Әл-Фараби атындағы Қазақ Ұлттық Университеті
ҚР БҒМ ҒК Ақпараттық және есептеуіш технологиялар институты

ӘОЖ 004.93’14

Қолжазба құқығы негізінде

Сатымбеков Максатбек Нургалиулы

Кластер тораптарының жүктемелерін динамикалық теңгеретін Agent-GRID көпагентті грид жүйесін дамыту

6D060200-Информатика

Философия докторы (PhD)
Дәрежесін алу үшін дайындалған диссертация

Ғылыми кеңесшілері:
т.ғ.д., профессор Пак И.Т
Шетелдік ғылыми кеңесшісі:
Матей Бел университеті (Словакия)
PhD, профессор Силади В.

Алматы, 2020

Мазмұны

НОРМАТИВТІК СІЛТЕМЕЛЕР

Бұл диссертация келесі стандарттарға сәйкес сілтемелер қолданылды:

ҚР МЖМБС 5.04.034.-211 «Қазақстан Республикасының Мемлекеттік жалпыға міндетті білім беру стандарты. Жоғары оқу орнынан кейінгі білім.Докторантура». Негізгі ережелер ҚР Білім және Ғылым министрімен бекітілген. «17 » маусым 2011 ж. №261. Астана 2011.

«Диссертацияларды және авторефераттарды рәсімдеу бойынша нұсқаулық», ҚР БҒМ, Жоғары аттестаттау комитеті, Алматы, 2004. МЕСТ 7.1-2003. Библиографиялық жазба.

БЕЛГІЛЕУЛЕР МЕН ҚЫСҚАРТУЛАР

JADE-Java agent development environment
MAS-Multiagent systems
FIFA-
ACL-Agent communication language
AMS-Agent Management System
DF-Directory Facilitator
BDI-Belief, desire, intention
ЕТ-есептеу техникасы

КІРІСПЕ

Зерттеу тақырыбының өзектілігі. Соңғы жылдары әлемде «жасыл есептеу» (Green computing) немесе басқаша айтқанда «экологиялық таза» технологиялар қарқынды дамып келеді [1]. Бүгінгі таңда инженерлік компаниялар ғана емес, сонымен қатар кез-келген жасалынған жаңа технологияларды сипаттау кезінде, экологиялық факторларға байланысты энергияны үнемдеу, қоршаған ортаға аз көмірқышқыл газын шығару және ресурстарды үнемдеу сияқты көрсеткіштерді қолданады.

San Murugesan [2] «жасыл есептеу»-ді қоршаған ортаға зияны тимейтін немесе минималды шығынға алып келетін, компьютерлерді, серверлерді және онымен байланысты ішкі жүйелерді, мысал ретінде айтатын болсақ мониторлар, принтерлер, есте сақтау құрылғылары, желілік байланыс жүйелерін құру, өндіру, қолдану және жою практикасын түсіндіріп өтті. Көріп отырғанымыздай, бұл анықтама компьютерлер мен қоршаған орта сияқты сөздерді қамтиды. Дүниежүзілік деректер орталығында электр қуатын пайдалану жыл санап артып келеді. Бұл өсім компанияның кірісіне қарағанда жоғары болды, сондықтан нәтижесінде ол қаржылық шығындарға әкеледі [3]. Электр энергиясын тұтынудың 50% -ы сайттың инфрақұрылымына кетеді. Осыдан кейін серверлердің көлемі шамамен 30% құрайды, ал қалғандары желілік жабдықтар, қоймалар, жоғары деңгейлі серверлер және орта деңгейлік серверлер арасында бөлінеді. Бұл зерттеу қолданылған инфрақұрылымның компания үшін қымбатқа түсетінін анықтады [4].

Бүгінгі таңда кез-келген білім беру мекемесінде кез-келген қызметкерде немесе студентте бір-бірімен жергілікті және ғаламдық деңгейде байланысқан есептеуіш құрылғылары бар. Есептеу кластерлері болмауы мүмкін, бірақ ғылыми есептерді шешу қажеттілігі туындайды. Бұл жағдайда ғылыми есепті шешу үшін қолжетімді құрылғыларды қолдануға мүмкіндік беретін архитектураларды қолдануғы болады [5]. Ерікті есептеу платформалары бұл функцияны ерікті ресурстарды пайдалану арқылы қамтамасыз етеді. Ерікті есептеу платформалары бұл функцияны ерікті ресурстарды пайдалану арқылы қамтамасыз етеді. Бұл шешімдер үнемі өзгеріп отыратын инфрақұрылым топологиясына және торап инфраұрылымына бейімделуге арналған. Бұл зерттеу көкейтөстілігі, кластерлік жүйелер жоғарғы өнімді есептеулерді шешу үшін қолжетімді болмауы мүмкін, бірақ жедел жоғарғы өнімді есептеу жүргізу қажеттілігі туындайды. Бұл ресми түрде келесі тезис түрінде тұжырымдалған:

Біріншіден, өнімділігі жоғары мәселелерді шешу үшін ерікті есептеулерді сәтті қолдануға болады. Есептеу қолданушылардың ұялы және дербес компьютерлік құрылғыларының ресурстарын қолдану арқылы жасалынады, ал ол құрылғылардың есептеу тиімділігі дәстүрлі кластерлік жүйелермен бірдей болып келеді.

Екіншіден, ерікті есептеудің негізгі мәні, қолданушылардың өз құрылғыларын пайдалануға мүмкіндік беруінде және сол қолданушылардың

құрылғыларының ресурстарын тарту бойынша келесі критерийлерді қарастыруға болады:

- Сол мекемеде жұмыс жасайтын қызметкерлерді есептеуге қатысу үшін өз ресурстарын пайдалануға мүмкіндік беруге міндетті ету.
- Қолданушылар өздерінің құрылғыларын қолдануға мүмкіндік бергені үшін (ақшалай) сыйақы алуы мүмкін.

Осылайша, ерікті компьютерлік желілердің ресурстарын пайдалану үшін Agent-GRID мультиагенттік GRID жүйесін құру қарастырылады. Әрбір есептеу торабы өзінің сипаттамаларын кез-келген уақытта, иесінің іс-әрекетіне байланысты өзгерте алады. Сондықтан, есептеу тораптары базасының негізінде құрылатын GRID жүйесін құруда, есептеу тораптарының динамикалық өзгеруін ескере отырып ұйымдастырудағы заманауи әдістерін дамыту қажеттігін туындатады [6]. Әрине, жеке есептеу тораптарының параметрлерінің жылдам өгеруіне сәйкес оның жүктемесін тиімді қамтамасыз ету күрделі жүйе болып табылады. Жоғарыда айтылғандардан қорыта келе, біз динамикалық өзгермелі параметрлері бар тораптардың негізінде құрылған GRID-тегі мәселелерді тиімді шешу үшін, GRID брокерін орталық торапты таңдау мен олардың арасындағы кіші тапсырмаларды үлестіру уақыты минималды болатындай, торап өнімділігінің өзгерістері жедел бақыланатындай етіп және деректермен алмасу тікелей жүзеге асырылатындай етіп жасау керектігіне көз жеткізді. Осы мәселені шешу үшін біз мультиагентті жүйелердің принциптерін қодану арқылы ұйымдастыруды қарастырамыз.

Диссертациялық жұмыстың мақсаты. Мультиагентті жүйелердегі өзін-өзі басқару және өздігінен білім алу қасиеттерін GRID жүйесін ұйымдастыруда қолдануды зерттеу және құру.

Мақсатқа қол жеткізу үшін келесі міндеттерді орындау қажет:

- GRID жүйесін ұйымдастыру әдістеріне талдау жасау;
- Мультиагентті жүйе негізінде GRID архитектурасын кеңейту.
- Қолданушы тапсырмаларын орындау үшін агенттердің ұйым құру алгоритмін дамыту.
- Құрылған әдістерді тәжірибеде тексеру және нәтежелерді талдау.

Зерттеу нысаны. Мультиагентті жүйедегі оқыту және өзін-өзі басқару, ерікті құрылымдағы мультиагентті жүйе негізінде жүктемені үлестіру.

Зерттеу пәні. Өнімділігі жоғары есептерді шешу және ұйымдастыру үшін ұялы телефондардың, дербес компьютерлердің ресурстарын тартатын мультиагентті жүйе.

Зерттеу әдісі. Тестілеу үшін JADE (Java Agent Development Environment) кітапханасымен бірге қолданылатын Java бадарламалау тілі. Машиналық оқыту алгоритмдері, ұжымдық шешім қабылдау әдістері, мультиагентті жүйелерді құру теориясы, тестілеу барысында алынған барлық деректер визуализацияланады және талданады.

Жұмыстың ғылыми жаңалығы. GRID жүйесін ұйымдастыруда мультиагентті жүйені қолдану әдісі, қолданушы тапсырмасын шешу үшін ұйым құру алгоритмі, тапсырмаларды адаптивті үлестіру кеңейтілген әдісі ұсынылды.

Жұмыстың теориялық маңыздылығы. Диссертациялық жұмыстың теориялық маңыздылығы күрделі жүйелерді жобалау және қолданбалы жасанды интеллект саласындағы білім жиынтығына негізделген. Атап айтқанда, олар жоғары өнімді инфрақұрылымдардағы жүктемені үлестіру туралы маңызды ақпаратты ұсынады.

Жұмыстың практикалық маңыздылығы. Жеке ерікті құрылғы иелерінің ресурстары(компьютер, ұялы телефон) негізінде құрылған жүйе компьтерлерге қызмет көрсету және ауыстыру сияқты шығындарды қажет етпеуі есептеу құнын төмендетеді. Жүйеде қызмет көрсетуге арналған қымбат серверлер мен инфрақұрылымдардың болмауына байланысты шығындарды азайту.

Қорғауға шығарылатын негізгі тұжырым. Мультиагентті ұйымдастыру әдісі динамикалық өзгеріп отыратын GRID-жүйесі ортасында ресурстардың өнімділігін жоғарылатуға мүмкіндік береді.

Зерттеу нәтижелерінің апробациясы. Ғылыми зерттеу жұмысының нәтижелері Ақпараттық және Есептеуіш технологиялар институтында, әл-Фараби атындағы Қазақ ұлттық университетінің информатика кафедрасының ғылыми семинарларында талқыланды және келесі халықаралық конференцияларда баяндамалар жасалды: ИНФОРМАТИЗАЦИЯ ОБЩЕСТВА, V международная. Научно-практическая конференция (2016, Астана, Қазақстан); Международная конференция Молодежь и Наука (2015, Павлодар, Қазақстан); Конференция ИИВТ МОН РК «Современные проблемы информатики и вычислительных технологий» 29-30 июня 2017 года (Алматы, Қазақстан); XIV международная азиатская школа-семинар «проблемы оптимизации сложных систем» 20 июля – 31 июля (2018, Иссык-куль, Қырғызстан); 2017 IEEE 14th International Scientific Conference on Informatics, Poprad,2017; 5th International Conference on Mechanics and Mechatronics Research (2018, Japan);

Публикациялар. Диссертация нәтижелері 17 дана жұмысында жарияланды. 1 мақала [5] Scopus дерекқорымен индекстелген материалдарда жарияланды, 3 мақала ҚР Білім және ғылым саласындағы бақылау комитеті ұсынған журналдарда жарияланды, 3 мақала халықаралық конференциялардың материалдарында жарияланды. Баспадан басқа нәтижелер келесі семинарларда жарияланды: Ақпараттық және есептеуіш технологиялар институтында (Алматы,2019), Әл Фараби атындағы Қазақ Ұлттық Университеті, ақпараттық технологиялар факультеті (Алматы, 2019).

Автор Диссертациялық жұмысты жазу кезінде және қолдау көрсеткендері үшін ғылыми жетекшісі техника ғылымдарының докторы, профессор Пак Иван Тимофеевичке (Ақпараттық және Есептеуіш технологиялар институты, Алматы, Қазақстан) және профессор Владимир Силадиге (Matej Bel университеті, Банска-Бистрица, Словакия) алғыс білдіреді.

1 ҚАЗІРГІ ТАҢДАҒЫ ГРИД ЖҮЙЕЛЕРІНІҢ КОНЦЕПЦИЯСЫ МЕН ГРИД ЖҮЙЕЛЕРІН ҰЙЫМДАСТЫРУ ӘДІСТЕРІНЕ ШОЛУ ЖАСАУ

1.1. Қазыргі таңдағы грид жұмысын ұйымдастыру әдістеріне талдау жасау.

Компьютерлік дәуірдің алғашқы күндерінен бастап адамзат күрделі мәселелерді шешу үшін есептеуіш технологияларды қолдана бастады. Күрделі есептерді шешу қажеттілігінің арқасында, адамзатты есептеулерді автоматтандыру және алғашқы электронды компьютерлерді жасауға алып келді. Қазіргі таңда, есептердің күрделігі өсті және қарқынды өсуде. Бастапқыда мұндай есептеу тапсырмаларын тек үлкен және қымбат суперкомпьютерлер ғана орындай алатын. [7] айтылғандай суперкомпьютерлер айтарлықтай қуатқа ие, бірақ олардың орын алу және электр қуаты шығынына алып келетін зардаптары орасан үлкен. Сондықтан осындай суперкомпьюьтерлерде есептеу бағасының жоғары болуына байланысты тек мемлекеттік деңгейде қолдануға ыңғайлы, шағын көлемдегі мекемелер және зерттеу институттар және жоғарғы оқу орындары үшін өте үлкен қаражатты қажет етеді сондықтан қолдану тиімсіз. Қазіргі таңда 20-ғасырдан бастап сәйкесінше арзан дербес компьютерлер мен мобильді құрылғылар қолданысқа ие бола бастады. Уақыт өте келе, компьютерлер қол жетімді және өнімді бола бастады. Осылайша, алғашқы қол жетімді жабдық негізінде жинақталған жергілікті компьютерлік желі арқылы құрылған үлестірілген жүйелер пайда болды. Мұндай жүйелерде шешілетін мәселелердің тізбесі суперкомпьютерлермен салыстырғанда әлдеқайда төмен болғанына қарамастан, құру және пайдалану құны жағынан төмен.

90 жылдардан бастап телекоммуникациялық технологиялар мен компьютерлік желілердің белсенді дамуы, географиялық үлестірілген дербес компьютерлерге тапсырмаларды жылдам жеткізуге мүмкіндік берді [8]. Бұл, бірінші, университеттердің және ғылыми орталықтардың есептеу желілерінде, содан кейін бүкіл қала және әлем бойынша компоненттері бір бірінен алшақ орналасқан кластерлік есептеу жүйелерін құруға мүмкіндік берді. Сол сияқты көптеген ірі ұйымдар өздерінің үлестірілген есептеу желілерін құру үшін кластерлік есептеу жүйесін ұйымдастыруда ұқсас тәсілді қолдана бастады, дегенмен, 90-ж аяғында, әр ұйымға өз есептерін шешу үшін өздерінің есептеу жүйесін құру тиімді еместігін көрсетті, соның салдарынан универсалды грид жүйесін құру идеясы пайда болды. Грид жүйесінің негізін қалаушылар Ян Фостер мен Карл Кессельман Грид- тің алғашқы анықтамасын берген болатын. Олардың айтуы бойынша: «Компьютерлік желі (Грид) - бұл өнімділігі жоғары компьютерлік ресурстарға сенімді, тұрақты, қарапайым және қымбат емес қол жетімділікті қамтамасыз ететін аппараттық-бағдарламалық инфрақұрылым» болып табылатынын атап өтті [9]. Келесі “грид жүйесінің анатомиясы” мақаласында Стив Тьюке мен бірге ұсынған тұжырымдамаларында әлеуметтік және политикалық аспектілерге байланысты: GRID- Бірнеше қатысушылармен бірге динамикалық түрде өзгеріп отыратын виртуалды организациялардың

арасында үйлесімді түрде есептерді бөліп беру және шешу болып табылады [10]. Негізгі тұжырымдама - жеткізушілер мен тұтынушылар арасында ресурстарды бөлу туралы келісімге қол жеткізу және алынған ресурстарды әртүрлі мақсаттарда пайдалану.

Қазіргі таңдағы грид жүйелерінің кейбір платформаларына шолу 1 кестеде жасалынған болатын:

Кесте 1. Грид жүйесінің платформалары

Жүйе	Платформа	Масштабтылық	Ресурс провайдері
Bayanihan [11]	Web or Middleware	Internet	Volunteer
Boinc [12]	Middleware	Internet	Enterprise
SETI@home [13]	Middleware	Internet	Volunteer
Distributed.net [14]	Middleware	Internet	Volunteer
Entropia [15]	Middleware	LAN or Internet	Enterprise or Volunteer
Qadpz [16]	Middleware	LAN or Internet	Enterprise
Sztaki [17]	Middleware	LAN or Internet	Enterprise
Javelin [18]	Web	Internet	Volunteer
Folding@home [19]	Web	Internet	Volunteer

Зерттеу әр түрлі ортақ қолданыстағы гетерогенді ортаның есептеу ресурстарын тартатын платформаға негізделген. Ресурс провайдерінің сипаттамасын анықтау біздің зерттеуіміз үшін маңызды, сондықтан біздің жүйеде біз ерікті есептеу жүйесін қолдануды ұйғардық. Корпоротивті грид жүйесі қауіпсіздік, тұрақтылық жағынан тиімді бірақ, есептеу қуаттылығы бойынша ерікті негіздегі есептеу платформасымен салыстырғанда тиімсіз[20].

Бүгінгі таңда бірқатар ерікті есептеуіш қосымшалар суперкомпьютерлер жиынтығымен және коммерциялық файлдар орналастырумен салыстыруға болатын ресурстар берді [21]. Есептеу қуаты мен сақтауды ұсынатын миллиондаған пайдаланушылармен байланысқан жобалар қазіргі уақытта медицинада, биоинформатикада, климаттық зерттеулерде, астрофизикада және молекулалық биологияда қолданылады [2]. Ерікті компьютерлік жобалардың мысалдары: Folding @ home [22], Storage @ home [23], Distributed.net [14], Баянихан [11] және Javelin [18].

1.2. Қазіргі таңдағы грид жүйесінің жұмысын ұйымдастыру әдістеріне шолу жасау.

1.2.1 Globus жүйесі

Бүгінгі күнде Globus Toolkit базалық бағдарламалық қамтамасыздандыру негізінде құрылған грид жүйесінің жұмысын ұйымдастыру әдісі кеңінен қолданылуда [24]. Globus Toolkit - бұл дайын шешім емес, стандарттар мен құралдар жүйесі. Globus жобасының негізгі стандарты - OGSA (Open Grid Services Architecture), ашық, кәсіпорын деңгейіндегі есептеу архитектурасы.

OGSA архитектурасы қызметтерді ұсынудың бірыңғай семантикасын, Grid қызметтерін құру, атау, стандартты тетіктерді анықтайды, әртүрлі хаттамалардың орналасуы мен байланысының ашықтығын қамтамасыз етеді және негізгі платформалардың негізгі механизмдерімен интеграцияланады. OGSA техникалық сипаттамалары грид ғаламдық форумы аясында әзірленуде, ол грид қауымдастығы үшін стандарттарды әзірлейді.

Жалпы, GlobusToolkit құралдарының негізінде жасалған грид келесідей жұмыс істейді. Әрбір үшін жергілікті кезектер жиынтығы ретінде грид-ке келіп түскен тапсырмалар, үлестірілген кезекке орналастырылады. Әрбір тапсырма триада <t,r,l> түрінде берілген, мұндағы t – есеп шешудегі күтілетін уақыт, r - оның дәрежесі (бағдарламадағы параллель бұтақтар саны), l MPI бағдарламасының орындалатын файлы. Пайдаланушылар веб-интерфейспен жабдықталған терминалдар арқылы немесе Globus Toolkit пакетін қолдана отырып, кезекке тапсырмаларды қоса алады. Әр терминал өз брокері басқаратын локалды кезекке ие. Жергілікті есептеуіш торап кезектері арасында тапсырмаларды қайта бөлу үшін арнайы алгоритм қолданылады. Осы тәсілге сәйкес әр жергілікті кезек брокері басқа кезектердің брокерлерімен өзара әрекеттеседі және олардың арасындағы тапсырмаларды қайта бөлуге жауап береді. Жергілікті кезектегі тапсырмалар саны белгілі бір сыни мәнге жеткенде, брокер тапсырманы қайта бөлу процедурасын орындайды. Бұл әртүрлі есептеуіш құрылғы арасында тапсырманы біркелкі үлестіруге кететін уақытты азайтуға мүмкіндік береді.

1.2.2. Condor жүйесі

Висконсин-Мэдисон университетінің жасаушылар тобы жасаған Condor жүйесі [25] грид есептеу желісін қолдауға арналған бағдарламалық жасақтама болып табылады. Condor тапсырмаларын басқару тетігі кезекке қоюды, жоспарлауды, басымдылықты және ресурстарды жіктеуді қарастырады. Condor жүйесінде ұсынылған жұмыс реті есептеу уақыты сағатпен есептелетін ұзақ мерзімді тапсырмаларды шешуге арналған. Condor жүйесінде брокерлік есептеу схемасы келесідей ұйымдастырылған. Condor жүйесіне қол жетімді барлық тораптар «ұсыну» бөлімінде: дискідегі жад көлемі, процессордың типі мен жылдамдығы, виртуалды жад мөлшері, орташа жүктеме және т.б. секілді өз мүмкіндіктерін жариялайды (жарнамалайды). Екінші жағынан, қолданушы «ресурстарға сұраныс» бөлімінде өз тапсырмаларының қажеттіліктерін сипаттайды. Condor брокері «Ресурстарға сұраныс» бөліміндегі «ұсыныс» бөлімінде көрсетілген ресурстармен сұранысты қанағаттандыратын брокер ретінде әрекет етеді. Сонымен қатар, жүйе өтінімдерді қарастырудың бірнеше деңгейінің тәртібін қамтамасыз етеді: ресурстарды бөлу басымдығы, пайдаланудың басымдылығы және сол қосымшаларды қанағаттандыратын тораптар арасындағы басымдылық.

Жүйенің аппараттық архитектурасы компьютерлердің үш түрін қамтиды:

Орталық брокер - жоспарлау функцияларын орындайды: барлық машиналар туралы ақпарат жинайды және қолда бар ресурстар мен қолданушы

арасында делдал ретінде әрекет етеді. Орталық брокер сәтсіздігі бүкіл кешеннің тоқтап қалуына әкеледі.

Пайдаланушы торабы- тапсырма іске қосылған кез-келген компьютер.

Аралық есептеу торабы- брокер басқаратын және тапсырмаларды тікелей орындайтын арнайы компьютер.

1.2.3. AppLeS жүйесі

AppLeS жүйесінің негізгі тәсілі, брокерлерді, бос ресурстарды іздейтін қосымшалар ретінде қолдануға негізделген [26]. Бұл өзара әрекеттесу моделінде тораптар үлестірілген кластерлерінде ресурстарды басқаруды жүзеге асыратын пайдаланушылар мен көптеген жергілікті брокерлер арасындағы аралық байланыс қызметін атқаратын сыртқы брокер бар. Арнайы торап қолданушы таңдаған белгілі бір брокер пайдаланушының міндеттері бағытталған кластерді анықтайды. Кластерді таңдау: жұмыс жүктемесінің дәрежесі, есептеу үшін қажетті деректердің қол жетімділігі және т.б. көптеген факторларға байланысты. Пайдаланушы таңдаған сыртқы менеджер кластерді таңдағаннан кейін, жергілікті кластер менеджері жаңа тапсырманы кезекке қояды. Сыртқы және жергілікті брокерлерден басқа жүйеде қашықтағы кластерлердегі мәліметтерге қол жеткізуге, деректерді жылжытуға және көбейтуге (репликациялауға) жауап беретін деректер басқарушысы бар. Бұл жүйеде пайдаланушылармен, сыртқы брокермен, жергілікті брокерлермен және деректер брокерімен өзара әрекеттесудің әртүрлі сценарийлері мүмкін. AppLeS жобасындағы ресурстарды брокерлендірудің негізгі идеясы брокер нақты бағдарламалық қосымшаны тиімді орындау үшін ресурстар жиынтығын таңдайды. Жүйенің әртүрлі қолданушылары мүлдем басқа критерийлерге сәйкес ортақ ресурстармен бөлісетін қосымшаларын оңтайландыруға тырысады. AppLeS жобасының негізгі тұжырымдамасы - бағдарламалық қосымшалардың ерекшеліктерін ескере отырып, көптеген жеке брокерлердің ісін жүзеге асыру.

1.2.4. X-COM жүйесі

Мәскеу мемлекеттік университетінің ғылыми-зерттеу орталығында жасалған X-Com грид жүйесі ағаш архитектурасына ие [27]. Жүйе келесі негізгі компоненттерді бөлуге болатын клиент-сервер архитектурасына негізделген:

брокерлер-бастапқы деректер жиынтығын блоктраға бөлуге, Есептеу тораптары арасында қосалқы тапсырмаларды үлестіруге, бағынышты есептеу тораптары жұмысын үйлестіруге, есептеу нәтижесінің тұтастығын бақылауға, нәтижені біртұтас етіп жинауға жауапты есептеу тораптары жиыны.

Тапсырма серверлері - бір нақты қосымшаның орындалуын қамтамасыз ететін есептеу тораптары. есептеу тораптарына келетін соңғы нәтижені өңдейтін және деректер бөліктерін құратын, қолданбалы тапсырманың серверлік бөлігімен қарым-қатынас орнатады. Тапсырма сервері клиенттерге бөліктерді бөліп беру логикасын жүзеге асырады.

Клиенттер- кіріс деректері тапсырма орындалатын есептеу торабына серверден тапсырма келіп түседі, ал есептеу тораптарында орындалған тапсырма қайтадан тапсырма серверіне жіберіледі. Аралық есептеу торабы кіріс деректер блогын өңдеуге, серверден есептеудің жаңа тапсырмаларын сұрауға және есептеу нәтижелерін тапсырма серверлеріне беруге жауапты.

