معماری های سرویس گرا

نویسنده: حمید فطانت

1-مقدمه..... 2
2- تعریف، مزایای و ادبیات موضوع..... 3
3-سه ویزگی های کلیدی رویکرد سرویس گرائی 5
4- سرویس مولفه و مشخصه ها ی سرویس وب..... 6
5-تعریف معماری سرویس گرا 11
6-سرویس های ترکیبی (ارکستریشن و کاریوگرافی)..... 15
7-مفاهیم اصلی و چرخه حیات درمعماری سرویس گرا 16
8-اصطلاحات و مفاهیم رایج در معماری سرویس گرا..... 19
9- خصوصیات اساسی جهت استفاده بهینه از سرویس ها 25
10-مقیاس پذیری از طریق رفتار آسنکرون و صف بندی 25
11-ویژگی های سرویس و محاسبات سرویس گرا..... 27
12- نرم افزار به‌عنوان سرویس 28
13-طراحی نرم افزار سرویس گرا 29
14-معماری سرویس گرای توسعه یافته..... 30
15-ویژگی های سیستم های نرم افزاری معماری سرویس گرا..... 33
16-یکپارچه سازی سیستم های سازمانی و تعامل پذیری بین سازمانی..... 36
18- یکپارچه سازی سیستم های سازمانی و تعامل پذیری بین سازمانی به کمک معماری سرویس گرا..... 37
نتیجه گیری..... 41

1-مقدمه

بی شک عصر حاضر بنام عصر اینترت و عصر فناوری اطلاعات رقم خورده است و همه ارکان زندگی را دچار تغییر و تحول نموده است. پارادایم های اصلی قرن بیستم در این عصر عمیقا تغییر کرده و مواردی همانند حذف زمان و مکان و افزایش پیچیدگی و همینطور اطلاعات به عنوان قدرت جایگزین پارادایم های گذشته شده اند. آنچه که پیامد این پارادایم ها است تغییر عمیق در سیمای سازمان, بنگاه و اساسا هر گونه جمعیت های انسانی است که بدلایل رشد فناوری می توانند بطور توزیع شده توسعه یافته و از امکانات و منابع یکدیگر استفاده کنند. بی شک در چنین صورتی با وجود سیستم ها و مکانیزم های متعدد و توزیع شده آن چیزی که می تواند بشدت بر روی زندگی مردم تاثیر گذارده مسئله شفافیت و همروندی در این نوع سیستم ها است که در غیر این صورت استفاده از آنها را بشدت غیر کارامد می کند. سرویس گرائی از جمله راه حلی در فناوری اطلاعات بوده که اگر بدرستی فهم شده و بدرستی بکار گرفته شود می تواند مسئله شفافیت و همروندی سیستم های توزیع شده را بطور قطعی حل کند.

همانند سرنوشت همه مفاهیم دیگر فناوری اطلاعات، سرویس و سرویس گرائی نیز در قرن حاضر از مفهومی فنی و مهندسی به مفهومی عمومی و به اصطلاح پوپولیستی تبدیل شده و همه بطور کلی از ان بجا و نابجا استفاده می کنند. این واقعیت، استفاده از این مفهوم را در عمل دچار مشکل کرده و در وحله اول فهم و درک اولیه ان را دچار مشکل می کند. بدلیل انفجار اطلاعاتی در عصر حاضر و توسعه و انتقال دانش در عرصه های مختلف فهم مفاهیم چند بعدی بکار گرفته شده در عصر حاضر براحتی میسر نبوده و بویژه این در باره مفاهیمی صادق است که مدلول آن ذهنی بوده و به تعبیر ابن سینا ثانویه باشد. در چنین حالتی و حتی در هنگامی که مدلول عینی و ملموس باشد می توان با بکارگیری مفهوم استعاره[1] (بکار گرفته شده و معرفی شده در هوش مصنوعی مدرن و علوم شناختی) مشکل مزبور را حل نموده و فهمی درست را در ذهن متبادر کرد. به این ترتیب رساله حاضر تلاشی است در راستای درک . و فهم درست استعاره های سرویس و معماری سرویس گرائی و بکارگیری درست آن در عمل بطوریکه بتواند در سیستم های توزیع شده بکار گرفته شده و مسائلی نظیر شفافیت، همروندی، قابلیت استفاده مجدد، دسترس پذیری و امثالهم را در سطح معماری و مهندسی حل کند.

آن چه که اهمیت سرویس گرائی را برای معماری شبکه محور دو چندان می کند علاوه بر مضامین بالا، اهمیت سرویس گرائی در به وحدت رساندن دو رویکرد اصلی این چارچوب است:

1- توسعه دیدگاه های فرایند های عملیاتی، سیستم های اطلاعاتی و معماری فنی

[1] Metaphore

2- توسعه یک روش محاسباتی شناختی که بر اساس آن بتوان فرایند حس سازی را توسعه داده و سرویس های لازم برای عامل های شناختی را در شبکه های اجتماعی توسعه دهد.

به این ترتیب شناخت سرویس گرائی اهمیتی اصلی داشته و فهم و بکارگیری آن درراستای یکپارچگی معماری نقشی کلیدی دارد.

2- تعریف، مزایای و ادبیات موضوع

معماری سرویس گرا به عنوان یکی از آخرین دستآوردها در تولید نرم افزار، به نظر می رسد، در سالهای آتی معماری غالب صنعت فناوری اطلاعات و ارتباطات باشد. علت بوجود آمدن این معماری، ایده ای بود که در ذهن تعدادی از معماران اولیه آن وجود داشت و آن اینکه نرم افزار برای یک سازمان به عنوان یک سرویس یا خدمت مطرح است. در مدل نرم افزار به عنوان سرویس شما نرم افزار خود را بگونه ای طراحی می کنید که قابل استفاده توسط سیستم های دیگر باشد یعنی دیگران می توانند برای استفاده از سرویس شما ثبت نام کرده و هر موقع که لازم باشد از خدمات آن بهره می برند، همانند حالتی که در مورد شبکه های تلویزیون کابلی وجود دارد. تا زمانی که شما به سرویس متصل هستید، شما می توانید هر لحظه که خواستید از سرویس های آن استفاده کنید. .

برای مدتهای طولانی برنامه نویسان سعی می کردند تا، کدهای خود را بصورت ماژولار[1] بنویسند، تا بتوان از آن در تولید نرم افزارهای دیگر استفاده کرد. تفاوت نوشتن کد بصورت ماژولار بر اساس معماری سرویس گرا در حجم مخاطبان آن است. دوباره به همان مثال اول برمی گردیم، وقتی شما کد خود را به منظور قابل استفاده بودن توسط نرم افزارهای دیگر، به شکل ماژولار می نویسید مانند این است که، یک شبکه تلویزیون کابلی درون یک ساختمان خاص دارید و بنابراین فقط ساکنین آن ساختمان می توانند از آن بهره برداری کنند. در جهان امروز طیف مخاطبانی که بالقوه می توانند از سرویس شما استفاده کنند، کل کاربران روی شبکه اینترنت است. بنابراین باید مکانیزمی بوجود می آمد، که می توانست پاسخگوی این محیط جدید (اینترنت) و کاربران آن باشد و به همین دلیل معماری سرویس گرا بوجود آمد. این معماری توسط افراد و موسسات و دانشگاه های مختلفی توسعه یافته و توسط شرکت های نرم افزاری بزرگی همانند شرکت مایکروسافت، ای بی ام و امثالهم حمایت شد. این دو شرکت نام برده شده دو شرکت طی سالهای اخیر از حامیان اصلی سرویس های وب و عامل بسیاری از ابداعات جدید در حیطه سرویس های وب همانندداپلیو اس ای[2] و دی دی ای[3] بوده اند . از نمونه های استفاده از این معماری در کشور

[1] Modular
[2] USE
[3] UDDI

خودمان، سازمان ثبت احوال کشور است که موظف شده تا پایگاه های اطلاعاتی خود را بصورت سرویس وب و مبتنی بر این معماری به سایر نهادها مانند نیروی انتظامی و سایر دستگاه ها ارائه دهد.

معماری سرویس گرا یا به اختصارSOA روشی جدید و در حال تکامل برای ساخت برنامه های توزیع شده[1] است. سرویس ها مولفه های توزیع شده با رابط های تعریف شده و مشخص هستند که پیام های به زبان XML را پردازش وتبادل می کنند. با رویکرد سرویس گرا می توان راه حل های را ارائه داد که به مرز دامنه های سازمان، شرکت یا دپارتمان محدود نیستند. با استفاده از معماری سرویس گرا می توان در شرکتی که دارای سیستم ها و برنامه های کاربردی مختلف بر روی سکو[2] های متفاوت است، یک راه حل یک پارچه سازی با استقلال زیاد[3] را توسعه داده بطوریکه جریانی یکنواخت و ناهماهنگ کار را تضمین کند. هر کس که از سایت های تجارت الکترونیکی به صورت برخط خرید کرده باشد، با مفهوم سرویس ها آشنا است. وقتی که سفارش خود را می دهید، باید اطلاعات کارت اعتباری خود را ارایه داده که به طور معمول توسط یک فراهم کننده سرویس ثانویه، تایید و شارژ شود. وقتی که سفارش پذیرفته شد، شرکت سفارش گیرنده با یک شرکت فراهم کننده سرویس حمل ونقل هماهنگ کرده و در نهایت کالای شما تحویل می شود. نیاز به معماری سرویس گرا از جنبه ای دیگر نیز به نحوه بارزی در برنامه های کاربردی تجارت الکترونیکی[4] مشهود است. اگر مثلا مولفه[5] مربوط به پرداخت با کارت اعتباری غیربرخط و یا غیر فعال باشد،قرار نیست که فرایند فروش متوقف شود. بلکه سفارش ها بایستی پذیرفته شوند وعملیات پرداخت به وقت دیگری موکول شود.

همانند سایر معماری های توزیع شده، سرویس گرا ئی توسعه برنامه های کاربردی را با استفاده از مولفه هائی که در دامنه های جدا از هم قرار دارند را ممکن می سازد معماری سرویس گرا از سرویس های وب به عنوان نقاط ورود برنامه کاربردی استفاده می کند که از لحاظ مفهومی معادل همان مولفه های پراکسی [6] و استاب[7] در سیستم های توزیع شده سنتی مبتنی بر مولفه هستند . با این تفاوت که در این جا ارتباط بین سرویس وب و استفاده کننده خیلی آزاداترانه ومستقل تر است .به علاوه معماری سرویس گرا به خاطر در بر داشتن فاکتورهایی نظیر قابلیت اطمینان سرویس، جامعیت پیام، یکسانی تراکنش و

[1] Application Distributed
[2] Platform
[3] loosly coupled
[4] Ecommerce
[5] Component
[6] proxy
[7] Stub

امنیت پیام اهمیتی حیاتی در تجارت الکترونیکی داشته و از این حیث منحصر به فرد است. در امور تجاری واقعی نمی توان روی سرویس هایی که فقط یک درخواست را به خاطر این که فهمیده شود پردازش کرد. و بطور معمول به قطعیت و اطمینان بیشتری نیاز است. واضح است که سیستم های مختلف ممکن است بعضی اوقات غیر فعال باشند و یا پاسخگویی آن ها در دفعات مختلف متفاوت باشد . با وجود این هیچکدام از این موارد نباید دلیلی برای کنار گذاشتن یاعدم پاسخ به یک درخواست باشند. علاوه بر آن نباید هیچ ابهامی در نحوه فراخوانی یک سرویس وجود داشه باشد. اگر سیستمی توانایی های خود را در قالب سرویس وب ارائه کند. در آن صورت نحوه فراخوانی آن سرویس باید به طور واضح مستند سازی و اعلام شود . بسیاری از مسائل دسترس پذیری و مقیاس پذیری برنامه های کاربردی امروزی در معماری سرویس گرا حل شده اند که احتمال نقض آن ها در هر مر حله ای از جریان کار بسیار زیاد است. در معماری سرویس گرا فرض بر این است که خطا وجود دارد و می تواند مهار شود، بنابراین برای مثال اگر یک سرویس نتواند یک پیام را در مرحله اول بپذیرد . این معماری طوری طراحی شده است که پیام مجددا می تواند فرستاده شود. واگر یک سرویس به طور کامل قابل دسترس نباشد، (که هرگز نباید در یک سیستم معماری سرویس گرا ی پایدار[1] انفاق بیفتد) آن وقت معماری طوری طراحی شده است که روی دادن خطاهایی که منجر به قطع کامل در خواست سرویس می شود،امکان پذیر نیست و چون خطاهای موقت در بخشی از جریان کار نمی توانند کل فرایند تجاری را از کار بیاندازند بنابراین معماری سرویس گرا قابلیت اطمینان را افزایش می دهد.

در حالت کلی، معماری سرویس گرا فرایندی تکامل یافته را ارائه می نماید و ازاین منظر می توان آن را بلوغ سرویس های وب و فناوری های یکپارچه سازی به حساب آورد . در معماری سرویس گرا به این امر توجه شده است که سیستم های با اهمیت حیاتی که بر مبنای فناوری های توزیع شده ساخته می شوند. باید تضمین های خاصی را تامین نمایند . در این گونه سیستم ها باید این اطمینان وجود داشته باشد که در خواست های سرویس به طور صحیح مسیر دهی و هدایت می شوند، در زمان مناسب به آن ها پاسخ داده می شود، و این سرویس ها به طور واضح و دقیق سیاست های ارتباطی و رابط های خود را اعلام می کنند.

3-سه ویژگی های کلیدی رویکرد سرویس گرائی

یک رویکرد موفق برای یک معماری بویزه معماری های فراسازمانی بکارگیری ان در سطوح مختلف تجرید بوده که در این صورت و در نهایت یک تصویر یکپارچه از همه مفاهیم در تمامی سطوح تجرید نتیجه گرفته می شود. نتیجه طبیعی مطالب مزبور این است که یکی از ویژگی کلیدی برای یک رویکرد طراحی در سطح سازمان یا فراسازمان امکان بکارگیری ان در سطوح

[1] Stable

مختلف تجرید بوده که موجب ایجاد یکپارچگی مفهومی بعنوان یکی از اهداف اصلی هر معماری می شود. رویکرد سرویس گرائی می تواند در سطح عملیات یاکسب و کار، سیستم های اطلاعاتی و برنامه های کاربردی، داده و زیر ساخت با موفقیت بکار گرفته شود.

