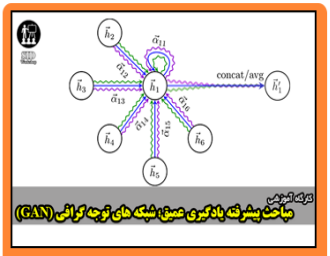
کارگاه های آموزشی مرکز اطلاعات علمی جهاد دانشگاهی

کارگاه آنلاین مکالمه روزمره انگلیسی

کارگاه آنلاین آموزش استفاده از وب آو ساینس



مباحث پیشرفته یادگیری عمیق؛
شبکه های توجه گرافی
(Graph Attention Networks)

مروری بر عامل گرایی درتوسعه نرم افزار برای سیستمهای چند عامله

شکوفه شفیعی
دانشجوی کارشناسی ارشد کامپیوتر
دانشگاه آزاد اسلامی واحد اراک
sh.shafeie@gmail.com

چکیده - مهندسی نرم افزار راهی برای غلبه بر پیچیدگی طراحی نرم افزار است. متدولوژیهای گوناگونی در این حیطه از علم مهندسی کامپیوتر ارائه شده است که هر کدام معایب و مزایاوکاربردهای خاص خود را دارد.یکی از متدلوژیهای جدید که برای تدوین نرم‌افزاربرای سیستمهای مهندسی پیچیده و توزیع شده که چندعامله هم می باشندو سیستمهای باز شبکه ای مانند اینترنت ایجاد شده است، مهندسی نرم افزار عامل گرا می باشدکه با توجه به خصوصیات این محیطها ازجمله سرعت متغیر، غیر قابل پیش بینی یا بازبودن که احتمال شکست اعمال درآنجا وجود داردبرای آنها مناسب می باشدوارائه متدلوژیهای مناسب را می طلبدکه در این مقاله به آن می پردازیم.

کلمات کلیدی- سیستم پیچیده نرم افزاری ، سیستمهای چندعامله،عامل،متدلوژی ،مهندسی نرم افزارعامل گرا

۱-مقدمه

مهندسی نرم افزار امروزه کاری بسیار مشکل است و هر روز مشکل تر هم می‌شود.[1] بنابراین نیاز به متدلوژیهای جدید برای کاربرد در طراحی و پیاده سازی سیستمهای پیچیده و توزیع شده[۱] امروزی امری ضروری به نظر می-رسد. این مقاله مقدمه ای بر متدلوژی جدیدی برای طراحی سیستمهای توزیع شده و پیچیده به نام متدولوژی عامل گرا است.

همانطورکه متدولوژیهای مهندسی نرم افزار تکنیک هایی را برای مدلسازی و ساخت سیستم های نرم افزاری ارائه می‌دهد؛متدلوژی مهندسی نرم افزار عامل گرا[۲] نیز امکاناتی را برای طراحی و ساخت سیستم های توزیعی پیچیده ارائه می‌دهد.

۲-مفهوم عامل[۳]

عامل ها اغلب به صورت موجودات نرم افزاری تعریف می‌شوند که به صورت خودمختار عمل کرده و می‌توانند یاد بگیرند. به طور کلی تعاریف گوناگونی برای عامل ارائه شده است:

طبق نظریه Russel و Norving عامل هر چیزی است که از طریق حسگرها[۴] محیطش را درک می کند و از طریق تاثیرگذارها بر روی محیط عمل می کند.[2]

در تعریف دیگری عامل به صورت سیستمهای محاسباتی خودمختار که در محیط پیچیده و پویایی قرار می گیرند، به شکل خودمختار در این محیط حس و عمل می کنند و با انجام این کار به مجموعه ای از اهداف یا وظایفی که برای آن طراحی شده اند، دست پیدا می کنند؛تشریح می شود[3] بارزترین خصوصیت برای عامل ها خود مختاری[۵] است که به معنی آنست که عامل

مختار است عمل دلخواه خود را انجام دهد. این خصوصیت، عامل را از دیگر برنامه های متداول نرم‌افزاری که به آنها گفته می شود دقیقاً چه کاری انجام دهند متمایز می کند و بدین دلیل از مهمترین مشخصه های عامل ها به شمار می رود.

عامل ها قادر هستند نسبت به انجام یا عدم انجام عمل یا درخواست صادر شده از طرف عامل دیگر تصمیم بگیرند؛ در حالیکه در مدل استاندارد شیء[6] بحثی در رابطه با رفتار انعطاف پذیر نمی شود.از جمله کاربردهای عامل می توان به مدیریت شبکه، مدیریت اطلاعات در محیطهای اطلاعاتی، مدل سازی و بهینه سازی فرایند های صنعتی، تجارت الکترونیکی، مدیریت فرایند های تجاری و سازمانی، واسطهای تطبیق پذیر کاربر و آموزش اشاره کرد.

