (në aspektin e përfaqësimit të tyre dhe zgjidhjen e problemeve të thjeshta) apo në zgjidhjen e një

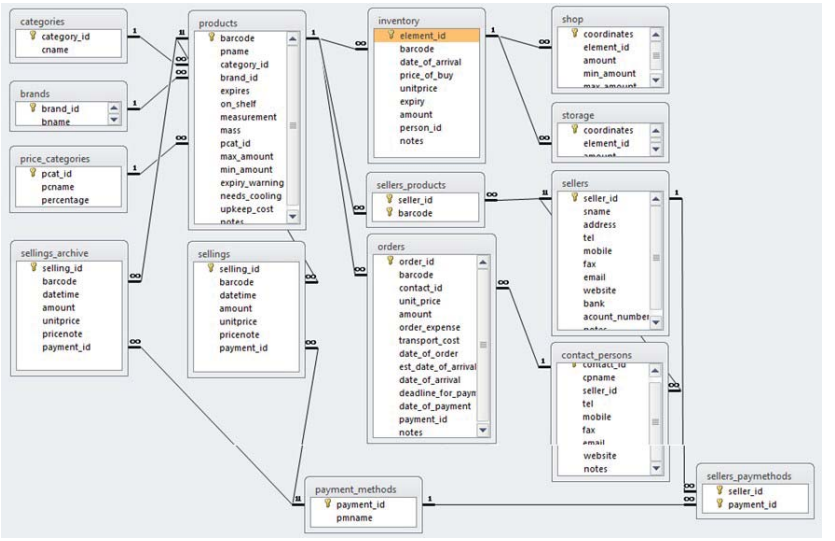

Figura 7. Baza e të dhënave për një organizim marketi si mjedis

aspekti të veçantë të tij. Në raste të tilla, kur problemet janë të ndërvarur, agjentët në sisteme duhet të bashkëpunojnë me njëri-tjetrin për të siguruar manaxhimin e ndërvarësisë siç duhet. Në fusha të tilla, një qasje bazuar në agjent do të thotë se problemi i përgjithshem mund të jetë ndarë në një numër të komponentesh të vogla dhe të thjeshta, të cilat janë më të lehta për të zhvilluar dhe ruajtur, dhe të cilat janë të specializuar në zgjidhjen e problemeve përbërëse.

7.1.2.1. Disa veçori të agjentëve të informacionit

Shtresa agjent përbëhet nga agjentë të informacionit në role të ndryshme. Agjentët nxjerrin informacion nga një proçes i kontrolluar sistem informacioni mbi një bazë të dhënash. Duke përdorur një ontologji të përbashkët të fushës, të dhënat heterogjene përpunohen në informacion të vlefshëm për përdoruesit. Arkitektura është eksperimentuar me skenarë të ndryshëm demonstrimi të motivuara nga sfidat e biznesit dhe ato të projektimit. Skenarët e kanë trajtuar informacionin nga burime të kombinuar heterogjene, monitorimin aktiv të sensorëve të proçesit të gjendjes operacionale të sistemit. Shtresa agjent konsiston në agjent informacioni që realizojnë detyra të ndryshme. Agjentët veprojnë si skuadër dhe ofrojnë shërbime të ndryshme duke zgjeruar gjendjen e njohjes së perdoruesit.

Një agjent i informacionit është një agjent që ka akses në të paktën një ose disa burimet potenciale të informacionit, dhe është në gjendje të krahasojë dhe të manipulojë informacionin e përfituar nga këto burime në mënyrë që t'i përgjigjet pyetjeve të paraqitura nga përdoruesit dhe agjent të tjerë të informacionit [97]. Burimet e informacionit mund të jenë shumë lloje, duke përfshirë për shembull, bazat e të dhënave tradicionale si dhe agjent të tjerë të informacionit. Një numër i studimeve kanë qenë të bërë nga agjentët e informacionit, duke përfshirë edhe një studim teorik se si agjentët janë në gjendje të përfshijnë informacion nga burime të ndryshme [98], si dhe një sistem prototip i quajtur IRA (rikthim informacion agjent) që është në gjendje të kërkojë për artikuj lirshëm nga një gamë e një baze të dhënash për dokumente [99]. Një sistem tjetër i rëndësishëm në këtë fushë lejon sistemet e bazave të të dhënave për të punuar së bashku për t'iu përgjigjur pyetjeve që janë jashtë fushëveprimit të të dhënave individuale.[100]

Në përgjithësi mund të themi se një agjent informacioni :

- Është një agjent softuere që lidhet ngushtë me një ose disa burime të dhënash
  - për qëllimet e përdoruesit njeri ("Agjent ndërfaqe"),
  - Proçese të përfshira në kryerjen e një detyrë ("Agjent detyrash").
- Ka akses në të paktën një ose disa burime potenciale të informacionit,

- Është në gjendje të krahasojë dhe të manipulojë informacionin e përfituar nga këto burime për t'iu përgjigjur pyetjeve të paraqitura nga përdoruesit dhe agjentët e tjerë të informacionit

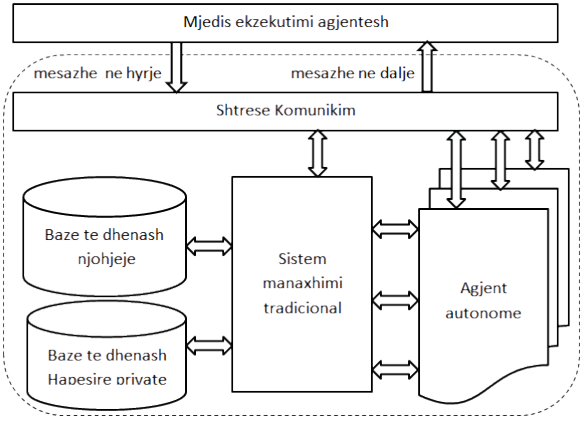

Figura 7. Agjent informacioni në sisteme me baza të dhënash

Agjenti i informacionit në baza të dhënash është i aftë të hyje në të dhëna dhe të sigurojë një këndvështrim multidimensional mbi informacionin te përdoruesi. Ai trajton ndryshimet në mjedisin ku operon. Percepton mjedisin dhe herë pas here ndryshon për të arritur objektivat e projektimit. Informacioni në këtë rast filtrohet sipas aktivitetit të përdoruesit.

7.1.2.2. Tre arkitektura për integrimin e Agjentëve dhe DBMS:

Ka tre arkitektura të integrimit mes agjentëve dhe DBMSs të propozuar: Layered, e integruar dhe Built-in. Çdo njëri nga tre arkitekturat e integrimit ka avantazhe dhe disavantazhe. Arkitektura e parë është e shtresëzuar dhe zbatohet në shume qasje ekzistuese. Një agjent i informacionit është çdo gjë që mund të shihet si mjedis përceptimi të tij nëpërmjet sensorëve dhe duke vepruar mbi atë mjedis nëpërmjet përpjekjeve të tij. [101] Një agjent informacioni është ai që ekzekuton detyra të tilla si perceptimet, analizon ato dhe në bazë të rezultatit, ai vepron pa kujtuar historinë e tij. Një pyetje është "si matet efektiviteti i një agjenti?" E pra kjo është shumë e vështirë për të parë një agjent dhe vlerësuar performancën e tij. Kjo është arsyeja pse, njeriu është ai që krijon një standard të asaj që do të thotë të jetë i suksesshëm në një mjedis dhe e përdor atë për të matur performancën e agjentëve. Arkitektura e përdorur vë agjentin mes ndërfaqes së perdoruesit dhe DBMS. Përdoruesit janë të përfaqësuar nga agjentët e tyre në shtresën e tretë. Qëllimi i agjentëve është të sjellë informacion individual te përdoruesit dhe mesazhet përkatëse aq më mirë sa të jetë e mundur. Për të përshtatur kërkesën e pronarit të tij të

informacionit, agjenti mbledh vlerësimet specifike nëpërmjet mesazheve të dhëna nga pronari i tij[102]. Agjentët komunikojnë nëpërmjet mesazheve dhe vlerësojnë informacionin e dhënë si zgjidhje për përdoruesit. Në mes të sistemit atje është një agjent ekzekutiv që ka rol të lehtësojë komunikimin midis agjentëve. Ai ka gjithashtu rolin për të vlerësuar performancën e agjentëve të tjerë dhe për të pranuar ose për të refuzuar rregjistrimin e një agjenti pranë agjencisë.

Le të rreshtojmë disa avantazhe dhe disavantazhe të arkitekturave të mbartura:

- **E shtresëzuar**
  - E lehtë për t'u implementuar
  - Mbart pak funksionalitete
- **E integruar**
  - Arrihet maksimumi i nivelit të agjencisë,
  - Komplekse në mënyrë ekstreme

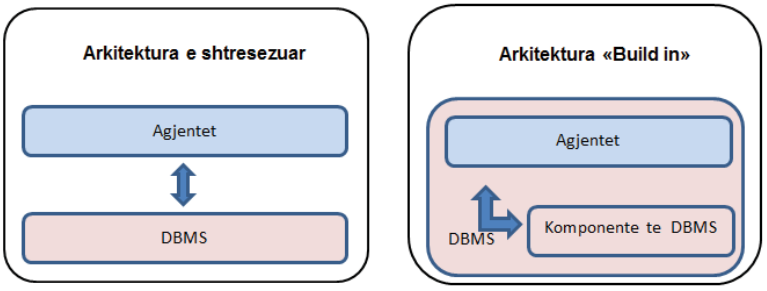

Figura 7. Dy arkitekturat e përdorura për integrimin e agjentëve në bazën e të dhënave

- **"Built-in"**
  - Mundëson ripërdorimin e komponentëve ekzistues të DBMS
  - Përmasat e shtrirjes së funksionaliteteve të DBMS varet në nivelin e çiftimit midis agjentëve dhe komponentëve të DBMS

Dy arkitektura bazë integrimi të agjentëve në DBMS për aplikacionin:

7.1.2.3.Agjensia softuere

Agjensia software eshte zhvilluar ne fillim VISUALStudio.NET 2003 ne gjuhen Visual Basic. Me vone eshte pershtatur dhe vazhduar ne VisualStudio 2010. Kjo agjensi perben sistemin e vetemanaxhimit per aplikacionin. Agjensia përbëhet

nga katër agjentë informacioni me detyra specifike te cilet ekzekutohen ne threade te pavarur nga njeri tjetri dhe nga sistemi kryesor.

Agjent 1: i quajtur *Display_agent* mundëson gjatë ekzekutimit të tij rifreskimin e të gjithë ekraneve qe afishojne çmimet për artikujt në bazën e të dhënave duke lejuar komunikimin me njësitë përkatëse harduere në adresat destinacion të artikullit të caktuar. Si agjent, ofron në kohë reale ndryshimet e ardhura nga ekzekutime të politikave manaxheriale nëpërmjet agjentëve të tjerë siç është Price_Agjent apo vetë përdoruesi. Kodi për këtë agjent është ndërtuar sipas një algoritmi të thjeshtë paraqitur në figurën përkatëse.

Algoritmi funksionon sipas pseudokodit në një instruksion të përsëritur vlerëdhënieje.

```
Kryej derisa  Butoni_Disply_agent == "I shtypur"
          Per i= 1 deri ne 30 kryej
              display[i]= cmimi_produkt[i];
```

Kodi i agjentit jepet si më poshtë :

```
Private Sub DisplayAgjentTask()
        Dim i As Integer
        Do While Me.btnDisplayAgjent.FlatStyle =
FlatStyle.Flat
            For i = 1 To 30
                ChangeDisplay(i,
DsWareHouse.inventory.FindByelement_id(i).unitprice)
            Next
            Thread.Sleep(700)
        Loop
    End Sub
```

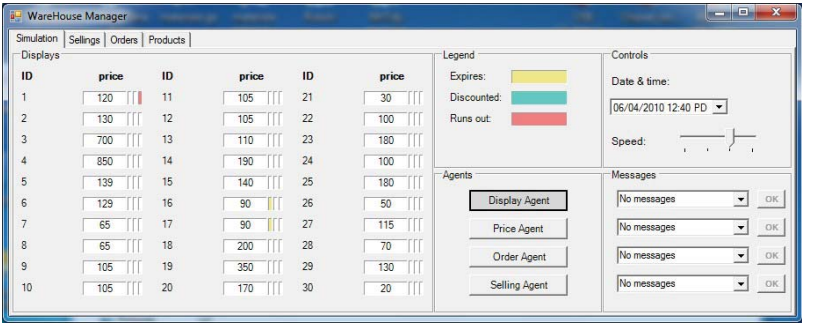

Figura 7. Ndërfaqe e aplikacionit me Display_agjent të aktivizuar

Agjenti 2: I quajtur *Price_agjent*, është edhe ky një agjent informacioni që ka një detyrë specifike paksa më komplekse se agjentët e tjerë. Ai arrin të implementojë në mënyrë automatike politika të biznesit që kanë të bëjnë me kontrollin e fluksit të shitjeve nëpërmjet ofertave promocionale që përpunohen me metoda formale matematikore të zbatuara. Transformimet matematikore mbështeten mbi disa parametra që kontrollohen dhe përpunohen nga agjenti për të arritur në një rezultat të sugjeruar tek përdoruesi. Për shkak të vetë llojit të sistemit me mision kritik për organizatën, ky shërbim është gjysmë autonomë dhe kufizohet vendimmarrja e agjentit duke u lejuar nga vetë përdoruesi.

Në transformimin formal, na duhet të llogaritëm me tre parametra kryesore si më poshtë:

```
Control_parameter                                    =
Daily_average(selling[i])*Expiry_date[i]-Today)-
Inventory[i]

Discount   =   (Daily_average(Selling[i])*Expiry_date[i]-
Today))/

                  (Inventory[i]Daily_average(Selling[i])*
Expiry_date[i]-Today)

Price[i] =  Price[i]*(1 – Discount)
```

Agjenti vepron vazhdimisht duke kërkuar vlerën e *Control_parameter* nëse ajo është pozitive apo negative. Parametri është llogaritur nga agjenti duke përdorur informacionin e mbledhur nga çdo rekord. Agjenti mund të zbulojë mjedisin e tij në një mënyrë të dytë të të perceptuarit që është : *Veprimi*. Ai dërgon kërkesa për DMBS dhe merr raporte nga databaza për tre variablave

1. A mundet te skadoje artikulli apo jo?
2. A ka eshte aplikuar ulje cmimi e meparshme?
3. A eshte njoftuar me pare manaxheri per ulje cmimi per çdo rekord

Nese keto asnje nga keto kushte nuk eshte i vertete, atehere vazhdon ekzaminimin e produktit perkates.

Ne fillim behet llogaritja e *Control_parameter* sipas formules se mesiperme duke perdorur informacionin mbi mesataren e shitjeve ditore te produktit (**Daily_average (shitje [i])),** daten e skadences se produktit **(Expiry_date [i])** si dhe gjendjen e sasise aktuale te produktit ne iventar **(Inventari [i]).**

Ne rastet kur *Control_parameter* eshte negativ, kjo do te thote qe kemi me shume sasi artikulli se sa sasia e parashikuar per t'u shitur brenda dates se skadences per

artikullin ne fjale. Ne kete rast llogaritet cmimi i ulur sipas formules se mesiperme per parametrin *Discount.*

Me pas, njoftohet manaxheri nepermjet nderfaqesper duke sugjeruar cmimin

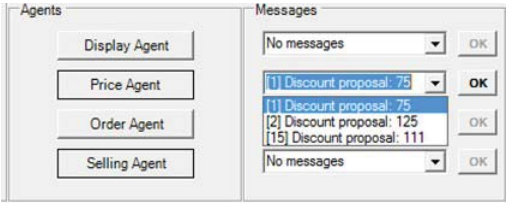

Figura 7. Mesazhe të gjeneruara nga Price_Agjent për operatorin

e ulur sic paraqitet ne figuren 32.

Algoritmi i ekzekutimit të këtij agjenti jepet si më poshtë:

```
Derisa Butoni_Price_agent == "aktiv"  kryej
per i = 1 deri ne Numer_artkuj
ne qofte se Artikull(i).LajmerimSkadenca<>null dhe
NukEshteLajmeruar(Artikull(i))
     atehere
          fillim
KontrolParameter=MesatarjaShitjesDitore(Artikull(i))*(Art
ikull(i).Skadenca-DataAktuale)-
Artikull(i).SasiNeMagazine;
          ne qofte se Artikull(i).Skadenca-DataAktuale>0
     dhe KontrolParameter<0
          Atehere
               Fillim
                    ÇmimiPropozuar=Artikull(i).Çmimi/Zbri
               tja
                    Afisho ÇmimiPropozuar
               Fund;
               Fund;
```

Agjenti ofron çmimin e ri, por ai nuk mund të vendosë për një vlerë të re të konfirmuar. Këtu është fundi i detyrës së agjentit dhe operatori mund të injorojë ose të vazhdojë vendimin e agjentit. Sistemi nuk është plotësisht i pavarur, sepse ka shumë faktorë të tjerë që e klasifikojnë atë si një sistem kritik për biznesin.

Kodi i agjentit që ekzekutohet pas aktivizimit, paraqitet si më poshtë :

```
    Private Sub PriceAgjentTask()
        Dim product As dsWareHouse.inventoryRow
        For Each product In DsWareHouse.inventory.Rows
            If (Not
DsWareHouse.products.FindBybarcode(product.barcode).Isexp
iry_warningNull) And
DsWareHouse.inventory.FindByelement_id(product.barcode).I
snotesNull And notyetmentioned(2, product.barcode) Then
                Dim ControlParameter, Discount As Decimal
                Dim Price As Integer
                ControlParameter =
DsWareHouse.DailyAvg.FindBybarcode(product.barcode).AvgOf
amountsum * DateDiff(DateInterval.Day,
Me.datePicker.Value.Date,
DsWareHouse.inventory.FindByelement_id(product.barcode).e
xpiry, FirstDayOfWeek.Monday) -
DsWareHouse.inventory.FindByelement_id(product.barcode).a
mount
                If DateDiff(DateInterval.Day,
Me.datePicker.Value.Date,
DsWareHouse.inventory.FindByelement_id(product.barcode).e
xpiry, FirstDayOfWeek.Monday) > 0 And ControlParameter <
0 Then
                    Discount =
DsWareHouse.inventory.FindByelement_id(product.barcode).a
mount /
DsWareHouse.DailyAvg.FindBybarcode(product.barcode).AvgOf
amountsum / DateDiff(DateInterval.Day,
Me.datePicker.Value.Date,
DsWareHouse.inventory.FindByelement_id(product.barcode).e
xpiry, FirstDayOfWeek.Monday)
                    Price =
Math.Floor(DsWareHouse.inventory.FindByelement_id(product
.barcode).unitprice / Discount)
                    Me.DsAgjentMessages.messages2.Addmess
ages2Row("[" + product.barcode.ToString + "] Discount
```

```
proposal: " + Price.ToString, product.barcode.ToString +
"," + Price.ToString, 2)
                    RefreshCombo2()
                    If Not Me.GetbtnOK2() Then
Me.SetbtnOK2(True)
                End If
            End If
        Next
    End Sub
```

Modeli i marrë jep një kornizë agjenti dhe ka një numër avantazhesh që vijnë nga bota e inteligjencës artificiale dhe arkitektura standarde bazuar në agjent. Modeli garanton një arkitekturë gjerësisht në dispozicion të mjedisit, të mbështetur edhe nga ekzekutimi. Në figurën më poshtë shihen qartë tre propozime të sugjeruara për operatorin për të ekzekutuar ndryshimet në bazën e të dhënave për çmimet e reja. Operatori nëpërmjet butonit OK në te djathtë mund të jap pëlqimin ose jo. Nëse ai lejon ndryshimet e sugjeruara, atëherë është agjenti Display_agjent që ekzekuton ndryshimet që shfaqen pranë artikujve në shitje. Çmimet promocionale dallohen nga ato bazë për shkak të ndryshimit të ngjyrës. Kjo është bërë për efekte të tërheqjes së vëmendjes së klientit mbi ofertat që bën organizata.