ویزگی دوم و کلیدی یک رویکرد طراحی موفق استفاده از بلاک های سازنده (کوچکترین عنصر طراحی) با انسجام بالا است که در این صورت پایداری و استحکام طراحی را می توان نتیجه گرفت.

ویژگی سوم و کلیدی یک رویکرد طراحی موفق ارتباط سست و ضعیف بین بلاک های سازنده بوده که در این صورت مولفه های ساخته شده بر پایه این بلاک ها دارای خواصی گوناگون از جمله سهولت در بروزرسانی و طول عمر طولانی هستند.

سرویس گرائی با استفاده از مفهوم سرویس مولفه (دانه ریزترین سرویس بعنوان یک واحد نرم افزاری) واجد هر سه ویزگی فوق هستند: بکارگیری سرویس گرائی در همه سطوح تجرید سازمان، انسجام بسیار بالای سرویس مولفه و بالاخره اتصال سست بین همه مولفه هائی که از ترکیب سرویس ها بوجود آمده اند نشان دهنده موفق بودن طراحی سرویس گرا است.

4- سرویس مولفه و مشخصه ها ی سرویس وب

بسیاری از ما آنقدر با فناوری های سرویس های وب آشنا هستیم که اغلب در باره این که خود سرویس ها واقعا چه هستند، فکر نمی کنیم. به این ترتیب برای انکه از معماری سرویس گرا را بدرستی فهم کرده و درک نماییم لازم است که ماهیت سرویس را بدرستی تشخیص دهیم. همچنانکه ماهیت عصر اطلاعات و انفجار ترکیباتی اطلاعات است بیشتر مفاهیم فناوری اطلاعات از جمله سرویس بطورمستقیم قابل تعریف نبوده و به همین دلیل از استعاره برای این منظور استفاده می شود. استعاره مفهومی قابل مانوس بوده که از ان برای تبیین مفهومی دیگر استفاده می شود. سرویس نیز استعاره ای[1] متعلق به دنیای تجارت بوده که بدلیل شناخت نسبی آن توسط عموم از این استعاره در این عرصه استقاده می کنیم.

در ادامه ابتدا به تعریف سرویس مولفه پرداخته که دانه ریز ترین نوع سرویس است و در حقیقت ترکیب های مختلف انها سرویس های دانه درشت را بدست می دهند. سه مشخصه ساختنی زیر در کنار یکدیگر و با هم ماهیت یک سرویس مولفه راشرح می دهند :

- سرویس ها مولفه ء مستقلی هستند که پیام های XML که دارای ساختاری مشخص و خوش تعریف[2]بوده را پردازش می کنند.

[1] Metaphore
[2] Well defined

- سرویس ها دارای رابط های خوش تعریف هستند که به وسیله یک سند مبتنی بر XML ارائه می شوند. به عنوان نمونه زبان توصیف سرویس وب[1] یا به اختصار WSDL خوانده می شود. محتویات این سند عملیاتی (متدهایی) که توسط سرویس ارائه می شوند را شرح می دهد. از جمله اطلاعات مربوط به انواع داده، اطلاعات نحوه اتصال به سرویس، جهت یافتن و ارتباط با عملیات سرویس وب است

- سرویس ها دارای نقاط نهائی[2] هستند که استفاده کنندگان از سایر سرویس ها می توانند بر اساس آدرس سرویس (معمولا URL) به آن ها متصل شوند. این همان چیزی است که ارتباط(جفت شدن) آزادانه خوانده می شود.

رنامه های کاربردی معماری سرویس گرا نیاز به پشتیبانی و امکانات زیر ساختی زیادی دارند. از جمله امکانات ارسال و دریافت مختلفی ، زیر ساخت امنیتی و پشتیبانی برای پیام رسانی مطمئن. شرکت های مختلفی، از جمله ای بی ام و مایکروسافت، برای ارائه مشخصه های استانداردی که دامنه گسترده فناوری های زیر ساخت معماری سرویس گرا را پوشش دهد، با یکدیگر همکاری می کنند. متاسفانه مشخصه های سرویس های وب در محیطی ارایه و توسعه می یابند که شرکت های دخیل در آن بیشتر رقیب هستند تا شریک. رقابت های میان شرکت ها باعث می شود که نتواند بر سر استانداردهای صحیح و مناسب به توافق برسند.اغلب،گروههای مختلف شرکت ها، برای موارد یکسان ، استانداردهای متفاوتی را دنبال می کنند . سازمان های غیر انتفاعی مثل اوآسیس[3] گرد همایی هایی برای همکاری در ارایه و توسعه استانداردها و مشخصه های سرویس های وب برگزار می کنند

سازمانWS-I [4] یک هدف اصلی دارد و آن ارائه مشخصه های استانداردی است که سرویس های وب بتوانند با استفاده از آن روی سکو های مختلف با هم تعامل داشته باشند. به بیان دیگر، هدف این سازمان این است که سرویس های وب بتوانند با هم کار کنند ،بدون توجه به این که تحت چه سکوئی عمل می کنند و یا با استفاده از چه ابزارهایی ایجاد شده اند . این مشخصه های سرویس های وب زمینه های گسترده ای را پوشش می دهند، از پروتکل های نقل و انتقال داده تا امنیت که مجموعه آن ها تحت عنوان پروفایل پایه WS-I جمع آوری شده اند .

1 Web Service Description Language
2 Endpoint
3 OASIS
4 Web Services Interoperability

برنامه های کاربردی معماری سرویس گرا نیاز به پشتیبانی و امکانات زیر ساختی زیادی از جمله امکانات ارسال و دریافت پیام های مطمئن و توسعه زیر ساخت های امنیتی و حمایتی برای آن دارند. شرکت های مختلفی، از جمله آی بی ام و مایکروسافت، برای ارائه مشخصه های استانداردی که دامنه گسترده فناوری های زیر ساخت معماری سرویس گرا را پوشش دهد، با یکدیگر همکاری می کنند .

متاسفانه مشخصه های سرویس های وب در محیطی ارایه می شوند و توسعه می یابند که شرکت های دخیل در آن بیشتر رقیب هستند تا شریک. رقابت های میان شرکت ها باعث می شود که نتواند بر سر استانداردهای صحیح و مناسب به توافق برسند .اغلب،گروههای مختلف شرکت ها، برای موارد یکسان ، استاندارهای متفاوتی را دنبال می کنند .(سازمان های غیر انتفاعی همانند اوآسیس گرد همایی هایی برای همکاری در ارایه و توسعه استانداردها و مشخصه های سرویس های وب برگزار می کنند).

مشخصه های سرویس های وب به طور عمده در گروه های زیر دسته بندی می شوند :

- نقل و انتقال[1]

این گروه ، پروتکل های ارتباطی برای انتقال داده های خام بین سرویس های وب را تعریف می کنند نمونه هائی همانند HTTP، HTTPSو SMTP از این جمله اند.

- پیام رسانی[2]

این گروه از مشخصه ها تعیین می کنند که پیام های XML که سرویس های وب تبادل می کنند. چه فرمتی باید داشته باشند. این گروه مشخصه های SOAP برای نحوه رمز گذاری پیام و مشخصه های XML و XSD برای کلمات کلیدی پیام [3] را شامل می شود. مشخصه های آدرس دهی سرویس های وب نیز در این گروه قرار دارد . این مشخصه ها اطلاعا ت مقصد پیام را از پروتکل نقل و انتقال داده ها، مستقل می سازد . برای مثال می توان با استفاده از مشخصه های آدرس دهی سرویس های وب، چندین مقصد برای یک پیام XML تعریف کرد.

- توصیف[4]

1 Tranport
2 Messaging
3 Vocabulary
4 Description

این گروه شامل مشخصه هایی برای تشریح و توصیف یک سرویس وب است. برای توصیف بطور معمول از WSDL برای قرارداد سرویس و از XSD برای تعریف شماهای نوع داده استفاده می شود. این گروه همچنین مشخصه سیاست گذاری سرویس وب [1] را شامل می شود که سیاست گذاری هایی که یک سرویس وب به هنگام ارتباط با یک سرویس گیرنده[۲] اعمال می کند را تشریح می کند. برای مثال یک سرویس ممکن است شرایط خاصی برای فراخوانی عملیاتش داشته باشد. مشخصه سیاست گذاری به سرویس وب این امکان را می دهد که به سرویس گیرنده های احتمالی قوانین اجرای یک درخواست سرویس موفق را گوشزد می کند. نهایتا، در این گروه مشخصه UDDI برای یافتن سرویس های وب گنجانده شده است .

- ضمانت های سرویس[۳]

سرویس های وب نباید فقط به سادگی پیام های XML را رد و بدل کنند. این سرویس ها باید تضمینی برای سرویس گیرنده ایجاد کرده که اولا پیام به نحوی ایمن انتقال داده می شود، ثانیا این که سرویس گیرنده باید حتما پاسخی دریافت کند، حتی اگر در نقطه ای از جریان کار، نقصی پیش آمده باشد. این گروه از مشخصه ها شامل مشخصه امنیت سرویس وب (برای تضمین رسیدن پیام ها) مشخصه پیام رسانی مطمئن سرویس وب (برای تضمین رسیدن پیام ها در شبکه های ناپایدار و بدون قابلیت اطمینان) و تعداد زیادی از مشخصه های مربوط به تراکنش است .

ترکیب سرویس[۴]

مجموعه گسترده ای از مشخصه های پروفایل پایه WS-I را نمی توان به طور کامل در هر سرویس وب پیاده کرد. به همین خاطر، توسعه دهندگان باید مشخصه های مهم و مناسب را انتخاب و در سرویس پیاده کنند. برای تامین این هدف، سرویس ها دارای ویژگی ترکیب سرویس هستند که به توسعه دهندگان اجازه می دهد مشخصه های مختلف را برای هر سرویس انتخاب کرده و آن ها را در سند WSDL گرد آوری و ثبت کنند. در ادامه تعدادی از مهمترین مشخصه های سرویس های وب و اهداف آن را بیان می کنیم:

1 WS-Policy
2 Client
3 Service Assurances
4 Service Composition

- امنیت سرویس وب[1] : مشخصه ای جامع از مجموعه ای از فناوری های متداول امنیتی از جمله امضاهای دیجیتال و رمز گذاری مبتنی بر نشانه های امنیتی شامل گواهی های X.509 است.

- سیاست گذاری سرویس وب[2] : به سرویس های وب امکان می دهد نیازها، ترجیحات[3] و توانایی های خود را براساس مجموعه ای از فاکتورها بیان و مستند سازی می کند کنند. البته تمرکز بیشتر با فاکتورهای امنیتی است . برای مثال سیاست گذاری یک سرویس وب می تواند شامل نیازهای امنیتی آن، نظیر رمز گذاری و امضای دیجیتال بر اساس یک گواهی X.509 باشد .

- آدرس دهی سرویس وب [4]: نقاط نهائی سرویس را در یک پیام مشخص کرده و امکان بروز رسانی این نقاط را در مواردی که پیام بین دو یا چند سرویس منتقل می شود را فراهم می سازد. این مشخصه به طور گسترده ای در حال حاضر جایگزین مشخصه قدیمی تر مسیر دهی سرویس وب[5] می شود.

- پیام رسانی سرویس وب[6] : امکان پشتیبانی از سایر پروتکل های کانال ارتباطی نظیر TCP را در کنار HTTP برای سرویس وب فراهم می سازد. این مشخصه ساخت و توسعه نرم افزارهای پیام رسانی، از جمله برنامه های کاربردی غیر همزمان که با استفاده از SOAP روی HTTP ارتباط برقرار می کنند، را تسهیل می کند.

- مکالمه ایمن سرویس وب[7] : با استفاده از نشانه های امنیتی[8] ارتباطات مورد اعتماد برای جلسات کاری فراهم می کند .

-پیام رسانی مطمئن سرویس وب[9] : مکانیزم هایی برای تضمین اطمینان جهت رسیدن پیام حتی در صورتی که یک یا چند سرویس در زنجیره سرویس ها قابل دسترس نباشند را فراهم می سازد. این مشخصه شامل روش هایی برای اعلام رسیدن پیام است، به طوری که فرستنده بتواند بفهمد که آیا گیرنده در دریافت پیام موفق بوده است یا نه.

با معرفی و ثبت مشخصه های جدید و بهبود مشخصه های قبلی ، مشخصه های سرویس های وب دائما در حال تکامل هستند. البته، مجموعه مشخصه های پایه ای که بیان شد، احتمالا برای مدتی به عنوان زیر بنای اصلی مشخصه های سرویس های وب باقی خواهند ماند، چرا که این مشخصه ها نیازهای اصلی و بنیادی برنامه های کاربردی سرویس گرا را پوشش می دهند .

1 WS-Seccurity
2 WS-Policy
3 Preferences
4 WS-Adressing
5 WS-Routing
6 WS-Messaging
7 WS-Secure Conversation
8 Security tokens
9 WS-Reliable Messaging

در ادامه جا دارد که از محصول [1] WSE نسخه دو مایکروسافت نام برده شود که مجموعه ای از ابزارهای مدیریت شده تحت[2] معماری نرم افزار دات نت را جهت پیاده سازی مشخصه های سرویس های وب برای توسعه دهندگان فراهم آورده است WSE جهت فراهم آوردن پشتیبانی بیشتری از زیرساخت ها(فراتر از آنچه که در حال حاضر به وسیله چارچوب کاری دات نت تامین می شود) برای راه حل ها ی معماری سرویس گرا ارایه شده است. همچنین به کمک WSE می توان زیر ساخت پردازشی برای میزبانی سرویس های وبی که مشخصه WS را پیاده سازی می کنند، فراهم نمود. برای مثال WSEبه شما امکان می دهد که به آسانی سرویس های وبی را بسازید که رمز گذاری و امضاهای دیجیتال را روی درخواست ها و پاسخ های سرویس وب اعمال می کنند. در نهایت WSEیک ابزار بهره وری است که برای هدایت توسعه دهندگان دات نت به سمت نسخه آینده ایندیاگو[3] طراحی شده است ایندیاگو از محصولات آینده مایکروسافت است که پشتیبانی یکپارچه ای را برای برنامه های کاربردی پیام رسان و سرویس گرا فراهم می کند.

