Cild 19 № 4 2019

UOT 681.51

UNİFİKASİYALI MODELLƏŞDİRMƏ DİLİNİN TƏTBİQİ İLƏ MULTİAGENT SİSTEMLƏRİN LAYİHƏLƏNDİRİLMƏSİNİN BƏZİ ASPEKTLƏRİ

SƏLİMOVA MEHRİBAN RƏŞİD qızı
Sumqayıt Dövlət Universiteti, assistent
mehriban_mr@mail.ru

***Açar sözlər*:** *unifikasiyalı modelləşdirmə dili, agent, multiagent sistem, arxitektura, reaktiv agent, avtonom agent.*

Proqram agentləri (PA) texnologiyaları son illərdən süni intellekt və informasiya texnologiyaları sahəsinin ən vacib və aktual konsepsiyalarından birini təmsil edir ki, bu da mürəkkəb kompüter sistemlərinin konseptualizasiya və qurulma üsullarını əsaslı olaraq dəyişir.

Multiagent sistemlərin əsaslı tətbiqi qərar qəbulunu dəstəkləyən informasiya sistemlərində, əsasən aktual və etibarlı informasiya təqdim etməyi bacaran mürəkkəb funksional mühit olan idarəetmə sistemlərində ifadə olunur.

Hal-hazırda elə sosial-iqtisadi sistemlər mövcuddur ki, onların multiagent texnologiyaların (MAT) köməyi ilə layihələndirilməsi aktual məsələdir, belə ki, çoxlu sayda müasir sistemlərin və təşkilatların mürəkkəbliyi elə səviyyəyə çatır ki, onlarda nəhəng informasiya axınının mövcudluğu mərkəzləşdirilmiş idarəetməni səmərəsiz edir. Bu isə agent texnologiyasının tətbiqini aktuallaşdırır.

Agent müəyyən mühitlərdə fəaliyət göstərən və öz məqsədlərinə uyğun olaraq avtonom davrana bilən hesablama sistemidir. Burada avtonomluq dedikdə, adətən, agentin insanların (və ya başqa agentlərin) birbaşa müdaxiləsi olmadan öz fəaliyyətinə və daxili vəziyyətinə nəzarətetmə qabiliyyətinin olması başa düşülür. Agentlərdən istifadə edən proqramlaşdırma sahəsindəki avtonomluq anlayışı ilə obyektyönlü proqramlaşdırmadakı inkapsulasiya arasında müəyyən oxşarlıq vardır [1].Obyekt bəzi vəziyyətləri inkapsulyasiya edir, yəni elə nəzarət edir ki, ona giriş yalnız özünün təqdim etdiyi metodlarla mümkün olur. Bundan fərqli olaraq, agent yalnız müəyyən vəziyyəti deyil, davranışı da inkapsullaşdırır. Agent-çevik davranışlı obyektdir. Yəni obyekt aşağıdakılara malik olmalıdır [2]:

1. Yuxarıda deyilən mənada avtonomluq;
2. Həssaslıq, yəni öz mühitini öz sensorları ilə qəbul etmək və buradakı dəyişikliklərə vaxtında cavab vermək;
3. Fəallıq, yəni öz mühitindəki hərəkətlərə sadəcə təsir etmək deyil, hadisələri qabaqcadan görə bilmək, qabaqlayıcı və məqsədyönlü hərəkət etmək, lazım gəldikdə, təşəbbüsü ələ almaq;
4. Sosiallıq – öz tapşırığını yerinə yetirmək üçün digər agent və insanlarla ünsiyyət qurmaq və partnyora öz tapşırığını yerinə yetirməkdə kömək etmək.

Multiagent sistemlər (MAS) – bir-birləri ilə qarşılıqlı fəaliyyət göstərə bilən agentlər çoxluğundan ibarət olan sistemdir.

Agentyönlü sistemlərin (AYS) qurulmasında iki yanaşmanı fərqləndirmək olar: vahid avtonom agentin realizasiyası və ya MAS-ın işlənməsi.