تحقیقات در زمینه عامل را می توان به چهار دسته عمده طبقه بندی کرد:

سطح میکرو: عامل در فن آوری عامل که شامل معماریهای کنترل عامل و بنیادهای عامل بودن است.

سطح ماکرو: جامعه عامل ها در فن آوری عامل که شامل همکاری، هماهنگی، ارتباط، مذاکره و اصول ساخت سیستمهای چند عامله است.

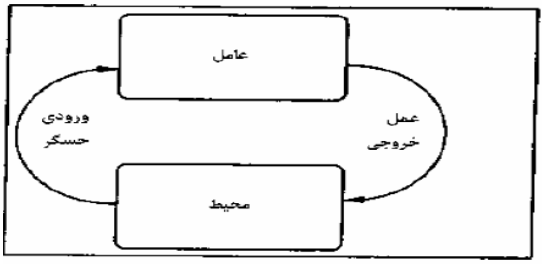
پیاده سازی سیستمهای مبتنی بر عامل که شامل محیطها، بستر های آزمایش با زبانهای برنامه نویسی و ارزیابی است.

مبانی توسعه سیستمهای مبتنی بر عامل که شامل استانداردها، مهندسی نرم افزار مبتنی بر عامل، تجاری سازی و مسائل عملی است.

عامل ها به عنوان مدل نسل بعدی برای سیستمهای مهندسی پیچیده و توزیع شده در نظر گرفته می شوند.[4]

عامل را به این شکل نیز تعریف می کنیم که یک سیستم کامپیوتری است که در محیطی تعبیه شده است که قادر است عملی خود مختار انجام دهد که این عمل در راستای رسیدن به اهدافی است که عامل برای آن طراحی شده است. شکل ۱ یک دید سطح بالا و مجرد از یک عامل را نشان می دهد.

در این شکل عامل در محیط خود نشان داده شده است و با عمل خروجی خود باعث تغییری در وضعیت محیط پیرامون خود می شود. محیط نیز در اثر انجام گرفتن این عمل از سوی عامل، عمل خروجی از عامل را به عنوان ورودی خود تلقی می‌کند و پاسخ خود را در جواب به عامل ابراز می دارد. سپس عامل از طریق ورودی حسگر خود، پاسخ محیط را حس می کند و دومرتبه عملی انجام می‌دهد که این عمل به ورودی محیط منتقل می شود و چرخه بالا از سر گرفته می شود.[1]



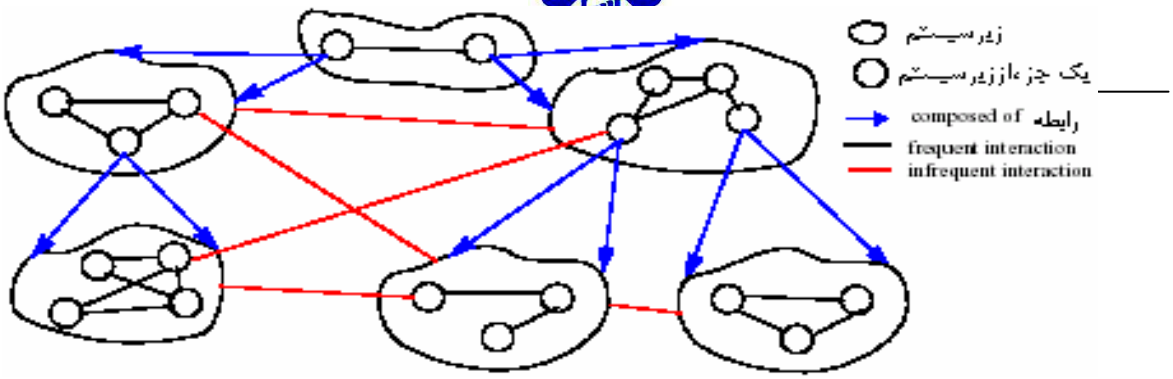
شکل ۱: یک عامل در محیط خود[1]

۳- مهندسی نرم افزار عامل گرا

مهندسی نرم افزار اساساً کار مشکلی است که هر روز سخت تر هم می شود.[1] به چند دلیل ازجمله زمان توسعه کمتر ، نیاز های غیر عاقلانه بیشتر و نامطمئن که در فضای بیشتری تغییر می کنند، ومحیطهای خطرناک بیشتر مانند اینترنت و سیستم های گوناگون مقیاس پذیر و توزیع شده در شبکه که در آنها پویایی و باز بودن سیستم افزایش یافته است. متدولوژی مهندسی نرم افزار عامل گرا امکاناتی را برای طراحی و ساخت سیستم های توزیعی پیچیده ارائه می دهد.

