دانشکده فنی و مهندسی

کارشناسی ارشد مهندسی کامپیوتر نرم‌افزار

گروه مهندسی کامپیوتر و فناوری اطلاعات

موضوع:

پیاده سازی سرویس های نرم افزاری(بر اساس روش های مطرح شده در سمینار۱)

نگارش:

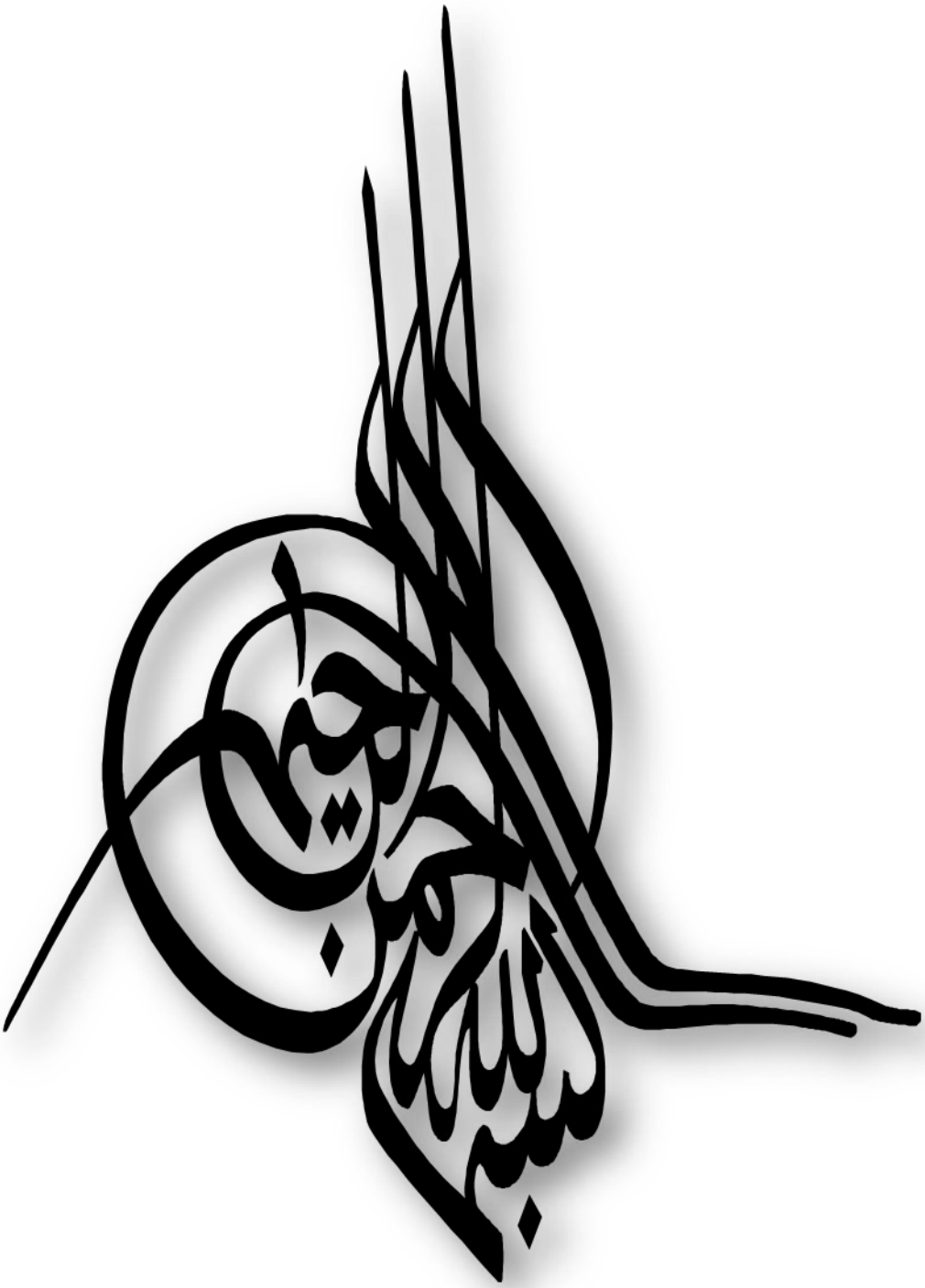
علیرضا رزمجو

۹۸۰۲۱۸۷۶۱

استاد راهنما:

دکتر رضوی

تابستان ۱۴۰۰



فهرست مطالب

فصل اول تعریف هدف.....٦

١.١ نگاهی به پیکربندی سرویس گرا.....٦

١.٢ چرخه حیات.....٦

١.٣ مراحل soa.....٨

١.٤ اصول طراحی مدل قالب سرویس ها.....١٣

فصل دوم.....١٧

ادبیات موضوع.....١٧

٢.١ تعاریف اصلی.....١٨

مدل تکامل یافته معماری سرویس گرا.....٣٣

چرخه حیات معماری سرویس گرا.....٣٤

٢.٢ پارامتر های مهم کیفی.....٤١

٢.٣ بررسی فاکتور دانه بندی سرویس ها.....٤٩

فصل سوم معرفی روش های گذشته.....٥٣

٣.١ دسته بندی روش های موجود.....٥٤

Zimmermannروش اول٢,٣.....٦١

Zimmermannروش دوم٣,٣.....٦٣

Zhangروش٤,٣.....٦٣

SOMAروش٦,٣.....٦٥

Portierروش٧,٣.....٧٣

Inganti روش ۸,۳ ..... ۷٤

فصل چهارم - ویژگی های راه حل مورد انتظار ۸۰

٤.۱ نتایج مورد انتظار از سرویس ها ۸۱

چکیده

سرویس می تواند معانی دیگری نیز داشته باشد، گاه معنای آن وسیع تر است، برای مثال یک جز توزیع شده که واسط آن با کسب و کار طراحی شده است. گاهی معنای آن جزئی تر می باشد مانند وب سرویس مبتنی بر **SOAP**

ابتدایی ترین مشکل برای پیاده سازی **SOA**این است که اکثریت با مفهوم اصلی ساختمان بلوکه ای، یعنی همان سرویس، موافق نیستند. بسیاری از تعارف سرویس، به مفاهیم بی ثبات و مختصر اشاره دارند، مانند قابلیت انعطاف پذیری و چابکی که برای برنامه نویسان بسیار مبهم می باشد. به جرأت می توان گفت، هیچکس در انتخاب نمی خواهد یک نرم افزار غیرقابل انعطاف و کند که نیازهای عملی را برآورده نمی سازد خلق کند .اما در واقعیت برخلاف کوشش های زیاد اغلب این اتفاق رخ می دهد. دانستن اینکه **SOA**می تواند سازگاری و چابکی به ارمغان بیاورد ستودنی است اما این به تنهایی کمک کننده نیست.

تمامی تعاریف دیگر برای دنیای واقعی خیلی جزئی تر می باشند، برای مثال :تعیین کردن اینکه سرویس باید جزئی از نرم افزار باشد که زبانش **SOAP**است می تواند گمراه کننده باشد. اما این یک محدودیت ساختگی است، اشخاص زیادی هستند که از **REST**استفاده می کنند و فکر می کنند که در حال ارائه سرویس هستند. شاید بهتر است که تعریف ما شامل چیزهایی درباره پیام های مبتنی بر **XML**که تفاوت بین **RESTfull webservice**و **SOAP-Based service**را به وجود می آورند، باشد. اما باز هم یک محدودیت برای تعریف کلی ما به حساب می آید.

چگونه به یک تعریف جامع و خاص برای سرویس دست پیدا کنیم که برای اهداف متنوع سرویس در دنیای واقعی پذیرفتنی باشد؟

بهتر است تمامی جنبه های تعریف سرویس را پیش از اینکه بیشتر پیش رویم، بررسی کنیم .

فصل اول تعریف هدف

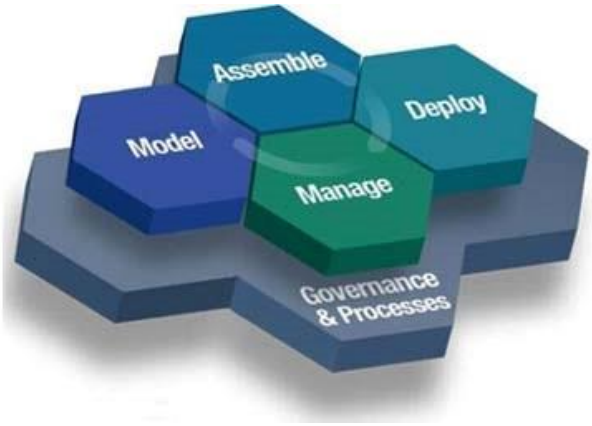
برای توسعه نرم افزارها در سطح یک سازمان، به منظور جلوگیری از بروز پیچیدگی، نیاز به انتخاب سطح تجرید مناسب است. یکی از روش های موجود برای این کار استفاده از سرویس ها و معماری سرویس گرا است. به طور کل می توان گفت معماری سرویس گرا رهیافتی است برای ساخت سیستم های توزیع شده که نیازمندی های نرم افزاری را به صورت 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.

۱.۱ نگاهی به پیکربندی سرویس گرا

معماری سرویس گرا (Service-Oriented Architecture) نوعی الگوی طراحی است. این الگو برای ارائه خدمات به برنامه‌های دیگر از طریق پروتکل های معماری سرویس گرا طراحی شده است. در پاسخی دیگر به سوال معماری سرویس گرا چیست، می‌توان گفت رهیافتی برای ساخت سیستم‌های توزیع شده است که انواع کارکردهای نرم افزاری را در قالب سرویس ارائه می‌کند. این سرویس‌ها علاوه بر اینکه به وسیله سایر نرم افزارها قابل فراخوانی هستند، برای ساخت سرویس‌های جدید نیز مورد استفاده قرار می‌گیرند.

۱.۲ چرخه حیات

بر اساس طرح IBM برای معماری سرویس گرا می توان یک چرخه حیات در نظر گرفت. در شکل ۱ این چرخه نشان داده شده است[۹]:



شکل شماره ۱ -۱: چرخه حیات معماری سرویس گرا [۱۱]

همانطور که در شکل مشاهده می شود، چرخه حیات معماری سرویس گرا از چهار مرحله ،مدل ،گردآوری نصب و مدیریت تشکیل شده و مرحله حاکمیت و فرآیندها به عنوان زیرساختی برای این مراحل چهارگانه عمل می کند. چارچوب کلی کار در این چرخه حیات به این صورت است که در فاز مدل نیازمندی های حرفه جمع آوری می شود و براساس این نیازمندی ها فرآیندهای حرفه طراحی می گردند، سپس در فاز گردآوری سرویس های موجود و سرویس های جدید در کنار هم قرار داده شده و فرآیندهای مورد نیاز حرفه را شکل می دهند. در فاز نصب فرآیندهای تولید شده در محیط مشتری نصب شده و در فاز مدیریت کاربران فرآیندهای نصب شده را هم از لحاظ فنی و هم از لحاظ مطابقت با نیازها مورد یررسی قرار می دهند. در فاز مدیریت بازخوردهای کابران به منظور توسعه فرآیندهای جدید حرفه به فاز مدل وارد می شوند و چرخه را ادامه می دهند. در زیر همه این مراحل، حاکمیت و فرآیندها جهت گیری سیاست های حرفه را برای پروژه معماری سرویس گرا تعیین می کنند.

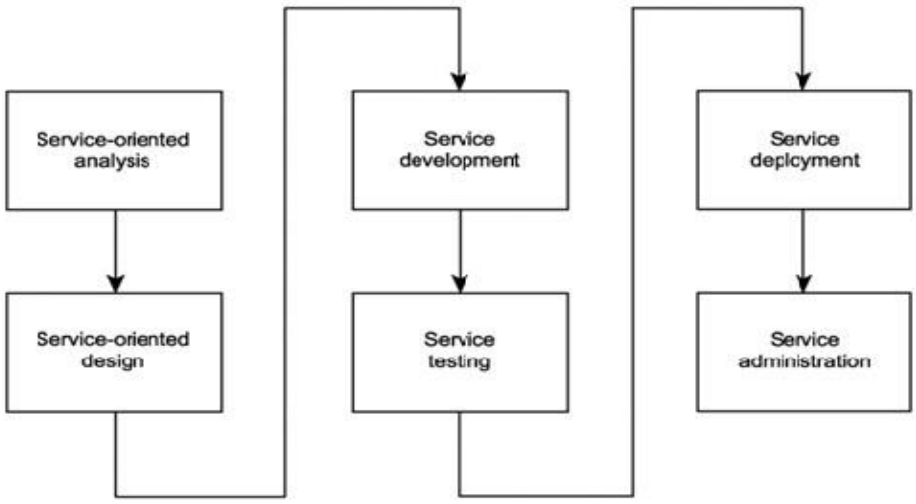
همانطور که در شکل شماره ۱ نشان داده شده است، اولین گام در چرخه حیات معماری گام مدلسازی می باشد. در این گام سرویس های مورد نیاز برای پاسخگویی به نیازهای حرفه تعیین شده و ویژگی های اصلی آنها مشخص می شود. به عبارت دیگر در این گام دو فعالیت اصلی انجام می شوند: ۱.

تشخیص سرویس ها (service identification) و ۲. تعیین ویژگی های سرویس ها (service specification) و ۳. عینیت بخشی به سرویس ها (service realization)

در ادامه برای آشنایی بیشتر با فرآیند تولید سیستم های نرم افزاری با استفاده از روش معماری سرویس گرا، به بیان فازهای پایه ای در معماری سرویس گرا می پردازیم.

۱.۳ مراحل soa

اصولا فازهای اصلی برای توسعه پروژه های مبتنی بر سرویس گرایی همانند فازهای متداول در توسعه پروژه های مبتنی بر روش های پیشین برای تولید سیستم های توزیع شده هستند. فازهای اصلی در سرویس گرایی در شکل زیر نشان داده شده اند:



شکل شماره ۲ -۱: فازهای SOA [۱]

در شکل فوق دو فاز ابتدایی (service-oriented analysis , service-oriented design) مسئول تطابق رهیافت در دست ساخت با ویژگی های شمرده شده برای سرویس گرایی هستند. در

این دو فاز ایتدایی سرویس ها شناسایی شده و از میان مجموعه سرویس های کاندید، سرویس هایی که وارد مرحله توسعه سرویس ها می گردند تعیین می شوند. به مجموعه این فعالیت ها شناسایی سرویس ها (*service identification*) می گویند. [1]

۱.۳.۱ تحلیل سرویس گرا

اهداف اصلی از تحلیل سرویس گرا عبارتند از[۱]:

- مشخص کردن مجموعه اصلی عملیات کاندید سرویس ها
- گروه بندی عملیات کاندید در بسته های منطقی. این بسته ها نشان دهنده سرویس های کاندید هستند.
- مشخص کردن مرزهای اصلی سرویس های کاندید به منظور جلوگیری از بروز رویهم افتادگی سرویس ها
- مشخص کردن درصد قابلیت استفاده مجدد برای هر بسته منطقی
- حصول اطمینان از وجود تناسب بین بسته های منطقی و کارکرد مورد انتظار آنها
- تعیین تمامی مدل های ترکیبی ممکن مراحل تحلیل سرویس گرا عبارتند از:
- مرحله ۱- تعیین نیازمندی های [خودکارسازی] حرفه: با توجه به اینکه نیازمندی های حرفه (که به منظور پوشش آنها دست به طراحی و توسعه سیستم ها می زنیم) جمع آوری شده اند، مستندات موجود در این زمینه به عنوان گام آغازین در فرآیند تحلیل سرویس گرا استفاده می شوند. در این روش تنها نیازمندی های مرتبط با رهیافت سرویس گرا (آنچه می خواهیم برایش راه حل سرویس گرا ارائه کنیم) در نظر گرفته می شوند.
- مرحله ۲- مشخص کردن سیستم های خودکارسازی کنونی: در این مرحله منطق های کاری موجود در سیستم های کنونی مورد بررسی قرار می گیرند. اطلاعات بدست آمده در این مرحله به منظور شناسایی سرویس های کاندید به کار گرفته می شوند.

- مرحله ۳- مدلسازی سرویس های کاندید: پس از اینکه واحدهای کاری شناسایی شد و این واحدها در غالب منطق های عملیاتی (سرویس ها) بسته بندی گردید، این بسته ها به صورت سرویس های کاندید در آمده و به منظور نمایش یک مدل ابتدایی از ترکیب سرویس ها به منظور پیاده سازی سیستم مورد نظر استفاده می گردند.

۱.۳.۲ طراحی سرویس گرا

در ادامه ۱۲ مرحله به عنوان فرآیند مدلسازی سرویس ها مشخص می گردند.[۱]

- مرحله ۱- تجزیه فرآیندهای حرفه

مستندات فرآیندهای حرفه را گرفته و آن را به مراحل فرآیندها تجزیه کنید. توجه شود که تجزیه منطق جریان کاری فرآیندها به ریزترین درجه مراحل فرآیند اهمیت زیادی دارد. این خروجی های بدست آمده ممکن است در درجه ریزدانگی با مراحل فرآیندهای موجود در مستندات متفاوت باشند

- مرحله ۲- تعیین عملیات کاندید سرویس های حرفه

برخی مراحل در فرآیندهای حرفه از ماهیتی برخوردارند که می توان گفت این مراحل به هیچ یک از واحدهای منطقی تعلق نداشته و نمی توان آنها را در قالب سرویس های کاندید بسته بندی کرد. به عنوان مثال مراحل پردازش های دستی ($^{\text{manual}}$) در فرآیندهای حرفه که نیازی به خودکارسازی ندارند و همچنین مراحل فرآیند که توسط سیستم های موروثی انجام می شوند و نیازی به بسته بندی در قالب سرویس ندارند از این دسته می باشند. با جداسازی این مراحل از مجموعه عملیات کاندید برای سرویس های حرفه ، یک گام به فرآیند مدلسازی حرفه نزدیک تر می شویم.

- مرحله ۳- تعیین همنواسازی (*orchestration*)

اگر قصد دارید یک لایه همنواسازی برای پروژه SOA خود تولید کنید، لازم است قسمت های فرآیندهای مختلف که در این لایه نمایش داده می شوند را تعیین کنید. به عبارت دیگر

برای تعیین چگونگی همنواسازی و ارتباط متقابل فرآیندها، قسمت های مختلف فرآیندها که با هم در ارتباط هستند باید تعیین شوند. توجه شود اگر قصد تولید لایه همنواسازی را ندارید، از این مرحله عبور کنید.

- مرحله ٤- تولید سرویس های کاندید حرفه

در این مرحله خروجی های مراحل پیش را به صورت مجموعه ای از واحد های منطقی کار دسته بندی کنید. این واحدهای منطقی کار همان سرویس های کاندید هستند.

- مرحله ٥- ارزیابی و اعمال شاخصه های سرویس گرایی

در این مرحله واحدهای عملیاتی که به صورت سرویس ها دسته بندی شده اند را از نظر تطابق با معیارهای سرویس گرایی بررسی می کنیم. معیارهای اصلی سرویس گرایی عبارتند از: قابلیت استفاده مجدد، خودمختاری، statelessness،discoverability . از میان این چهار مفهوم دو مفهوا ابتدایی در این مرحله مورد توجه قرار می گیرند.

- مرحله ٦- تعیین ترکیبات سرویس های کاندید

تعدادی از سناریوهایی که به صورت متداول اجرا می شوند را مشخص کرده و برای هر سناریو مراحل فرآیندهایی که در مراحل پیش تعیین شده بودند را بررسی کنید. با این کار می توان: ایده خوبی در مورد کیفیت بسته بندی عملیات در سرویس ها بدست آورد، ارتباطات بالقوه میان لایه سرویس و لایه همنواسازی مشخص می گردد، ترکیبات بالقوه میان سرویس ها تعیین می شود، این کار موجب آشکار شدن منطق های جریان کاری و مراحل فرآیندی که در نظر گرفته نشده بودند می شود.

- مرحله ٧- بازبینی گروه بندی عملیات در سرویس ها

با توجه به نتایج بدست آمده از مرحله ٦، گروه بندی عملیات در سرویس ها را بررسی نموده و سازماندهی عملیات در گروه ها را در صورت نیاز تغییر دهید. ممکن است در این مرحله سرویس های جدیدی نیز تولید گردند.

- مرحله ٨- تحلیل نیازمندی های پردازشی برنامه های کاربردی

تا پایان مرحله ٦، سرویس های مبتنی بر حرفه در لایه سرویس تولید شده اند. مراحل بعدی دلخواه بوده و برای فرآیندهای پیچیده حرفه و محیط های بزرگ مبتنی بر سرویس مناسب هستند. در مراحل آینده تمرکز بر روی سرویس های برنامه کاربردی می باشد. در این مرحله تمامی گام های فرآیندها به منظور شناسایی نیازمندی های پیاده سازی و فناوری مورد تحلیل قرار می گیرند

- مرحله ۹- تعیین عملیات کاندید سرویس های برنامه های کاربردی

برنامه های کاربردی را به مراحل کوچک تجزیه کنید. توجه کنید این مراحل را طوری نامگذاری کنید که ارتباط آنها با توابع شان مشخص باشد.

- مرحله ۱۰-تولید سرویس های کاندید برنامه های کاربردی

خروجی های مرحله پیش را در سرویس های برنامه کاربردی گروه بندی کنید. منطق گروه بندی اینجا ارتباطات منطقی بین مراحل فرآیندها است.

- مرحله ۱۱- بررسی ترکیبات سرویس های کاندید

سناریوهایی را که در مرحله ۵ مشخص کرده اید را بررسی نمایید، با این تفاوت که در این مرحله سرویس های برنامه های کاربردی را نیز دخالت دهید. در این مرحله سعی نمایید تناظری بین سرویس های برنامه کاربردی و سرویس های حرفه ارائه دهید.

- مرحله ۱۲- بررسی گروه بندی سرویس های برنامه های کاربردی

پس از بررسی سناریوها در مرحله ۱۱، امکان بروز تغییراتی در گروه بندی سرویس ها وجود دارد. این امر به خاطر آشکار شدن مراحل از قلم افتاده در گام های پیشین است.

پس از این مرحله تمامی سرویس های کاندید و نحوه ترکیب آنها مشخص گردیده اند. در برخی منابع توصیه شده است تا در یک مرحله دلخواه (مرحله ۱۳) سرویس های کشف شده در محلی ذخیره سازی گردند تا در صورت نیاز در آینده مورد بررسی قرار گیرند.

اکنون که با فازهای اصلی معماری سرویس گرا آشنا شدیم و فعالیت های عمده در این فازها را مطالعه کردیم، با یکی از مهمترین گام ها در چرخه حیات معماری سرویس گرا یعنی گام مدلسازی سرویس ها (service modeling) بیشتر آشنا می شویم.