WSEیک محصول در حال تکامل است و بنابراین در حال حاضر همه مشخصه های سرویس های وب را پشتیبانی نمی کند، ولی بسیاری از مشخصه های مهم نظیر امنیت و سیاست سرویس های وب پشتیبانی می شوند . به خاطر اینکه معماری سرویس گرا تحت تاثیر مجموعه ای از استانداردها و مشخصه های فنی است که خودشان در حال تغییر هستند لذا می بایست که نسخه های WSE برای هماهنگی با نسخه های جدید این استانداردها و فناوری ها باید چرخه انتشار انعطاف پذیری داشته باشند. چارچوب معماری دات نت جدا کند، تا به این ترتیب بتواند انتشار نسخه های این محصول را با انعطاف پذیری بیشتری برنامه ریزی کند.

5-تعریف معماری سرویس گرا

دربخش قبل به سرویس های مولفه یا دانه ریز پرداخته شد که در ادامه توانستیم به سرویس های وب و مشخصه های مختلف آنها را بپردازیم. اکنون هر گاه که یک سیستم اطلاعاتی توسعه می یابد نیازمند بکار گیری معماری خاصی داشته تا بتواند سرویس های مولفه را با هم تلفیق و در نتیجه سیستم ها و یا در حقیقت سرویس های دانه درشت را پدیدار نماید. در هر صورت برای بکار گیری سرویس های مولفه نیازمند رویکرد های خاصی بوده که به عنوان معماری سرویس گرا از آنها

1 Web Services Enhancements (WSE) 2.0
2 .NET
3 Indigo

یاد شده واصلی ترین خصوصیت آنها نگاه خارجی به مولفه و مولفه ها بوده در همه آنها بدون توجه به درون هر سرویس مولفه ارتباطات بیرونی ملاک بوده و مورد توجه قرارمی گیرد.

تعاریف گوناگونی از معماری سرویس گرا ارائه شده است که از جمله آنها می توان به تعاریف زیر اشاره کرد:

- چارچوبی وسیع و استاندارد که سرویس ها در آن ساخته، استقرار و مدیریت می شوند و هدفش افزایش چابکی زیر ساخت های فناوری اطلاعات در جهت واکنش سریع به تغییرات در نیازهای کسب و کار می باشد.

- معماری سرویس گرا شامل سیاست ها، تجارب و چارچوب هائی است که کارکردهای سیستمی را قادر می سازد بصورت مجموعه ای از سرویس های توزیع شده در اندازه های مورد نظر سازمان تعریف شوند. این سرویس ها با کمک تعریف یک واسط استاندارد از پیاده سازی مجزا شده اند.

- معماری سرویس گرا یک محصول نیست بلکه پلی است بین کسب و کار و فناوری به کمک مجموعه ای از سرویس های متکی بر فناوری که دارای قوانین، استانداردها و اصول طراحی مشخص هستند.

- مجموعه قوانین ، سیاستها و چهارچوبهایی که نرم افزارها را قادر می سازد تا عملکرد خود را از طریق مجموعه سرویس های مجزا و در عین حال مربوط به هم در اختیار سایر درخواست کنندگان قرار دهند تا بتوانند بدون اطلاع از نحوه پیاده سازی و تنها از طریق رابطهای استاندارد و تعریف شده ، این سرویس ها را پیدا و فراخوانی نمایند.

- معماری سرویس گرا روشی برای ساخت سیستمهای توزیع شده ای است که در آنها عملکرد سیستم به صورت سرویس در اختیار کاربران و یا سایر سرویس ها قرار می گیرد.

از دیگرتعاریف ارائه شده می توان به "واحدهای نرم افزاری آماده در شبکه [1] یا سرویس های در سطح کسب و کار[2] " اشاره کرد.

در حال حاضر ، فناوری سرویس های وب و پیاده سازی نمونه های موفق از آن، نشان داده است که معماری سرویس گرا می تواند به عنوان راه حلی عملی و دست یافتنی در طراحی سیستمهای جدید و یکپارچه سازی سیستمهای بزرگ موجود ، مطرح گردد. به این ترتیب ذکر تفاوت سرویس های وب و معماری سرویس گرا در اینجا لازم به نظر می رسد:

سرویس های وب مجموعه ای از فناوری ها همچون XML ، WSDL و UDDI می باشند که امکان ارائه راه حل و برنامه نویسی جهت رفع مشکلی خاص را فراهم می نماید. در حالی که معماری سرویس گرا یک معماری است و لی از مجموعه

[1] Network-available Software Unit
[2] Business-level services

مشخصی از فناوری ها فراتر می باشد. در حقیقت اگرچه معماری سرویس گرا بر اساس این فناوری ها راه حل ارائه می نماید ، اما در عین حال مستقل از هر یک از آنها است. آنچه اهمیت دارد تعریف سرویس به عنوان مهمترین عنصر این معماری می باشد.

سرویس، رفتار قرادادی تعریف شده ای ا ست که هر ماژولی می تواند آنرا جهت استفاده از سایر بخش های سیستم تهیه و پیاده سازی نماید.

در این معماری ، همه توابع به عنوان سرویس تعریف می شوند. این توابع شامل وظایف کسب و کار[1] و تراکنش های کسب و کار[2] می باشند که تراکنش های کسب و کارخود شامل توابع سطح پایین[3] و توابع سیستمی سرویس ها[4] هستند. سرویس ها بصورت مستقل طراحی و پیاده سازی شده و به عنوان جعبه سیاه عمل می نمایند. ماژول های دیگر در خارج از این ماژول نیازی به دانستن نحوه انجام کار در این سرویس را ندارند و تنها به نتیجه آن نیازمندند.

ماژول ها، سرویس های خود را از طریق رابط ها[5] در اختیار ماژول ها دیگر قرار میدهند که این رابطها قابل دستیابی و فراخوانی هستند، بدون اینکه محل قرار گیری آنها اهمیت داشته باشد (رابطهای محلی یا دور) . در ضمن این رابطها می توانند به همان نرم افزار کاربردی یا به آدرسی در محل دیگری از شبکه مرتبط باشند .

رابط ها به عنوان کلیدی در برقراری این ارتباط ها هستند و از طریق آنها نوع پارامترهای ورودی و نتایج (خروجی) مشخص می گردد.

بعلاوه با ایجاد قابلیت توزیع شدگی به شکلی مناسب و بدون وابستگی زیاد ، می توان سرویس ها را در قسمتهای مختلف شبکه و بصورت بعضا تکراری (مخصوصا برای سرویس های مهم) قرار داد تا این سرویس ها همیشه در دسترس بوده و در صورت زیاد شدن درخواست ها بتوان با استفاده از تکنیکهای Load Balancing به تمامی آنها به‌خوبی پاسخ داد.

معماری سرویس گرا رویکردی است برای ساخت سیستم های توزیع شده که کارکردهای نرم افزاری را در قالب سرویس ارائه می کند. این سرویس ها هم توسط دیگر نرم افزارها قابل فراخوانی هستند و هم برای ساخت سرویس های جدید مورد استفاده قرار می گیرند، این رویکرد برای یکپارچه سازی فناوری ها در محیطی که انواع مختلفی از سکوهای نرم افزاری و سخت افزاری وجود دارد ایده آل است.

---

[1] Business functions
[2] Business transactions
[3] Low-level functions
[4] System service functions
[5] Interface

معماری سرویس گرا از دیدگاه های مختلف قابل بررسی است، هر فرد یا ذی نفع بر طبق جایگاه خود تصویری از این معماری دارد، در ادامه از سه دیدگاه کارشناسان کسب و کار، معماران و طراحان سیستم های اطلاعاتی مورد بررسی قرار می گیرد.

کارشناسان عملیاتی یا کسب و کار: مجموعه ای از سرویس ها که سازمان مایل به ارائه آنها به مشتریان یا شرکاء خود است. (تعریف سرویس کسب و کار)

معماران : سبکی از معماری که حاوی قوانین، الگوها و ضوابطی است که منجر به ایجاد خصایصی نظیر پیمانه ای بودن ، بسته بندی ، اتصال سست ، استفاده مجدد و ترکیب پذیری شده و از نظر ساختار از یک ارائه دهنده سرویس و یک درخواست کننده سرویس تشکیل شده است.

طراحان و پیاده سازها: یک سبک(مدل) برنامه نویسی که از استانداردهائی مانند (معماری سرویس گرا UDDI، WSDL) و فناوری هائی نظیر سرویس های وب استفاده می کند و قابلیت تعامل پذیری بین مولفه های نرم افزاری را بدون توجه به سکو و فناوری پیاده سازی آنها پشتیبانی می کند.

برای معماری سرویس گرا تعاریف متنوع و بعضا مختلفی ارائه شده که هر کدام از نگاهی به تبین خصوصیات آن پرداخته اند، برای درک بهتر این مفهوم و آگاهی از کلیه برداشت ها و نگاه های موجود، در ادامه تعدادی از این تعاریف آورده شده است.

در جمع بندی از تعاریف معماری سرویس گرا به ویژگیهای مشترک زیر می توان اشاره نمود:

- هم راستای کسب و کار سازمان یا عملیات در فضای جنگ است
- هم موضوعی فنی است و هم نوعی سبک تفکر است
- مبتنی بر اتصال سست است و از پیام رسانی استفاده می کند
- قادر به ساخت سیستم های ترکیبی است
- مهمترین دستاورد آن انعطاف پذیری و چابکی فناوری اطلاعات در برابر تغییرات کسب و کار است.
- منجر به تعامل پذیری سامانه ها/سازمانها می گردد
- امکان ارائه یک سرویس با واسطه های متنوع را محقق می سازد
- زیرساخت ارتباطی برای این معماری می بایست مستقل از پروتکل های لایه های زیرین باشد.

مقایسه میان ویژگیهای معماری سرویس گرا با رهیافت های گذشته در جدول 1، نشان داده شده است.

جدول 1- مقایسه سرویس گرائی و رویکرد های گذشته

معماری سرویس گرا	رویکرد های گذشته
ارتباطات ارزش آفرین است	ارتباطات هزینه بر بود
مبتنی بر فرآیند	مبتنی بر کارکرد
ساخت برای تغییر	ساخت برای بقا
توسعه تدریجی	تولید یکمرتبه
فدراسیون نرم افزارها	نرم افزارهای تعامل ناپذیر
مستقل از سکو	تک سکو
اتصال سست	اتصال محکم
پیام محور	شیء گرا

6-سرویس های ترکیبی (ارکستریشن[1] و کاریوگرافی[2])

اکنون در این بخش به مهمترین مفهوم در معماری سرویس گرا پرداخته که توجه به آن می تواند مفهوم سرویس را بدرستی در محیط ها و یا سیستم های توزیع شده بکار گرفته و شفافیت و همروندی را موجب شود. دو واژه پر کاربرد در حوزه کسب و کار و معماری سرویس گرا که معمولا به جای هم اشتباه گرفته می شوند، ارکستریشن و کاریگرافی نام دارند. ارکستریشن مربوط به ترتیب اجرای سرویس ها در فرآیند است، ارکستر اصلی مجموعه ای از سرویس ها را فراخوانی می کند تا نتیجه مورد نظر حاصل شود و فرآیند تکمیل گردد، ممکن است سرویس های خارج سازمان نیز در این راستا فراخوانی و استفاده شوند، این کار با کمک موتور فرآیند (رهبر ارکستر) محقق می شود. در عوض کاریگرافی به فرآیندهایی گویند که بدون موتور فرآیند اقدام به تبادل پیام کرده و ترتیب و توالی پیامهای مبادلاتی را خود بازیگران ثبت و کنترل می کنند.

بنابراین ارکسترال به معنای وجود یک موتور فرآیندی است که ترتیب و توالی را کنترل کرده و از شرکاء داخلی یا خارجی برای

[1] Orchestral services
[2] Choreography

انجام کارها استفاده می نماید. نمونه این مدل سیستم مدیریت فرآیندهای کسب و کار[1] است که فرآیندها در موتور فرآیندی اجرا می شوند.

کاریگرافی به معنای پردازش های توزیع شده بین چند فرآیند است که بدون یک رهبر مرکزی با هم تعامل دارند یا چندین موتور فرآیندی که در کنار و هم سطح هم اجرا می شوند و با همکاری هم هدفی را محقق می سازند. نمونه این موضوع در پردازش های توزیع شده و یا فعالیت های بین سازمانی که هر دو طرف با مشارکت هم به دنبال یک هدف هستند دیده می شود.

مهمترین تفاوت کاریگرافی و ارکستریشن در داشتن مالک و کنترل کننده مرکزی است(شکل 1)، ارکستریشن درشرایطی کاربرد دارد که مجموعه ای از فعالیت ها زیر نظر یک عامل انجام شود و آن عامل کلیه قوانین مربوط به گردش کار و شرط ها را در خود داشته باشد، در عوض کاریگرافی برای مواقعی مناسب است که مجموعه فعالیت ها بین دو شریک تقسیم شده و هیچکدام به تنهایی قادر به کنترل و مدیریت جریان کار نبود و نیاز به کنترل دو طرفه باشد.

شکل 1- وجود یا عدم وجود هدایت کننده مرکزی تفاوت میان ارکستریشن و کاریوگرافی

7-مفاهیم اصلی و چرخه حیات درمعماری سرویس گرا

[1] Business Procees Management System(BPMS)

سیستم های اطلاعاتی به سرعت در حال رشد هستند؛ سازمان ها نیازمند پاسخگویی سریع به نیازمندی های جدید کسب وکار هستند. این در حالی است که معماری های نرم افزاری موجود به حد نهایی قابلیت های خود رسیده اند. معماری مبتنی بر سرویس معماری سرویس گرا قدم تکاملی بعدی برای کمک به سازمان ها جهت مدیریت چالش های پیچیده است.