Agjenti 3: *Order_agjent* është një tjetër agjent informacioni që përpunon informacionin në bazën e të dhënave mbi bazën e informacionit të ardhur nga lexuesi i barkodit në arkat e marketit. Agjenti përllogaritë sasitë e shitura për çdo artikull dhe , mbi bazën e një algoritmi, gjeneron mesazhe të manaxheri për nevoja furnizimi për një artikull të veçantë.

Ky mesazh sugjeron furnitorin, adresën, numrin e telefonit si informacione të nevojshme për të kryer porosinë përkatëse. Kjo lehtëson manaxherin në marrjen e informacionit në kohë reale dhe mbajtjen e gjendjes së marketit të rifreskuar.

Algoritmi për këtë agjent jepet si më poshtë:

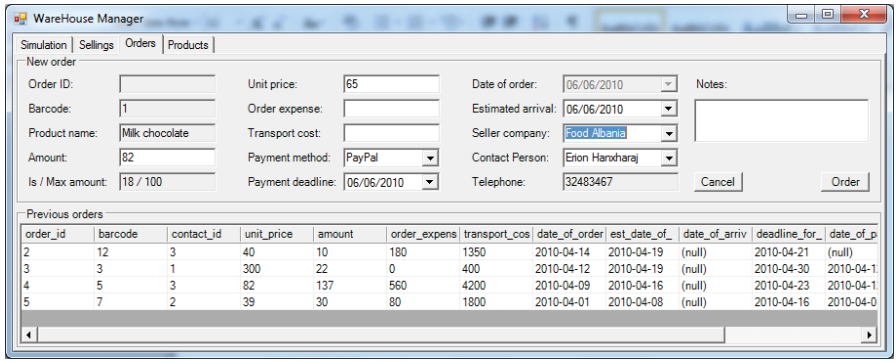

Figura 7. Simulimi i ofertës për porosi të gjeneruar nga Order_Agjent

```
Derisa Butoni Order_agent = aktiv kryej
per i = 1 deri ne Numer_artikuj
                ne                qofte                se
Artikull(i).SasiNeMagazine<Artikull(i).SasiMinimale
          atehere
          Fillim
                    ne                qofte                se
               Nuk_Eshte_Lajmeruar(Artikull(i))
               atehere
                Fillim
               Afisho(Artikull(i).SasiNeMagazine,
               Artikull(i).SasiMaksimale)
                Fund;
                NdryshoNgjyren(Display(i))
          Fund;
```

Aktivizimi i agjentit të porosive bëhet në ndërfaqen e përbashkët të agjensisë , duke startuar ekzekutimin e kodit:

```
   Private Sub OrderAgjentTask()
        Dim product As dsWareHouse.inventoryRow
        Do While Me.btnOrderAgjent.FlatStyle =
FlatStyle.Flat
            For Each product In
DsWareHouse.inventory.Rows
```

```
                If product.amount <
DsWareHouse.products.FindBybarcode(product.barcode).min_a
mount Then
                    If notyetmentioned(3,
product.barcode) Then
                        Me.DsAgjentMessages.messages3.Add
messages3Row("[" + product.barcode.ToString + "]
remained: " + product.amount.ToString + "/" +
DsWareHouse.products.FindBybarcode(product.barcode).max_a
mount.ToString, product.barcode.ToString, 3)
                        RefreshCombo3()
                        If Not Me.GetbtnOK3() Then
Me.SetbtnOK3(True)
                    End If
                    ChangeInventoryDisplayColor(product.b
arcode, True)
                End If
            Next
            Thread.Sleep(700)
        Loop
    End Sub
```

Agjenti 4: *Selling_agjent* simulon shitjen e artikujve duke zbritur në mënyrë të çfarëdoshme numrin e sasive të artikujve të rregjistruar në bazën e të dhënave të marketit. Ai akseson bazën e të dhënave më të drejtë të ndryshimit të sasisë me hap 1 dhe shpejtësia e ndryshimit është e rregullueshme nëpërmjet rrëshqitësit *Speed* në ndërfaqen e simulimit.

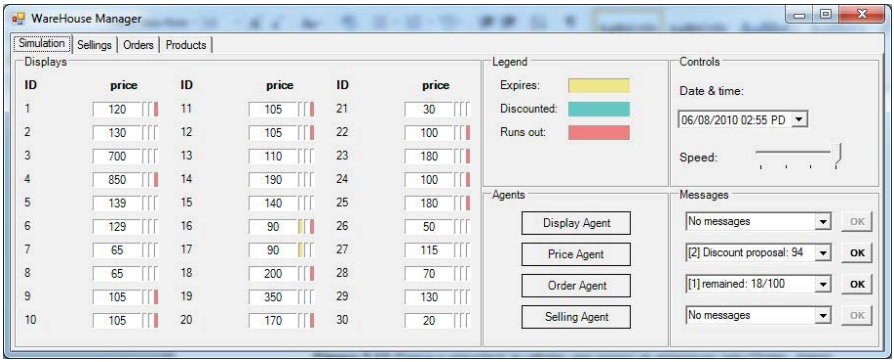

Figura 7. Simulimi i shitjeve nga Selling_agjent

Informacioni i ndryshuar nga ky agjent sjell ndryshime nga agjenti i gjenerimit të porosive. Pa këtë agjent, simulimi nuk mund të ekzekutojë shitjet në mjedisin real. Algoritmi i shitjeve është mjaft i thjeshtë dhe është paraqitur në figurën me shënimin përkatës.

```
Derisa Butoni_Selling_agent = "aktiv" kryej
    NrProdukti=Random(1,NrArtikuj);
    ne qofte se Artikull(NrProdukti).SasiNeMagazine>0
atehere
        Fillim
        Shto(Shitjet,NrProdukti,DataKoha,Shenime,Artiku
    ll(NrProdukti).Çmimi)
        Artikull(NrProdukti).SasiNeMagazine=
        Artikull(NrProdukti).SasiNeMagazine-1
        Fund;
```

Këto agjent që ekzekutohen si thread-e të pavarur, janë autonomë duke qëndruar në një makinë të vetme ose edhe në një mjedis të shpërndarë dhe mund të komunikojnë dhe të bashkëpunojnë me njëri-tjetrin.

Kodi i agjentit të shitjeve jepet si më poshtë:

```
Private Sub SellingAgjentTask()
        Dim productnr As Integer
        Dim rnd As New Random
        Dim checkTab As New CheckStateDelegate(AddressOf
CheckTabControl1State)
```

```
        Do While Me.btnSellingAgjent.FlatStyle =
FlatStyle.Flat
            productnr = rnd.Next(1, 31)
            If
DsWareHouse.inventory.FindByelement_id(productnr).amount
> 0 Then
                Dim notes As String
                If
DsWareHouse.inventory.FindByelement_id(productnr).Isnotes
Null Then
                    notes = ""
                Else
                    notes =
DsWareHouse.inventory.FindByelement_id(productnr).notes
                End If
DsWareHouse.sellings.AddsellingsRow(DsWareHouse.products.
FindBybarcode(productnr), Me.datePicker.Value, 1,
DsWareHouse.inventory.FindByelement_id(productnr).unitpri
ce, notes,
DsWareHouse.payment_methods.FindBypayment_id(1))
                DsWareHouse.inventory.FindByelement_id(pr
oductnr).amount =
DsWareHouse.inventory.FindByelement_id(productnr).amount
- 1
                If Me.CheckTabControl1State(2) Then
                    Me.grdSellings.CurrentRowIndex = 0
                End If
            End If
            Thread.Sleep(sleeptime)
        Loop
    End Sub
```

7.1.2.4. Diagrama e rrjedhës së të dhënave

Rasti studimor do të tregojë se zhvillimin e një sistemi për manaxhimin e agjent të bazuar informacionit do të jetë shumë i dobishëm. Në një mjedis të tregut të

marrëdhënieve midis produkteve, klientët dhe shitësit kanë një shkëmbim të vazhdueshëm të informacionit, ku kërkesa kryesore është garancia e nivelit të lartë të performancës së shërbimit.

Këtu ne paraqesim diagramin e rrjedhës së të dhënave të sistemit bazuar në agjentë. Sistemi është i bazuar në skedarët e të dhënave të cilat marrin të gjithë informacionin. Agjencia e përbërë nga katër agjentë informacioni është përfshirë për të administruar bazën e të dhënave. Çdo agjent nevojitet për të kryer veprime që zbulojnë ndryshimet në mjedisin e vet. Agjentët mund të perceptojnë duke përdorur pyetje (veprim). Komponentët e DBMS manaxhojnë këmbimin e informacionit në mes të agjentëve dhe bazës së të dhënave.

Agjentët e administrimit softuere ndajnë informacionin e bazës së të dhënave për sistemin e manaxhimit të të dhënave. DBMS (data softuere) manaxhon të dhënat midis agjentëve dhe bazës së të dhënave në mënyrë statike. Mbi bazën e grupeve të kërkesave të identifikuara për mjedisin në fjalë, çdo agjent mbulon përkatësisht:

1. Njohje të shitjeve dhe inventarin
2. Paraqitje të ndryshimeve në çmime për zonë artikulli.
3. Njohje të sasive, furnitorëve dhe kohës për porosi
4. Paraqitje të pikave të shërbimit
5. Informacioni për manaxherin vjen në dy modalitete: *off line* dhe *on line.*

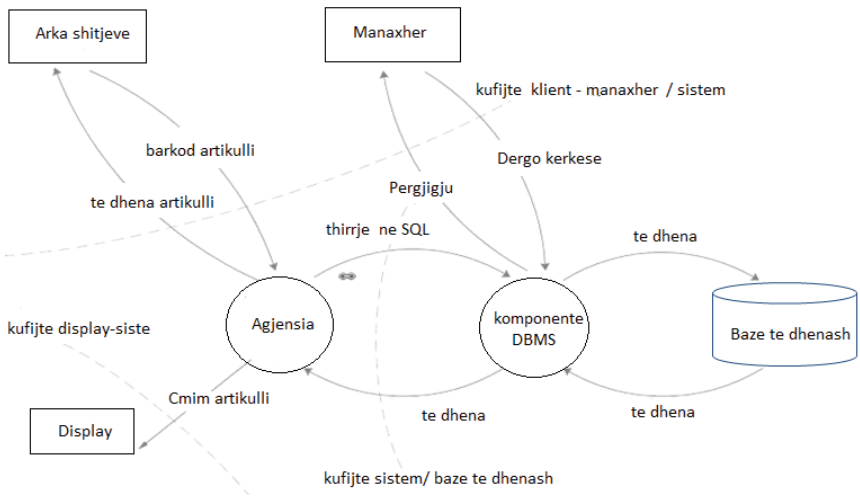

Figura 7. Diagrama e rrjedhës së të dhënave (DFD) për sistemin bazuar në agjent.

Ne ndajnë modulin Administrate Softuere në këto funksionalitete të perbëra duke zhvilluar katër agjentët të pavarura. Figura 7.11 tregon DFD e sistemit. Manaxheri ka nevojë për informacion në dy mënyra: off line dhe on line. Çdo agjent i aktivizuar jep shërbime dhe ofron sugjerime për çmimet ose krijon porosi ose urdhëra nga zbulimi i zonave që kanë nevojë , shërbimi është vigjilent për çdo shënim, ose bën raportet e kërkuara, jep zgjidhje të furnizimit, tregon edhe pikat ku shërbimet nga operatori njeri janë të nevojshme. Për shembull, një agjent i shtuar vizualizimi ofron të dhëna për t'i shpërndarë në një rrjet duke paraqitur një hartë të koordinatave për çdo ID të një produkti të caktuar.

7.1.2.5. Përmbledhje

Në fund të kësaj faze japim disa konsiderata. Ky dokument paraqet një model të arkitekturës së sistemit të manaxhimit të informacionit që zbaton përfitimet e përdorimit të teknikave bazuar në agjent. Në proçesin e studimit të arkitekturave të ndryshme, ne kemi kombinuar arkitekturën e shtresëzuar me atë Build in në funksion të rritet niveli i abstraksionit.

- Agjensia funksionon mbi katër agjent informacioni të pavarur që ekzekutohen si thread-e me proçedura përkatëse. Rezultatet jepen në kohë reale duke rifreskuar gjendjet për ndryshime të dedektuara

- Agjentët janë të kontrollueshëm për manaxherin duke kaluar në modalitete të ndryshme të sistemit. Besueshmëria e agjensisë testohet nëpërmjet modalitetit të dyfishtë në ekzekutim duke kontrolluar rezultatin e barabartë të agjentit.

- Mospërputhja e testeve automatikisht bllokon agjentin dhe rifreskon kodin në gjendjen e impostuar në një pikë kontrolli.

Ne përdorim nje metodë unike për të zhvilluar agjentët e pavarura të informacionit, ku çdo agjent ka një detyrë të veçantë për të përfunduar. Agjentët veprojnë të pavarur, megjithatë ata mund të bashkëpunojnë me përdoruesit. Ne mësuam mbi shpërndarjen e funksioneve në një sistem të bazës së të dhënave. Është një model që mund të zgjidhë shumë mirë kompleksitetin e DBMS-ve duke përdorur agjentët e informacionit si mënyrë e lehtë për të mbështetur shërbimet komplekse të bazës së të dhënave. Ne kemi zhvilluar katër agjentë informacioni në zbatimin e funksioneve të kërkuara të cilët janë testuar. Rezultatet e dhëna nga ekzekutimi i simulimit konfirmojnë vlefshmërinë e përdorimit të modelit. Ky testim është i rëndësishem sepse kjo tregon se agjentët inteligjent do të jenë teknologjitë më të mira që do të çojnë në përmirësime të rëndësishme në cilësinë dhe sofistikimin e sistemeve softuere. Aftësia e agjentëve të pavarur të planifikuar për të ndjekur qëllimet dhe veprimet e tyre për të bashkëpunuar, koordinuar, dhe të negociojnë më të tjerët, dhe për t'iu përgjigjur në mënyrë fleksibël dhe inteligjente në situata dinamike dhe të

paparashikueshme, do të zgjerojë përdorimin e tyre fuqishëm në shumë aplikacione të tjera vetëmanaxhuese.

KAPITULLI 8

Zhvillimi inkremental i Sistemit Multiagjent

8.1. Një sistem autonom i manaxhimit të informacionit me shërbim autotransport.

8.1.1. Kërkesat funksionale dhe veçoritë emergjente të nënsistemit

Zhvillimi inkremental sjell evolim të sistemit si një fazë të veçantë të ciklit të jetës së softit, mjaft i njohur në inxhinierinë e proçeseve të softit. Në këtë fazë mund të zhvillohet sistemi sipas modelit evolucionar apo të mund të shtohen funksionalitete të reja. Kjo bëhet e mundur në sajë të përdorimit të stilit të dekompozimit të zhvilluar mbi arkitekturën e sistemin tonë bazë.[103]

Gjatë kësaj faze, objektivat kryesore janë shtrirë në projektimin dhe zhvillimin e një nënsitemi që sjell:

- Shërbim i automatizuar bashkangjitur sistemit.
- Entitete autonome që implementojnë detyra specifike në të njëjtin mjedis
- Agjentë robotikë për të simuluar një shërbim transporti të automatizuar

Aplikacioni ka disa veçori emergjente të cilat vijnë nga kërkesat mbi këtë sistem:

- Sistemit i shtohet një nënsistem gjysmë autonom në një mjedis pjesërisht të njohur. Ky komponent do të bazohet mbi arkitekturën e sistemeve multiagjentë robotikë me sistem kontrolli qëndror dhe decentralizim të kontrollit në raste të ndryshimit të gjendjes statike të mjedisit. Kontrolli i centralizuar lejon hapësira për autonomi të sjelljes së agjentëve në situatë të paqarta të mjedisit si komponent lidhës.
- Sistemi performon shërbime të auto-transportimit bazuar në agjentë robotikë. Këto agjentë kanë elementët e tyre të përbashkët që lejojnë zhvillim të shpejtë dhe tashmë të testuar të këtyre komponentëve. Koordinimi dhe bashkëveprimi në situata të parashikuara kontrollohet nga sistemi i manaxhimit të informacionit.
- Adresat e pikave të shërbimeve për secilin agjent gjenerohen nga sistemi i manaxhimit të informacionit për organizatën në mënyrë të

çfarëdorshme simuluar nga agjenti i shitjeve. Agjentët kanë një vendqëndrim statik ku kthehen në përfundim të detyrës.

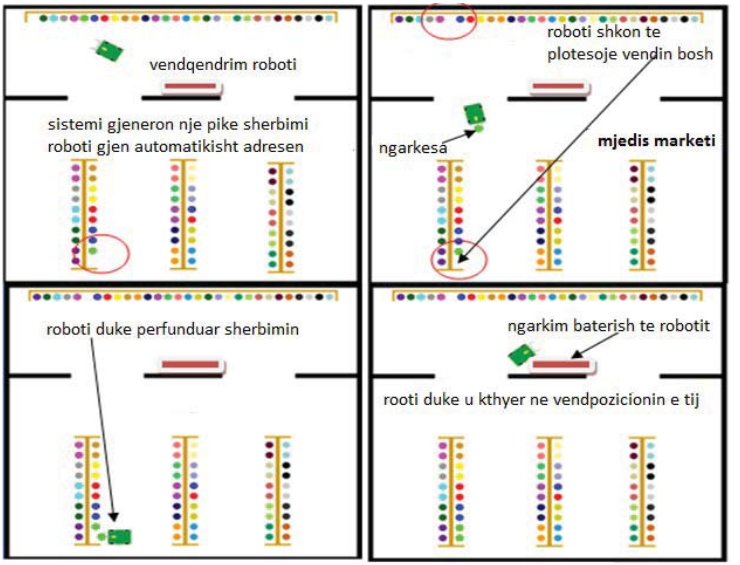

Figura 8. Ndërfaqe e simulimi të nënsistemit të autotransportimit

- Agjentët komunikojnë në një komunikim wireless të simuluar me sistemin kryesor duke marrë instruksione dhe duke dërguar të dhëna për gjendjen e tyre prezente.