آنالیزوطراحی وپیاده سازی نرم افزار به عنوان یک مجموعه ازعاملهای دارای اثر متقابل بر یکدیگر و مستقل یک آینده روشن جهت ایجادپیشرفت در مهندسی نرم افزاررانشان می‌دهد.

ازآنجاکه نرم افزارهای صنعتی بطور ذاتی پیچیده می باشندوذاتاً به وسیله یک تعداد زیادی از بخش ها که بر روی یکدیگر اثر متقابل هم دارند مشخص می شوند؛این پیچیدگی[7] یک صفت ذاتی برای انواع وظایف و کارهایی است که نرم افزار برای آن ها بکار می رود. نقش



شکل ۲: نمایش یک سیستم پیچیده[1]

مهندسی نرم افزار در این میان تهیه و آماده سازی ساختارها و تکنیک هایی است که دستکاری کردن و کار با این پیچیدگی را ساده تر نمایدکه البته خود این پیچیدگی یک تعداد قواعد و ترتیب های مهم را ارائه می دهدکه غالباً شکل و ترکیب یک ساختار سلسله مراتبی رادارد قابل تجزیه و تحلیل به مجموعه های متوالی از زیر سیستم ها می باشدکه هر یک از آنها نیزبه نوبه خود یک توالی دیگر را تشکیل می دهند.

یک نمایش قانونی از یک سیستم پیچیده به صورت شکل ۲ می باشد:[1]

با معرفی این مشاهدات، مهندسان نرم افزار یک تعداد ابزارهای قوی جهت مدیریت کردن این پیچیدگی های ارائه شده؛اختراع نموده اندکه همانطورکه در شکل ۳ هم مشهود است عاملها ازآن حمایت می کنندوتطابق میان این سیستمهای پیچیده با سیستمهای چندعامله را نشان می دهد. این ابزارهای قوی مدیریتی به صورت مکانیسم های اصلی زیر می باشد.

۳-۱-مکانیسم های اصلی مدیریت سیستمهای پیچیده نرم‌افزاری

۳-۱-۱-تجزیه کردن[8]:

اصلی ترین روش جهت دستکاری کردن یک مساله بزرگ این است که آنرابه قسمت های کوچکتر و البته قابل مدیریت کردن که هرکدام از آن ها می تواند در تجزیه و تحلیل های نسبی مورد بحث واقع شوند تقسیم بندی کنیم.

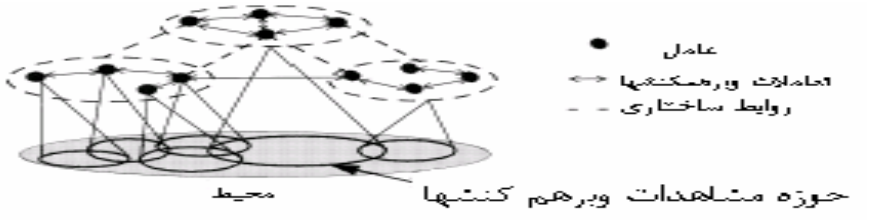
3-۱-۲-انتزاع[9]:

فرآیند تعریف یک مدل ساده شده از سیستم که بر روی برخی از جزئیات یا خصوصیات، تاکید می کند و از برجسته جلوه نمودن سایرین جلوگیری می نماید.

3-۱-۳-سازمان دهی[10]:

به فرآیند تشخیص و مدیریت نمودن روابط دارای اثر متقابل در هم که میان مسائل متعددی که مولفه ها را حل می نمایند، گفته می شود.

دونکته اساسی وجود داردکه ازنظر کیفی موجب تفاوت اثرات متقابلی که عاملها بر همدیگردارند از اثرات متقابلی که درسایرالگوهای مهندسی نرم افزار رخ میدهند می شوند. اولا اینکه: تعاملات بر مبنای عاملها، عموما از طریق یک زبان سطح بالا که ارتباط میان عاملها را فراهم می نماید یعنی یک زبان توصیفی[11] صورت می پذیرد. درنتیجه تعاملات معمولا دریک سطح دانش[12] بر حسب



شکل۳ :سیستم چند عامله معادل با سیستم پیچیده[1]

اینکه چه اهدافی می بایست در چه زمانی و به وسیله چه افرادی دنبال شوند،جریان می یابندو به وسیله متدهای رسیدگی و بازرسی یا فراخوانی های تابعی که دریک سطح گرامری دقیق عمل می نماید،انجام می شوند. ثانیا اینکه: عاملها جهت حل مسائل انعطاف پذیر بوده وپاسخ به تعاملات نیزلازمست دریک حالت دارای انعطاف پذیری مشابه صورت پذیرد.