معماری مبتنی بر سرویس حالت بلوغ یافته معماری مبتنی بر مولفه، طراحی مبتنی بر واسطه (شی گرا) و سیستم های توزیع شده است. در معماری مبتنی بر مولفه عملکرد کلی به وظایف کوچک تری تقسیم می شود که هر یک در یک مولفه بسته بندی خواهند شد. یک سیستم توزیع شده، تعمیمی از یک معماری مبتنی برمولفه است که به مولفه ئی که در موقعیت های فیزیکی مختلف وجود دارند، اشاره می کند.

مهم ترین مزیت معماری مبتنی بر مولفه سهولت در استفاده مجدد و تغییر هدف مولفه ی خاص و سهولت در امر نگهداری سیستم است. استفاده مجدد و تغییر هدف معمولاً مهم ترین پیشران های کسب و کار جهت استفاده از این نوع معماری در دهه 90 میلادی بوده است. بر اساس منطق معماری مبتنی بر سرویس، سیستم های نرم افزاری بزرگ می توانند از گردآوری مجموعه هایی از عملکردهای مستقل و قابل استفاده مجدد تشکیل گردند.

برخی از این عملیات می تواند از طریق سیستم های موجود و یا سیستم های دیگر فراهم گردد ولی سایر عملیات لازم باید پیاده سازی شوند. هر سرویس امکان دسترسی به مجموعه خوش تعریفی از عملیات را می دهد. سیستم به عنوان یک کل به صورت مجموعه ای از تعاملات بین این سرویس ها طراحی می شود. معماری مبتنی بر سرویس، سرویس هایی را که سیستم از آنها تشکیل شده را تعریف می کند و تعاملات لازم بین سرویس ها جهت ارائه رفتار مشخص را توصیف می کند و در نهایت سرویس ها را به یک یا چند پیاده سازی در فناوری های خاص تصویر می کند.

معماری سرویس گرا بر اساس استفاده از اشیاء و مولفه توزیع شده است و تکامل بعدی در محیط های محاسبه ای است. این معماری در حال حاضر مدل مرجع استانداردی ندارد. اما پیاده سازی های موجود مفاهیم مشترکی را مورد استفاده قرار می دهند که در ادامه این مفاهیم پایه مورد بررسی قرار می گیرند.

- سرویس

یک سرویس رفتار تعریف شده قراردادی است که می تواند به وسیله یک جزء برای استفاده جزء دیگر پیاده سازی شده و فراهم شود.

- شرح سرویس

شرح سرویس پارامترهای فنی، قیود و سیاست هایی را شامل می شود که قالب های لازم برای فراخوانی سرویس را تعریف می کند. هر سرویس باید شامل تعریف سرویس در قالب استاندارد باشد. این موضوع کاربردها و کاربران انسانی را قادر می سازد تا با بررسی شرح سرویس و تعیین موضوعاتی نظیر این که سرویس چه کاری انجام می دهد، چگونه به آن انتصاب می یابند و این که چه پروتکل های امنیتی (در صورت وجود) باید به همراه آن مورد استفاده قرار گیرد، از آن سرویس استفاده کند.

این اعلان همچنین ممکن است شامل جزئیاتی در مورد هر فرایند ضمنی و دیگر واژه های کاری و قانونی باشد که ممکن است در زمان فراخوانی سرویس اتفاق بیفتد. به عنوان مثال، اگر یک استفاده کننده سرویس، سرویسی را فراخوانی کند که یک درخواست خرید را برای فراهم کننده سرویس ارسال نماید، و اجرا موفقیت آمیز باشد، این موضوع ممکن است منجر به مسئولیت مالی نسبت به فراهم کننده سرویس یا بخش قانونی دیگر می شود.

در حالیکه طبیعت سرویس ها ممکن است تغییر کند، استاندارد مشترکی جهت اعلان یک سرویس هنگام تهیه یک زیرساخت مطلوب است. دو نمونه از چنین استانداردهای موجود ebXML و WSDL هستند.

- اعلان و یافتن سرویس ها

شرح سرویس باید به شیوه ای قابل دسترسی در اختیار کاربران بالقوه قرار گیرد که به این امر اعلان سرویس اطلاق می شود. یابش، زمانی انجام می شود که یک مشتری بالقوه اطلاعاتی در مورد وجود یک سرویس، پارامترهای قابل اعمال و واژگان آن به دست آورد. یافتن، بحث تصدیق هویت جهت اجرای سرویس را شامل نمی شود؛ گرچه این جزئیات ممکن است در الگوی یافتن قرار گیرد.

مولفه ی اعلان و یافتن در معماری سرویس گرا به شیوه های مختلف از جمله استفاده از روش ثبات / مخزن و یا روش دایرکتوری سرویس قابل پیاده سازی هستند.

- پیاده سازی به روش ثبات / مخزن

یک ثبات / مخزن جزئی است که در آن کاربران امکان ذخیره و مدیریت سرویس های لازم برای عملکرد سازمانشان را خواهند داشت. این موضوع شامل سرویس هایی است که تسهیم بین بیش از یک کاربر (همانند xml schemas و شرح web-service) را فراهم می آورد که به ثبات به گونه ای منتسب می شود که ثبات در مورد تمامی رویدادهای قابل ارزیابی نسبت به محصولات در مخزن اطلاع دارد.

- پیاده سازی به روش دایرکتوری

دایرکتوری یک واسط است که اطلاعات انتساب به محصولات را فراهم می آورد. افرادی که مالک محصولات هستند و یا آنها را کنترل می کنند، می توانند یک مدخل به دایرکتوری باز کرده تا به محصولات ارجاع داده و خود انتسابات به آن را توضیح دهند. دیگران ممکن است این اطلاعات را بازیابی کرده و از آن جهت انتساب به محصولات استفاده کنند. مهم ترین مشکل دایرکتوری این است که هیچ کنترل یا اطلاع رسانی در صورت حذف، تغییر و جایگزین شدن یک محصول انجام نمی دهد و قادر به اعلام این رویدادها به کاربران نیست.

هر دو پیاده سازی دایرکتوری و ثبات / مخزن امکان سازگاری با یکدیگر را دارند. به این معنا که عملکرد اجازه می دهد که محتوا از یک پیاده سازی توسط یک پیاده سازی دیگر مورد ارجاع قرار گیرد.

پیاده سازی ها دایرکتوری و ثبات / مخزن دارای چندین استاندارد است که رایج ترین آنها عبارتند از:

1. OASIS ebXML Registry-Repository Technical Specifications16
2. OASIS Universal Description and Discovery Interface (UDDI ،Technical Specification17.2

- خصوصیات مدل داده ای مرتبط

در هنگام فراخوانی یک سرویس، پارامترهای مشخصی ممکن است برای انجام درخواست سرویس مورد نیاز باشد. سرویس همچنین ممکن است پارامترهایی را به کاربر سرویس بازگرداند. W3C WSDL یک نمونه شناخته شده از پیاده سازی این بخش است.

8-اصطلاحات و مفاهیم رایج در معماری سرویس گرا

- فراهم کننده سرویس: یک موجودیت نرم افزاری که خصوصیات سرویس را پیاده سازی می کند.
- درخواست کننده سرویس: یک موجودیت نرم افزاری که یک فراهم کننده سرویس را فراخوانی می کند، به طور سنتی این مورد به عنوان "کلاینت" شناخته می شود؛ اما یک درخواست کننده سرویس می تواند یک کاربر برنامه کاربردی و یا سرویس دیگر باشد.

• موقعیت یاب سرویس: یک نوع خاص از فراهم کننده سرویس که به عنوان یک ثبات عمل می کند و امکان جست وجوی واسطه های فراهم کننده سرویس و موقعیت سرویس را می دهد.

• واسط سرویس: یک نوع خاص از فراهم کننده سرویس است که می تواند درخواست سرویس را به یک یا چند فراهم کننده سرویس منتقل کند.

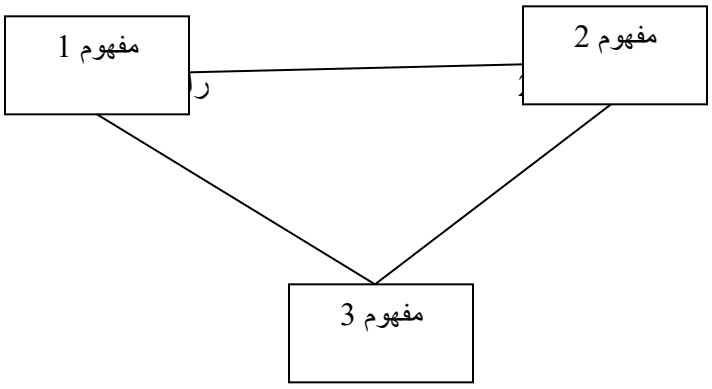
- چارچوب معماری سرویس گرا

- زیرساخت معماری سرویس گرا

- نقشه یا مدل مفهومی سرویس گرائی

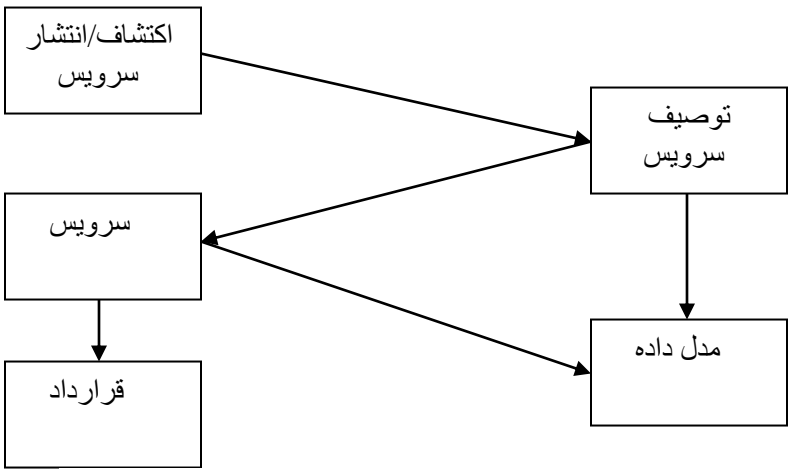
شکل های 3 و 4 و 5 نقشه هایا مدل های مفهومی هستند که مفاهیم پایه معماری سرویس گرا را مستقل از معنی و فناوری ها تشریح می کنند. نقشه های مفهومی غیر رسمی بوده و نمایش گرافیکی از مفاهیم و رابطه بین آنها را شامل می شود. شکل 2 نمونه ای از یک نقشه یا مدل مفهومی است که در حقیقت یک شبکه معنی از مجموعه ای از مفاهیم و ارتباطات بین آن ها است. ارزش مدل های مفهومی تشریح مفاهیم پایه یک هستان شناسی مستقل از معنی از یک طرف و مستقل از فناوری از طرف دیگر است.

رابطه 1



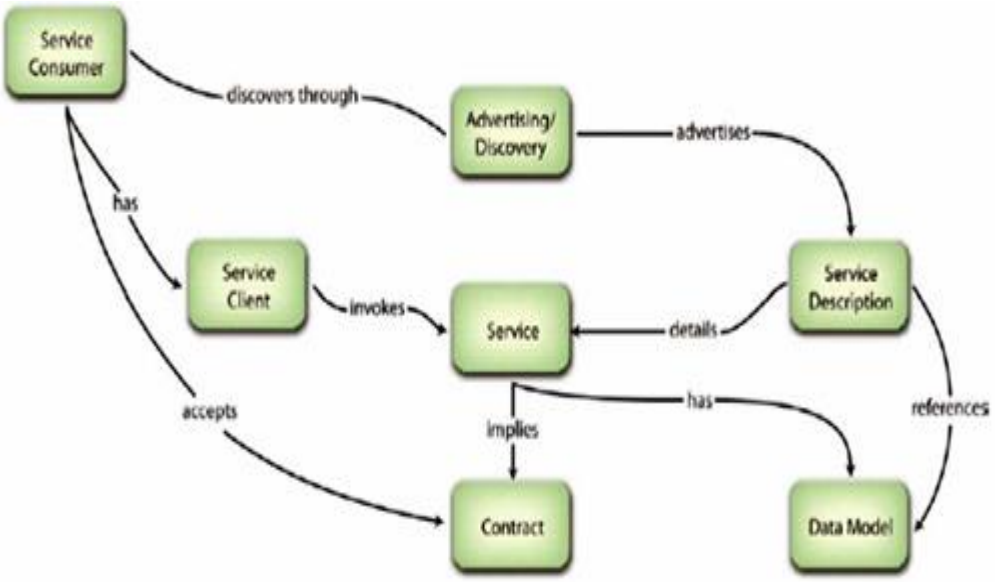
شکل2- مثالی از نقشه مفهومی

اغلب معماری هایی که معماری سرویس گرا نامیده می شوند، شامل یک فراهم کننده سرویس، یک کاربر سرویس و برخی زیرساخت های پیام رسانی هستند.



شکل 3- مدلی از فعالیت های اصلی معماری سرویس گرا

- مفاهیم اختیاری و زیرساخت های معماری سرویس گرا اشتراکی



شکل4-مفاهیم اختیاری برای معماری سرویس گرا و نمایش تعامل آنها با مفاهیم پایه این معماری

جهت اجرای تعهدات در زمان اجرا، اغلب معماری سرویس گرا ها ممکن است شامل مفاهیمی اضافه بر آنچه در شکل 3 نشان داده شده است، باشند. مفاهیم دیگر کاربران سرویس، کلاینت سرویس، قرارداد پذیرش سرویس و فراخوانی سرویس هستند. فراخوانی یک سرویس یا وجود کاربر سرویس در مدل معماری سرویس گرا الزامی نیست. فراهم کننده یک سرویس،

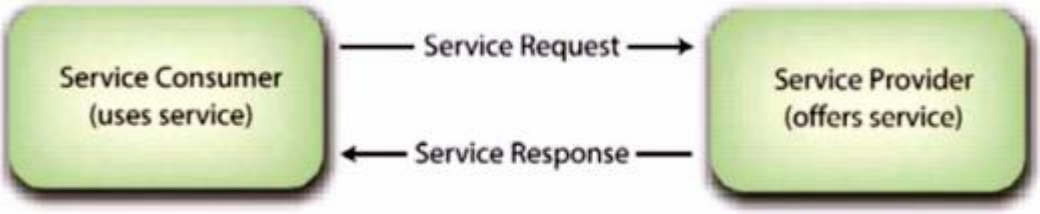
قرارداد سرویس جهت استفاده آن و شرح سرویس را ارائه می کند. شرح سرویس به مدل داده های مرتبط با فراخوانی سرویس ارجاع می کند.