8.2. Konceptimi në dekompozim i nënsistemit.

Figura 8.2 është një paraqitje konceptuale e dekompozimit të agjentit si entitet autonom në mjedisin e sistemit në të cilin ai shërben. Modeli i dekompozimit të aplikacionit, i trajtuar në pjesën e kapitullit të katërt, është zhvilluar në këtë nënsistem duke iu përmbajtur parimeve themelore të arkitekturës për sistemet multiagjentë.Një sistem softuer multiagjent ofrohet për të zgjidhur një problem me anë të strukturimit të sistemit në një numër entitete autonome vendosur në një mjedis në mënyrë që të arrihen kërkesat funksionale dhe të cilësisë të sistemit. Në veçanti, një sistem multiagjent strukturat e sistemit shihen si një numër elementësh që bashkëveprojnë për të arritur kërkesat e sistemit. Kjo është pikërisht ajo që përcakton arkitekturën softuere me elementët përbërës si:

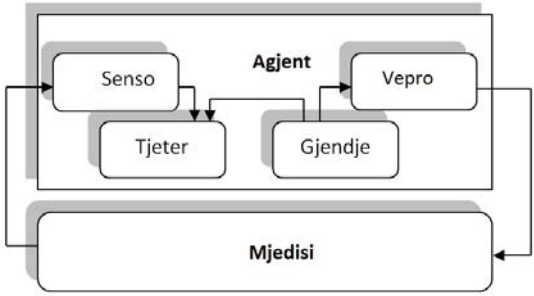

Figura 8. Bllokskema e dekompozimit për agjentin e transportit

- **Elementët softuere** (ose në elemente arkitekturore të përgjithshme) sigurojnë funksionalitetin e sistemit, ndërsa atributet e kërkuara të cilësisë (performanca, përdorshmëria, modifikueshmeria, etj) janë arritur kryesisht përmes strukturave të arkitekturës softuere.

- Elementë arkitekturore tipike të arkitekturës softuere multiagjentë të sistemit janë agjentë, mjedisi, burime, shërbimet, etj ndërkohë që marrëdhëniet ndërmjet elementëve janë shumë të ndryshme.

Me pak fjalë, sistemet multiagjentë janë një familje e pasur modelesh arkitekturore me karakteristika të veçanta, të dobishme për një shumëllojshmëri sfiduese të fushave të aplikimit. Ka aplikime që ofrojnë nivele të ndryshme të kompleksitetit dhe forma të ndryshme të dinamikës dhe ndryshimit. Çdo agjent ka informacion jo të plotë ose aftësi jo të plota për zgjidhjen e problemit në tërësi dhe, si rrjedhim ka një këndvështrim tëi kufizuar duke kërkuar që llogaritjet të jenë asinkrone.[104]

Në figurën 8.3 jepet bllokskema konceptuale e evolimit të sistemit kryesor. Çfarë është sistemi i manaxhimit të informacionit që gjeneron kërkesën për detyrë. Ka nje sistem kontrolli softuere realizuar nëpërmjet një simulatori që përcjell instruksionet e nevojshme të agjentëve të nevojshme për të realizuar detyrën e gjeneruar. Ka një ndërfaqe që vendos komunikimin me operatorin për kontroll manual në situata ekstreme.

Detyrat gjenerohen nga një sistem kontrolli i sherbimit i cili është pjesë e sistemit qëndror të informacionit. Detyra dekompozohet në disa proçese si marrja urdhër nga agjent të shërbimit, proçesi i simulimit të blerjes, përcaktimi i adresave përkatëse për çdo artikull, etj.

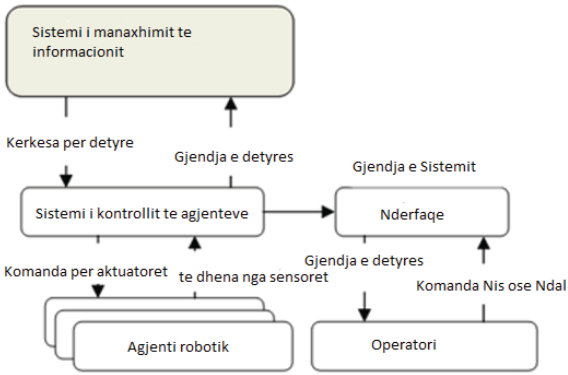

Figura 8. Arkitektura e evolimit të sistemit të autotransportimit

Shperndarja e detyrës bëhet me këtë proçedurë:

```
PrivateSub RobotAgentTask()
Dim product AsdsWareHouse.inventoryRow
DoWhile RobotsActive
ForEach product In DsWareHouse.inventory.Rows
If product.amount < 
DsWareHouse.products.FindBybarcode(product.barcode).min_a
mount Then
IfNot RobotTasks.Contains(product.barcode.ToString) Then
Try
                            RobotTasks.Add(True, 
product.barcode)
Me.flashRobots.FlashVars = "targetFromHQ=" + 
(product.barcode - 1).ToString
Catch ex AsException
                            MsgBox(ex.Message)
EndTry
EndIf
EndIf
Next
Thread.Sleep(700)
Loop
```

```
EndSub
```

Në mënyrë që të kryej këtë detyrë, sistemi i kontrollit i duhet të kryejë:

- **Caktim i rrugës**: rrugët janë të krijuara nga sistemi softuere i simulimit dhe duhet t'i caktohet agjentit robotik që është i aftë ta ekzekutojë atë.
- **Rutimi**: roboti orientohet në mjedisin që simulohet kur ekzekuton transportin. Pozicioni i tij rifreskohet në mjedisin e simulimit në rastet e ndryshimeve të gjendjes statike të mjedisit.
- **Informacion mbi Trafikun** në mjedis në përgjithësi është dinamik megjithëse gjendja e sistemit është statike, rruga e mirë për robotin varet dhe në gjendjen aktuale të trafikut. Duke përdorur sensorë vizualë, robot gjejnë rrugët efikase pa përplasje me objekte të tjera.

Nënsistemi përbëhet nga mjedisi i populluar me një grup të agjentësh robotike që bashkëpunojnë për të zgjidhur problemin e transportit në mënyrë të decentralizuar. Sjellja e sistemit që zhvillojmë bazohet në:

1. Veprime navigimi të orientuar nga një plan pathi paraprak e nxitur nga instruksione të ardhur nga sistemi i kontrollit të agjentëve dhe
2. stimuj të perceptuar në mjedis si dhe stimuj të brendshëm.

Konceptimi logjik i nënsistemit përfshin formulimin me pak fjalë se:

Agjentët robotike punojnë bazuar në gjendjen e brendshme të sistemit dhe vendimmarrja e tyre lidhet me:

- Planifikimin *off line* (informacion statik i sistemit);
- Planifikimin *on line* (informacion dinamik lidhur me ndryshimet e mjedisit), apo çështje të brendshme të agjentit.
- Mjedisi enkapsulon burimet dhe mundëson agjentët për të hyrë në burimet.

8.2.1. Dekompozimi në nivel të parë.

Sistemi që ne propozojmë, është një prototip i cili transporton ngarkesa nga një pikë në një tjetër duke manaxhuar gjetjen e pikave të shërbimit në një mjedis pjesërisht të njohur. Ky model integron mjedisin dhe agjentin si dhe realizon integrimin e mekanizmave të adaptimit për agjentët. Ne kemi ndarë modelin në dy pjesë: mjedisi dhe agjent robotik. Moduli Mjedis përbëhet nga një grup i moduleve me flukse informacioni midis tyre. Modulet përfaqësojnë funksionalitete themelore të

mjedisit. Modeli përbëhet nga tre module kryesore duke filluar nga poshtë sipas nivelit të kompleksitetit:

• Konteksti i vendosjes (referuar harduere dhe softuere të caktuar dhe të burimeve të jashtme me të cilat sistemi multiagjent ndërvepron (sensorë dhe aktuatorë, printer, rrjeti komunikimit, baza e të dhënave, etj.) dhe

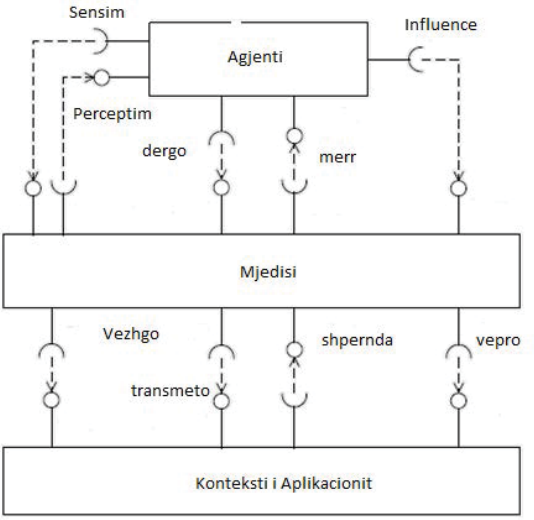

Figura 8. Modeli i dekompozuar në nivel të parë

• Mjedis i aplikacionit. (i referohet pjesës së mjedisit që është projektuar për aplikacionin).

• Agjentët robotike që performojnë këto sjellje kërkohet të përfundojnë me sukses detyrat e veta, duke shmangur përplasjen me pengesa që shfaqen në mënyrë dinamike apo një situatë e papritur në rrugën e tyre, e cila është paraplanifikuar nga një softuere simulator.

8.2.1.1. Konteksti i vendosjes dhe Mjedisi virtual

Konteksti i vendosjes konsiston në një rrjet që bën të mundur adresimin e vendndodhjes së secilit artikull me një ID në bazën e të dhënave. Këtij konteksti i bashkangjitet një rrjet elektronik afishimi i cmimeve dhe të dhëna të tjera të artikullit , si dhe elementë të tjerë që mund të shtohen në këtë kontekst. Mjedisi virtual ku operon agjenti, përcakton kontekstin e vendosjes për një aplikacion të veçantë në simulator.

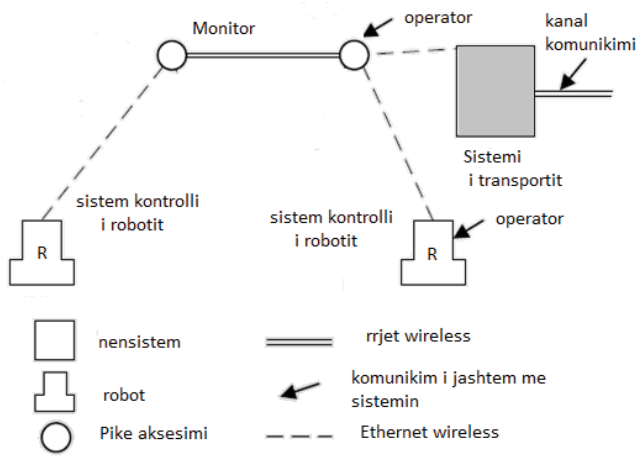

Figura 8. Konteksti i vendosjes për sistemin e autotransportimit

Përse është i nevojshëm përcaktimi i kontekstit të vendosjes? Përpara se të zgjedhim tipin e agjentit që do të adoptojmë, duhet të japim përcaktime të mjedisit ku do të jetojë agjenti. Është e kuptueshme që për një mjedis të varfër, një agjent elementar do të ishte i favorizuar në aspektin e orientimit. Ndërkohë që për të ndërvepruar me një mjedis kundërshtues është e nevojshme të zgjidhet një agjent më i sofistikuar.

Konteksti i vendosjes shërben si një ndërmjetës midis agjentit dhe mjedisit të aplikacionit sepse agjentit dhe mjedisit u nevojitet të komunikojnë nëpërmjet proçeseve që zhvillohen mbi kontekstin e vendosjes.

Duke parë tërësinë e mjediseve dhe sipas problemit që kërkojmë të zgjidhim, është e natyrshme të diskutohet mbi nivelet e inteligjencës për agjentin e zgjedhur, cili prej tyre arrin të kryejë detyrën e caktuar në mënyrë optimale. Arkitektura e agjentit mund të ndryshojë , por ajo që e bën përcaktues është programi i agjentit i cili mbart një proçes analize komplekse që i lejon atij të dallojë veprimet e duhura në bazë të të dhënave nga perceptimi. Zgjidhja ideale do të ishte nëse do të dihej cila sekuencë perceptive do të fitohet nga agjenti gjatë gjithë proçesit. Ky do të ishte një agjent i privuar nga autonomia megjithëse ideal, pasi në realitet nuk është e mundur të parashikohen të gjitha sekuencat perceptive. Nga ana tjetër të gjitha përpjekjet për të parashikuar të gjitha situatat do të rëndonin mbi kodin duke sjellë një sistem të ngadalshëm dhe jo të besueshëm.

Cila do të ishte zgjedhja? Sjellja e agjentit do të përcaktohet nga një bashkësi rregullash kusht- veprim. Nuk është mjaft racional pasi vepron mbi bazën e reflekseve të thjeshta (**simple reflex agjent**). Nëse veprimet nuk merren vetëm në bazë të perceptimeve prezente por edhe të atyre të mëparshme, do të thotë që agjenti

përditëson njouritë e tij dhe kjo është pjesë e rëndësishme e tij që ka të bëjë me evolucionin mjedisor (**agjent keeping track of the world).**

Nëse veprimet e agjentit nuk varen vetëm nga gjendja e mjedisit dhe nga sekuencat perceptive por edhe nga objektivi, për më tepër nga mënyra më bindëse si ai arrin këtë objektiv (në kohën më të shkurtër, me përpjekjen më të vogël apo koston më të ulët), kemi të bëjmë me një agjent të tipit **utility-based.** Që të mund të komunikojnë, agjentët dhe sistemi duhet të koordinojnë me njëri-tjetrin. Agjentët duhet të koordinojnë për kursin e mbajtur, përcaktimin e transportit, për shmangien e përplasjeve, etj. Një model është që të sigurojë një infrastrukturë për komunikim që mundëson agjentët për të shkëmbyer mesazhe për të koordinuar sjelljet e tyre. Një qasje e tillë megjithatë, do të jetë komplekse në funksion të koordinimit të agjentëve dhe e tillë do të rezultojë në një arkitekturë komplekse të agjentëve, në veçanti për agjentët robotikë. Ne kemi zgjedhur një zgjidhje që mundëson agjentët të shfrytëzojnë mjedisin për të koordinuar sjelljen e tyre. Kjo qasje ndan përgjegjësitë në sistem dhe ndihmon manaxhimin e kompleksitetit.

Roboti është vendosur në një mjedis fizik, megjithatë, ky mjedis është shumë i kufizuar: Roboti nuk mund të manipulojë mjedisin, përveç se të lëvizë, të kap dhe të lëshojë ngarkesa. Kjo kufizon mënyrën se si agjentët të mund të shfrytëzojnë mjedisin e tyre. Ne prezantojmë një mjedis virtual që në këndvështrimin tonë është një medium për agjentët robotikë për të shkëmbyer informacion dhe për të koordinuar sjelljen e tyre.

Përveç kësaj, mjedisi virtual shërben si një abstraksion i përshtatshëm që mbështet organizimin e sjelljes së agjentëve duke dhënë zgjidhje në lidhje me planifikimin e pathit, lehtësimin e komunikimit të mesazheve dhe kontrollin fizik të robotit. Figura 8.6 tregon një model të nivelit të lartë të shërbimit të automatizuar të nënsistemit të transportit**.**

8.3. Dekompozimi në nivel të dytë: Agjenti robotik

8.3.1. Agjent dhe objekt: dy koncepte të ndryshme

Megjithë ngjashmërinë në dukje midis agjentëve dhe objekteve të përdorura në gjuhët e programimit të orientuar nga objekti duhet të dallojmë këto entitete të ndryshme. Objektet janë entitete komputacionale që enkapsulojnë disa gjendje, janë të afta të performojnë disa veprime (methods) në këto gjendje, dhe të komunikojnë nëpërmjet kalimit të mesazheve. Diferencat midis agjentëve dhe objekteve mund të përmblidhen në përcaktimet:

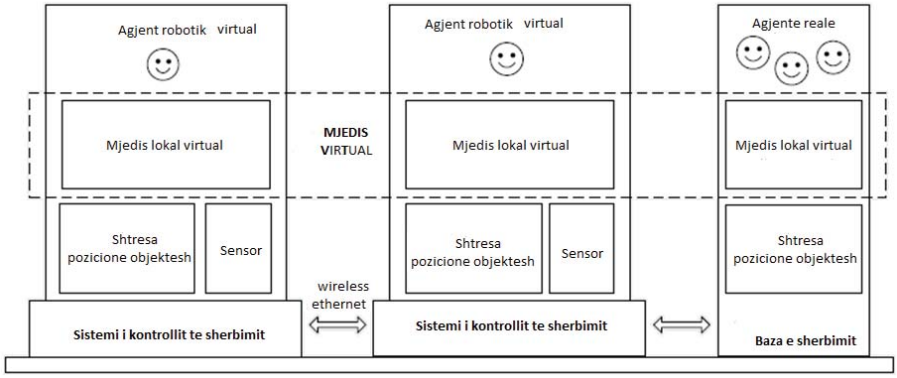

Figura 8. Mjedisi i aplikacionit për sistemin e transportimit

- Agjentët nuk thërrasin metoda, por kërkojnë veprime që të performohen.
- Objektet nuk kanë sjellje autonome fleksibël .
- Çdo agjent ka threadin e vet të kontrollit.
- Agjentët janë më të përdorshëm dhe kanë struktura më të kuptueshme.

Megjithatë nuk mund t'i veçojmë pasi për ndërtimin dhe implementimin e funksioneve, programimi i orientuar nga objekti mund të përdoret me disa modifikime. Gjithashtu, agjentët janë më të përshtatshëm në sistemet e gjerë ku gjithë nënsistemet duhet të këmbejnë informacion për të rritur ose mbajtur disa gjendje të dëshiruara. Agjentet janë entitete inteligjente me cilësi që nuk i kanë objektet. Agjentët inteligjentë operojnë duke performuar një ndryshim të shpejtë dhe të paparashikueshëm në mjedise të hapura ku ka një probabilitet të lartë që veprimet mund të dështojnë.[105] Agjentët janë të aftë për të performuar veprime.