Avtonom agent yalnız istifadəçi ilə qarşılıqlı fəaliyyət göstərir və agentyönlü proqramlar çərçivəsində lazım olan bütün funksional imkanları həyata keçirir. Buna qarşı MAS – biliklərə əsaslanan və müəyyən mühitlərdə fəaliyyət qabiliyyətinə malik olan, eyni zamanda bir-birilə müəyyən münasibətlərdə və qarşılıqlı təsirdə olan, müəyyən tapşırıqları formalaşdıran, fərdi və birgə fəaliyyətlər toplusuna malik olan ayrı-ayrı altsistemlərin (agentlərin) birləşməsi kimi qurulur.

MAS-ın işlənmə prosesi çətin və zəhmətli prosesdir, belə ki, layihəçi üçün dəqiq formalaşdırılmalı və bəzi müəyyən standartlara əsaslanmalıdır [3].

AYS-in layihələndirilməsinin metodologiyası hələ ilkin inkişaf mərhələsindədir [3-7]. Məlum yanaşmaları 4 əsas sinfə bölmək olar [4]:

-obyektyönlü proqramlaşdırma (OYP) metodları və uyğun genişlənmələrdən istifadə edən texnologiyaya əsaslananlar;

-mühəndis biliklərin ənənəvi metodlarından istifadə edənlər;

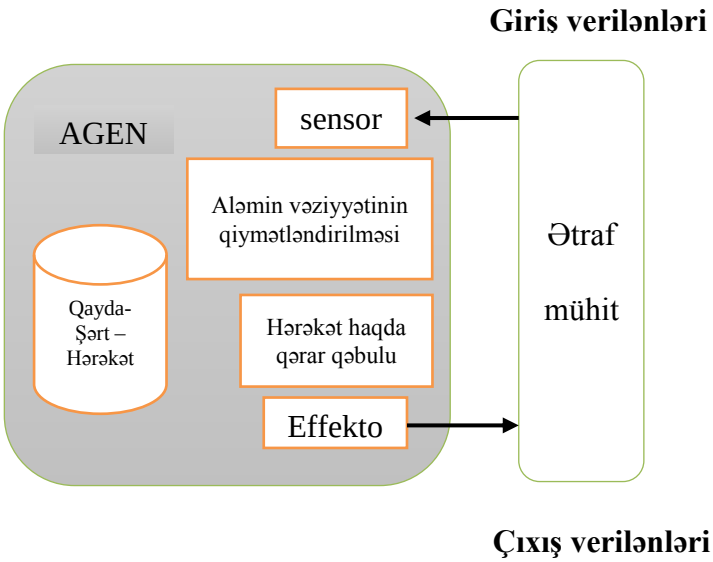
-təşkilatyönlü məlumatlara əsaslananlar;

- bu üç sinfin metodlarını müxtəlif dərəcədə kombinə edənlər.

Bu məqalədə obyektyönlü analiz və unifikasiyalı modelləşdirmə dilindən (UML) istifadə etməklə, agentlərin layihələndirilməsi prosesinin bəzi aspektlərinə baxırıq.

Proqram agentlərində irəli sürülmüş tələbatlar toplusu ilə OYP-dəkı xüsusyyətlətləri müqayisə etdikdə aydın olur ki, OYP texnologiyası MAS-ların proqram realizasiyası üçün daha əlverişlidir. Belə yanaşmadan agentlər arasında qarşılıqlı fəaliyyət üçün mesaj mübadiləsində, eyni zamanda toplanma və miras prinsiplərində istifadə olunur.

Agenti UML sinfi və metodları terminləri ilə təsvir etməzdən öncə informasiyalı ətraf mühitlə qarşılıqlı fəaliyyətdə olan sadə agentin arxitekturasına baxaq (şəkil 1). Burada ətraf mühit kimi İnternet, korporativ şəbəkə və ya virtual xüsusi şəbəkə (VPN) ola bilər.



***Şəkil 1**. Sadə agentin arxitekturası.*

Şəkildən göründüyü kimi, agent ətraf mühiti öz sensorundan qəbul edir, sonra qəbuletmə əsasında öz effektorları ilə hərəkəti seçir və yerinə yetirir. Sensorlu giriş mesajları şəbəkəyə ötürə bilər [5].