شرح سرویس از طریق یکی از چندین روش اعلان می شود. کاربر سرویس خصوصیات سرویس را از طریق مکانیزم اعلان شناسایی می کند. شناسایی، پذیرش قرارداد را شامل نمی شود. همچنین فراخوانی سرویس را نیز القا نمی کند. این امر صرفاً عملی برای پیدا کردن اطلاعات در مورد سرویس است. این امر لزوماً در یک عمل اتفاق نمی افتد و ممکن است به تعدادی از کارها نیاز داشته باشد. همچنین ممکن است از طریق وسایل غیر الکترونیکی صورت پذیرد.

شکل 4 سایر عملکردهای خاص اختیاری را حذف کرده و از پیاده سازی های خاص نظیر پیام رسانی معماری سرویس گرا P ، WDSL و ebXML خودداری می کند.

- الگوهای معماری سرویس گرا

شکل 5 یک الگوی ساده سرویس را نمایش می دهد. در جایی که یک فراهم کننده سرویس، سرویس را پیشنهاد می دهد و یک کاربر سرویس، از سرویس ها استفاده می کند. چندین نوع از پروتکل های ارتباطی ممکن است زوج درخواست/ پاسخ را مورد استفاده قرار دهد و روش های متنوعی نظیر سنکرون یا آسنکرون ممکن است استفاده شود. معماری سرویس گرا به هیچ پروتکل ارتباطی خاص محدود نمی شود. شکل 5 الگوی "درخواست – پاسخ" را نمایش می دهد.



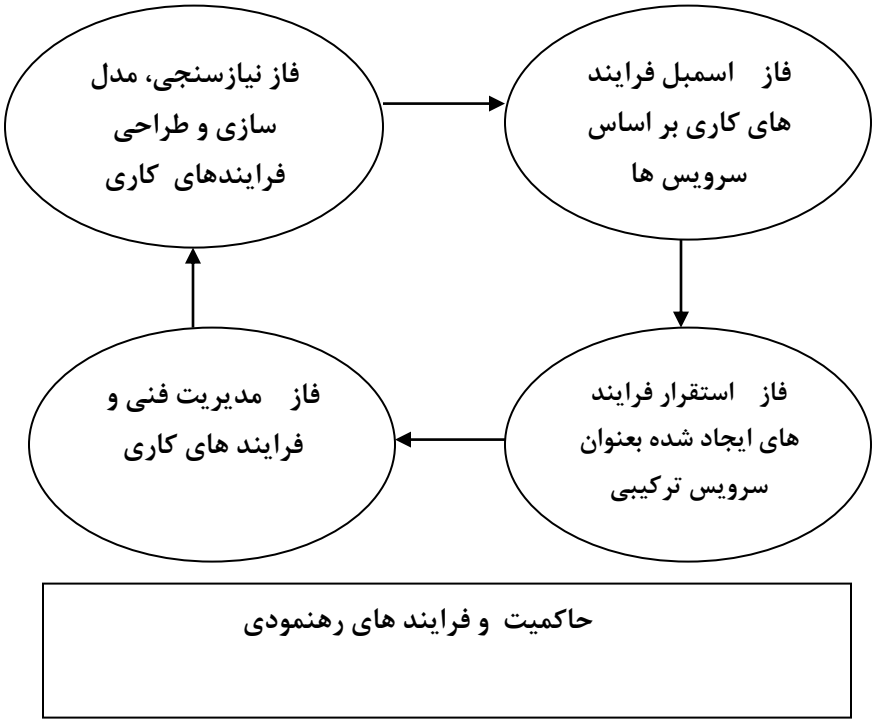
شکل5-الگوی پایه برای معماری مبتنی بر سرویس

- چرخه حیات معماری سرویس گرا

بر اساس طرح آی بی ام، برای معماری سرویس گرا می توان یک چرخه حیات در نظر گرفت(شکل 6). در فاز "مدل" نیازمندی های کسب و کار جمع آوری شده و فرایندهای کسب و کار آنها طراحی می شود. بعد از بهینه شدن فرایندها، از طریق کنار هم قرار دادن سرویس های موجود و سرویس های جدید این فرایندهای کسب و کار شکل می گیرد. سپس این

فرایند ها در یک محیط امن و با قابلیت تجمیع بالا مستقر می شوند. بعد از نصب فرایندهای کسب و کار، کاربران آن ها را هم از منظر فنی و هم از منظر فرایندی مورد نظارت و مدیریت قرار می دهند.

اطلاعات جمع آوری شده در فاز مدیریت به چرخه حیات بازخورد خواهد داشت تا بهبود پیوسته فرایندها را امکان پذیر سازد. در زیر همه این مراحل در چرخه حیات، حاکمیت و فرایندهایی هستند که رهنمود ها و افق های آینده را برای پروژه معماری سرویس گرا فراهم می کنند(شکل 6).



شکل 6- چرخه حیات معماری مبتنی بر سرویس

فاز 1- مرحله مدل سازی

فاز مدل با جمع آوری و تحلیل نیازمندی های کسب و کار آغاز می شود که بعداً برای مدل کردن، شبیه سازی و بهینه کردن فرایندهای کسب و کار مورد استفاده قرار می گیرند. فرایندهای کسب و کار حاصل برای طراحی سرویس های نرم افزاری مرتبط و سطوح سرویس جهت حمایت از این فرایندها مورد استفاده قرار می گیرند. در طول این فاز، مدلی جهت ایجاد درک مشترک بین کسب و کار و فناوری اطلاعات در فرایندهای کسب و کار، اهداف و خروجی ها استفاده می شود. به

علاوه این مدل می تواند این اطمینان را به وجود آورد که کاربردهای حاصل، نیازمندی های کسب و کار تعریف شده را براورده می سازد. این مدل همچنین می تواند مبنایی جهت اندازه گیری کارآیی کسب و کار باشد.

فاز2- مرحله گردآوری

در طول فاز گردآوری، کتابخانه سرویس های موجود می تواند جهت یافتن سرویس های مورد نظر و موجود در سازمان بررسی شود. در صورتی که سرویس مورد نظر یافت نشد این امکان وجود دارد که یک سرویس جدید ایجاد و پس از تست به مجموعه افزوده شود. هنگامی که سرویس های مورد نیاز فراهم شد، سرویس ها جهت پیاده سازی فرایندهای کسب و کار هماهنگ می گردند.

فاز3- مرحله استقرار

در طول فاز پیاده سازی، مقیاس و محیط زمان اجرا جهت تامین نیازمندی های سطوح سرویس به وسیله فرایندهای کسب وکار پیکربندی می شود. پس از پیکربندی یک فرایند کسب وکار، امکان پیاده سازی آن در یک محیط امن، مطمئن و مقیاس پذیر سرویس ها وجود خواهد داشت. محیط سرویس ها به گونه ای بهینه سازی می شود که علاوه بر اجرای مطمئن فرایندهای کسب وکار، امکان انعطاف پذیری جهت بروز کردن به طور پویا و در صورت تغییر نیازمندی های کسب وکار را فراهم می آورد. این رویکرد مبتنی بر سرویس همچنین هزینه و پیچیدگی نگهداری سیستم را نیز کاهش می دهد.

فاز4- مرحله مدیریت

فاز مدیریت شامل نظارت و نگهداری از زمان پاسخ و در دسترس بودن سرویس می شود. همچنین مدیریت منابع سرویس های زیرین در این فاز انجام می شود. درک کارایی زمان واقعی فرایندهای کسب وکار امکان ایجاد بازخورد ضروری به مدل فرایند کسب و کار جهت بهبود دائمی را فراهم می آورد. این کار همچنین مدیریت و نگهداری کنترل نسخه برای سرویس های تشکیل دهنده فرایندهای کسب و کار را شامل می شود. فاز مدیریت در نهایت امکان اتخاذ تصمیمات کسب و کار بهتر و سریع تر را فراهم می سازد.

فاز5 - مرحله حاکمیت و فرایندها

حاکمیت و فرایندها جهت موفقیت هر نوع پروژه معماری سرویس گرا ضروری هستند. جهت تخمین موفقیت، ممکن است یک مرکز تعالی در کسب وکار، برای پیاده سازی سیاست های حاکمیتی و دنبال کردن استانداردهای حاکمیتی بین المللی

جهت اهداف کنترلی برای اطلاعات و فناوری مرتبط ایجاد گردد. پیاده سازی سیاست های حاکمیتی قوی می تواند منجر به پروژه های معماری سرویس گرا موفق گردد.

9- خصوصیات اساسی جهت استفاده بهینه از سرویس ها

- دانه درشت بودن: عملکردها روی سرویس ها به طور متفاوت پیاده سازی می شوند تا کارآیی بیشتری را در برگیرند و بر روی مجموعه های داده ای بزرگ تر در مقایسه با طراحی مبتنی بر مولفه عمل می کند.
- طراحی مبتنی بر واسط: سرویس ها، واسط های مجزا تعریف شده را پیاده سازی می کنند. مزیت این امر آن است که چندین سرویس می توانند یک واسط مشترک را پیاده سازی کنند و یک سرویس می تواند چندین واسط را پیاده سازی کند.
- قابل یافت بودن: سرویس ها لازم است هم در زمان طراحی و هم در زمان اجرا قابل یافت باشند، نه تنها با شناسه یکتا بلکه همچنین با شناسه واسط و با نوع سرویس.
- نمونه منفرد: بر خلاف توسعه مبتنی بر جزء که از مولفه بر حسب نیاز نمونه هایی ایجاد می شود، هر سرویس یک نمونه منفرد و همواره در حال اجرا است که مجموعه ای از کلاینت ها با آن ارتباط برقرار می کنند.
- اتصال سست: سرویس ها با دیگر سرویس ها و کلاینت ها از طریق تبادل اطلاعات استاندارد xml با یکدیگر در ارتباط هستند؛ این ارتباط باعث کاهش وابستگی و جداسازی بر اساس پیام رسانی می شود.
- آسنکرون: به طور کلی، سرویس ها از رویکرد انتقال پیام آسنکرون استفاده می کنند. اما این امر ضروری نیست. در بعضی مواقع در پیاده سازی سرویس ها از انتقال پیام سنکرون نیز استفاده می شود.

10-مقیاس پذیری از طریق رفتار آسنکرون و صف بندی

بهتر است که از ماهیت آسنکرون در سرویس ها استفاده شود. با توجه به سربار انتقال اضافه و همچنین این انتظار که سرویس ها، ماهیتاً در فواصل فیزیکی دور از یکدیگر خواهند بود، کاهش زمان انتظار درخواست کننده برای پاسخ بسیار اهمیت دارد. از طریق آسنکرون کردن فراخوانی یک سرویس، با یک پیام بازگشت مجزا، به درخواست کننده امکان ادامه اجرا تا زمان فراهم شدن پاسخ داده می شود.

البته این به معنای اشتباه بودن رفتار سنکرون سرویس نیست، بلکه به این معنا است که رفتار سرویس آسنکرون مطلوب است، به خصوص در جایی که هزینه های ارتباطی زیاد است و یا تاخیر شبکه قابل پیش بینی نیست.

با استفاده از فراخوانی آسنکرون، به فراهم کننده این امکان داده می شود که از چندین رشته کاری جهت مدیریت چندین درخواست کلاینت استفاده کند. جهت اجرای فراخوانی آسنکرون، درخواست کننده باید نشانی بازگشت را به سرویس پیاده ساز یک واسط ارسال کند.

بطور کلی معماری سرویس گرا به مفهومی اشاره می کند که اساس آن تجربیات بدست آمده در تولید سیستم های نرم افزاری بر پایه دو اصل اساسی در صنعت مهندسی نرم افزار یعنی تولید نرم افزار با انسجام زیاد و در عین حال با اتصال کم است. بنابراین ایده های برنامه نویسی سرویس گرا ایده ای جدید نیست و قبلا ابطور ضمنی از آن استفاده شده است. اما جمع آوری بهترین تجربیات از تولید چنین سیستمهایی بصورت مجتمع و ناظر به وضعیت فناوری امروز بشر، که همان مفاهیم مطرح شده در معماری سرویس گرا است چیز جدیدی است. در زیر بصورت دقیق تر این بحث را ادامه می دهیم.

شرکتهای بزرگ نرم افزاری هم در جهت گام برداشتن برای رسیدن به این دو اصل،فناوریهایی را بوجود آوردند که به برنامه نویسان اجازه دهد تا به این دو هدف در تولید نرم افزارهای خود تا حد زیادی دست یابند. برای مثال می توان به فناوریهایی مانند CORBA , +COM و RMI و موارد دیگر، اشاره کرد. خوب پس مشاده کردید که موضوع برنامه نویسی سرویس گرا، مفهموم جدیدی نیست و این معماری تلاشی دیگر در جهت تولید نرم افزارهای با همبستگی و یا انسجام زیاد و در عین حال با چسبندگی و اتصال کم است. ممکن است بپرسید، پس چرا با وجودفناوریهای قدرتمندی چون CORBA,COM+,RMIچیز جدیدی بوجود آمد؟ مگرفناوریهای قبلی موفق نبودند؟ بله مهمترین اشکال در معماری های قدرتمندی چون موارد مذکور این بود که تولید کنندگان آنها سعی داشتند، کهفناوریخود را بر بازار غالب نمایند. رویایی که هرگز به حقیقت نمی پیوست .