Клиенттер менеджері- бірінші қосылғаннан бастап, барлық клиенттер жүгінетін қызметтік есептеу торабы. Клиент менеджері есептеу түйіндерін, талаптарға сай келетін тапсырма серверлеріне бағыттайды. Тапсырма аяқталғаннан кейін клиенттер қайтадан менеджерге жүгінеді. Қызметтік есептеу торабы мен орындаушы есептеу торабы арасындағы әрекеттестікті ұйымдастыру үшін, сонымен қатар көптеген есептеуіш торабын қосу кезінде, жүйенің масштабталуын қамтамасыз ету үшін, жоба аралық брокерлерді қолданады. Осының арқасында жүйе, өздігінен, еркін түрдегі схеманы алады. Жалпыланған түрде X-COM негізінде жасалған грид жұмысын келесі түрде ұсынуға болады. Жүйені біз, орындаушы есептеу торабы бар кластерлер қосылған қызметтік есептеу торабының брокері ретінде қарастыра аламыз. Тапсырманы орындауға сұраныс және оны шешуге арналған мәліметтер тапсырма серверінен жүйеде бар кез-келген брокерлерге жіберіледі, сол жерден ол желі арқылы таратылып, тапсырма тораптың белгілі бір ішкі жиынына орналастырылады.

1.3 Қазіргі ГРИД-жүйесінің жалпыланған моделі.

Қолданыстағы грид жүйелеріндегі мәселелерді тиімді шешу мүмкіндігін бағалау үшін грид жүйесінің жалпыланған моделін құру қажет. Жалпы алғанда, бұрын қарастырылған грид жүйесінің кез-келгені келесі түрдегі көптеген түйіндерден тұрады:

- *тұтынушы ДК.* Бұл соңғы тұтынушылардың грид-пен байланыс орната алуы үшін және тұтынушы интерфейсінің бағдарламалық инструменттері орнатылған компьютерлер. (атап айтқанда, тапсырмаларды орындалуға жіберу және нәтижені алу);

орындаушы есептеу тораптары. Есептеу тапсырмаларын шешуге тікелей қатысатын орындаушы тораптары. грид тұрғысынан, орындаушы есептеу торабы бірігіп есептеу желісін құрайды;

қызметтік есептеу тораптары. грид-те есептеу процесін ұйымдастыру (жоспарлау) функцияларын орындайтын торап.

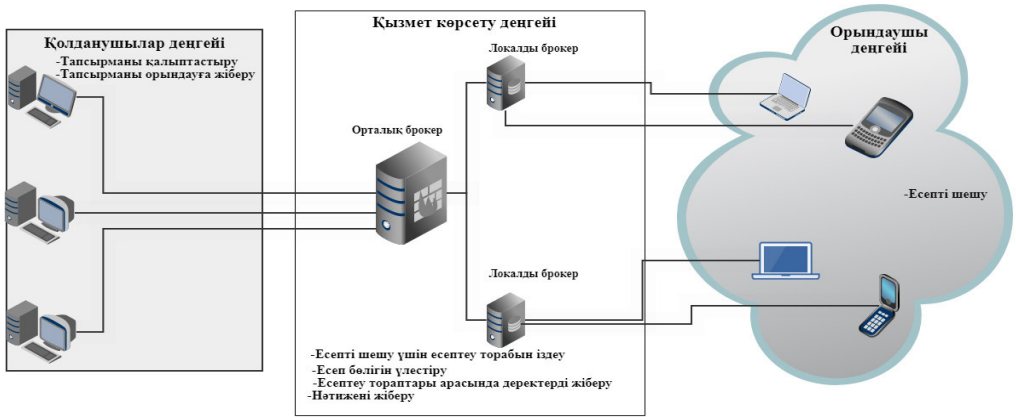
Бұл жағдайда грид жүйесін логикалық түрде үш деңгейге бөлуге болады, олардың әрқайсысы белгілі бір типтегі есептеу тораптарынан тұрады және грид функцияларының белгілі бір жиынтығын орындауға жауап береді:

-*Қолданушы деңгейі*- Грид жүйесінде шешілуі тиіс, тапсырма құрылатын ақпараттық-бағдарламамалық-құралдар.

-*Орындаушы деңгейі*- Грид-те есепті шешу үшін қолданылатын ақпараттық-бағдарламамалық-құралдар.

-*Қызметтік деңгей*- Грид жұмысын қамтамасыз ететін ақпараттық-бағдарламамалық-құралдар.

Жалпы алғанда, қазіргі грид жүйелерінің көпшілігі бірдей дәрежелі архитектура бойынша құрылады, 1 суретте сол архитектураның схемасы көрсетілген. Бұл схемада басты рөлді, әдетте, қызметтік есептеу торабы иерархиялық көп деңгейлі ағаштың көмегімен жүзеге асырылатын және грид жұмысын жоспарлау функцияларын орындайтын [9] қызмет деңгейі ойнайды. Іс жүзінде, қызмет көрсету деңгейі грид пайдаланушысы мен қызметтік есептеу торабы желісі арасындағы байланыс. 1 суретте жалпы қазіргі таңдағы грид жүйесінің моделі көрсетілген.



Сурет 1-Қазіргі таңдағы GRID жүйесінің жалпыланған моделі

Есеп шығару кезінде грид жүйесінде іске асырылуы керек функциялардың тізімін толығырақ қарастырайық.

Тапсырманы қалыптастыру. Әр түрлі грид жүйелері әр түрлі схемаларға сәйкес құрастырылған, бағдарламалауда жалпы қабылданғаннан бір немесе басқа жолмен ерекшеленетін стандарттарды қолданады. Сондықтан грид-де шешілетін тапсырма осы ерекшеліктерді ескере отырып құрылуы керек.

Тапсырманы орындауға жіберу. Тапсырма құрылғаннана кейін, қолданушы оны грид-ге енгізілген қызметтік есептеуіш тораптың біреуіне жібереді. Жүйені жобалау схемасына байланысты ол қатаң түрде анықталған қызметтік есептеуіш тораптың арасында пайдаланушы өз міндетіне лайықтысын таңдау керек.

Ресурстарды жоспарлау - грид жүйесінде есептеу ресурстарының үлестіруін басқару және қолданушылардың есептерін шешу. Қазіргі заманғы грид-де ресурстарды үлестіру тапсырмасы келесі ішкі тапсырмалардан тұрады.

- *компьютерлік желіде қолданушы есебін шешуге жарамды есептеу торабын іздеу.* Компьютерлік желінің барлық есептеуіш тораптары белгілі бір қолданушы есебін шешуге қойылатын талаптарды қанағаттандыра алмайтындықтан, компьютерлік желіден оның шешілуін қамтамасыз ете алатын белгілі бір есептеуіш торап жиынын таңдау қажет болады.

- *қолданушы тапсырмаларын анықталған есептеу тораптарының арасында бөлу.* Қолданушы мәселесін шешуге жарамды есептеу тораптарының жиынтығы табылғаннан кейін, проблеманың бөліктерін алынған есептеу тораптарының жиынтығына орналастыру проблемасы туындайды.

- *Есепті шешу*. есептеу желісінде есептеу тораптарының қажетті саны бөлінгеннен кейін және арасында есеп бөліктері бөлінген бойда, есепті шешу процесі басталады.

- *Есептеу тораптар арасында деректерді Ауыстыру.* Есептеу жүйелері кіші тапсырмаларды шеше алуы үшін бір тапсырмадан екінші тапсырмаға жіберілетін мәліметтер сәйкес есептеу тораптары арасында ауыстырылуы керек.

-*Есептеу нәтижелерін қолданушыға жіберу*. Есеп нәтижесі Грид компьютерлік желісінің кез-келген есептеу торабында пайда болғанда, есептеу торабы нәтижені қызығушылық танытқан қолданушыға жіберуі керек.

1.4 Қазіргі таңдағы GRID жүйесінің мәселелері

Грид модельдерінің жұмысын зерттеу нәтижесінде [28] пайдалы жүктеме коэффициенті жоғары болады егер, есепті шешу процесінде есептеу тораптарының параметрлерінде өзгерістер аз болса және жекелеген есептеу тораптары арасындағы мәліметтердің алмасуы аз болған жағдайда.

Қазіргі заманғы грид-дің негізгі ауқымын шектейтін мәселе ол - жекелеген тапсырмалар арасындағы интенсивті мәліметтермен алмаспайтын есептерді шешу. Осы типтегі мәселелер әлсіз байланысқан деп аталады.

Сонымен бірге, грид-де есептеу тораптары арасында жиі ақпарат алмасу қажеттілігін туындататын есептер бар. Осы мәселеге мысал ретінде күрделі жүйелерді модельдеу секілді есептерді айтып кетуге болады [29].

Бүгінгі күні өндіріс үшін осындай мәселелерді шешу ішкі тапсырмалары арасында мәліметтерді жылдам беруге мүмкіндік беретін жоғары өнімді есептеу жүйелері қолданылады [30].

Алайда, мұндай есептеу жүйелері қымбат және оларды басқару қиын, сондықтан осындай мәселелерді шешуде грид–ті қолдану жұмыстың актуалдығын көрсетеді.

Жоғарыда айтылған ойларды талдай отырып, грид-дің орталықтандырылған брокерімен байланысқан мәселелерді шешуде тиімділігінің төмендеуін мына ерекшеліктер арқылы көрсетуге болады:

-есептеу барысында деректерді бір тораптан екінші торапқа жіберу мәселесі туындайды, ол торап бірінші сол деректі орталық брокерге жібереді, ал орталық брокерден сол есепті қажет ететін торапқа жібереді. Осылайша, деректер бірнеше рет тасымалдау керек – алдымен есептеу торабынан брокерге, содан кейін брокерден басқа есептеу торабына. Егер деректерді бір есептеу торабына тікелей беру мүмкін болса, тасымалдау уақыты айтарлықтай қысқаруы мүмкін [31];

-қосалқы тапсырмалар арасында тасымалданатын деректер қажет емес болғанша бір жерде сақталуы керек, бұл орталық брокерден үлкен жадты қажет етеді;

Мұның бәрі, өз кезегінде, осындай типтегі мәселелерді шешу кезінде жүйе жұмысының күрт төмендеуіне әкеледі.

Мысалы, эксперименттік зерттеулер көрсеткендей, әлсіз байланысқан мәселелерді шешкен кезде қазіргі грид жүйелерінің жылдамдығы 75-90% құрайды [32]. Сонымен қатар, байланысты есептерді шешкен кезде жүйе жұмысы 40% -ға дейін төмендейді [33].

Соңғы жылдары ғаламдық желі Жердің көп бөлігін қамти бастады, ал дамыған елдерде компьютерлер әр отбасында пайда бола бастады, ДК-нің орташа қуаты артып келеді. Сонымен қатар, уақыттың едәуір бөлігі, мұндай дербес компьютерлер бос немесе минималды есептеу жүктемесімен жұмыс істейді. Сондықтан, қазіргі уақытта осы бос ресурстарды грид жүйелерін құру үшін есептеу түйіндері ретінде пайдалану өзекті мәселе болып отыр [34]. Негізгі мақсат - жүйенің инфрақұрылымын құру және қолдау шығындарын азайту, ескірген және ақаулы жабдықты жаңарту, персоналды, үй-жайларды күту және тағы басқалар есебінен есептеулерді ұйымдастыруға кететін шығындарды азайту.

Жоғарыда айтылғандай, жеке дербес компьютерлердің есептеу ресурстарына негізделген (Есептеу торабы- «фрилансерлер») грид-жүйелерін құру тәсілдерінің басты артықшылығы, есептеу шығындарын төмендету болып табылады. Ұйымның көмегімен инфрақұрылымды ұстап тұру, есептеу торабы жабдықтарын сатып алудың, жөндеудің, техникалық қызмет көрсетудің және жаңартудың қажеті болмайды, мұның бәрін «фрилансерлердің» есептеу тораптарының иелері жүзеге асырады. Сонымен бірге, заманауи грид жүйелерінде «фрилансерлерді» қолдану жүйелі мәселелерді шешу кезінде жүйенің тиімділігін айтарлықтай төмендетеді. Шынында да, байланысқан есептерді шешу үшін, жоғарыда көрсетілгендей, тораптар арасында қарқынды ақпарат алмасуы қажет, ал екінші жағынан, олардың иелерінің әрекеттеріне байланысты торап параметрлерінің өзгеруі салдарынан жүйе өнімділігінің төмендеуі мүмкін

Жағарыда айтылған мәселелерді қорыта келе параметрлері динамикалық өзгеріп отыратын тораптардан құрылаған GRID жүйесінде есептеулерді тиімді жүргізу үшін GRID жүйесінің жұмысын торапты таңдау және тораптарға есеп бөлігін жіберу минималды, тораптардың өнімділігінің үнемі бақыланып отыруын қамтамасыз ету, есеп бөліктері торап арасында алмасуы тікелей болуы қажет. Осы Функцияларды мультиагентті жүйелердің принциптарын қолдану арқылы жүзеге асыруға болады.

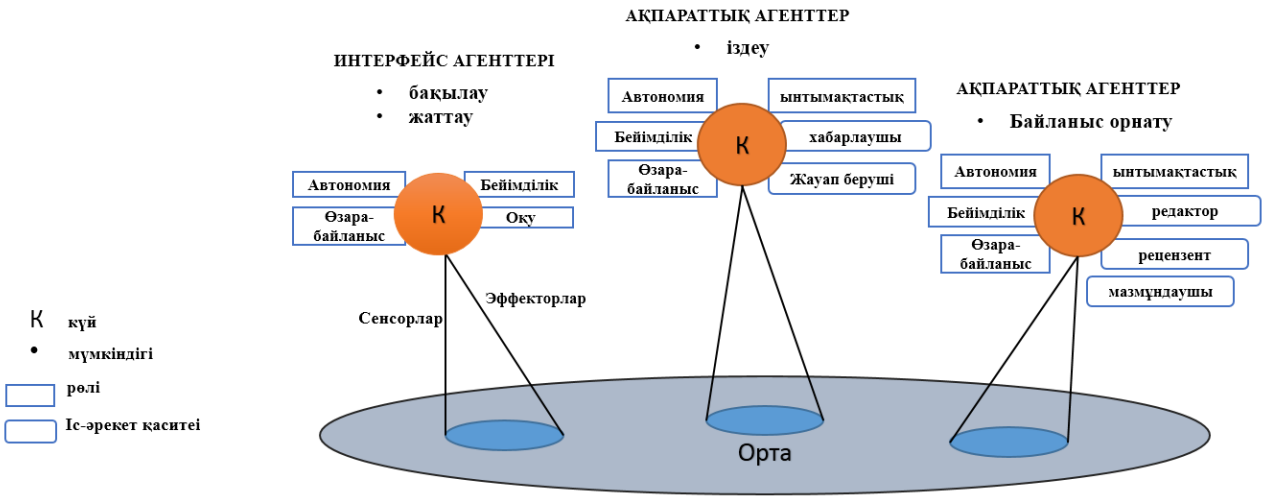
1-ші бөлім бойынша қорытынды:

- Қазіргі таңдағы грид жүйесінің бірнеше платформаларына зерттеу жүргізілді
- Грид жүйенің ұйымдастыру мәселесіне талдау жүргізілді
- Талдау нәтижесінде қазіргі грид жүйелерінің артықшылықтары мен кемшіліктеріне талдау жасалынып және сол кемшіліктерді ескере отырып жаңа ұйымдастыру әдістері мен алгоритмдерді қажет ететіні туралы анықталды.

2 ГРИД ЖҮЙЕСІН ҚҰРУДА МУЛЬТИАГЕНТТІ ТЕХНОЛОГИЯНЫ ТАҢДАУДЫҢ НЕГІЗДЕМЕСІ

Мультиагенттік жүйелер – бұл күрделі міндеттер мен мәселелерді шешу үшін көптеген өзара әрекеттесуші агенттерден тұратын жүйелерді қолданатын жасанды интеллект бағыты [35].

Көпагентті жүйелер теориясында («мультиагенттік жүйелер» атауы да жиі кездеседі) қарама – қарсы үдеріс негізге алынады. Бір агент жаһандық мәселе туралы бар болғаны аз ғана түсінікті игереді деп есептеледі, демек ол жалпы міндеттердің кейбір бөлігін ғана шеше алады деген сөз. Осыған байланысты күрделі міндеттерді шешу үшін кейбір көптеген агенттерді құрып, олардың арасында бірыңғай көпагентті жүйелерді құруға мүмкіндік беретін тиімді өзара әрекеттесуді ұйымдастыру қажет. Көпагентті жүйелерде әрқайсысы ұйымдар мен топтардың мүшелері болып саналатын бүкіл агенттер арасында белгіленген ережелер бойынша міндеттердің барлық ауқымы бөлінеді. Күрделілігі агенттің мүмкіндігінен анықталатын қандай да бір рөлді әрбір агентке беру тапсырмаларды бөлуді білдіреді [35]. 2 суретте мультиагентті жүйелердің жұмыс жасау моделі көрсетілген.



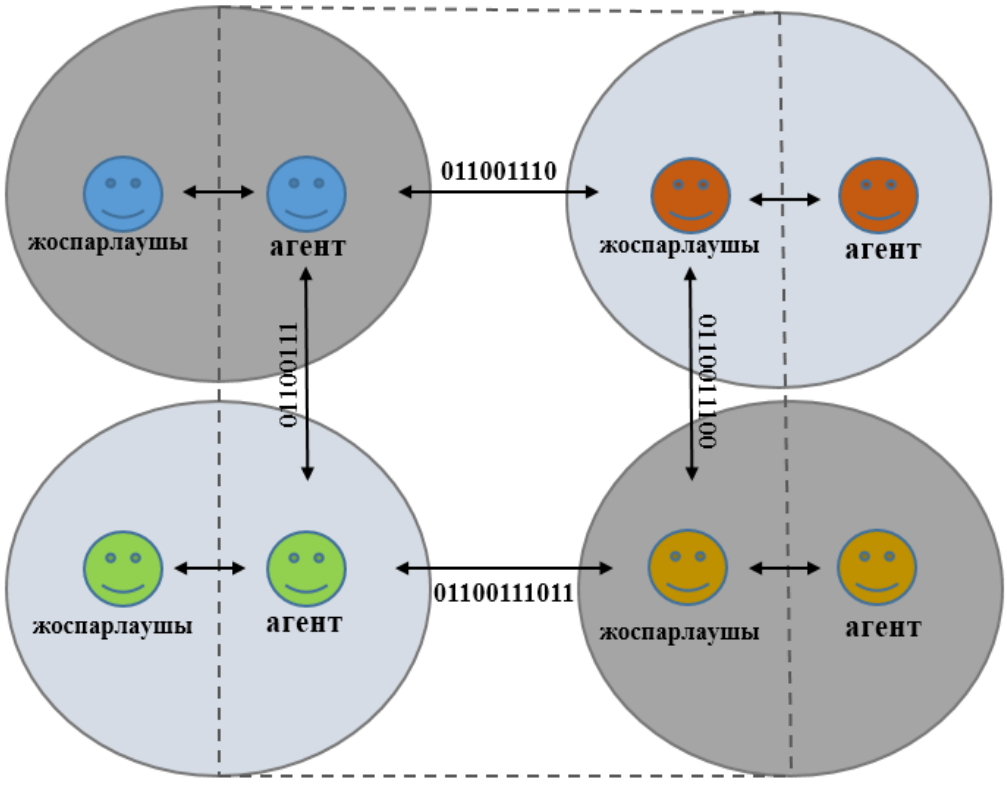
Сурет 2- Көпагентті жүйелер

Көпагентті жүйелерде міндеттерді бөлу үдерісін ұйымдастыру үшін мәселенің бөлінген шешімінің жүйесі немесе орталықтандырылмаған жасанды интеллект жүйесі құрылады. Бірінші нұсқасында кейбір бірыңғай «орталықтың» басқаруымен жаһандық міндеттерді бөлшектеп байланыстыру үдерісі және табылған шешімдер композициясының кері үдерісі болады. Сонымен көпагентті жүйе агенттер үшін белгіленген рөлге және жаһандық міндеттердің қосалқы міндеттерге бөліну нәтижелеріне қарай қатаң түрде жоғарыдан төмен қарай жобаланады. Міндеттерді бөлу орталықтандырылмаған жасанды

интеллектіні қолданған жағдайда агенттердің өзара әрекеттесу үдерісінде іске асады да спонтанды сипатта болады. Бұл көпагентті жүйелерде резонанстық, синергетикалық эффектілердің жиі пайда болуына әсеп соғады [36].

Көпагентті жүйе технологиясы өзінің белсенді дамуында он жылдық тарихы болса да, әлі қалыптасу кезеңінде жүр. Құрамдас жүйелер мен негізгі түсініктерді нысандаудың теориялық негізінің саласында, әсіресе діл түсінігін нысандау саласында белсенді зерттеу жұмыстары жүргізіліп жатыр. Бұл бөліктегі негізгі жетістіктер әлі практикалық түрде жүзеге асырудың акпестісіне бағыттала қоймады, әзірге практикадан алшақ келеді [37]. Сондай-ақ ділдік түсініктерді ресімдегенде жасанды интеллекте әзірленген мүмкіндік пен анықсыздыққа негізделетін әдістер, толығымен анықталмаған түсініктер, нашар құрылымданған түсініктерге арналған барлық тәсілдер толықтай ескерілмейді. Бұл көлемді, жаңа және таза іс - әрекет өрісі өзіне лайықты мамандарға арналғандығы көрсетіледі.– өзара алмасу арқылы сыртқы ортада әрекет ететін жүйе.

Агент –өзге де агенттермен бірге коммуникациялық байланыс жасай отырып іс- әрекет (іс-қимыл)жасап өз нысанасына ұмтылатын программалық немесе аппараттық модуль/объект(тетігі,кетігі)[38]. 2 суретте Агенттердің өзара қызметі бейнеленген.



Сурет 3-Агенттердің қызметі

Обьект - бұл біркелкі өзіндік ерекшелігі мен тәртіп ережесі бар шынайы өмірдің көптеген мәндерінің (даналарының) немесе виртуалды мәндердің абстракциясы.

Келтірілген анықтамаларға сүйенсек, обьект түсінігі агентті анықтауда елеулі рөл атқаратын ортаның болуымен байланысты емес. Негізінде обьект өзіне ұқсас обьектілердің болғанын талап етпейді, ал агент жалғыз бола алмайды. Сонымен, агент – барлық ерекшеліктерді игерген, сонымен қатар қосымша сапасы да бар обьектінің бір тармағы болады.

Прагматикалық көзқараспен қарағанда, агент – бұл белгіленген міндетті шешуді қамтамасыз ететін және жеке агенттермен алуға болмайтын кешенді мәселелерді шешу үшін басқа агенттер желісімен өзара байланыста әрекет ететін жүйе. Мультиагенттік желідегі агенттер гетерогенді, яғни әр түрлі класстарға жатады. [39]

Бір немесе бірнеше бірлестікке ұйымдастырылған және белгілі бір міндетті шешуге арналған бағдарламалық агенттердің көбісін Мультиагенттік жүйе (МАЖ) деп түсінеміз [40].

Зерттей келе Көпагентті жүйелерді жобалау мен талдауға керекті әр түрлі ұжымдасқан дерексіздіктер анықталды. Келесі кезекте жаңадан жобалау барысы болады.

2.1 Анықтау және жіктеу

Агенттерді зерртеушілер бірнеше анықтама берді. Солардың ішіндегі агенттер туралы сипаттың кейбір жалпы түсініктері: дербестік, мақсат-бағдарлануы, ынтымақтастық, икемділік, өзін-өзі басқару, уақытша үздіксіздігі, сипаты, икемділік, ұтқырлық, үйренуге қабілетті болу мүмкіндігі.

Автономды агенттер деп сол жүйенің ішінде орналасып және ортаның бір бөлігі бола отырып, сол ортаның әсерін сезе алатын,болашақта сол әсерге жауап қайтара отырып, сол қабылдаған әсерді қолдана отырып болашақта қандай іс-әрекет жасайтынын анықтап отыруды айтамыз.

Жасанды интеллект саласында, ақалды агенттерге қатысты соңғы кең мағынадағы анықтама ол логика-бағдарламалау базасының парадигмасын ауыстыру [41].

Агенттердің кең ауқымды типтері бар.

- Агент интерфейсі компьютерлік программа болып табылады, олар жасанды интеллект әдістерін қолдана отырып, қолданушыларға компьютерлік есептермен еркін байланысуға мүмкіндік беру.
- Мобильді агенттер хосттар мен байланыса отырып бүкіл әлемдік желі жүйесінде (WWW) еркін қозғала отырып, қолданушының өзінің атынан іздеген ақпаратын тауып, қолдагушыға қайтара алатын программалық жабдықтама процесі болып табылады.
- Кооперативтік агенттер көп агентті жүйеде бір ортадағы агенттермен хабар алмасып және оларға жауап қайтарып отыра алады. Осындай агенттердің сенсорлық қабілетінің шектеулі болуына байланысты ортадағы күйі

өте жіңішке болу мүмкін. Қарым-қатынас, мақсаты тек өзіне ғана тән емес, басқа агенттердің мақсаты мен бір болған кезде пайда болады.

- Реактивті Агенттер олар қоршаған ортада ішкі символикалық модельге ие емес. Оның орнына, реактивті агенттер қоршаған ортаға кейбір ортаға немесе оқиға реттелуін ынталандыру немесе кірісіне «жауап» береді. Бұл қоршаған ортадағы оқиға, агент реакциясы немесе жауап триггерлері болып табылады. Агент есептеу өрісіне экономика, бизнес (коммерциялық дерекқорлар), басқару, телекоммуникация (желі басқару) және (е-сауданы дамыту үшін сияқты) электрондық қоғамдар кіреді. Деректер базасын, статистика, және машиналық оқытудан бастап техникасы кеңінен таралған агент қосымшалары пайдаланылады. Телекоммуникация саласында, агент технологиясы (құны мен орындау тұрғысынан) тиімді қолдау үшін пайдаланылады, мақсаты бәсекеге қабілетті телекоммуникациялық орталарда тіркелген және ұтқыр пайдаланушылар үшін қызмет ету [42].