پذیرفتن روش مبتنی بر عامل به عنوان یک روش مهندسی نرم افزار به معنی تجزیه و تقسیم نمودن مساله به اجزای متعددی است که دارای اثرات متقابل برهم بوده و خودمختار و مستقل نیز عمل نمایندو دارای وجود خارجی معینی جهت انجام دادن باشندوشامل عاملها و روابط و اثرات متقابل آنها بر یکدیگر و سازمان دهی آنهاومکانیسمهای واضح جهت توصیف و مدیریت نمودن روابط سازمانی پیچیده و متغیری که میان عاملها وجود دارند،می باشد.

با مشاهده این واقعیت که مهندسی نرم افزارعاملگرابرای سیستمهای چندعامله یک شیوه جدید و سریعاً درحال رشد است، این خطراحتمالی وجود دارد که محققان در مورد توانایی عاملگرایی بیش از حد، خوشبین باشند. برهمین اساس Jennings و Wooldridge ، درمورداین خطرات هشدارداده اند.[1,5]

۳-۲-مخاطرات وچالشهای[۱۳] ناشی از بکارگیری عاملها درمهندسی نرم افزار

خطرهای ناشی از بکارگیری عاملها درمهندسی نرم افزار را به پنج دسته اساسی می توان تقسیم بندی کرد که عبارتند از:

اشتباهات در سیاست گذاریهای مربوط به بکارگیری عامل می تواند رخ دهد، اگر مفهوم عاملها بیش از واقع دیده شود، یا به عنوان یک راه حل سراسری به کار برده شود.اشتباهات مفهومی ممکن است رخ دهد، اگر برنامه نویس فراموش کند که عاملها، نرم افزار و در واقع نرم افزار چند مسیره هستند. اشتباهات تحلیل و طراحی، ممکن است رخ دهد، اگر برنامه نویس از تکنولوژی مربوطه مانند سایر متدلوژیهای مهندسی نرم افزار چشم پوشی کند. اشتباهات سطح عاملی، ممکن است رخ دهد، اگر برنامه نویس سعی کند که در سیستم عامل ،از هوش مصنوعی بسیار زیاد یا بسیار کمی استفاده کند. اشتباهات سطح اجتماعی می تواند واقع شود، اگر برنامه نویس هرجا، عاملها را در نظر داشته باشد یا عاملهای خیلی کمی در سیستم به کار گیرد.

۴-مقایسه عاملها و اشیاء

برای نشان دادن اینکه آیا روشهای مبتنی بر عامل یک پیشرفت حقیقی را نسبت به حالتهای جاری که درصنعت شاهد آن هستیم نشان می دهند یا خیر؛ لازمست که روشهای مبتنی بر عامل با سایر روشهای مهم ومقدماتی مهندسی نرم‌افزار مانند روش شی گرا یا قطعه گرا[۱۴] مقایسه شود.[1]

این حقیقت را نمی توان انکار کرد که در بعضی جهات عاملها از اشیا مشتق می شوند. همانطورکه مهندسی نرم افزار عامل گرا از مهندسی نرم افزار شی گرا به وجود آمده است. عامل ها و اشیا، ویژگیها و رفتارهای مشابهی دارند. اما عامل ها برخودمختاری و تعامل بین عامل های مختلف بیشتر تمرکز دارند. مهندسی نرم افزار عامل گرا تکامل یک الگوی مهندسی نرم افزار جدید است،گرچه جنبه های زیادی را از مهندسی نرم افزار شی گرا قرض گرفته است .این تفاوتها وشباهتها به شرح زیرند.

۴-۱- تفاوت بین عاملها واشیاء

-اشیا دارای سازمان متمرکزی هستند، درحالی که عامل ها محاسبه توزیعی را مجاز می شمارند. اشیا در یک سیستم بیشتر بایکدیگر یکپارچگی دارند، درحالی که یکپارچگی عامل ها ضعیف است.

-عاملهانمی توانند مانند اشیا آزادانه ایجاد شده و از بین بروند.

-رفتار شی ثابت است و هرگز تغییرنمی کند، درحالی که عامل ها از تجربیاتشان یاد می گیرند و رفتارشان را تغییر می دهند.

- تعاملات اشیا اکثراً توسط شی دیگر درخواست می شود، درحالی که تعاملات عامل ها شامل واکنش به وقایع محیطشان یا درخواست از سایر عامل هاست.

- تعاملات اشیا معمولاً همگام است، درحالی که تعاملات عامل ها معمولاً ناهمگام است .
- الگوی عامل مبتنی بر کپسوله بندی قویتری از الگوی شی است. عامل ها اجازه می دهند که رفتارشان مجزا از متدها و پارامترهای مربوط به اشیا کپسوله شود.
- ارتباط شی به یک ارتباط یک به یک محدود می شود، درحالی که عامل ها می توانند ارتباط چندگانه یا موازی داشته باشند.