۱.٤ اصول طراحی مدل قالب سرویس ها

با توجه به شکل ۱ مشاهده می کنید مدلسازی سرویس ها اولین قدم در چرخه حیات معماری سرویس گرا است. اصلی ترین هدف از مدلسازی سرویس ها بالا بردن سطح تجرید و کاهش پیچیدگی در کار با سرویس ها می باشد. برای این منظور ابتدا در دامنه مورد بررسی، سرویس های مورد نیاز را شناسایی می کنیم. این شناسایی بر اساس اهداف حرفه مورد بررسی و فرآیندهای آن حرفه انجام می شود.

سپس ویژگی های هریک از این سرویس ها را به صورت جزئی تر مشخص کرده و نحوه تعامل آنها را با هم تعیین می کنیم. برای این کار از نمودارهای مختلفی مانند نمودار ارتباط سرویس ها، نمودار همکاری بین سرویس ها (collaboration diagram) و ... استفاده می شود.

در آخر به عنوان آخرین مرحله از مدلسازی سرویس ها، آنها را بر اساس ویژگی ها و خصوصیاتشان و هم چنین با توجه به ویژگی های حرفه مورد بررسی از نظر چگونگی تولید مورد بررسی قرار می دهیم. به این فعالیت عینیت بخشی به سرویس ها گفته می شود. در این قدم مشخص می شود که مثلا یک سرویس به طور کامل پیاده سازی شود و یا با استفاده از سرویس های موجود در سیستم های موروثی پیاده سازی شود و ...

فعالیت هایی که در بالا به عنوان فعالیت های مرحله مدلسازی سرویس ها به آنها اشاره شد، سه فعالیت اصلی در حیطه مدلسازی سرویس ها بوده و عبارتند از مرحله تشخیص سرویس ها (service identification)، مرحله توصیف سرویس ها (service specification) و مرحله عینیت بخشی

به سرویس ها (service realization). در این گزارش سمینار، تمرکز روی گام اول یعنی تشخیص سرویس ها (service identification) و بررسی اهداف، ویژگی ها و فعالیت های پیشنهاد شده برای اجرای این گتم است. در ادامه پس از آشنا شدن با جایگاه این فاز و بیان اهمیت آن با معیارهای سنجش تناسب سرویس های شناخته شده و روش های ارائه شده در این زمینه آشنا می شویم. سپس با بررسی کیفیت و توجه به هر یک از فاکتورهای سرویس ها در روش های ارائه شده، به نقد این روش ها پرداخته و سعی در تعریف راه حلی جدید به منظور پوشش کاستی های این روش ها داریم.

۱.٤.۱ اهمیت گام تشخیص سرویس ها

می دانیم که تشخیص سرویس ها اولین قدم از فاز مدلسازی سرویس ها است و فاز مدلسازی سرویس ها نیز خود اولین فاز از چرخه حیات معماری سرویس گرا می باشد. بنابراین برای ارائه راه حل سرویس گرا در حیطه یک حرفه یا سازمان اولین فعالیت انجام شده، تشخیص سرویس ها می باشد. اهمیت این گام از همین جا ناشی می شود. زیرا اگر در این گام سرویس های نامناسبی را به اشتباه تشخیص دهیم، در ادامه کار به مشکل بر خواهیم خورد. اهمیت این فاز از سه جنبه قابل بررسی است :

۱. وابستگی گام های بعد به خروجی این گام: از آنجا که این گام اولین گام در فاز مدلسازی سرویس گرا و همچنین در تمام چرخه حیات معماری سرویس گرا می باشد، خروجی این گام به عنوان مبنایی برای تمامی گام های آینده بوده و خروجی گام های بعد بر اساس خروجی این گام تولید می شوند. به همین علت اگر در این گام سرویس های نادرستی را تشخیص دهیم، در گام های بعد بر اساس این سرویس های نادرست، راه حل و سیستم ناقص و نادرستی را تولید خواهیم کرد.
۲. هزینه و زمان مورد نیاز برای رفع خطاها: با توجه به اینکه خطاهای احتمالی در این فاز در تمامی فازهای آینده تاثیر خواهند گذاشت، در صورتی که در آینده متوجه بروز خطاها شویم به هزینه و زمان بیشتری جهت رفع این خطاها نیاز خواهیم داشت. به عنوان مثال هنگام پیاده

سازی سرویس ها در فاز استقرار (deploy) اگر از وجود سرویس های نادرست و نابه جا آگاه شویم باید هزینه تمامی فعالیت های فازهای گردآوری (assemble) و مدلسازی (model) را دوباره پرداخت کنیم و به همین نسبت نیز به زمان بیشتری نیاز خواهیم داشت. بنابراین اگر در این فاز با استفاده از روشی خطاهای احتمالی را تشخیص دهیم، به میزان قابل ملاحظه ای بر کیفیت محصول نهایی و کاهش هزینه و زمان تولید محصول تاثیر خواهیم گذاشت.

۳. تسهیل فرآیند :نگهداری از آنجا که معماری سرویس گرا یک فرآیند موردی نبوده و همواره در دوره حیات حرفه و سازمان ادامه خواهد داشت، پیش بینی نیازهای آینده و تشخیص سرویس های مورد نیاز در آینده بر تسهیل فرآیند توسعه سیستم تاثیر مثبت خواهد داشت. به همین علت اگر در فاز تشخیص سرویس ها با دانستن هدف نهایی حرفه و داشتن آگاهی درست نسبت به تغییرات احتمالی در آینده حرفه بتوان سرویس هایی را تشخیص داد که علاوه بر پوشش نیازهای اکنون در حیطه سازمان، تا حدی نیازهای احتمالی آینده سازمان را نیز تحت پوشش خود قرار دهند و همچنین سرویس هایی با امکان پیکربندی آسان طراحی نمود، می توان تاحد بسیار زیادی فرآیند نگهداری سیستم تولید شده را تسهیل بخشیده و هزینه و زمان مورد نیاز برای اعمال تغییرات در فرآیندهای سیستم و وظیفه مندی های آن را به میزان قابل توجهی کاهش داد.

علاوه بر سه جنبه فوق گام تشخیص سرویس ها از جنبه های دیگری نیز اهمیت دارد و قابل بررسی است. اما سه جنبه فوق به عنوان اصلی ترین فاکتورهای اهمیت برای این گام مطرح بوده و اساس توجه به این گام می باشند.

نتیجه گیری

در این فصل پس از معرفی مفهوم معماری سرویس گرا، اهداف فازها و چرخه حیات آن با گام های موجود در چرخه حیات معماری سرویس گرا تا حدی آشنا شدیم. همچنین گام اولین این چرخه، یعنی مدلسازی سرویس گرا را شرح داده و نسبت به فعالیت های موجود در این گام شناخت کلی پیدا نمودیم. در ادامه فعالیت تشخیص سرویس ها به عنوان اولین فعالیت در فاز مدلسازی و چرخه حیات معماری سرویس گرا مورد بررسی قرار گرفت و دلایل اهمیت این گام تعیین گردید.

پس از آشنایی با جایگاه تشخیص سرویس ها (service identification) و دلایل اهمیت آن در ادامه تلاش داریم روشی را به منظور انجام این فعالیت معرفی کنیم. برای این کار در فصل های آینده به معرفی مفاهیم موجود در حیطه تشخیص سرویس ها پرداخته و سپس به معرفی روش های موجود در این زمینه می پردازیم. همچنین این روش ها را به منظور تطبیق با فاکتورهای مورد نظر برای سرویس های تشخیص داده شده بررسی نموده و نقاط قوت و ضعف هریک از این روش ها را بیان می نماییم. پس از شناخت کاستی های موجود در این روش ها به معرفی روشی جدید جهت رفع این نواقص می پردازیم.

فصل دوم

ادبیات موضوع

مقدمه

در فصل قبل در مورد مفهوم سرویس،معماری سرویس گرا، فازها و چرخه حیات آن توضیحاتی داده شد.

همچنین با فازهای تشخیص سرویس ها (service identification)، توصیف سرویس ها (service specification) و عینیت بخشی به سرویس ها (service realization) به عنوان فعالیت های اصلی در گام مدلسازی سرویس ها (service modeling) از چرخه حیات معماری سرویس گرا آشنا شدیم و اهمیت فاز تشخیص سرویس ها بیان گردید.

در این فصل به منظور چلوگیری از بروز ابهام، برای هر یک از مفاهیم موجود در دامنه مسئله تشخیص سرویس ها، تعریف روشن و شفافی ارائه خواهد شد. همچنین برخی از اصلی ترین فاکتورهای کیفی سرویس ها بیان خواهند گردید. این فاکتورهای کیفی به عنوان معیاری جهت سنجش ویژگی کیفی سرویس های تشخیص داده شده حین فرآیندهای تشخیص سرویس ها ، مورد توجه قرار خواهند گرفت. در آخر نیز فاکتور دانه بندی سرویس ها به عنوان فاکتوری تاثیر گذار بر معیارهای کیفی سرویس ها معرفی شده و نحوه ارتباط آن با سایر فاکتورهای کیفی تشریح می گردد.

۲.۱ تعاریف اصلی

همانطور که پیش از این عنوان شد در ادامه برای هریک از مفاهیم م.جود در دامنه تشخیص سرویس ها

(service identification) تعریفی ارائه خواهد گردید .

سرویس

کاری که به وسیله یک سرویس دهنده انجام میشود که ممکن است انجام یک درخواست کوچک روی داده مانند دریافت یا ذخیره اطلا عات باشد یا مربوط به انجام کاری پیچیده تر مانند چاپ یک تصویر باشد. در معماری سرویس گرا معمولا سرویس را به صورت خاص تر معرفی می کنند و به نوعی مضمون آن را با شئ و مؤلفه مرتبط می دانند. واضح است که ضرورتی برای استفاده از متدولوژی شئ گرا یا مبتنی بر مؤلفه جهت معماری سرویس گرا وجود ندارد و اصلا همان گونه که بارها گفته شده معماری سرویس گرا مستقل از سکو است، اما می توان گفت مفهوم و مضمون »سرویس« شباهت هایی با شئ و مؤلفه داشتهاست.

از آنجا که در حیطه نرم افزارهای مقیاس وسیع در سطح سازمان ها استفاده از سطح تجرید کلاس ها، اشیا و ... به علت گستردگی حیطه نرم افزار باعث افزایش پیچیدگی می گردد، از مفهوم سرویس به عنوان ابزاری جهت بالا بردن سطح تجرید و در نتیجه کاهش پیچیدگی استفاده می کنیم. به عبارت دیگر سرویس ها سطح بالایی از تجرید را معرفی می کنند. در سازمان های بزرگ که از چندین حرفه مرتبط با هم تشکیل شده اند، می توان برای پاسخ به نیازهای هر حرفه، سرویسی ارائه نمود و این سرویس ها را به منظور پوشش تمامی نیازهای سطح سازمان با هم مجتمع کرد.

از جهت دیگر، سرویس ها را می توان مصداق تکمیل شده IC های نرم افزاری دانست. همانطور که در دنیای سخت افزار با استفاده چینش مختلف از IC ها می توان مدارهای مختلف را تولید کرد ایده

استفاده از چنین تکنیکی در ذهن مهندسین نرم افزار پدید آمد. به این منظور ابتدا روش های توسعه سیستم های نرم افزاری مبتنی بر مؤلفه ها (component based development) یا CBD معرفی شدند و در ادامه روش های مبتنی بر سرویس یا روش های سرویس گرا (service oriented) ظهور پیدا کردند.

مؤلفه

همانطور که پیش از این اشاره شد، مولفه ها نقش ICهای نرم افزاری را دارند. علاوه بر این بارزترین ویژگی مولفه مستقل از سکو بودن (platform independent) آنهاست. این ویژگی به این معناست که مولفه ها به پیاده سازی خاصی وابسته نیستند و می توانند روی سکوهای سخت افزاری ،سیستم عاملی و نرم افزاری متفاوتی اجرا شوند.

اصلی ترین تفاوت میان مؤلفه ها و سرویس ها در این است که سرویس ها از طریق شبکه ارتباطی قابل دستیابی هستند. این ویژگی باعث افزایش چشمگیر قابلیت استفاده مجدد سرویس ها می گردد. به این ترتیب برای استفاده از سرویس های مشخص با کارکرد معین استفاده به پیاده سازی آنها نیست بلکه می توان در صورت یافتن سرویس هایی با قابلیت های مورد نیاز از طریق برقرای تفاهم نامه با ارائه دهنده سرویس از آن استفاده کرد. همچنین سطح تجرید سرویس ها از سطح تجرید مولفه ها بوده و به این منظور استفاده از سرویس ها در سیستم هایی با دامنه های وسیع به علت کاهش پیچیدگی توصیه می گردد.

معماری سرویس گرا

»معماری سرویس گرا« مفهومی جدید نیست و از دهه ۹۰ وجود داشته است ولی آنچه جدید است توانائی اجرا و عینیت بخشیدن به آن است که به کمک ابزارها و پروتکل های مربوطه میسر شده

است. این معماری توسط دو شرکت IBM, Microsoft بوجود آمد، که هر دو شرکت طی سالهای اخیر از حامیان اصلی سرویسهای وب و عامل بسیاری از ابداعات جدید در حیطه سرویسهای وب، مانند UDDI ,WSE بوده اند. 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 معماریهای نرم افزاری موجود به حد نهایی قابلیتهای خود رسیدهاند. معماری مبتنی بر سرویس قدم تکاملی بعدی برای کمک به سازمانها جهت مدیریت چالشهای پیچیده است. معماری مبتنی بر سرویس حالت بلوغ یافته معماری مبتنی بر اجزا، طراحی مبتنی بر واسطه (شی گرا) و سیستم های توزیع شده است. در معماری مبتنی بر اجزا عملکرد کلی به کارهای کوچک تری تقسیم میشود که هر یک در یک جزء بسته بندی خواهند شد. یک سیستم توزیع شده، تعمیمی از یک معماری مبتنی بر اجزا است که به اجزائی که در موقعیتهای فیزیکی مختلف وجود دارند، اشاره می کند. مهم ترین مزیت معماری مبتنی بر اجزا سهولت در استفاده مجدد و تغییر هدف اجزای خاص و سهولت در امر نگهداری سیستماست. استفاده مجدد و تغییر هدف معمولاً مهم ترین پیشران های کسب و کار جهت استفاده از این نوع معماری در دهه ۹۰ میلادی بودهاست. بر اساس منطق معماری مبتنی بر سرویس، سیستمهای نرم افزاری بزرگ می توانند از گردآوری مجموعههایی از عملکردهای مستقل و قابل استفاده مجدد تشکیل گردند. برخی از این عملیات می تواند از طریق سیستمهای 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 میدهد. سیستم به عنوان یک کل به صورت مجموعه ای از تعاملات بین این سرویس ها طراحی میشود. معماری مبتنی بر سرویس، سرویس هایی را که سیستم از آنها تشکیل شده را تعریف میکند و تعاملات لازم بین سرویسها جهت ارائه رفتار مشخص را توصیف میکند و در نهایت سرویسها را به یک یا چند پیاده سازی در تکنولوژی های خاص تصویر میکند. معماری سرویس گرا بر اساس استفاده از اشیاء و اجزا توزیع شده است و قدم تکامل بعدی در محیطهای محاسبه ای است. این معماری در حال حاضر مدل مرجع استانداردی ندارد؛ اما پیاده سازیهای موجود مفاهیم مشترکی را مورد استفاده قرار میدهند.[۹]

تفاوت سرویس و معماری سرویس گرا

آن گونه که به نظر میرسد در مورد ارتباط میان معماری سرویس گرا و سرویسهای وب نوعی سردرگمی عمومی وجود دارد. در یکی از گزارش های Gartner مورخ آوریل ۲۰۰۳، V. Natis Yefim این گونه تقاوت میان آنها را شرح می دهد: "سرویس های وب راجع به مشخصه های تکنولوژی هستند، در حالی که معماری سرویس گرا یک قاعدهٔ طراحی نرم افزار است. شایان ذکر است که WSDL سرویس های وب یک استاندارد تعریف رابط مناسب معماری سرویس گرا است: این نقطهای است که سرویس های وب و معماری سرویس گرا اساسا به یکدیگر پیوند می خورند." اساسا، معماری سرویس گرا یک الگوی معماری است، در حالی که سرویسهای وب سرویسهای پیادهسازی شده توسط مجموعه ای از استانداردها میباشند؛ سرویس های وب یکی از روش هایی است که شما با استفاده از آن می توانید معماری سرویس گرا را پ یادهسازی نمایید. مزیت پیادهسازی معماری سرویس گرا با سرویس های وب این است که شما به یک رویکرد بی طرفانه نسبت به پلات فرم به منظور دستیابی به سرویس ها و قابلیت همکاری بیشتر با سایر قسمت ها دست می یابید همچنان که فروشندگان بیشتر و بیشتری مشخصههای بیشتر و بیشتری از سرویسهای وب را پشتیبانیمی نمایند

چرا از معماری سرویس گرا استفاده میکنیم؟

واقعیت موجود در سازمان های IT این است که زیربنا در میان سیستم های عامل، برنامه های کاربردی، نرم افزارهای سیستمی، و زیربنای کاربردی به صورت ناهمگن است. برخی برنامه های کاربردی موجود برای اجرای فرایندهای فعلی تجارت مورد استفاده قرار گرفته اند، بنابراین آغاز از صفر برای ساختن زیربنای جدید یک رویکرد قابل انتخاب محسوب نمیگردد. سازمان ها باید به

شکلی سریع به تغییرات تجاری واکنش نشان دهند؛ از سرمایههای موجود در برنامههای کاربردی و زیربنای کاربردی به منظور تمرکز بر روی نیازمندی های تجاری جدیدتر استفاده نمایند؛ کانالهای جدید تعامل با مشتریان، شرکا، و تامین کنندگان را پشتیبانی کنند؛ و یک معماری که تجارت ارگانیک را پشتیبانی نماید به کار گیرند. معماری سرویس گرا با طبیعت اتصال آزادانه خود به سازمان ها امکان بهره گیری از سرویس های جدید یا ارتقای سرویس های موجود را به شیوه ای قطعه قطعه به منظور تمرکز بر نیازمندیهای تجاری فراهم می آورد، امکانی را برای قابل استفاده نمودن سرویس ها در کانال های متفاوت فراهم می سازد، و سازمان موجود و برن امههای کاربردی نسل قبل را به عنوان سرویس ها ارایه میکند، در نتیجه سرمایه های زیربنای IT موجود را حراست می نماید.

تعریف معماری سرویس گرا

برای معماری سرویس گرا تعاریف متنوع و بعضا مختلفی ارائه شده که هر کدام از نگاهی به تبیین خصوصیا ت آن پرداخته اند، برای درک بهتر این مفهوم و آگاهی از کلیه برداشتها و نگاههای موجود، در ادامه تعدادی از این تعاریف آورده شده است.

- یک چارچوب راهبردی از فناوری که به تمام سیستمهای داخل و خارج اجازه ارائه یا دریافت سرویس های خوش تعریف را می دهد[۱۷].
- روشی برای طراحی و پیاده سازی نرم افزارهای گسترده سازمانی به وسیله ارتباط بین سرویس هائیکه دارای خواص اتصال سست، دانه درشتی و قابل استفاده مجدد هستند[۱۸].
- سبکی از معماری که از اتصال سست سرویسها جهت انعطاف پذیری و تعامل پذیری حرفه و بصورت مستقل از فناوری پشتیبانی می کند و از ترکیب مجموعه سرویس های مبتنی بر حرفه تشکیل شده که این سرویس ها انعطاف پذیری و پیکربندی پویا را برای فرآیندها محقق میکنند[۱۹].

- چارچوبی وسیع و استاندارد که سرویس ها در آن ساخته، استقرار و مدیریت می شوند و هدفش افزایش چابکی زیر ساختهای فناوری اطلاعات در جهت واکنش سریع به تغییرات در نیازهای کسب و کار میباشد[۲۰].

معماری سرویس گرای مقدماتی

معماری سرویس گرای مقدماتی، راهی برای تبادل اطلاعات بین عاملهای نرم افزاری بوسیله پیغام تعریف می نماید. این عامل ها، 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.

معماری سرویس گرای توسعه یافته

معماری مقدماتی در لایه پایینی این معماری لایه ای قرار گرفته است. لایه ترکیب سرویس در معماری توسعه یافته، شامل توابع و نقش های لازم برای یکپارچه کردن چند سرویس بعنوان سرویس ترکیبی می باشد. سرویس ترکیبی بدست آمده، توسط Aggregator Service بعنوان یک سرویس مقدماتی استفاده میگردد و یاتوسط درخواست کنندگان سرویس بکارگرفته میشود. Service Aggregator

تهیه کننده سرویسی است که سرویس های ارائه شده توسط سایر تهیه کنندگان را یکپارچه مینماید 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 سرویسهای 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 سرویسها : بکارگیری، مجتمع سازی و دسته بندی ویژگیهای سرویسهای ترکیب شده جهت بدست آوردن خواص ترکیبی جدید که دربردارنده کارایی، هزینه، امنیت، جامعیت، قیاس پذیری، در دسترس بودن و قابلیت اطمینان می باشد.

مفاهیم اصلی در معماری سرویس گرا

اتصال سست (loose coupling)

یک ویژگی برای سیستمهای اطلاعاتی است که در آن واسط های بین اجزاء(ماژولها) به گونه ای طراحی می شوند که وابستگی بین این اجزاء حداقل شود و در نتیجه ریسک اثر تغییر یک جزء بر سایر اجزاء کاهش یابد. درمعماری سرویس گرا منظور از اتصال سست قابلیت تعامل بین سرویس ها به صورت مستقل از کدنویسی و مکان سرویسها است، بگونه ای که سرویس ها در زمان اجرا می

توانند تغییر مکان داده، روالهای داخلی خود را تغییر دهند یا حتی از یک فناوری جدید تر استفاده کنند، بدون اینکه تاثیری منفی بر سرویس گیرندگان گذاشته شود.

هم نواسازی و هم خوانی (orchestration)

دو واژه پر کاربرد در حوزه کسب و کار و معماری سرویس گرا که معمولا به جای هم اشتباه گرفته می شوند، هم نواسازی و هم خوانی نام دارند. هم نواسازی در خصوص ترتیب اجرای سرویس ها در فرآیند بحث می کند، ارکستر اصلی مجموعهای از سرویس ها را فراخوانی می کند تا نتیجه مورد نظر حاصل شود و فرآیند تکمیل گردد، ممکن است سرویسهای خارج سازمان نیز در این راستا فراخوانی و استفاده شوند، این کار با کمک موتور فرآیند محقق می شود. در عوض هم خوانی به فرآیندهایی گویند که بدون موتور فرآیندی(رهبر ارکستر) اقدام به تبادل پیام کرده و ترتیب و توالی پیامهای مبادلاتی را خود بازیگران ثبت و کنترل میکنند. در هم نواسازی، یک کنترل کنندهٔ مرکزی، جریان گردش کارها را بین چندین عامل (سرویس، کارگر، سیستم، ...) تقسیم می کند.یکی از کاربرهای این مفهوم در شکستن فرآیند های بزرگ به اجزای کوچکتر است به طوریکه این اجزا تحت نظارت هم نواساز اصلی عمل نموده و نتیجه آنها برای همان هم نواساز ارسال شود. این کار پیچیدگی کار را کاهش میدهد، بدین ترتیب منطق جریان کار به صورت جداگانه نگهداری میشود و بسط و تغییر آن ساده تر می شود. اجزاء(عامل) نباید دانشی از منطق جریان کار اصلی داشته باشند، آنها فقط به در خواست های هم نواساز پاسخ می دهند و هر جزء یک واحد خود شمول و مستقل به حساب می آید.