Duke analizuar një arkitekturë abstrakte, për agjentët inteligjente mund të paraqesim skematikisht:

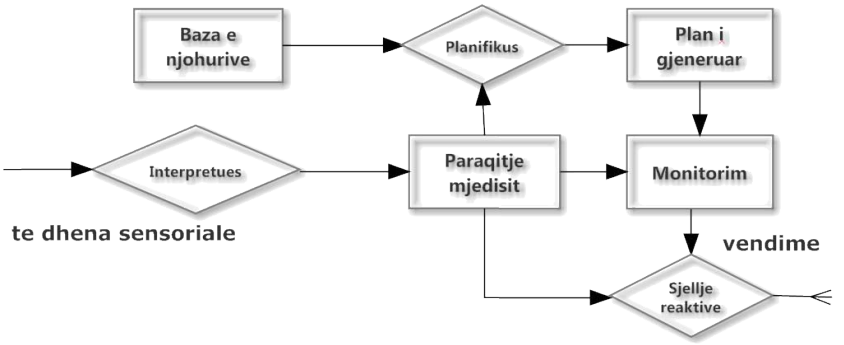

Figura 8. Arkitektura e thjeshtuar e agjenti robotik transportues

8.3.2. Simulimi i sjelljes së agjentit robotik.

Probleme shkencore të përballura. Aplikacioni ka realizuar një arkitekturë softuere për dy agjentë robotikë virtual, operues në një mjedis simulimi. Sistemi i simulimit të robotit është bërë në fillim në Flash Professional CS3 në gjuhën ActionScript 3.0, më voëe është vazhduar në Flash Professional CS5.

Simulimi kërkon të ofrojë veçoritë e projektimit të sjelljes së një sistemi inteligjent në mjedise të ndryshme si dhe varësitë e projektit nga lloji i mjedisit operues. Një mjedis i tillë i paparashikuar dhe dinamik është pjesërisht i njohur.

Kërkohet që agjenti të dijë të reagojë i gatshëm në situata të papritura, të mund të fitojë njohuri të reja mbi botën e jashtme dhe të jetë në gjendje të arsyetojë në bazë të njohurive të tij jo të plota që ka për mjedis rrethues. Agjenti duhet të jetë në gjendje të vëzhgojë në kohe reale sjelljet e objekteve të tjerë, sjellje të cilat nuk mund t'i parashikojë komplet ose të mundet t'i kontrollojë me qëllim ekzekutimin e veprimeve që shmangin konfliktet. Ndryshimet e mjedisit duhet të monitorohen vazhdimisht që të mund të lejojë agjentin të vendosë se çfarë duhet të bëjë.

Sinteza e rezultateve të pritshme. Ka mjaft aspekte për t'u konsideruar në projektimin e një agjenti autonom, duke filluar nga marrja e informacionit mbi mjedisin rrethues deri në projektimin e komponentëve në nivel të ulët, që lejojnë arritjen e objektivit me sukses. Janë arritur këto rezultate të dukshme:

1.Është projektuar një sistem softuere i aftë të integrojë në mënyrë efikase aftësinë për ekzekutimin e detyrave komplekse me kapacitete reaktive të nevojshme për të menaxhuar paqartësitë në një mjedis dinamik. Kjo arkitekturë është e tipit hibrid, e aftë të përmbledhë kapacitetet reaktive të agjentit me ato deliberative.

2. Janë implementuar module që përbëjnë këtë arkitekturë, në veçanti:

- moduli i marrjes së informacionit i cili merret me krijimin dhe mbajtjen e modelit të botës
- moduli i simulimit të mjedisit dinamik
- moduli që merr vendime për të përcaktuar veprimet që ndërmerren .

3.Janë përcaktuar kriteret kryesore për realizimin e veprimeve primitive që lejojnë reagime në kohë reale. Projekti është mbështetur mbi zhvillimin inkremental duke shtuar fuksionalitete prezente me modifikime të vogla për të bërë përshtatjen.

8.3.3. Projektim arkitekturori modulit të agjentit robotik

Moduli agjent robotik përbëhet nga katër module me rrjedhat midis moduleve. Modulet përfaqësojnë funksionalitete themelore e një agjenti të vendosur. Moduli Njohuri ofron funksionalitet për të hyrë dhe përditësimin e njohurive aktuale e agjentit. Sensing modul pranon kërkesa nga perceptimi. Vendimmarrje dhe modulet e komunikimit për të rinovuar njohuritë e agjentit në lidhje me mjedisin. Vendimmarrja dhe komunikimi përdorin njohuritë aktuale e agjentit për të marrë vendimet e duhura. Moduli komunikimi shkruan vendndodhjen aktuale në njohuritë e agjentit i cili do të përdoret nga ana e vendimmarrjes modul për të lëvizur agjent efikase drejt destinacionit. Marrjes së vendimit ofron funksionalitet për një agjent për përzgjedhjen dhe duke u thirrur në ndikimet në mjedis. Vendimmarrja përbëhet nga dy funksione themelore: Ndikimi i përzgjedhjes dhe ekzekutimi në actuatorë.

Për të zgjedhur ndikime të përshtatshme, një agjent përdor në bazë të sjelljes dhe veprimit, mekanizmin e përzgjedhjes shtrirë në rolet dhe i integruar te angazhimet. Ekzekutimi ofron funksionalitet për të nxitur ndikimet e zgjedhura në mjedis. Nepermjet sensoreve, këta agjentë marrin informacion mbi mjedisin dhe nëpërmjet aktuatorëve ekzekutojnë veprime lëvizjeje.

Një sistem kontrolli ka për detyrë të përpunojë të dhënat nga sensorët dhe të komandojë aktuatorët**.**

Agjenti dhe mjedisi i tij mund te konsiderohen si një sistem dinamik në të cilin verifikohen herë pas here ndryshime për shkak të veprimeve të vet agjentit ose të ngjarjeve të jashtme. Një mjedis i quajtur dinamik mund të ndryshojë nderkohë që agjenti është duke vepruar në të. Mjediset dinamike janë më të vështirë për t'u trajtuar pasi agjenti ka nevojë të vazhdojë të vëzhgojë botën ndërkohë që po vendos të kryejë një veprim dhe të marrë parasysh edhe rrjedhen e kohës. Në simulimin tonë dinamiciteti i mjedisit përcaktohet nga faktori pengesë e rastësishme dhe e paparashikueshme. Mjedisi është i njohur pjesërisht. Agjenti gjatë lëvizjes nëpër të duhet të jetë i gatshëm të reagojë dhe të arsyetojë mbi bazën e njohurive jo te plota. [106] Në fakt bota reale është plot me situata të tilla të paqarta, megjithatë sistemi

jonë kufizohet në mobilitetin e objektivit dhe të gjenerimit të pengesave statike dhe dinamike. Simulimet luajnë një rol të rëndësishëm në projektimin, analizimin dhe testimin e sistemeve. Gjëndjet sekuenciale të modelit gjenerohen në mënyrë dinamike sipas qellimit që mund të jetë një nga qëllimet që sistemi kërkon të arrijë.

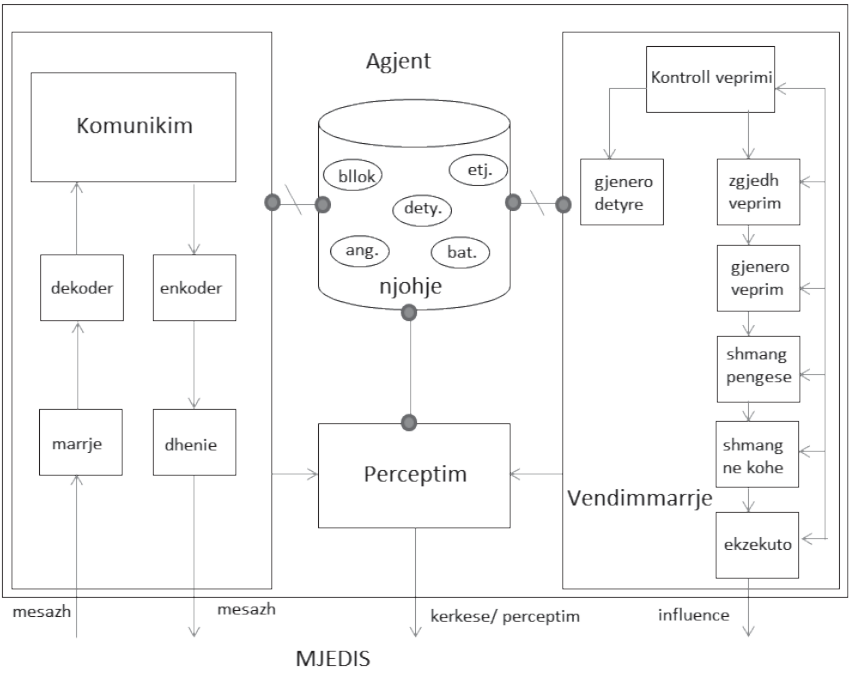

Figura 8. Bllokskema e agjentit robotik në nivel të dytë

Zgjedhja e veprimit bazuar në sjellje drejtohet nga stimuli të perceptuara në mjedis sikurse dhe stimul të brëndshëm.

Agjentët robotikë punojnë për gjendjen e brëndshme për marrjen e vendimeve duke ekzekutuar:

- **Simulimi off line**

- **Planifikim off line** (informacion statik i mjedisit të hartës gjeneruar nga një softuere simulimi); Sistemi kontrollon ekzekutimin e planit të lëvizjes në intervale sekuencash duke marrë vendime për drejtimin e lëvizjes nëse mjedisi ka ndryshime. Sistemi bazuar në agjentë manaxhon situata të paqarta me plane specifike. Agjentët në këto raste

kanë nevojë për më shumë kompromise midis stabilitetit dhe adaptimit të ndryshimeve.

- **Planifikim on line** (information dinamik i lidhur me ndryshimet e mjedisit); ose problem të brendshme të lidhura me agjentin.

• **Simulimi On line.** Vendosen disa pengesa në mjedisin e simulimit duke e bërë atë më kompleks. **Në çdo hap**, agjenti duhet të performojë dy veprime:

– Nderveprim me softuere të simulimit nëse agjenti nuk ndjen një pengesë, (**simulim off line**)

– ndërveprim me mjedisin nëse agjenti ndjen një të tillë,

– orientim duke zgjedhur hapin tjetër. (**simulim on line**)

Sistemi është projektuar të kontrollojë vazhdimisht ndryshimet e mjedisit duke shmangur dështimet.

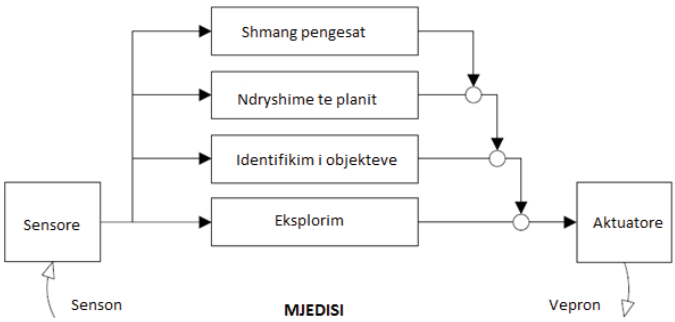

Figura 8. Proceset në një arkitekturë të agjentit robotik

Në navigim, agjenti shfaq veçori emergjente të nënsistemit autonom duke performuar dy sjellje bazë:

- **Ndërveprim** .Agjenti synon të përfundojë me sukses misionin (shërbim i kompletuar)

- **Lëvizje.** Agjenti paraqet një strategji të përpunuar për çdo detyrë specifike.

Algoritmi i trajtimit të dz sjelljeve jepet si më poshtë

```
Ne qofte se Rruge_lire== "Vërtetë" Atehere
                Ec_përpara
Ne qofte se Ka_Pengesa()="Vërtetë" Atehere
                Eksploro Perndryshe
```

```
Ndrysho_drejtim;
```

Mjedisi i simulimit ka një rol të rëndësishëm në këtë aplikacion. Një sistem i bazuar në agjentë që pretendon të jetë vetëmanaxhues, ndërmerr veprimet e duhura bazuar në gjendjen e perceptuar në mjedisin e vet. Projektimi kërkon nënsisteme të funksioneve si monitorim, vendim marrje, dhe ekzekutim veprimi. Koordinimi i sjelljeve të agjentëve brenda të njejtit mjedis kushtëzon njohjen e pjesshme të mjedisit.

8.3.4. Konceptimi i kontrollit në dy plane

Kontrolli në sistemet multiagjentë mbështetet në sjellje të modeluara të simuluara në mjedis virtual. Modeli i kontrollit është një model hibrid që përfshin një kontroll dyplanësh:

- **Kontroll i centralizuar për gjendje pa perceptim ndryshimesh**. Ky modalitet mbështetet nga një sistem konrolli qëndror që koordinon dhe komunikon me agjentët robotikë në kohë reale sipas një plani lëvizjeje mbi një mjedis virtual. Nëpërmjet algoritmave të navigimit në mjedise të njohur, është lehtësisht i planifikueshëm pathi i shërbimit (figura 8.10). Detyrat kryesore të sistemit të kontrollit janë

  – **Percaktim i rrugës së agjentit**: pathi gjenerohet nga sistemi i manaxhimit të informacionit duke përcaktuar agjentin e aftë për ekzekutim të detyrës.

  – **Rrugëzim i agjentit robotik**: agjentët navigojnë në mjedisin warehouse në mënyrë efiçente edhe në situata të paparashikuara.

  – **Mbledhje të informacionit për trafikun**: sensorët e simuluar kompletojnë njohjen e mjedisit duke shmangur objektet në pathin e planifikuar

  – **Komunikim sistem-agjent**. Entitete gjysmë autonome komunikojnë me nënsistemin e kontrollit. Sistemi i manaxhimit vendos për adresat e shërbimit.[107]

- **Kontroll i decentralizuar për prezenca të perceptuara jashtë hartës së njohjes.** Sistemi i kontrollit në mjedis të tillë lëshon kontrollin dhe planifikimi i pathit bëhet në kohë reale. Figura më poshtë paraqet modelin e kontrollit për sistemin e bazuar në agjentë. Këmbimi i mesazheve midis agjentit dhe sistemit kontroll realizohet në kohë reale(figura 8.11):

Figura 8. Algoritmi për perceptimin e mjedisit

Figura 8. Modeli logjik i sistemit bazuar mbi simulimin off line dhe on line

8.3.5. Komunikimi Wireless

Sistemi në modelin logjik të tij parashikon të përmbajë komponentë lidhës komunikimi që janë të domosdoshëm për të mbajtur të rifreskuar njohjen e botës në të cilën agjenti robotik ofron sherbimin e tij. Mjedisi virtual i simuluar nga sistemi i informacionit në kontekstin e vendosjes përsërit modelin e botës reale ku agjenti zhvillon veprimtarinë e tij, por duke patur parasysh dinamikën e tij, është e nevojshme që sistemi të shperndajë kontrollin e lëvizjes tek vet agjenti duke lejuar vetë manaxhimin e situatave të parashikuara.[108]

Kështu, duke përmbledhur kërkesat për nevojën e komunikimit në kohë reale të agjentit me sistemin apo vetë agjentëve, do të rreshtonim:

- Path i gjeneruar në mjedis të njohur me gjendje të parifreskuar
    - Kontrolli i centralizuar i shpërndarjes së detyrave:
- Agjent robotik jo plotësisht autonom.
- Sistemi transmeton instruksione sipas softuer-it të simulimit
    - Sistemi pretendon që njeh mjedisin në gjendjen e tij prezente

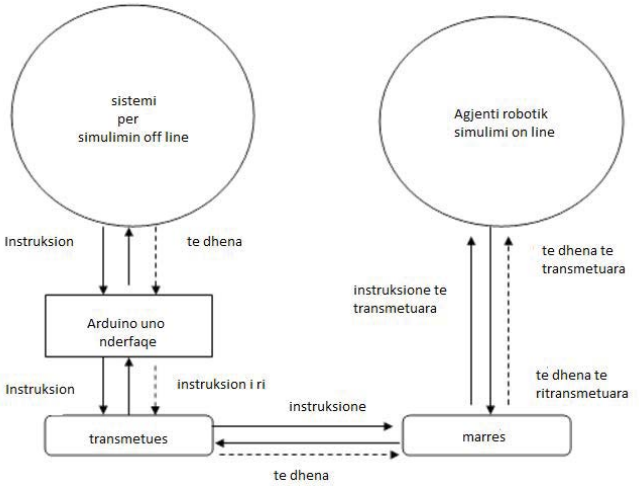

Figura 8. DFD për komunikimin wireless

Komunikimi duhet të realizohet nëpermjet këmbimit të mesazheve duke zhvilluar: Modulet transmitter dhe receiver në kanale të ndryshme frekuencash (bëhet fjalë për mjedisin real). Në përputhje me këto kërkesa, kemi realizuar vendosjen e modulit të komunikimit nëpermjet një nderfaqeje komunikimi wireless dhe komponenteve transmetuaes të lidhur në agjentin robotik dhe në nderfaqen e tipit ARDUINO UNO. Përsa i takon kanalit të vendosur, kemi përshtatur dhe zhvilluar dy protokolle: Receiver protocol, Transmitter protocol sipas skemës së figurës 8.12.

8.3.6. Protokollet receiver and transmitter

Protokolli për marrjen e të dhënave**: *receiver protocol***

```
int GetMessage()
uint8_t buflen = VW_MAX_MESSAGE_LEN;
    if (vw_get_message(buf, &buflen))
    {
    int i;
        char inStr[25];
        char inChar=-1;
    for (i = 0; i < buflen; i++)
    {
            inChar = buf[i];
            inStr[i] = inChar;
```

```
    }
        inStr[i]='\0';
    }
return command;
}
```

Protokolli për dërgimin e të dhënave: ***transmitter protocol***

```
void SendMessage(String msgToSend)
{
    int command=-1;
    uint8_t buf[VW_MAX_MESSAGE_LEN];
{
  char msgtmp[25];
  msgToSend.toCharArray(msgtmp,25);
    vw_send((uint8_t *)msgtmp, strlen(msgtmp));
vw_wait_tx();
}
```

8.3.7. Përmbledhje

Në këtë kapitull ne trajtuam zhvillimin e sistemit të manaxhimit të informacionit duke shtuar funksionalitete vetëmanaxhuese të sistemit siç ishte shërbimi i transportit të automatizuar. Në këtë fazë të rëndësishme të proçeseve të inxhinierisë së softit është e mundur të realizohet evolimi i sistemit software duke shtuar dhe përmirësuar kërkesat funksionale të tij. Në këtë fazë mundem të zhvillonim sistemin sipas modelit evolucionar. Kjo u bë e mundur në sajë të përdorimit të stilit të dekompozimit të zhvilluar mbi sistemin tonë bazë.