بنابراین با توجه به این موضوع که این فناوری ها قادر به تعامل مناسب با یکدیگر نبودند عملا اصل انسجام زیاد بصورت خود بخود رد می شد. البته معماری های مذکور اشکالات دیگری هم داشتند که نسبت به مورد بالا از اهمیت کمتری برخوردار است که از جمله آنها می توان به عدم هماهنگی با اصول امنیتی مورد استفاده در اینترنت اشاره کرد. البته بعدها راه حل هایی هم برای این مشکل بوجود آمد (مانند (RPC Over HTTP) اما به این علت که از روز اول، در طراحی این فناوری ها این امر در نظر گرفته نشده بود، از کارایی مناسبی برخوردار نبودند .مفهموم همبستگی زیاد و در عین حال با چسبندگی و

اتصال کم، وقتی بخواهد در جهت ارزیابی یک سیستم نرم افزاری ، مورد استفاده قرار گیرد بسیار مبهم می شود. حتی اگرکسی بتواند ایده های همبستگی و چسبندگی را باهم ترکیب کند! برای جلوگیری از چنین ابهاماتی، شما می تواند از ویژگی های معماری سرویس گرا به عنوان یک راه برای ارزیابی میزان همبستگی و چسبندگی و اتصال یک سیستم نرم افزاری یا یک فناوری استفاده کنید. اگرچه مفاهیم مطرح شده در معماری سرویس گرا دقیقاً همان مفاهیم همبستگی زیاد و در عین حال چسبندگی کم نیستند، اما سیستمهایی که بر اساس معماری سرویس گرا طراحی و پیاده سازی شده اند، نشان داده اند که توانسته اند تا حد بسیار زیادی ویژگی های همبستگی زیاد و در عین حال چسبندگی کم را بخوبی در خود ایجاد و حفظ کنند.

11-ویژگی های سرویس و محاسبات سرویس گرا

محاسبات سرویس گرا (SOC) ، نمونه ای از محاسبات است که در آن طراحی و توسعه سیستم های کاربردی بر پایه سرویس به عنوان عنصر اساسی ، انجام می گیرد. سرویس ها عناصری هستند که مستقل از سکو بوده و در ساخت سیستمهای توزیع شده سریع و ارزان قیمت کمک می نمایند. همچنین سازمانها را قادر می- سازند تا توابع خود را از طریق زبانها و پروتکلهای بر پایه XML پیاده سازی و بر روی اینترنت یا اینترانت ارائه نمایند.

از آنجا که سرویس ها از طریق سازمانها و شرکتهای گوناگون تهیه می شوند و جهت دسترسی کاربران مختلف می بایست همواره در دسترس باشند ، رعایت ویژگیهای زیر ضروری می باشد:

- مستقل از فناوری باشند؛ به این معنا که بکارگیری و مکانیزم فراخوانی و پیدا کردن سرویس ها به راحتی و از تمام محیطها (سیستمهای عامل مختلف و زبانهای برنامه سازی گوناگون) میسر بوده و وابسته به سکو خاصی نباشد.
- وابستگی بسیار پایینی بین درخواست کننده و ارائه دهنده سرویس وجود داشته باشد و یا بعبارتی اتصال سست بین آن ها باشد. به این معنی که درخواست کننده نباید هیچ نیازی به دانستن ساختار داخلی و نحوه پیاده سازی سرویس داشته باشد. برای این منظور ، فراخوانی سرویس از طریق بکار گیری مکانیزم پیام بجای فراخوانی API انجام می گردد.

- درخواست کننده نباید نیازی به دانستن محل قرارگیری سرویس داشته باشد و بعبارتی معماری سرویس گرا می بایست

شفافیت مکان[1] را پشتیبانی نماید. به این ترتیب که محل قرارگیری سرویس و مشخصات آن در مخزنی قرار می گیرد و درخواست کننده ، محل و اطلاعات لازم را از طریق بازیابی آن از این مخزن بدست می آورد.

سرویس ها می توانند به دو شکل ساده و ترکیبی ارائه شوند. سرویس های ترکیبی ، سرویس هایی هستند که بر اساس بکارگیری چند سرویس ساده (یا ترکیبی) ایجاد می شوند. برای مثال ، ممکن است سیستم توزیع شده ای بر اساس چند سرویس ساده صدور صورتحساب ، ثبت سفارش ، مدیریت روابط مشتری و ... سرویس های ترکیبی گسترده تری در ارتباط با کسب و کار ای خاص ایجاد نماید.

سیستم های ساخته شده بر اساس سرویس، ترکیبی از سرویس های مستقل هستند که عملکردهای خود را از طریق رابطهای تعریف شده ای در اختیار کاربران (بالقوه) خود قرار میدهند. زبان WSDL[2] از جمله راه هایی است که بطور گسترده برای تعریف این رابطها بکار می رود تا بوسیله آن جزئیات لازم برای اتصال درخواست کننده به ارائه دهنده سرویس تعریف شود.

12- نرم افزار به‌عنوان سرویس

اصل ارائه شده " نرم افزار- بعنوان- سرویس" از محاسبات سرویس گرا ، بر اساس مدل ASP مطرح گشته است. ASP هویت سوم شخصیست که بکارگیری سرویس های نرم افزاری و دسترسی مشتری را به بسته نرم افزاری از طریق شبکه، مدیریت و میزبانی می نماید.به‌عبارتی ASP ها راهی برای رفع نیازهای IT شرکتها از طریق واگذاری بخشی از این نیازها یا تمامی آنها به بیرون از سازمان می باشند.

برای این منظور ASP با استفاده از زیر ساختهای خود ، ارتباط بین مشتری و نرم افزار ارائه شده را برقرار کرده و دسترسی وی به داده ها و توابع موجود را بصورت در دسترس (online)، مدیریت می نماید.

اگرچه نظریه "نرم افزار بعنوان سرویس" اولین بار توسط ASP ارائه شد ، اما مشکلات این روش باعث ایجاد کدهایی می شد که معمولا قابل استفاده مجدد نبوده و محدود به مشتری خاص می بود ، بعبارتی وابستگی زیادی بین سرویس ارائه شده و سیستم استفاده کننده بوجود می آمد.

[1] Location Transparency
[2] Web Service Description Language

معماری سرویس گرا اجازه میدهد تا نظریه "نرم افزار بعنوان سرویس"، گسترش یافته تا از طریق آن بتوان پردازشها و تراکنشهای پیچیده را بعنوان سرویس هایی با قابلیت استفاده مجدد ارائه کرد و به این ترتیب سیستمها را مستقل از سرویس ها طراحی و تولید نمود.

13-طراحی نرم افزار سرویس گرا

معماری سرویس گرا شیوه ای منطقی برای طراحی سیستمهای نرم افزاری است که از طریق آن سرویس هایی جهت ارائه به کاربران یا سایر سرویس های توزیع شده بر روی شبکه ایجاد می شود و این سرویس ها از طریق رابط های تعریف شده در دسترس می باشند.

معماری سرویس گرای مقدماتی ، راهی برای تبادل اطلاعات بین عاملهای نرم افزاری بوسیله پیام تعریف می نماید. این عاملها ، درخواست کننده سرویس (مشتری) و یا تهیه کننده سرویس می باشند. علاوه بر این دو، عامل دیگری بعنوان عامل کشف سرویس نیز وجود دارد. در معماری سرویس گرا معرفی سرویس ها و همچنین نحوه ارتباط این سه شرکت کننده نیز اهمیت دارد. این ارتباطات عبارتند از : منتشر کردن سرویس

پیدا کردن سرویس و متصل شدن به سرویس.

در یک سناریو بر پایه سرویس ، تهیه کننده ، سرویس را پیاده سازی کرده و از طریق شبکه به ارائه توضیحات آن سرویس برای درخواست کننده یا عامل کشف سرویس می پردازد. در خواست کننده معمولا درخواست پیدا کردن سرویس را به عامل کشف سرویس میدهد تا از طریق آن به توضیحات ارائه شده سرویس و محل آن دسترسی پیدا کند. سپس با بکارگیری این اطلاعات به تهیه کننده سرویس متصل شده و از سرویس ارائه شده استفاده می نماید.

بدین ترتیب ، سرویس بعنوان ماژول نرم افزاری پیاده سازی شده ای که توسط یک رابط تعریف شده مستند گردیده است و نه تنها بوسیله خود تهیه کننده بلکه توسط سایر عواملی که از نحوه پیاده سازی آن خبر ندارند ، نیز قابل استفاده و فراخوانی است. این ویژگی جعبه سیاه بودن سرویس از اصل ماژولاریتی در مهندسی نرم افزار به ارث رسیده است. البته سرویس با تعاریفی مانند ماژول ، ماژول نرم افزاری یا شی تفاوت دارد ، زیرا نه تنها در سطح برنامه ها و نرم افزارهای کاربردی ، بلکه در سطح کسب و کار و حتی مابین سازمانها نیز قابل بکارگیری و استفاده مجدد می باشد.

در واقع سرویس ها، نمایش عملکرد معنی داری از کسب و کار هستند که می توانند بنا به نیاز مشتری در سرویس ها و توابع بزرگتر یا جدید بکار گرفته شوند.

رابطها به سادگی مکانیزمی جهت برقراری ارتباط بین سرویس و نرم افزارها یا سایر سرویس ها ایجاد می نمایند. از لحاظ تکنیکی، رابط سرویس ها ، توضیحاتی در مورد نام و امضاء متدهای یک سرویس هستند که توسط درخواست کننده ، قابل فراخوانی می باشند.

14-معماری سرویس گرای توسعه یافته

معماری سرویس گرای مقدماتی به همه مسائل موجود در یک معماری سرویس گرا نمی پردازد. از جمله این مسائل، مدیریت، هماهنگ سازی سرویس ها ، متناسب کردن آنها ، امنیت ، مدیریت تراکنشها و ... می باشد. این نکات می بایست در معماری سرویس گرای توسعه یافته در نظر گرفته شده است.

معماری مقدماتی در لایه پایینی این معماری لایه ای قرار گرفته است. لایه ترکیب سرویس در معماری توسعه یافته ، شامل توابع و نقشهای لازم برای یکپارچه کردن چند سرویس بعنوان سرویس ترکیبی می باشد. سرویس ترکیبی بدست آمده ، توسط ترکیب سرویس[1] بعنوان یک سرویس مقدماتی استفاده می گردد و یا توسط درخواست کنندگان سرویس بکارگرفته می شود. ترکیب سرویس تهیه کننده سرویسی است که سرویس های ارائه شده توسط سایر تهیه کنندگان را یکپارچه می نماید تا از آنها سرویس های جدید بسازد، همچنین مشخصات و کدهایی را تهیه می کند تا در مورد سرویس های ترکیبی عملیات زیر را انجام دهد:

- *متناسب کردن* : کنترل اجرای سرویس های ترکیب شده و مدیریت گردش داده ها در بین آنها و انتقال آن به خروجی.
- *کنترل کردن* : مجوز دادن به رخدادها و اطلاعات تولید شده توسط سرویس های ترکیبی جهت به اشتراک گذاشتن و منتشر کردن رخدادهای ترکیبی سطح بالاتر (برای مثال از طریق فیلتر کردن و خلاصه سازی)
- *مطابقت دادن* : حصول اطمینان از حفظ جامعیت سرویس های ترکیبی از طریق تطبیق دادن محدودیتها و نوع پارامترهای سرویس های بکار رفته.
- *ترکیب خواص سرویس ها* : بکارگیری ، مجتمع سازی و دسته بندی ویژگی های سرویس های ترکیب شده جهت بدست آوردن خواص ترکیبی جدید که دربردارنده کارایی ، هزینه ، امنیت ، جامعیت ، قیاس پذیری ، در دسترس بودن و قابلیت اطمینان می باشد.

[1] Service Aggregator

استانداردهای جدید ارائه شده با عنوان زبان اجرای پردازشهای کسب و کار برای سرویس های وب ، تلاشی است که بر اساس XML ، تعریف سرویس های وب جدید را که از ترکیب سرویس های موجود بدست می آیند ، ارائه دهد.یک پردازش BPEL بصورت انتزاعی با ارجاع و اتصال به نوع پورت[1] های تعیین شده در WSDL ای ایجاد می شود که در سرویس های وب موجود در یک پردازش ، تعریف شده است.

مدیریت نرم افزارهای کاربردی مهم و بحرانی تجارت الکترونیک ، می بایست نظارت عمیق و جامعی در محیطهای توزیع شده داشته باشد. خارج از دسترس بودن یک عنصر کلیدی در سیستمهای توزیع شده، تاثیر منفی زیادی بر کل چرخه گذاشته و باعث بیرون رانده شدن ارائه کننده سرویس از بازار می شود.

برای رویارویی با چنین موقعیتهایی ، سازمانها نیاز به کنترل دائم سرویس و حصول اطمینان از سلامتی سیستم دارند. کارایی می بایست همیشه ، در هر شرایطی و با هر بار کاری ، در سطح قابل قبولی باشد.

برای مدیریت قسمتهای مهم و بحرانی و سرویس های ویژه ، معماری توسعه یافته ، در لایه مدیریت سرویس بعنوان بالاترین سطح ، سرویس های مدیریت شده را ارائه کرده است.

این لایه شامل دو قسمت مدیریت عملکرد و مدیریت بازار می باشد. کارکرد مدیریت عملکرد بدین صورت است که قسمتهای مهم سیستم را پشتیبانی می نماید. این لایه جزئیاتی از کارایی سیستم را جهت ارزیابی آن ارائه میدهد و بدین صورت سازمان را قادر می سازد تا بر اساس وضعیت نرم افزار و تکمیل شدن تراکنشهای کسب و کار ، تصمیم گیری نماید. اپراتور سرویس ، مسئول انجام امور مربوط به این واحد است.

مدیریت عملکرد ، قابلیت بسیار مهم و کلیدی است که می تواند صحت و کارایی کلی سیستم را کنترل نماید و بدین ترتیب از بروز مشکلات شدید و خطا در سرویس ها جلوگیری کند.

این خطاها ممکن است بر اثر اجتماع و هماهنگ سازی سرویس ها و بخاطر عدم رعایت توافقهای در سطح سرویس (SLA) اتفاق بیافتد.

مدیریت و کنترلهای مناسب باعث کاهش احتمال بروز چنین خطاهایی می شود؛ بدین ترتیب که صحت ، پایداری و همچنین مناسب بودن ارتباط بین توابع بکار رفته در سرویس های ورودی و خروجی را بررسی و کنترل می نماید.

قسمت دیگر در لایه مدیریت ، مدیریت بازار می باشد. مسئولیت این واحد ارائه پروتکلها و قوانین استاندارد در سطح کسب و کار می باشد تا از این طریق امکان استفاده از سرویس های تعبیه شده در بازارهای مختلف بوجود آید.