مزایای معماری سرویس گرا

مولفان و شرکت های پشتیبانی کننده معماری سرویس گرا در خصوص مزایای استفاده از این رهیافت دلایل زیادی را مطرح کرده اند که در ادامه بعضی از آنها تشریح می شود:

۱. سیستم های چابک: معماری سرویس گرا شما را قادر میسازد تا به سرعت سیستم های خود را تغییر دهید. این چابکی هم از جهت کارکردهای سیستم و هم از جهت تغییر جغرافیائی یا ارتقاء سکوها و حتی تغییر تامین کننده فناوری می تواند باشد.

۲. یکپارچگی آسان با شرکاء داخلی و خارجی : می توان گفت قابلیت یکپارچگی سیستمها و سکوها مهمترین موردی است که معماری سرویس گرا به آن پرداخته است.

۳. استفاده مجدد : استفاده مجدد از کد برنامه یا سیستم ها، از گذشته مورد توجه متدهای تولید و توسعه نرم افزار بوده است، معماری سرویس گرا قابلیت استفاده مجدد را هم در سطح کارکردی(سرویس) و هم در سطح داده ها مهیا میکند.

٤. پشتیبانی از محصولات با طول عمر کوتاه : رقابت تجاری در دنیا به شدت افزایش پیدا کرده و نیاز به کاهش زمان بازاریابی و تولید برای محصولات جدید میباشد. معماری سرویس گرا وعده میدهد که با وجود سرویس های خوش تعریف و قابلیت استفاده مجدد از آنها در یک سازمان، پشتیبانی سریع از محصولات جدید امکان پذیر است.

٥. بهبود بازگشت سرمایه: معماری سرویس گرا مجموع هزینه صرف شده برای فناوری اطلاعات و سرویس های حرفه را به دو روش کاهش می دهد. اول با حذف هزینههای میان افزارها و فناوریهای اختصاصی و جایگزین کردن آن با فناوری های استاندارد مانند وب سرویس و دوم با ترکیب کارکردهای حرفه در غالب سرویس هائی که توسط واحدهای مختلف قابل استفاده باشد.

٦. نگاشت مستقیم فرآیندهای حرفه به فناوری اطلاعات: نقش کلیدی معماری سرویس گرا اتصال بین کسب و کار و فناوری اطلاعات است، بدین ترتیب فرآیندها میبایست از نگاه سرویس گرا دیده شوند و در سطح مدیریت حرفه پشتیبانی شوند.

۷. توسعه و اجرای تدریجی: معماری سرویس گرا یک پروژه عظیم و بزرگ و یکجا نیست بلکه از تکامل و تبدیل تدریجی سیستمهای فعلی و تعریف سرویسهای جدید بصورت تدریجی ایجاد می شود.

۸. قابلیت انعطاف و تغییر آسان از یک ارائه دهنده سرویس به دیگری : موضوع انعطاف در معماری سرویس گرا در هر دو مورد سرویس های داخل سازمانی و خارجی صدق میکند.

زبانی مبتنی بر XML که جهت توصیف ویژگیهای عملیاتی سرویس های وب استفاده میشود و دارای دو بخش تعریف واسط و پیاده سازی است. قسمت واسط برای استفاده متقاضیان سرویس بوده و ممکن است شامل چندین پیاده سازی باشد درحالیکه تعریف پیاده سازی مشخص میکند که چگونه واسط به وسیله یک ارائه دهنده مشخص پیاده سازی شده است. اجزاء تشکیل دهنده WSDL

۱. نوع(type): پارامترهای ارسالی و دریافتی را مشخص می کند.

۲. پیام (message): پارامترهای ورودی و خروجی و نوع آنها را مشخص می کند، پیام میتواند شامل چند بخش باشد.

۳. عملیات(operation): متدهای سرویس های وب بوده و دارای پیامهای ورودی و خروجی هستند.

٤. نوع درگاه(port type): مجموعه ای از عملیات است.

٥. مقیدسازی(binding): مشخص می کند چگونه عملیات مربوط به نوع درگاه فراخوانی می شود.

٦. سرویس(service): مجموعه ای از نقاط انتهائی.

UDDI

برنامه واسطی است برای انتشار و شناسائی سرویسهای وب و شامل یک مخزن میشود که ارائه دهندگان به انتشار و تبلیغ سرویس خود میپردازند تا دیگران بتوانند آن را شناسائی کنند. از نظر مفهومی مخزن شامل صفحات سفید(اطلاعات سرویس ها)، صفحات زرد(دسته بندی صنعتی) و صفحات قرمز(اطلاعات فناوری) خواهد بود و دارای این ویژگی ها است:

- کتابچه ای برای ذخیره اطلاعات مربوط به سرویس های وب است
- در آن واسط سرویس های وب که توسط WSDL توصیف شده است، ذخیره میشود.
- ارتباطات با آن توسط SOAP است.

BPEL

زبان اجرای فرآیند های حرفه با این مشخصات:

- زبانی مستقل از سکو و مبتنی بر XML
- زبانی برای توصیف رفتار فرآیند های حرفه به کمک سرویس ها
- دارای ساختارهایی برای کنترل جریان و شرطهای انشعاب است.
- قابلیت پوشش به موارد پیچیده تری چون فرآیندهای تودر تو و الحاق و شکست زیر فرآیندها را دارد
- به عنوان یک استاندارد با حق امتیاز رایگان ارائه شده
- از WSDL برای توصیف واسط سرویس ها استفاده میشود.

متدولوژی بهبود مداوم برای معماری سرویس گرا

سیستم های سرویس گرا می بایست بصورت تدریجی و تکاملی تهیه و توسعه داده شوند، ایجاد سازمان مبتنی بر سرویس های اتوماسیون شده بصورت یکدفعه و در یک پروژه میسر نیست، بنابراین لازم است توسعه تدریجی مد نظر قرار گیرد، از طرف دیگر پس از چندین مرحله و پیاده سازی تمامی سرویس ها در سازمان کار پایان نمیپذیرد، خصوصیات سیستم های سرویس گرا تغییرات آنها بر طبق نیازهای حرفهاست، ایجاد سرویس های جدید یا تغییر و اصلاح پیاده سازی سرویس های موجود باید در نظر گرفته شود و لذا توسعه مداوم باید در دستور کار مدیران فناوری اطلاعات قرار گیرد. در ادامه راهنمائی برای ایجاد معماری سازمانی معرفی خواهد شد، اگرچه این روش توسط مولف آن تحت عنوان متدولوژی بهبود مداوم ارائه شده، ولیهمانطور که گفته شد معماری سازمانی هم اکنون دارای یک متدولوژی مدون و مورد قبول نبوده و آنچه هم اکنون تحت عنوان متد وجود دارد درحقیقت راهنمائی ها و تمرین هایی در این حوزه محسوب میشوند[۷]

مراحل متدولوژی

۱. تعریف و طراحی سرویس

این مهمترین مرحله در متدولوژی بوده و نیازمندی های مربوط به آن از طریق مشتریان داخلی، صاحبان حرفه و شرکاء تجاری بدست می آید. برای تعریف سرویس سه روش وجود دارد:

- تعریف وپیاده سازی سرویسهای جدید بر اساس نیازمندیها: در این روش سرویس های مورد نیاز بصورت کامل تعریف وپیاده سازی میشوند. این سرویسها بگونه ای تعریف و طراحی میشوند که علاوه بر برآورده نمودن نیاز درخواست کنندگان مستقیم آن، توسط سایر سرویسها نیز قابل فراخوانی و استفاده باشند.
- تعریف سرویس های جدید بر اساس مولفه ها یا سیستم های اطلاعاتی موجود: در این روش واسطی برای استفاده از کارکردها و امکانات سیستمهای اطلاعاتی موجود تهیه میشود. این

نوع سرویس ها فاقد منطق حرفه یا کد پیاده سازی بوده و صرفا واسطی برای تعامل با سیستم های موروثی هستند و حیات آنها وابسته به این سیستم ها است.

- تعریف سرویسها با ترکیب سرویسهای موجود: در این روش درصورتیکه واحدهای سازنده اصلی برای براورده کردن نیازمندیها بصورت آماده موجود باشد، سرویس جدید تنها نیاز به فراخوانی مجموعه ای از سرویس های موجود وپایه دارد. این روش آسان ترین و مقرون به صرفه ترین روش است و طراحان باید تلاش کنند حداکثر استفاده را از قابلیت استفاده مجدد نمایند البته به شرطی که ابتدا سرویسهای پایه(واحدهای سازنده) تعریف و پیاده سازی شده باشند.

در این راستا طراح سرویس، مدلی را که شامل تعریف و توصیف سرویسهای مورد نیاز است به کمکزبان های موجود مدلسازی مینماید، وی همچنین باید طریقه پیاده سازی این سرویس ها را تجزیه و تحلیل نموده و چگونگی ایجاد سرویس ها را براساس سه روش گفته شده، مشخص نماید.

۲. طراحی کیفیت سرویس

در این گام بهترین سکو و فناوری برای پیاده سازی و استقرار سرویس ها مد نظر قرار می گیرد و شاخصهایی مانند قابلیت اطمینان، امنیت، در دسترس بودن و کارائی مورد بررسی قرار گرفته و فرآیند طراحی و انتخاب سرویس ها تکمیل میشود. در این راستا معمار یا طراح کیفیت سرویس جزئیات مقید سازی سرویس ها و چگونگی پیاده سازی و استقرار آنها را مشخص می کند و روش تحقق شاخصهای کیفیت سرویس را تعیین می نماید.

۳. پیاده سازی و استقرا سرویس ها

در این گام بر حسب مدلهای تهیه شده در گامهای قبلی، سرویس ها پیاده سازی و مستقر می شوند. در پیاده سازی بایست به موضوعاتی چون پیکربندی، مدیریت نسخه ها و رضایتمندی از سرویس ها توجه شود. برای گام استقرار طبق قاعده ابتدا یک مرحله پیش ارائه وجود دارد و سپس استقرار نهائی

صورت میپذیرد، برآورده نمودن نیازهای کیفیت سرویس نظیر قابلیت اطمینان، امنیت و کارائی نیز از وظایف این فاز است. اگرچه تست به عنوان یک گام جداگانه در متدولوژی دیده نشده است اما باید مورد توجه قرار گیرد، فعالیتهای مربوط به تست و تولید بستههای تست قبل از استقرار سرویس ها ضروری است.

٤. ارکستریشن سرویس ها

بعضی سرویس ها را میتوان بدون پیاده سازی و از طریق فراخوانی سایر سرویس ها ایجاد نمود، این فعالیت کاملا با فعالیتهای مربوط به طراحی و پیاده سازی متفاوت است، اولا چگونگی تعریف توالی فراخوانی سرویس ها برای ایجاد سرویس جدید مستقل از موارد مربوط به طراحی است و دوما از نظر زمانی تا سرویس های پایه طراحی و پیاده سازی نشوند، تعریف سرویس های ارکستریشنی ممکن نخواهد بود. اگرچه از نظر هزینه و زمان ارکستریشن سرویس ها بسیار مقرون بصرفه و مطلوب بوده و یکی از اهداف معماری سرویس گرا نیز ایجاد واحدهای قابل استفاده مجدد می باشد ولی این گونه سرویس های ترکیبی دارای مسائل خاص خود هستند چرا که دارای ارتباط و وابستگی محکمی با سایر سرویس ها بوده و می بایست موارد مهمیمد نظر قرار گیرد بطوریکه سرویس گیرنده تفاوتی بین این سرویسها و سرویس های پایه احساس نکند.

٥. انتشار سرویس ها

بعد از اینکه طراحی و پیاده سازی سرویسها پایان یافت نوبت به انتشار آنها می رسد، برای این منظور این فعالیتها انجام می شود: •فایل WSDL دراختیار سرویس گیرندگان قرار می گیرد، در صورتیکه سرویس گیرندگان داخلی باشند این فایل بصورت محلی در اختیار آنها قرار می گیرد ولی در صورتیکه سرویس برای استفاده عمومی باشد، فایل میبایست بر روی اینترنت قرار گیرد. •توصیفات فنی و غیر فنی سرویسها باید تعیین شود، دسته بندی های مختلفی از انواع سرویسها از نظر نوع سرویس و

مشخصات فنی وجود دارد، این مشخصات باید به گونه ای مستند و منتشر شود که متقاضیان به راحتی بتوانند سرویس مورد نظر خود را شناسائی و فراخوانی نمایند.

٦. معاهده سطح سرویس

شامل توافق های بین منتشرکنندگان و دریافت کنندگان سرویس است و جنبههای مختلفی از جمله قابلیت اطمینان، امنیت و کارائی را در بر می گیرد. قابلیت اطمینان: برای هر سرویس می بایست معیارهای قابل اندازه گیری برای قابلیت اطمینان آن فراهم شود، برای نمونه دردسترس پذیری ۹۹ درصدی سرویس یکی از مقیاس های مناسب برای این منظور است، یکی دیگر از این مقیاسها تضمین تحویل(دریافت) یکباره پیام است. امنیت: تصدیق هویت و مجوزسنجی دو معیار اساسی در امنیت سرویسها هستند. تصدیق هویت جهت بررسی مجوز عاملی است که درخواست سرویس نموده و معمولا از طریق کنترل شناسه و رمز عبور انجام می شود. مجوز سنجی کنترل کننده سطوح دسترسی و مجوزهائی است که عامل تصدیق هویت شده میتواند داشته باشد. کارائی: یکی از معیارهای کارائی که اهمیت بالائی نیز دارد، زمان پاسخ است و به مدت زمانی گفته می شود که سرویس دهنده پس از دریافت درخواست، پاسخ می دهد و معمولا برحسب ثانیه یا میلی ثانیه است. یکی دیگر از این معیارها توان عملیاتی است و به تعداد پاسخ گوئی یک سرویس دهنده در یک واحد زمانی گفته میشود، مانند ۱۰۰ پاسخ در ثانیه.

۷. مدیریت و دیده بانی سرویس ها

این گام شامل مواردی چون کنترل و دیده بانی کارائی سرویس ها، تحلیل گزارشات و آمارها و هماهنگ کردنمستمر سرویس ها بر طبق معاهده سطح سرویس است.

۸. میزان کردن سرویس ها

این گام به تنظیم سرویس ها در راستای رفع نواقص و ارتقاء شاخص های کارائی اختصاص دارد، دادههای مورد نیاز برای این منظور از گام قبل بدست آورده می شود.

مدل تکامل یافته معماری سرویس گرا

چندین مدل برای سنجش بلوغ معماری سرویس گرا منتشر شده که در اینجا یکی از آنها مورد بررسی قرار می گیرد، این مدل شامل ۵ سطح می باشد : [۹].

- سطح ۱(سرویس های اولیه) : در این مرحله سازمان اولین آشنایی با معماری سرویس گرا را تجربه نموده و مقدمات برپایی آن فراهم میشود. تعدادی سرویس بصورت آزمایشی پیاده سازی و مستقر می شوند و پروتکل های اصلی چون WSDL SOAP ، مورد بررسی و استفاده آزمایشی قرار می گیرند. هدف این مرحله تحقیق و آزمایش بوده و در صورت دریافت نتیجه مطلوب مراحل بعدی آغاز می شود.
- سطح ۲(سرویس های معماری شده): این مرحله کارائی و مزایای پیاده سازی کامل و یکپارچه معماری سرویس گرا در کاهش هزینه و زمان و بهبود انعطاف پذیری را اثبات میکند. استانداردهای

بیشتری در QoS مانند WS-Security و WS-Reliable Messaging مد نظر قرار میگیرند، در این زمان افسر ارشد اطلاعاتی مسئولیت برپاسازی معماری را بصورت رسمی به دست می گیرد.

- سطح ۳(سرویس های همکار): در این مرحله ارکستریشن سرویسها و پیاه سازی سرویس های ترکیبی شروع می شود. انواع فرآیندها و پردازش ها(مبتنی بر رویداد، محاسباتی، عملیاتی و..) مورد بررسی قرار گرفته و به یکی از دو صورت پیاده سازی مستقل یا ترکیب از سایر سرویسها ایجاد می شوند، هدف این مرحله حداکثر استفاده از سرویسهای پایه برای تعریف و ایجاد سرویسهای جدید استو استانداردهائی چون BPEL مورد استفاده قرار میگیرند.

- سطح ٤(سرویس های اندازه پذیر) : تمرکز این سطح از سطوح بلوغ بر شاخص های اندازه گیری کارائی حرفه بوده و دیده بانی فعالیت های حرفه به عنوان یک اقدام کلیدی می بایست مورد توجه قرار گیرد. به دلیل وارد شدن معیارهای اندازه گیری کارائی در این مرحله، حضور و نظارت نمایندگانی از مدیریت مالی و برنامه ریزی سازمان ضروری خواهد بود.
- سطح ٥(بهینه سازی سرویس ها) : در این مرحله بهینه سازی سرویسهای سازمانی مورد نظر است وهدف آن دستیابی و محقق نمودن تکامل مستمر و تدریجی در سازمان با کمک افسر ارشد اجرائی(CEO) است.

چرخه حیات معماری سرویس گرا

بر اساس طرح IBM، برای معماری سرویس گرا میتوان یک چرخه حیات در نظر گرفت. 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 سرویسهای موجود و سرویسهای جدید این فرایندهای کسب و کار شکل می گیرد. سپس این سرمایه ها در یک محیط امن و با قابلیت تجمیع بالا نصب میشود. بعد از نصب فرایندهای کسب و کار، کاربران IBM این فرایندهای کسب و کار را هم از منظر فنی و هم از منظر فرایندهای کسب و کار مورد نظارت و مدیریت قرار میدهند. 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 رهنمودها و افق های آینده را برای پروژه معماری سرویس گرا فراهم میکنند.

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 میگیرند. فرایندهای کسب و کار حاصل برای طراحی سرویس های نرم افزاری مرتبط و سطوح سرویس جهت حمایت از این فرایندها مورد استفاده قرار میگیرند. 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 میسازد. این مدل همچنین میتواند مبنایی جهت اندازه گیری کارآیی کسب و کار باشد.

مرحله گردآوری

در طول فاز گردآوری، کتابخانه سرویسهای موجود می تواند جهت یافتن سرویسهای 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 سرویسهای مورد نیاز فراهم شد، سرویس ها جهت پیاده سازی فرایندهای کسب و کار هماهنگ میگردند.

مرحله نصب

در طول فاز پیاده سازی، مقیاس و محیط زمان اجرا جهت تامین نیازمندیهای سطوح سرویس به وسیله فرایندهای کسب وکار پیکربندی میشود. پس از پیکربندی یک فرایند کسب وکار، امکان پیاده سازی آن در یک محیط امن، مطمئن و مقیاس پذیر سرویسها وجود خواهد داشت. محیط سرویس ها به گونهای 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 نیازمندیهای کسب وکار را فراهم می آورد. این رویکرد مبتنی بر سرویس همچنین هزینه و پیچیدگی نگهداری سیستم را نیز کاهش می دهد.

مرحله مدیریت

فاز مدیریت شامل نظارت و نگهداری از زمان پاسخ و در دسترس بودن سرویس می شود. همچنین مدیریتمنابع 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.

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 تکنولوژی مرتبط ایجاد گردد. پیاده سازی سیاستهای حاکمیتی قوی میتواند منجر به پروژه های معماری سرویس گرا موفق گردد.

تشخیص سرویس ها

تشخیص سرویس ها اولین گام در فاز مدلسازی در چرخه حیات معماری سرویس گرا است. فاز شناسایی سرویس ها یکی از بحرانی ترین فازها در موفقیت پروژه معماری سرویس گرا است زیرا در این فاز نیازمندی های حرفه شناخته شده و پوشش داده می شوند.[۱] هدف این فاز تولید مجموعه ای از سرویسهای کاندید برای پروژه سرویس گرا و مشخص کردن عملیات آنها است.[۱] برای

شناسایی سرویسهای کاندید تحلیل گر در مورد استفاده از راهبردها و روش های مختلف تصمیم گیری می کند.[۲]

نقش های درگیر در این فار عبارتند از : معمار نرم افزار (نقش اصلی) و تحلیل گر حرفه (نقش کمکی). در خلال این فاز مدل کاری سرویسها تولید می شود و در پایان فاز، این مدل به معمار نرم افزار برای انجام

عملیات مربوط به فاز توصیف سرویس ها (Service Specification) تحویل داده می شود.[۱] اصلی ترین ورودی های این فاز سرویس های موجودی که سازمان به آنها (از هر طریقی مانند تملک، خریداری و ...) دسترسی دارد می باشند. این سرویس ها معمولا در انباره سرویس های سازمان موجود می باشند. البته باید توجه داشت که در سازمان هایی که اولین تجربه معماری سرویس گرای خود را انجام میدهند ،طبیعتا این ورودی وجود ندارد، و سرویس های تماما باید تولید شده و پس از انجام فازهای مختلف در انباره برای استفاده های در پروژه های سرویس گرایی آینده قرار بگیرند.[۱]

توصیف سرویس ها

پس از شناسایی مجموعه سرویس های کاندید فاز توصیف سرویس ها آغاز می گردد. این فاز توسط معمار نرم افزار به همراه طراحان (به صروت اختیاری) انجام می شود.[۱] هدف این فاز توصیف کامل عناصر طراحی معماری سرویس گرا است.

فاز توصیف سرویس ها را می توان وظیفه مربوط به معماری در فرآیند معماری سرویس گرا دانست. این دید اهمیت این فاز را به خوبی نمایان می سازد.[۱]

برای روشن تر شدن تمایز میان فاز توصیف سرویس ها و شناسایی سرویس ها می توان فاز شناسایی سرویس ها را (Service Identification) را به عنوان فعالیت های مربوط به تحلیل مدل سرویس و فاز توصیف سرویس ها (Service Specification) را به عنوان فعالیت های مربوط به طراحی مدل سرویس دانست.