Gjatë kësaj faze, ne arritëm disa objektiva kryesore siç ishin:

- Shërbim i ri i bashkangjit sistemit tonë duke mos ndryshuar arkitekturën bazë të sistemit.
- Entitete autonome që ne shtuam në arkitekturë janë elemente të gatshme që implementojnë detyra specifike në të njëjtin mjedis me agjentët e tjerë
- Arritëm të ndërtojmë algoritma kontrolli të centralizuar dhe shperndajmë këtë kontroll në situata të caktuara orientimi
- Ndertuam një aplikacion i cili ka disa veçori që rreshtojnë edhe rezultatet e stimulimit tonë:

- Sistemit i shtohet një nënsistem gjysmëautonom në një mjedis pjesërisht të njohur. Ky komponent do të bazohet mbi arkitekturën e sistemeve multiagjentë robotikë me sistem kontrolli dhe decentralizim të kontrollit në raste të ndryshimit të gjendjes statike të mjedisit. Kontrolli i decentralizuar lejon hapësira për autonomi të sjelljes së agjentëve në situata të paqarta të mjedisit si komponent lidhës dhe bashkërendues.
- Sistemi performon shërbime të auto-transportimit bazuar në agjentë robotikë. Këto agjentë kanë elementet e tyre të përbashkët që lejojnë zhvillim të shpejtë dhe me kosto të ulët, si dhe besueshmëri për shkak se kemi të bëjmë me sjellje tashmë të testuara të këtyre komponentëve. Koordinimi dhe bashkëveprimi në situata të parashikuara kontrollohet nga sistemi i manaxhimit të informacionit.

Adresat e pikave të shërbimeve për secilin agjent gjenerohen nga sistemi i manaxhimit të informacionit për organizatën në mënyrë të çfarëdoshme simuluar nga agjenti i shitjeve. Agjentët kanë një vendqëndrim statik ku kthehen në përfundim të detyrës.

- Agjentët komunikojnë në një komunikim wireless të simuluar me sistemin kryesor duke marrë instruksione dhe duke dërguar të dhëna për gjendjen e tyre prezente.
- Sistemi në modelin te tij dekompozuar përmban komponente lidhës komunikimi që janë të domosdoshëm për të mbajtur të rifreskuar njohjen e botës në të cilën agjenti robotik ofron shërbimin e tij.
- Ne kemi ndertuar një mjedis virtual i simuluar nga sistemi i informacionit në kontekstin e vendosjes që përsërit modelin e botës reale ku agjenti zhvillon veprimtarinë e tij, Kontrolli i sistemit ekzekutohet në multimodalitet duke kombinuar kontrolin e centrlizuar me atë të shpërndarë duke patur parasysh dinamikën e mjedisit.

Figura 8. Simulimi i agjentit robotik në mjedis real në sistemin e autotransportit

KAPITULLI 9

Arkitektura Multiagjente në Mjedise Reale

9.1. Aplikacionet vetëmanaxhuese në mjedise reale

Në këtë kapitull, ne kemi realizuar një aplikacion të ri duke u mbështetur mbi te njëjtën arkitekturë, të projektuar në kapitujt e mëparshëm. Bëhet fjalë për një aplikacion që bazohet mbi një agjent robotik të vendosur në një mjedis të njohur pjesërisht. Duke marrë si të njohur vetëm konturet e mjedisit në plan dy dimensional, përcaktohte detyra e agjentit për të zbuluar hartën e mjedisit.

Në kemi zhvilluar një mjedis virtual të njohur në përmasat e një kutie të zezë brënda të cilit agjenti navigon duke sensuar pengesa dhe objekte në mënyrë të njëpasnjëshme. Agjenti dërgon të dhëna për çdo pikë të pengesës së percepuar me një hap të përcaktuar nga perdoruesi. Sistemi i kontrollit përpunon këto të dhëna duke integruar në një vijë të ngjyrosur bashkësinë e tyre. Le të shohim se si agjenti orientohet dhe realizon navigimin duke përdorur sensimin dhe komunikimin e informacionit në valë ultra të shkurtra.

9.1.1. Modeli i përpunimit të informacionit

Në[109], është zhvilluar dhe krijuar një model i përgjithshëm sistemi eksplorues që simulon tre nivelet tipike të performancës së operatorëve njerëzorë: aftësi, rregull dhe njohuri bazë të performancës (figura 9.1).

Një *sjellje bazuar në kompetenca* përfaqëson performancë sensor-motor gjatë veprimeve apo aktiviteteve që zhvillohen pa kontroll dhe të integruara mjaft mire në sjelljen globale të sistemit. Performanca është bazuar kryesisht në kontrollin sensorial i cili është një model shumë fleksibël dhe efikas për eksplorim të brëndshëm të botës. Kontrolli sensorial (*feedback)* nuk është duke luajtur një rol të rëndësishëm përsa kohë shqisat sensoriale të njeriut janë mjaft të ngadalta për të bërë korrigjime të shpejta. Megjithatë, koordinimi i lëvizjes bazohet në një kontroll të thjeshtë. Në përgjithësi, aktivitetet njerëzore mund të zbërthehen në një sekuencë të aftësive sensor-motor.

Në nivelin e ardhshëm - *sjellje të bazuar në rregulla* - performanca e veprimeve është e orientuar nga qëllimi, por të strukturuara përmes rregullave të depozituara në kohë (proçedura). Këto rregulla mund të jenë marrë me anë të udhëzimeve, ose mund të rrjedhin nga përvoja dhe të konkludohet nga zgjidhja e problemeve apo planifikimit. Rregullat janë të përcaktuara në mënyrë tipike si "nëse ... atëherë ...përndryshe". Ndërsa sjellja bazuar në kompetenca nuk kërkon

vëmendje të vetëdijshme të një entiteti, performanca bazuar në rregulla mbështetet në përgjithësi në eksplicite si *know-how.* Për më tepër, ndërsa rregullat mund të raportohen nga një person, ky i fundit nuk do të jetë në gjëndje të përshkruajnë se si ai e kontrollon dhe çfarë informacioni ka ai mbi atë çka ndërton rregullin.[110]

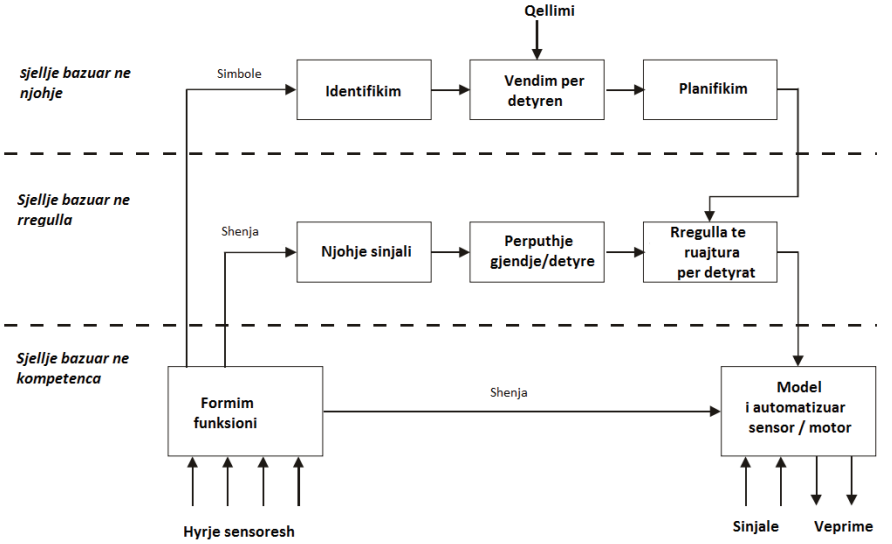

Figura 9. Modeli i perpunimit të informacionit

Në nivelin më të lartë është sjellja e kontrolluar nga njohja. Ajo kërkon një model mendor të sistemit ose duke bërë të mundur që një projektues të zhvillojë dhe parashikojë plane të ndryshme për të arritur një qëllim të caktuar.

Informacioni i vërejtur nga mjedisi në kategoritë e ndryshme të përpunimit të informacionit të njeriut mund të jetë ndarë në sinjale, shënja dhe simbole. Në nivelin të bazuar në kompetenca, informacioni perceptohet si sinjale kohë-hapësirë që tregojnë kohën dhe hapësirën e sjelljes në mjedis. Sinjalet nuk kanë kuptim apo rëndësi përveçse si të dhëna fizike të drejtpërdrejta të kohës dhe hapësirës. Te njeriu, ata janë të përpunuara nga organizmi si variablave të vazhdueshme. Një proçes formim funksioni duke patur input shqisor ose sinjale, prodhon shenja, pra, tregues të një gjëndje të caktuar (ose situatë) të mjedisit. Shënjat aktivizon (zgjidh) ose modifikon rregulla të cilat në kthim kontrollojnë renditjen e subroutinave. Arsyetimi funksional dhe gjenerimi i rregullave të reja janë të bazuara në informacionin e perceptuar si simbole. Simbolet janë të përcaktuara nga, dhe i referohen, paraqitjes së brendshme konceptuale e cila është bazë për arsyetimin dhe planifikimin. Ata përfaqësojnë variabla, marrëdhënie dhe cilësi dhe mund të përpununohen në mënyrë formale. Simbolet mund të komunikojne me të tjerë, për të cilët shenjat nuk munden. [111]

9.1.2. Përshkrimi i sistemit multiagjent në modalitetin eksplorues

Le të përpiqemi të ndërtojmë një aplikacion bazuar mbi një sistem inteligjent që prodhon njohje të mjedisit ku vepron. Në përbërje të këtij sistemi, është një prototip robot Mr. Tidy i cili ka një platformë katër rrotëshe ku vendosen modulet e tij. Përmasat e robotit prototip janë rreth 20 cm x 10 cm x 7cm. Komunikimi vizual mund të realizohet nëpërmjet një display tip 88-RC. Platforma lëviz mbi një sistem aktuator me mikrokontrollera për secilën rrotë. Sistemi i orientimit dhe navigimit përbëhet nga një sistem 12 sensorë të shpërndarë në mënyrë simetrike. Mbi platformën janë lidhur modulet transmitter dhe receiver për komunikimin wireless. Gjithashtu është lidhur moduli GPS për sinkronizimin e pozicionit në mjedis real dhe atë virtual.

Mjedisi është simuluar në përmasa drejtkëndore në kufijtë dhe me një konfigurim të çfarëdoshëm brenda kufijve. Moduli tjetër është një sistem PC që përmban monitorin ku do të ndërtohet harta e mjedisit gjatë ekplorimit. Arkitektura e zhvilluar në kapitujt e mëparshëm funksionon si arkitekturë reference në këtë aplikacion duke përshtatur komponentët dhe lidhjet e tyre në funksion të kërkesave të reja të sistemit. Adaptimi i modelit të zhvilluar do të realizohet nëpërmjet dedektimit të mekanizmave të variancës që i referohen kushteve të reja të kontekstit të vendndodhjes dhe mjedisit të aplikacionit.[112] Kështu, do të fokusohemi në këto veçori patur parasysh kërkesat funksionale të sistemit.

9.2. Kërkesa funksionale

Sistemi do të funksionojë në modalitetin eksplorues i mjedisit nëpërmjet agjentit robotik në mjedis ku vepron me këto atribute.

- Agjenti nuk njeh mjedisin në mënyrë të plotë.
- Mjedisi është i paaksesueshëm nga operator njeri por janë të matshëm kufijtë e jashtëm të tij.

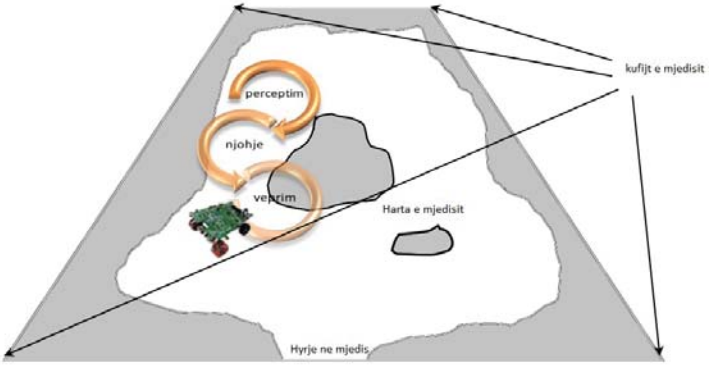

Figura 9. Pamje e përgjithshme e aplikacionit ku agjenti eksploron

- Supozohet një mjedis i aksesueshëm nga përmasat e agjentit robotik.
- Agjenti navigon duke ekzekutuar një algoritëm eksplorimi që ndjek sjellje të bazuara në njohje, d.m.th. ecje në një drejtim të zgjedhur në mënyrë të çfarëdoshme deri sa ai senson një prezencë pengese.
- Agjenti komunikon me sistemin e kontrollit nëpërmjet modulit transmetues/ marrës me dy protokolle komunikimi të zhvilluar
- Agjenti transmetonte të dhëna sapo lokalizon pengesa brënda mjedisit.
- Sistemi konverton të dhënat në koordinata në një mjedis virtual dhe integron këto koordinata në vijë të vazhdueshme si kufizuese e mjedist të paaksesueshëm.
- Çdo gjendje e re e agjentit krahasohet me të dhënat e para të transmetuara nga agjenti dhe kjo lejon ndryshimin e drejtimit të lëvizjes për një zbulim të ri.
- Simulimi duhet të paraqesë në kohë reale hartën e zbuluar nga agjenti.
- Sistemi gjeneron pas një kohe të vendosur nga operatori nje informacion grafik që integrohet në një hartë të mjedisit të zbuluar.
- Koha e eksplorimit përcakton edhe cilësinë e hartimit.

9.2.1. Arkitektura harduere e aplikacionit

Në figurën 9.3 jepet arkitektura harduere e cila përmban ndryshime komunikimi të elementëve. Pjesët kryesore janë tre nënsisteme që mund të dekompozohen në:

- **Mjedisi i aplikacionit**. Është një botë reale e cila pret agjentin. Gjatë simulimit të aplikacionit, ky mjedis është parandërtuar dhe organizuar nga projektuesi. Supozohet i paaksesueshëm nga njeriu.
- **Agjenti robotik**. Përfaqësohet nga prototipi i përdorur në aplikacionin e fazës së dytë. Modulet kyesore të tij përbëhen nga blloku sensorial, sistemi i komunkimit wireless, sistemi i kontrollit, ndërfaqet seriale dhe aktuatorët motorike.

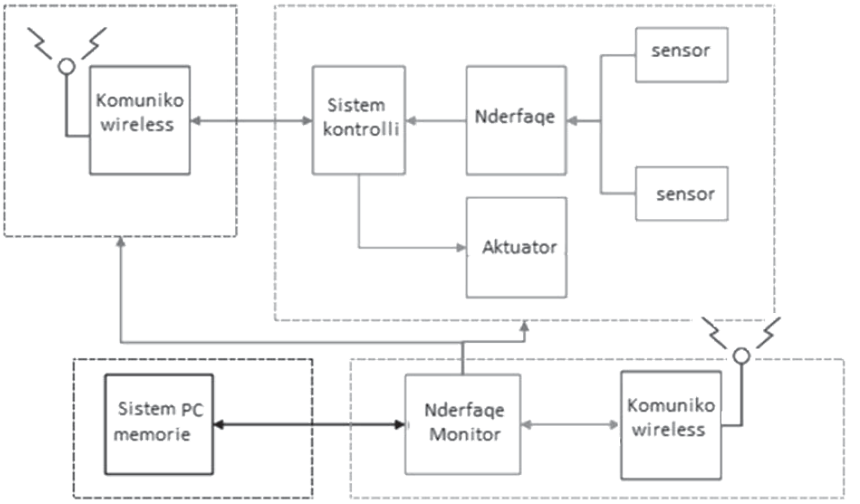

Figura 9. Arkitektura hardure e aplikacionit

- **Sistemi qendror PC**. Mbart peshën e aplikacionit. Ofron monitorin për paraqitje rezultati, sistemin përpunues grafik si dhe kujtesat për mbledhjen e informacionit. Pjesë e sistemit PC është edhe ndërfaqja ArduinoUno që lejon vendosjen e kanaleve të komunikimit.
- **Sistemi i ushqimit.** Është ushqim i vazhduar 9 V, 25mA. Ka një përshtatës për të ushqyer nga e njëjta bateri bllokun e komunikimt me 3.5 dhe 5 Volt përkatësisht për marrësin dhe transmetuesin wireless.

9.2.2. Arkitektura softuere e aplikacionit

Arkitektura e sistemi është dekompozuar në shtresën sensor, një shtresë perceptim, një shtresë e kontrollit, si dhe një shtresë aktuator. Të tre nënsisteme kryesore të sistemit robotik, për të arritur navigimin, ndërveprimin dhe komunikimin, janë të ndërlidhura me shtresa të ndryshme dhe secila prej tyre përbëhet nga disa module. Figura 3 tregon modulet individuale, nënsisteme dhe shtresa. Një përshkrim i detajuar i nënsistemeve dhe modulet e tyre është dhënë në vijim.

Nënsistemi i navigimit. Në mënyrë që roboti të lundrojë në mënyrë të sigurtë dhe me një qëllim të caktuar, sistemi PC

duhet të jetë në gjendje të lokalizojë agjentin robotik, të gjenerojë një paraqitje të mjedisit, dhe të gjejë një drejtim përmes tij. Ky nënsistem paraqet modelet e përdorura në lokalizimin gjatë simulimit në fazën e dytë. Në fund, ai furnizon sistemin qëndror me të dhëna dy dimensionale që prodhojnë hartë.

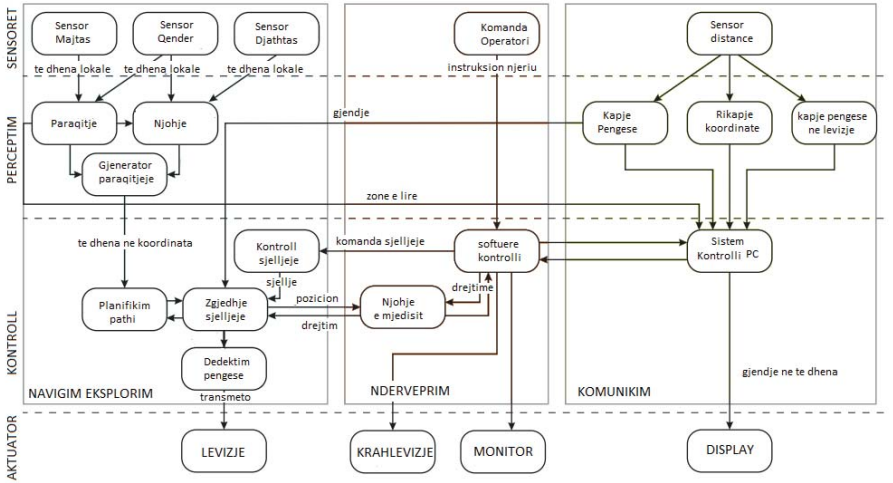

Figura 9. Arkitektura softuere e aplikacionit.