2.2 jade платформасы

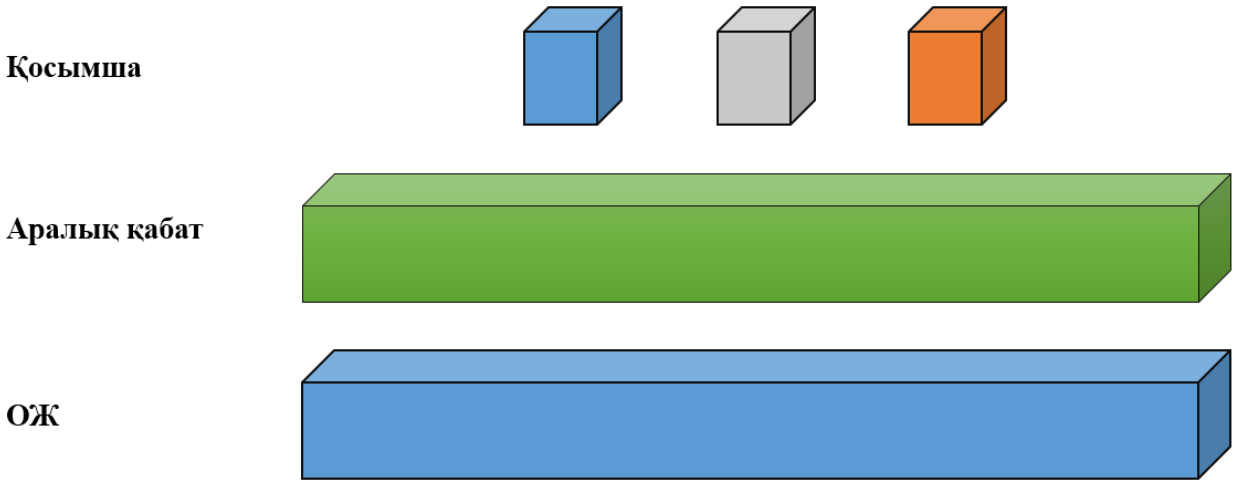
Java Agent Development Framework (JADE) – интеллектуалды агенттерге арналған FIPA- стандартын ұстанатын мультиагенттік жүйелер мен ұсыныстарды өңдейтін бағдарламалық орта [43].

Оған жатады:

- *агенттердің орындау ортасы*. Агент тіркеледі де ортаның басқаруымен жұмыс істейді;
- *класс кітапханасы***,** агенттік жүйелерді өңдеу үшін қолданылады;
- *графикалық утилит жиынтығы*, әкімшілік және белсенді агенттердің өмірлік іс - әрекетін бақылауға арналған.

JADE бағдарламалық ортасы Java тілінде кез келген жобаға қосыла алады. JADE – бұл «нүкте - нүкте» транспорттық сәулет негізінде бөлінген мультиагенттік ұсыныстарды құруға арналған, TILAB компаниясы жасап шығарған аралық қабатты бағдарламалық қамтамасыз ету (4 суретті қараңыз). Интеллекті де, бастама да, ақпарат та, ресуртар және бақылау да белгіленген желі компьютері бойынша бөлінгендей, толығымен мобильдік терминал бойынша да бөліне алады [44]. Орта JADE терминологиясында агенттер деп аталатын тораптармен динамикалық түрде өзара әрекеттестікте бола алады. Агенттер бағдарламалық орта талаптарына және қажеттіліктеріне сай жүйеде кейде пайда болып, кейде жоғалып кетеді [45].

Тораптар арасындағы коммуникация желі түріне (сымдық, сымсыз) байланысты болмайды. Олар толығымен симметриялық болып табылады және әрбір торап сұраныстарды ынтагерлікпен өзгертіп, оларға жауап бере алады.



Сурет 4- Аралық қабат

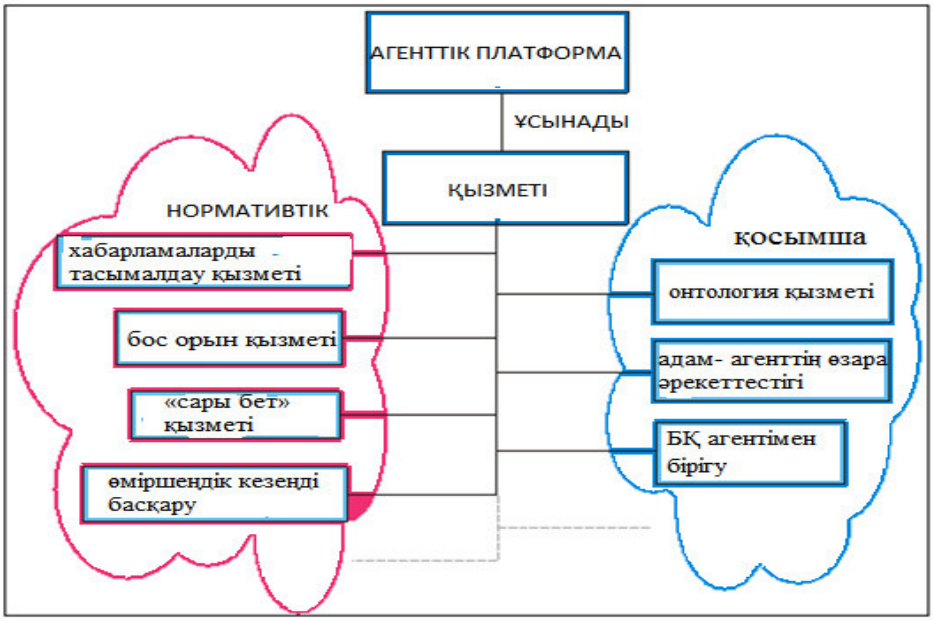
JADE платформасы толығымен Java тілінде әзірленген. Платформаның негізге алынатын үдерістері мыналар:

Қызметтік сәйкестік - JADE өнімі FIPA спецификациясына (5– суретті қараңыз) сәйкес жасалған. JADE – агенттер осы стандартты ұстанатын бөтен агенттермен де өзара әрекеттесе алады [46].

біркелкілік – JADE өнімі Java платформасының нұсқасына да, желінің базалық құрылғысына да байланысты емес қолданбалы бағдарламалық интерфейстердің (API) гомогендік жиынтығын ұсынады. Толығырақ айтсақ, осы БҚ орындау үдерісінде J2EE, J2SE, J2ME қоршаулары үшін бірдей API-ді ұсынады. Әзірлеушілер ұсыныстарды дамытқанда Java–ны орындау ортасының түрін анықтап алуы тиіс.

Қолдану қарапайымдылығы – қарапайым жәнеинтуициялық – түсінікті API интерфецстерінің жиынтығы қолданушыдан аралық қабатты БҚ – дің күрделі логикасын жасырады.

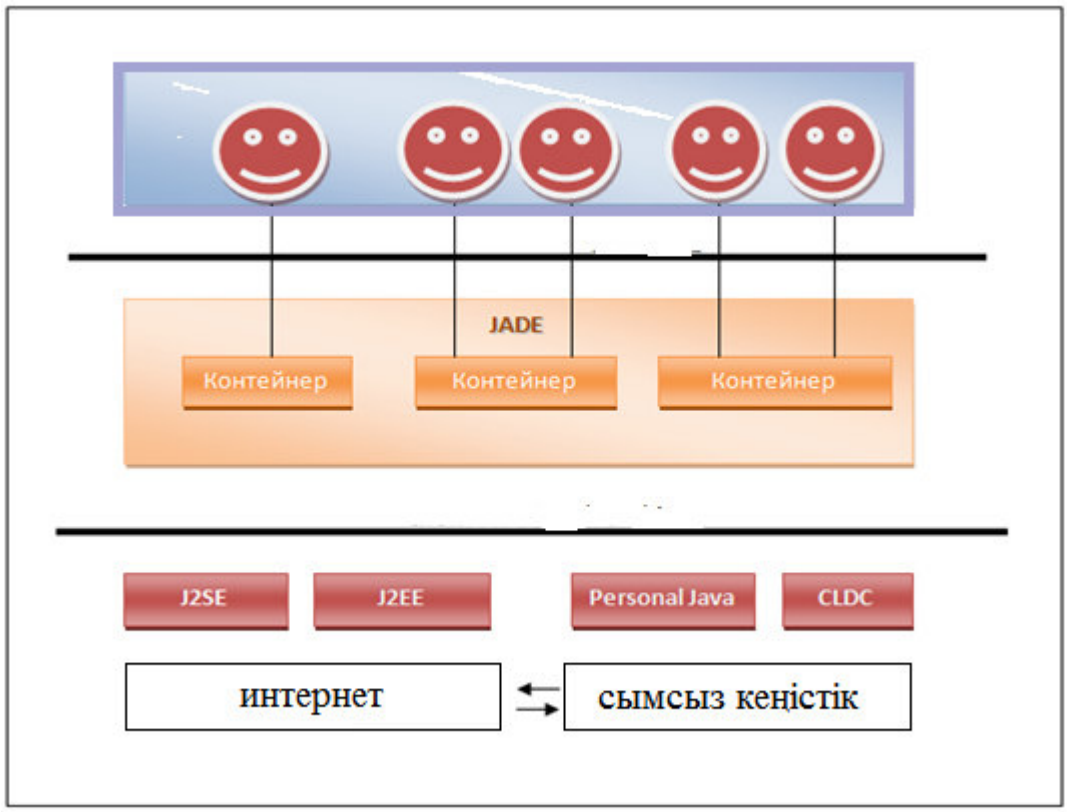
«Әдіс бойынша әзірлеу» үдерісі – программистерге аралық қабаттың БҚ ұсынатын барлық мүмкіндіктерді қолданудың қажеттігі жоқ. Программистерден платформаның қолданылмаған қызметтері туралы бірнәрсе білу талап етілмейді. Іске қосылмаған қызметтердің ешқайсысы қосымша есептеулі үстеме шығындарды тудырмайды.



Сурет 5- FIPA стандарты

JADE платформасына қолданбалы агенттерді әзірлеу үшін талап етілетін бағдарламалық кітапханамен (яғни Java – класс жиынтығы) қатар, құрылғыда агент орындалғанға дейін белсенді болуы тиіс базалық қызметті ұсынатын орындау ортасы жатады. JADE – нің әрбір данасы орындау барысында контейнер деп аталады (себебі онда агенттер «бар» (тұрады)). Барлық контейнерлердің жиынтығы платформа деп аталады және агенттерден (сонымен бірге ұсыныс әзірлеушілерден) өте төменгі деңгейде (аппараттық қамтамасыз ету, операциялық жүйелер, желі түрлері, JVM) орналасқан мезанизмдердің бөлінген сипаты мен күрделілігін жасыратын біркелкі қабатты ұсынады [47].

JADE бағдарламалық қамтамасыз етуі (6 сурет) J2ME CLDC/MIDP1.0 ортасымен үйлеседі. Ол құрамында Nokia 3650, Motorola Accompli008, Siemens SX45, PalmVx, Compaq iPaq, Psion5MX, HP Jornada 560 болған түрлі мобильді терминалдармен GPRS желісінің базасында сынау стендісінде тестен өтті. Орындау үдерісінде JADE 100КБ жуық оперативтік зерденің орнын алады, бірақ бұл JVM мен JADE – нің бірлескен компеляциясы есебінен 50КБ – ға дейін азаюы мүмкін. JADE өнімі шектелген ресурстары бар ортаны шектеулермен қанағаттандырып қана қоймай, NET және J2EE сияқты күрделі сәулетпен біріктірілген өте әмбебап болып табылады, бұл жерде JADE бірнеше өзара әрекеттесетін жақтармен қолданбалы белсенді ұсыныстарды орындауға арналған қызмет болады. Оперативтік зерделе JADE алатын орынның шектелген көлемі осы бағдарламалық қамтамасыз етуді Java технологиясын ұстанатын барлық ұялы телефондарға орналастыруға мүмкіндік береді [48].



Сурет 6-JADE сәулеті

Қызметтік көзқарас бойынша JADE бағдарламалық құралы стационарлық және мобильді ортада «нүкте - нүкте» сияқты бөлінген ұсыныстар үшін қажетті базалық қызметтерді ұсынады. Ол әрбір агентке динамикалық түрде басқа агенттерді табуға және олармен «нүкте - нүкте» жүйе парадигмасына сәйкес хабарламалармен алмасуға мүмкіндік береді. Ұсыныс көзқарасы бойынша әрбір агент ерекше бір атпен құпияланады да бірнеше қызмет түрлерін ұсынады. Ол өз қызметтерін тіркеп, түрлендіре алады немесе аталмыш қызметтерді ұсынатын агенттерді іздей алады. Сонымен қатар агент өзінің өмірлік кезеңін бақылауға қабілетті, жеке алғанда басқа агенттермен сөйлесе алады.

Агенттер асинхронды хабарламалар алмасу жолымен сөйлеседі. Коммуникация моделі бөлінген және әлсіз байланысқан коммуникациялар, яғни бір бірі туралы білмейтін гетерогенді мәндер арасындағы коммуникациялар үшін жаппай қолданылады. Агент коммуникация іске асу үшін алушыға хабарлама жібереді. Агенттің аты идентификатор қызметін атқарады (сілтемені алушы – объектіге көрсетудің қажеттігі жоқ) және сөйлесетін агенттер арасында уақытша тәуелділік болмайды. Алушы мен жіберушінің бір уақытта қосылуы талап етілмейді. Алушының болмауы да мүмкін және жіберушіге белгісіз болуы да мүмкін. Жіберуші соңғы мекен – жайды көрсету үшін жіберушінің ерекшелігін (мысалы, футболға қызыққан барлық агенттер) анықтай алады.

Коммуникацияның осындай түріне қарамастан, коммуникацияда хабарламаларды жіберу қауіпсіздігі сақталады. Қауіпсіз арналар талап етілетін

қосымшалар үшін JADE платформасы агенттер белгілеген құқықтарды тексеру және аутентификацияға арналған тиісті механизмдерді ұсынады. Демек, талап етілгенде қосымша хабарлама жіберушінің идентификаторының шынайылығын тексере алады және рұқсат етілмеген әрекеттердің орындалуының алдын алады (мысалы, агентке агент - бостан хабарлама алуға болады да оған хабарлама жіберуге болмайды). Агенттер алмасатын барлық хабарламалар тарнспорттық деңгейге қажетті ақпараттарынан тұратын сауыттың ішінде беріледі. Бұл осылардан басқа хабарлама мазмұнын оның сауытынан бөлек шифрлауға мүмкіндік береді.[49]

Хабарлама құрылымы FIPA спецификациясымен анықталған ACL тілінің көмегімен қойылады және көптеген паралельді әңгімелесулер мен күрделі өзара әрекеттерді мақсат тұтқан жауап күтілетін ауыспалы, контексті, алушыны, хабарламаны, уақытты анықтайтын өрістерден тұрады.

Әрі қарай күрделі коммуникацияларды тоқтатпау үшін JADE бағдарламалық қамтамасыз етуі келіссөздерді енгізу, аукцион және тапсырмаларды жіберу сияқты белгілі бір тапсырмаларды орындауға арналған өзара әрекетесудің паттерндерінің тірек (қаңқасының) жиынтығын ұсынады. Программистер осы қаңқаны қолдана отырып, жалпы алғанда қосымша логикасымен тығыз байланыста болмайтын барлық аспектілерден, тапсырмаларды синхрондау, таймауттар, қателік жағдайлары сияқты мәселелерден құтыла алады. JADE агенттік платформасы хабарлама мазмұнын өңдеу мен құру механизмдерін іске асыруды жеңілдету үшін контентті басқаруға сәйкес (мысалы Java объектілері) форматтар XML және RDF қоса алғанда контент алмасуға сәйкес келетін форматта тікелей автоматты және кері конверсиялау үшін қолдауды ұсынады. Бұл қолдау программистерге онтологияны графикалық түрде жасауға мүмкіндік бере отырып, антологияны жасаудың кейбір құралдарында құрылған, мысалы Protégé. JADE платформасы жағдайда қорытынды жүйесінің базалық шағын моторына қатысты тұнық емес болып табылады, егер спецификалық қосымшалар үшін қорытынды нәтижелері талап етілсе, ол программистерге ұнамды қорытынды жүйесін қайта қолдануға мүмкіндік береді. Ол JESS және Prolog модульдерімен интегралданады.

Масштаптылықты ұлғайту немесе шектелген ресурстары бар ортаның шектеулерін қанағаттандыру үшін JADE БҚ Java –ның бір ағымында көптеген паралельді тапсырмаларды орындау мүмкіндігін ұсынады. Коммуникация сияқты бірнеше қарапайым тапсырмалар соңғы автоматтар тәрізді ұсынылған өте күрделі тапсырмаларды құру үшін қосыла алады.

JADE бағдарламасы J2SE және Personal Java ортасында код пен жағдайдың мобильділігін ұстанады. Яғни агент хоста жұмысты тоқтатып, басқа жойылған хосқа көше алады (кодты көшірмелеуге және агентті осы хоста алдын ала орнату қажетілігі жоқ), сосын орындауды тоқтатқан жерден бастап қайта бастауға болады. Бұл атқарымдық қызмет мысалы, қосымшаға ешқандай әсер етпей жүктелген машиналарға агенттердің орнын ауыстыру жолымен орындау режімінде есептік жүктелімдерді бөлуге мүмкіндік береді.

Платформа аттардың қызметі (әрбір агенттің өз ерекше атын тексеретін қызмет) мен бірнеше хост бойынша бөлінген «сары бет» қызметінен тұрады. Агенттік қызметтердің ұйымдасқан салаларын анықтау үшін федерация бағандарын құруға болады. Басқа ең маңызды қызмет ретке келтіру кезеңін және қосымшаның өмірлік кезеңін басқару/бақылауды ұстанатын графикалық құралдардың мол жиынтығының қолжетімді болуынан тұрады. Осы құралдар арқылы алыста тұрып агенттерді басқаруға (толық ашылған немесе қараусыз қалған болса да): агенттердің келіссөздерін етуге, агенттер алмасатын хабарламаларды қарауға, тапсырмалар мониторингісімен айналысуға, агенттің өмірлік кезеңін бақылауға мүмкіндік туды.

Функционалдық бөлшектерінің сипаттамасы, жеке алғанда тіпті мобильді терминалдар, тапсырмалар, коммуникация және жаңа тораптар арқылы алыстан активтендіру (жылдамдату) мүмкіндігі p2p түрі сияқты проактивті, интеллектуалды, бөлінген қосымшаларды орындау және өңдеу үшін сәйкес келетін JADE платформасын жасайды.

Кесте 2-JADE агенттік платформасының негізгі сипаттамасы

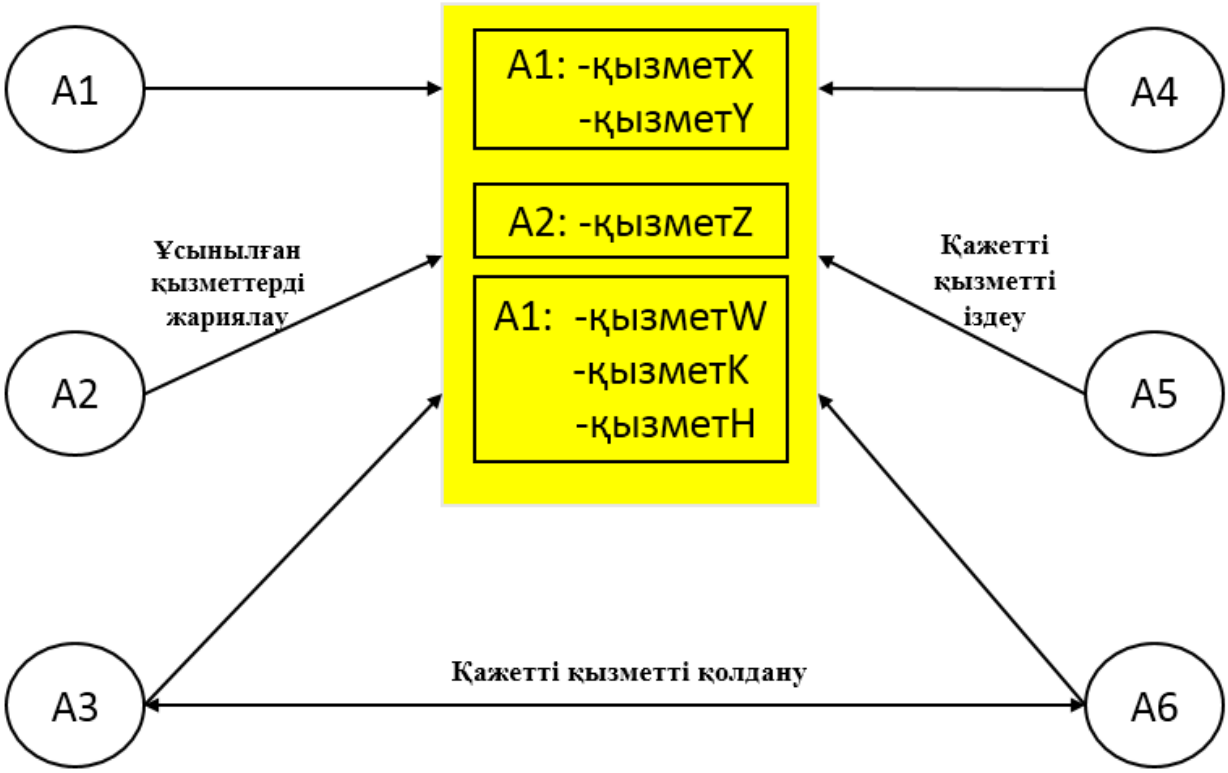
Атауы	JADE – Java Agent Development Framework
Әзірлеуші	TILAB
Сайт	http://jade.tilab.com/
Тілі	Java: J2EE, J2SE, J2ME CLDC/MIDP1.0 platforms
Техникалық/функционалдық сипаттамасы	- «peer-to-peer» коммуникациясы бар бөлінген, көпжақты қосымша;; - FIPA стандартына сәйкестік; – Агенттің өмірлік кезеңін басқару; - онлайн режімінде федерация бағандарын құру мүмкіндігімен «ақ бет» және «сары бет» қызметі; – Мониторинг, басқару, ретке келтіру кезеңдерін ұстанатын графикалық құралдар; – Агент кодының көшуін және қараусыз қалған жағдайын қолдау; – Өзара әрекеттестіктің күрделі хаттамаларын қолдау (мысалы, contract-net); – XML және RDF форматында хабарлама контентін басқару және құруды қолдау; – JSP жағынан бірлесуді қолдау; – Қосымша деңгейінде қауіпсіздікті қолдау (осы уақытта – тек J2SE платформасы

	үшін); – Орындау режімінде көлік хаттамаларын таңдау мүмкіндігі; – Қолжетімді көліктік хаттамалар: JAVA-RMI, JICP, HTTP және IIOP.
Желілік қоршау	Bluetooth, GPRS, W-LAN, Internet базаларында сынау жүргізілді.
Терминалдар	Java MIDP 1.0, немесе Personal Java, не J2SE ортаны қолдаушы барлық терминалдар.

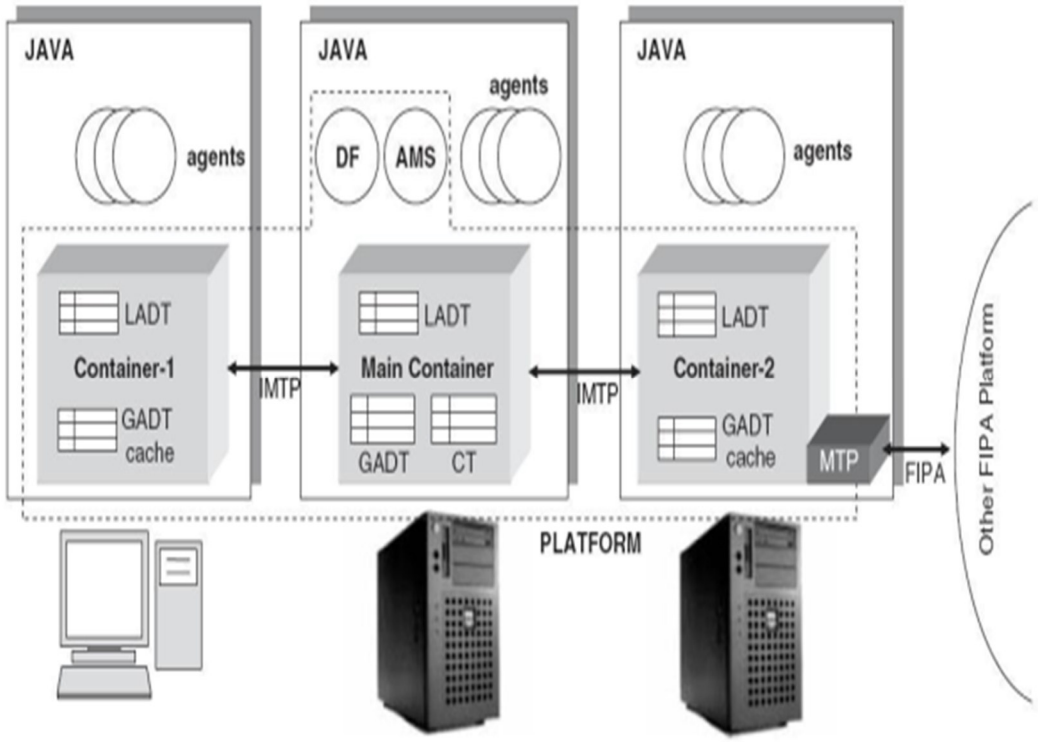
Платформа ішінде агенттердің өзара әрекеттесуіне қажетті сервистерді қамтамасыз ететін агенттер жүйесінің жиынтығы бар:

* *AMS (Agent Management System)* – осы агент барлық платформаны басқарады. Бұл агент платформа агенттерінің өзара әрекеттесуі үшін қажет, ол платформаның ақ бет сервисіне агенттердің кіруін қамтамасыз етеді және олардың өмірлік кезеңдерін басқарады. AMS агентті құру және жою, контейнерлерді жою, платформа жұмысының аяқталуы сияқты платформаның түрлі операцияларын қамтамасыз етеді. Әрбір агент AID дербес идентификатор алу үшін AMS-те тіркелуі керек. AMS агентінің іске қосылуы платформаның бас контейнерінің ішінде жүргізіледі, бірақ кез келген контейнерде болуы мүмкін. Платформаның жұмыс істеуімен байланысты әрекеттерді жасайтын агент алдымен AMS агентінен құптауын сұрауы қажет.[50]

* *DF (Directory Facilitator*) – бұл агент сервистердің тіркелуін және сары беттерде сервис бойынша агент іздеуді қамтамасыз етеді. Платформа агенттері қажетті сервистітіркеу туралы ақпарат алу үшін DF-агенттінде жазыла алады.



Сурет 7 - Сары бет сервисі



Сурет 8- Негізгі сәулеттік элементтер арасындағы қатынас

Жоғарыдағы суретте JADE платформасының сәулеттік элементтері көрсетілген. JADE платформасы желіде таратылған контейнерлер жүйесінен тұрады [51].