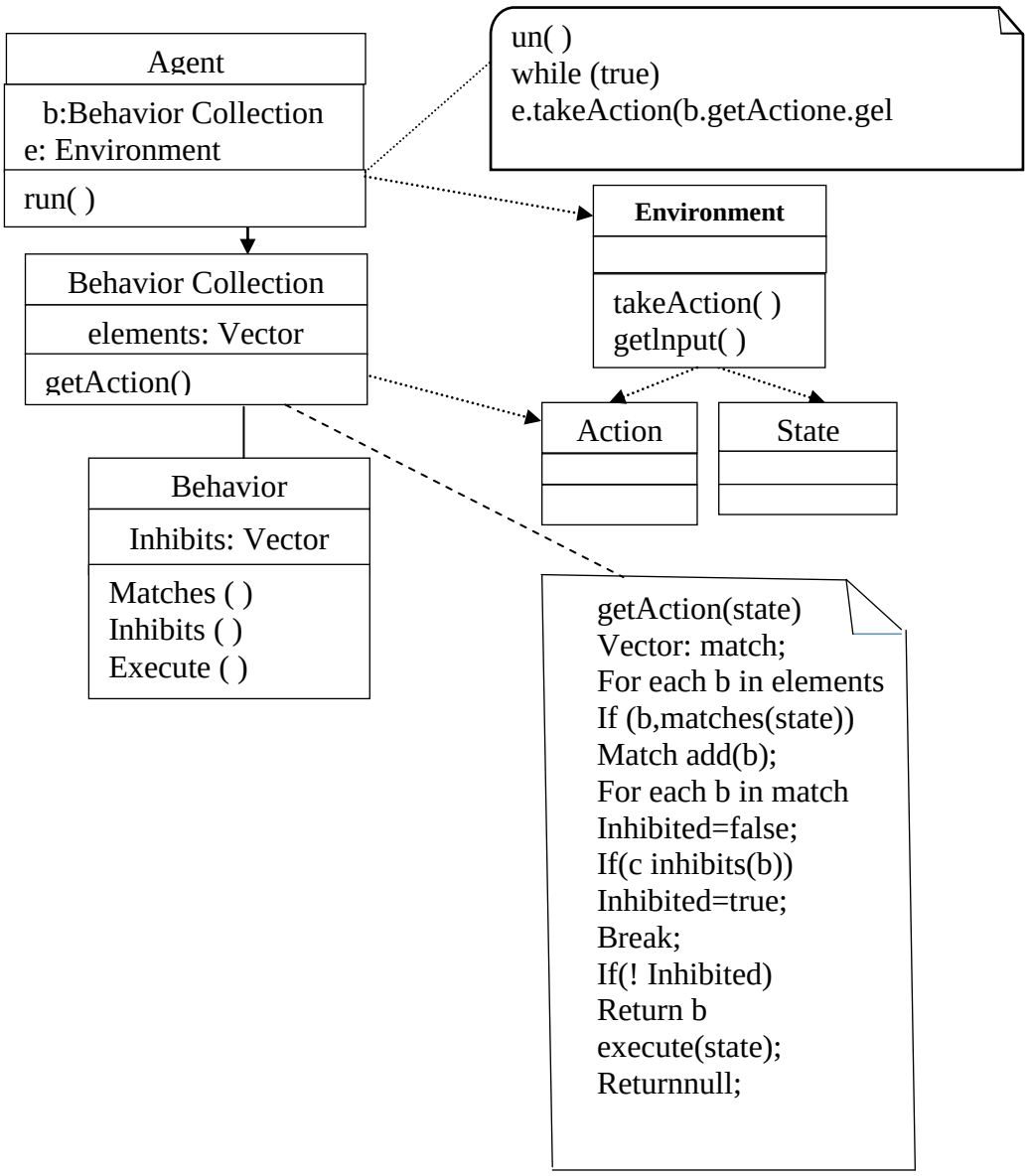
Agentlərin proqramlaşdırılması üçün OYP-nin müxtəlif dillərindən istifadə etmək olar. Lakin daha çox JAVA dilindən istifadə olunur. Burada agent-obyektdir və bu xüsusiyyətinə görə bir sıra fərqliliyə malikdir. Aktiv olmaq üçün agent Java dilinin istənilən obyekti kimi daxili hadisələr ardıcıllığına malik obyekt olmalıdır.