Ndërveprimi sistem-agjent. Aftësia për të krijuar një model të paparashikueshëm për mjedise statike, të bazuar në informacionin e prodhuar nga sensorët është pjesa kryesore për sistemet autonome që operojnë në mjedis të vërtetë. Moduli Gjenerator_paraqitje krijon një hartë vetëm nga enkoduesi dhe të dhënat lokale, dhe e përdor atë për të marrë një vlerësim të vendndodhjes së robotit. Një planifikim_pathi është përdorur duke u kombinuar me një zgjedhës_sjellje për të vendosur drejtimin e lëvizjes në hartë dhe e kombinuar me filtra nga moduli kontroll_sjellje adreson dhe fikson problemin e plotë të dedektimit të pengesave. Filtra të të dhënave lejojnë përafrime të shpërndarjeve me propabilitet arbitrar të koordinatave të dedektuara, duke e bërë më efikase zbulimin dhe modelimin e mjedisit. Avantazhi kryesor i nënsistemit është se nuk kemi planifikim off line por planifikim lëvizje në kohë reale duke depozituar vazhdimisht njohje të një mjedisi të përkthyer në mjedis real. Kjo njohje grumbullohet në modulin njohje_mjedisi dhe lejon një softuere_kontrolli të prezantojë agjentin gradualisht mbi këtë mjedis virtual. Ky nënsistem parashikon edhe hyrje komandash për fillime ose dhënie fund proçeseve në mënyrë manuale.

Komunikimi sistem-agjent. Këto nënsisteme garantohen në misionin e tyre nëpërmjet nënsistemit të komunikimit. Nënsitemi është përdorur totalisht me modulet që japin parametra të saktë nga sensorët e distancës drejt sistemit të kontrollit të PC-së. Skema punon edhe për të dhënat të klasifikuara për Kapje_pengese, Rikapje_koordinate dhe Kapje_pengesë në lëvizje, në skenarë mjedisi me terren josimetrik dhe prezenca objektesh. Pikat e klasifikuara janë përfshirë në një rrjet të veçantë i cili është ndërfutur pastaj me rrjet pikash virtuale. Komunikimi midis sistemit dhe agjentit është i vendosur në kohë reale dhe Display jep koodinatat reale të

agjentit duke i rifreskuar apo gjendje të tjera të vlerësueshme nga sistemi i kontrollit të PC-së.

9.2.3. Shtresa Kontroll

Shtresa mban funksionet kur navigimi në mjedise pjesërisht të njohur nga roboti përballet me situata të ndryshme që kërkojnë ndryshime të sjelljes së agjentit. Ai duhet të jetë në gjendje të eksplorojë mjedisin në mënyrë të sigurtë, të dedektojë një rrugë, të krijojë modele të një mjedisi apo objekti, ose lëviz në një drejtim të caktuar. Në vijim po vendosim funksionet kryesore të kësaj shtrese.

Eksplorues. Roboti është në gjendje të shqyrtojë mjedisin e tij në mënyrë që të gjejë pengesa duke bashkëvepruar me hartën e vet dhe duke rritur njohjen e tij mbi këtë hartë. Duke pasur parasysh një vijë kufitare në hartë, rajonet kufitare midis zonave të njohura dhe të panjohura nuk janë plotësisht të identifikuara siç është përshkruar në fazën e dytë mjedisi. Shpenzimet e kohës së eksplorimit janë të lidhura me madhësinë e mjedisit. Agjenti arrin të maksimizojë efiçencën e kërkimit me një funksion të brendshëm që kapërcen pikat me koordinata që kanë abshisë ose ordinatë të njëjtë.

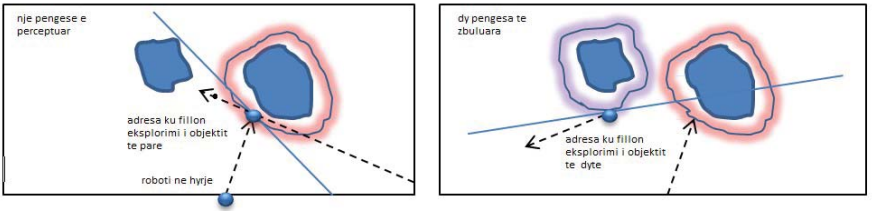

Figura 9. Modeli gjeometrik i algoritmit te eksplorimit

Dedektues: Agjenti është në gjendje të dedektojë distancat e sakta të vendndodhjes së tij nga pengesa në këndin 90 me drejtimin e lëvizjes. Në këtë sjellje, moduli i kontrollit është përdorur për të kontrolluar gjithë ndryshimet e gjendjeve të agjentit. Rajoni i zbuluar është përcaktuar nga ky modul rreth pozicioneve të gjurmuar nëpërmjet perceptimit të prezencës së objektit pengesë. Si pikë shmangjeje nga zbulimi i objektit të parë konsiderohet adresa e parë e perceptimit të pengesës. Roboti eksplorues planifikon një drejtim eksplorimi në mënyrë të çfarëdoshme, të përcaktuar me një algoritëm mbulimi të zonës me kënd 15 gradë si në figurën 9.5

Planifikues: Kjo është një sjellje e rëndësishme navigimi, në të cilën agjenti navigon sipas informacionit nga të dhënë nga kushtet e algoritmit. Kontrolli i sjelljes bazohet në informacionin e nxjerrë nga ndërveprimi i nënsistemeve duke përcaktuar kufizimet e nevojshme në planifikim, në mënyrë që roboti të lëvizë në mënyrë të sigurtë në mjedis drejt drejtimit të caktuar. Në mënyrë që të planifikojë një drejtim të

vlefshëm, një algoritëm zgjedh një pikënisje random në një drejtim të pazgjedhur më parë.

Joaktiv: Roboti mund të ndalet menjëherë dhe të gjitha komandat në lëvizje mund të jenë ndërprerë. Kjo gjendje mund të arrihet nga komanda të dhëna nga operatori. Është e nevojshme në situata emergjente kur sjellja e mëparshme ndërpritet, ose kur kemi ndërveprim me njerëz ose objekte të tjerë. Zgjedhja korrekte dhe kontrolli i këtyre sjelljeve, si dhe verifikimi i përputhshmërisë së pikave të gjeneruara, janë kryer nga modulet me të njëjtin emër që janë paraqitur në figurën 9.3.

Pikat e gjetura janë të krijuara me një frekuencë të paracaktuar, derisa objektivi është arritur, një pengesë është zbuluar, ose kontrolli dhe qëndrueshmëria në mision dështon.[113]

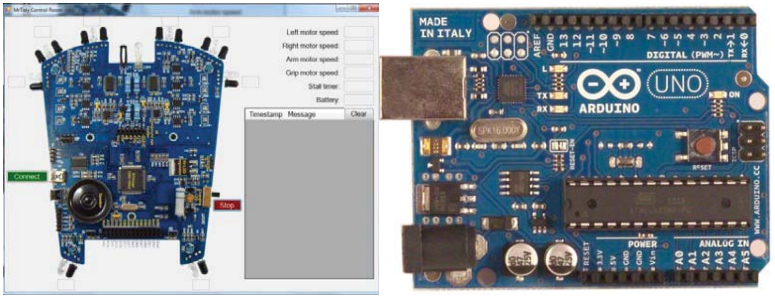

Figura 9. Ndërfaqja harduere dhe softuere e komunikimit sistem –agjent

9.2.4. Kontrolli i sjelljes së robotit

Ky modul është përgjegjës për monitorimin e ekzekutimit dhe dështimin e sjelljes aktive. Ai rinxjerr të dhëna të përshtatshme, p.sh.drejtimin nga modulet e tjera. Sipas një qëllimi, sjellja rrjedh me një frekuencë paracaktuar dhe përcillet nga moduli i planifikimit të rrugës. Për të shmangur lëvizje të luhatura, si dhe për të arritur qëllimin e drejtuar nga moduli navigim, pathi që rezulton, është i kontrolluar për qëndrueshmërinë e tij me anë të sjelljes së parashikuar. Në rast se asnjë rrugë e qëndrueshme nuk është gjetur në vazhdimësi të tentativave të shumta, devijimi nga drejtimi tolerohet nga qëndrueshmëria në rritje. Kjo mundëson agjentin për të reaguar në situata ku një pikë qëllimi apo drejtimi është bllokuar menjëherë nga një objekt dinamik, p.sh. njerëz ose objekt tjeter i lëvizshëm apo statik. Rritja e devijimi tolerohet në një afat të shkurtër kur dedektohet divergjencë nga qëllimi. Në situatat kur agjenti senson pengesa të qëndrueshme, p.sh. kur pika e dedektuar e marrë nga qëllimi qëndron brenda mjedisit, devijimi që tolerohet nga qëllimi aktual vazhdimisht rritet, kjo do të çojë në një konflikt kritik me objektivin e sjelljes. Kështu, sa herë që toleranca të bëhet shumë e madhe, sjellja prezente ndalohet dhe raportohet një gjendje dështimi, duke bërë të mundur që nënsistemi Ndërveprim të kërkojë për drejtime të reja.[114]

Në rast se është gjetur një rrugë e vlefshme, pika tjetër është përcjellë te moduli Shmang_pengesë i cili gjeneron komandat motorike për platformën lëvizëse me rrota. Ky modul merr parasysh pengesa dinamike në afërsi të robotit dhe siguron që navigimi i agjentit të jetë i sigurtë në mjedis.

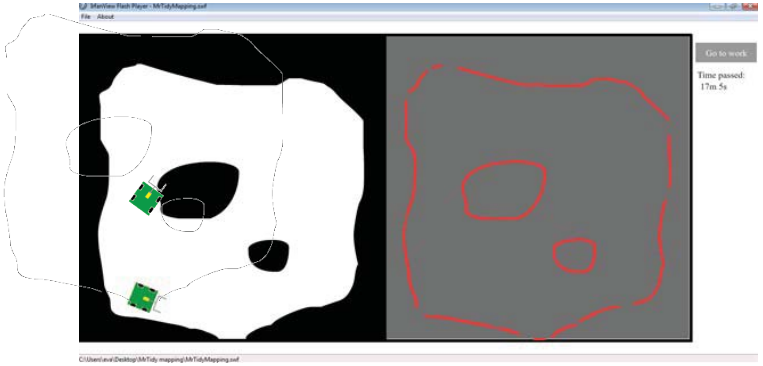

Figura 9. Ndërfaqja e simulimit të agjentit robotik si eksplorues mjedisi.

9.3. Shtresa sensor.

Mjeti trasnportues është i tipit Mr.Tidy 2011. Një set sensorësh kontrollojnë hapësirën përqark mjetit duke siguruar distancat e sigurisë. Përpara se të realizohet programimi i sistemit të kontrollit të navigimit, kalohet në proçesin e testimit dhe të kontrollit të modulit të sensimit. Kjo fazë konsiston në:

1. Testim të distancës së perceptimit të të gjithë sensorëve
2. Testim te distancës së sigurisë për gjithë sensorët.

Përpara se të përcaktojmë këto parametra, le të ndërtojmë një hartë perceptimi të mjedisit për mjetin transportues. Figura paraqet skematikisht shpërndarjen e sensoreve të identifikuar me një numër identifikimi.

Pas matjeve në proçesin e testimit, vlerat e matura janë paraqitur në tabelë 9.1.

Tabela 9. Vlerat e kodifikimit të distancës së perceptimit të sigurtë në Arduino

Id sensor	1	2	3	4	5	6	7	8
Distanca në cm	4.5	5	2	4	3	3	5	5
Vlera në kod	800	850	600	500	600	500	350	500
L. mesatare e perceptimit	4	2.5	2.5	4	3	3	5	5

1. Matjet e bëra në distanca të rritura japin edhe ligjin e perceptimit të lejuar të sensorëve duke ndërtuar tetë grafikë ndjeshmërie që do të përdoren në

algoritmin e navigimit në planifikimin online të pathit për shmangjen e perplasjeve në mjedis, me pengesa të paparashikuara Distanca kufi e lejuar pa perplasje është në perimetër 2 deri 2.5 cm me gabim 0.3 deri 0.5 cm

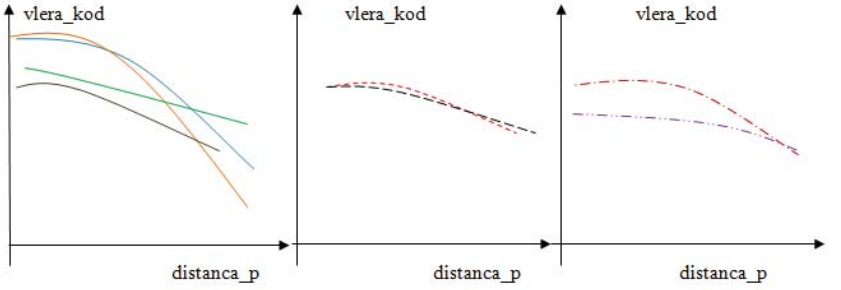

Figura 9. Grafikët e zonës së sigurtë të perceptimit për çdo sensor

2. Distanca e sigurisë ka dy vlera:
    a. Për pjesën ballore 7 deri në 8 cm.
    b. Për pjesën e pasme 3 deri në 4 cm
3. Distanca e perceptimit për:
    a. Pjesa majtas ka vlere deri 10 cm
    b. Pjesa djathtas deri 6.5 cm
    c. Prapa majtas deri 8cm
    d. Prapa djathtas 8.5 cm.

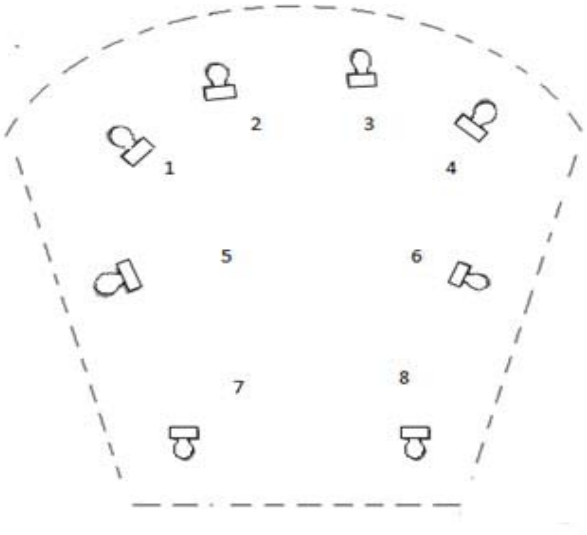

Figura 9. Pëcaktimi i hartës së sensorëve të perceptimit të mjetit transportues

Tabela 9. Vlerat e matjes së distancës së perceptimit të sigurtë nga 8 sensorët përpara, anash

distanca	Vlera_kod1	distanca	Vlera_kod2	distanca	Vlera_kod3	distanca	Vlera_kod4
10	240	10	95	12	95	12	300
8	350	8	300	10	300	7.5	350
6	400	6.5	400	8	400	6.5	400
4	500	4.5	500	6.5	450	5.3	500
3.5	600	3.5	600	5.5	600	4.3	600
distanca	Vl_kod5	distanca	Vlera_kod6	distanca	Vlera_kod7	distanca	Vlera_kod8
10	240	10	95	12	95	12	300
8	350	8	300	10	300	7.5	350
6	400	6.5	400	8	400	6.5	400
4	500	4.5	500	6.5	450	5.3	500
3.5	600	3.5	600	5.5	600	4.3	600
3	700	3	800	3	800	3.2	800
2.5	800	2	900	2	950	2.8	900
3	700	3	800	3	800	3.2	800
2.5	800	2	900	2	950	2.8	900

9.4. Rezultate të simulimeve

Aplikacioni zhvillohet në mjedis virtual dhe atë real. Gjatë simulimit, është konfiguruar një mjedis me dy pengesa dhe me kufij të zgjedhur në mënyrë apriori nga operatori.

- Vërejmë se marrdhëniet e mjedisit me agjentin kanë lidhje direkte me sjellje të agjentit si dhe ka një varësi të dukshme midis tyre.
- Sistemi softuere në projektim integron me sukses aftësinë e ekzekutimit të detyrave komplekse duke përdorur kapacitete refleksive të agjentit

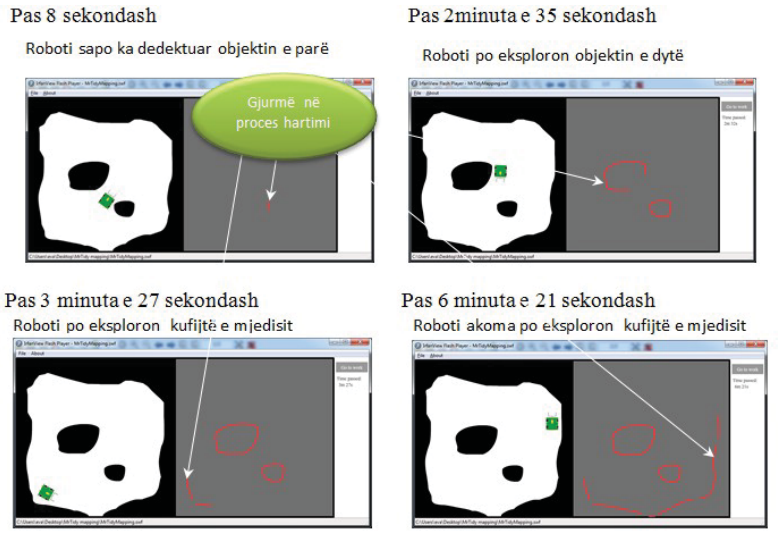

Figura 9. Simulimi i agjentit gjatë eksplorimit të mjedisit në 4 kohë ndryshme

- Agjenti manaxhon situata të paqarta në mjedis dinamik duke toleruar për të siguruar vazhdimësinë e detyrës.
- Gjthashtu vërejmë tentativa për më tepër sjellje reaktive në mjedise të paqartë.
- Shpejtësia e misionit është një tjetër kërkesë ndaj agjentit por që nuk kompromenton suksesin .
- Koha në dispozicion nuk kompromenton performancën e agjentit edhe në mjedise dinamike.

9.5. Përfundime të kapitullit

- Sistemet multiagjente siguron një mënyrë të thjeshtë për të zhvilluar aplikacione vetëmanaxhuese duke lejuar diskutimin e ndërveprimeve komplekse midis mjedisit dhe tipeve të ndryshme të agjentëve.
- Ne kemi krijuar një bazë teorike dhe praktikë në zhvillimin e tipeve të ndryshme të agjentëve si dhe të arkitekturave të tyre që stabilizojnë vendimmarrjen në situata të pazakonta të mjedisit të aplikacionit.
- Sjellje të ndryshme të agjentëve robotikë kanë qënë në fokus të punimit për të vëzhguar nëse hipoteza mbi avantazhet e modelit bazuar në agjent vërtetohet përsa i takon rritjes së inteligjencës të sistemit.
- Stili i dekompozimit në module është një model i inxhinierisë softuere që mbështetet mjaft mirë nga agjentët si entitete autonome

- Modeli i propozuar bazuar në kombinimin e planifikimit offline me planifikimin on line është një zgjidhje efikase për realizimin e veçorive emergjente të sistemit gjatë proçeseve të inxhinierisë së softit.
- Projektimi i sistemit virtual ishte një eksperiencë që kërkonte zgjidhje të situatave të veçanta.
- Praktikisht, sistemi ekzekuton detyra në dy modalitete:
  - Kërkesë për shërbime. (sistemi transmeton instruksione)
  - Kërkesë për Eksplorim.(sistemi merr të dhëna)
- Sistemi mund të implementohet në një fazë tjetër nëpërmjet zhvillimit të një prototipi në mjedis real.
- Në një kërkim të mëtejshëm do të diskutohen mjediset dinamike dhe koha e realizimit të objektivave.