Әдетте әрбір хоста бір контейнерден болады (бірақ ретке келтіру мақсатында олар бірнеше болуы мүмкін). Агенттер жоғарыда сипатталған жүйелік сервистер қамтамасыз етілетін контейнерлердің ішінде болады. Жүйеде платформаның бастапқы жүктелу орнын көрсететін тек бір бас контейнер болады. Бас контейнер бірінші құрылады, ал кейінірек жасалған контейнерлер бас контейнерде тіркелуі қажет.

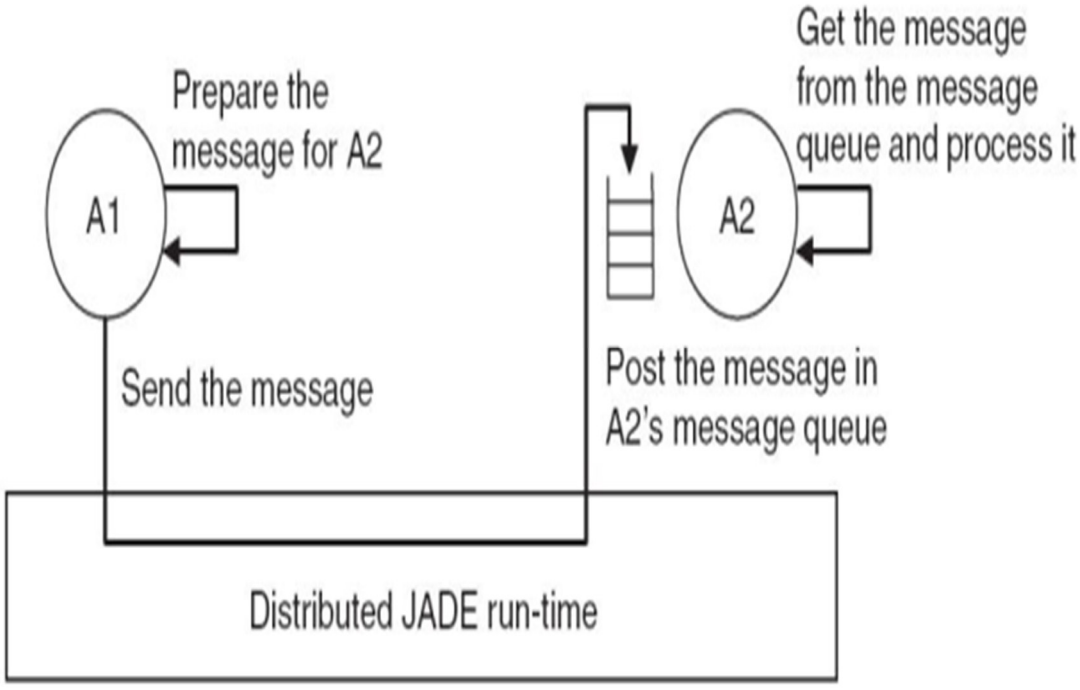
Бас контейнер контейнерлер сызбасынан (CT-container table) тұрады. Бұл сызбада платформа құрамына кіретін барлық контейнерлердің объектілері мен транспорттық мекен – жайына жіберілетін сілтемелер сақталады.

Әрбір контейнердің ішінде:

- *GADT (global agent descriptor table)* – бұл платформа агенттерінің ағымдағы мәртебесі мен орны сақталатын регистірі.
- *LADT (local agent descriptor table)* – GADT сияқты, бірақ контейнер агенттерінің регистірі болады.

Хабарламалармен алмасу сервисі

Хабарламалармен алмасу сервисі JADE платформасы сәулетінің негізі болатын бөлімдерінің бірі. Сервис хабарламаларды асинхронды түрде жіберуге негізделген. Әрбір агенттің өз «пошта жәшігі», яғни агентке бағытталған барлық хабарламалар орналасатын кіруші хабарламалар кезегі бар. Агент хабарламалар кіруші хабарламалар кезегіне орналасқан кезде ол туралы құлақтанады (хабарландырылады).



Сурет 9-Хабарламаларды асинхронды алмасу диаграммасы

МАЖ - бұл жасанды интеллект идеялары мен әдістерімен, заманауи жергілікті және ғаламдық компьютерлік желілермен, таратылған мәліметтер базасымен және таратылған есептеу, аппараттық және бағдарламалық құралдармен қамтамасыз етілген ғылыми-техникалық жетістіктер мен артықшылықтарды бірлесіп пайдалануға бағытталған ақпараттық технологиялардың салыстырмалы жаңа парадигмасы.

МАС дамуы көптеген факторларға байланысты. Біріншіден, қазіргі заманғы жүйелер мен ұйымдарының күрделелілік деңгейінің өсуі үлкен көлемдегі ағындардың пайда болуына және сол ағындарды орталыққа жіберу және шешім қабылдау үшін уақыттың көп кетуі орталықтан басқару жүйесінің тиімсіздігіне алып келеді [52]. Компьютерлік жүйелердің өзі де барған сайын күрделене түсуде және әртүрлі функционалды сипаттамалары бар және бір-бірінен қашықтағы әр түрлі мамандармен өзара әрекеттесетін әр түрлі сипаттағы бірнеше ішкі жүйелерді қамтиды. Сонымен қатар, күрделіліктің жоғарылауымен жүйелердің сенімділігі төмендейді және олардың мақсатты функциясын тұжырымдау қиынға соғады.

Екіншіден, шешілетін есептеулер және жасалып жатқан жүйелер гетерогенді және үлестірілген: а) кеңістікте; б) функционалды тұрғыдан, себебі ешкім қазіргі заманғы күрделі жүйені жалғыз өзі жасай алмайды.

Үшіншіден, МАЖ - бұл ашық жүйе. Сонымен қатар өзінің құрылым мен праметрлерін дамыту арқылы ол өзгеріп отыратын ортаға бейімделе алатын мүмкіндігі бар. Осылайша, МАЖ күрделі деректер мен білімдерді үлестіру арқылы жүзеге асыруға, ақпараттық және интеллектуалды қолдау деңгейінің едәуір өсуін қамтамасыз етуге, иерархияның әртүрлі деңгейлерінде басқару мен шешім қабылдау процедураларының тиімділігін арттыру мақсатында пәндік аймақ туралы білімді өңдеуді ұйымдастыруға көмектеседі.

[53] -де авторлар күрделі жүйелердің келесі негізгі элементтерін анықтайды:

- шешім қабылдау орталықтандырылмаған, үлестірілген. Күрделі жүйелер бір-бірімен сөйлесе алатын агенттер деп аталатын байланысқан автономды элементтерден тұрады.

- Агенттердің автономиясы толық емес. Кез-келген күрделі жүйеде және / немесе секілді агенттер ұстанатын принциптер, ережелер, заңдар бар.

- Күрделі жүйенің ғаламдық мінез-құлқы құрамдас агенттердің өзара әрекеттесуінен туындайды. Алайда агенттердің шешім қабылдау еркіндігі шектелгендіктен, күрделі жүйелер мінез-құлық үлгілерін көрсетеді. Мұнда құраструшылардың таңдау мүмкіндігі бар. Белгісіздік дәрежесін жүйені белгіленген кең заңдылықтарға сүйенуге мәжбүр етіп реттеуге болады. Толық болжамдылыққа бағытталмауы керек, өйткені ол қажет болған жағдайда жүйенің өзін-өзі ұйымдастырып, бейімделуіне жол бермейді.

- Күрделі жүйелер сызықтық емес: ең кіші сыртқы әсерлер жүйенің іс-әрекетіне үлкен өзгеріс тудыруы мүмкін, құбылыс көбелектің әсері немесе өзін-

өзі үдету деп аталады. Сондай-ақ, күрделі жүйелер автокаталитикалық мінез-құлықты, яғни кез-келген сыртқы әсерсіз жаңа құрылымдарды құра алады

- үлестірілген шешім қабылдау, шешім қабылдау элементтерінің (агенттерінің) өзара байланысын білдіреді. Агенттер арасындағы байланыс мықты, бөлінбейтін немесе мүлдем жоқ болуы мүмкін. Агенттер арасындағы байланыс түрі жүйенің бұзылған кездегі жауабын анықтайды. Дизайнерлер тізбек өзгерісінен болған толқындарға кететін уақытты азайту үшін агенттер арасындағы белгілі бір байланыстарды әлсіретуі мүмкін.

- Автономия дегеніміз интеллектілікті білдіреді. Интеллект белгісіздікті шешу үшін білімді қолдану мүмкіндігін білдіреді.

Берілген күрделіліктің анықтамасы мен қазіргі зерттеудің шешімді жобалау саласы арасында бірнеше сәйкестік бар.

Біріншіден, түйіндер есептеу техникасына өз еркімен қосылатындықтан, шешім қабылдау орталықтандырылмаған, яғни желідегі құрылғыларды басқаратын орталық инфрақұрылым бөлімшесі жоқ. Екінші жағынан, олар құрылғы қабылдағаннан кейін тапсырманы орындау міндетіне байланысты толық автономды бола алмайды. Осылайша, құрылғылардың алдын-ала анықталған себептерсіз тапсырмалардан бас тартуына жол бермейтін жергілікті басқару механизмі бар.

Екіншіден тапсырманы орындау құрылғылар сол тапсырманы шешу үшін қалай ұйымдасатынна байланысты, сондықтан жаһандық жүйенің іс-әрекеті жеке компоненттердің іс-әрекетіне тәуелді. Жүйені құрастыру жағынан қарайтын болсақ бұл деген соөз құрастырушы есептеуді орындаудың жалпы шаблонын құрады, ал жүйе соған сәйкес өзін-өзі ұйымдастырады [54].

Соңында, жеке құрылғылыраның мобильділігіне байланысты, олар ұйымға кез-келген уақытта қосылуы мүмкін және кез-келген уақытта ұйымнан шығып ктуі мүмкін. Бұл агенттер арасындағы байланысты әлсіретеді және жүйе жұмысының тоқтауына әсер етпейтіндей үлсетірілген шешім қабылдау механизімін құруды талап етеді. Ұсынылған аргументтер қоршаған ортаның күрделі екенін көрсетеді және нақты шешімдерді қажет етеді. Сондықтан мәселені шешу күрделі жүйелердің мінез-құлқын білдіретін әдістерді қолдану арқылы шешілу керек.

2.3 Жасанды интеллект және күрделі іс-әрекет

Біз күрделіліктің басты анықтамасы ретінде [55] қолданамыз, соған сәйкес бірнеше ерекшеліктерін атап өтуге болады:

- тәуелсіздік. тәуелсіздік дегеніміз жүйенің автономиясын ғана емес, сонымен бірге гетерогенділікті, және оның жеке компоненттерінің бір-бірінен тәуелсіздігін білдіреді;

- автономия. Жүйе компоненттерінде орталықтандырылған немесе жергілікті басқару механизмі жоқ. Керісінше, олардың іс-әрекеті шектеулер мен ережелер жиынтығымен басқарылады.

- пайда болу. Пайда болу - бұл жүйенің күтпеген әрекеті оның компоненттерінің бақыланбайтын өзара әрекеттесуі нәтижесінде пайда болатын құбылыс. Іс жүзінде сыртқы көрініс сонымен қатар оның бірде-бір компонентінде шешімнің берілген алгоритмі болмаған жағдайда жүйелік нәтижеге жетуді білдіреді. Басқаша айтқанда, әр компонент жалғыз өзі шеше алмайтын мәселелерді шешу үшін бірлесіп жұмыс жасайтын компоненттер;

- Баланстың болмауы. Күрделі жүйелер әдетте ешқашан тепе-теңдікте болмайды, керісінше хаос жағдайына жақын болады;

- сызықтық емес. Күрделі жүйелердің әрекеті әдетте сызықтық емес;

- өзін-өзі ұйымдастыру. Стандартты емес жағдайда күрделі жүйені қайта құру керек, осылайша жүйе дамиды;

- өзара даму. Бұрын жүйенің сыртқы ортаның өзгеруіне байланысты дамитыны атап өтілген болатын. Алайда жүйені өзгерту қоршаған ортаны модификацияланған жүйені қолдануға мәжбүр етеді. Сонымен, жүйе қоршаған ортаға өзгерістер әкеледі, ал жүйе ортаны өзгертеді;

Бір қарағанда компьютерлік жүйелерді жобалау кезінде неге осы сипаттамаларды ескеру қажет екендігі түсініксіз. Алайда, тәжірибе көрсеткендей, көбінесе модельденетін экономикалық, биологиялық және әлеуметтік процестер күрделі, сондықтан табысты жұмыс істеу үшін ең қуатты ақпараттық жүйелер де үнемі қайта қарауды және логикалық компоненттерді мезгіл-мезгіл жаңартып отыруды қажет етеді.

2.4. Күрделі компьютерлік жүйелерді жобалау

Компьютерлі жүйелерде -мультиагентті тәсіл- секілді жүйе күрделі жүйелерді құру және жобалауда белсенді түрде дамып келеді. Ол информатика, әлеуметтану және психология қиылысында құрылған агент тұжырымдамасына негізделген. Оның құрылу себебі адамның ниет, сенім, білім, парасаттылық және старгиалық ойлау секілді есептеу қабілетін беру мүмкіндігіне арналған.

Осы бағыттағы зерттеу топтарының шашыраңқылығына байланысты агенттің бірыңғай анықтамасы жоқ. Алайда көптеген ғалымдар келесі анықтаманың негізгі мәнді қамтитындығымен келіседі: «агент дегеніміз - белгілі бір ортада орналасқан және мақсатқа жету үшін икемді және тәуелсіз әрекеттерді орындауға қабілетті, жинақталған есептеуіш құрылым». [56] Анықтамадан әр агент қолданушы белгілеген мақсатқа ие болуы және оған жету үшін «икемді» әрекеттер тізбегін орындауы керек екендігі түсінікті. Яғни, тәуелсіз болу және өз әрекеттерін олар жұмыс істейтін жағдайға сәйкес реттеу. Бұл агент интерфейстер арқылы деректерді қабылдау және қоршаған ортаға барлық қол жетімді құралдармен жауап беру арқылы қоршаған ортамен өзара әрекеттесу мүмкіндігіне ие болуы керек дегенді білдіреді. Мысалы, агент басқаратын робот камераны пайдалану арқылы қоршаған ортаны қабылдай алады және дененің механикалық бөліктері, мысалы, аяқтар және қол арқылы жауап бере алады.

Агенттің әрекеті реактивті және белсенді болып жіктеледі. [57] Агенттің реактивтілігі - бұл оның әрекеті қоршаған ортаның өзгеруіне жауап болатын мінез-құлық. Проактивтілік - бұл агент, болашақта қоршаған ортаға ұсынылатын өзгерістерден пайда табу үшін дәйекті әрекеттер жасайтын мінез-құлық.

Осы классификация негізінде интеллектуалды агентті басқа есептеу объектілерінен ажырату қиын, өйткені нақты уақыттағы жүйелер белсенділік қасиеттеріне ие және барлық белгілі бағдарламалар сол немесе басқа жолмен реактивті. Алайда, ақылды агент іс-әрекеттің мәнерінен және мағыналық мазмұнынан көрінетін бірдей әлеуметтік мінез-құлыққа ие.

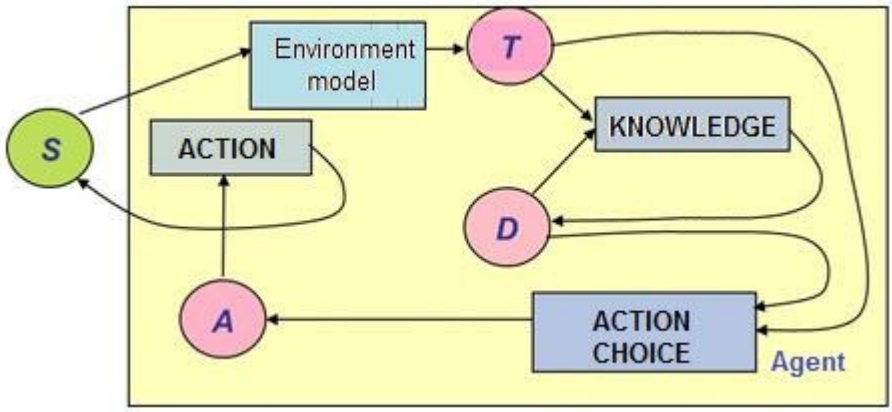
- Агенттің әлеуметтік қызметіне [58] жатады: - оқыту. Сыртқы ортаның өзгеру заңдылықтарын талдау және ол туралы білімді жинақтау негізінде агент жоспарламаған және бастамаған әрекеттерді орындау мүмкіндігі;

- іс-әрекеттің интерактивті принципі. Өзін-өзі ұйымдастыру қағидатын толығымен іске асыру үшін агенттерге жүйенің басқа қатысушылары мен қолданушыларының тікелей бұйрықтарына бағынудың қажеті жоқ. Керісінше, жеке тапсырмалар мен іс-әрекеттер тараптар мен жүйені пайдаланушылар арасында ерікті түрде келіссөздер жүргізілуі керек, ал әрекеттерді орындау тенденциясы агентпен жұмыс жасайтын өз мақсаттарымен, білімімен және ережелерімен белгіленуі керек;

- жауапкершілікті тапсыру. Бұрын айтылғандардан, агент қолданушыларға немесе бұрын енгізілген алгоритмдерге қарамастан шешім қабылдауы керек екендігі түсінікті. Алайда, бұл агенттердің шешімдеріне тікелей әсер етпейтіндігіне байланысты жүйеге жауапкершілікті пайдаланушыға беруді талап етеді. Енді агенттердің бостандықтарын шектейтін және функциялардың дұрыс орындалуын қадағалайтын тетіктер қажет (жүйеге сенім), сонымен қатар талаптарды (пайдаланушылар), олардың сипаттамаларын (Business Analyst) және іске асыруды (бағдарламашы) ресімдеуге қойылатын талаптардың жоғарылауы қажет.

Жалпы бұл құбылыстарды адамдарда салыстырмалы түрде нашар білуіне байланысты, қазіргі кезеңде оларды есептеу жүйелері шеңберінде толық жүзеге асыру мүмкін емес. Осыған байланысты әлеуметтік сипаттамалардың даму деңгейінде агенттердің градациясы бар.

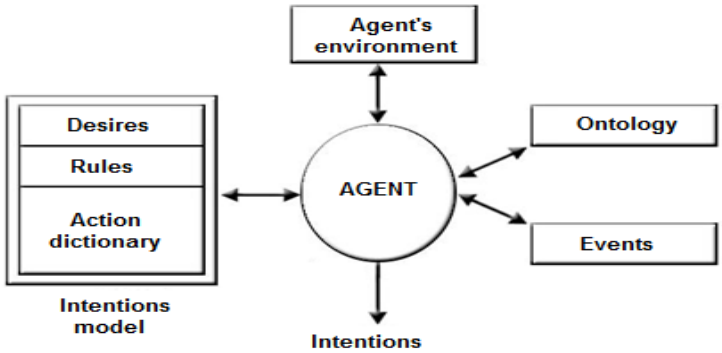
Өнеркәсіптік ауқымда ең «примитивті», бірақ оңай жүзеге асырылатын «реактивті агент» [59]. Осы типтегі агенттердің мінез-құлқы белгілі бір сигналдарды алған кезде бағдарламаланған іс-әрекеттерді тікелей орындаумен сипатталады. Брукстың еңбектерінде айтылған негізгі идея интеллектуалды мінез-құлыққа оқиғаларды екіұшты түсіндіру ережелерін сипаттамай және баламалы мінез-құлық бағдарламалау алгоритмдерінсіз қол жеткізуге болады дейді. Ол бірінен соң бірі ретімен орындалатын тапсырмалар жиынтығына негізделген. Ақырғы автоматтарды қолданатын жүйелердің салыстырмалы түрде қарапайым сипаттамасын, дамудың құралдарының қол жетімділігін, соның ішінде өнеркәсіптік, тез шешім қабылдау агенттерін осы агенттер класының артықшылықтары қатарынан анықтауға болады.



Сурет 10 – Реактивті агент

Логикалық агенттер анағұрлым күрделі анализге ие [60]. Агенттердің бұл түрі логикалық бағдарламалау тілдерінде жүзеге асырылады, ойлау механизмі бар және әлемді символдық белгілермен көреді. Іс жүзінде бұл тәсіл модельдік және уақыттық логиканы кейбір белгілер доменімен кеңейту негізінде жүзеге асырылады. Агенттің бұл түрі практикалық қолдануды таппады, сондықтан біз оларды қарастырмаймыз. Біріншіден, белгілер жиынтығының домендік сипаттамасы ауыр процесс. Екіншіден, логикалық агенттердің пайымдауы көп уақытты қажет етеді, олар жауап берген кезде оған деген қажеттілік жойылады.

Сонымен, BDI (сенім, тілек, ниет) агенттері сенім, қалаулар мен ниеттердің қасиеттерін философиялық тұрғыдан сипаттайтын модальды логикаға негізделген [61]. Бұл контекстке деген сенім агент үшін қол жетімді, бірақ толық емес немесе қате болуы мүмкін әлем туралы деректерді білдіреді. Агенттің қалауында әзірлеуші белгілеген мақсаттар болады, ал ниет ниетке жету үшін қажетті қадамдарды білдіреді. Агенттің ниеттерін үнемі қалыптастыруға және түрлендіруге мүмкіндік беретін ішкі механизм бар. Сол механизм ниет жоспарын да жасайды, оны жоспар деп те атайды.



Сурет 11 – BDI агент

BDI агентінің мінез-құлық схемасы келесідей:

- Агенттің белгілі бір қалауы бар және осы қалауды аяқтай алатын ниетті қалыптастырады;
- Ниетке сәйкес әрекет ету міндеттеме қажет;
- Бұл ниеттердің аяқталуына әкелетін кіші мақсаттар ағашын жасайды және оңтайлы кіші ағашты таңдайды;
- Мақсаттар ағашынан кіші мақсатты таңдайды және оған жетуге жетелейтін әрекетті орындайды.
- Агент мақсатқа жеткенде немесе оның қол жетімсіздігіне сенімді болған кезде, ол өзінің мінез-құлқын сол стандартты циклға сәйкес қайталайды. Біздің көзқарасымыз бойынша, агенттердің бұл түрі өзін-өзі ұйымдастыруға жету үшін ең перспективалы болып табылады, туындайтын және тәуелсіздік.

Гибридті агенттер де бар. Бірақ қазіргі уақытта оларды тәуелсіз класс ретінде ақтайтын әдебиет жоқ.

Агенттің өзі жеке тривиальды тапсырманы шеше алады, бірақ ауқымды мәселелерді шешу үшін бір ғана агент күші жеткіліксіз. Басқаша айтқанда, бірнеше агенттердің ынтымақтастығы болып табылатын мультиагенттік жүйені (МАЖ) құру қажет.

Городецкий [62] МАС-қа келесідей анықтама берді: МАС дегеніміз - кез келген жалғыз шешушінің күшінен тыс мәселелерді шешуге қабілетті белгілі бір есептердің (агенттердің) еркін байланысқан шешушілерінің желісі. «Ол сонымен қатар келесі қасиеттерді анықтайды: МАЖ:

-Әр агент қоршаған орта туралы ақпаратты толық білмейді;

- агенттерді толық басқару шектеулі немесе жоқ;

-Агенттер қолданатын деректер орталықсыздандырылған;

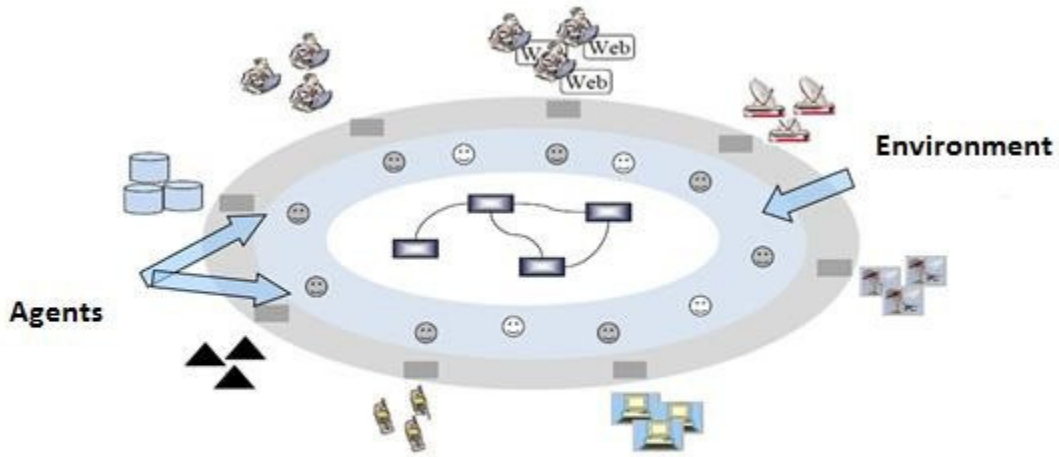
-Агенттер асинхронды түрде жұмыс істейді;

Агенттер өз әрекеттерін жоғары деңгейдегі тілде хабарламалар алмасу арқылы үйлестіреді.

Мониторингтің ғаламдық механизмінің болмауы, компоненттер арасындағы өзара байланыс және компоненттердің тәуелсіздігі туралы интерактивті ұйымдастыру қағидасының жоқтығы туралы сұрақтар бұрын қарастырылған.

Әрбір жүйелік компоненттің (агенттің) қоршаған ортаның жеке моделіне ие болуы (1-тармақ) және мәліметтерге қол жеткізудің әр түрлі деңгейлері оларды жеке агент қабылдаған мән-жайларды адекватты бағалау мүмкіндігін жоққа шығарып, ұжымдық шешім қабылдауға мәжбүр етеді. Сонымен қатар, ұзақ мерзімді перспективада бұл агентке ұжымдық шешім қабылдау сапасы негізінде, әсіресе эволюция тұжырымдамасына жақын компьютерлік жүйелердің принципиалды жаңа сапасы болып табылатын жұмыс алгоритмдерінің динамикалық өзгеруі негізінде өзін-өзі тәрбиелеуге мүмкіндік береді.

Алайда, MAS-тің ойдағыдай жұмыс істеуі үшін агенттер қандай мақсаттарды көздесе де, қандай платформада жүзеге асырылса да ынтымақтастыққа мүмкіндік беретін орта қажет (1.1.2.3-сурет).



Сурет 12 – агенттер, мультиагентті жүйе және ОЖ.

3 GRID ЖҮЙЕСІН ҰЙЫМДАСТЫРУДА МУЛЬТИАГЕНТТІ ЖҮЙЕНІҢ ПРИНЦИПТЕРІН ҚОЛДАНУ.

Қазіргі кезде ғылым мен техниканың күрделі мәселелерін шешкен кезде үлестірілген есептеу жүйелері кеңінен қолданылады. Оларға көп кластерлі есептеу және грид жүйелері жатады.