Mühitin analizi nəticəsində Java dilində agentin yaranan tipik hadisələr ardıcıllığının tsiklinin proqram kodu aşağıdakı kimidir:

```
Environment e; // mühitin vəziyyəti
RuleSet r; //qaydalar yığını
while (true) {  state = senseEnvironment(e); //mühiti qiymətləndirmək
```

```
a = chooseAction(state, r); //hərəkəti seçmək
e.applyAction(a); //hərəkəti yerinə yetirmək  }
```

Bu dövri proses sonsuzdur və agentin davamlılığını təmin edir.



Şəkil 2. *UML diaqramlar sinfi şəklində reaktiv agentin arxitekturası*

Obyekt şəklində realizə olunan agent üçün avtonomluq obyektin bütün metodlarının qapalı elan olunması yolu ilə həyata keçirilir. Onda agent yalnız özü öz metodlarını çağıra bilər və başqa heç bir agent onu məqsədlərindən uzaq heç bir hərəkət etməyə məcbur edə bilməz. Digər agentlər belə agentlə mühitdə hadisələri və mesajları dolayı generasiyaetmə yolu ilə əlaqə yaradır ki, həmin agent bu ünsiyyəti qəbul edə və cavab verə bilər.

Agentə digər agentlərlə məsləhətləşmə imkanını ünsiyyət yaradır. Mesajların ötürülməsi və qəbulu yolu ilə həyata keçirilən məsləhətlər agentə öz fəaliyyəti ilə əməkdaşlığı uzlaşdırmağa imkan verir [6].

İşdə əsasən agentlərin başa düşülməsini və yaradılmasını asanlaşdıran UML sinfinin diaqramlarının köməyi ilə tipik agent arxitekturalarının (reaktiv və BDİ-arxitektura)

layihələndirilməsinə baxılır. Verilən nümunələr agent arxitekturasının hər bir funksional aspektini əks etdirmir, yalnız tipik agent arxitekturalarının realizasiyasının ümumi sxemini verir [8].

Qeyd edək ki, reaktiv agentlər – mühitin vəziyyəti haqda informasiyanı saxlamır, hədəfqurma bacarığı yoxdur, sadəcə cari qəbuletməyə "ƏGƏR-ONDA" qaydalar tipinə uyğun olaraq cavab verir. Agentin reaktiv arxitekturalar sinfinin diaqramı şəkil 2-də göstərilmişdir.

Burada Agent sinfi (klass) – agenti, Behavior Collection sinfi - agentin davranış metodları kolleksiyasını, Behavior sinfi – agentin davranışlar metodunu, Action sinfi –agentin fəaliyyəti və nəhayət, Environment və Stare sinifləri –agenti əhatə edən mühit və ona uyğun vəziyyətləri təmsil edir.

Agent sinfinin run( ) –metodu agentin cari davranış metodları və ətraf mühitin vəziyyətləri ilə təyin olunan hərəkətləri yerinə yetirir. OYP-də obyektlər kolleksiyası getAction() hərəkət seçimi funksiyasının proqram kodunu dəyişmədən davranış metodlarını əlavə etməyə və silməyə imkan verir, bu zaman davranış metodları siyahısına baxmaq üçün interfeysdən istifadə etmək olar.