۴-۲-شباهتهای عاملهاو اشیا
-هر دو نگرش برنامه نویسی ماژولار را استفاده می کنند.
-هردو پارامتر داخلی خود را دارند.
-هر دو با عوامل پیرامون خود تعامل دارند.
-هردو از کپسوله سازی ومخفی سازی اطلاعات استفاده می‌کنند.
-هر دو درمورد سیستمشان همه چیز را نمی دانند.

۵- مکانیسمهای ارتباطی میان عاملهادر سیستمهای چندعامله

مکانیسمهای ارتباطی میان عاملها عبارتند از:[6]

۵-۱-KIF (knowledge Interchange Format):
یک فرم وصورت اضافی از منطبق مرتبه اول جهت کدگذاری مفهوم یک سیستم مبتنی بر دانش می باشد.[7]

۵-۲-KQML (Knowledge Query and Manipulation Language):
زبانی جهت پشتیبانی نمودن از ارتباطات مانند KIF است که میان عاملهای نرم افزاری ساخته شده است.[8] KQML یک زبان و پروتکل ارتباطی سطح بالا و پیام گرا برای تبادل اطلاعات بین عامل هاست. زبان KQML به سه لایه تقسیم می شود: لایه محتوا، پیام واقعی را حمل می کند که عامل ارسال کننده می خواهد به عامل دریافت کننده تحویل دهد. لایه ارتباطی مجموعه ای از پارامتر هایی را رمزگذاری می کند که برخی از پارامترهای ارتباطی سطح پایین تر را مانند یک ارسال کننده، یک دریافت کننده و یک شناسه منحصر بفردی که یک پیام را به طور انحصاری شناسایی می کند، را توصیف می کند. لایه پیام عمل گفتار پیام را مشخص می کند. آنتولوژی که مفاهیم را در محتوا تعریف می کند و زبانی که محتوا در آن رمزگذاری می شود.

۵-۳- FIPA (Foundation for Intelligance physical Agents) :
FIPA یک سازمان مبتنی بر عضویت است که توصیفاتی را به منظور عمل نمودن میان عاملهای نرم افزاری ناهمگن ساخته و پشتیبانی می کند.[9] مخزن و انباره کاتالوگ FIPA هزاران استاندارد FIPA؛ شامل ارتباط عاملها، انتقال پیامها، مدیریت، معماری و ساختار و برنامه های گوناگون است. بیش از توجه نمودن به طراحی سیستمها و توافق بر روی اینکه یاد بگیرند چگونه با یکدیگر ارتباط برقرار کنند، FIPA سیستمهای ارتباطی سخت افزاری را برای فعالیت میان عاملها از روشهای ساختار پیامهاگرفته تا مدیریت نمودن را جستجو می کند.

۵-۴-SOAP (Simple object Access Protocal):
SOAP یک توصیف مبتنی بر XML از رد وبدل کردن پیغام از یک کامپیوتر به کامپیوتر دیگر می باشد. همچنین فراخوانی متد راه دور اختیاری و انتقال پیام از طریق HTTP را هم توصیف می کند.[10]

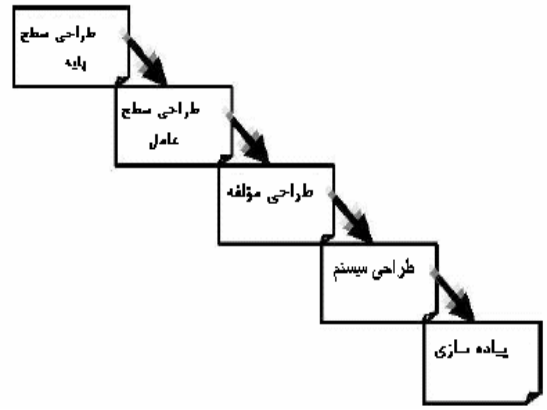
۵-۵-WSDL (Web Service Description Language):
WSDL یک سند XML را توصیف می کند تا تعیین کنند که چگونه می توان یک سرویس مبتنی بر وب را دریافت نمود .این اعمال به وسیله خدمات و سرویس ها پشتیبانی می شود. WSDL یک محصولی از UDDI / SOAP (Universal Description ,Discovery and Integration)[11]و کاری گروهی می باشد و در رابطه با SOAP استفاده می شود تا فرایندهای مستقل را به خدمات مبتنی بر وب راه دور اتصال دهد.

در جامعه سیستمهای چندعامله[۱۵] (MAS) این امر پذیرفته شده است که فواصل ارتباطی میان عاملها در سیستمهای چندعامله معنایی بیش از آنچه که در سیستمهای توزیع شده رسمی می دهد دارند.