Орындаушыларды таңдау кезінде тапсырманың ерекшеліктерін ескеру, орындаушылардың бір-бірімен тікелей өзара әрекеттесуіне мүмкіндік беру, есептеу торабы параметрлерінің өзгеруіне жылдам жауап беру арқылы қазіргі грид-дің көрсетілген кемшіліктерін жою үшін грид ұйымдастыру процесін икемді және адаптивті ету арқылы шешуге болады. Ол үшін брокерлік функциялардың бір бөлігін қызмет деңгейінен орындаушы деңгейіне ауыстыру ұсынылады [63]. Сонда орындаушылар деңгейіне есептеу желісі мен тапсырмалар туралы ақпарат қолжетімді болады, ол деректерді есептеу тораптары арасында тікелей алмастыруға мүмкіндік береді және дерек алмасу уақытының қысқаруына алып келеді. Сонымен қатар, әрбір қарапайым брокерлерді есептеу тораптарында орналастыру оларға осы есептеу тораптары параметрлеріндегі өзгерістерді жылдам қадағалауға мүмкіндік береді, бұл ерікті есептеу ресурстарына негізделген грид құруға мүмкіндік береді. Осылайша, параметрлері динамикалық өзгермелі есептеу тораптары негізінде құрылған біртұтас есептерді тиімді шешу үшін брокерлік функцияларды тікелей грид-тегі есептеу тораптарына ауыстыруға мүмкіндік беретін орталықтандырылмаған мультиагенттік ұйымның принциптерін әзірлеу ұсынылады [64]. Басқару жүйесін құрудағы көп агенттік тәсіл бір-біріне тәуелді емес көптеген субъектілердің - өзара әрекеттесу процесінде белгілі бір келісімдерге келетін агенттердің бір уақытта жұмыс жасауын болжайды, бұл өз кезегінде жүйеде пайда болған эмергентті мінез-құлқын анықтайды. Қазіргі кезде мултиагентті өзара әрекеттесу принциптері көптеген әр түрлі мәселелерді шешуде сәтті қолданылуда [65] [66].

Қазіргі уақытта әдебиетте агент ұғымына қатысты көптеген әр түрлі анықтамалар бар. Әдебиетте агентті жүйенің жұмыс процесінде өзінің атқарушы есептеу түйінінің мүдделерін қорғайтын белсенді тұлға ретінде түсінуді ұсынады [67]. Сонымен қатар, мултиагентті жүйеде бір-бірімен өзара әрекеттесе алатын тәуелсіз агенттер орналасқан және олардың әрқайсысы өз иесінің қызығушылығын барынша жүзеге асыруға тырысады. Жүйеде тәуелсіз агенттердің ұжымдық өзара әрекеттесуі жүйенің ғаламдық мақсатына қол жеткізуге мүмкіндік береді. Агенттердің әрқайсысы сырттан басқарылмайды, бірақ олардың ұжымы жалпы тапсырманы орындайды, бұл жүйені орталықтандырылмаған етеді. Мултиагентті тәсілді орталықтандырылмаған грид жүйесін ұйымдастыруда қолдануға болады. Грид-те жүйенің негізгі өзара әрекеттесуші элементтері ретінде агенттерді клиентінің де, брокердің де функцияларын орындайтын етіп жасау қарастырылады. Агенттер физикалық тұрғыдан әр түйінде орналасады және олар процесс барысында мултиагентті грид жүйесінің мүддесін қорғайды. Бұл жағдайда жүйе әрбір мәселені шешу

үшін агенттерден құралған виртуалды ұйым, яғни агенттер ұжымы құрылуы керек. Бұл жұмыста болашақта ұйымды бір нақты мәселені шешу үшін жиналған агенттер ұжымы ретінде түсіну ұсынылған [68]. Қауымдастық құрудың басты мақсаты – өз есебін (қолданушы грид) белгілеген уақытқа дейін шешу, сонымен қатар ұйым құрамы тапсырманы орындау барысында өзгеруі мүмкін. Есептеу тораптарының істен шығуы, өнімділігінің төмендеуі, деректерді беру жылдамдығының төмендеуі және есептеу желісінің басқада өзгерістері қолданушы тапсырмасын шешудегі уақыттың ұлғаюына алып келеді және ұйымның өзгерсіне алып келеді.

Ұйым құру процесінде есептің бөлігі осы есепті шешу үшін құрылған қауымдастық арасында үлестірілуі керек. Бұл жағдайда есептеу тораптарының әркелкілігіне байланысты, есепті шешу уақыты кең шектерде өзгеруі мүмкін. Сондықтан, есепті шешу уақытын қысқарту үшін, үйымда тапсырмаларды орналастыруды оңтайландырған жөн. Сонымен қатар, жүктемені біркелкі бөлу үшін оңтайландыру процесі барлық қауымдастық агенттері арасында параллельденуі керек[69].

Мұндай жүйені құру схемасының қазіргі жүйелерден айырмашылығы - грид брокерінің мультиагентті ұйымы есептеу тораптары- «фрилансерлердің» есептеу ресурстарын тиімді пайдалануға мүмкіндік береді, және агенттер өздерінің торап параметрлерін өз бетінше бақылай алады және егер олар өзгерген жағдайда, ұйымда тапсырманы қайта бөледі немесе ұйымның құрамын қайта өзгерте алады. Тораптарды осылай ұйымдастыруда, грид, деректерді қызметтік деңгей тораптарының көмегінсіз тікелей жібере алуы үшін, тапсырма туралы және есептеу желісі туралы ақпаратты жеткілікті мөлшерде біледі [68].

Алайда, осындай мултиагентті ұйымдастыру жүйелері жүйе мен қолданушылар арасындағы байланысты күрделендіреді, себебі грид орталықсыздандырылғаннан кейін, классикалық грид-дегідей пайдаланушы жүйемен орталық брокер арқылы өзара әрекеттесе алмайды, бірақ қандай-да бір жолмен өзінің тапсырмасын бірнеше агенттерге беруі керек.

3.1 Қолданушы тапсырмасын орындау үшін мультиагентті одақ құру

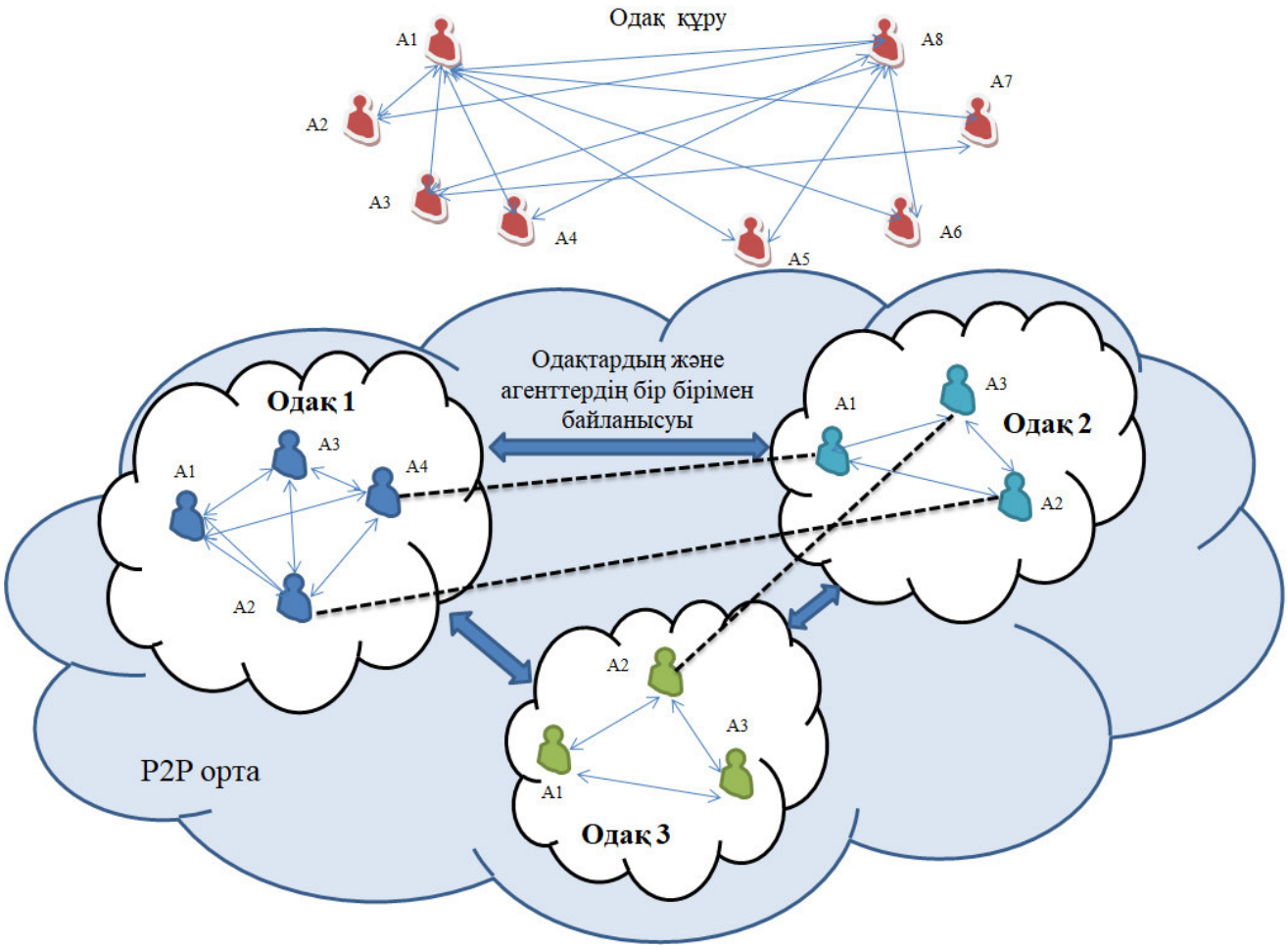
МАЖ-де жеке агенттер қауымдастықтарға - одақтарғаға біріге алады [ ]. Қазіргі кезде коалициялар көп агенттік жүйелердегі үлкен теориялық тақырып болып табылады. Коалиция теориясының дамуына үлкен үлес М.Дж. Вулдридж [], Дж.Видал [ ], Ю.Шохам [132], К.Лейтон-Браун [ ], К.Бинмор [ ]. Осы жұмыс шеңберінде одақ құру тетігін агент бірінші кезекте басқа агент ресурстарын онымен келісілген жоспарға сәйкес (белгілі бір уақыт аралығында) пайдалану үшін қолданады. МАЖ-дегі белсенді нысандар деп біз агенттер мен одақтарды айтамыз. Грид-ді басқару кезінде әр белсенді объектінің бірнеше мақсаты болады, олардың әрқайсысының өз басымдылығы болады деп ойлаймыз. Сонда, одақ деп біз белгілі бір сандағы агенттердің олардың басым мақсаттарының үйлесімді бағдарлануы негізінде қауымдастыққа уақытша бірігуі болып табылатын құрылымды айтамыз [ ]. Агенттің мұндай қауымдастыққа кіруі оның

басым мақсатына жету қажеттілігімен және осы бағытта одақтастарды іздеумен байланысты. МАС шеңберінде әрбір агент белгілі бір уақытта тек бір одақ мүшесі бола алады. Одаққа біріккен кезде агенттердің ресурстары жалпыға айналады. Одақ құру құру модельдері және одақ серіктестері арасындағы міндеттерді бөлу, сондай-ақ жұмысты жоспарлау модельдері МАС-тағы агенттердің өзара әрекеттесу процестерін сипаттауға және зерттеуге мүмкіндік береді [70].

Басқарудың матрицалық құрылымына сәйкес ұйымдастырылған және бірнеше зерттеу институттарының модельдерінде одақтық аппараттарды қолдану институттардың ресурстарынан асатын үлкен тапсырыс түскен кезде тиімді болады. Бұл жағдайда белгілі әдістер мен модельдер барлық тапсырыстардың уақытында орындалуына кепілдік бермейді. Одақ бөлімшелердің басшылары-агенттеріне келісімді түрде келісуге және үлкен тапсырысты уақытында орындау үшін жалпы ресурстарды пайдалану бойынша бірлескен іс-қимыл жоспарын құруға мүмкіндік береді. Осылайша, коалицияларды құру және торлы автомат қолдану мултиагентті модельдеу мүмкіндіктерін кеңейтеді және оларды бағдарламалық қамтамасыздандыру жүйесін дамытуда қолдану орынды [71].

Одақтық модельде 2.2-суретте көрсетілген процестер жұмыс істейді, олар:

- одақ құру (өзара тиімді ынтымақтастық, жанжалды жағдайларды шешуде бәсекеге қабілеттілікті күшейту агенттері бастамашылық етеді);
- агенттердің одақ ішіндегі олардың жалпы мақсатына жетуге бағытталған өзара әрекеттестігі (келесі негізгі процестерді қосқанда: белгілі бір әдістерге сәйкес агенттердің бірлескен шешімдерін қалыптастыру, уақыт шектеулерін ескере отырып, одақтың жұмыстарының оңтайлы жоспарларын құру.
- коалициялардың бір-бірімен және жеке агенттермен өзара әрекеттесуі (агенттер мен коалициялардың белгілі бір өзара әрекеттесу әдістеріне сәйкес мінез-құлық стратегияларын қолданудан туындайтын);
- Екінші деңгейлі мақсаттарға қол жеткізуге бағытталған агенттердің өзара әрекеттестігі (агент мінез-құлқының белгілі бір стратегиясын қолданудан туындайды).



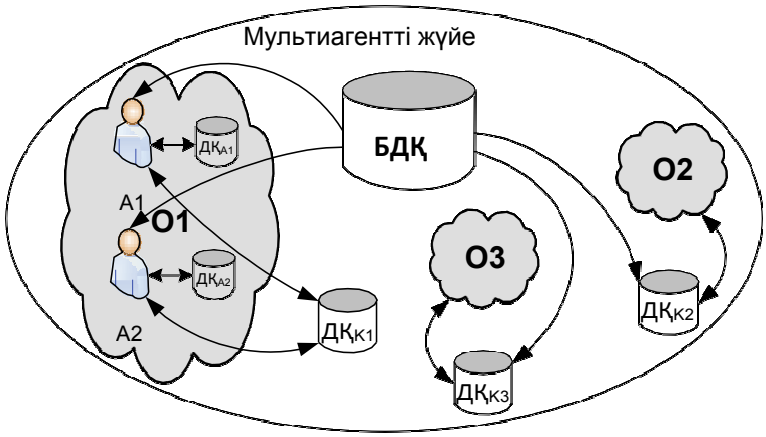
Сурет 13- Агенттердің өзара одақ құру схемасы

Ұсынылып отырған одақтық моделді сипаттау үшін келесі негізгі ұғымдар анықталған. Көп агенттік жүйе келесі құрылымнан тұрады:

$$MAS = \{A_1, \dots, A_k\}, P_A, \{K_1, \dots, K_a\}, P_K, G, t_{mas} \quad (1)$$

бұл жердегі $\{A_1, \dots, A_k\}-$ МАЖ-дегі бірнеше агенттер; $\{K_1, \dots, K_a\}$-МАЖ-дегі одақ жиыны; G-МАЖ-дегі жалпы білім қоры; , P_A - агенттің жұмыс жоспары; P_K-коалицияның жұмыс жоспары; t_{mas}- МАЖ-дегі уақыт;

Әрбір K_i одағы және әр агент A_i қажетті ақпаратты ала отырып, KB_{Ki}және KB_{Aj}білім базаларына қол жеткізе алады. G (жалпы білім беру қоры) - бұл барлық одақтар мен МАЖ агенттері үшін қол жетімді ақпараттық қоймасы болып табылады. Онда осы модельде жұмыс істейтін барлық грид параметрлерінің сипаттамасы; агенттер мен одақтар арасындағы қақтығыстарды шешудің әдістері $\{Q_1, \dots, Q_s\}$ және агенттер мен коалициялардың мінез-құлық стратегиясының $\{Str_1,\dots,Str_y\}$ алгоритмдері бар. МАЖ-дегі агенттер мен одақтар арқылы ақпаратты қабылдау және тіркеу схемасы 2.3 суретте көрсетілген.



Сурет 14- Агенттердің одақ ішінде білім дерек қоры арқылы байланысуы

Грид одақ моделіндегі агенттер келесі құрылым ретінде ұсынылған:

$$A = Aty_A,\ \{G_{A1},\ldots,G_{An}\},\ KB_A,\ \{Str_{A1},\ldots,Str_{Av}\} \qquad (2)$$

мұндағы: $Name_A$ - агенттің аты; $\{G_{A1},\ldots,G_{An}\}$ - агент мақсаттары; KB_A - агенттердің білім қоры; $\{Str_1,\ldots,\ Str_y\}$ - агенттің өзара әрекеттесуінің рұқсат етілген стратегияларының жиынтығы. МАЖ-дегі A_iагентінің әрқайсысы KB_{Ai} білім қорына ие және ол білім қорында мынандай мәліметтер сақталады: нақты МАЖ агенттері бар $\{Str_{Ai}^1,\ldots,Str_{Ai}^v\ \}$ мінез-құлқының қазіргі стратегиялары, агенттің өз ресурстарын $\{Res_{Ai}^1,\ldots,Res_{Ai}^m\ \}$ пайдалану туралы мәліметтер, PD_{Ai} агентінің іс-қимыл жоспары, PW_{Ai} агентінің жұмыстарды жасау жоспары.

МАЖ-дегі A_i әр агенттерінің әрқайсысы өзінің KB_{Ai} білімдер базасында сақталған $\{G_{Ab}^1,\ldots,G_{Ai}^n\}$ мақсат жиынтығына ие, соның ішінде G_{Ai}^D басым мақсат және $\{\ G_{Ab}^1,\ldots,G_{Ai}^h\}$ қосалқы мақсаттар жиынтығы бар. A_iагентінің G_{Ai}^Dбасым мақсатына жеткенде немесе оның қоршаған ортаның күйінің өзгеруіне байланысты өзгеруі кезінде, A_i агентінің жаңа доминантты мақсаты оның екінші деңгейлі $G_{Ab}^1,\ldots,G_{Ai}^h$мақсаттарының жиынтығына айналады. A_i агент $\{G_{Ab}^1,\ldots,G_{Ai}^h\}$ мақсаттарының ішінен белгілі бір мақсатқа жету үшін Str_{Ai}^{Ai}-ді басқа A_jагентімен өзара әрекеттесу стратегиясын таңдайды. Агенттердің арасындағы қақтығыстар агенттердің мінез-құлық стратегияларын пайдалану немесе аукциондарды ұйымдастыру негізінде белгілі бір $Q_1,\ldots,Q_s\}$ әдістеріне сәйкес шешіледі.

МАЖ-дегі әр A_i агенті өзінің $Res_{Ai}^1,\ldots,Res_{Ai}^m$ ресурстарын басқарады. Басқару жеке KB_{Ai} дерек қорының және жалпы GKB білім дерек қорын қолдану арқылы L_{Ai} өмірлік циклі негізінде A_i агентінің PD_{Ai} жоспарының көмегімен жүзеге асады. L_A- өмірлік циклі МАЖ-дегі агент анықталғаннан бастап оның жұмыс істеуі аяқталғанға дейінгі барлық кезеңіне сәйкес келеді.

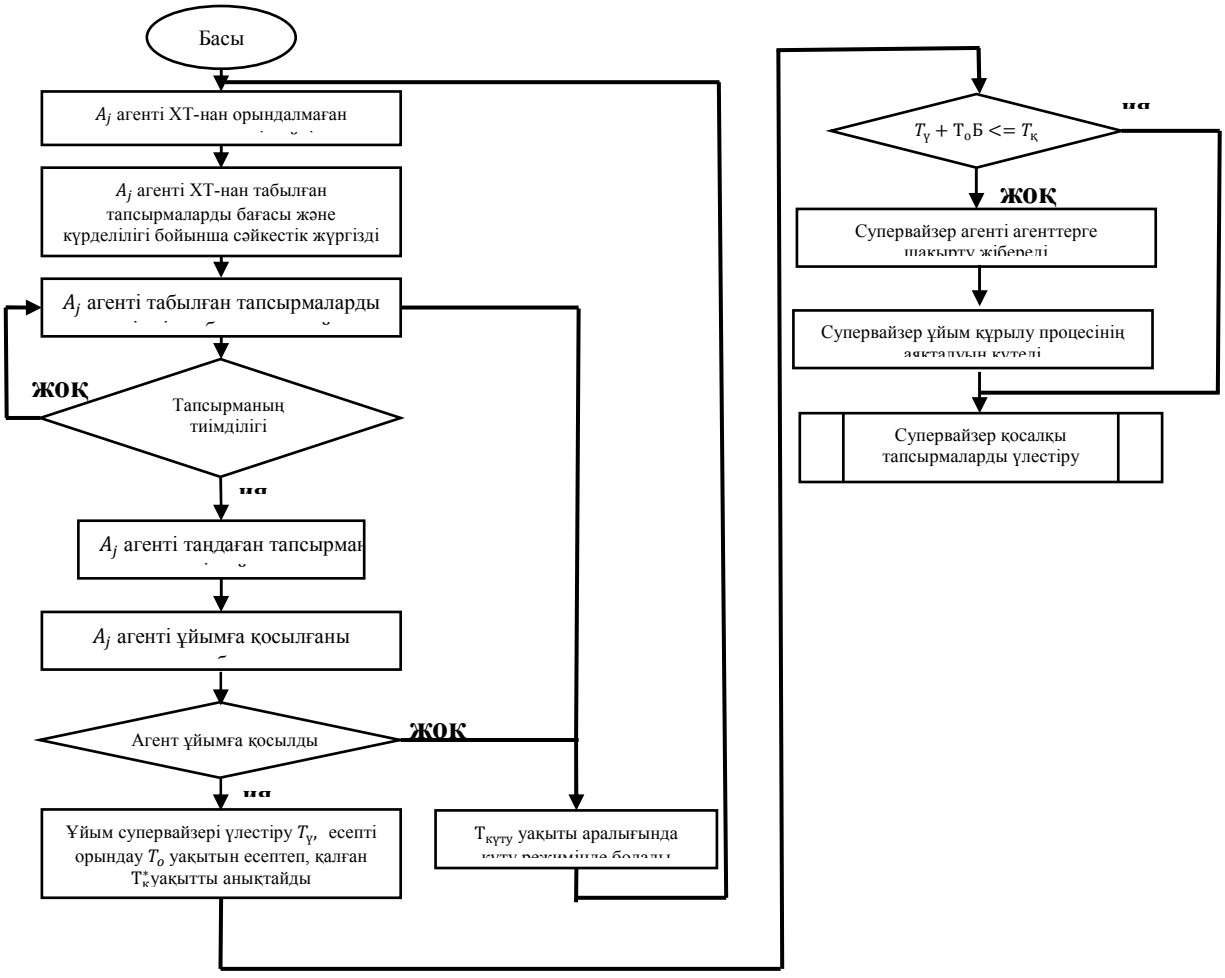
МАЖ пайдалану процесінде $\{A_1,\ldots,A_k\}$ агенттері бір-бірімен әрекеттесе алады, өзара тиімді ынтымақтастықты ұйымдастыру және қайшылықтар

туындаған кезде, қажетті ресурстарды алу мүмкіндігін арттыру мақсатында $\{K_1, \dots, K_a\}$ одағын құрылады.

МАЖ-дегі агенттер коалициясы келесі құрылымнан тұрады:

$$K = Name_K, \{A_1,\dots,A_m\}, G_K, \{Str_1,\dots,Str_v\}, KB_K, t_{WAi} \quad (3)$$

мұндағы: $Atau_K$ - одақ атауы; $\{A_1, \dots, A_k\}$ – одаққа кіретін агенттер жиынтығы; GK - коалицияның мақсаты; $\{ Str_1,\dots,Str_v \}$ - одақтың мінез-құлқы үшін қабылданатын стратегиялардың жиынтығы; KB_K - коалицияның білім базасы; t_{WAi} – одақтың жұмысты орындау уақыты.



Сурет 15-Мультиагентті ұйым құру алгоритмі

3.2 Агенттер арасында тапсырмаларды бейімді бөлу әдісі.

Агенттер қауымдастығындағы кіші тапсырмаларды үлестіру әдісінің негізгі идеясы қауымдастық мүшелері арасында барлық кіші тапсырмаларын үлестіру емес, қоғамдағы барлық агенттерді тиімді жұмысқа жұмылдыру үшін қажетті минималды бөлігін ғана үлестіру негізделген. Бұл келесі қосалқы тапсырмалардың тораптар арасында бейімделіп бөлінуіне байланысты, ол өз

кезегінен торапт босаған кезінде ғана тапсырманы жіберіп отырады. Бұл тораптардың параметрлері туралы нақты ақпаратты анықтау, яғни оларға бейімделу үшін тапсырмалардың келесі бөлігін бөлуге мүмкіндік береді [72].

Ұсынылған әдістің негізгі ережелерін қарастырайық.

- Қауымдастық құрамына кіретін жеке торап «фрилансерлерінің» параметрлері тапсырманы орындау кезінде өзгеруі мүмкін, барлық қосалқы тапсырмаларды үлестірудің қажеттілігі болмайды, өйткені бұл үлестірудің тиімділігі уақыт өте келе өзгеруі мүмкін. Сондықтан үлестіру «тереңдігін» бір тапсырмаға шектеу ұсынылады.
- Бұл жағдайда келесі қосалқы тапсырманы таңдау процесі жеңілдейді. Қоғамдастықтан алынған барлық мәліметтерге сәйкес агент, қосалқы тапсырманы бірнеше тапсырма ішінен таңдап алады. Сонымен қатар, ішкі жиыннан тапсырманы таңдау үшін келесі эвристикалық әдісті қолдану ұсынылады: агент таңдалған кіші тапсырмалардың әрқайсысын шешкен жағдайда қоғамдастықтың тапсырманы шешу барысында уақыт жоғалтуын бағалауы керек, содан кейін шығындар минималды болатын шешімді қабылдауы керек. Мұндағы уақыт жоғалту дегеніміз - таңдау агенті мен қоғамдастықтың қалған агенттері белгілі бір тапсырманы шешу уақыты арасындағы максималды айырмашылықты білдіреді. Осылайша, тапсырманы таңдау үшін агент шешімге дайын әрбір ішкі проблеманың кез-келгенін алады және басқа агенттердің әрқайсысы оны шешуге кететін уақытты салыстырады. Содан кейін ол әр тапсырма үшін алынған минималды шығынды таңдайды және сәйкес тапсырманы шешуге кіріседі. Мұндай эвристика агенттердің осы торапта тиімсіз тапсырмалардың орындалуына жол бермейді;
- A_i агенті келесі қосалқы тапсырманы алу үшін өзінің нақты уақыттағы мүмкіндігін анықтап отыру үшін агент өз торабының P_j өнімділігін үнемі бақылап отыруы қажет. Бұл мәнді алдыңғы тапсырманың IED шешімінің нәтижелері бойынша бағалауға болады.