Hər bir davranış metodu ətraf mühitə uyğun olduqda həyəcanlanır və onların hər biri digər davranış metodlarını bloklaya bilər. Burada getAction() funksiyası O(n) kimi ölçülmüş icra müddətinə malik olduğundan elə də effektli deyildir. Burada n- davranış metodlarının sayıdır. Bu funksiyanın icra müddətini həllər ağacından istifadə etməklə $O(\log_n)$-ə qədər və ya aparat vasitələrindən istifadə etməklə, O(1)-ə qədər azaltmaq olar. Layihəçi təminat verməlidir ki, ətraf mühitin hər vəziyyətinə ən azı bir davranış metodu uyğun gəlir. Bunun üçün bütün giriş siqnallarına uyğun gələn, lakin mühitin digər vəziyyətləri üçün davranışlar metodları tərəfindən bloklanan davranışlar metodunu təyin etmək olar.

BDİ-agent arxitekturası tələb-arzu-inam (belief-desire-intention, BDİ) modeli əsasında qurulmuşdur. Qeyd edək ki, arzu, tələb və inam uyğun olaraq onun vəziyəytini, məqsəd və planlarını şərh edə bilər. İnam özündə şübhəni (inanıram ki, A tapşırığını yerinə yetirə bilərəm), digər agentlərin imkanlarına inamı (Mən inanıram ki, B agenti "C" tapşırığını yerinə yetirə bilər) və ətraf mühit haqqında təsəvvürləri (mənim sensorlarım əsasında güman edirəm ki, mən divarın 3 m yaxınlığındayam) saxlayır. Tələb o vaxta kimi qorunur ki, onlar yerinə yetirilmir və ya onların qurulması mümkünsüz olsun.

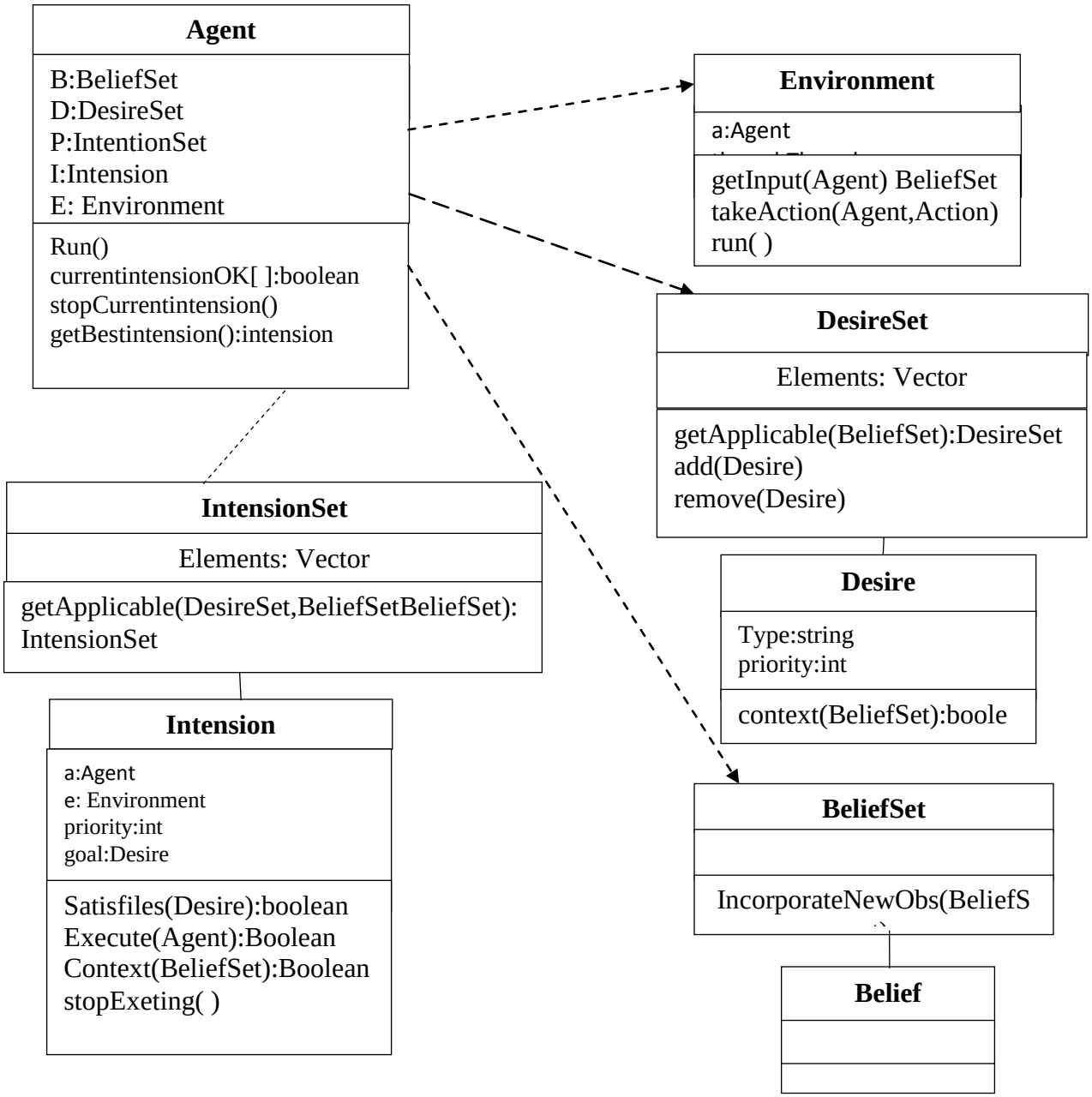
BDİ-agentlərin arxitekturası UML siniflərinin terminləri ilə şəkil 3-də göstərilmişdir. Burada Beliesef, Desireset və İntentionSet – inam, arzu və tələb toplusunu təqdim edir. Belief, Desire və İntention sinifləri isə - uyğun olaraq inam, arzu və tələbdir. Bu arxitekturanı əvvəlki çoxhəlli metodun yerinə istifadə edirlər. Bununla da yerinə yetirilən sistem mövcud tələbin tətbiq olunub-olunmadığını kəsilməz dövri olaraq yoxlayır. Əgər o tətbiq olunmursa, agent stopCurrentİntention() metodunu yerinə yetirir və bu metod da tələb sinfinin stopExecuting() metodunu çağırır.