در خلال فاز توصیف سرویس ها مدل سرویس به روشنی مشخص می شود و در پایان این فاز، این مدل به طراحان برای انجام فاز عینیت بخشیدن به سرویس ها و توسعه دهندگان برای پیاده سازی سرویس ها داده می شود.[۱]

وظایف اصلی درون این فاز عبارتند از: تحلیل زیرسیستم ها (Subsystem Analysis) ، توصیف مؤلفه ها (Component Specification) و تخصیص سرویس ها (Service Allocation).
در این فاز ویژگی ها و قابلیت های سرویس ها تشریح می گردند و به سؤالهایی از جمله : سرویس چه خدمتی ارائه می دهد؟ منطق داخلی سرویس ها چیست؟ نیازمندی های غیر وظیفه مندی هر سرویس چیست؟ و... جواب داده می شود.

تحلیل زیرسیستم ها

ورودی این مرحله زیرسیستم هایی (سرویس ها) که از مرحله تجزیه دامنه بدست آمده اند می باشند و در اینمرحله وابستگی بین این زیر سیستم ها و جریان های بین آنها مشخص می گردد.[۳]
در تحلیل زیرسیستم ها مدل های اشیاء به منظور مشخص کردن نحوه انجام کار در داخل زیر سیستم و همچنین نمایش زیر سیستم های درون زیر سیستم در حال بررسی (سلسله مراتب زیرسیستم ها) تولید می شوند.
ساختار طراحی زیر سیستم ها برای تولید ساختار پیاده سازی مؤلفه ها – که خود اساسی برای پیاده سازی سرویس ها هستند – استفاده می شود.[۳]
در این مرحله خصوصیات سرویس ها از جمله : وابستگی بین آنها، نحوه ترکیب آنها، پیام های بین آنها ، فاکتور های کیفیت آنها و... تعیین می شوند.برای این کار ٦ مرحله زیر انجام می گردند.

۱. تست *Litmus* :

این تست به این سؤال که "از میان سرویس های کاندید کدامیک برای پیاده سازی مناسب است؟" پاسخ می دهد. باید توجه داشت تمامی سرویس هایی که در مرحله تجزیه دامنه مشخص شدند برای پیاده سازی مناسب نیستند. برای پیدا کردن زیر مجموعه ای از این سرویس ها در روش تست

Litmus براساس موارد مختلف از جمله تطابق سرویس با نیازهای حرفه، تکراری نبودن سرویس و ... تصمیم گیری می شود. منظور از تکراری نبودن سرویس این است که وظایفی که سرویس در دست بررسی انجام می دهد توسط سرویس های دیگر پوشش داده نشوند.

۲. .مشخص کردن وابستگی میان سرویس ها :بازبینی جزئیات سرویس ها باعث مشخص شدن وابستگی سرویس به سرویس های دیگر و نحوه تعامل میان سرویس ها می شود.

۳. مشخص کردن ترکیب سرویسها و جریان میان آنها: با بررسی حیطه های کاری و فرآیندهای حرفه نحوه تولید سرویس ها از ترکیب سرویس های دیگر و جریان کاری میان سرویس ها برای انجام فرآیندهای حرفه را مشخص می کند.

٤. مشخص کردن نیازهای غیروظیفه مندی :با استفاده از نیازهای غیر وظیفه مندی ، ویژگی های کیفی سرویس ها مشخص می گردد.

٥. مشخص کردن پیام های سرویس :در این قسمت قالب پیام های ورودی و خروجی سرویس و محتوای آنها مشخص می شود.

٦. مستندسازی تصمیمات مدیریتی :برخی مواقع ترکیب سرویس ها منجر به برخی تصمیمات مدیریتی درحالت های مختلف کاری سرویس می شود. این تصمیمات مدیریتی در این مرحله مستندسازی می شوند.

تخصیص سرویسها

پس از مشخص کردن سرویس ها و جزئیات آنها در این مرحله درباره تخصیص سرویس ها به مؤلفه ها تصمیم گیری می کنیم.[۲] و [۳] اغلب این تخصیص یک-به-یک است.

همچنین در این مرحله نحوه تخصیص سرویس ها و مؤلفه به لایه های معماری سرویس گرا مشخص می شود.[۳] برای تصمیم گیری در مورد چگونگی تخصیص سرویس ها و مؤلفه ها به لایه های

معماری علاوه بر توجه به معماری نرم افزار باید به معماری محیط فنی و عملیاتی که سیستم در آن اجرا می شود توجه نمود.[۳]

توصیف مؤلفه ها

در مرحله بعد از فاز توصیف سرویس ها ، جزئیات مؤلفه هایی که سرویس ها را پیاده سازی می کنند مشخص می شود.[۳] این توصیف دربرگیرنده جزئیاتی در مورد داده های مؤلفه، قوانین، سرویس ها و نحوه پیکربندی مؤلفه می باشد. همچنین در این مرحله پیام های مؤلفه ها و خصوصیات رویدادهای آنها مشخص می گردد.

[3]

پس از مشخص کردن خصوصیات سرویس ها و نحوه اجرای سرویس ها توسط مؤلفه ها و همچنین تعریف جزئیات مؤلفه ها، باید در مورد نحوه تولید و در اختیار گرفتن سرویس ها تصمیم گیری شود.

عینیت بخشی به سرویس ها

در این فاز در مورد مباحث مربوط به "امکان سنجی فنی" پیاده سازی یا استخراج سرویس ها از سیستم های موروثی تصمیم گیری می کند.[۳] و [۸] گزینه های دیگری که در این زمینه وجود دارد عبارتند از تجمیع سرویس ها و تولید سرویس مورد نیاز، تغییر سرویس های قدیمی برای تولید سرویس مورد نیاز ، سفارشسرویس و ... .[۳]

تصمیمات دیگری که در این فاز گرفته می شود – علاوه بر تصمیمات در مورد وظیفه مندی – عبارتند از :

امنیت، مدیریت و نظارت بر سرویس ها. [۳]

۲.۲ پارامتر های مهم کیفی

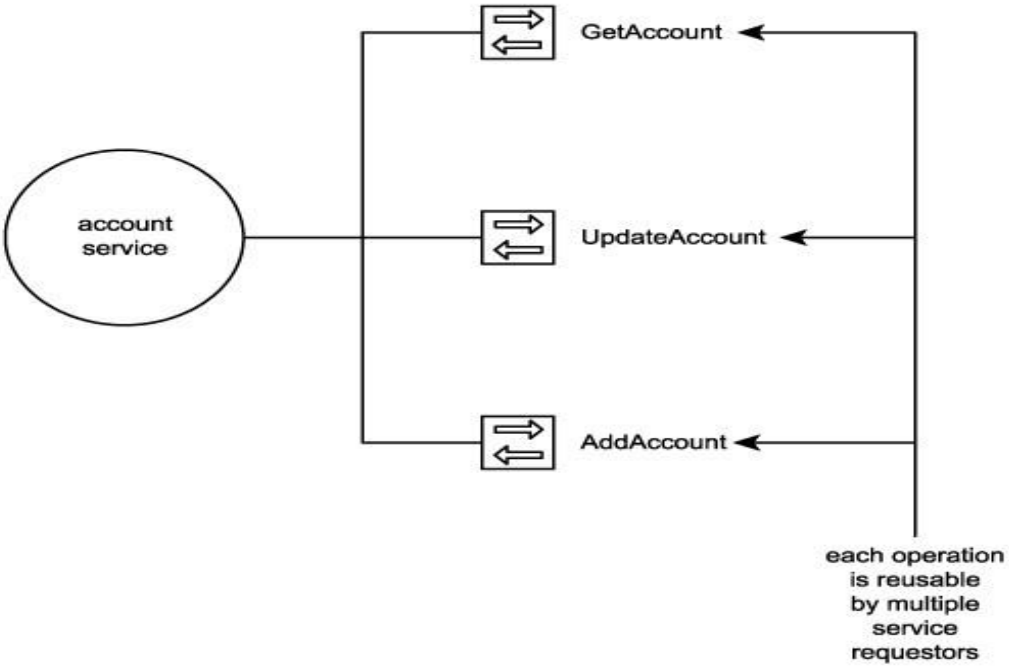
پس از آشنایی با مفاهیم موجود در زمینه معماری سرویس گرا و فاز شناسایی سرویس ها، در این بخش با معیارهای کیفی در سرویس ها بیشتر آشنا می شویم.

از آنجا که گام شناسایی سرویس ها یکی از پر اهمیت ترین گام ها در چرخه معماری سرویس گرا به شمار می رود، آشنایی با برخی معیارها در مورد کیفیت سرویس ها و سنجش سرویس های معرفی شده طی مراحل این فاز بر اساس این معیارها می تواند تا حدی دید درستی نسبت به کیفیت فرآیندهای شناسایی سرویس ها و پروژه سرویس گرایی در اختیار بگذارد. به عبارت دیگر به علت اهمیت گام تشخیص سرویس ها، باید به نحوی از کیفیت خروجی های این گام که همان سرویس های کشف شده در این گام هستند، شناخت پیدا کرد. استفاده از معیارهای کیفی به همین منظور انجام می گیرد. باید توجه داشت که معیارهایی که در ادامه معرفی می شوند، تمامی معیارهای معرفی شده در زمینه سنجش کیفیت سرویس ها نبوده بلکه معیارهای پر اهمیت در این زمینه می باشند. همچنین نیاز نیست تمامی سرویس های کشف شده در فاز تشخیص سرویس ها تمامی معیارهای زیر را به تمامی داشته باشند.

- قابلیت استفاده مجدد (reusability)

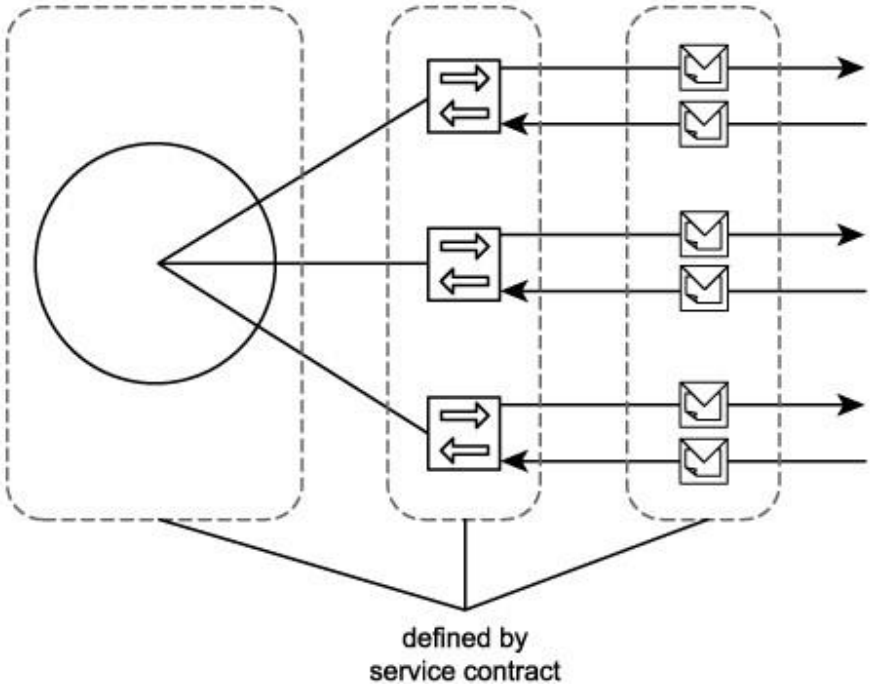
یکی از اصلی تریم معیارها در تشخیص سرویس ها قابلیت استفاده مجدد سرویس ها می باشد. همانطور که پیش از این گفته شد، سرویس ها در ادامه ایده IC های نرم افزاری بوجود آمدند. اصلی ترین ایده در این زمینه استفاده مجدد از این ICهای نرم افزاری به منظور تولید سیستم های پیچیده گوناگون است. همانطور که با ترکیبات مختلف IC های سخت افزاری می توان مدارهای گوناگون و متنوعی را تولید کرد، استفاده از این IC های نرم افزاری نیز می توانند تولید سیستم های نرم افزاری مختلف و گوناگونی را موجب گردند.

این قابلیت باعث کاهش چشمگیر هزینه و زمان تولید سیستم های نرم افزاری می گردد.



شکل شماره ۲ -۱ - قابلیت استفاده مجدد [۱]

قراردادهای سرویس مناسب برخی اطلاعات مفهومی در مورد شیوه انجام عملیات توسط سرویس ها را نیز در اختیار قرار می دهند. این اطلاعات هنگام توافق میان فراهم کننده سرویس و متقاضی آن مورد استفاده قرار می گیرد.



شکل شماره ۲ -۲ - قرارداد مشترک [۱]

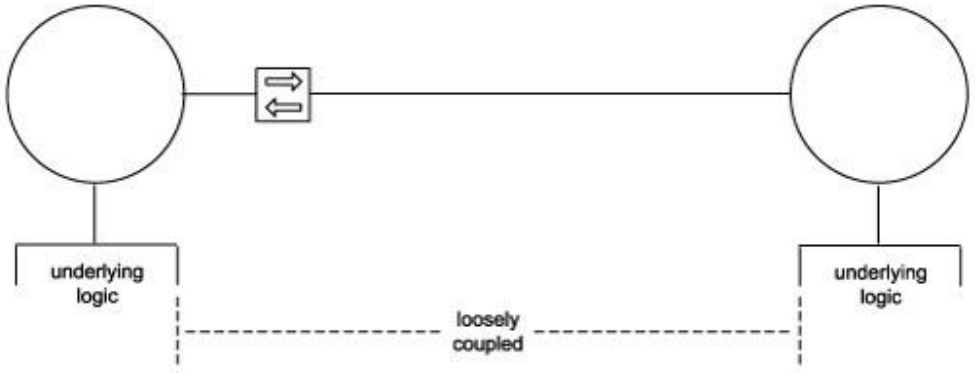
- وابستگی کم میان سرویسی (loosely coupled)

به این علت که حیطه وظیفه مندی حرفه ها و سازمان ها به صورت پویا تغییر می کند، و سیستم های نرم افزاری این سازمان ها موظف به پاسخگویی به نیازهای سازمان هستند، در نتیجه سیستم های نرم افزاری باید پاسخگوی به تغییرات در وظیفه مندی های مورد نیاز باشند.

به همین علت توسعه سیستم ها به گونه ای که در برابر تغییرات قابلیت انعطاف پذیری بالایی داشته باشند یکی از دغدغه های اصلی در حیطه طراحی . توسعه سیستم های نرم افزاری است. این امر باعث کاهش هزینه و زمان مورد نیاز جهت اضافه کردن قابلیت های جدید به سیستم های نرم افزاری می گردد.

یکی از راههای دستیابی به این ویژگی در معماری سرویس گرا، توسعه سیستم ها به نحوی که ارتباط محدود با یکدیگر داشته باشند است. به عبارت دیگر در معماری سرویس گرا به منظور دستیابی به انعطاف پذیری بالا سعی می شود سیستم از سرویس هایی تشکیل شود که دارای ارتباطات کم و

محدود با یکدیگر باشند. این امر سبب می شود در صورت نیاز به تغییر یک سرویس، سرویس های کمتری به منظور ارتباط با آن دستخوش تغییر شوند . همین امر انعطاف پذیری سیستم را افزایش داده و هزینه و زمان مورد نیاز برای پاسخگویی به تغییرات را کاهش می دهد.

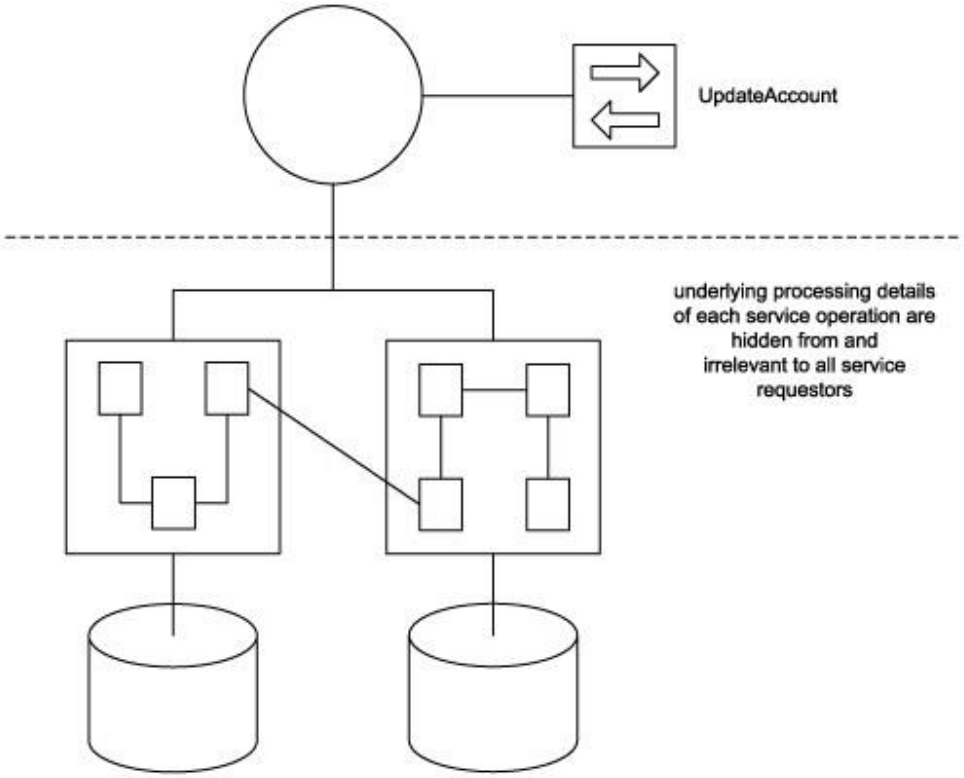


شکل شماره ۲ -۳ - وابستگی کم میان سرویسی [۱]

- تجرید (abstraction)

یکی از دلایلی که از معماری سرویس گرا عمتدا در توسعه سیستم های مقیاس وسیع استفاده می شود، این است که در این سیستم ها به علت گشتردگی حیطه مسئله با موجودیت ها و ارتباطات بسیار زیادی مواجه هستیم. در صورتی که در توسعه این سیستم ها بخ.اهیم از روش های سنتی استفاده کنیم به علت مواجه با حجم زیاد موجودیت ها و ارتباطات دچار سردرگمی خواهیم شد. به همین علت برای کاهش پیچیدگی در این سیستم ها از موجودیتی به نام سرویس به منظور بالا بردن سطح تجرید و در نتیجه کاهش پیچیدگی استفاده می شود.

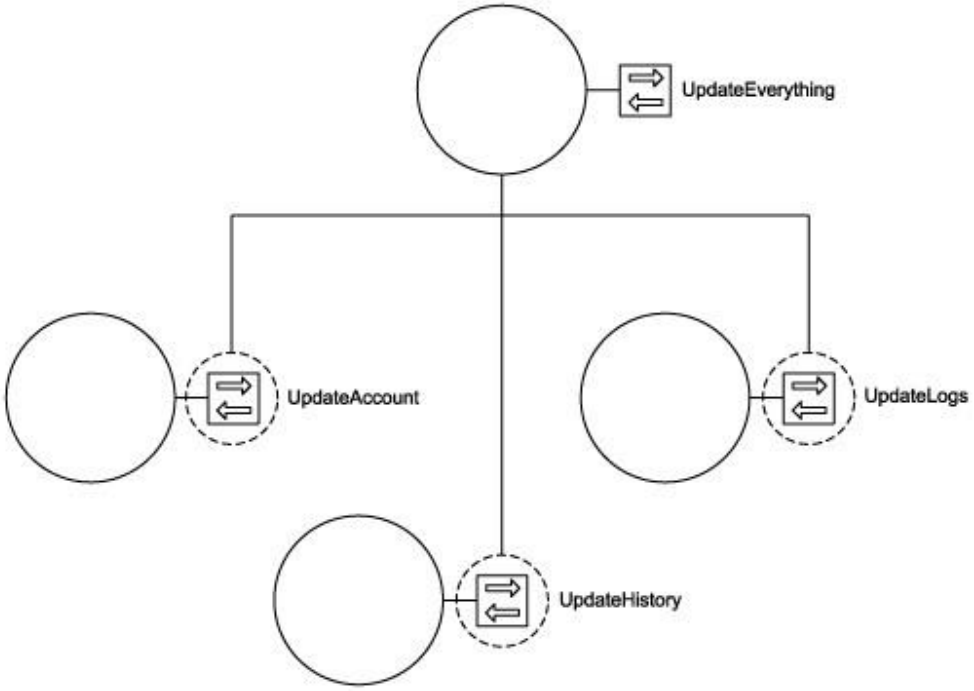
سرویس ها به صورت جعبه سیاه هایی عمل می کنند که محتویات درونشان را از جهان خارج پنهان می سازند. این امر موجب کاهش موجودیت ها و ارتباطات آشکار در حیطه مسئله شده و پیچیدگی را به میزان چشمگیری کاهش می دهد. به همین علت توصیه می گردد سرویس ها از سطح تجرید بالایی برخوردار باشند.



شکل شماره ۲ -٤ ¯ تجرید [۱]

- قابلیت ترکیب (composable)

سرویس ها هر کدام وظیفه مندی خاصی را پوشش می دهند. یکی از ویژگی های اصلی سرویس ها قابلیت ترکیب می یاشد. این امر به این خاطر است که گاهی نیاز است به وظیفه مندی جدیدی که حاصل از ترکیب وظیفه مندی های پوشش داده شده توسط دو یا چند سرویس است پاسخ داده شود. به این منظور می توان از ترکیب این سرویس ها استفاده کرد. این ویژگی باعث افزایش قابلیت استفاده مجدد سرویس ها شده و هزینه و زمان توسعه سرویس های جدید را به صورت قابل ملاحظه ای کاهش می دهد.