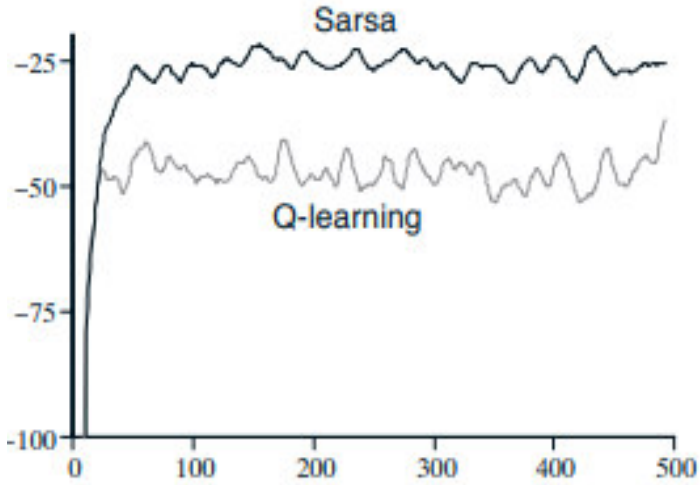
Құрылған агенттер арасында тапсырмаларды адаптивті үлестіріп отыру мақсатында супервайзер агентті біз машиналық оқыту әдісі арқылы оқытуды ұйғардық. Ол үшін біз оқи отырып үйрену алгоритмін қарастырдық.

Оқи отырып үйрену (reinforcement learning) –машиналық үйретудің бір әдісі болып табылады. Оқи отырып үйрену -сандық мәнді қабылдай отырып кейбір келген сыйлық сигналдарын барынша арттырып, табылған жағдайды қалай іс-әрекет үшін қолдануға үйретуді айтамыз. Үйренушіге машиналық үйренудегі секілді қандай іс әрекет жасаған тиімді болатыны айтылмады. Оның орнына, әр түрлі іс-әрекет жасай отырып, қай іс-әрекет өзіне жақсы сыйлық алап келетінін табу керек. Аса тиімді және қызықты жағдайда іс-әрекет тек қана сыйлық үшін ғана қажет емес, ол табылған жағдай арқылы келесі күйлерге өтіп отырады. Бұл екі сипаттама- үлгі әдісі мен қателер әдісі, күшейте отырып үйрету әдісінің ең басты бөлігі болып табылады. Агент ортамен байланыса отырып белгілі бір мақсатқа жетуді көздейді, ол мақсатқа тез және кедергісіз жету үшін агент ортаны танып білу керек, қандай кедергі бар қай жол тиімді екенін. Ортаны танып білу үшін агент белгілі бір іс-әркет жасай отырып өзіне қай іс-

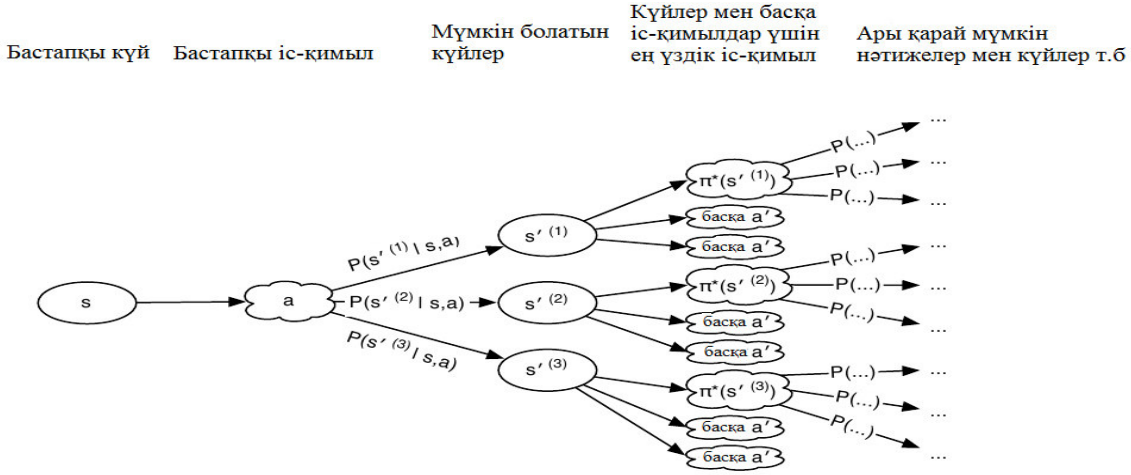
әрекет тиімді екенін біліп отыру керек, сол жасалған іс-әрекет тиімді екенін білу үшін алі жасалмаған іс әрекеттерді жасауы тиіс. Агент жаңа іс-әрекет жасау үшін кезінде өзінің бұрынғы тиімді іс әрекеттеріне сүйене отырып жасауы тиіс. Іс-әрекет барынша тиімді ьолғанға дейін агент өзінің жұмысын тоқтатпайды. Күшейте отырып үйрету әдісінің мынадай қағидалары бар

- өзара іс-қимыл арқылы оқыту;
- Мақсатты дайындық;
- Қоршаған ортамен өзара іс-қимыл арқылы оқыту.

Бағалау функциясы – күшейту функциясы осы сәтті қайсы күй тиімді екенін көрсетсе, бұл функция жалғасатын кезеңде қандай күй тиімді екенін көрсетеді. Күшейту функциясы - күшейе отырып үйрену әдісі кезінде мақсатты анақтайды. Екі түрлі алгоритмді салыстыру нәтижесінде Sarsa алгоритмін таңдап алған болатынбыз. Төмендегі суретте Sarsa және Q-оқыту әдістерін салыстыру көрсетілген [74].

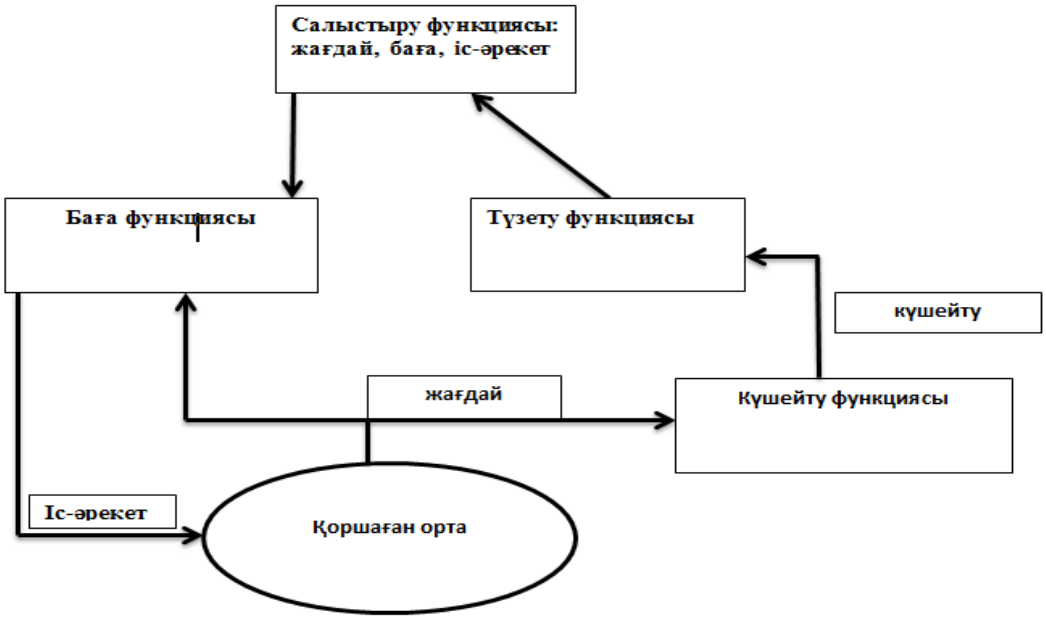


Сурет 17- Sarsa және Q-оқыту алгоритмдерін салыстыру нәтижесі.



Сурет. 16- Агенттердің ортамен байланыса отырып үйрену схемасы

Агент және орта дискретті уақыт қадамы бойынша өзара байланысып отырады t=1,2,3,4,5……. Әр t уақыт қадамында агент ортаның күйі туралы мәлімет қабылдап отырады, $s_t \epsilon S$ бұл жердегі S мүмкін болатын барлық күйлер, осы күйлерге сүйене отырып агент $a_t \in A(s_t)$ іс-әрекет жасайды мұндағы $A(s_t)$, S_t күйіндегі мүмкін болатын барлық іс-әрекет жиыны. Агент келесі бір қадам жасаған кезде, өзінің іс-әрекетіне жауап ретінде агент өзіне $r_{t+1} \in R$ сандық күшейуін қабылдайды, және өзін s_{t+1} күйіне өткізеді [73].



Сурет 18- Оқи отырып үйрену әдісі арқылы орындалатын жүйе жұмысының принципі

Әр уақыт қадамында, агент әрбір әрекетті таңдау ықтималдығына ие бола отырып күйлерді салыстырады. Бұл бейне агенттердің ережелері деп аталады және былай белгіленеді π_t бұл жердегі $\pi_t(s,a)$ ықтималдығы $a_t = a$ егер $s_t = s$ болғандағы жағдайға тең. Тәжірибе нәтижесінде агент өзінің ережелерін өзгертіп отыратынын күшейте отырып оқыту әдісі көрсетіп отыр. Агенттің мақсаты ,ұзақ мерзімді жұмыс істеу кезіндегі алынған күшейтулердің санын барынша нығайту. t қадам уақытынан кейін тізбектеле алынған күшейтулердің саны былайша өрнектеледі $r_{t+1}, r_{t+2}, r_{t+3}, \dots$. оқыту кезінде күтілген нәтиже барынша нығая түседі , ол нәтиже былай белгіленеді R_t.

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_T \quad (4)$$

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Choose a from s using policy derived from Q
         (e.g., ε-greedy)
   Repeat (for each step of episode):
      Take action a, observe r, s'
      Choose a' from s' using policy derived from Q
         (e.g., ε-greedy)
      Q(s, a) <-- Q(s, a) + α[r + γQ(s', a')- Q(s, a)]
      s <-- s'; a <-- a';
   until s is terminal
```

Сурет-19 Sarsa алгоритмі

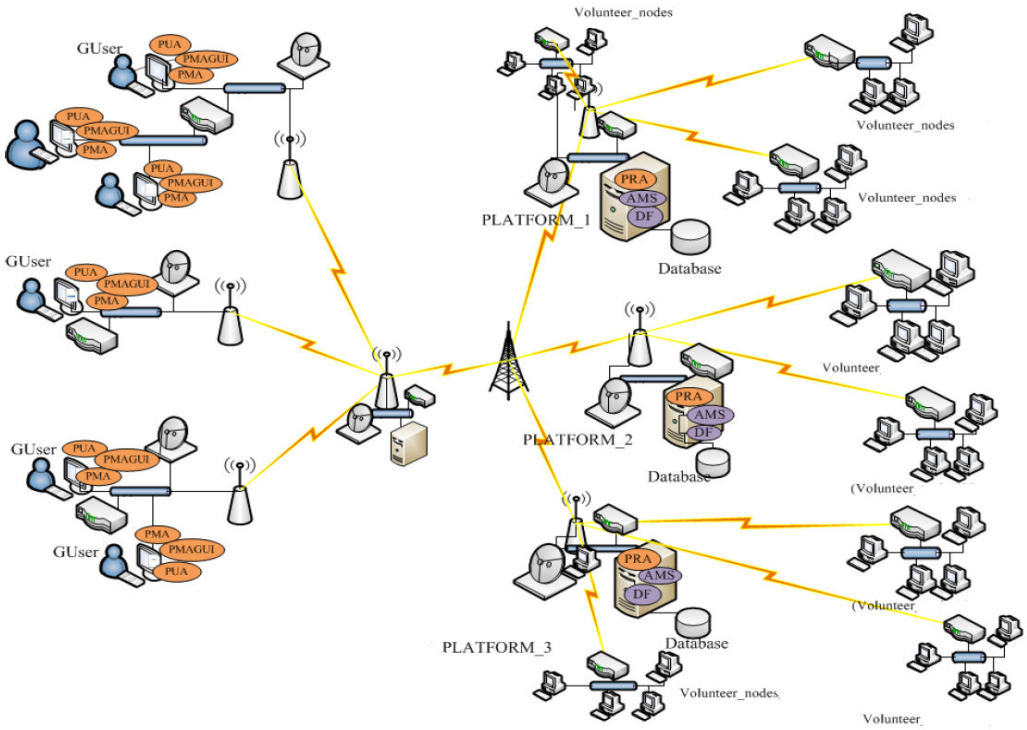
бірнеше әрекеттердің ішінен тиімді әрекетті таңдау үшін (5) формуланы қолданамыз.

$$P = \frac{e^{\frac{Q(s,a)-max_b\, Q(s,b)}{T}}}{\sum_a e^{\frac{Q(s,\ a)-max_b\, Q(s,b)}{T}}} \quad (5)$$

Жоғарыда аталған алгоритм жүйеге келіп түскен тапсырмаларды шешу үшін GRID-де агенттер ұйымын құруға мүмкіндік береді [76]. Бұл жағдайда қолданушышының тапсырмасын шешуге арналған ұйым құрамы оның тораптарының ағымдағы көрсеткіштеріне байланысты бейімделіп отырады. Басқаша айтқанда, үйым құрамы оның құрамына кіретін «фрилансерлердің» тораптарының динамикалық өзгеретін параметрлеріне үнемі бейімделіп отырады.

4. AGENT-GRID ГРИД ЖҮЙЕСІ ҮШІН БАҒДАРЛАМАЛЫҚ КЕШЕНДІ ҚҰРУ.

Мәселелерді шешуде Agent-Grid компонентінің өзара әрекеттесуі осы бөлімде талқыланады. Agent-Grid платформасы прокси-сервер арқылы қосылады [75]. Нәтижесінде біріктірілген Agent-Grid инфрақұрылымы және оның дәйектілік сызбасы сәйкесінше 15-суретте көрсетілген. Келесі шарттар анықталды: (1) прокси арқылы бөлек платформалар қосады және (2) ресурстар жеткізушісі арқылы ресурстарды бөліседі, осылайша түйінге қосылады. интервалдан кейін жаңартылады.



Сурет 20- Agent-Grid инфрақұрылымы

ACL хабарламалары объект ретінде жүзеге асырылады. Жіберу әдісі (8-сурет) құрылған хабарламаны бағыттау үшін қолданылады. Агент байланысы хабарламаны асинхронды жіберуге негізделгендіктен, белсенді агенттерге кіріс және шығыс хабарламаларын сақтауға арналған пошта жәшіктері тағайындалады.

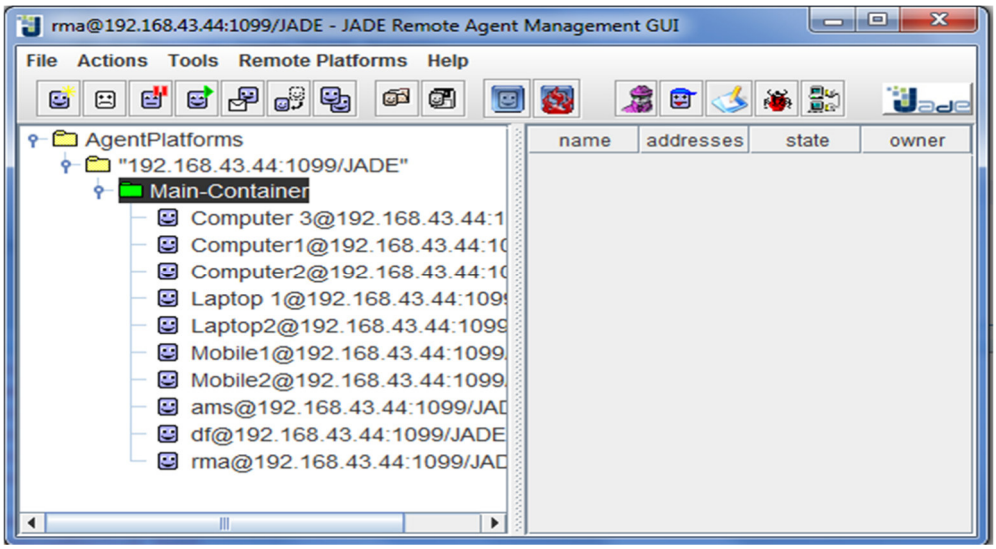
```
ACLMessage out_bound = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
out_bound.addReceiver(storageagent[0]);
out_bound.setContent(FILENAME);
out_bound.setConversationId("Save_File");
out_bound.setReplyWith("out_bound"+System.currentTimeMillis());
myAgent.send(out_bound);
```

сурет 21-Хабар нысаны

```
MessageTemplate receive = MessageTemplate
        .MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
ACLMessage msg = myAgent.receive(receive);
if (msg != null) {
```

Сурет 22-Хабарлама алу

Agent-grid есептеушісі жалпыға ортақ процессорлық ресурстарды бір жүйеге біріктіреді. Супервайзерлер ресурстарды анықтауды, келіссөздер жүргізуді және ресурстарды жеткізушілердің қызметіне басымдық беру үшін таратылған MAS-да шешім қабылдауды жүзеге асырады. MA жобасы және оның агент қатысушылары суретте көрсетілген.



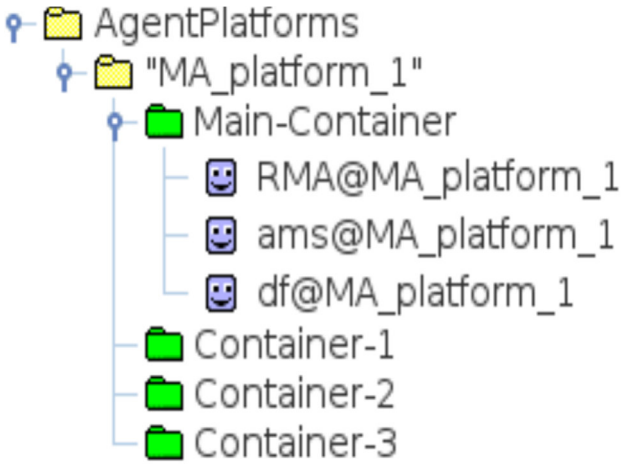
Сурет 23-МАЖ есептеу интерфейсі

Сурет 24-Агенттердің күйін анықтау интерфейсі

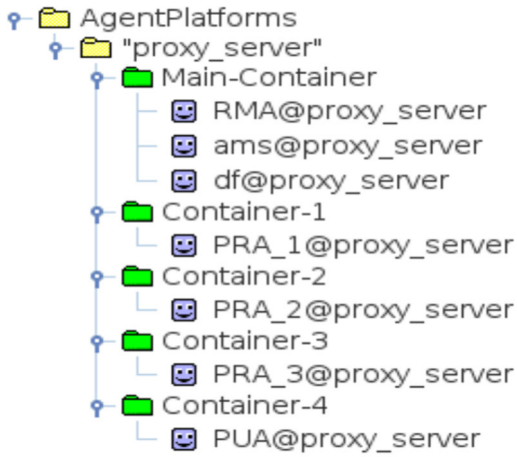
```
root@msclab-Aspire-E1-572:/home/msc-lab/Desktop# java -cp http.jar:iiop.jar:jade
.jar:jadeTools.jar:jadex_jadeadapter.jar:jess.jar:jsr94.jar jade.Boot -container
 -host 172.20.56.49 -port 1099
```

Сурет 25-Jade runtime іске қосу

Компонент процессордың ортақ мүмкіндіктерін пайдалану кезінде MAS-тің функционалдық және техникалық сәйкестігіне бағытталған. Ресурстарды бөлу кезінде CNP-ді бағалау үшін брокер анықтаған үш платформа құрылды. Платформалар прокси-сервер арқылы біріктірілді. Agent-Grid платформасы және прокси-сервер профильдері төменде көрсетілген.



Сурет 26 -Платформа



Сурет 27-Прокси сервер

Әр түрлі процессорлық сипаттамалары бар Linux жүйесінде жұмыс істейтін түйіндер NPA ресурстық провайдері арқылы платформаларға ортақ пайдаланылды. Тасымалдау кезінде сәтті инициализация үшін PMA үшін JADE контейнерлерін бірдей Java нұсқасымен жұмыс жасайтын түйіндерде инициализациялау маңызды болды. 53-суреттегідей негізгі орта орнатылды.

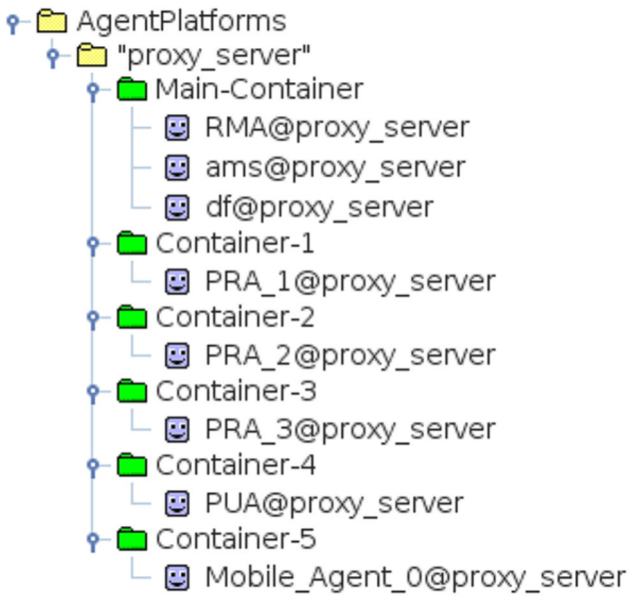
```
msc-lab@msclab-Aspire-E1-572:~$ update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-6-openjdk-i386/jre/bin/java   1061       auto mode
  1            /usr/lib/jvm/java-6-openjdk-i386/jre/bin/java   1061       manual mode
* 2            /usr/lib/jvm/java-7-openjdk-i386/jre/bin/java   1051       manual mode