Beləliklə, tılıb öz işinin kəsilməsi və yaddaşın təmizlənməsinə cavab verir.

Hər tələbə bu bacarığını verməklə, biz təəbin öz işini dayandırdığı anda nəzərdə tutulmuş ehtiyatların resurslarının yaranması nəticəsində meydana gələn çıxılmaz vəziyyətlər (deadlock) aradan qaldırır.

Növbəti proqram kodu BDİ-agent proqramının iki əsas dğvrünü izah edir: A, B, D və İ dəyişənləri uyğun olaraq agenti, onun inam, arzu və tələblərini təmsil edir:

```
Agent::run(){
Environment e; e.run(); //ətraf mühitin obyektinin işə salınması
while(true) {
I = a.getBestIntention();
if (I.execute(a)) //əgər tələb yerinə yetirilibsə
aşDremove(I.goal); //arzunu aradan qaldırmaq
}
}
```



Şəkil 3. *UML diaqramlar sinfi şəklində agentin BDİ-arxitekturası*

Agentin Run () metodu ən yaxşı tətbiq oluna bilən tələbi seçir və yerinə yetirir. Əgər tələb yerinə yetirmə funksiyası true qiymətini qaytarırsa, onda arzu əldə edilmiş olur və arzular toplusundan silinir. Əgər aşkar edilsə ki, daha tətbiqə yaramır, onda tələb dərhal öz execute() metodunun çağırışından çıxır.

Yuxarıda OYP dilindən istifadə etməklə agent arxitekturalarına dair ümumi göstərişlər verilmişdir.

UML dilinin istifadəsi agent arxitekturasının daha yaxşı başa düşülməsi, onların layihələndirilməsinin və realizasiyasının əlverişliliyi ilə əsaslandırılır.

Ümumiyyətlə, agent sinfi diaqramlarının UML-də qurulması onların oynadıqları funksional təyinatı, onların arasındakı qarşılıqlı fəaliyyəti və agentlərarası asılılıq imkanlarını nəzərdə tutur.