KAPITULLI 10

Konkluzione

10.1. Përmbledhje

Agjentët inteligjentë tashmë kanë filluar të hyjnë kryesisht në zhvillimin e sistemeve software. Duke iu bindur këtij fakti, punimi vërtetoi se përdorimi i tyre në aplikacione softuere konsiderohet si mundësi rivaliteti me arsyetimin njerëzor në nivel të lartë besnikërie. Ne u motivuam nga fakti se agjentët mund të përdoren për të kryer disa detyra që normalisht do të kërkonin më shumë se një operator njeri dhe mjaft kohë për t'u kryer. Në punimin tonë kemi trajtuar dhe vlerësuar fillimisht proceset e inxhinierisë së softit duke konkluduar se projektimi i një arkitekture softuere përcakton pikën më të vështira në mplementimin e një sistemi kompleks. Gjithashtu, përshkruam se si duhet të ndajmë kompleksitetin në komponentë që mbartin funksionalitete specifike dhe se si gjatë projekimit, arkitektura duhet të ndahet në disa nivele në module komponentësh dhe sesi këto elemente duhet të ndërveprojnë me njeri tjetrin për të përmbushur objektivat e sistemit.

Në këtë punim ne diskutuam për zhvillimin arkitekturor të modelit bazë softuere, se si ka evoluar konceptimi arkitekturës si një mjet komunikimi midis stakeholderave apo si një paraqitje e vendimeve të projektimit që ka influencën më të madhe në cilesitë e sistemit duke konkluduar se: **Arkitektura softuere jep një model të mbikyrshëm se si një sistem është ndërtuar dhe se si ai punon. Ky model mund të transferohet tek sistemet e tjera me kërkesa të ngjashme dhe mund të promovojë një përdorim të gjerë të modelit arkitekturor.**

Gjithashtu, u fokusuam në zhvillimin e një modeli bazë arkitekturor duke u përqëndruar në atë të sistemeve multiagjente. Trajtuam projektimin arkitekturor bazuar në agjentë në një cikël jete të zhvillimit softuere. Më tej diskutuam shkurtimisht kërkesat e sistemit për të mundësuar projektimin arkitekturor. Kërkesat e ndryshme të sistemit grumbullohen në bazë të vendimeve arkitekturore bazuar mbi modelet arktitekturore siç janë stilet arkitekturore apo arkitekturat bazë. Përdorimi i metodave ekzistuese në vlerësimin e arkitekturave të tilla bazuar në agjentë dhe përshtatja e tyre në bazë të dokumentimit të kerkesave u bë nëpërmjet analizës dhe deduktimit të problematikës. U ngritën disa hipoteza që u vërtetuan me sukses. Përdorimi i agjentëve të bazuar në simulim për të eksploruar konfigurime të reja në sistemet robotike do të lejojë inxhinierizimin e një platforme aktuale rreth konfigurimeve të komponentëve runtime përpara se një sistem i të plotë të jetë ndërtuar. Duke qëndruar në sistemet multiagjentë, arritëm në përfundimin se këto

modele ofrojnë një mjet për të eksploruar sjelljet emergjente në sistemet software që mund të përdoren për të bërë imituar sjellje që arrijnë në konkluzione dhe hipoteza në lidhje me botën reale.

Një përfundim tjetër i rëndësishëm ishte fakti se agjentët autonome janë në gjendje të ekzekutojnë planet e tyre bazuar në njohjen që kanë në lidhje me mjedisin, dëshirat e tyre për të kryer detyrën e caktuar, si dhe synimet e tyre si të përmbushin detyrat që kanë. Duke vëzhguar sjelljet e simuluara nga këndvështrimi mjedisor, vërehet se sjelljet e tyre reaktive mund të çojë në zbulimin e rezultateve të papritura që mund të jenë të kushtueshme në një sistem të vërtetë real.

10.2. Përfundime teorike

Gjatë studimeve dhe hulumtimeve të shumta në planin teorik, u arrit të krijohet një bazë njohjeje e konsiderueshme mbi zhvillimin e arkitekturave agjente. Kështu, ne kemi nxjerrë disa përfundime të rëndësishme. Duke u nisur nga parimet fillestare të reaktivitetit, janë zhvilluar një numër i madh arkitekturash bazuar në agjentë. Tashmë janë identifikuar tre klasa modelesh:

- Agjentë robotikë që synojnë ndërveprim dinamik me mjedisin. Pjesa harduere e agjentit robotik realizon direkt perceptimin në veprim, duke aftësuar reaktivitetin në kohë reale.
- Agjentë të bazuar në sjellje që vënë në dukje nevojën për zgjedhje veprimi dinamik dhe fleksibël duke synuar përshtatjen me mjedise komplekse. Arkitekturat që mbështesin agjentët të bazuar në sjellje, përballen me arbitrimin e sjelljeve të ekzekutuara paralelisht dhe lejojnë ndryshime të qëllimeve në mënyrë dinamike gjatë kohës së ekzekutimit të veprimeve që performojnë sjellje të agjentit.
- Agjentë hibridë që shpjegojnë njohjen prezantuese rreth mjedisit. Arkitekturat për agjentë hibridë integrojnë arsyetimin e brëndshëm për mjedisin dhe planifikimin me reaktivitetin ndaj stimujve duke synuar kombinimin e avantazheve të planifikimit me përgjigjet e shpejta reaktive.

Këto modele kanë dy cilësi të përbashkëta:

Agjentët kanë të njëjtën arkitekturë të veçantë. Arkitekturat ndryshojnë në mënyrën si ato zgjidhin problemin e përzgjedhjes së veprimit. Këto arkitektura nuk mbështesin ndërveprimin social.

Modelet theksojnë rëndësinë e dinamizmit mjedisor. Megjithatë, vetë mjedisi është konsideruar si botë e jashtme, që do të thotë mjedisi nuk është pjesë e modelit të arkitekturës së sistemit bazuar në agjentë.

Në këtë punim, ne trajtuam sistemet multiagjente si sisteme ku entitete të pavarur ndajnë informacionin në një mjedis të përbashket dhe koordinojne sjelljet e tyre. Mjedisi shihet tani si modul i pavarur në arkitekturë dhe përcaktues në sjelljen e agjentëve brenda sistemit. Mbi këtë objektiv bazë trajtuam sjelljen reaktive kolektive midis nje bashkësie agjentësh të vendosur në një mjedis. Arritëm në përfundimin se sjellja agregate e një sistemi multiagjent shfaqet nga ndërveprimi i agjentëve ku secili ndjek një bashkësi të rregullave të sjelljeve të thjeshta elementare.

Në këtë punim u njohëm me termin *stigmergy* si koncept që tregon se si entitete individuale ndërveprojnë në mënyrë indirekte nëpërmjet një mjedisi të përbashkët: një individ ndryshon mjedisin dhe të tjerët i përgjigjen këtij ndryshimi. Folëm në mënyrë të veçantë për sistemet multiagjente robotike, ku u vu theksi në rëndësinë e arkitekturës bazë për agjentët dhe mjedisin. U përcaktuan karakteristika të rëndësishme të sistemeve multiagjente në lidhje me mjedisin.

Arritëm në përfundime të rëndësishme duke zgjeruar këndvështrimin teorik si:

- Shtrirja e arkitekturave të agjentëve në zhvillimin e mekanizmave të përzgjedhjes së veprimit në lidhje të ngushtë me perceptimin dhe komunikimin me mjedisin.
- Aftësimi i agjentëve për të patur një ndërveprim social dhe për të luajtur role të ndryshme në bashkëpunime të ndryshme.
- Promovimi i mjedisit si nje abstraksion i nivelit të parë që mund të përdoret në mënyrë krijuese në aplikacionet bazuar në agjentë.

10.3. Konsiderata mbi aplikacionin

Ky punim paraqet një model të sistemeve multiagjente që implementon përfitimet e përdorimit të teknikave bazuar mbi agjent në vetë manaxhimin e njohjes nga sistemi. Në proçesin e studimit arkitekturave të ndryshme, ne kemi kombinuar arkitekturën e shtresëzuar me atë *build in* në funksion të rritjes së niveli i abstraksionit.

Ne përdorim nje metodë unike për të zhvilluar agjentë të pavarur, ku çdo agjent ka një detyrë të veçantë për të përfunduar. Agjentët veprojnë të pavarur, megjithatë ata mund të bashkëpunojnë me njëri tjetrin si dhe me përdoruesit.

Ne mësuam mbi shpërndarjen e funksioneve në një sistem të manaxhimit të një baze të dhënash. Është një model që mund të zgjidhë shumë mirë kompleksitetin e DBMS-ve duke përdorur agjentët e informacionit si mënyrë e lehtë për të mbështetur shërbimet komplekse në baza të dhënash. Rezultatet e dhëna nga ekzekutimi i simulimit konfirmojë vlefshmërinë e përdorimit të modelit.

Ky punim është i rëndësishëm sepse tregon se agjentët inteligjente do të jenë teknologjitë më të mira që çojnë në përmirësime të rëndësishme të cilësisë dhe performancës së sistemeve softuere. Aftësia e agjentëve të pavarur, të planifikuar për të ndjekur qëllimet dhe veprimet e tyre për të bashkëpunuar, koordinuar, dhe të negociojnë më të tjerët, dhe për t'iu përgjigjur në mënyrë fleksibile dhe inteligjente me situata dinamike dhe të paparashikueshme, do të zgjerojë përdorimin e tyre fuqishëm në shumë aplikacione të tjera vetëmanaxhuese.

10.4. Konkluzione

Në kuadër të proceseve te inxhinierise se softit është e mundur të realizohet evolimi i sistemit software duke shtuar dhe përmiresuar kërkesat funksionale të tij. Zhvillimi inkremental është një model mjaft i përdorur në ciklin e jetës së softit, mjaft i njohur në inxhinierinë e proçeseve të softit. Duke përdorur këtë model, në këtë fazë mundem të zhvillonim sistemin. Kjo u be e mundur në sajë të përdorimit të stilit të dekompozimit të zhvilluar mbi sistemin tonë bazë.

Ne arritëm disa objektiva kryesore siç ishin:

- Shërbim i ri i bashkangjitet sistemit tone duke mos ndryshuararkitekturën bazë të sistemit.
- Entitete autonome që ne shtuam në arkitekture, janë elemente të gatshme që implementojnë detyra specifike në të njëjtin mjedis me agjentet e tjere. Arritem te ndertojme algoritma kontrolli te centralizuar dhe shperndajme kete kontroll ne situata te caktuara orientimi
- Ndërtuam nje aplikacion që ka disa veçori që rreshtojnë edhe rezultatet e similimit tonë. Sistemit i shtohet një nënsistem gjysmëautonom në një mjedis pjesërisht të njohur. Ky komponent bazohet mbi arkitekturën e sistemeve multiagjente robotike me sistem kontrolli dhe decentralizim të kontrollit në raste të ndryshimit të gjendjes statike të mjedisit. Kontrolli i centralizuar lejon hapësira për autonomi të sjelljes së agjentëve në situatë të paqarta të mjedisit si komponent lidhës.
- Sistemi performon shërbime të autotransportimit bazuar në agjentë robotike. Këto agjentë kanë elementët e tyre të përbashkët që lejojnë zhvillim të shpejtë dhe tashmë të testuar të këtyre komponentëve. Koordinimi dhe bashkeveprimi në situata të parashikuara kontrollohet nga sistemi i manaxhimit të informacionit. Adresat e pikave të shërbimeve për secilin agjent gjenerohen nga sistemi i manaxhimit të informacionit për organizatën në mënyrë të çfarëdorshme simuluar nga agjenti i shitjeve. Agjentët kanë një vendqëndrim statik ku kthehen në përfundim të detyrës.
- Agjentët komunikojnë në një komunikim wireless të simuluar me sistemin kryesor duke marrë instruksione dhe duke dërguar të dhëna për gjendjen e tyre prezente. Ne kemi ndërtuar nje mjedis virtual të simuluar nga sistemi i informacionit

ne kontekstin e vendosjes që përsërit modelin e botes reale ku agjenti zhvillon veprimtarinë e tij, Kontrolli shtrihet në multimodalitet duke kombinuar kontrolin e centralizuar me atë të shpërndarë, duke patur parasysh dinamikën e mjedisit.

10.5. Në fokus: sistemet multiagjente në aplikacione inteligjente

Agjentët inteligjentë kanë hyrë kryesisht në aplikacione bazuar mbi sistemet software. Përdorimi i tyre në simulime ka dhënë një nivel të kënaqshëm të besnikërisë që mundëson rivalitet tëagjentëve me arsyetimin dhe veprimin njerëzor. Shpesh, agjentët janë përdorur për të kryer disa detyra që normalisht do të kërkonin një operator njeri për t'u kryer. Përdorimi i agjentëve të implementuar në simulime për të eksploruar konfigurime të reja në sistemet robotike, do të lejojë inxhinierinë e kësaj kategorie softuere të japë një guidë më të plotë të zhvillimit të një sistemi softuere vetëmanaxhues.

Sistemet multiagjente ofrojnë një mjet për të eksploruar sjellje emergjente në sistemet e software që mund të përdoren për të realizuar modele në lidhje me botën reale. Agjentët autonome janë në gjendje të ekzekutojnë planet e tyre bazuar në besimet e tyre në lidhje me mjedisin, dëshirat dhe synimet e tyre për të kryer detyrën e caktuar, si dhe të përmbushin detyrat. Duke vëzhguar sjelljet e simuluara në mjedise virtuale, studimi i sjelljeve të tyre reaktive mund të çojnë në zbulimin e rezultateve të papritura që mund të jenë të kushtueshme në një sistem të vërtetë inteligjent.

Sistemet robotike kanë qenë tradicionalisht të zbatuar me një sërë algoritmash inxhinierike dhe teknika që lejojnë robotët për të kryer një seri sjelljesh. Bashkimi i një numri sjelljesh të nivelit të ulët në një sistem robotik mund të prodhojë një sjellje emergjente në nivel të lartë.. Me një grup të fuqishëm të sjelljeve të testuara në simulim, një sistem robotik mund të optimizohet dhe nga ana tjetër të kursehet kohë dhe kosto.

Duke futur teknika të agjentëve me atributet fizike të një sistemi robotik, kjo lejon inxhinierët në fushën e robotikës për të eksploruar mënyra të reja për të venë në punë robotë inteligjentë. Robotët mund të pajisen me të njëjtën arkitekturë të implentuar në një agjent inteligjent në simulim që të ketë bindjet e veta, dëshirat dhe qëllimet gjatë kryerjes së një detyre reale në mjedis gjithashtu real.

Në mënyrë që të kapërcehet hendeku midis simulimit dhe sistemeve reale robotike, simulimi duhet të plotësojë një kriter që është imitimi me vërtetësi i botës së vërtetë. Kjo nuk do të thotë se simulimi duhet plotësisht të modelojë botën reale, megjithatë, shumë situata mund të merret si të mirëqena në simulim që nuk mund të përballohen në botën reale. Kjo tezë përpiqet të përcaktojë një sistem agjent si sistem që punon në menaxhimin e burimeve, që merr parasysh kufizimet e shumta në sistemet e vërteta robotike. Motivimi im për këtë tezë ishte që të ketë një model zhvillimi në kërkimin mbi teknika të bazuar agjent për alokimin e burimeve për të

trajtuar lëvizjen e entiteteve robotike në mënyrë të koordinuar për të arritur një qëllim final të përbashkët.

10.6. Mbyllje

Agjentët autonome sigurojnë një pasuri të burimeve të njohjes nga trajtimi i detyrave që janë përfunduar normalisht nga homologët e tyre njerëzore. Përdorimi i një sistemi robotik është parashikuar të lehtësojë operatorin njerëzore për të bërë detyra që janë të vështira ose në thelb të rrezikshme. Nëse një detyrë shërbimi është realizuar normalisht nga njeriu, tani ajo mund të plotësohet nga një njeri që kontrollon një robot, i cili ka fituar terren me shpejtësi.

Përfitimi i vërtetë i një sistemi robotik është punësimi i tij dhe reduktimi i ndërhyrjes së njeriut me punë që robotit i janë caktuar për të përfunduar[115].

SHTOJCE A

Algoritmat e Navigimit te Agjentëve Robotikë

1. **Algoritmi kryesor i lëvizjes**

2. **Algoritmet ne modalitet Kerko_Majtas dhenKerko_Djathtas**

4. Algoritmi i navigimit të robotit kur ka pengesa

5. Algoritmi i navigimit të robotit në modalitetin Eksploro

SHTOJCE B

Algoritmat e Agjentëve të Informacionit

1. **Algoritmi display_agent**

2. **Algoritmi per agjentin : Price_agjent**

3. **Algoritmi Order_agent**

4. **Agjenti: Selling_agjent**

REFERENCA

1) IEEE. Recommended Practice for Architectural Description of Software Intensive Systems, IEEE Std 1471-2000. Institute of Electrical and Electronics Engineers, 2000.

2) I. Jacobson, G. Booch, J. Rumbaugh. The Unified Software Development Process. Addison-Wesley, 1999.

3) P. Agre, D. Chapman. Pengi: An Implementation of a Theory of Activity. In National Conference on Artificial Intelligence, Seattle, WA, 2007.

4) P. Agre, D. Chapman. What are Plans for? Designing Autonomous Agents, MIT Press, 2001.

5) T. Al-Naeem, I. Gorton, M. Babar, F. Rabhi, B. Benatallah. A Quality-driven Systematic Approach for Architecting Distributed Software Applications. In 27th International Conference on Software Engineering, New York,NY, USA, 2005. ACM Press.

6) J. Allen, G. Ferguson. Actions and Events in Interval Temporal Logic. Journal of Logic and Computation, Special Issue on Actions and Processes, 4:531{579, 1994.

7) M. Arbib. Schema Theory. Encyclopedia of Artificial Intelligence, 1998.

8) R. Arkin. Motor Schema-Based Mobile Robot Navigation. International Journal of Robotics Research, p. 921-12, 1999.

9) R. Arkin. Integrating Behavioral, Perceptual, World Knowledge in Reactive Navigation. Designing Autonomous Agents, MIT Press, 1999.

10) R. Arkin. Behavior-Based Robotics. Massachusetts Institute of Technology, MIT Press, Cambridge, MA, USA, 1998.

11) S Arora, A. Raina, A. Mittal. Collision Avoidance Among AGVs at Junctions. In IEEE Intelligent Vehicles Symposium, 2000

12) S Arora, A. Raina, A. Mittal. Hybrid Control in Automated Guided Vehicle Systems. In IEEE Conference on Intelligent Transportation Systems, 2001.

13) C. Atkinson, T. Kuhne. Aspect-Oriented Development with Stratified Frameworks. IEEE Software, p.81-89, 2003.