Press enter to keep the current choice[*], or type selection number: 2
```

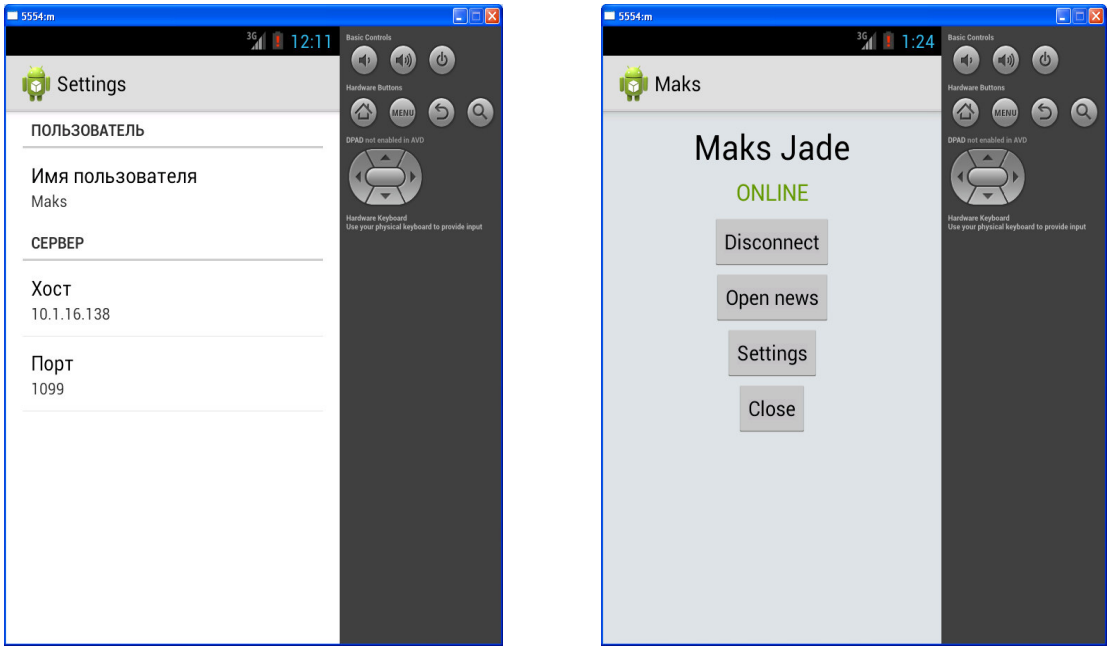
Сурет 29-Java конфигурациясы

55-суреттегі прокси-профиль тапсырманы жүктеуде және есептеу ресурсын таңдауда көрсетілген. Өңдеу уақыты PMA қолдану, қосымшаны өңдеу және нәтижелерді бастапқы түйінге қайтару арасындағы интервал ретінде анықталды.



Сурет. 28 Гетерогенді жүйе.

Гетерогенді ресурстарды тарту арқылы жүйе болғандықтан, біз ерікті қолданушылардың мобильді құрылғыларында қолданған болатынбыз. Мобильді құрылғы үшін құрылған жасақтаманы төмендегі суреттен көруге болады.



Сурет 30-Мобильді құрылғы үшін құрылған Jade жасақтама

5. ҚҰРЫЛҒАН МУЛЬТИАГЕНТТІ ГРИД ЖҮЙЕСІНІҢ НӘТИЖЕСІ ЖӘНЕ ТАЛДАУЛАРЫ.

Екі онжылдықта жер ресурстарын, қоршаған ортаның жай-күйін, түрлі қауіп-қатерлерді бақылау саласындағы түрлі мәселелерді шешу үшін жоғары спектрлі ажыратымдылықты кескіндерді қолдану бойынша зерттеулер жүргізіліп келеді. Гиперспектральды кескіндердің қолдану аймақтары өте үлкен. Гиперспектральды кескінді талдау қашықтықтан зондтау мәселелерін шешудің тиімді және өзекті қосымшаларының біріне айналды. Бүгінгі таңда гиперспектральдік кескін, басқа ЖҚЗ-ға (Жерді қашықтықтан зондтау) қарағанда, дәлірек және егжей-тегжейлі ақпарат алуға мүмкіндік береді. Гиперспектральды қашықтықтан зондтау деректерін өңдеу нәтижелері агроөнеркәсіптік кешенде, геологиялық барлауда, қоршаған орта мониторингінде көбірек қолданылуда, бұл осы тақырыптық бағыттар бойынша ГК талдауы бойынша басылымдардың көбеюімен расталады.

K-орташа әдісі кластерлік нүктелердің осы кластерлердің орталықтарынан жалпы квадраттық ауытқуын азайтуға тырысады:

$$\phi = \sum_{\mathbf{x}_i \in \mathbf{X}} \min_{s_j \in S} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2 , \tag{6}$$

Бұл мәселенің нақты шешімі NP-толық болып табылады. Әдеттегі k-алгоритмінде бастапқы шешім кластерлік орталықтарды кездейсоқ таңдау арқылы қайталанбалы түрде ізделінеді. Бұл алгоритмнің бірнеше кемшіліктері бар. Жаһандық минимумның орнына жалпы стандартты ауытқудың жергілікті минимумының орташа жетістігі, сондай-ақ баяу конвергенция.

K-means ++ алгоритмі осы кемшіліктерді орталықтардың бастапқы жуықтауын таңдаудың оңтайлы алгоритмін қолдану арқылы жояды:

- Бірінші орталық μ_j X жиынынан кездейсоқ таңдалады.
- Қалған центрлер μj X-тан кездейсоқ таңдалады:

$$p(x) = \frac{D(x)^2}{\sum_{\mathbf{x}_i \in \mathbf{X}} D(\mathbf{x}_i)^2} , \tag{7}$$

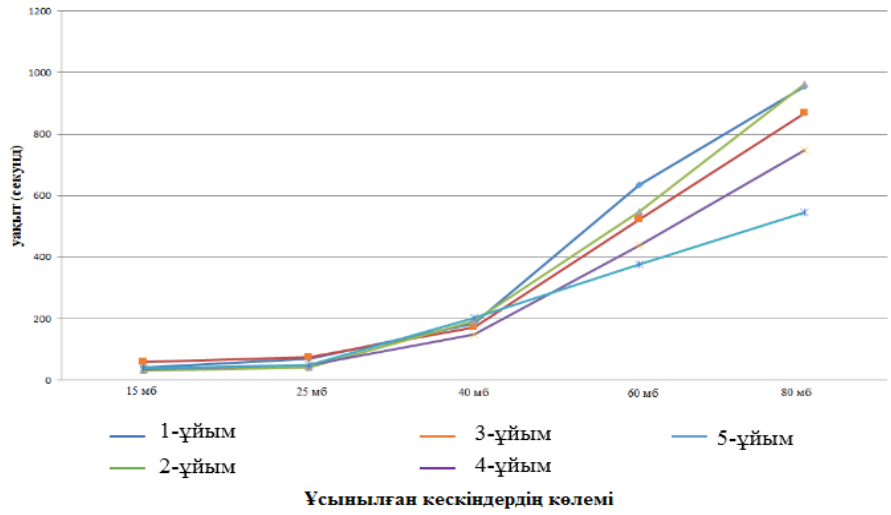
Барлық орталықтар таңдалғаннан кейін әдеттегі k-алгоритмі іске қосылады.

Кесте 3-Қолданылған кластерлеу алгоритмдерін салыстыру нәтижелері

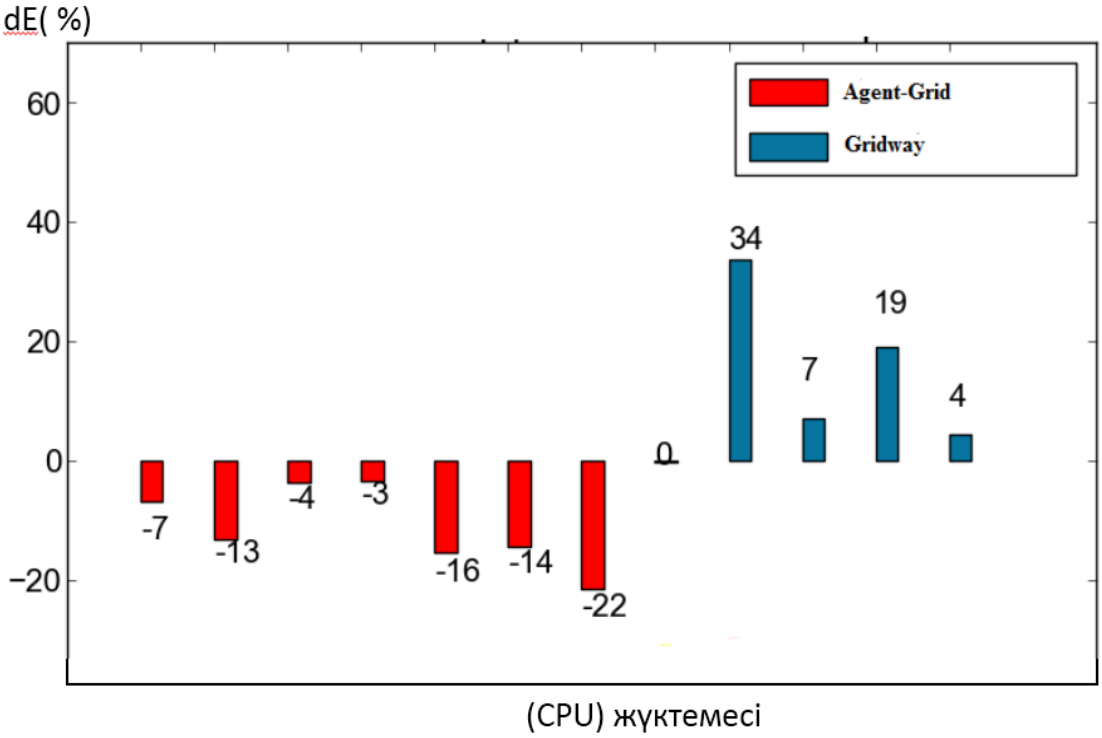
өлшемі, d	қателік ϕ		уақыт T, с	
	ISODATA	KM++	ISODATA	KM++
10	$2{,}17 \cdot 10^3$	$9{,}58 \cdot 10^2$	17,42	15,09
20	$7{,}16 \cdot 10^4$	$2{,}1 \cdot 10^3$	32,11	27,16
40	$9{,}67 \cdot 10^5$	$7{,}8 \cdot 10^3$	70,05	52,17
80	$2{,}62 \cdot 10^7$	$2{,}5 \cdot 10^5$	298,51	102,52
200	$3{,}45 \cdot 10^8$	$7{,}2 \cdot 10^6$	872,08	315,11

Жүйенің өнімділігін тексеру үшін біз келесі инфрақұрылымды құрдық: Intel Core i5 процессорларының 3-буыны және 4 Гб жедел жады бар 13 компьютер; Intel Xeon процессорлары және 8, 16 және 16 Gb жедел жады бар 3 HP серверлік машиналары; 64 виртуалды машинасы бар HP визуализация кластері; және Android ОЖ басқаратын 8 ұялы телефон. Тестілеу үшін гиперспекиральді кескіндерді кластерлеу ұсынылды: кіріс өлшемдері: 15 мб, 25мб, 40 мб, 60 мб және 80 мб көлемді (кіріс мөлшері - бұл гиперкубалық құрылымның бір өлшеміне арналған нүктелер саны). Жобалардың кез-келген жиынтығында әр түрлі ішкі инфрақұрылымдар саны болды.

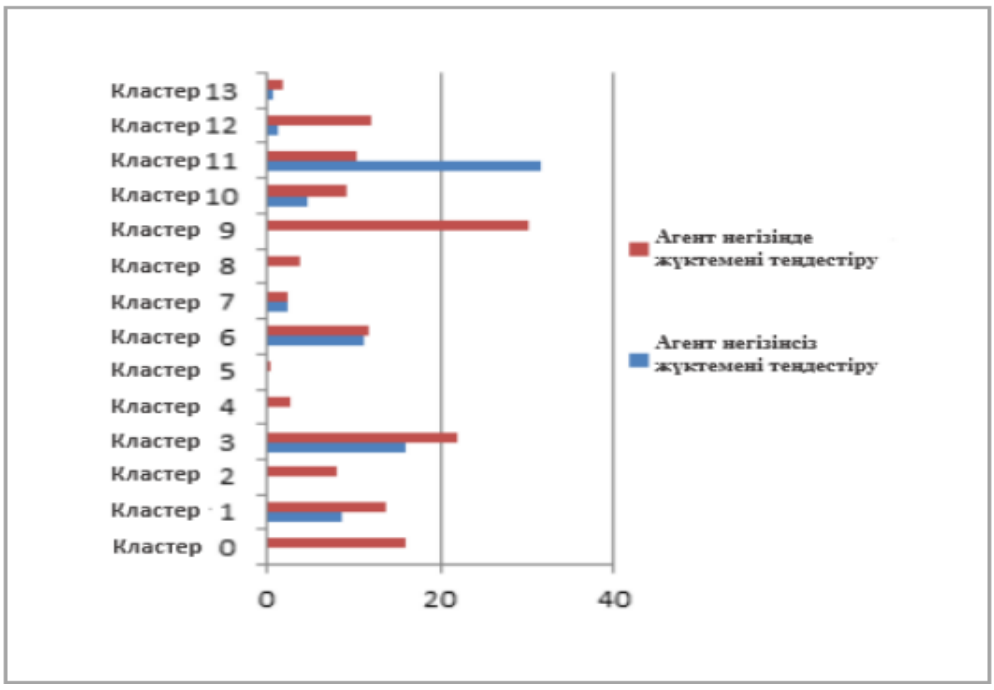
Алдымен біз қол жетімді ішкі инфрақұрылымдар санының артуымен жүйенің жұмысының күтілетін өсуін алдық. Атап айтқанда, ішкі инфрақұрылымдар санының бірден беске өсуі есептеу уақытын үш есе азайтады.



Сурет 31-ішкі одақтардың прототипін тестілеу нәтижесі.



Сурет 32-Agent-grid және Gridway грид жүйесін ұйымдастыру әдістерінің тиімділігі



Сурет. 33-Агенттік негіздегі және агенттік негізінсіз кластер жүктемесін теңдестіру (%).

ҚОРЫТЫНДЫ

Бұл жұмыста атқарушы есептеу түйіндерінің саны мен параметрлері динамикалық өзгеретін жағдайдағы күрделі есептерді шешуге арналған GRID операциясын көп агенттік басқарудың әдістері мен алгоритмдерін жасаудың ғылыми мәселесі шешілді. Жұмыста ұсынылған әдістер мен алгоритмдер GRID жүйелерін жеке қолдардағы есептеу ресурстарына құруға мүмкіндік береді, бұл GRID-тегі есептеу шығындарын едәуір төмендетуге, сонымен қатар күрделі мәселелерді шешу кезінде өнімділікті едәуір арттыруға мүмкіндік береді. Осы жұмыстың тақырыбы бойынша зерттеу және әзірлеу барысында келесі ғылыми жаңалығы бар теориялық және қолданбалы нәтижелер алынды:

1) қазіргі қолданыстағы GRID жүйелеріне талдау жүргізілді. Қолданыстағы тор жүйелерінің мәселелерін анықтау үшін олардың GRID жүйесінің моделі ұсынылған.

2) Орталықтандырылмаған көп агенттік әдіс күрделі мәселелерді шешуде GRID өнімділігін арттыруға мүмкіндік беретін «фрилансерлер» негізінде құрылған GRID-те брокерді ұйымдастыру.

3) Пайдаланушы есебін шешуге арналған агенттер ұйымын құру әдісі ұсынылған, оны шешуге қатысатын агенттердің санын азайту үшін есепті шешуге уақытты бағалаумен ерекшеленеді. Ұсынылған әдіс қажетті уақыт шеңберінде күрделі мәселені шешуге жеткілікті көлемдегі қауымдастықтар құруға мүмкіндік береді.

4) агенттер қауымдастығында кіші тапсырмаларды бөлу әдісі әзірленді, атқарушы биліктің ағымдағы параметрлерін ескере отырып әр түрлі түйінде шешілетін келесі қосымша тапсырманы тағайындау кезінде есептеу түйіндері. Ұсынылған әдіс кластер түйіндерінің жүктемесін теңдестіру үшін қауымдастық түйіндері арасындағы кіші тапсырмаларды бөлуді азайтуға мүмкіндік береді, бұл мәселені шешу уақытын қысқартады.

Біз жасаған жүйе қолданыстағы жүйелерге қарағанда жылдамырақ жұмыс істейді, ұсынылған архитектура торлы жүйелерді ұйымдастырған кезде көп агенттік жүйелерді пайдалану өзін-өзі ұйымдастыру мүмкіндігіне және өзін-өзі оқытуға байланысты есептеу уақытын қысқартады деп көрсетті. Есептеу жүйесінде біздің жүйенің жұмысын тексеруге арналған эксперименттер жүргізілді; ұсынылған модельдер мен алгоритмдердің үдеуін тексеру үшін эксперименттер байланысты және ажыратылған есептермен өткізілді. Ұсынылған модель штаттан тыс ресурстарды, яғни гетерогенді түйіндерді пайдаланған кезде тезірек жұмыс істейді.

ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР

1 David J. Brown, Charles Reams 2010. Toward Energy-Efficient Computing. Communications of the ACM CACM Volume 53 Issue 3, March 2010 DOI: 10.1145/1666420.1666438.

2 Murugesan, San, ”Harnessing Green IT: Principles and Practices,” IT Professional vol.10, no.1, pp.24,33, Jan.-Feb. 2008 doi: 10.1109/MITP.2008.10.

3 Luiz Andr Barroso, Jeffrey Dean, and Urs Hlzle,Web Search for a Planet: The Google Cluster Architecture IEEE Micro 23, no. 2 (April 2003)..

4 Parthasarathy Ranganathan. 2010. Recipe for efficiency: principles of power-aware computing. Commun. ACM 53, 4 (April 2010). DOI=10.1145/1721654.1721673.

5 Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. / International J. Supercomputer Applications, 15(3), 2001.

6 Distributed Computing Technologies, Inc. URL: Distributed.net (Дата обращения: 25.02.2012).

7 The TOP500 statistics // The TOP500 project URL: http://i.top500.org/stats (Дата обращения: 02.09.2012).

8 Николай Виноградов Анализ современного состояния ГРИД проектов в мире //Jet Info online - информационный бюллетень компании "Инфосистемы Джет" URL: http://www.jetinfo.ru/stati/?nid=0161 d223cb9ed29f66470cbf7931ae22.

9 Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. / Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

10 Фостер Я., Кессельман К., Тьюке С. Анатомия грид: создание масштабируемых виртуальных организаций, 2001. http://gridclub.ru/library/publication.2004-11- 29.7104738919.

11 L. F. G. Sarmenta and S. Hirano, “Bayanihan: building and studying web-based volunteer computing systems using Java,” Future Generation Computer Systems, vol. 15, pp. 675–686, 1999.

12 D. P. Anderson, “BOINC: A system for public-resource computing and storage,” in International Workshop on Grid Computing, 2004, pp. 4–10.

13 “SETI@home.” [Online]. Available: http://setiathome.ssl.berkeley.edu/. [Accessed: 21-Feb-2015].

14 “Distributed.net.” [Online]. Available: www.distributed.net/. [Accessed: 23-Feb-2015].

15 A. Chien, B. Calder, S. Elbert, and K. Bhatia, “Entropia : architecture and performance of an enterprise desktop grid system,” Journal of Parallel and Distributed Computing, vol. 63, pp. 597–610, 2003.

16 M. Vladoiu and Z. Constantinescu, “Development Journey of QADPZ - A Desktop Grid Computing Platform,” International journal of Computers, Communicatuons & Control, vol. 4, no. 1, pp. 82–91, 2009.

17 P. Kacsuk, J. Kovacs, Z. Farkas, A. C. Marosi, G. Gombas, and Z. Balaton,

"SZTAKI Desktop Grid (SZDG): A flexible and scalable desktop grid system," Journal of Grid Computing, vol. 7, no. 4, pp. 439–461, Sep. 2009.

18 B. O. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, K. Schauser, and D. Wu, "Javelin: Internet-Based Parallel Computing Using Java," Concurrency and Computation: Practice and Experience, vol. 9, no. 11, pp. 1139–1160, 1997.

19 A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande, "Folding@home: Lessons from eight years of volunteer distributed computing," in IEEE International Parallel and Distributed Processing Symposium, 2009, pp. 1–8

20 Z. Constantinescu, "A Desktop Grid Computing Approach for Scientific Computing and visualisation," Norwegian University of Science and Technology, 2008.

21 D. P. Anderson, C. Christensen, and B. Allen, "Designing a Runtime System for Volunteer Computing," in IEEE Computer, 2006.

22 "Folding@home." [Online]. Available: http://folding.stanford.edu/. [Accessed: 17-Feb-2015].

23 A. L. Beberg and V. S. Pande, "Storage@home: Petascale Distributed Storage," in IEEE International Parallel & Distributed Processing Symposium, 2007, pp.1–6.

24 GLOBUS Toolkit URL: http://www.globus.org/toolkit/ (Дата обращения: 12.11.2011).

25 Computing with HTCondor // The Department of Computer Sciences at the University of Wisconsin-Madison URL: http://www.cs.wisc.edu/condor/downloads (Дата обращения: 12.11.2012).

26 Francine Berman Adaptive Computing on the Grid Using AppLeS URL: http://walfredo.dsc.ufcg.edu.br/papers/10369.pdf (Дата обращения: 19.12.2012).

27 СИ. Соболев. Архитектура нового поколения системы метакомпьютинга X-Com // Распределенные вычисления и Грид-технологии в науке и образовании. Труды третьей международной конференции. Дубна, 30 июня - 4 июля 2008 г. С. 123-126.

28 Kourpas, "Grid Computing: Past, Present and Future - An Innovation Perspective," 2006.

29 I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in Grid Computing Environments Workshop, 2008, pp. 1–10.

30 A. Dimakis, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," IEEE Transactions on Information Theory, vol. 56, no. 9, pp. 4539–4551, 2010.

31 Y. Wang and A. Merchant, "Proportional-share scheduling for distributed storage systems," in 5th USENIX Conference on File and Storage Technologies, 2007, p. 4.

32 Полежаев Петр Николаевич Экспериментальное исследование алгоритмов планирования задач для грид-систем с использованием симулятора / Материалы Международная конференция «Высокопроизводительные вычисления» HPC-UA'2012 (Украина, Киев, 8-10 октября 2012 года)

33 Ивашко Е.Е., Головин А.С. Вычислительная эффективность BOINC-GRID / Материалы Международная конференция «Высокопроизводительные вычисления» HPC-UA'2012 (Украина, Киев, 8-10 октября 2012 года) С187-193

34 Каляев А.И. Мультиагентная организация облачных вычислений на базе сети компьютеров частных пользователей / «Высокопроизводительные вычислительные системы» Труды молодых ученых ЮФУ и ЮНЦ РАН, Таганрог: Изд-во ЮФУ, 2012, С68-72

35 М.Н. Сатымбеков, И.Т.Пак, А.М. Мукышева, Көпагентті жүйені қолдану арқылы кластер тораптарының жүктемелерін оңтайландыру, Вестник КазАТК № 2 (101), 2017

36 M. R. Genesereth, “Software Agents,” Communications of the ACH, vol. 37, no. 7, 1994.

37 N. R. Jennings, K. Sycara, and M. Wooldridge, “A Roadmap of Agent Research and Development,” Autonomous Agents and Multi-Agent Systems, vol. 1, no. 1, pp. 275–306, 1998.

38 Shoham and Y, “Agent-oriented programming,” Artificial Intelligence, vol. 60, no. 1, pp. 51–92, 1993.

39 G. Weiss, Ed., Multiagent Systems:A Modern Approach to Distributed Modern Approach to Artificial Intelligence. MIT Press Cambridge, Massachusetts, 2000.

40 G. Weiß, “Adaptation and Learning in Multi-Agent Systems: Some Remarks and a Bibliography,” in IJCAI Workshop on Adaption and Learning in Multi-Agent Systems, 1995, pp. 1–21.

41 “FIPA.” [Online]. Available: http://www.fipa.org/. [Accessed: 31-Mar-2015].

42 F. Bellifemine, A. Poggi, and G. Rimassa, “JADE–A FIPA-compliant agent framework,” in Conference on the Practical Application of Intelligent Agents and Multi-agent Technology, 1999, pp. 97–108.

43 “FIPA.” [Online]. Available: http://www.fipa.org/. [Accessed: 31-Mar-2015].

44 F. Bellifemine, A. Poggi, and G. Rimassa, “JADE–A FIPA-compliant agent framework,” in Conference on the Practical Application of Intelligent Agents and Multi-agent Technology, 1999, pp. 97–108.

45 M. T. Kone, A. Shimazu, and T. Nakajima, “The State of the Art in Agent Communication,” Knowledge and Information Systems, vol. 2, no. 3, pp. 259–284, 2000.

46 B. Chaib-draa and F. Dignum, “Trends in Agent Communication Language,” Computational Intelligence, vol. 18, no. 2, pp. 89–101, May 2002.

47 T. Finin, R. Fritzson, D. McKay, and R. McEntire, “KQML as an Agent Communication Language,” in International Conference on Information and knowledge Management, 1994, pp. 456–463.

48 J. R. Searle, Speech Acts: An Essay in the Philosophy of Language. Cambridge: The University Press, 1969.

49B. Fabio, C. Giovanni, and G. Dominic, Developing Multi-Agent Systems with JADE. John Wiley & Sons, 2007.

50 A. Liekna, E. Lavendelis, and A. Grabovskis, “Experimental Analysis of Contract NET Protocol in Multi-Robot Task Allocation,” Applied Computer Systems, vol. 13, no. 1, pp. 6–14, 2012.

51 R. G. Smith, “The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver,” in IEEE Transactions on Computers, 1980, pp. 1104–1113.

52 R. Davis and R. G. Smith, “Negotiation as a Metaphor for Distributed Problem Solving,” Artificial Intelligence, vol. 20, no. 1, pp. 63–109, 1983.

53 C. Yu and T. N. Wong, “A multi-agent architecture for multi-product supplier selection in consideration of the synergy between products,” International Journal of Production Research, 2015.

54 A. More, S. Vij, and D. Mukhopadhyay, “Agent Based Negotiation using Cloud - an Approach in E-Commerce,” in ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol I, Springer International Publishing, 2014, pp. 489–496.

55 Wooldridge M. Agent-based software engineering // IEEE Proceedings Software Engineering. -1997. Vol.144. –P.26-37.

56 Wooldridge M., Jennings N. Intelligent Agents: Theory and Practice // Knowledge Engineering Review. -1995. –Vol.10. No.2 -P. 115-152.

57 Foner L. N. What’s An Agent, Anyway? A Sociological Case Study. - Cambridge, MA: MIT Media Lab, 1993. –167 p.

58 Brookes R. Intelligence without Representation // Artificial Intelligence. - 1991. -Vol.47. -P. 139-159.

59 Bellifemine F., Caire G., Greenwood D. Developing Multi-Agent Systems with JADE. –London:John Willey and Sons Ltd., 2007. -305 p.

60 Rao A., Georgeff M. BDI Agents: from Theory to Practice // Proceedings of the 1st International Conference on Multi-Agent Systems. – 1995. -P. 312-319.

61 Venkatesan V., Portchelvi V. Architecture for Services Orchestration using BDI Agent // URL: http://msdn.microsoft.com/en-us/library/bb898865.aspx. Accessed on: 11.11.2013.

62 Городецкий В.И. Современного состояния теории и практики // Computer Science E-Days. -2011.

63 M.N. Satymbekov*, I.T. Pak, L. Naizabayeva, and Ch.A. Nurzhanov Multi-agent grid system Agent-GRID with dynamic load balancing of cluster nodes, Open Engineering. 2017; 7:485–490.

64 V. Siládi, M. Povinský, L. Trajtel' and M. Satymbekov, "Adapted parallel quine-McCluskey algorithm using GPGPU," *2017 IEEE 14th International Scientific Conference on Informatics*, Poprad, 2017, pp. 327-331. doi:10.1109/INFORMATICS.2017.8327269.

65 M. N. Kalimoldayev, V. Siladi, M. N. Satymbekov and L. Naizabayeva, "Solving mean-shift clustering using MapReduce Hadoop," *2017 IEEE 14th International Scientific Conference on Informatics*, Poprad, 2017, pp. 164-167. doi:

10.1109/INFORMATICS.2017.8327240.

66A Yeleussinov, T Islamgozhayev, M Satymbekov and A Kozhagul, “CVCER: Robot to Learn Basics of Computer Vision and Cryptography”, 5th International Conference on Mechanics and Mechatronics Research (ICMMR 2018) ,417 (2018) 012013 doi:10.1088/1757-899X/417/1/012013.

67Ж.Н. Оразбеков, Ч.А. Нуржанов, М.Н. Сатымбеков, Ж.Б. Султанғазы, Г. Тлеубердиева, Корпоративтік портал өндірістік деректер ағынын өңдеу процесінің anylogic ортасында имитациялық модельденуі, Труды Университета ҚарМТУ *№1 (70) 2018*.

68 Шаяхметова А.С., Сатымбеков М., Анализ современного состояния рынка программных продуктов по байесовским сетям, новости науки казахстана Научно-технический журнал, № 2 (136), Алматы 2018

69 Naizabayeva L., Orazbekov ZH.N., Nurzhanov CH.A, M. N.Satymbekov, G. Turken, Distributed database for corporate information control system over enterprises network, №2 2018 Вестник КазНИТУ.

70Сатымбеков М.Н. Үлестірілген ортада агенттерді оқыту алгоритмі, Еуразия Ұлттық Университеті, ИНФОРМАТИЗАЦИЯ ОБЩЕСТВА, V международная. Научно-практическая конференция, Астана,2016.

71 Сатымбеков М.Н., Gaia технологиясын пайдаланып жаңалық тарату көпагентті жүйесін жобалау, Международная конференция Молодежь и Наука, Павлодар, 2015.

72 Сатымбеков М.Н., Нуржанов Ч.А., Клеточные автоматы, МАТЕРИАЛЫ научной конференции ИИВТ МОН РК «Современные проблемы информатики и вычислительных технологий» 29-30 июня 2017 года.

73Сатымбеков М.Н., Үлестірілген ортада агенттерді оқыту алгоритмі, Вестник КазАТК № 2 (97), 2016.

74 Brugnoli, M., Heymann, E., Senar, M.A., et al. "Grid scheduling based on collaborative random early detection strategies". 18th Euromicro Conf.

75Parallel,Distributed and Network-based Processing, Pisa, Italy, pp. 35–42 ,February 2010.https://doi.org/10.1109/PDP.2010.57

76Wu, J., Xu, X., Zhang, P.C., et al. "A novel multi-agent reinforcement learning approach for job scheduling in grid computing", Future Gener. Comput. Syst., 27, (5), pp. 430–439,2011.https://doi.org/10.1016/j.future.2010.10.009

ҚОСЫМША А.

Авторлық құқық объектісіне құқықтарды
мемлекеттік тіркеу туралы

КУӘЛІК

№ 2932 6 желтоқсан 2017 ж.

Қазақстан Республикасы Әділет министрлігінде авторлардың өтініші бойынша авторлары **Жасұлан Нұрлесұлы Оразбеков, Максатбек Нургалиулы Сатымбеков, Лязат Найзабаева** болып табылатын авторлық құқықпен қорғалатын объектіге айрықша мүліктік құқықтар **«Имитационная модель обмена и обработки данных специализированного корпоративного портала» (ЭЕМ-ге арналған бағдарлама)** атауымен тіркелгені куәландырылады.

Авторлардың өтініші бойынша авторлық құқықпен қорғалатын объектіге айрықша мүліктік құқықтар және **2017 жылғы 10 шілдеде** жасалған объект **Ж.Н. Оразбековке, М.Н. Сатымбековке, Л. Найзабаеваға** тиесілі және авторлар жоғарыда көрсетілген объектіні жасаған кезде басқа адамдардың зияткерлік меншік құқығы бұзылмағандығына кепілдік береді.

Тізілімде **2017 жылғы 6 желтоқсанда** жасалған № 2932 жазба бар.

Министрдің орынбасары Э. Әзімова

СВИДЕТЕЛЬСТВО

о государственной регистрации прав
на объект авторского права

№ 2932 6 декабря 2017 г.

Настоящим удостоверяется, что в Министерстве юстиции Республики Казахстан зарегистрированы исключительные имущественные права на объект авторского права под названием **«Имитационная модель обмена и обработки данных специализированного корпоративного портала» (программа для ЭВМ)**, авторами которого по заявлению авторов являются **Оразбеков Жасұлан Нұрлесұлы, Сатымбеков Максатбек Нургалиулы, Найзабаева Лязат.**

По заявлению авторов исключительные имущественные права на объект авторского права, созданный **10 июля 2017 года**, принадлежат **Оразбекову Ж.Н., Сатымбекову М.Н., Найзабаевой Л.** и авторы гарантируют, что при создании вышеуказанного объекта не были нарушены права интеллектуальной собственности других лиц.

Запись в реестре за № 2932 **от 6 декабря 2017 года** имеется.

Заместитель министра Э. Азимова

ИС 0537

ҚОСЫМША Ә.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ

РЕСПУБЛИКА КАЗАХСТАН

СВИДЕТЕЛЬСТВО

О ВНЕСЕНИИ СВЕДЕНИЙ В ГОСУДАРСТВЕННЫЙ РЕЕСТР ПРАВ НА ОБЪЕКТЫ, ОХРАНЯЕМЫЕ АВТОРСКИМ ПРАВОМ

№ 4198 от «24» июня 2019 года

Фамилия, имя, отчество, (если оно указано в документе, удостоверяющем личность) автора (ов):
ШАЯХМЕТОВА АСЕМ СЕРИКБАЕВНА, ТҰРДАЛЫҰЛЫ МҰСА , МАМЫРБАЕВ ОРКЕН ЖУМАЖАНОВИЧ, САТЫМБЕКОВ МАКСАТБЕК НУРГАЛИҰЛЫ, СЕЙСЕНБЕКОВА ПЕРИЗАТ БЕКБОЛАТҚЫЗЫ

Вид объекта авторского права: программа для ЭВМ

Название объекта: BayesClass

Дата создания объекта: 06.05.2019

Подписано ЭЦП

Оспанов Е.К.

ҚОСЫМША Б.