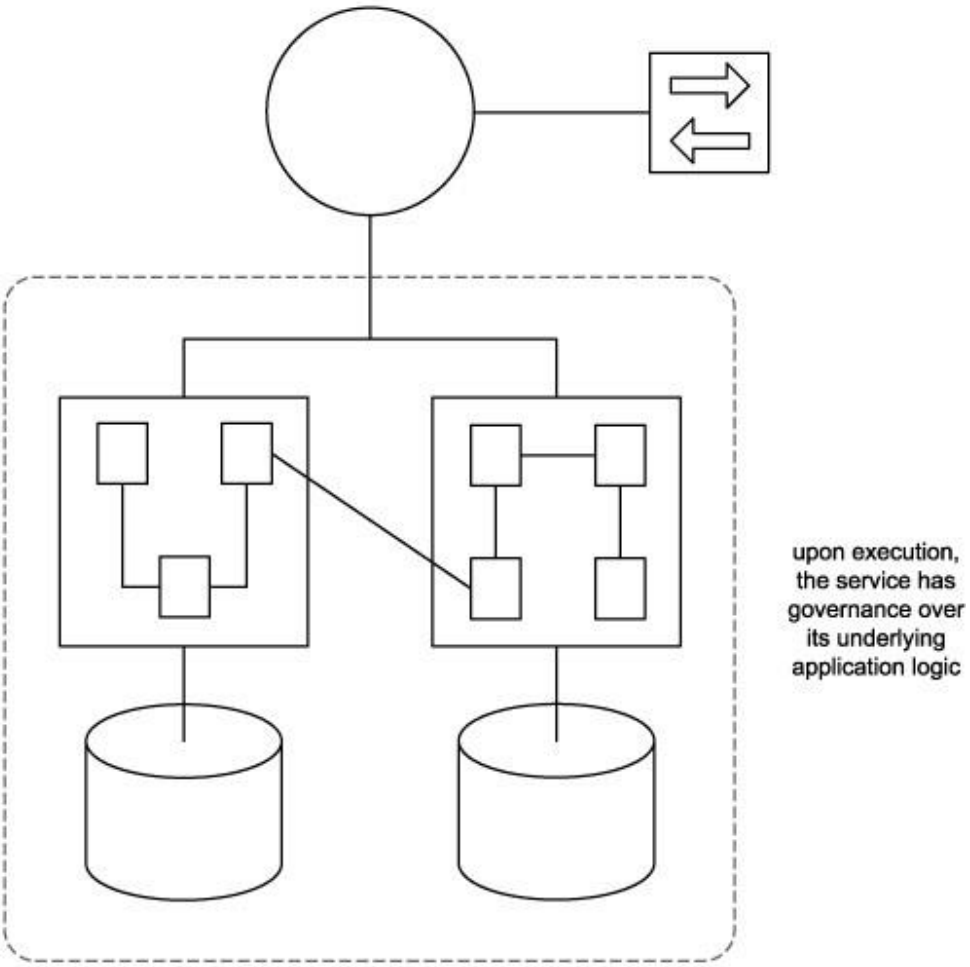


شکل شماره ۲ -۵ - قابلیت ترکیب [۱]

- خودمختاری (autonomous)

خودمختاری سرویس ها نیازمند آن است که محدوده سرویس ها به خوبی تعریف شده باشد. این ویژگی باعث کاهش وابستگی سرویس به دیگر سرویس ها شده و سطحی از استقلال را برای سرویس ها فراهم می آورد. خودمختاری سرویس ها جزو اولین مواردی است که حین تجزیه دامنه مسئله به سرویس ها باید در نظر گرفته شود.

البته باید توجه داشت خودمختاری سرویس به این معنا نیست که سرویس در حیطه وظیفه مندی خود اختیار تام دارد بلکه به این معناست که سرویس در هنگام اجرا در حیطه وظیفه مندی اش دارای اختیار تام است.

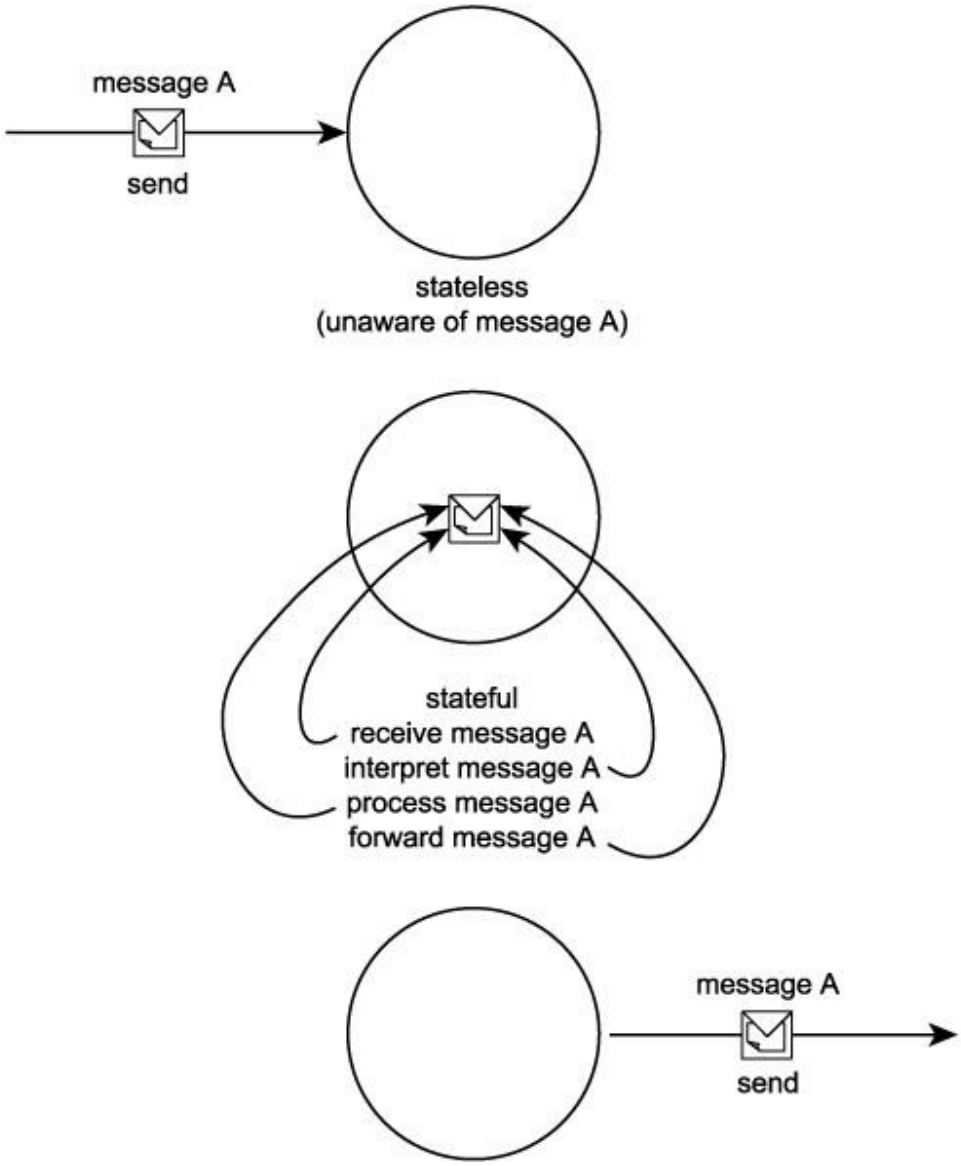


شکل شماره ۲ -٦ – خودمختاری [۱]

- نداشتن وضعیت خاص (stateless)

سرویس ها بهتر است میزان و یا مدت زمان نگهداری اطلاعات حالت خود را تا حد ممکن کاهش دهند.

هنگامی که به عنوان مثال یک سرویس در حال پردازش یک پیام (message) است، سرویس اصطلاحا دارای حالت مشخص (state full) می گردد.



شکل شماره ۲ -۷ - نداشتن حالت مشخص [۱]

البته ترجیح داده می شود سرویس ها وضعیت مشخص نداشته باشند (stateless). این امر ویژگی های قابلیت استفاده مجدد (reusability) و مقیاس پذیری (scalability) سرویس ها را افزایش می دهد.

۲.۳ بررسی فاکتور دانه بندی سرویس ها

در قسمت قبل با ویژگی های اصلی سرویس ها آشنا شدیم. این ویژگی ها در اکثر متون مربوط به سرویس ها آورده شده اند و تمامی روش های ارائه شده در زمینه تشخیص سرویس ها (service identification) بر اساس برآورده کردن تعدادی از این ویژگی ها بنا نهاده شده اند.

مسئله دیگری که در معماری سرویس گرا بر کیفیت سیستم تولید شده اثر زیادی می گذارد و در روش های ارائه شده کمتر به آن توجه شده است، فاکتور دانه بندی سرویس ها می باشد. این بدان معنی است که اگر سرویس ها بزرگ و یا کوچک باشند بر برخی از ویژگی های سیستم نهایی تاثیر مستقیم می گذارد. در زیر اثر این ویژگی بر چهار ویژگی اصلی سیستم های نرم افزاری بررسی می گردد.

- انعطاف پذیری (flexibility)

اگر سرویس ها بزرگ باشند برای تغییر در یکی از جنبه ها یا وظیفه مندی های سیستم باید دامنه گسترده ای از سیستم دستخوش تغییر شود. به همین علت بزرگی اندازه سرویس ها موجب کاهش انعطاف پذیری سیستم می گردد. در مقابل هر چه سرویس ها اندازه کوچکتری داشته باشند، برای تغییر در قسمتی از سیستم تنها تغییر در حیطه سرویس هدف کفایت می کند و به همین علت تغییر جنبه ای از سیستم سربار زیادی را ناشی نمی شود و در نتیجه انعطاف پذیری سیستم افزایش می یابد.

- قابلیت استفاده مجدد (reusability)

قابلیت استفاده مجدد یکی از بارزترین و اصلی ترین ویژگی ها در معماری سرویس گرا به شمار می رود.

همانطور که پیش از این اشاره شد، قابلیت استفاده مجدد باعث کاهش زمان و هزینه تولید سیستم های نرم افزاری شده و تا حد ممکن در توسعه سیستم های نرم افزاری تلاش در افزایش این قابلیت داریم.

دانه بندی سرویس ها بر اساس حیطه وظیفه مندی سرویس ها تعیین می شود. به عبارت دیگر هرچه سرویسی حیطه بزرگتری از وظیفه ندی را پوشش دهد ، اصطلاحا ـن سرویس را بزرگتر و هر چه حیطه کوچکتری را پوشش دهد، اصطلاحا سرویس را کوچکتر گویند. در صورتی که سرویس بزرگ باشد، حیطه بزرگتری از وظیفه مندی های حرفه را پوشش می دهد و اصطلاحا سرویس در حیطه حرفه تنظیم شده و وابستگی بیشتری به حرفه خواهد داشت. همچنین به علت تفاوت در ماهیت حرفه های گوناگون امکان استفاده مجدد از آن سرویس در حرفه های دیگر کاهش می یابد و باعث کاهش قابلیت استفاده مجدد سرویس می گردد.

در مقابل هرچه سرویس کوچکتر باشد، وابستگی کم تری به حرفه داشته و به احتمال زیاد سرویس مطابق با وظیفه مندی خاصی طراحی شده است. به علت امکان بروز وظیفه مندی های مشابه در حیطه حرفه های گوناگون امکان استفاده مجدد سرویس افزایش می یابد.

- کارآیی (performance)

از آنجا که برای برآورده کردن فرآیندهای حرفه تعدادی از سرویس ها با هم همکاری می نمایند، ارتباطات بین سرویس ها به این منظور نوعی سربار تلقی می گردند. به عبارت دیگر هنگام فراخوانی سرویس ها به منظور برآورده کردن فرآیند حرفه، مقداری از کارآیی سیستم جهت برقراری ارتباط میان سرویس های مختلف هدر می رود. حال در صورتی که سرویس ها کوچک باشند، برای برآورده کردن فرآیندهای حرفه به تعداد بیشتری سرویس نیاز است و در نتیجه ارتباطات میان سرویسی افزایش یافته و سربار بیشتری به سیستم تحمیل می گردد. در نتیجه استفاده از سرویس های کوچک باعث کاهش کارآیی سیستم می گردد.

در مقابل هر چه سرویس ها بزرگتر باشند، برای پوشش نیازمندی های یک فرآیند حرفه به تعداد سرویس کمتری نیاز بوده و در نتیجه سرباز ناشی از ارتباطات میان سرویسی کاهش می یابد. همین امر باعث افزایش کارآیی سیستم نهایی می گردد.

- پیچیدگی (complexibility)

آخرین فاکتوری که دانه بندی سرویس ها بر آن اثر می گذارد، فاکتور پیچیدگی سیستم می باشد. تاثیر دانه بندی سرویس ها بر این فاکتور تا حدودی شبیه تاثیر دانه بندی بر فاکتور کارآیی سیستم می باشد. در صورتی که سرویس ها کوچک باشند، به تعداد بیشتری سرویس جهت پوشش دامنه حرفه نیاز داریم. طبیعی است که این سرویس ها نیاز به تعامل با یکدیگر دارند. همین امر باعث می شود در صورت داشتن سرویس های کوچک پیچیدگی سیستم به علت تعداد زیاد ارتباطات افزایش یابد. در سمت مقابل هرچه اندازه سرویس ها بزرگ تر باشد، به تعداد سرویس کمتر و در نتیجه ارتباطات کمتر میان سرویسی نیاز خواهد بود. همین امر پیچیدگی سیستم نهایی را کاهش خواهد داد.

با توجه به موارد فوق مشاهده می شود نمی توان با قاطعیت عنوان کرد سرویس های بزرگتر بهتر هستند یا سرویس های کوچک تر. بلکه باید با توجه به نیازمندی های حرفه و ویژگی های دامنه مسئله با مصالحه بین فاکتورهای انعطاف پذیری، قابلیت استفاده مجدد، کارآیی و پیچیدگی اندازه مناسبی را برای سرویس ها تعیین نمود.

نتیجه گیری

به منظور دستیابی به درک یکسان از دامنه مسئله و مفاهیم موجود در آن، در این فصل برای مفاهیم موجود در حیطه سرویس گرایی و مسائل مربوط به آن از جمله سرویس، مؤلفه، معماری سرویس گرا، متدولوژی های سرویس گرا، استانداردها و پروتکل ها، مدل بلوغ معماری سرویس گرا و چرخه حیات معماری سرویس گرا و فازهای آن تعریفی یکسان ارائه شد. این امر باعث خواهد شد در ادامه مطلب ابهامی در خصوص مفاهیم مطرح شده وجود نداشته باشد.

در ادامه نیز برخی معیارهای کیفی به منظور سنجش کیفیت سرویس های کشف شده در روش های تشخیص سرویس ها معرفی گردیدند و اهمیت هر یک ذکر شد. در آخر هم فاکتور اندازه سرویس ها به عنوان فاکتوری که بر برخی ویژگی های سیستم نهایی تاثیر گذاشته اما کمتر مورد توجه است مورد بررسی قرار گرفت.

در فصول آینده روش های ارائه شده در زمینه تشخیص سرویس ها معرفی شده و ویژگی های هریک بیان خواهند شد. همچنین هر کدام از این روش ها بر اساس معیار های معرفی شده در این بخش سنجیده شده و ارزیابی خواهند گردید و نقاط قوت و ضعف آن ها بیان خواهند شد.

سپس با توجه به نقاط ضعف موجود در راه حل های کنونی، اصول راه حلی جهت پوشش نقاط ضعف موجود بیان شده و ویژگی های این راه حل عنوان خواهند گردید.

فصل سوم معرفی روش های گذشته

مقدمه

با توجه به گسترش چشمگیر استفاده از معماری سرویس گرا در توسعه سیستم های نرم افزاری مقیاس وسیع و همچنین اهمیت فاز مدلسازی در چرخه حیات معماری سرویس گرا، روش های مختلفی به این منظور ارائه شده است. ار آنجا که گام تشخیص سرویس ها به عنوان اولین گام در مدلسازی سرویس گرا و چرخه حیات معماری سرویس گرا مطرح است و وابستگی سایر گام ها به خروجی این گام، گام تشخیص سرویس ها (service identification) از اهمیت ویژه ای برخوردار است.

در فصل اول با اهمیت گام تسخیص سرویس ها و جایگاه آن در چرخه حیات معماری سرویس گرا آشنا شدیم. بدیهی است با توجه به اهمیت این گام روش های گوناگونی جهت انجام آن ارائه شده باشد. در فصل دوم نیز پس از آشنایی با مفاهیم موجود در زمینه معماری سرویس گرا، با برخی فاکتورهای کیفی جهت سنجش کیفیت سرویس ها آشنا شدیم. در این فصل قصد داریم با روش های ارائه شده در زمینه تشخیص سرویس ها آشنا شده و با توجه به فاکتورهای ارائه شده در فصل دوم به ارزیابی این روش ها بپردازیم. در آخر نیز نقاط قوت و ضعف هر یک از این روش ها را تا حد امکان شرح خواهیم داد.

۳.۱ دسته بندی روش های موجود

در مقاله ای که در سال ۲۰۰۷ توسط Jan-Willem Hubbers،Art Ligthart و Linda Terlouw در SOA Magazine ارائه شد [٤] روش های تشخیص سرویس ها (service identification) در ۱۰ رده کلی دسته بندی شدند. این رده ها بر اساس ویژگی های کلی این روش ها تعیین شده و همچنین نقاط قوت و ضعف هر یک از رده ها به صورت کلی بیان شده است. در جدول زیر این گروه ها به اختصار بیان گردیده اند:

روش	مزایا	معایب
تجزیه فرآیندهای حرفه	تطابق با نیازهای حرفه	کاهش استفاده مجدد، بروز افزونگی
وظیفه های حرفه	جلوگیری از افزونگی	کاهش استفاده مجدد
اشیاء موجودیتی حرفه	کاهش تغییرات طراحی در آینده	نیاز به تحلیل مجدد برای کشف تمامی اشیاء و سرویس ها
مالکیت و مسئولیت	شفافیت مالکیت سرویس ها	روش وابسته
هدف گرایی	تطابق با سیاست های حرفه	کاهش استفاده مجدد، بروز افزونگی
مبتنی بر مولفه	سادگی، تشخیص سرویس های جدید برای اهداف ناشناخته	مشکلات ناشی از تفاوت در ماهیت مولفه و سرویس
دارایی موجود (پایین به بالا)	کاهش زمان، کاهش هزینه	مشکلات ناشی از برنامه های با
		طراحی ضعیف
Front-Office Application Usage Analysis	سرعت، استفاده مجدد	مشکلات ناشی از برنامه های با طراحی ضعیف

زیرساخت	کاهش هزینه ها	کاهش تجرید
نیازهای غیروظیفه ای	توجه به نیازهای کیفی	روش وابسته

جدول شماره ۳-۱ - رده بندی روش های ارائه شده [٤]

در زیر هر یک از روش های فوق به اختصار توضیح داده شده اند.

- تجزیه فرآیندهای حرفه

در روش هایی که از این ایده استفاده می کنند، ابتدا تمامی فرآیندهای حرفه شناسایی شده و سپس به تجزیه هر یک از این فرآیندها پرداخته می شود. در این روش پس از تجزیه فرآیندهای حرفه به زیر فرآیندها و وظایف (task) پس از رسیدن به سطح قابل قبولی از دانه بندی، سرویس ها شناسایی می گردند.

مزیت اصلی این روش این است که به علت تمرکز بر فرآیندهای حرفه، سرویس های شناسایی شده با فرآیندهای درون حرفه تطابق دارند.

نقطه ضعف این روش در این است که به علت تمرکز بر فرآیندهای یک حرفه خاص، امکان استفاده مجدد از سرویس های شناسایی شده، به علت تفاوت در ماهیت حرفه ها، کاهش می یابد. همچنین به این علت که در روش هایی که از این منطق استفاده می کنند فرآیندهای حرفه به صورت جداگانه مورد بررسی قرار می گیرند، امکان بروز افزونگی بوجود می آید. زیرا ممکن است یک وظیفه مندی خاص در یک فرآیند حرفه دیده شده باشد و در سرویسی پوشش داده شده باشد، اما در فرآیند دیگری نیز آن را در سرویس دیگری پوشش دادهباشیم. این مسئله سبب می شود امکان رویهم افتادگی در وظیفه مندی سرویس ها بوجود آید و افزونگی بروزکند.

- وظیفه های حرفه

این روش تا حد زیادی شبیه روش تجزیه فرآیندهای حرفه است اما تفاوت این دو روش در این است که در این روش فرآیندهای حرفه به صورت جداگانه بررسی نمی شوند بلکه ابتدا مدلی از فرآیندهای حرفه بوجود می آید که در آن تمامی فرآیندهای حرفه دیده می شوند سپس مدل فرآیندهای حرفه به منظور یافتن سرویس ها مورد تجزیه و تحلیل قرار می گیرد.

این روش علاوه بر مزیت روش تجزیه فرآیندهای حرفه، از بروز افزونگی و رویهم افتادگی سرویس ها جلوگیری می کند و با استفاده از مدل فرآیندهای حرفه تا حد زیادی مدیریت وظیفه مندی سرویس ها را مدیریت می نماید. البته باید توجه داشت مزیت روش تجزیه فرآیندهای حرفه بر این روش سادگی و سربار کمتر است زیرا در این روش برای شناسایی سرویس ها یک مرحله اضافی نسبت به روش پیش یعنی تولید مدل فرآیندهای حرفه را باید انجام داد که خود موجب بروز افزونگی در روال شناسایی سرویس ها می گردد.

نقطه ضعف این روش نیز مانند روش قبل کاهش استفاده مجدد سرویس هاست که به علت تمرکز بر روی فرآیندهای یک حرفه خاص بوجود می آید.

- اشیاء موجودیتی حرفه

در این روش با استفاده از روش هایی مانند استفاده از ماتریس CRUD و اشیاء موجودیتی حرفه(business entity)، می توان سرویس های مورد نیاز را شناسایی نمود در این روش با استفاده از این ماتریس ها، اشیاء موجودیتی حرفه و فرآیندهای مقدماتی حرفه (elementary business elements) می توان سرویس های کاندید را بر اساس روابط مشخص میان ای اشیاء و فرآیندها تشخیص داد. در [۳] از این روش به منظور تشخیص سرویس ها استفاده شده است.

ازمزایای این روش این است که با توجه به موجودیت های دامنه حرفه، امکان بروز تغییرات در سرویس ها در آینده کاهش می یابد زیرا معمولا در دامنه حرفه های مختلف موجودیت های جدیدی

بوجود نمی آیند وهمین امر پایداری مجموعه سرویس های شناسایی شده با این روش را افزایش می دهد.

البته در این روش در صورتی که یکی از اشیاء موجودیتی حرفه با سایر موجودیت ها ارتباطی نداشته باشد، درنظر گرفته نمی شود و لذا در این روش به یک مرحله بازبینی جهت یافتن اینگونه اشیاء نیاز است. همین امر به عنوان عیب اصلی این روش ها ذکز شده است.

- مالکیت و مسئولیت

در این روش برای جلوگیری از بروز افزونگی و رویهم افتادگی، پس از اینکه با استفاده از یکی از روش های دیگر برخی سرویس هایی کاندید شناسایی شدند، حیطه هر یک از سرویس ها به دقت تعیین می گردد.

همانطور که از تعریف این روش مشخص است حسن آن در تعیین دقیق مرز سرویس ها و وظیفه مندی آنهاست اما عیب این روش وابسته بودن به سایر روش هاست. زیرا همانطور که پیش از این گفته شد، در این روش سرویس های کاندیدی که توسط سایر روش ها شناسایی شده بودند را به منظور تعیین دقیق حیطه وظیفه مندی پالایش می کنیم و حال اگر در روش اولیه نقص و کاستی وجود داشته باشد، این خطا در خروجی این روش نیز بروز خواهد کرد.