به منظور تشخیص کار سیستمهای چندعامله از مسائل ارتباطی که در سیستمهای توزیع شده وجود دارند و به منظور بحث و تبادل نظر بر روی مسائل ارتباطی میان عاملها، معمولا افراد بیشتر برروی تعاملات و اثرات متقابل توجه می نمایند تا برقراری ارتباط.

۶- متدلوژی سیستمهای چند عامله

متدلوژی سیستمهای چند عامله شبیه به متدلوژی های مهندسی نرم افزار رسمی و قدیمی می باشد، اما برای استفاده در الگوی عامل توزیع شده به صورت ویژه درآمده است.این متدلوژی سطوح اصلی زیر راکه در شکل۴ هم دیده می شود دنبال می کندکه به طورواقعی سیستم راتوصیف می نماید.[6]

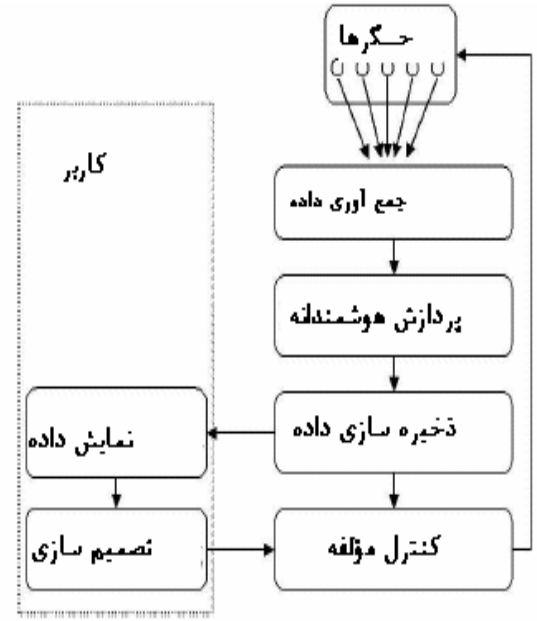


شکل۴: متدلوژی سیستمهای چند عامله

۶-۱- طراحی سطح پایه (اصلی):

اولین مرحله در متدلوژی سیستمهای چند عامله طراحی سطح پایه می باشد،که انواع پایه وتعاملات و اثرات متقابلی[۱۶] را که عاملها بر هم دریک سیستم دارندرا در برمی گیرد.دراین سطح،اینکه آیا یک عامل دارای هوشمندی[۱۷] است یا نه و اینکه چگونه آن هوشمندی به دست می آید و یا اینکه چگونه یک عامل تعریف می شود مهم نمی باشد.سطح پایه تنها با تعریف سطح بالا ازانواع عاملها و اهداف آنها و واسطهای خارجی آنها درارتباط است.سطح پایه همین واسطهای خارجی است که پروتکلهای ارتباطی میان عاملها را تعریف می‌نماید.درمرحله بعدی اینکه هر عا مل چگونه می تواند باسایر عاملها تعامل داشته باشد را توصیف می نماید مراحل موجود در طراحی سطح پایه عبارتند از:

مرحله ۱- مشخص نمودن انواع عاملها ومسؤولیتها ونقشهای آنها.مرحله ۲- مشخص نمودن برهمکنشها و روابط ممکن میان انواع عاملهامرحله که این تعاملات شامل گفتگوهای عاملها می‌شود وپروتکلهای ارتباطی مورد استفاده را تعریف می‌نماید.این پروتکلها فواصل ممکن پیغامها را که ممکن است میان عاملها جهت دستیابی به تعاملات ردوبدل شود تعیین می‌کند.درسطح پایه یک ارتباط منطقی ممکن میان تمام مولفه های مدل طراحی شده و ممکن است مانند شکل ۵ بیان شود:



شکل۵: تعاملات منطقی ممکن میان عاملها

مرحله۳- تبیین نمودن پروتکلهای هماهنگ سازی برای هریک از انواع تعاملات میان عاملها

۶-۲- طراحی سطح عامل:

دراین سطح معماریها وساختار عامل برای هریک از انواع تخصصی عاملها تعریف می گرددکه یک مساله کلیدی در طراحی می باشد که بر سایر فعالیتهای طراحی اثر می‌گذارد.انتخاب معماری برمبنای وظایف کارکردی صورت می‌گیرد.معماری که دریک سیستم