[1] portType

در ضمن برخی از تسهیلات و سرویس های پایه برای امور مالی ، تضمین کیفیت و ... در این لایه قرار می گیرد تا از این طریق بازارهای مختلف بتوانند در کمترین زمان به سرویس ها دسترسی یابند.

این قسمت از لایه مدیریت ، توسط سازندگان بازار که کنسرسیومی از شرکتهای فعال در این عرصه هستند ، کنترل و نگهداری می گردد.

درنهایت ، با توجه به نکات مذکور می توان معماری سرویس گرا را روشی در جهت بهبود طراحی و استفاده از سیستمهای نرم افزاری دانست ، اگرچه مشکلات و چالشهای پیش روی آن همچنان نیازمند بررسی تجارب گذشته و نیز ارائه راه حل مناسب پیرامون مسائل مطرح در این معماری می باشند.

مهمترین مفاهیم و اصول در نظر گرفته شده در طراحی سرویس گرا به شرح زیر می باشد:

1- کپسوله سازی سرویس[1] تاکید بر متمرکز کردن عملیات وابسته به داده در یک واحد (کپسول) مشخص و پنهان کردن پیاده سازی و مکانیزم درون واحد نرم افزاری است.

2- اتصال سست بین سرویس ها[2] تاکید بر استقلال سرویسها و کاهش وابستگی سرویسها به یکدیگر است فقط کافی است سرویسها از وجود هم آگاه باشند.

3- قرارداد سرویس دهی[3] ارتباط بین سرویس ها بر اساس قرارداد تعریف شده ای است که در اسناد فنی بطور مشخص ذکر میشود.

4- مجرد ساختن سرویس[4] تاکید بر جدا کردن پیاده سازی از رابط (جهان خارج) و پنهان کردن مکانیزم و نحوه انجام کار در درون واحد ارائه دهنده سرویس میباشد.

5- استفاده مجدد و بکارگیریف سرویس[5] تاکید بر طراحی سرویسها به نحوی است که بتوان آنها را در سامانه های مختل بکار برد. با تاکید بیشتر بر استفاده مجدد.

6- قابلیت ترکیب سرویس[1] به معنی آنست که سرویسها به نحوی طراحی شوند که با برخورداری از قابلیت ترکیب شدن ایجاد سرویسهای مرکب (کامپوزیت) امکانپذیر باشد.

---

[1] Service encapsulation
[2] Service loose coupling
[3] Service contract
[4] Service abstraction
[5] Service reusability

7- خودگردانی سرویس[2] عبارتست از قابلیت و قدرت سرویس در بکارگیری و مدیریت منابع خود بطور مستقل و همچنین کنترل کامل بر منطق پیاده سازی خود.

8- بدون حالت بودن سرویس[3] به این معنی است که سرویس باید در مورد فعالیت های گذشته (فراخوانی های گذشته) کمترین اطلاعات را نگه دارد و تاکید بر طراحی سرویس بنحوی است که حالتهای وابسته به گذشته کمتری داشته باشد.

9- قابلیت کشف شدن سرویس[4] به این معنی است که سرویس باید در یک محیط شبکه با استفاده از سازوکارهای مناسب توسط برنامه های دیگر آشکار شود .

15-ویژگی های سیستم های نرم افزاری معماری سرویس گرا

استفاده کننده از سرویس هیچ لزومی ندارد از جرئیات پیاده سازی سرویس در سمت سرویس دهنده مطلع باشد - محل سرویس دهنده باید از نظر استفاده کننده از سرویس پنهان باشد (در انجام امور مرتبط با استفاده از سرویس) و تنها در زمان اجرا سرویس گیرنده از مکان سرویس دهنده آگاه خواهد شد. - نرم افزار مبتنی بر معماری سرویس گرا باید بتواند با نرم افزارهای موجود روی سایر سکو ها تعامل داشته باشد. - چندین نسخه از سرویس باید بصورت همزمان در کنار هم فعالیت کنند زیرا با توجه به طیف گسترده استفده کنندگان در صورت بروزرسانی سرویس در سمت سرویس دهنده، به سرعت امکان بروزرسانی استفاده کنندگان سرویس وجود ندارد همچنین تعدادی از ویژگی هایی که هر نرم افزار، اعم از اینکه مبتنی بر این معماری باشد یا نباشد، باید داشته باشد به شرح زیر است: - کارایی زیاد - امنیت بالا (تضمین محرمانگی، صحت اطلاعات و همیشه در دسترس بودن) و همچنین کنترل دسترسی - قابلیت اطمینان بالا بخصوص وقتی سر و کار با تراکنش های چند مرحله ای است .

سرویس های وب به عنوان پایه معماری سرویس گرا :سیر تکامل و رشدXML ، با پیدایش سرویس های وب همراه بود. یک سرویس وب بهترین راه حل برای پیاده سازی معماری سرویس گرا است، مخصوصا وقتی دیدگاه استفاده از کل کاربران

---

[1] Service composability
[2] Service autonomy
[3] Service statelessness
[4] Service discoverability

اینترنت به عنوان کاربران بالقوه سرویس مطرح باشد. شما پایه کار خود را بر پروتکل HTTP بنا می نهید، پروتکلی که از همه پروتکل های دیگر روی اینترنت قابل دسترس تر است. با نگاه به قابلتهای سیستم های نرم افزاری مبتنی بر معماری سرویس گرا، شما متوجه خواهید شد که سرویس های وب بسیاری از موارد مطرح شده در بالا را رعایت می کنند اما تعدادی از اصول مطرح شده را هم زیر پا می گذارند که آن را بررسی می کنیم:

-کارایی XML :که عنصر اصلی سازنده سرویس های وب است، نسبت به سایر مکانیزم های انتقال اطلاعات دودوئی از سربار بسیآر زیادی برخودار است .

-قابلیت اطمینان در تراکنش ها: اگر شما در یک تراکنش از یک سرویس وب استفاده کنید، چگونه می توانید صحت تراکنش را تضمین کنید در حالی که تمام کارهای شما مبتنی بر اینترنت و پروتکل HTTP است؟

-امنیت: شما چگونه می توانید کاربران سرویس خود را تصدیق هویت کنید تا بعد از آن بتواند صلاحیت آنها را در استفاده از سرویس تان مورد بررسی قرار دهید؟ همچنین یک نکته منفی دیگر در مورد سرویس های وب در حال حاضر، عدم پشتیبانی اکثر محیط های تولید نرم افزار (IDE) برای تولید و استفاده از آنها است و در عین حال فراهم کردن قابلتهایی مانند کمک به برنامه نویس در استفاده از متدها و غیره یا پیدا کردن خطاها در زمان کامپایل و نه زمان اجرا. بنابراین، مگر اینکه موارد فوق به نحوی حل نگردد، ممکن است استفاده از سرویس های وب به عنوان پایه معماری سرویس گرا مورد سوال قرار گیرد. البته در هر حال سرویس های وب از این نظر که طیف کاربران بالقوه آنها اینترنت است بسیار مورد توجه هستند. در حال حاضر هم در اکثر سازمانها برای تمامی نرم افزار ها یک واسط بصورت وب سرویس جهت فراهم کردن استفاده از آن برای سازمانهای همکار فراهم می شود و یا حتی در داخل سازمان و در مواردی که استفاده از نرم افزار مذبور در داخل سازمان بسیار استفاده شود، با توجه به مشکلات کارایی سرویس های وب، یک واسط بصورت یکی ازفناوریهای برنامه نویسی مبتنی بر مولفه مانند +COM و یا CORBA برای نرم افزار ایجاد می شود. آماده شدن برای معماری سرویس گرا: همانطوری که ذکر شد، با وجود اینکه تعدادی نکات منفی در استفاده از سرویس های وب به عنوان پایه معماری سرویس گرا وجود دارد اما این موارد قابل حل هستند .

برای مثال در مورد بحث کارایی، می توان از پردازنده ای قدرتمند تر استفاده کرد و یا مشکل امنیت را می توان با استفاده از زیرساختهای مبتنی بر رمزنگاری های نامتقارن حل کرد. در هر حال اگر شما تا بحال برای معماری سرویس گرا آماده نشده

اید، در هر حال لازم است تا به این سمت پیش روید زیرا همانطور که در ابتدای این مقاله اشاره شد، نرم افزارهای مبتنی بر این معماری، نسل غالب سالهای آینده خواهند بود. بدین منظور باید اندکی تفکر خود را در مورد طراحی نرم افزار، تغییر دهید. در زیر به مهمترین آنها اشاره می شود:

- توصیه که می شود که با سرویس دهنده ها از طریق واسط های چاق ارتباط برقرار کنید و از استفاده از واسط های پرحرف بپرهیزید. به عبارت دیگر سعی کنید عملیاتی که شامل چندین فراخوانی است از طریق یک فراخوانی انجام دهید. هر بایت اطلاعاتی که شما روی اینترنت می فرستید محسوس است زیرا روی اینترنت اولا پهنای باند محدود است و همچنین در مورد هر انتقال باید عملیات تحلیل نام و مسیریابی انجام شود.

-سعی شود که حتی الامکان اطلاعات مربوط به وضعیت را در سمت سرویس دهنده نگهداری نشود سعی شود این کار را به استفاده کنندگان واگذار گردد. برای مثال اگر شما یک سازمان باشید که تعداد زیادی مراجعه کننده دارد و شما نیاز به اطلاعات مراجعه کننده ها دارید، اگر بخواهید خودتان تمام اطلاعات مربوط به مراجعه کنندگان خود را نگهداری کنید به یک انبار بسیار بزرگ نیاز خواهید داشت . بهتر است از مراجعه کنندگان خود بخواهید که اطلاعات خودشان را نگهداری کنند، نه خود سازمان شما بخواهد آنها را نگهداری کند.

- سعی شود از واسط های بسیار خوش تعریف برای سرویس های خود استفاده گردد زیرا وقتی مبنا بر اساس سرویس های وب بنا شود لازم نیست این واسط ها در اختیار استفاده کنندگان از سرویس خود قرار گیرند.(از طریق WSDLسرویس وب خود)

- سعی شود به سوی استفاده از روشهای غیرهمزمان برای فراخوانی های خود پیش روید زیرا بسیاری از سرویس ها به استفاده کنندگان خود بصورت غیرهمزمان سرویس می دهند(مانند سرویس های وب) بنابراین برای سرویس گیرندگان بهتر است از این روش تبعیت کنند. این روش مناسبی نیست که سرویس گیرنده به علت اینکه سرویس دهنده هنوز پردازش را شروع نکرده است ، بلاک شود .به عبارت دیگر سعی کنید دید خود را از حالت درخواست/پاسخ (مطرح در معماری Client/Server)به دید مبتنی بر پیام تغییر دهید؛ یعنی وقتی که سرویس گیرنده یک پیام را برای سرویس دهنده ارسال کرد سرویس دهنده بعد از مدتی از طریق یک پیام به سرویس گیرنده پاسخ خواهد داد .

-برای تصدیق هویت و کنترل دسترسی به روشهای دیگر فکر کنید. مکانیزهای امنیتی در مورد سرویس های وب متفاوت است. در مورد مکانیزهای امنیتی مورد استفاده از روشهای خاص یک سکو استفاده نکنید زیرا قابلیت تعامل سیستم شما را با سایر سرویس ها بخطر می اندازد(مانند Integrated Windows Authentication) اخیرا هم یک گسترش در

مشخصات سرویس های وب با نام ws-security بوجود آمده است که از آن جهت پیاده سازی امنیت در سروی های وب استفاده می شود .

-از سکوئی استفاده شود که اجازه دهند بطور همزمان چندین نسخه از یک سرویس را در کنار هم نگه دارد (مراجعه به قابلیتهای سیستم های نرم افزاری مبتنی بر معماری سرویس گرا) همچنین لازم به یاد آوری است که فناوری هایی همانند COM+,CORBA,RMIدر حیطه خود فنآوری های موفقی بوده و هستند و تعداد بسیار زیاد سیستم هایی که از این معماری ها استفاده می کنند دلالت بر موضوع مزبور دارد. سرویس های وب شامل مفاهیمی است که در مورد این فناوری ها وجود ندارد، اما این به این معنی نیست که سرویس های وب در زمانی کوتاه جایگزین این فناوری ها خواهند بود؛ و بنابراین سعی شود در کنار این فناوری ها از سرویس های وب بهره گرفته شود.

16-یکپارچه سازی سیستم های سازمانی و تعامل پذیری بین سازمانی

در حوزه مباحث فناوری اطلاعات جمله معروفی است که می گوید "هیچ نرم افزاری یک جزیره نیست". سیستم های اطلاعاتی یک سازمان زمانی می توانند موثر و کارامد باشند که با هم تعامل و ارتباط منلاسبی داشته باشند. امروزه این مورد یکی از اهداف مدیران اطلاعاتی سازمانهاست، البته نباید تصور کرد که ارتباط بین سیستم های اطلاعاتی فقط مختص به انتقال بایت های داده می شود! اجرای فرایندهای کسب و کار، وابسته به نرم افزارها و سیستم های اطلاعاتی متنوعی است که هر کدام در زمانی و با فناوری خاصی تهیه شده اند. لذا اتوماسیون چنین فرایندهائی منوط به تعامل پذیری سیستم های مختلف سازمانی است. بدین منظور مشکلات زیادی پیش روی سازمانهاست که می بایست بصورت مناسب برطرف شوند. در این راستا نیاز به ایجاد چارچوب و زیرساختی است که قادر به حل مشکلات ذیل باشد:

- اتصال با نرم افزارهای ناهمجور از لحاظ فناوری که دارای سکوها و ساختارهای متفاوتی هستند: وب سرویس بهترین راه حل در این حوزه است. در برخی شرایط دیگر به اشتراک گذاری فایل ها یا دسترسی به بانک های اطلاعاتی مشترک می تواند کارساز باشد.