- هدف گرایی

در این روش برخلاف روش تجزیه فرآیندهای حرفه، به فرآیندهای حرفه توجه نمی کنیم بلکه به اهداف مورد نظر این فرآیندها ویا به عبارت دیگر به اهداف مورد نظر حرفه توجه می نماییم. روش کار در این رده از تکنیک ها در این است که ابتدا اهداف حرفه شناسایی شده و سپس این اهداف به منظور رسیدن به دانه بندی مناسب به زیر اهداف تجزیه می گردند. پس از رسیدن به دانه بندی مناسب، هر کدام از زیر اهداف به یک سرویس جهت برآورده کردن آن هدف متناظر می گردند.

مزیت این روش تطبیق کامل با سیاست های حرفه است. این بدان معنی است که چون در شناسایی سرویس ها اهداف و سیاست های حرفه را مد نظر قرار داده ایم، خروجی های این روش بر سیاست های حرفه کاملا تطبیق خواهند داشت و سیاست های حرفه را به طور کامل پوشش می دهند.

نقطه ضعف این روش نیز مانند روش تجزیه فرآیندهای حرفه و سایر روش های متمرکز بر ماهیت حرفه، کاهش قابلیت استفاده مجدد است و این امر به علت تفاوت در ماهیت حرفه های گوناگون با یکدیگر میباشد. همچنین به علت تمرکز بر اهداف و عدم توجه به وظیفه مندی های مورد نیاز امکان بروز افزونگی ورویهم افتادگی میان سرویس ها رد این روش وجود دارد.

- مبتنی بر مؤلفه

در این روش از روش های شناسایی مؤلفه ها استفاده می شود. با توجه به بلوغ نسبی در روش های شناسایی مؤلفه ها در این روش ابتدا با یکی از این روش ها مؤلفه های موجود در دامنه حرفه شناسایی می شوند و سپس این مؤلفه ها برای تبدیل به سرویس ها مورد پالایش قرار می گیرند.

اصلی ترین مزیت این روش سادگی آن است. زیرا با یکی از روش های شناسایی مؤلفه ها می توان مؤلفه ها را به آسانی شناسایی نمود و سپس این مؤلفه های شناسایی شده را به سرویس ها تبدیل نمود. همچنین در این روش به علت اینکه دامنه مسئله دو بار مورد پالایش قرار می گیرد، برخی سرویس های ناشناخته را نیز می توان کشف نمود و این امر نیز به عنوان یکی دیگر از مزیت های این روش مطرح است.

نقطه ضعف اصلی این روش نیز از تفاوت در ماهیت سرویس ها و مؤلفه ها ناشی می شود. زیرا فاکتورهای مهم در ارزیابی سرویس ها با فاکتورهای ارزیابی مؤلفه ها متفاوت است و در صورت استفاده از این روش ها، سرویس هایی شناسایی می شوند که مطابق با فاکتورهای مناسب برای مؤلفه ها می باشند . فاکتورهای لازم برای سرویس ها را ندارند.

همچنین این روش به منظور بررسی دوباره دامنه مسئله، سربار روال شناسایی سرویس ها را افزایش می دهد.

این رده از روش ها بر استفاده از دارایی های موجود تاکید دارند به همین علت به این روش ها، روش های پایین به بالا نیز می گویند. در این روش ها سیستم موجود و موروثی سازمان جهت کشف سرویس ها مورد بررسی، ترزیابی و پالایش قرار می گیرند و در صورت تطبیق قسمتی از سیستم های موجود با فاکتورهای سرویس، از روی آن سرویس مورد نیاز تولید می گردد.

تصلی ترین مزیت این روش کاهش هزینه و زمان در تولید سرویس هاست زیرا در این روش سرویس ها از درون سیستم های موروثی استخراج شده و نیاز به هزینه و زمان زیاد برای تولید ندارند.

نقطه ضعف اصلی این روش نیز در استفاده از سیستم های موروثی ناشی می شود. زیرا در صورتی که اینسیستم های موروثی طراحی ضعیفی داشته باشند و یا در منطق آنها اشتباهی وجود داشته باشد این مورد در سرویس های آینده نیز تکثیر شده و اثر گذار خواهد بود.

در این روش برنامه ها به دو قسمت پیش زمینه و پس زمینه تقسیم می گردند. پس از تولید قسمت پیش زمینه به منظور کشف سرویس ها، پرسوجوهایی (query) طراحی شده و با استفاده از آنها سرویس های مورد تیاز کشف می گردند.

مزیت این روش در سرعت کشف سرویس ها و قابلیت استفاده مجدد از آنهاست. زیرا با استفاده از این پرسوجوها به سرعت می توان سرویس های مورد نیاز را شناسایی نمود و با توجه به وابسته نبودن این سرویس ها به دامنه حرفه ای خاص، امکان استفاده مجدد از آنها افزایش می یابد.

اما نقطه ضعف این روش نیز از طراحی ضعیف برنامه های پیش زمینه و پرس و جوها ناشی می گردد زیرا در صورتی که این برنامه ها از کیفیت پایینی برخوردار باشند، این امر بر کیفیت سرویسهای شناسایی شده نیز اثر خواهد گذاشت.

در این روش، با توجه به اینکه در صورت اجرای سرویس ها بر روی یک سکوی سخت افزاری کارآیی انها افزایش می بابد، یک سری عملیات به منظور اجرای سرویس ها بر سکوهای سخت افزاری مشخص انجام می گیرد.

این امر باعث کاهش هزینه در توسعه سرویس ها می گردد که همین مسئله به عنوان اصلی ترین مزیت این روش ها مطرح می باشد.

اما نقطه ضعف اصلی این روش ها که با اصلی ترین ویژگی سرویس ها در تعارض است این است که در صورت توجه به مسائل مربوط به سکوهای سخت افزاری سرویس ها ، موجب کاهش سطح تجرید سرویس ها شده و یکی از اصلی تریم اصول در طراحی سرویس ها را نقض می نماییم.
در تمامی روش های قبل تمرکز بر نیازهای وظیفه مندی سرویس ها می باشد. در این روش برخلاف تمامیاین روش ها، تمرکز بر ویژگی های غیر وظیفه مندی و کیفی سرویس ها قرار می گیرد. در این روش پس از شناسایی سرویس ها توسط یکی از روش های قبل، طراحی سرویس را به منظور دستیابی به ویژگی های کیفی مورد انتظار تغییر می دهیم.
همانطور که واضح است نقطه قوت این روش توجه و تمرکز بر ویژگی های کیفی سرویس هاست که می تواند تاثیر زیادی بر نحوه استفاده از سرویس ها داشته باشد.
نقطه ضعف اصلی این روش نیز در وابسته بودن آن است. این بدان معناست که ابتدا باید مجموعه سرویس های کاندید توسط یکی از روش های فوق شناسایی شوند و سپس از این روش استفاده نماییم. همچنین در صورت وجود نقص در خروجی روش های فوق این خطا در خروجی این روش نیز تکثیر پیدا می کند.

پس از آشنایی با رده بندی کلی روش ها و قوت و ضعف هر یک از آنها در ادامه به معرفی برخی روش های ارائه شده در این زمینه می پردازیم. باید توجه داشت روش هایی که در ادامه معرفی می گردند، روش های شاخص در این حسطه هستند که بیشتر از سایر روش ها مورد استقبال قرار گرفته اند و مجموعه زیر حاوی تمامی روش های ارائه شده در این زمینه نمی باشد.

2.3 روش اول Zimmermann

در این روش به این نکته که روش های گذشته مانند روش تحلیل و طراحی شیء گرا (OOAD)، معماری سازمانی (EA) و مدلسازی فرآیندهای حرفه (BPM) الگوهای معماری سرویس گرا را

پوشش نمی دهند اشاره شده است و همچنین با توجه به این موارد نتیجه گیری شده است که به روشی جدید جهت تحلیل و طراحی در زمینه سرویس گرایی نیاز است.

روش های EA، OOAD و BPM به صورت جداگانه خروجی های مفیدی را تولید نمی کنند. در این مقاله روش SOAD (Service Oriented Analysis and Design) یا تحلیل و سراحی سرویس گرا به عنوان روشی ترکیبی از روش های قبل معرفی می شود.

نیازمندی های روش SOAD عبارتند از:

- فرآیندها و یادداشت ها (notations) باید به صورت رسمی یا غیر رسمی تعریف شده باشند.
- با روش OOAD اشیاء و کلاس ها مشخص شده و BPM مدل های مبتنی بر رویداد (event driven) را تولید می کند و SOAD این خروجی ها را در کنار هم قرار می دهد.
- باید توجه داشت که این روش مبتنی بر موارد کاربری (use case driven) نیست و موارد کاربری می توانند در مرحله دوم استفاده شوند.

در این مقاله جهت تشخیص سرویس ها (service identification) از روش های زیر استفاده می شود:

- تحلیل مستقیم و غیر مستقیم حرفه
- تجزیه دامنه حرفه (domain decomposition)
- در نظر گرفتن دامنه بندی سرویس ها
- قواعد نامگذاری سرویس ها

معایب این روش عبارتند از:

- در این مقاله یک روش عملیاتی معرفی نشده است به عبارت دیگر یک روش قابل استفاده و کاربردی در این مقاله ذکر نشده است.
- در این روش به اهداف حرفه توجهی نشده است. به عبارت دیگر در هیچ یک از مراحل کشف سرویس ها این امر که تمامی اهداف حرفه توسط روش ذکر شده پوشش داده می شوند یا خیر، توجه نشده است.

3.3 روش دوم Zimmermann

این Analysis and Design Techniques for Service-oriented Development مقاله با عنوان ارائه شده O. Zimmermann، N. Schlimm، G. Waller و M. Pestel توسط and Integration است. [3]

در این مقاله پس از بررسی روش (Service Oriented Modeling and Architecture SOMA) روشی جهت تبدیل مدل های موارد کاربری و فرآیندهای حرفه پیشنهاد گردیده اما روشی مشخص جهت این امر ارائه نشده است.

همچنین در این روش مسائلی مانند دانه بندی سرویس ها و سایر موارد کیفی سرویس ها در نظر گرفته نشده اند.

4.3 روش Zhang

این مقاله با عنوان Service Identification and packaging in Service-oriented Reengineering توسط Zh. Zhang، R. Liu و H. Yang ارائه شده است. [16]

در این مقاله مراحل تشخیص سرویس ها (service identification) به صورت زیر تعیین شده اند:

- تشخیص سرویس ها در دامنه مسئله . خروجی این فرآیند مدل دامنه (domain model) مسئله خواهد بود.
- تشخیص سرویس ها در سیستم های موروثی. این گام با استفاده از روش های خوشه بندی (clustering) انجام می گردد.
- تطبیق وظیفه مندی های کشف شده در سیستم های موروثی با وظیفه مندی های حرفه (functions) در سرویس های منطقی مشخص شده. با این کار در مورد استفاده از سرویس های موروثی برای پوشش نیازمندی ها و یا ساخت سرویس های جدید تصمیم گیری می شود.

اشکال عمده این روش عدم توجه به فاکتور دانه بندی سرویس ها است. در این روش همانطور که در بالا اشاره شد، ابتدا سرویس های موجود در دامنه مسئله شناخته شده و سپس سرویس های موجود در سیستم های موروثی کشف می شوند و در مورد استفاده یا عدم استفاده از این سرویس های موروثی تصمیم گیری می شود و در هیچ مرحله ای دانه بندی سرویس ها به عنوان فاکتوری مهم و تاثیر گذار بر ویژگی سیستم نهایی در نظر گرفته نمی شود.

5.3 روش Amsden

Modeling SOA: Part ۱. □□□□□□□ □□□□□□□□□□□□□□□ □□□□□□□ توسط J. Amsden ارائه شده است. [15] این مقاله با عنوان

در این مقاله بر موارد زیر تاکید شده است:

- تلاش برای ارائه سرویس هایی به منظور پوشش دهی اهداف حرفه
- توجه روی مفهوم جداسازی دغدغه ها (separation of concerns)
- توجه روی مفهوم اتصال سست (loose coupling)

در این مقاله روش کار به این ترتیب است که ابتدا با تمرکز روی اهداف حرفه شروع به کار کرده سپس فرآیندهای حرفه که به این اهداف پاسخ می دهند را شناسایی می کند. پس از شناسایی نیازهای حرفه، سرویس ها را به منظور پوشش آنها تشخیص می دهد.

در این مقاله از روش تجزیه دامنه (domain decomposition) برای شناسایی سرویس های کاندید از روی مدل حرفه (business model) اشتفاده می شود.

اشکال عمده روش های ارائه شده در این مقاله نیز در نظر نگرفتن فاکتور دانه بندی سرویس هاست. این روش توجه خوبی بر پوشش دهی اهداف حرفه و فرآیندهای آن دارد اما فاکتور دانه بندی سرویس ها را به عنوان فاکتوری تاثیر گذار بر ویژگی سیستم نهایی در نظر نمی گیرد.

6.3 روش SOMA

این مقاله Service-Oriented Modeling and Architecture (SOMA) توسط ش Ali با عنوان Arsanjani ارائه شده است. [۱۱]

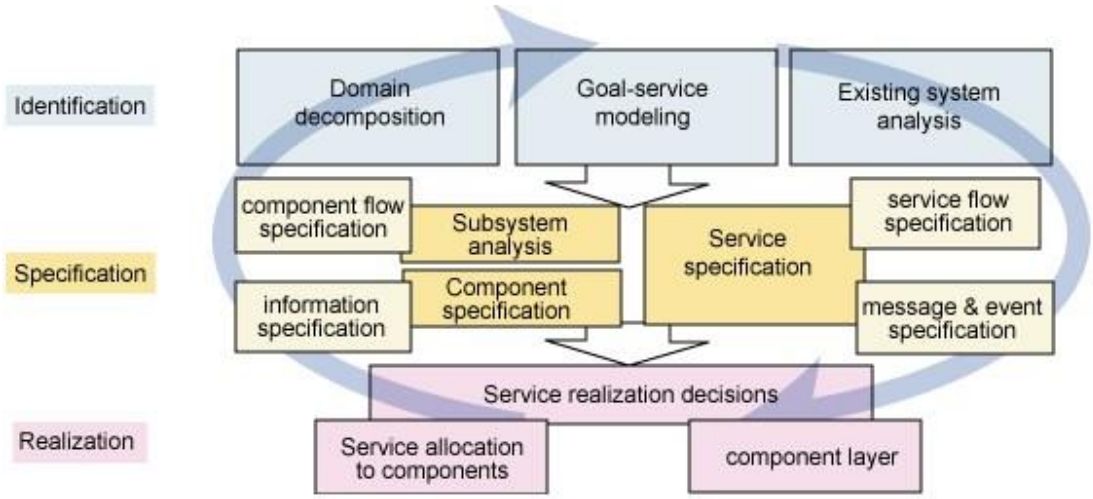
در این مقاله مفاهیم زیر آورده شده اند:

- روش های OOAD معمول مفاهیمی چون سرویس، جریان و مؤلفه ها را پوشش نمی دهند.
- روش SOMA شامل فعالیت های مدلسازی، تحلیل و طراحی و فعالیت هایی به منظور تعریف مؤلفه های موجود در تمامی لایه های SOA می باشد.
- در این مقاله برای تشخیص سرویس ها ابتدا از روش بالا به پایین به منظور یافتن سرویس هایکاندید و سپس از روش مدلسازی هدف سرویس به منظور بررسی پوشش دهی اهداف

توسطسرویس های شناخته شده اشتفاده می شود. در آخر نیز از روش پایین به بالا استفاده می شود.

- در روش بالا به پایین از روش domain decomposition استفاده می شود. در این روش دامنه حرفه به نواحی مختلفی از نظر وظیفه مندی تجزیه می شود. همچنین جریان ها یا فرآیندها به زیر فرآیندها و موارد کاربری سطح بالا تجزیه می شوند. این موارد کاربری اغلب کاندیدهای خوبی برای سرویس های حرفه هستند.
- در روش پایین به بالابا استفاده از روش تحلیل دارایی های موجود، سیستم های موروثی به منظور یافتن مؤلفه هایی جهت پوشش وظیفه مندی های شناخته شده تحلیل می شوند.
- در روش میانی از روش مدلسازی هدف–سرویس به منظور کشف سرویس هایی که تا به حال شناخته نشده بودند استفاده می شود. در این روش سرویس ها را به اهداف، زیر اهداف و KPIها متناظر می کنیم.

در شکل زیر فعالیت های روش SOMA در تمامی مراحل مدلسازی سرویس ها آمده است:



شکل شماره ۳ -۱ - فعالیت های SOMA [۱۱]

در زیر فازها و فعالیت های روش SOMA را به اختصار شرح می دهیم.

فاز شناسایی سرویسها *Identification*

فاز شناسایی سرویسها یکی از بحرانی ترین فازها در موفقیت پروژه معماری سرویس گرا است زیرا در این فاز نیازمندی های حرفه شناخته شده و پوشش داده می شوند. هدف این فاز تولید مجموعه ای از سرویس های کاندید برای پروژه سرویس گرا و مشخص کردن عملیات آنها است. برای شناسایی سرویسهای کاندید تحلیل گر در مورد استفاده از راهبردها و روش های مختلف تصمیم گیری می کند.

نقش های درگیر در این فار عبارتند از : معمار نرم افزار (نقش اصلی) و تحلیل گر حرفه (نقش کمکی). در خلال این فاز مدل کاری سرویس ها تولید می شود و در پایان فاز، این مدل به معمار نرم افزار برای انجام عملیات مربوط به فاز توصیف سرویس ها (Service Specification) تحویل داده می شود.

اصلی ترین ورودی های این فاز سرویس های موجودی که سازمان به آنها (از هر طریقی مانند تملک، خریداری و ...) دسترسی دارد می باشند. این سرویس ها معمولا در انباره سرویس های سازمان موجود می باشند. البته باید توجه داشت که در سازمان هایی که اولین تجربه معماری سرویس گرای خود را انجام می دهند ،طبیعتا این ورودی وجود ندارد، و سرویس های تماما باید تولید شده و پس از انجام فازهای مختلف در انباره برای استفاده های در پروژه های سرویس گرایی آینده قرار بگیرند. همانطور که در شکل شماره ۱۰ مشاهده می کنید ،فعالیت های درون این فاز عبارتند از: تجزیه دامنه حرفه (Domain Decomposition)، مدلسازی سرویس-هدف (Goal-Service Modeling) و تحلیل دارایی های موجود (Existing Asset Analysis). به طور کل می توان گفت این فاز دارای ترکیبی از تحلیل های بالا به پایین (تجزیه دامنه حرفه)، پایین به بالا(تحلیل دارایی های موجود) و تحلیل میانی(مدلسازی سرویس- هدف) است.[۱۱].در روش بالا به پایین با بررسی

نیازمندی های حرفه، سرویس ها استخراج می شوند، سپس با استفاده از تحلیل پایین به بالا سرویس های قابل استفاده در سیستم های موروثی شناسایی می گردند.

تجزیه دامنه حرفه

در این مرحله دامنه حرفه به فرآیندهای حرفه، زیر فرآیندها و موارد کاربری تجزیه می گردد. از دید حرفهدامنه مجموعه ای حیطه های وظیفه مندی است. در این مرحله این حیطه ها را بر اساس دو پرسش کلیدی زیربر حسب نیازمندی ها تجزیه می کنیم :

- آیا این وظیفه مندی در حیطه وظایف سازمان قرار می گیرد؟ آیا این وظیفه مندی توسط یک واحد سازمان انجام می شود یا توسط بیش از یک واحد؟
- اگر این نیازمندی فراتر از حیطه سازمان است ، چه سازمان های دیگری در انجام آن با سازمان همکاری دارند؟

با استفاده از دو پرسش فوق دامنه مسئله را به فرآیندهای سازمان، زیر فرآیندها و موارد کاربری تجزیه می کنیم. این موارد کاربری (سطح سوم تجزیه) معمولا کاندیدهای خوبی برای سرویس ها می باشند.

تحلیل دارایی های موجود

پس از شناخت سرویس های کاندید مورد نیاز برای پاسخگویی به نیازهای سازمان (تحلیل بالا به پایین)، سیستم های موروثی سازمان به منظور شناسایی سرویس هایی که می توان از آنها در پروژه استفاده کرد تحلیل می شوند (تحلیل پایین به بالا). این کار باعث کاهش هزینه و زمان پروژه می گردد. به این فعالیت تحلیل دارایی های موجود می گویند. البته باید توجه داشت که گاهی نیاز است تا دارایی های موجود (سیستم های موروثی) به منظور مطابقت با نیازهای جدید، واحدبندی مجدد شوند.

مدلسازی سرویس- هدف

همانطور که گفته شد، موارد کاربری که در تجزیه دامنه حرفه شناسایی شدند کاندید های خوبی برای سرویس ها می باشند. در این مرحله مدلی به نام مدل سرویس-هدف به منظور بررسی کامل بودن مجموعه سرویس های کاندید تولید می شود.

از طریق برگزاری جلسه با مالکان حرفه و پرسش از آنها در مورد اهداف موجود در حیطه پروژه، می توان زیر هدف های پروژه که پیش نیاز رسیدن به اهداف کلان پروژه و سازمان هستند را شناسایی کرد. هر زیر هدف را نیز می توان به زیر هدف های کوچکتر تجزیه نمود. این عمل تا زمانی که سرویس هایی که برآورده کننده زیراهداف هستند به روشنی مشخص شوند، ادامه پیدا می کند. به درخت حاصل از این تجزیه، درخت هدف -سرویس گفته می شود.

با در دست داشتن مدل هدف-سرویس و شناسایی سرویس های مربوط به هر هدف می توان کامل بودن مجموعه سرویس های کاندید را بررسی کرد.

پس از انجام این مرحله مجموعه ای از سرویس های کاندید برای پیاده سازی به منظور پوشش نیازهای سازمان و مجموعه ای از سرویس های موجود از طریق سیستم های موروثی که برای پاسخگویی به برخی از نیازمندی های سازمان قابل استفاده هستند تولید شده است. همچنین با استفاده از درخت هدف-سرویس کامل بودن مجموعه سرویس های کاندید مورد بررسی قرار گرفته است.

فاز توصیف سرویس ها *Specification*

پس از شناسایی مجموعه سرویس های کاندید فاز توصیف سرویس ها آغاز می گردد. این فاز توسط معمار نرم افزار به همراه طراحان (به صروت اختیاری) انجام می شود. هدف این فاز توصیف کامل عناصر طراحی معماری سرویس گرا است.

فاز توصیف سرویس ها را می توان وظیفه مربوط به معماری در فرآیند معماری سرویس گرا دانست. این دید اهمیت این فاز را به خوبی نمایان می سازد.

برای روشن تر شدن تمایز میان فاز توصیف سرویس ها و شناسایی سرویس ها می توان فاز شناسایی سرویس ها را (Service Identification) را به عنوان فعالیت های مربوط به تحلیل مدل سرویس و فاز توصیف سرویس ها (Service Specification) را به عنوان فعالیت های مربوط به طراحی مدل سرویس دانست.

در خلال فاز توصیف سرویس ها مدل سرویس به روشنی مشخص می شود و در پایان این فاز، این مدل به طراحان برای انجام فاز عینیت بخشیدن به سرویس ها و توسعه دهندگان برای پیاده سازی سرویس ها داده می شود.