```
package agents.broker; import jade.core.Agent;
import jade.core.behaviours.*;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription; import
jade.domain.FIPAAgentManagement.ServiceDescription; import
jade.domain.FIPAException;
import jade.lang.acl.ACLMessage; import jade.lang.acl.MessageTemplate; import
models.Task;
import org.apache.log4j.Logger; import utils.Config;
import utils.Performative; import behaviours.myReceiver;

import java.io.IOException; import java.util.ArrayList; import java.util.Random;

public class Broker extends Agent {

public static Logger _log = Logger.getLogger(Broker.class); Random r = new
Random();
int bestPrice = Integer.MAX_VALUE; ACLMessage bestOffer = null;

protected void setup() {

final String convID = getLocalName() + hashCode() + System.currentTimeMillis()
% 10000 + "_" + r.nextInt(150);

DFAgentDescription dfd = new DFAgentDescription(); ServiceDescription sd = new
ServiceDescription(); sd.setType(Config.SERVICE_FREE); dfd.addServices(sd);

ACLMessage msg = new ACLMessage(Performative.SEARCH_PERFORMER);

msg.setConversationId(convID);
SequentialBehaviour seq = new SequentialBehaviour();
ParallelBehaviour par = new ParallelBehaviour(ParallelBehaviour.WHEN_ALL)
```

```
try {
DFAgentDescription[] result = DFService.search(this, dfd);
utils.Log.error(String.valueOf(result.length));
MessageTemplate template = MessageTemplate.and(
MessageTemplate.MatchPerformative(Performative.CAN_PERFORM),
MessageTemplate.MatchConversationId(convID)
);

for (int i = 0; i < result.length; i++) { msg.addReceiver(result[i].getName());
par.addSubBehaviour(new myReceiver(this, Config.WAIT_TIME, template) { public
void handle(ACLMessage msg) {
if (msg != null) { utils.Log.debug(msg.getSender().getLocalName()); int offer =
Integer.parseInt(msg.getContent());
if (offer < bestPrice) { bestPrice = offer; bestOffer = msg;
}
}
}
});
}
} catch (FIPAException ex) { ex.printStackTrace();
}

seq.addSubBehaviour(par); seq.addSubBehaviour(new OneShotBehaviour() {
public void action() {

System.out.println("best price " + bestPrice + " offered by " + bestOffer.getSender());
ACLMessage confirm = bestOffer.createReply(); Task task = new Task();
task.setChain(new ArrayList<Integer>()); task.setNumber(6);
try {
confirm.setContentObject(task);
confirm.setPerformative(Performative.DO_PERFORM);
```

```
} catch (IOException e) { e.printStackTrace();
}
}

});

MessageTemplate templateResult = MessageTemplate.and(
MessageTemplate.MatchPerformative(Performative.FINISH_PERFORM),
MessageTemplate.MatchConversationId(convID)
);

seq.addSubBehaviour(new myReceiver(this, Config.WAIT_TIME, templateResult)
{
public void handle(ACLMessage msg) { if (msg != null) {
utils.Log.error(msg.getContent());
}
}
});
addBehaviour(seq); send(msg);
}
}

package agents.performer;

import behaviours.Receiver; import  jade.core.Agent; import
jade.domain.DFService; import behaviours.Register; import behaviours.Receiver;
```

ҚОСЫМША В

```
import org.apache.log4j.Logger;

import java.util.ArrayList; import java.util.HashMap;

import static utils.Config.*;

public class Performer extends Agent {

public static Logger _log = Logger.getLogger(Performer.class); protected
void setup() {
Object[] args = getArguments();

myRoom = (Integer)args[0];
length = new HashMap<Integer, Integer>(); length =
(HashMap<Integer,Integer>)args[1];

chain = new ArrayList<Integer>(); if(args.length > 2)
chain = (ArrayList<Integer>)args[2]; chain.add(myRoom);
addBehaviour(new Register(SERVICE_FREE)); addBehaviour(new
Receiver());
}

protected void takeDown() { try {
DFService.deregister(this);
} catch (Exception e) { utils.Log.error(e.getLocalizedMessage());
}
}

}

package behaviours;
```

```
import jade.core.Agent;
import jade.core.behaviours.SimpleBehaviour;

public class DelayBehaviour extends SimpleBehaviour { private long
timeout, wakeupTime;
private boolean finished = false;

public DelayBehaviour(long timeout) { super();
this.timeout = timeout;
}

public void onStart() {
wakeupTime = System.currentTimeMillis() + timeout;
}

public void action() {
long dt = wakeupTime - System.currentTimeMillis(); if (dt <= 0) {
finished = true; handleElapsedTimeout();
} else
block(dt);
}

protected void handleElapsedTimeout() {
}

public boolean done() { return finished;
}
}

package behaviours;

import jade.core.behaviours.*; import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription; import
jade.domain.FIPAAgentManagement.ServiceDescription; import
jade.domain.FIPAException;
import jade.lang.acl.ACLMessage;
```

```
import jade.lang.acl.MessageTemplate; import jade.lang.acl.UnreadableException;
import models.Task;
import org.apache.log4j.Logger; import utils.Config;
import utils.Log;
import utils.Performative; import utils.Calc;
import static utils.Config.*;

import java.io.IOException; import java.util.HashMap; import java.util.HashSet;
import java.util.Random; import java.util.Set;

public class MapBehaviour extends OneShotBehaviour {

private static Logger _log = Logger.getLogger(MapBehaviour.class); private
ACLMessage message;
private Random r = new Random();
private Integer bestLength = Integer.MAX_VALUE; private Integer bestPrice =
Integer.MAX_VALUE; private ACLMessage bestOffer = null;
private ACLMessage reply; private Boolean found; private Boolean finish; private
Integer resNumber; private Task task;
private Set<Integer> blacklist = new HashSet<Integer>();

public MapBehaviour(ACLMessage message) { super();
try {
this.task = (Task)message.getContentObject();
} catch (UnreadableException e) {
e.printStackTrace(); //To change body of catch statement use File | Settings | File
Templates.
}
this.message = message;
}
```

```
@Override
public void action() {

myAgent.addBehaviour(new Register(SERVICE_BUSY)); found = false; finish =
false;
blacklist = new HashSet<Integer>();
final String convID = myAgent.getLocalName() + hashCode() +
System.currentTimeMillis() % 10000 + "_" + r.nextInt(150);

reply = message.createReply();
reply.setPerformative(Performative.FINISH_PERFORM);
reply.setConversationId(message.getConversationId());

try {
task = (Task)message.getContentObject();
} catch (UnreadableException e) {
e.printStackTrace(); //To change body of catch statement use File | Settings | File
Templates.
}

if(task.getNumber() == 0)
{
reply.setContent("1"); myAgent.send(reply); return;
}

final DFAgentDescription dfd = new DFAgentDescription(); ServiceDescription sd =
new ServiceDescription(); sd.setType(SERVICE_FREE);
dfd.addServices(sd);

final SequentialBehaviour seq = new SequentialBehaviour(); final ParallelBehaviour
par = new
ParallelBehaviour(ParallelBehaviour.WHEN_ALL);

OneShotBehaviour search = new OneShotBehaviour() {
@Override
public void action() {

utils.Log.debug("[" + myAgent.getLocalName() + "] Search started");
```

```
try {

ACLMessage msg = new ACLMessage(Performative.SEARCH_PERFORMER);
msg.setConversationId(convID);

DFAgentDescription[] result = DFService.search(myAgent, dfd);
final HashMap<String,Integer> map = new HashMap<String,Integer>();
utils.Log.debug("[" + myAgent.getLocalName() + "] result count is " +
result.length);
MessageTemplate template = MessageTemplate.and(
MessageTemplate.MatchPerformative(Performative.CAN_PERFORM),
MessageTemplate.MatchConversationId(convID)
);

finish = true;

for (int i = 0; i < result.length; i++) {

if(myAgent.getLocalName() == result[i].getName().getLocalName()) continue;

if(blacklist.contains(getRoom(myAgent, result[i].getName())))
{
continue;
}

finish = false; msg.addReceiver(result[i].getName());
                par.addSubBehaviour(new myReceiver(myAgent,
template) {    Config.WAIT_TIME,

                public void handle(ACLMessage msg) { if (msg != null) {

                utils.Log.error(getRoom(myAgent, msg).toString());
```

```
int price = Integer.parseInt(msg.getContent()); int length = Integer.MAX_VALUE;
try {
length = Config.length.get(getRoom(myAgent, msg));
} catch (Exception e) { utils.Log.error(e.getLocalizedMessage());
}

utils.Log.debug("[" + myAgent.getLocalName() + "] length from "
+ msg.getSender().getLocalName() + " is " + length);

if (length < bestLength) { bestLength = length; bestPrice = price; bestOffer = msg;
}
else if (length == bestLength) { if(price < bestPrice)
{
                                bestLength = length; bestPrice = price; bestOffer = msg;


                            }
                            }
                            }
                            }
                        });
                }

myAgent.send(msg);

} catch (FIPAException ex) { ex.printStackTrace();
}
}
};

seq.addSubBehaviour(search); seq.addSubBehaviour(par);

seq.addSubBehaviour(new OneShotBehaviour() {
```

```
public void action() { try {
resNumber = 1;
ACLMessage confirm = bestOffer.createReply(); Task nextTask = new Task();
nextTask.setChain(chain); nextTask.setNumber(task.getNumber() - 1);

confirm.setContentObject(nextTask);
confirm.setPerformative(Performative.DO_PERFORM); myAgent.send(confirm);

} catch (IOException e) { utils.Log.error(e.getLocalizedMessage());
}
}

});

MessageTemplate templateResult = MessageTemplate.and(
MessageTemplate.MatchPerformative(Performative.FINISH_PERFORM),
MessageTemplate.MatchConversationId(convID)
);

seq.addSubBehaviour(new myReceiver(myAgent, Config.WAIT_TIME,
templateResult) {
public void handle(ACLMessage msg) { if (msg != null) {

resNumber = Integer.valueOf(msg.getContent()) * task.getNumber(); found = true;
finish = true;

}
}
});

SequentialBehaviour seq2 = new SequentialBehaviour(); SimpleBehaviour cycle =
new SimpleBehaviour() {
@Override
```

```
public void action() {

SequentialBehaviour seq2 = new SequentialBehaviour();

seq2.addSubBehaviour(seq); seq2.addSubBehaviour(new
OneShotBehaviour() {
@Override
public void action() { if(!finish && !found)
{
blacklist.add(getRoom(myAgent, bestOffer));
}
}
});

myAgent.addBehaviour(seq2);

}

@Override
public boolean done() { return finish;
}
};

seq2.addSubBehaviour(cycle);

seq2.addSubBehaviour(new OneShotBehaviour() {
@Override
public void action() { if(found)
{
reply.setContent(String.valueOf(resNumber));
}
else
{
reply.setContent("-1");
}
myAgent.send(reply);
}
});
```

```
seq2.addSubBehaviour(new OneShotBehaviour() {
@Override
public void action() {
myAgent.addBehaviour(new Register(SERVICE_FREE));
}
});

myAgent.addBehaviour(seq2);
}
}

package behaviours;

import jade.core.Agent;
import jade.core.behaviours.SimpleBehaviour; import
jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
public class myReceiver extends SimpleBehaviour { private
MessageTemplate template;
private long timeOut, wakeupTime;
private boolean finished; private ACLMessage msg;

public ACLMessage getMessage() { return msg;
}

public myReceiver(Agent a, int millis, MessageTemplate mt) { super(a);
timeOut = millis; template = mt;
}

public void onStart() {
                wakeupTime = (timeOut < 0 ? Long.MAX_VALUE
                        : System.currentTimeMillis() + timeOut);
}

public boolean done() { return finished;
```

```
}

public void action() { if (template == null)
msg = myAgent.receive(); else
msg = myAgent.receive(template);

if (msg != null) { handle(msg); finished = true; return;
}
long dt = wakeupTime - System.currentTimeMillis(); if (dt > 0)
block(dt); else {
finished = true; handle(msg);
}
}

public void handle(ACLMessage m) { /* can be redefined in sub_class */ }

public void reset() { msg = null; finished = false; super.reset();
}

public void reset(int dt) { timeOut = dt;
reset();
}
}

package behaviours;

import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
```

ҚОСЫМША Г.

```
import jade.lang.acl.ACLMessage; import jade.lang.acl.MessageTemplate;
import org.apache.log4j.Logger;
import utils.Performative;
public class Receiver extends CyclicBehaviour { MessageTemplate mt1 =
MessageTemplate.MatchPerformative(Performative.SEARCH_PERFORMER);
ACLMessage msg;
public static Logger _log = Logger.getLogger(Receiver.class.getName());

@Override
public void action() {

msg = myAgent.receive(mt1); if(msg != null)
{
utils.Log.debug("[" + myAgent.getLocalName() + "] " + "Need help to " +
msg.getSender().getLocalName());
myAgent.addBehaviour(new Map(myAgent, msg));
}
}

}
package behaviours; import jade.core.Agent;
import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.ParallelBehaviour; import
jade.core.behaviours.SequentialBehaviour; import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription; import
jade.domain.FIPAAgentManagement.ServiceDescription; import
jade.domain.FIPAException;
import jade.lang.acl.ACLMessage; import jade.lang.acl.MessageTemplate;
import utils.Config;
import utils.Performative;
```

```
import utils.Calc;

import java.util.Random; import java.util.logging.Level; import
java.util.logging.Logger;

public class Reducer extends OneShotBehaviour {

public static Logger log = Logger.getLogger(Reducer.class.getName()); Random r =
new Random();
int bestPrice = Integer.MAX_VALUE; ACLMessage bestOffer = null; Integer
mustCalc = null; ACLMessage caller = null;

public Reducer(Agent a, Integer mustCalc, ACLMessage caller) { super(a);
this.mustCalc = mustCalc; this.caller = caller;
}

@Override
public void action() {

final String convID = myAgent.getLocalName() + hashCode() +
System.currentTimeMillis() % 10000 + "_" + r.nextInt(150);

DFAgentDescription dfd = new DFAgentDescription(); ServiceDescription sd = new
ServiceDescription(); sd.setType("map");
dfd.addServices(sd);

ACLMessage msg = new ACLMessage(Performative.SEARCH_PERFORMER);
msg.setConversationId(convID);
SequentialBehaviour seq = new SequentialBehaviour();
ParallelBehaviour par = new ParallelBehaviour(ParallelBehaviour.WHEN_ALL);

try {
DFAgentDescription[] result = DFService.search(myAgent, dfd);
log.log(Level.WARNING, String.valueOf(result.length));
```

```
MessageTemplate template = MessageTemplate.and(
MessageTemplate.MatchPerformative(Performative.CAN_PERFORM),
MessageTemplate.MatchConversationId(convID)
);

for (int i = 0; i < result.length; i++) { msg.addReceiver(result[i].getName());
par.addSubBehaviour(new myReceiver(myAgent, Config.WAIT_TIME *
Integer.parseInt(msg.getContent()) * Integer.parseInt(msg.getContent()), template)
{
public void handle(ACLMessage msg) { if (msg != null) {
int offer = Integer.parseInt(msg.getContent()); if (offer < bestPrice) {
bestPrice = offer; bestOffer = msg;
}
}
}
});
}
} catch (FIPAException ex) { ex.printStackTrace();
}
seq.addSubBehaviour(par); if(bestOffer == null)
{
Calc.fibonacci(mustCalc);
}
else
{
seq.addSubBehaviour(new OneShotBehaviour() {

public void action() {
System.out.println("best price " + bestPrice + " offered by" + bestOffer.getSender());
ACLMessage confirm = bestOffer.createReply();
confirm.setContent(String.valueOf(mustCalc));
```

```
confirm.setPerformative(Performative.DO_PERFORM); myAgent.send(confirm);
}

});
}

MessageTemplate templateResult = MessageTemplate.and(
MessageTemplate.MatchPerformative(Performative.FINISH_PERFORM),
MessageTemplate.MatchConversationId(convID)
);

seq.addSubBehaviour(new myReceiver(myAgent, Config.WAIT_TIME *
Integer.parseInt(msg.getContent()) * Integer.parseInt(msg.getContent()),
templateResult) {
public void handle(ACLMessage msg) { if (msg != null) {
System.out.println(msg.getContent());
}
}
});

myAgent.addBehaviour(seq); myAgent.send(msg);
}
}

package behaviours;

import jade.core.behaviours.Behaviour; import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription; import
jade.domain.FIPAAgentManagement.Property;
import jade.domain.FIPAAgentManagement.ServiceDescription; import
jade.domain.FIPAException;
import org.apache.log4j.Logger; import static utils.Config.*;
public class Register extends Behaviour {
```

```
public static Logger _log = Logger.getLogger(Register.class); private String
service;
public Register(String service) { super();
this.service = service;
}

public void action() {
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(myAgent.getAID());
ServiceDescription sd = new ServiceDescription(); sd.setType(service);
sd.setName(service + "-" + myAgent.getLocalName()); sd.addProperties(new
Property(PROP_ROOM, myRoom)); dfd.addServices(sd);

try {
DFService.register(myAgent, dfd);
} catch (FIPAException fe) { try {
DFService.modify(myAgent, dfd);
} catch (FIPAException e) {
e.printStackTrace(); //To change body of catch statement use File | Settings | File
Templates.
}
} finally {
utils.Log.debug("[" + myAgent.getLocalName() + "]" + " Registered as " +
service);
}
}

public boolean done() { return true;
}
}

package models;

import java.io.Serializable;
```

ҚОСЫМША Ғ.

```
import java.util.ArrayList;
public class Task implements Serializable { private Integer number;
private ArrayList<Integer> chain;

public Integer getNumber() { return number;
}

public void setNumber(Integer number) { this.number = number;
}

public ArrayList<Integer> getChain() { return chain;
}

public void setChain(ArrayList<Integer> chain) { this.chain = chain;
}
}

package utils;

import jade.core.AID; import jade.core.Agent;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription; import
jade.domain.FIPAAgentManagement.Property;
import jade.domain.FIPAAgentManagement.ServiceDescription; import
jade.domain.FIPAException;
import jade.lang.acl.ACLMessage; import jade.util.leap.Iterator;

import java.util.ArrayList; import java.util.HashMap;

public class Config {

public static Integer WAIT_TIME = -1;
```

```
public static HashMap<Integer,Integer> length; public static Integer myRoom;
public static ArrayList<Integer> chain; public static String PROP_ROOM =
"room";
public static String SERVICE_FREE = "free"; public static String
SERVICE_BUSY = "busy";

public static Integer getRoom(Agent myAgent, AID sender)
{
final DFAgentDescription dfd = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType(SERVICE_FREE);
dfd.addServices(sd); Integer room = null;
DFAgentDescription[] res = new DFAgentDescription[0]; try {
res = DFService.search(myAgent, dfd); ServiceDescription sdd;
Property property; for(DFAgentDescription desc : res)
{

Iterator it1 = desc.getAllServices(); sdd = (ServiceDescription)(it1.next());
Iterator it2 = sdd.getAllProperties(); property = (Property)(it2.next());

if(desc.getName().getLocalName().equals(sender.getLocalName()))
{
room = Integer.valueOf((String)property.getValue());
utils.Log.error(desc.getName().getLocalName() + " is equal to " +
sender.getLocalName());
utils.Log.error(desc.getName().getLocalName() + " is = " +
(Integer.valueOf((String)property.getValue())));
}
else
{
```

```
utils.Log.error(desc.getName().getLocalName() + " is not equal to " +
sender.getLocalName());
}
}
if(room == null)
utils.Log.error("Error with searching " + sender.getLocalName());
} catch (FIPAException e) {
e.printStackTrace(); //To change body of catch statement use File | Settings | File
Templates.
}
return room;
}

public static Integer getRoom(Agent myAgent, ACLMessage message)
{
return getRoom(myAgent, message.getSender());
}

}

package utils;

import java.util.logging.Level; import java.util.logging.Logger;

public class Log {
public static Logger logger = Logger.getLogger(Log.class.getName()); public static
void error(String s)
{
logger.log(Level.WARNING, s);
}

public static void debug(String s)
{
logger.log(Level.INFO, s);
}

}

package utils;
```

ҚОСЫМША Д.

```
import jade.lang.acl.MessageTemplate; public class Performative {
public static int SEARCH_PERFORMER = 1; public static int
CAN_PERFORM = 2; public static int DO_PERFORM = 3;
public static int FINISH_PERFORM = 4;

public static MessageTemplate template(int perf, String convID)
{
return MessageTemplate.and( MessageTemplate.MatchPerformative(perf),
MessageTemplate.MatchConversationId(convID)
);
}
}

package behaviours;

import jade.core.Agent;
import jade.core.behaviours.SequentialBehaviour; import
jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate; import
jade.lang.acl.UnreadableException; import models.Task;
import org.apache.log4j.Logger; import utils.Config;
import utils.Performative; import java.util.Random; import static
utils.Config.*;

public class Map extends SequentialBehaviour {
public static Logger _log = Logger.getLogger(Map.class); Random r = new
Random();
ACLMessage msg; String convID = null;

public Map(Agent a, ACLMessage m) {
```

```
super(a); msg = m;
convID = m.getConversationId();
}

public void onStart() {

addSubBehaviour(new DelayBehaviour(r.nextInt(2000)) {

        public void handleElapsedTimeout() { ACLMessage reply =
        msg.createReply();
        reply.setPerformative(Performative.CAN_PERFORM); Integer price =
        r.nextInt(2500);
        utils.Log.debug("[" + myAgent.getLocalName() + "] " + "My price is " +
price);
        reply.setContent(Integer.toString(price)); myAgent.send(reply);
}

});

MessageTemplate tmjob = MessageTemplate.and(
MessageTemplate.MatchPerformative(Performative.DO_PERFORM),
MessageTemplate.MatchConversationId(convID)
);
addSubBehaviour(new myReceiver(myAgent, Config.WAIT_TIME, tmjob) {
public void handle(ACLMessage msg) {
if (msg != null) { try {
Task task = (Task)msg.getContentObject();

utils.Log.debug("[" + myAgent.getLocalName() + "] " +
msg.getSender().getLocalName() + " send me task with number " +
task.getNumber());

addSubBehaviour(new MapBehaviour(msg));

} catch (UnreadableException e) { utils.Log.error(e.getLocalizedMessage());
}
```

```
}
}
});
}
}

# Root logger option log4j.rootLogger=INFO, stdout

# Direct log messages to stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %- 5p
%c{1}:%L - %m%
```