Agentlərarası qarşılıqlı əlaqə və asılılıq yalnız tam MAS qurulduqda mümkün olur ki, belə MAS mesajların nəqlini təşkil edən infrastruktur kataloq xidmətini və hadisələrin məlumatlandırılmasını təmin edir.

Agentlərin MAS-da qarşılıqlı əlaqəsi "agent-kliyent" mesajlarının qəbulu, ötürülməsi və bu mesajların "agent-server" ilə emalı vasitəsilə baş verir. Eyni zamanda müxtəlif vəziyyətlərdə eyni agentlər həm kliyent, həm server kimi çıxış edə bilərlər. Agentlərin qarşılıqlı hərəkət protokollarının təsviri ardıcıllıq (sequence diaqram) diaqramının qurulması yolu ilə yerinə yetirilə bilər. Bu tip diaqram agentlər arasında məlumatların ötürülməsi ardıcıllığını əks etdirməyə imkan verir.

Agentlərin davranışlarının spesifikasiyasının UML dilində təsviri üçün vəziyyətlər diaqramı və fəaliyyət diaqramından istifadə etmək əlverişlidir.

Müəyyən davranışa malik hər MAS agenti öz davranış ssenarisini həyata keçirərkən müəyyən hərəkətlər yerinə yetirərək müxtəlif vəziyyətdə ola bilər, yəni bir vəziyyətdən digərinə keçə bilər.

Real sistemlərdəki bir çox agentlərin davranışlarını sonlu avtomatlar nəzəriyyəsi baxımından təsvir etmək olar, belə ki, agentin davranışı onun vəziyyətlərində (fəaliyyət rejimlərində) əks olunur və vəziyyətlər diaqramı (statechart diagram) onu qrafik təsvir etməyə imkan verir.

Hərəkət diaqramı vəziyyətlər diaqramının sonrakı inkişafını nəzərdə tutur. Bu tip diaqramdan həm də modelləşdirilən obyektin vəziyyətlərini əks etdirmək üçün də istifadə edilə bilər, lakin bu diaqramın əsas təyinatı obyektin fəaliyyət prosesinin vəziyyətini əks etdirməkdən ibarətdir. Qeyd olunan diaqramlar proseslərin yalnız ardıcıllığını deyil, həm də budaqlanmasını və hətta sinxronlaşmasını göstərə bilir.

MAS-ın layihələndirilməsi üçün baxılan yanaşma, UML universal modelləşdirmə dilindən istifadəsinə, layihəçiyə mövcud standartlara əsaslanan MAS-ların çevik başa düşülməsinə imkan verir [6].

Ümumi halda, UML dilinin köməyi ilə MAS-in işlənməsi və reallaşdırılmasında sistemin ümumi xarakteristikasını, agentlərin və məqsədlərin strukturunu, eyni zamanda agentlərin qarşılıqlı əlaqə və davranış protokollarının spesifik xüsusiyyətlərini seçmək olar. Bunun üçün UML dilinin aşağıdakı diaqramlarından istifadə etmək olar:

- sistemdən diaqram variantları şəklində istifadə etməklə prosesin funksional təsviri;
- agentlərin UML-də paketlər şəklində identifikasiyası və cavabdehlik oblastının təyini;
- diaqram siniflərinin istifadəsi ilə agentlərin arxitekturasının təsviri;
- fəaliyyət diaqramından istifadə etməklə hər agentin tapşırığının spesifikasiyası;
- protokolların təsviri.

Beləliklə, UML dilinin diaqramlarının əsas növlərini və onların MAS-in işlənməsi zamanı müxtəlif aspektlərin təsvirində tətbiq imkanlarını təhlil edərkən, belə nəticəyə gəlmək olur ki, layihələndirmənin obyektyönlü metodologiyasına əsaslanmaqla MAS-ın layihələndirilməsi rahat yerinə yetirilir, MAS-ın işlənməsi prosesinin formalaşmasına imkan verir, həmçinin MAS-in reallaşdırılması üçün zəruri olan müxtəlif prosesləri birləşdirir [7].