همانطور که در شکل شماره ۱۰ مشاهده می کنید ، وظایف اصلی درون این فاز عبارتند از: تحلیل زیرسیستم ها (Subsystem Analysis) ، توصیف مؤلفه ها (Component Specification) و تخصیص سرویس ها (Service Allocation). در این فاز ویژگی ها و قابلیت های سرویس ها تشریح می گردند و به سؤالهاییاز جمله : سرویس چه خدمتی ارائه می دهد؟ منطق داخلی سرویس ها چیست؟ نیازمندی های غیر وظیفهمندی هر سرویس چیست؟ و... جواب داده می شود.

تحلیل زیرسیستم ها

ورودی این مرحله زیرسیستم هایی (سرویس ها) که از مرحله تجزیه دامنه بدست آمده اند می باشند و در این مرحله وابستگی بین این زیر سیستم ها و جریان های بین آنها مشخص می گردد.

در تحلیل زیرسیستم ها مدل های اشیاء به منظور مشخص کردن نحوه انجام کار در داخل زیر سیستم و همچنین نمایش زیر سیستم های درون زیر سیستم در حال بررسی (سلسله مراتب زیرسیستم ها) تولید می شوند.

ساختار طراحی زیر سیستم ها برای تولید ساختار پیاده سازی مؤلفه ها – که خود اساسی برای پیاده سازی سرویس ها هستند – استفاده می شود.

در این مرحله خصوصیات سرویس ها از جمله : وابستگی بین آنها، نحوه ترکیب آنها، پیام های بین آنها ، فاکتور های کیفیت آنها و... تعیین می شوند.برای این کار ٦ مرحله زیر انجام می گردند.

١. تست *Litmus* : این تست به این سؤال که "از میان سرویس های کاندید کدامیک برای پیاده سازی مناسب است؟" پاسخ می دهد. باید توجه داشت تمامی سرویس هایی که در مرحله تجزیه دامنه مشخص شدند برای پیاده سازی مناسب نیستند. برای پیدا کردن زیر مجموعه ای از این سرویس ها در روش تست Litmus براساس موارد مختلف از جمله تطابق سرویس با نیازهای حرفه، تکراری نبودن سرویس و ... تصمیم گیری می شود. منظور از تکراری نبودن سرویس این است که وظایفی که سرویس در دست بررسی انجام می دهد توسط سرویس های دیگر پوشش داده نشوند.

٢. .مشخص کردن وابستگی میان سرویس ها :بازبینی جزئیات سرویس ها باعث مشخص شدن وابستگی سرویس به سرویس های دیگر و نحوه تعامل میان سرویس ها می شود.

٣. مشخص کردن ترکیب سرویسها و جریان میان آنها :با بررسی حیطه های کاری و فرآیندهای حرفه نحوهتولید سرویس ها از ترکیب سرویس های دیگر و جریان کاری میان سرویس ها برای انجام فرآیندهای حرفه را مشخص می کند.

٤. مشخص کردن نیازهای غیروظیفه مندی :با استفاده از نیازهای غیر وظیفه مندی ، ویژگی های کیفی سرویس ها مشخص می گردد.

٥. مشخص کردن پیام های سرویس :در این قسمت قالب پیام های ورودی و خروجی سرویس و محتوای آنها مشخص می شود.

٦. مستندسازی تصمیمات مدیریتی :برخی مواقع ترکیب سرویس ها منجر به برخی تصمیمات مدیریتی در حالت های مختلف کاری سرویس می شود. این تصمیمات مدیریتی در این مرحله مستندسازی مس شوند.

تخصیص سرویسها

پس از مشخص کردن سرویس ها و جزئیات آنها در این مرحله درباره تخصیص سرویس ها به مؤلفه ها تصمیم گیری می کنیم. اغلب این تخصیص یک-به-یک است.
همچنین در این مرحله نحوه تخصیص سرویس ها و مؤلفه به لایه های معماری سرویس گرا مشخص می شود. برای تصمیم گیری در مورد چگونگی تخصیص سرویس ها و مؤلفه ها به لایه های معماری علاوه بر توجه به معماری نرم افزار باید به معماری محیط فنی و عملیاتی که سیستم در آن اجرا می شود توجه نمود.

توصیف مؤلفه ها

در مرحله بعد از فاز توصیف سرویس ها ، جزئیات مؤلفه هایی که سرویس ها را پیاده سازی می کنند مشخص می شود. این توصیف دربرگیرنده جزئیاتی در مورد داده های مؤلفه، قوانین، سرویس ها و نحوه پیکربندی مؤلفه می باشد. همچنین در این مرحله پیام های مؤلفه ها و خصوصیات رویدادهای آنها مشخص می گردد.
پس از مشخص کردن خصوصیات سرویس ها و نحوه اجرای سرویس ها توسط مؤلفه ها و همچنین تعریف جزئیات مؤلفه ها، باید در مورد نحوه تولید و در اختیار گرفتن سرویس ها تصمیم گیری شود.

فاز عینیت بخشیدن به سرویسها *Realization*

در این فاز در مورد مباحث مربوط به "امکان سنجی فنی" پیاده سازی یا استخراج سرویس ها از سیستم های موروثی تصمیم گیری می کند.[۱۱] گزینه های دیگری که در این زمینه وجود دارد عبارتند از تجمیع سرویس ها و تولید سرویس مورد نیاز ، تغییر سرویس های قدیمی برای تولید سرویس مورد نیاز ، سفارش سرویس و

تصمیمات دیگری که در این فاز گرفته می شود – علاوه بر تصمیمات در مورد وظیفه مندی – عبارتند از :

امنیت، مدیریت و نظارت بر سرویس ها.

با توجه به فازهای سه گانه متدولوژی SOMA و چرخه حیات معماری سرویس گرا (شکل شماره ۱) می توان سه فاز فوق را به شکل زیر در مراحل چرخه حیات معماری سرویس گرا گنجاند:
فازهای شناسایی و توصیف سرویس ها در مرحله مدل، فاز عینیت بخشیدن به سرویس ها در مراحل گردآوری، نصب و مدیریت.

۷.۳ روش Portier

این مقاله با عنوان SOA terminology overview, Part 3: Analysis and design توسط B.Portier ارائه شده است. [۱۲]

در این مقاله نکات زیر آورده شده اند:

- تحلیل به این علت برای SOA حیاتی است که تطابق فرآیندهای نرم افزاری و نیازمندی های حرفه در فاز شناخت سرویس ها (service identification) انجام می شود.
- شناخت سرویس ها (service identification) فعالیت مرکزی در تحلیل سرویس گراست. هدف service identification شناخت گروه سرویس های مفهومی و عملیات آنهاست.
- Service identification توسط تحلیل گر حرفه که به کعکار نرم افزار کمک می کند انجام می گردد.

- تحلیل بالا به پایین : در این روش فرآیندهای حرفه و موارد کاربری تجزیه می شوند. مزیت این روش این است که از تطابق میان سرویس ها با نیازهای حرفه اطمینان داریم.
- تحلیل پایین به بالا : این روش روی تحلیل دارایی های فناوری اطلاعات موجود تمرکز دارد. مزیت این امر استفاده مجدد است.
- تحلیل میانی (meet-in-the-middle) : این روش روی تطبیق دادن نیازمندی ها (سرویس های شناخته شده توسط روش بالا به پایین) و آنچه توسط فناوری اطلاعات موجود است (خروجی های مرحله پایین به بالا) تمرکز دارد.

همانگونه که از مطالب فوق مشخص است این روش نیز به فاکتور دانه بندی سرویس ها توجهی ندارد.

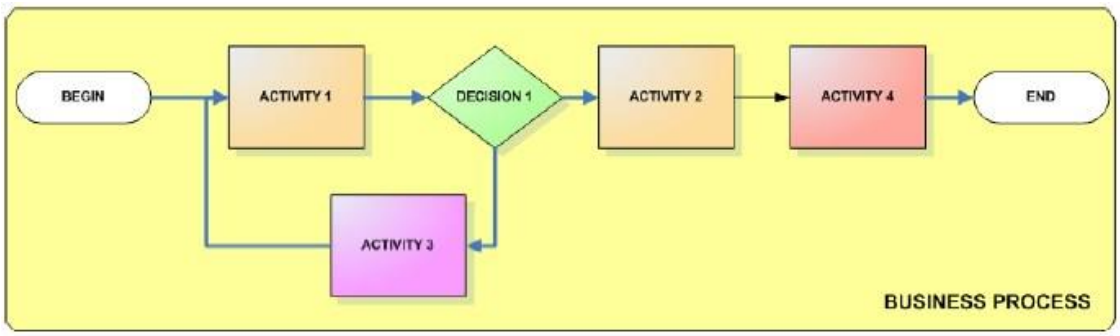
8.3 روش Inganti

این Service Identification: BPM and SOA Handshake توسط S. Inaganti و G.K. Behara مقاله با عنوان ارائه شده است.

[١٤] در این مقاله موارد زیر ذکر گردیده اند:

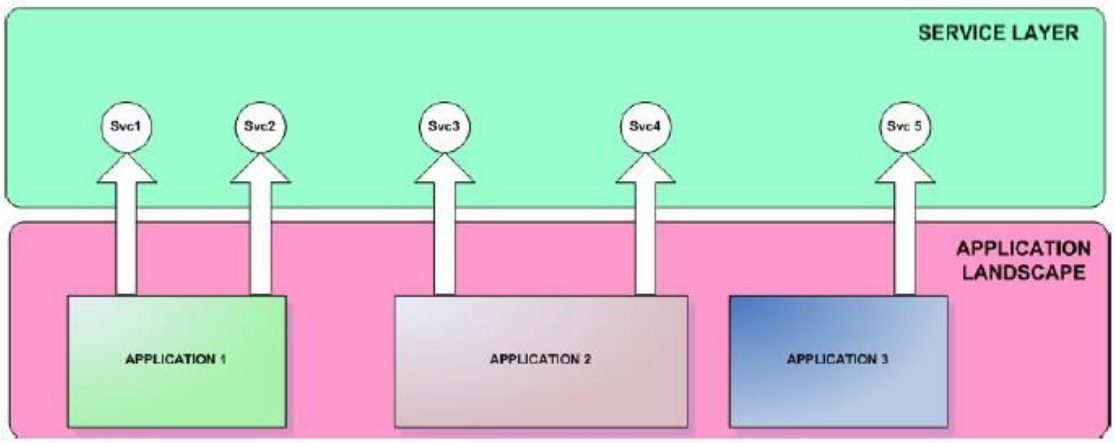
- دلایل اهمیت شناسایی سرویس ها (service identification) : کم بود مستندات فرآیند حرفه، تخصص در تحلیل، استفاده مجدد، کم بود درگیری اجرایی به همراه بروز اختلال در تیم های پروژه
- اشتباهاتی که در شناسایی سرویس ها (service identification) رخ می دهند ممکن است در طراحی و پیاده سازی ادامه پیدا کنند.

- روش های کلی در گام شناسایی سرویس ها عبارتند از: ۱. روش های بالا به پایین ۲. روش های پایین به بالا
- در روش های بالا به پایین ممکن است از روش های مبتنی بر فرآیندهای حرفه و یا روش های مبتنی بر موارد کاربری استفاده شود. تصمیم گیری در مورد هر یک از روش ها بستگی به گستردگی دامنه مسئله دارد.
- در این مقاله عنوان شده است که بهتر است دو تیم داشته باشیم. تیم سرویس ها و تیم برنامه های کاربردی. تیم سرویس ها دنبال سرویس ها و نیازمندی برای هر سرویس می گردد و تیم برنامه های کاربردی همزمان نیازمندی هر زیر سیستم را شناخته و به دنبال سرویس برای مطابقت با زیرسیستم است.
- در این روش ابتدا فرآیندهای حرفه به صورت مجموعه ای از فعالیت ها (^{activity}) که با هم در ارتباط هستند مشخص می شوند. سپس وظیفه مندی های برنامه های کاربردی مشخص شده و هر کدام به صورت یک سرویس معرفی می گردند. سپس عمل تطبیق بین این سرویس ها و فعالیت ها انجام می شود. اگر فعالیتی وجود داشت که سرویسی آن را پوشش نمی داد، فعالیت به زیر فعالیت ها شکسته شده و برای زیر فعالیت ها سرویس جدید بوجود می آوریم. در شکل های زیر این موارد نشان داده شده اند:



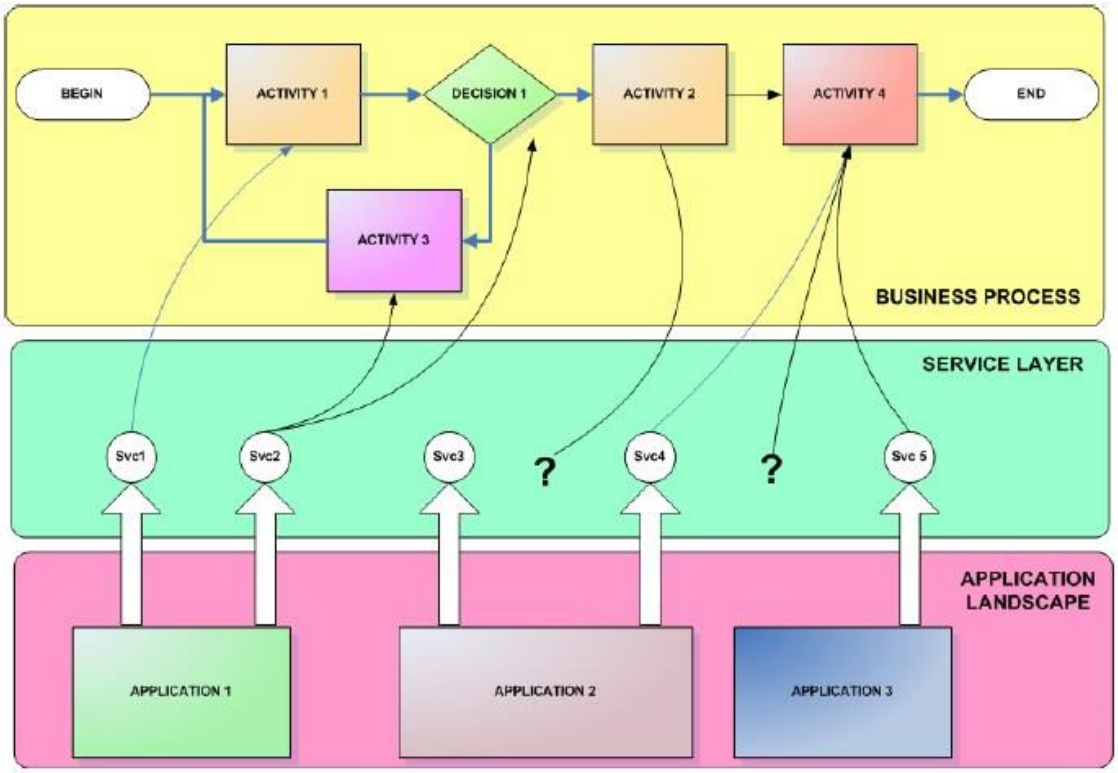
شکل شماره ۳ -۲ - بالا به پایین [۱٤]

در شکل شماره ۱۱، توسط روش بالا به پایین یک فرآیند حرفه به تعدادی فعالیت (activity) تجزیه شده است.



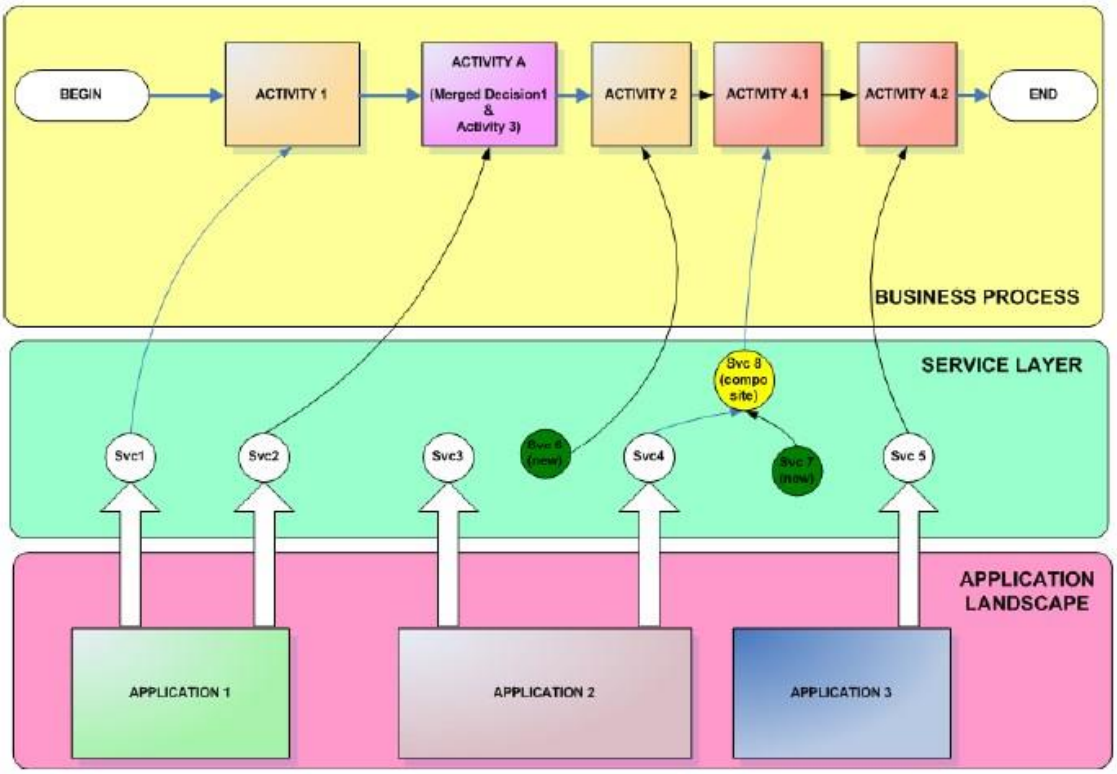
شکل شماره ۳ -۳ - روش پایین به بالا [۱٤]

در شکل شماره ۱۲، از روی وظیفه مندی های برنامه های کاربردی، سرویس ها تعیین می گردند. به این ترتیب که هر یک از واسط های برنامه های کاربردی به یک سرویس متناظر می گردند.



شکل شماره ۳ -٤ - تناظر فعالیت ها و سرویس ها [١٤]

در شکل شماره ۱۳، سرویس های شناخته شده به فعالیت ها متناظر می گردند. مشاهده می کنید فعالیت activity 2 توسط هیچ سرویسی پوشش داده نمی شود. برای رفع این مشکل مطابق شکل شماره ١٤ فعالیت ها تجزیه می گردند.



شکل شماره ۳ –۵ – تجزیه فعالیت ها [۱٤]

در شکل فوق برای حل مشکل، سرویس جدیدی به منظور پوشش فعالیت activity2 تولید شده است.

همچنین سرویس جدیدی نیز از ترکیب سرویس ٤ و یک سرویس جدید به منظور پوشش فعالیت Activity4 تولید گردیده است.

اشکال روش این مقاله این است که در آن فرآیندها به صورت جداگانه در نظر گرفته می شوند، لذا امکان بروز افزونگی و رویهم افتادگی سرویس ها در آن وجود دارد.

نتیجه گیری

در این فصل با روش های ارائه شده در زمینه تشخیص سرویس ها آشنا شدیم. همانطور که در این فصل بیان شد در کل ۱۰ رده از تکنیک ها به منظور کشف سرویس ها وجود دارد . این رده ها تعریف شده و مزایا و معایب هر یک بیان گردید.

همچنین در ادامه با ۷ روش مختلف که در هفت مقاله مطرح ارائه شده بودند آشنا شدیم و مزایا و معایب این روش ها را مشاهده نمودیم.

در فصول آینده با شناخت نسبت به حیطه مسئله و ویژگی های سرویس های مورد انتظار و همچنین ویژگی های روش های ارائه شده در این زمینهقصد داریم روشی جدید ارائه نماییم که نقاط ضعف روش های موجود را پوشش داده و از نقاط قوت این روش ها نیز بهره ببرد.

فصل چهارم ویژگی های راه حل مورد انتظار

مقدمه

در فصول قبل با مفهوم گام تشخیص سرویس ها و جایگاه آن در چرخه حیات معماری سرویس گرا و فاز مدلسازی سرویس گرا آشنا شدیم. همچنین روش های پرکاربرد در حیطه تشخیص سرویس ها معرفی شده و مورد تجزیه و تحلیل قرار گرفتند.

در این فصل قصد داریم تا با شناخت فاکتورهای لازم به منظور ارزیابی کیفیت سرویس های نرم افزاری و همچنین آگاهی از نقاط ضعف روش های موجود در زمینه تشخیص سرویس های نرم افزاری، به بیان چاچوبی جهت ارائه روشی به منظور تشخیص سرویس ها که پوشش دهنده نقاط ضعف دیگر روش ها بوده و در عین حال از نقاط قوت آنها نیز سود ببرد بپردازیم. به این منظور در این فصل ابتدا مروری کوتاه بر ویژگی های مورد انتظار سرویس های نرم افزاری کرده، سپس نقاط ضعف عمده در سایر روش ها عنوان می گردد.

در ادامه نیز با توجه به این دو فاکتور اصول روش مورد انتظار بیان خواهد شد.

٤.١ نتایج مورد انتظار از سرویس ها

در فصل دوم و در بخش ۲.۲ معیارهای کیفی سرویس ها بیان شدند، همچنین در بخش ۲.۳ فاکتور دانه بندی سرویس ها به عنوان عاملی تاثیر گذار بر برخی جنبه های سیستم نهایی مطرح شد. در این بخش قصد داریم تا با مروری کوتاه بر مطالب بخش های فوق ویژگی های مورد انتظار از خروجی های روش تشخیص سرویس ها که همان سرویس های نرم افزاری هستند را متذکر شویم. در منبع [۱] توماس ارل، ویژگی های زیر را به عنوان فاکتورهای کیفی سرویس ها بیان کرده است:

- **سرویس ها قابلیت استفاده مجدد دارند** به این معناست که سرویس ها به گونه ای طراحی می گردند که پتانسیل استفاده مجدد را دارا باشند. البته ممکن است سرویسی هیچگاه مورد استفاده مجدد قرار نگیرد اما دارای این خصوصیت باشد.
- **سرویس ها یک قرارداد رسمی را به اشتراک می گذارند** به این معناست که سرویس ها به منظور برقراری ارتباط با سایر سرویس ها تنها نیاز به اشتراک گذاری قرارداد رسمی خود دارند که در این قرارداد سرویس به معرفی ویژگی های اصلی خود، وظایفی که ارائه می دهد و ویژگی هایی که طرف استفاده کننده برای استفاده از سرویس باید داشته باشد می پردازد.
- **سرویس ها اتصال سست با یکدیگر دارند** به این معناست که سرویس ها باید به گونه ای طراحی شوند که ارتباطشان با سایر سرویس ها تا حد امکان کم باشد. در صورتی که سرویس ها این ویژگی را داشته باشند، امکان توسعه، تست و اجرای مستقل سرویس ها بوجود می آید همچنین در صورت وجود این ویژگی در سرویس های مختلف در سیستم نهایی، قابلیت نگهداشت سیستم نهایی به صورت چشمگیری افزایش پیدا می کند زیرا در صورت نیاز به تغییر در ماهیت یک سرویس (نیاز به تغییر در واسط های سرویس) تعداد سرویس کمی که با آن در ارتباط هستند نیاز به تغییر دارند.