استفاده می شود،از مؤلفه های کلی ساخته می شود که ممکن است برای سایر برنامه های کاربردی نیز قابل پذیرش باشد.مؤلفه های کلی می توانند مطابق با نقشها و وظایفی که عامل درسازمانها برعهده می‌گیرد تغییر شکل دهد.همان طور که در مذاکره عاملها گفته شد،به همان مقدار هم قابلیتهای واکنشی نیز نیاز می باشد. عاملهای مذاکره کننده مسؤول مدیریت تعاملات میان کاربر ومدلسازی کاربر می باشند.عاملهای واکنشی[18] نیز استخراج اطلاعات برمبنای درخواست وتقاضا را انجام می دهند.در هردوی عاملهای مذاکره کننده[19] و واکنشی ساختار عامل می بایست امکاناتی را جهت مدیریت نمودن نخهای ارتباطی به صورت مستقل فراهم نماید.مراحل طراحی معین در سطح عامل عبارتند از:مرحله ۱-نگاشت نمودن وتبدیل رفتارهایی که در مذاکرات عامل تعریف شده است به مولفه های داخلی

مرحله ۲-تعریف نمودن ساختارهای داده که در مذاکرات عامل تعیین می شوند.این ساختارهای داده ای ورودی یا خروجی عامل را نمایش می دهند.

۶-۳- طراحی مؤلفه:

وقتی که ساختار عامل تعریف شد،مولفه هایی که مشخص شده اند می بایست طراحی شوند،اگرچه در صورتی که مولفه هایی وجودداشته باشند،که بتوانند دوباره مورد استفاده واقع شوند خوبست و هدف نهایی ما نیز این است که عاملها را بدین صورت تعریف نماییم تا بتوانیم از مزایای تکنولوژی هایی که چنین مولفه هایی را قادر می سازندمانند Java Beans [12]استفاده نماییم،مولفه های واضح یک عامل عبارتند از:

۱-طراحی کنندگان

۲-مکانیسمهای استدلال و استنتاج[20]

۳-الگوریتمهای جستجو

۴-الگوریتمهای یادگیری

۵-الگوریتمهای آنالیزگروتجزیه و تحلیل کننده

۶-۴- طراحی سیستم:

این مرحله زمانی رخ می دهد که طراحی محدوده ،عاملها ومولفه ها و اجزای لازم تکمیل شده باشد.به وسیله تعریف نمودن محدوده و قلمرو در ابتدا، طراحی سیستم تبدیل به تمرین انتخاب انواعی از عاملهای مورد نیازمی شود که می‌بایست پارامترهای مشخص شده در تعریف عامل را به خوبی تعیین نماید. علی رغم اینکه این قسمت به طور تکنیکی قسمتی از طراحی سیستم نمی باشد،زمانی که یک سیستم تعریف شد می توانیم صفات ویژه و مشخص مورد علاقه مانند قابلیت اطمینان را بررسی نماییم.

مراحل مشخص در طراحی سیستم عبارتند از:

۱-انتخاب انواعی از عاملها که مورد نیاز هستند.

۲-تعیین تعداد عاملهایی که هریک از انواع لازمه زیر رااحتیاج داشته باشند که تعریف آنها بصورت زیراست:

۲-۱-محل فیزیکی یا آدرس عامل

۲-۲-انواع گفتگوها و مذاکراتی که دردامنه تعریف شده باشد.

۲-۳-سایر پارامترهایی که دردامنه تعریف شده باشند.تعریف نمودن یک دامنه و سیستم نیازمند مجموعه ای از ابزارهای رسمی است که مارا قادربه استفاده مجدداز مولفه هاواجزای موجود وترکیب نمودن اجزای جدید وآنالیز نمودن صفات متعددی ازسیستم می کند.

۶-۵- پیاده سازی[21]:

نهایتا پس از اینکه مدل چند عامله توصیف شد ،آخرین مرحله پیاده سازی می باشد؛که درآن تشخیص می دهیم که چه ابزارها و زبانهای برنامه سازی لازمست استفاده شود.امروزه این ابزارها معمولا می بایست مستقل از ساختار[22] و انعطاف پذیر باشندو به خصوص در مدل ما قابلیت فراهم آوردن یک قالب کاری برای عامل را داشته باشند که به عنوان مثالjava(محصول شرکت سان)این امکان را فراهم آورده است.[13]

۷-نتیجه‌گیری

مهندسی نرم افزار به سوی طراحی های با استفاده از مفاهیمی چون واحدهای مستقل و خودمختاردارای ارتباط با یکدیگر پیش رفته است. نرم افزارها امروزه به وسیله ایجاد ماجولهای توزیع شده و گسترده مستقل از یکدیگرکه به وسیله افراد مختلف و ممکن است با زبانهای برنامه نویسی متفاوتی نوشته شده باشند ،طراحی می