Hazırda MAS-ların layihələndirilməsinin ümumi metodologiyası olmadığından yuxarıda deyilənlərin əsasında aşağıdakı nəticəyə gəlmək olur ki, UML dilinin istifadəsi ilə reallaşan və obyektyönlü layihələndirmə metodologiyasına əsaslanmaqla proqram agentlərinin işlənməsi bu yanaşmadan istifadəni aktuallaşdırır və onun bir sıra elementlərinin agent nəzəriyyəsinə əlavə edilməsi MAS-ların işlənməsinin hər bir addımda optimallaşdırılmasını və rahatlığını təmin edir.

ƏDƏBİYYAT

1. Виттих В.А. Мультиагентные модели взаимодействий для построения сетей потребностей и возможностей в открытых системах / В.А.Виттих, П.О.Скобелев //Автоматика и телемеханика №1. 2003, с.177-185
2. Люгер Д.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем. М.: Вильямс, 2003, 864 с.

3. Hüseynov A.H., Məmmədova M.R. Avtomatlaşdırılmış layihələndirmədə agentyönümlü əlavələrin işlənmə sisteminin arxitekturasının təyini // SDU. Elmi xəbərlər. Təbiət və texniki elmlər bölməsi, c. 11, №3, 2011, s.103-108
4. Котенко И.В., Уланов А.В. Многоагентные моделирование защиты информационных ресурсов в сети Интернет // Известия РАН. Теория и системы управления. №5. 2007, с.74-88
5. Ferber J. A meta-model for the analysis and design of organizations in multi-agent systems/ J.Ferber.O.Gutkneeht / In Proceeding of the 3rd International Conference on Multiagent Systems (ICMAS 98). IEEE CS Press, 1998.
6. Mylopoulos J., Kolp M. and Castro J. UML for agent-oriented software development: The Tropos proposal / In Proc.of the 4thInt.Conf. on the Unified Modeling Language UML'01. Toronto, Canada, 2001.
7. Андреев В. Методы и средства создания открытых мультиагентных систем для поддержки процессов принятия решений / В.Андреев, В.А.Виттих, С.В.Батишев // Известия РАН. Теория и системы управления. №1. 2003, с.126-137
8. Hüseynov A.H., Əliyeva A.Q., Məmmədova M.R. ÇİS-lərin və onun elementlərinin idarə sistemi üçün produksiyalar bazasının işlənməsi və onların təsnifatı // SDU. Elmi xəbərlər. Təbiət və texniki elmlər bölməsi. c. 12, №1, 2012, s. 99-103

РЕЗЮМЕ

НЕКОТОРЫЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ МНОГОАГЕНТНЫХ СИСТЕМ С ВНЕДРЕНИЕМ УНИФИЦИРОВАННОГО ЯЗЫКА МОДЕЛИРОВАНИЯ

Салимова М.Р.

***Ключевые слова**: унифицированный язык моделирования, агент, многоагентная система, архитектура, реактивный агент, автономный агент.*

В статье рассматривается ряд вопросов проектирования многоагентных систем с использованием унифицированного языка моделирования. С использованием интерактивного языка программирования выдается ряд общих приказов на разработку архитектуры агентов. Проанализировано поведение агентов, изучено взаимодействие агентов в структуре многоагентных систем. Рассмотрены правила взаимодействия протоколов взаимодействия агентов с использованием языка унифицированного моделирования.

SUMMARY

SOME ASPECTS OF DESIGN OF MULTIAGENT SYSTEMS WITH THE APPLICATION OF UNIFIED MODELING LANGUAGE

Salimova M.R.

***Key words**: unified modeling language, agent, multi-agent system, architecture, reactive agent, autonomous agent.*

The article discusses a number of design issues of multi-agent systems using a unified modeling language. Using an interactive programming language, a number of general orders are issued for the development of an agent architecture. The behavior of agents is analyzed, the interaction of agents in the structure of multi-agent systems is studied. The rules of interaction of agents interaction protocols using the language of unified modeling are considered.

Daxilolma tarixi: İlkin variant 10.04.2019
Son variant 10.09.2019