- **سرویس ها منطق درونی خود را پنهان می کنند** بدان معناست که تنها قسمتی از سرویس که از طریق جهان خارج قابل مشاهده است، قرارداد رسمی سرویس می باشد. این امر سبب می شود منطق درونی سرویس و نحوه عینیت بخشی به تعهدات قرارداد رسمی سرویس از دید جهان خارج و متقاضی سرویس پنهان بماند. از جمله مزیت های این امر افزایش امنیت سرویس می باشد. زیرا در صورتی که راههای ارتباطی با سرویس تنها به واسط های از پیش تعیین شده آن محدود گردد امکان نفوذ به درون سرویس و ایجاد اختلال در کار آن کاهش یافته و تشخیص موارد نفوذ آسان می گردد.علاوه بر این این ویژگی تا حد زیادی سبب جلوگیری از پیدایش پیچیدگی در توسعه سیستممی شود. زیرا اگر سرویس ها منطق درونی خود را پوشش نمی دادند برای توسعه سیستم باید مواردمربوط به موجودیت های درونی سرویس ها را نیز در نظر می گرفتیم که این امر سبب افزایش موجودیت های مورد مطالعه و روابط میان آنها شده و به شدت باعث افزایش پیچیدگی در امر طراحی و توسعه سیستم های نرم افزاری می شود.
- **سرویس ها قابلیت ترکیب دارند** به این معناست که سرویسی ها ممکن است از ترکیب چند سرویس دیگر تولید شوند. این امر باعث می گردد منطق حرفه در دانه بندی های متفاوت پوشش داده شده و باعث افزایش قابلیت استفاده مجدد و افزایش سطح انتزاع (abstraction) می شود.
- **سرویس ها خودمختار هستند** (**autonomous**) به این معناست که منطق و وظیفه مندی های درون سرویس با مرز آشکاری مشخص می شوند. سرویس در این مرز کنترل فعالیت ها را در دست دارد و به سرویس های دیگر برای اجرا وابسته نمی باشد.
- **سرویس ها وضعیت خود را نگهداری نمی کنند** (**stateless**) به این معناست که سرویس ها لازم نیست اطلاعات مربوط به وضعیت را مدیریت نمایند زیرا داده های مربوط به مدیریت وضعیت ممکن است روی قابلیت اتصال سست تاثیر منفی بگذارد. به همین

علت توصیه می شود سرویس ها به گونه ای طراحی گردند که کمترین اطلاعات وضعیتی را ثبت کنند.

- **سرویس ها قابل کشف هستند** به این معناست که سرویس ها باید اجازه دهند توصیفشان توسط افراد و متقاضیان سرویس درک و کشف شده تا توسط آنها مورد استفاده قرار گیرند. به عبارت دیگر باید مکانیسمی وجود داشته باشد که نام سرویس ها و ویژگی های آنها را در اختیار متقاضیان قرار دهد به طوری که متقاضیان با استفاده از این مکانیسم سرویسی که نیازمندی هایشان را پوشش می دهد کشف نموده و از آن استفاده کنند.

در بالا با هشت ویژگی اصلی سرویس ها آشنا شدیم. توصیه می شود در سرویس هایی که توسط فرآیندهای تشخیص سرویس ها شناسایی می شوند تا حد امکان این ویژگی ها رعایت شده باشند البته این به این معنی نیست که در صورتی که سرویسی یکی از ویژگی های فوق را نداشت لزوما از آن سرویس استفاده نمی کنیم بلکه این فاکتورها را می توان به عنوان معیاری برای سنجش کیفیت سرویس های شناخته شده و در نهایتارزیابی کیفیت روش های استفاده شده برای شناسایی سرویس ها استفاده نمود.

علاوه بر موارد فوق، در بخش ۲.۳ با فاکتور اندازه سرویس ها آشنا شدیم و دیدیم این فاکتور بر چهار ویژگی مهم در سیستم نهایی تاثیر می گذارد. این ویژگی ها عبارتند از:

- انعطاف پذیری (flexibility)
- قابلیت استفاده مجدد (reusability)
- کارآیی (performance)
- پیچیدگی (complexibility)

به عنوان مرور بر موارد فوق می توان گفت:

اگر سرویس ها بزرگ باشند برای تغییر در یکی از جنبه ها یا وظیفه مندی های سیستم باید دامنه گسترده ای از سیستم دستخوش تغییر شود. به همین علت بزرگی اندازه سرویس ها موجب کاهش انعطاف پذیری سیستم می گردد. در مقابل هر چه سرویس ها اندازه کوچکتری داشته باشند، برای تغییر در قسمتی از سیستم تنها تغییر در حیطه سرویس هدف کفایت می کند و به همین علت تغییر جنبه ای از سیستم سربار زیادی را ناشی نمی شود و در نتیجه انعطاف پذیری سیستم افزایش می یابد. قابلیت استفاده مجدد یکی از بارزترین و اصلی ترین ویژگی ها در معماری سرویس گرا به شمار می رود.

همانطور که پیش از این اشاره شد، قابلیت استفاده مجدد باعث کاهش زمان و هزینه تولید سیستم های نرم افزاری شده و تا حد ممکن در توسعه سیستم های نرم افزاری تلاش در افزایش این قابلیت داریم.

دانه بندی سرویس ها بر اساس حیطه وظیفه مندی سرویس ها تعیین می شود. به عبارت دیگر هرچه سرویسی حیطه بزرگتری از وظیفه ندی را پوشش دهد ، اصطلاحا ـن سرویس را بزرگتر و هر چه حیطه کوچکتری را پوشش دهد، اصطلاحا سرویس را کوچکتر گویند. در صورتی که سرویس بزرگ باشد، حیطه بزرگتری از وظیفه مندی های حرفه را پوشش می دهد و اصطلاحا سرویس در حیطه حرفه تنظیم شده و وابستگی بیشتری به حرفه خواهد داشت. همچنین به علت تفاوت در ماهیت حرفه های گوناگون امکان استفاده مجدد از آن سرویس در حرفه های دیگر کاهش می یابد و باعث کاهش قابلیت استفاده مجدد سرویس می گردد.

در مقابل هرچه سرویس کوچکتر باشد، وابستگی کم تری به حرفه داشته و به احتمال زیاد سرویس مطابق باوظیفه مندی خاصی طراحی شده است. به علت امکان بروز وظیفه مندی های مشابه در حیطه حرفه هایگوناگون امکان استفاده مجدد سرویس افزایش می یابد

از آنجا که برای برآورده کردن فرآیندهای حرفه تعدادی از سرویس ها با هم همکاری می نمایند، ارتباطات بین سرویس ها به این منظور نوعی سربار تلقی می گردند. به عبارت دیگر هنگام فراخوانی سرویس ها به منظور برآورده کردن فرآیند حرفه، مقداری از کارآیی سیستم جهت برقراری ارتباط

میان سرویس های مختلف هدر می رود. حال در صورتی که سرویس ها کوچک باشند، برای برآورده کردن فرآیندهای حرفه به تعداد بیشتری سرویس نیاز است و در نتیجه ارتباطات میان سرویسی افزایش یافته و سربار بیشتری به سیستم تحمیل می گردد. در نتیجه استفاده از سرویس های کوچک باعث کاهش کارآیی سیستم می گردد.

در مقابل هر چه سرویس ها بزرگتر باشند، برای پوشش نیازمندی های یک فرآیند حرفه به تعداد سرویس کمتری نیاز بوده و در نتیجه سرباز ناشی از ارتباطات میان سرویسی کاهش می یابد. همین امر باعث افزایش کارآیی سیستم نهایی می گردد.

آخرین فاکتوری که دانه بندی سرویس ها بر آن اثر می گذارد، فاکتور پیچیدگی سیستم می باشد. تاثیر دانه بندی سرویس ها بر این فاکتور تا حدودی شبیه تاثیر دانه بندی بر فاکتور کارآیی سیستم می باشد. در صورتی که سرویس ها کوچک باشند، به تعداد بیشتری سرویس جهت پوشش دامنه حرفه نیاز داریم. طبیعی است که این سرویس ها نیاز به تعامل با یکدیگر دارند. همین امر باعث می شود در صورت داشتن سرویس های کوچک پیچیدگی سیستم به علت تعداد زیاد ارتباطات افزایش یابد. در سمت مقابل هرچه اندازه سرویس ها بزرگ تر باشد، به تعداد سرویس کمتر و در نتیجه ارتباطات کمتر میان سرویسی نیاز خواهد بود. همین امر پیچیدگی سیستم نهایی را کاهش خواهد داد.

٤.٢ نقاط ضعف

در فصل سوم با روش های معمول در زمینه تشخیص سرویس ها (service identification) آشنا شدیم. همانطور که گفته شد در کل ۱۰ رده روش در این زمینه تا به حال معرفی شده است و با نقاط قوت و ضعف این گروه ها نیز تا حدی آشنا گردیدیم. همچنین برخی از روش های پر استفاده در این زمینه معرفی شده و به اختصار نقاط قوت و ضعف آنها مورد بررسی قرار گرفت.

به منظور ارائه روشی جدید که نقاط ضعف و قوت روش های موجود را پوشش دهد نیاز به شناسایی دقیقروش های گذشته است. همانطور که در فصل پیش دیده شد، اشکالات زیر به صورت عمده در روش هایگذشته به چشم می خورد:

۱. عدم ارائه روش مشخص و یا وابسته بودن روش معرفی شده:

در برخی از روش های ارائه شده، گام های مدون و مشخصی به منظور رسیدن به مجموعه سرویس های کاندید بیان نشده است. به عبارت دیگر در برخی از روش ها تنها به خروجی ها و ویژگی های خروجی ها اشاره شده گاهی تنها مراحل رسیدن به خروجی ها که همان سرویس ها هستند بیان شده است در صورتی که توضیحی در مورد این مراحل، نحوه تقدم و تاخر آنها و سایر موارد داده نشده است.

۲. در نظر نگرفتن فاکتورهای کیفی سرویس ها :

در برخی از روش های ارائه شده، تنها برخی فعالیت ها به منظور رسیدن به مجموعه سرویس های کاندید بیان شده و در این مراحل گام هایی به منظور رعایت و رسیدن به ویژگی های کیفی گنجانده نشده است. به عبارت دیگر ایندسته روش ها تنها به پوشش نیازهای وظیفه مندی و نیازهای دامنه مسئله بسنده کرده اند و به ویژگی های کیفی مورد انتظار از سرویس ها توجهی نشده است.

۳. در نظر نگرفتن فاکتور اندازه سرویس:

در بسیاری از روش ها به فاکتور اندازه سرویس توجهی نشده است. همانطور که پیش از این عنوان گردید فاکتور دانه بندی سرویس ها یکی از فاکتورهای تاثیر گذار بر چهار ویژگی کیفی سیستم نهایی می باشد که عدم توجه به آن ممکن است به سیستم نهایی ضربه بزنند. در مقاله *Inganti* [۱٤] به فاکتور دانه بندی سرویس ها تا حدی توجه شده است. البته باید در نظر داشت که در این مقاله هم دانه بندی سرویس ها بر اساس برنامه های کاربردی (application) موجود در زیر ساخت حرفه و سیستم های موروثی تنظیم می شود. به اینصورت که اگر فعالیتی توسط واسطی از برنامه پوشش داده شود به صورت سرویس شناخته

شده و اگر توسط یک واسط پوشش داده نشود، به زیر فعالیت ها تجزیه می گردد. حال اگر در سازمانی سیستم های موروثی پاسخگوی فعالیت ها نباشند و یا سازمان سیستم های موروثی نداشته باشد نمی توان دانه بندی مناسبی را برای سرویس ها تعیین نمود.

٤. امکان بروز افزونگی:

برخی از روش ها به این علت که به صورت غیر متمرکز به شناخت سرویس ها می پردازند ممکناست سرویس هایی با رویهم افتادگی وظیفه مندی تولید کنند به این صورت که یک وظیفه مندیخاص توسط دو یا چند سرویس مختلف برآورده شود که این امر باعث صرف زمان و هزینه بیهوده جهت این سرویس ها می گردد.

٥. عدم تعریف دقیق محدوده سرویس ها:

برخی روش ها به تعیین دقیق وظیفه مندی های سرویس ها نمی پردازند. به عبارت دیگر در بسیاری از روش ها گامی جهت تعیین دقیق حیطه وظیفه مندی سرویس ها وجود ندارد. این امر علاوه بر ایجاد ابهام در ادامه فعالیت های فاز مدلسازی سرویس گرا خود می تواند یکی از دلایل عمده در بروز افزونگی در سرویس ها باشد.

٦. عدم کنترل پوشش دهی نیازهای حرفه توسط سرویس های ارائه شده:

در بسیاری از روش ها گامی جهت بررسی این امر که آیا مجموعه سرویس های شناخته شده تمامی اهداف حرفه را پاسخ می دهند یا خیر وجود ندارد. در صورتی که سرویس های شناخته شده کامل نباشند یا به عبارت دیگر سرویس های شناخته شده تمامی اهداف را پوشش ندهند در مراحل بعد ممکن است سیستمی ناقص تولید گردد و یا در صورت تشخیص این امر در مراحل بعد هزینه و زمان زیادی جهت رفع این مشکل مورد نیاز است. به همین علت بهتر است گامی در روش مورد نظر به منظور بررسی مجموعه سرویس های کاندید از نظر کامل بودن گنجانده شود.

٧. عدم انجام کنترل کیفی سرویس ها:

در عمده روش ها گامی به منظور سنجش و ارزیابی کیفیت سرویس های شناخته شده وجود ندارد.

در صورتی که چنین گامی در روش وجود داشته باشد علاوه بر کسب آگاهی نسبت به کیفیت سرویس های شناخته شده، می توان کیفیت سیستم نهایی را بهبود بخشید و همچنین از توسعه سرویس هایی با کیفیت پایین جلوگیری کرد.

البته باید توجه داشت موارد فوق در تمامی روش ها وجود ندارد و این موارد به عنوان موارد با اهمیت که در تعدادی از روش ها رعایت نشده اند قابل ذکر هستند.

٤.٣ ویژگی های روش مورد انتظار

پس از آشنایی با ویژگی های مورد انتظار سرویس ها و نقاط ضعف موجود در روش های جاری در این بخش به ترسیم چارچوبی جهت ارائه روشی به منظور فرآیند تشخیص سرویس ها در حیطه معماری سرویس گرا می پردازیم.

همانطور که در بخش ٤.٢ عنوان شد، تعدادی معیار کیفی جهت ارزیابی سرویس ها وجود دارد. فرآیند تشخیص سرویس ها باید به گونه ای بطراحی شود که به صورت ضمنی سرویس هایی با این معیارها را تولید نماید. به عبارت دیگر نمی توان با استفاده از روشی سرویس ها را شناسایی کرده و سپس طی مراحلی این مجموعه سرویس ها را جهت تطبیق با این معیارها پیکربندی نمود، بلکه فرآیند تشخیص سرویس ها باید به گونه ای طراحی سود که خود سرویس هایی را شناسایی کند که دارای این ویژگی ها باشند.

البته باید توجه داشت که برخی از معیارهای کیفی سرویس ها را نمی توان طی فرآیند تشخیص سرویس ها لحاظ نمود. به عنوان مثال قابلیت کشف سرویس ها در حیطه تشخیص سرویس ها نمی گنجد و برای رسیدن به این ویژگی به اقداماتی در گام توصیف سرویس ها (service specification) نیاز است. با بررسی هایی از این دست مجموعه ای از فاکتورهای کیفی که در مرحله تشخیص سرویس ها قابل توجه اند بدست می آید که به شرح زیر هستند:

- قابلیت استفاده مجدد

- قابلیت ترکیب
- اتصال سست
- خودمختاری
- دانه بندی مناسب برای سرویس ها

البته در مورد ویژگی خودمختاری می توان گفت در فاز تشخیص سرویس ها باید سرویس هایی را شناسایی کرد که دارای حیطه مشخصی از وظیفه مندی باشند. همچنین در مورد سطح تجرید بالا (پنهان سازی اطلاعات درونی سرویس) و stateless بودن سرویس ها باید گفت این ویژگی ها طی فعالیت هایی در گام توصیف سرویس ها (service specification) قابل دستیابی می باشند.

برای رسیدن به قابلیت استفاده مجدد، باید سرویس ها به گونه ای طراحی شوند که در سیستم های دیگر قابل استفاده باشند. به این منظور باید تا حد امکان سرویس ها را طبق فرآیندهای استاندارد حرفه تهیه کرد. به عبارت دیگر برای رسیدن به قابلیت استفاده مجدد، می توان گامی را به منظور استاندارد سازی فرآیندهای حرفه به مجموعه گام های فعالیت تشخیص سرویس ها اضافه نمود. به این ترتیب سرویس های شناسایی شده در حیطه حرفه هایی که از این فرآیند استاندارد پیروی می کنند، قابل استفاده خواهند بود.

برای رسیدن به قابلیت ترکیب، سرویس ها برای ارتباط با یکدیگر باید از واسط های استانداردی طبعیت کنند که این امر نیز در گام توصیف سرویس ها باید تامین گردد. البته در فاز تشخیص سرویس ها می توان سرویس ها را به گونه ای شناسایی کرد که از لحاظ منطقی قابلیت ترکیب با برخی سرویس های دیگر را داشته باسند که با توجه به اینکه روش اصلی شناخت سرویس ها بر اساس فرآیندهای حرفه است، استاندارد سازی فرآیندهای حرفه به این امر کمک شایانی می نماید.

یکی از ویژگی های اصلی سرویس ها اتصال سست آنهاست. برای رسیدن به این ویژگی می توان از روشی به این شرح استفاده کرد:

ابتدا فرآیندهای حرفه به فعالیت ها، زیر فعالیت ها و .. تجزیه می گردد. این تجزیه تا سطحی ادامه می کند که در آن هر بخش تنها یک وظیفه مشخص و قابل تعریف را انجام دهد. سپس گرافی با استفاده از این بخش ها تولید می گردد. به این ترتیب که هر بخش به عنوان یگ گره در گراف نمایش داده می شود و در صورتی که بخش با بخش دیگری در ارتباط باشد، بین آن دو گره در گراف یالی رسم می شود. سپس گراف تولید شده ار می توان با استفاده از الگوریتم هایی به چندین پارتیشن تقسیم کرد به طوری که ارتباط درون پارتیشنی ماکسیسمم و ارتباط بین پارتیشنی مینیمم باشد. در آخر هر کدام از این پارتیشن ها را می توان به صورت یک سرویس در نظر گرفت. به این ترتیب سرویس هایی با اتصال سست شناسایی می شوند که دارای دانه بندی مناسبی می باشند.

علاوه بر گام های پیشنهادی فوق می توان گام هایی را جهت ارزیابی کامل بودن مجموعه سرویس های کاندید و همچنین کنترل کیفی سرویس ها بر اساس فرمول های مشخص در فرآیند تشخیص سرویس قرار داد.

نتیجه گیری

در این فصل به اختصار به تبیین چارچوبی جهت راه حل پیشنهادی در زمینه تشخیص سرویس های نرم افزاری پرداختیم. به این منظور ابتدا فاکتورهای کیفی مورد انتظار سرویس ها شناسایی شده و سپس فاکتورهایی که در گام تشخیص سرویس ها قابل دستیابی هستند مشخص گردید. همچنین نقاط ضعف روش های جاری در این زمینه تعیین شده و مورد تجزیه و تحلیل قرار گرفت. در آخر نیز گام هایی جهت دستیابی به ویژگی های کیفی مورد نظر و پوشش نقاط ضعف سایر روش ها معرفی شد. در ادامه این گزارش به تعریف مسئله اصلی می پردازیم. به عبارت دیگر به تعریف راه حل پیشنهادی و گام های رسیدن به آن جهت ارائه روشی به منظور تشخیص سرویس های نرم افزاری خواهیم پرداخت.

فهرست منابع

[1] Thomas Erl, Service Oriented Architecture: Concepts, Technology, and Design, Prentice Hall, 2019

[2] O. Zimmermann, N. Schlimm, G. Waller, and M. Pestel "Analysis and Design Techniques for Service-oriented Development and Integration," IBM Deutschland [3] Pooyan Jamshidi, Saeed Mansour, ASIM: Toward Automatic Transformation of Enterprise Business Model to Service Model, IEEE Transactions on service computing

[4] Jan-Willem Hubbers, Art Lightart, Linda Terlouw, Ten Ways to Identify Services, The SOA Magazine, 2007

[5] Solmaz Boroumand, Working with SOA and RUP, The SOA Magazine, March 2008

[6] Mark Endrei, Jenny Ang and Ali Arsanjani, Patterns: Service Oriented

Architecture and Web Services, IBM Corp, April 2004

[7] Bernhard Borges, Service Oriented Architecture, www.searchwebservices.com

[8] Iran's Information Architecture committee: www.esoa.ir

[9] Wikipedia : www.wikipedia.org

[10] O. Zimmermann, P. Krogdahl, and C. Gee, "Elements of Service-Oriented Analysis and Design," IBM® developerWorks

[11] A. Arsanjani, "Service-Oriented Modeling and Architecture (SOMA)", IBM® developer Works

[12] B. Portier, "SOA terminology overview, Part 3: Analysis and design," IBM® developer Works®

[13] S. Johnson, "Modeling service-oriented solutions," IBM® developer Works

[14] S. Inaganti, and G.K. Behara, "Service Identification: BPM and SOA Handshake," Technical Report. Business Process Trends

[15] J. Amsden, “Modeling SOA: Part 1. Service identification,” IBM® developer Works®, October 2007

[16] Zh. Zhang, R. Liu, and H. Yang, “Service Identification and packaging in

Service-oriented Reengineering,”

[17] Linthicum, “What Level Is Your SOA?”, 2004

[18] Oasis: SOA Adoption Blueprint, 2006, Available: www.oasisopen.org

[19] Borges, Holly and Arsanjani, “Service Oriented Architecture”, ٢٠٠٤,

Available: http://searchwebservices.techtarget.com/originalContent/

[20] Knorr, Rist, “10 Steps to SOA”, ٢٠٠٤, Available:

http://weblog.infoworld.com/article/05/11/07/45FEsoastep1_1.htm