شوند. سیستمهای چند عامله نیز در این میان یک نقش اساسی به منظور رفتارنمودن به عنوان یک جانشین برای سیستمهای شی گرا بازی می کنندو رفتارهای محلی را با خودمختاری و تصمیم گیریهای توزیع شده ترکیب می نمایند، لذا این سیستمها نظیر هر سیستم نرم‌افزاری نیاز به روشهای مهندسی نرم‌افزار دارند و ارائه متدولوژی برای طراحی سیستمهای چندعامله که دارای ویژگی توزیع شدگی هم هستند امری ضروری به نظر می رسد . متدولوژی ای که دراین مقاله برای سیستمهای چندعامله به آن پرداخته شد، بادنبال کردن سطوح اصلی زیر توصیف سیستم به طورواقعی رادربرمی گیرد.این سطوح عبارت اند از: طراحی سطح پایه- طراحی سطح عامل- طراحی مؤلفه- طراحی سیستم- پیاده سازی؛که درهر سطح یک سری مراحل باید انجام شود. از آنجایی که مهندسی نرم افزارعاملگرا یک شیوه جدید و سریعاً درحال رشد است، این خطر وجود دارد که محققان در مورد تواناییهای عاملگرایی بیش از حد، خوشبین باشند؛ چراکه برخی از خصوصیات عامل که به مدل قدرت می بخشد، ممکن است منجر به بروز این چالشها شود.درخصوص جلوگیری از بروز چنین اشتباهاتی لازم است آنهارا بررسی نموده و نیز به مقایسه عاملگرایی با سایر روشهای مهندسی نرم افزار پیشین بپردازیم که تاحدودی در این مقاله به آن اشاره گردید.

علیرغم این مشکلات و با در نظر گرفتن محاسن و قدرت عاملگرایی و مدل ذکر شده ، این امر قابل پیش بینی است که مهندسی نرم افزار در آینده به سوی عاملگرایی پیش رود همچنانکه امروزه بر مبنای شی صورت می گیرد.

مراجع

[1] Nicholas R. Jennings and Michael Wooldridge “Agent-Oriented Software Engineering” Department of Electronic Engineering Queen Mary & Westfield College University of London London E1 4NS, United Kingdom.

[2]Stuart J. Russell and Peter Norvig (1995) “Artificial Intelligence: A Modern Approach” Prentice-Hall, 32-33.

[3]Maes P.(1995) ”Modeling Adaptive Autonomous Agents, Artificial Life: An overview”, MIT Press.

[4]C. Guilfoyle and E. Warner (1994) “Intelligent Agents: The New Revolution in Software” Ovum Report.

[5]Michael Wooldridge and Nicholas R.J ennings"Pitfalls of Agent-Oriented Development"Department of Electronic Engineering Queen Mary & Westfield CollegeUniversity of London, London E1 4NSUnited Kingdom.

[6]Vira Smirnova “Multi-agent System for Distributed Data Fusion in Peer-to-Peer Environment” University of JyväskyläDepartment of Mathematical Information Technology Master’s Thesis Mobile Computing 28.11.2002.

[7] Genesereth, M., Fikes, R., “Knowledge interchange format version 3.0 reference manual”, Stanford University technical report: Stanford, CA., 1992.

[8] Finin, T., Labrou, Mayfield J., “KQML as an Agent Communication Language, in Bradshaw J”, Software Agents, MIT Press Cambridge, 1997.

[9] Foundation for Intelligent Physical Agents, “FIPA Home Page” http://www.fipa.org/.

[10] Ballinger, K. “Web Services Interoperability and SOAP”. Microsoft Developer Network Library, May 1,2001.

[11] UDDI home page - http://www.uddi.org/

[12] Enterprise JavaBeans Tutorial http://developer.java.sun.com/developer/onlineTraining/Beans/EJBTutorial/

[13] Java Sun homepage -http://java.sun.com/

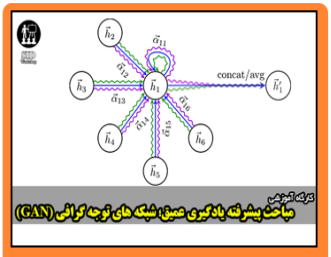
زیرنویس ها

[1]Distributed
[2] Agent oriented software Engineering (AOSE)
[3] Agent
[4] Sensor
[5]Autonomous
[6] Object
[7] Complexity
[8] Decomposition
[9] Abstraction
[10] Organisation
[11] declarative
[12] Knowledge level
[13]pitfalls
[14]Component-Based
[15] Multi Agent System(MAS)
[16] Interaction
[17]Intelligence
[18] Reactive
[19] Cognitive
[20] Inference
[21] Implementation
[22] platform

کارگاه های آموزشی مرکز اطلاعات علمی جهاد دانشگاهی

کارگاه آنلاین مکالمه روزمره انگلیسی

کارگاه آنلاین آموزش استفاده از وب آو ساینس



مباحث پیشرفته یادگیری عمیق؛ شبکه های توجه گرافی (Graph Attention Networks)