- اجرای فرایندهای اتوماسیون شده: زیرساخت فناوری باید شامل موتور فرایندی برای اجرای منطق و گردش کار فرایند باشد. با اجرای فرایند توسط این موتور، هر فعالیت فرایند می تواند توسط یکی از سیستم های متفاوت سازمانی اتوماسیون شود. اگرچه انجام فعالیت ها به عهده این سیستم هاست اما مدیریت و یکپارچگی فرایندها توسط این زیرساخت مدیریت می

شود.

- ارتباط با سیستمها/سامانه های سایر سازمانها: بدین منظور نیاز به استانداردها و پروتکل هائی است که مورد توافق سازمانها و کشورهای مختلف باشد. تبادل داده الکترونیکی (EDI) و سایر استانداردهای موجود وظیفه ارتباط با دیگر سازمانها و شرکاء را به عهده دارند.

- مدیریت و دیده بانی فرایندها: در زمانی که فرایندهای سازمان بصورت اتوماسیون در حال اجرا می باشند، مکانیزمی جهت مدیریت و دیده بانی فرایندها لازم است تا مدیریت سازمان بتواند به کمک این اطلاعات، تصمیمات درست در خصوص اصلاح یا بهبود فرایند را اتخاذ نماید.

انچه گفته شد مجموعه ای از نیازهائی بود که یک چارچوب جامع برای حفظ یکپارچگی و تعامل پذیری سیستم های اطلاعاتی به آن نیاز دارد و معماری سرویس گرا با همین رویکرد ارائه شده است.

17-چالش های فناوری در حوزه سیستم های کلان اطلاعاتی

مهمترین چالش در حوزه معماری سیستم های اطلاعاتی، عدم یکپارچگی و تعامل پذیری سیستم های اطلاعاتی (داخل سازمانی / بین سازمانی) است. در این راستا استراتژی های فناوری اطلاعات عموما شامل موارد زیر است:

- یکپارچه سازی سیستم های اطلاعاتی داخل سازمانی که معمولا تحت عنوان Enterprise Application Integration(EAI) شناخته می شود. این موضوع یکی از اهداف معماری سرویس گرا (Service Oriented Architecture) نیز می باشد.

- اتوماسیون فرایندهای سازمان که تحت عنوان Business Process Management شناخته می شود و امروزه ارزش زیادی برای سازمانها در بردارد.

- تعامل پذیری سیستمها/سامانه های بین سازمانی که تحت عنوان Business-to-Business (B2B) شناخته می شود.

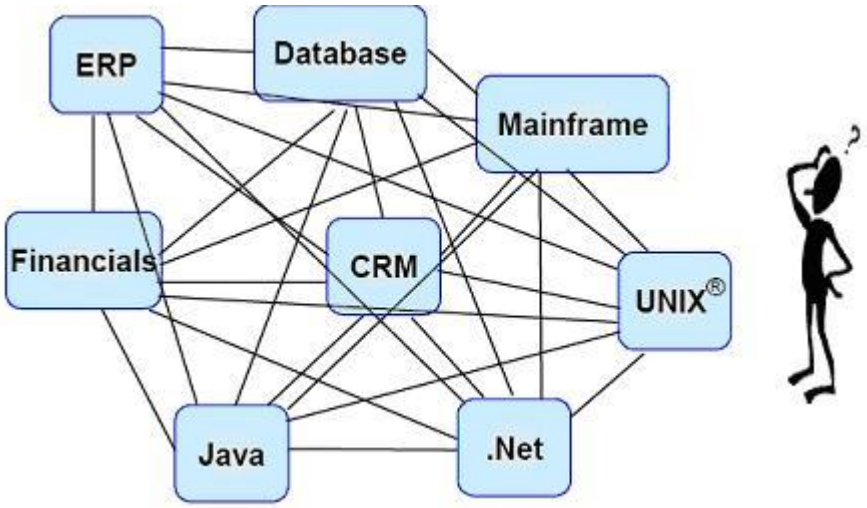
در بخش "ج" چگونگی عینیت بخشیدن به سه استراتژی گفته شده توسط معماری سرویس گرا بررسی می گردد.

18- یکپارچه سازی سیستم های سازمانی و تعامل پذیری بین سازمانی به کمک معماری سرویس گرا

در ادامه چگونگی عینیت بخشیدن به سه استراتژی گفته شده توسط معماری سرویس گرا بررسی می گردد.

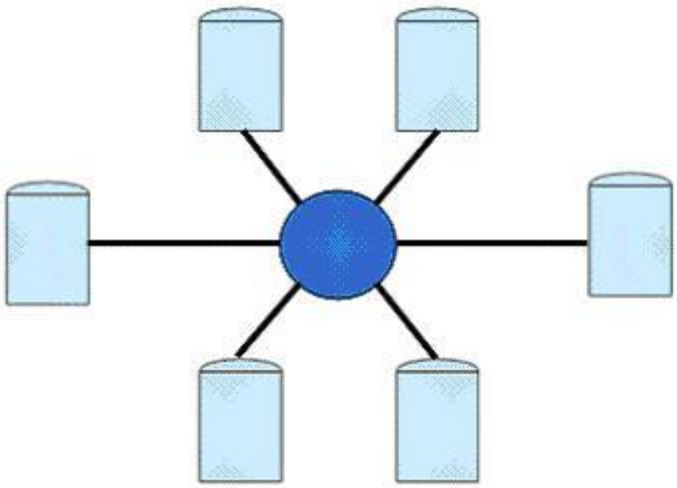
آ- یکپارچه سازی سیستم های اطلاعاتی

راه حل معماری سرویس گرا برای یکپارچه سازی سیستم های اطلاعاتی، ارتباط بین سیستم های اطلاعاتی به کمک وب سرویس است. از اواخر دهه 90 برای چالش تعامل پذیری سیستم های اطلاعاتی رویکرد هائی ارائه شده که معروفترین انها اتصال نقطه به نقطه (Peer-to-Peer) و یکپارچگی مبتنی بر یک مترجم مرکزی بوده است. در حالت نقطه به نقطه(شکل 2) برای هر تعامل بین دو سیستم اطلاعاتی در سازمان لازم است که استاندارد و مسیر ارتباطی مربوطه تعریف و فراهم گردد. طبیعی است که چنین رویکردی بسیار هزینه بر و دست و پا گیر خواهد بود.



شکل 7- رویکرد اتصال نقطه به نقطه برای ارتباط بین سیستم های اطلاعاتی سازمان

در حالت مترجم مرکزی نیز میان افزاری (Middle-Ware) به عنوان مترجم بین همه سیستم های اطلاعاتی عمل می کرد به گونه ای که مانند یک هاب مرکزی تمامی پیامهای ارسالی به این واسط ارجاع می شد و پس از ترجمه به پروتکل و فناوری مربوط به سیستم دوم، ارسال می گشت.(شکل 8) این گزینه نیز با دشواریهائی همراه بود که مهمترین انها وجود انواع پروتکل های ناهمجور و عدم جامعیت بود. اما در معماری سرویس گرا اصل بر این است که همه سیستم های اطلاعاتی با یک واسط استاندارد و مورد توافق جهانی تعامل داشته باشند. این واسط وب سرویس(Web Service) نام دارد و پروتکل های مورد استفاده ان نیز شامل معماری سرویس گرا UDDI،WSDL،P می شود، همه این پروتکل ها بسطی از XML هستند که استانداردی جهانی و مورد توافق همه سکوها، فناوری ها و سازندگان است.



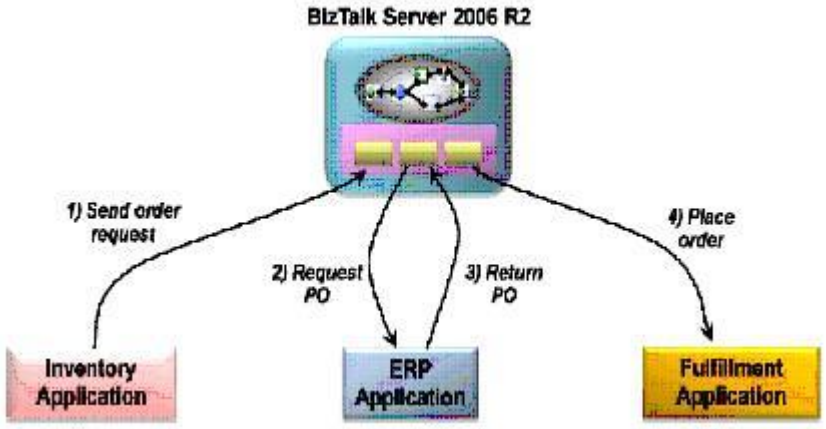
شکل 8- رهیافت مترجم مرکزی برای ارتباط بین سیستم های اطلاعاتی سازمان

ب- یکپارچگی اتوماسیون فرایندهای سازمان در قالب ارکستریشن

معماری سرویس گرا برای مدیریت و اجرای فرایندهای سازمان از مفهوم ارکستریشن کمک گرفته است(که در قسمت های قبل توضیح داده شد)، در این رهیافت منطق و جریان کار فرایند از فعالیت های ان جدا می شود، به گونه ای که جریان گردش فرایند در قالب BPEL مدیریت می شود ولی هر کدام از فعالیت های فرایند می توانند توسط سیستم های اطلاعاتی مختلف پیاده سازی شوند. بدین ترتیب امکان تغییر جریان منطق کار بدون نیاز به تغییر سیستم های پشتیبان میسر می شود که این امر کمک شایانی به انعطاف پذیری فناوری در پاسخ به تغییرات کسب و کار می کند.

برای تعریف و پیاده سازی فرایندها سه نقش اصلی تحلیل گر، طراح و مدیر وجود دارد. تحلیل گر جریان گردش فرایندهای کسب و کار سازمان را شناسائی و مدلسازی می کند، سپس طراح این مدل را به شکل سیستمی و قابل اتوماسیون در می آورد. در اینجا هر فعالیت فرایند به یک سرویس سیستمی(System Service) یا یک فعالیت انسانی(Task Human) نگاشت می شود. همچنین طراح در هنگام اتوماسیون فعالیت ها با سایر سیستم های اطلاعاتی سازمان، سازگار کننده مناسب را جهت برقرای تبادل تنظیم می نماید، برای هر تعامل باید نگاشت ها، پروتکل ها و سایر تنظیمات انجام شود. در حالی که تحلیل گر و طراح مشغول کار بر روی فرایندهای خاصی هستند، مدیر وظیفه کنترل و نظارت بر مجموعه تعاملات و ارتباطات بین سیستم های اطلاعاتی و تنظیم مجوزها و سطوح دسترسی را دارد. در شکل 9 چگونگی پیاده سازی یک فرایند که با سه سیستم اطلاعاتی ارتباط دارد توسط ابزار BizTalk نشان داده شده است. در این مثال هر سیستم دارای فناوری و استانداردای مخصوص به خود است و معماری سرویس گرا باید قادر باشد یکپارچگی بین این سیستم ها را حفظ نماید. نکته جالب اینکه

سیستم های اطلاعاتی به کار گرفته شده برای اتوماسیون فرایند مزبور از وجود فرایند اطلاع ندارند، آنها تنها به پیامهائی که ارسال شده جواب می دهند(جداسازی منطق فرایند از اتوماسیون فعالیت های آن)، این موضوع کمک شایانی به چابکی فناوری اطلاعات در پیاده سازی فرایندهای جدید و تغییر فرایندهای موجود می کند که ارزش استراتژیکی برای سازمانها دارد.



شکل 9- پیاده سازی فرایندهای کسب و کار به کمک موتور
فرایندی(BizTalk) مبتنی بر معماری سرویس گرا

ج- تعامل پذیری بین سازمانی

اگرچه ارتباط و یکپارچگی سیستم های اطلاعاتی سازمانی ضروری است اما از آن مهمتر(سخت تر) تعامل پذیری بین سازمانی است، چراکه میزان تنوع فناوری ها و پروتکل ها در بین چند سازمان به مراتب بیشتر از میزان آن در بین سیستمهای داخل سازمانی است. در شرایط اقتصادی و تجاری جدید، سازمانها نیاز دارند که بصورت موثر از اطلاعات یکدیگر استفاده کنند، از طرف دیگر "فرایندهای بین سازمانی" در حال گسترش هستند. تجارت که زمانی سازمانی و حوزه ای بود اکنون به سمت جهانی شدن و اکوسیستمی پیش می رود و تعیین مرز و اندازه جغرافیائی برای سازمانها دشوار شده است. در چنین شرایطی نیاز به تبادل اطلاعات بین سازمانها به شدت احساس می شود. راه حل معماری سرویس گرا برای این حوزه استفاده از وب سرویس های جهانی است که در وب قابل شناسائی و فراخوانی هستند برای استفاده از این وب سرویس ها قبل از هر چیز انها باید توسط متقاضیان شناسائی شوند. بدین منظور دایرکتوری ثبت و شناسائی سرویس ها (UDDI) ایجاد شده است. ارائه دهندگان سرویس مشخصات سرویس های خود را در این دایرکتوری ها ثبت می کنندو متقاضیان نیز با جستجوی سرویس مورد نظرشان (مانند موتورهای جستجوری صفحات وب) می توانند از این سرویس ها در سازمان خود بهره برند. نتیجه این امر امکان استفاده از انواع مختلفی از سرویس های جهانی است که توسط ارائه دهندگان مختلف فراهم شده است.

نتیجه گیری

در این مقاله به معماری سرویس گرا پرداخته شده که یکی از ابزار های اصلی برای توسعه بیشتر فناوری ها و محصولات نرم جهان معاصر است. نکته مهمی که این رویکرد بدست می دهد بکار گیری آن در سطوح مختلف تجرید سازمانی یا فضای جنگ است. این نوع معماری بر روی سطح کسب و کار یا عملیات، سطح سیستم های اطلاعات، داده و فناوری می تواند بکار گرفته شود. سرویس گرائی با استفاده از مفهوم سرویس مولفه (دانه ریزترین سرویس بعنوان یک واحد نرم افزاری) واجد هر سه ویزگی فوق هستند: بکارگیری سرویس گرائی در همه سطوح تجرید سازمان، انسجام بسیار بالای سرویس مولفه و بالاخره اتصال سست بین همه مولفه هائی که از ترکیب سرویس ها بوجود آمده اند نشان دهنده موفق بودن طراحی سرویس گرا است.

1

1