14) P. Avgeriou. Describing, Instantiating and Evaluating a Reference Architecture: A Case Study. Enterprise Architect Journal,. Fawcette Technical Publications. 2003

15) O. Babaoglu, H. Meling, A. Montresor. Anthill: A Framework for the of Agent-Based Peer-to-Peer systems. In 22nd International Conference on Distributed Computing Systems, IEEE Computer Society, Digital Library Vienna, Austria, 2003

16) F. Bachmann and L. Bass. Managing Variability in Software Architectures. In Symposium on Software Reusability, ACM Press New York, NY, USA, 2001.

17) S. Bandini, M. L. Federici, S. Manzoni, G. Vizarri. Towards a Methodology for Situated Cellular Agent Based Crowd Simulations. In 6th International Workshop on Engineering Societies in the Agents World, ESAW, 2005.

18) S. Bandini, S. Manzoni, and C. Simone. Dealing with Space in Multiagent Systems: A Model for Situated Multiagent Systems. In 1st International Joint Conference on Autonomous Agents and Multiagent Systems. ACM Press, 2002.

19) Bandini, S. Manzoni, and G. Vizzari. MultiAgent Approach to Localization Problems: the Case of Multilayered Multi Agent Situated System. Web Intelligence and Agent Systems, p.155-166, 2004.

20) S. Bandini, S. Manzoni, and G. Vizzari. A Spatially Dependent Communication Model. In 1st International Workschop on Environments for Multiagent Systems, Lecture Notes in Computer Science, Vol. 3374. Springer-Verlag, 2005.

21) M. Barbacci, R. Ellison, A. Lattanze, J. Sta®ord, C. Weinstock, Quality Attribute Workshops. Technical Report CMU/SEI-2003-TR-016, Software Engineering Institute, Carnegie Mellon University, PA, USA, 2003.

22) M. Barbacci, M. Klein, T. Longstaf, C. Weinstock. Quality Attribute Workshops. Technical Report CMU/SEI-95-TR-21, Software Engineering Institute, Carnegie Mellon University, PA, USA, 1995.

23) L. Bass, P. Clements, R. Kazman. Software Architecture in Practice. Addison Wesley Publishing Comp., 2003.

24) J. Batman. Characteristics of an Organization with Mature Architecture Practices. Essays on Software Architecture,. Software Engineering Institute, 2006

25) K. Beck and R. Johnson. Patterns Generate Architectures. In ECOOP'94: Proceedings of the 8th European Conference on Object-Oriented Programming, Lecture Notes in Computer Science, Vol. 821 Springer-Verlag., London, UK, 1994.

26) F. Bellifemine, A. Poggi, and G. Rimassa. Jade, A FIPA-compliant Agent Framework. In 4th International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, 1999.

27) S. Berman, Y. Edan, and M. Jamshidi. Decentralized autonomous AGVs in material handling. Transactions on Robotics and Automation, 19(4), 2003.

28) C. Bernon, M-P. Gleizes, S. Peyruqueou, and G. Picard. Adelfe: A Methodology for Adaptive Multiagent Systems Engineering. In 3th International Workshop on Societies in the Agents World, ESAW, Lecture Notes in Computer Science, Vol. 2577. Springer-Verlag, 2002.

29) E. Bonabeau, F. Henaux, S. Gu¶erin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in telecommunications networks with ant-like agents. In Second International Workshop on Intelligent Agents for Telecommunication a Applications, IATA, Paris, France, Springer-Verlag.1998.

30) N. Bouck¶e. Situated Multiagent System Approach for Distributing Control in Automatic Guided Vehicle Systems. Master Thesis, Katholieke Universiteit Leuven, Belgium, 2003.

31) N. Bouck¶e, T. Holvoet, T. Lefever, R. Sempels, K. Schelfthout, D. Weyns, J. Wielemans. Applying the Architecture Tradeoff Analysis Method to an Industrial Multiagent System Application. 2005.

32) N. Bouck¶e, D. Weyns, T. Holvoet, K. Mertens. Decentralized allocation of tasks with delayed commnencement. In 2nd European Workshop on MultiAgent Systems, EUMAS, Barcelona, Spain, 2004.

33) N. Bouck¶e, D. Weyns, K. Schelfthout, T. Holvoet. Applying the ATAM to an Architecture for Decentralized Contol of a AGV Transportation System. In 2nd International Conference on Quality of Software Architecture, QoSA, Vasteras, Springer Sweden, 2006.

34) P. Kruchten. The 4+1 View Model of Architecture. IEEE Software, 12(6) 4250, 1995.

35) N. Rozanski, E. Woods. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison Wesley Publishing , 2005.

36) P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford. Documenting Software Architectures: Views and Beyond. Addison Wesley Publishing Comp., 2002.

37) C. Boutilier and R. I. Brafman. Partial-order planning with concurrent interacting actions. Journal on Artificial Intelligence Research, p.105-136, 2001.

38) L. Breton, S. Maza, P. Çastagna. Simulation multi-agent systems: comparaison avec une approche predictive. 5e Conference Francophone de Modelisation et Simulation, 2004.

39) M. Brodie, I. Rish, S. Ma, and N. Odintsova. Active probing strategies for problem determination. In 18th International Joint Conference on Articial Intelligence, 2003.

40) R. Brooks. Achieving artificial intelligence through building robots. AI Memo 899, MIT Lab, 1986.

41) R. Brooks. The Behavior Language; User's Guide. AI Memo 1227, MIT Lab, 1990.

42) R. Brooks. Intelligence without reason. In 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, 1991.

43) H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference Architecture for Holonic Manufacturing Systems: PROSA. Jounal of Manufactoring Systems, p.37, 255, 274, 1998

44) J. Bryson. Intelligence by Design, Principles of Modularity and Coordination for Engineering Complex Adaptive Agents. PhD Dissertation, MIT, USA, 2001.

45) F. Buchmann and L. Bass. Introduction to the Attribute Driven Design Method. In 23rd International Conference on Software Engineering, IEEE Computer Society.Toronto, Ontario, Canada, 2001.

46) G. Cabri, L. Ferrari, and F. Zambonelli. Role-Based Approaches for Engineering Interactions in Large-Scale Multi-agent Systems. In Software Engineering for Multi-Agent Systems II, Lecture Notes in Computer Science, Vol. 2940. Springer-Verlag, 2004.

47) M. Calder, M. Kolberg, E. Magill, and S. Rei®-Marganiec. Feature Interaction: A Critical Review and Considered Foreçast. Computer Networks, p.115-141, 2003.

48) J. Çastro, M. Kolp, and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. Informatica Systems, p:365-389, 2002.

49) P. Clements, R. Kazman, and M. Klein. Evaluating Software Architectures: Methods and Case Studies. Addison Wesley Publishing Comp., 2002.

50) P. Clements and L. Northrop. Software Product Lines: Practices and Patterns. Addison Wesley Publishing Comp., August 2001.

51) P. Cohen and H. Levesque. Teamwork. Nous, Special Issue on Cognitive Science and Artificial Intelligence, p. 487-512, 2002.

52) C. Cuesta, M. del Pilar Romay, P. de la Fuente, and M. Barrio-Solo¶rzano. Architectural Aspects of Architectural Aspects. In 2nd European Wirkshop on

Software Architecture, EWSA, Lecture Notes in Computer Science, Vol.3527. Springer, 2005.

53) R. Brooks. Intelligence without reason. In 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, 1991.

54) R. Custers. The Agent Network Architecture Extended for Cooperating Robots. Master Thesis, Katholieke Universiteit Leuven, Belgium, 2003.

55) Y. Demazeau. Multi-Agent Systems Methodology. In 2nd Franco-Mexican School on Cooperative and Distributed Systems, LAFMI 2003,

56) J. Deneubourg and S. Goss. Collective Patterns and Decision Making. Ecology, Ethology and Evolution, p.295-311, 1989.

57) M. Dorigo and L. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, p. 53-66, 1997.

58) B. Dunin-Keplicz and R. Verbrugge. Calibrating collective commitments. In Multi-Agent Systems and Applications III, 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS, Prague, Czech Republic, Lecture Notes in Computer Science, Vol. 2691. Springer, 2003.

59) E. Durfee and V. Lesser. Negotiating Task Decomposition and Allocation Using Partial Global Planning. Distributed Artificial Intelligence, p. 229-244, 1999.

60) M. Fayad and D. Schmidt. Object-Oriented Application Frameworks, Guest Editorial. Communications of the ACM, Special Issue on Object-Oriented Application Frameworks, p. 32-38, 1997.

61) J. Ferber. An Introduction to Distributed Artificial Intelligence. Addison-Wesley, 1999.

62) J. Ferber, F. Michel, and J. Baez. AGRE: Integrating environments with organizations. In 1st International Workschop on Environments for Multiagent Systems, Lecture Notes in Computer Science, Vol. 3374. Springer-Verlag, 2005.

63) J. Ferber and J. Muller. Influences and Reaction: a Model of Situated Multiagent Systems. 2nd International Conference on Multi-agent Systems, Japan, AAAI Press, 1996.

64) A. Garcia, U. Kulesza, and C. Lucena. Aspectizing Multi-Agent Systems: From Architecture to Implementation. In Software Engineering for Multi-Agent Systems III, SELMAS 2004, Lecture Notes in Computer Science, Vol. 3390. Springer, 2005.

65) M. Genesereth and N. Nilsson. Logical Foundations of Artificial Intelligence. Morgan Kaufmanns, 1997.

66) F. Giunchiglia, J. Mylopoulos, A. Perini. The TROPOS Software Development Methodology: Processes, Models and Diagrams. 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems AAMAS'02, ACM Press, New York, 2002.

67) M. Griss, I. Jacobson, and P. Jonsson. Software Reuse: Architecture, Process and Organization for Business Success. Addison Wesley Professional, 1997.

68) S. Hadim, N. Mohamed. Middleware Challenges and Approaches for Wireless Sensor Networks. IEEE Distributed Systems Online, 2006.

69) P. Modi, S. Mancoridis, W. Mongan, W. Regli, and I. Mayk. Towards a Reference Model for Agent-Based Systems. In Industry Track of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan, 2006.

70) S. Hayden, C. Carrick, Q. Yang. A Catalog of Agent Coordination Patterns. In 3th International Conference on Autonomous Agents, ACM Press.New York, NY, USA, 1999.

71) A. Helleboogh, T. Holvoet, D. Weyns, Y. Berbers. Extending time management support for multi-agent systems. In Multiagent and Multiagent-based Simulation, New York, USA, Lecture Notes in Computer Science, Vol. 3415, 2005.

72) A. Helleboogh, T. Holvoet, D. Weyns, Y. Berbers. Towards time management adaptability in multi-agent systems. In Agents and Multiagent Systems III: Adaptation and Multiagent Learning, Lecture Notes in Computer Science, Vol. 3494, 2005.

73) A. Helsinger, R. Lazarus, W. Wright, J. Zinky. Tools and Techniques for Performance Measurement of Large Distributed Multiagent Systems. In 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, Melbourne, Victoria, Australia. ACM, 2003.]

74) T. Holvoet, E. Steegmans. Application-Specific Reuse of Agent Roles. In Software Engineering for Large-Scale Multi-Agent Systems, Lecture Notes in Computer Science, Vol. 2603. Springer-Verlag, New York, 2003.

75) T. Holvoet, P. Valckenaers. Exploiting the Environment for Coordinating Agent Intentions. In 3th International Workshop on Environments for Multiagent Systems, E4MAS, Hakodate, Japan, 2006.

76) S. Hoshino, J. Ota, A. Shinozaki, H. Hashimoto. Design of an AGV Transportation System by Considering Management Model in an ACT. Intelligent Autonomous Systems, p.505-514, 2006.

77) M. Huhns, L.M. Stephens. Multiagent Systems and Societies of Agents. In Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence. MIT Press, 2000.

78) N. Jennings. An agent-based approach for building complex software systems. Communications of the ACM, 44, p:35- 41, 2001.

79) N. Jennings, A. Luomuscio, S. Parsons, C. Sierra, M. Wooldridge. Automated Negotiation: Prospects, Methods and Challenges. International Journal of Group Decision and Negotiation, p.199-215, 2001.

80) R. Johnson and B. Foote. Designing reusable classes. Journal of Object Oriented Programming, vol 1(2):22-35, 1988.

81) L. Kaelbling. Goals as Parallel Program Specifications. In 7th National Conference on Artificial Intelligence, Minneapolis, Minnesota, 1998.

82) L. Kaelbling and J. Rosenschein. Action and Planning in Embedded Agents. Designing Autonomous Agents, MIT Press, 1990.

83) E. Kendall, C. Jiang. Multiagent System Design Based on Object Oriented Patterns. Journal of Object Oriented Programming, vol 10(3) p. 41-47, 1997.

84) J. Kephart. Research Challenges of Autonomic Computing (IBM). Invited talk, International Conference on Software Engineeging, St. Louis, USA, 2005.

85) C. Kim and J. Tanchoco. Operational Control of a Bi-directional Automated Guided Vehicle Systems. International Journal of Production Research, vol 31(9), 2002.

86) D. Kinny, M. Ljundberg, A. Rao. Planning with Team Activity. In 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Lecture Notes in Computer Science, Vol. 830. Springer-Verlag, London, UK, 1998.

87) M. Kolp, P. Giorgini, J. Mylopoulos. A Goal-Based Organizational Perspective on Multi-agent Architectures. In 8th International Workshop on Intelligent Agents, London, UK, Springer – Verlag, 2002.

88) Y. Labrou. Standardizing Agent Communication. New York, NY, USA, Springer-Verlag, .2001.

89) C. Larman. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. Prentice Hall, 2002.

90) P. Maes. Situated Agents can have Goals. Designing Autonomous Agents, MIT Press, 1990.

91) P. Maes. Modeling Adaptive Autonomous Agents. Artificial Life Journal, vol 1(1-2) p.135,162, 1994.

92) R. Makar, S. Mahadevan, M. Ghavamzadeh. Hierarchical MultiAgent Reinforcement Learning. In 5th International Conference on Autonomous Agents, 2001.

93) C. Malcolm, T. Smithers. Symbol Grounding via a Hybrid Architecture in an Autonomous Assembly System. Designing Autonomous Agents, MIT Press, 1999.

94) M. Mamei and F. Zambonelli. Co-Fields: A Physically Inspired Approach to Distributed Motion Coordination. IEEE Pervasive Computing, vol 3(2)p.52-61, 2004.

95) M. Mamei, F. Zambonelli. Field-based Coordination for Pervasive Multiagent Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

96) M. Mamei, F. Zambonelli, L. Leonardi. Distributed Motion Coordination with Co-Fields: A Case Study in Urban Traffic Management. In 6th IEEE Symposium on Autonomous Decentralized Systems, Pisa, Italy. IEEE Press, 2003.

97) M. Mamei, F. Zambonelli, L. Leonardi. Tuples On The Air: A Middleware for Context-Aware Computing in Dynamic Networks. International Conference on Distributed Computing Systems Workshops, 2003.

98) X. Mao, E. Yu. Organizational and social concepts in agent oriented soft- ware engineering. In Agent-Oriented Software Engineering V, 5th International Workshop, AOSE, New York, NY, USA, Vol. 3382. Springer-Verlag, 2004.

99) F. Michel, A. Gouaich, J. Ferber. Weak Interaction and Strong Interaction in Agent Based Simulations. In Multi-Agent-Based Simulation 4th International Workshop, MABS, Melbourne, Australia, Vol. 2927. Springer-Verlag, 2003.

100) J. Odell, H. V. D. Parunak, M. Fleischer, and S. Breuckner. Modeling Agents and their Environment. In Agent-Oriented Software Engineering III, Third International Workshop, Bologna, Italy, 2002, Vol. 2935. Springer-Verlag, 2003.

101) L. Padgham, M. Winiko. Prometheus: A Methodology for Developing Intelligent Agents. In Agent-Oriented Software-Engineering III, Springer-Verlag Vol. 2585. 2005.

102) D. Parnas. On a "Buzzword": Hierarchical Structure. Software pioneers: Contributions to software engineering, p. 429-440, 2002.

103) H. V. D. Parunak and S. Brueckner. Analyzing Stigmergic Learning for Self-Organizing Mobile Ad-Hoc Networks (MANET's). In Engineering Self-Organising Systems, Methodologies and Applications, ESOA, Vol. 3464. Springer, 2005.

104) H. V. D. Parunak, S. Brueckner, and J. Sauter. Digital Pheromones for Coordination of Unmanned Vehicles. In Environments for Multiagent Systems, E4MAS, Vol. 3374. Springer, 2005.

105) H. V. D. Parunak, S. Brueckner, J. Sauter, and R. Matthews. Global Convergence of Local Agent Behaviors. In 4th Joint Conference on Autonomous Agents and Multiagent Systems, Utrecht, The Netherlands, 2005.

106) E. Cipi, B. Cico, G.Vasmatics, Intelligent marketing: possibilities of a revolution in supermarkets organization using agent based systems. EBES conference Istanbul, Turkey26-28 May, 2010

107) E. Cipi, B. Cico, Intelligent agents as tools to provide better solutions in database applications: IADIS 2010, ISA Conference, , Freiburg, Germany, 29-31 July 2010

108) E. Cipi, B. Cico Information agents as a new paradigm for developing software applications in database systems Proceedings of the Four-th annual south east European Doctoral student Conference, September, Vol2,ISBN 978-960-9416-04-7, South East European Research Centre, pg.501-508. Greece13-14 2010

109) E. Cipi, B. Cico, Multi agent system as modulated architectures in the software engineering process, , ISBN 978-99956-59-13-4, international Conference on Economy and Technology , Tirana, Albania, 10-12 May 2011

110) Eva Cipi, Alketa Hyso, Software Agents Integrated in Navigation Systems as Better Solutions to Provide Intelligent Behaviors on Dynamic and Unpredicted Environments, Multi-Disciplinary Conference of International Journal Of Sciences And Arts, , Freiburg-Gottenheim , Germany, 28-2 December 2011

111) Eva Cipi , Betim Cico, **"**Science, Technology and Innovation International Journal", New functionalities of a information system adding agents in software applications, Vol1 , , best paper category . ISSN 2223-2257, August 2011

112) Eva Cipi , Betim Cico "International journal of information system and security", University of PitsBurg, USA, Information agents as a new paradigm for developing software applications in database systems, Vol4 , best paper category . ISSN ONLINE 1744-1773 , 1744-1765, April 2011

113) Eva Cipi , Betim Cico The World of Science " International Journal , Online journal, Simulation of an Agent Based System Behavior in a Dynamic and Unexpected Environment, Vol3 ,. www.ijscis.org . ISSN 2221-0741, May 2011,

114) Eva Cipi, Alketa Hyso, Intelligent Agent Solutions to Improve Strategic Level of Self – Management in Information Systems as One of Required Attributes in

Business Environment, International Journal Of Sciences And Arts. ISSN: 1944-6934 , Volume 05, Number 01, 2012

115) M. Pechouchek, D. Steiner, S. Thompson. Proceedings of the Industry Track of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, Utrecht, The Netherlands, 2005.