دوره دوازدهم، شماره دوم، پاییز و زمستان ۱۳۹۸

مجله فناوری اطلاعات در طراحی مهندسی

Information Technology in Engineering Design

http://ited.sinaweb.net

ارائه روشی برای افزایش کیفیت سرویس‌دهی یکپارچه در معماری سرویس گرا

رضا رضائی* (۱)

(۱) گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، واحد ساوه، دانشگاه آزاد اسلامی، ساوه، ایران.*

(تاریخ دریافت : ۱۳۹۸/۹/٦ تاریخ پذیرش : ۱۳۹۹/۳/۱۲)

چکیده

معماری سرویس گرا نوعی معماری سازمانی است که از سرویس برای ساده‌سازی و اجرای فعالیت‌های یکپارچه‌سازی استفاده و در سال‌های اخیر باعث شده است تا سازمان‌ها، سرویس‌ها و داده‌های خود را به‌صورت باز ارائه و از توسعه‌دهندگان بیرون از سازمان نیز در جهت سرویس‌دهی و انتشار سرویس‌های خود استفاده نمایند با توجه به اینکه معماری سرویس گرا تلفیقی از سرویس‌های مختلف توزیع‌شده می‌باشد، کیفیت سرویس‌دهی در این معماری‌ها نیز باید به‌صورت برآیندی از کیفیت نحوه سرویس‌دهی به سرویس دهنگان مشخص شود. روش‌های موجود که تاکنون برای این نوع سرویس‌دهی تعریف‌شده‌اند عبارت‌اند از معماری سرویس گرا، معماری میکرو سرویس، معماری منولوتیک که هرکدام از آن‌ها با چالش‌های همچون سرویس‌دهی یکپارچه به توسعه‌دهندگان روبرو هستند، در این مقاله قصد داریم روشی جهت افزایش کیفیت سرویس‌دهی یکپارچه در معماری سرویس گرا با پشتیبانی از مفاهیم سرویس گرایی و ارتباط چندکانالی و با چند سناریو که مهم‌ترین چالش بالا بردن افزایش کیفیت سرویس‌دهی در یک سازمان می‌باشد را ارائه نمایم.

واژه‌های کلیدی: معماری سرویس گرا، دستگاه‌های توزیع‌شده، مهندسی مجدد فرآیند، میکرو سرویس.

* عهده‌دار مکاتبات:

نشانی:گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، واحد ساوه، دانشگاه آزاد اسلامی، ساوه، ایران.

پست الکترونیکی: *rezarezaei@iau-saveh.ac.ir*

۱. مقدمه

معماری سرویس گرا، سبکی از معماری سامانه‌های اطلاعـاتی است که هدف آن دستیابی به اتصال سست در ارتباطات بین مؤلفه‌های نرم‌افزاری است [۱-۵] و در تعریف دیگرمعماری سرویس گرا رهیافتی است برای ساخت سامانه‌های توزیع‌شده که کارکردهای نرم‌افزاری را در قالب سرویس ارائه می‌کند. این سرویس‌ها هم توسط دیگر نرم‌افزارها قابل فراخوانی هستند و هم برای ساخت سرویس‌های جدید مورداستفاده قرار می‌گیرند. درواقع هدف اصلی معماری سرویس گرا یکپارچه‌سازی و ارتباط بین کسب‌وکار با سازمان می‌باشد که مهم‌ترین دستاورد آن انعطاف‌پذیری و چابکی فناوری اطلاعات در برابر تغییرات حرفه است و منجر به تعامل‌پذیری توسعه‌دهنده و سرویس‌دهنده می‌گردد [٦].

هدف از بهینه کردن معماری سرویس گرا این است با توجه به پیچیدگی معماری‌های سرویس گرا در پیاده‌سازی و استقرار سرویس‌ها که با پیچیدگی خاصی روبرو و همچنین کنترل فرآیند آن‌ها نیز دشوار و دارای پیچیدگی محیطی می‌باشند، بتوان با ارائه روشی زمان تولید و کیفیت سرویس‌دهی قابل‌ارائه را با توجه به تغییرات گسترده کسب کارها بر اساس نیاز مشتریان ایجاد و با کنترل و ایجاد فرآیند بین سرویس‌ها در گذرگاه سرویس و ترکیبت آن با سرویس مدیریت فرآیند بین سرویس‌ها ارائه خدمات نمود. برای رسیدن به این هدف، معماری سرویس گرا باید خصوصیات خاصی ازجمله مدیریت و کنترل فرآیندها و مخزن سرویس، گذرگاه سرویس، انبار داده و مدیریت کانال یا سرویس و مانیتوریگ را داشته باشد. این ویژگی‌ها، قابلیت تغییر را در زمینه‌های مختلف ازجمله تولید و انتشار سرویس را فراهم می‌کنند [۷]. ازآنجایی‌که که معماری سرویس گرا یک معماری تلفیقی است بحث کیفیت سرویس‌دهی نیز باید در این قالب مطرح گردد. تحقیقات گسترده‌ای و متعددی در این حوزه انجام‌شده است که بیشتر آن‌ها حول محور کیفیت سرویس صورت گرفته است که نشانه اهمیت فراوانی است ما نیز در این مقاله پس از طرح کیفیت در معماری سرویس گرا ویژگی‌های خاص آن به بررسی روش‌ها و مدل‌های ارائه‌شده در این معماری پرداخته و درنهایت روشی پیشنهادی برای افزایش کیفیت سرویس‌دهی یکپارچه در معماری سرویس گرا بیان می‌نمایم [۸-۱۱].

برای افزایش کیفیت سرویس‌دهی در معماری‌های سرویس گرا شاخص‌های یا خصوصیات کیفی مدنظر در ذیل تعریف‌شده است [۱۲-۱٤].

- در دسترس‌پذیری

سرویس‌ها با توجه به سطح خدمات قابل‌ارائه باید در دسترس باشد و در صورت ازکارافتادن سرویسی، سرویس‌های دیگر بتوانند به کار خود ادامه دهند درواقع در دسترسی‌پذیری ترکیبی از پارامترهای اطمینان و قابلیت نگه‌داری است [١٦,١٥].

- مقیاس‌پذیری

اشاره به وضعیتی دارد که در آن میزان نفوذ تقریباً نسبت به تغییر تعداد یا واحدها یا اندازه ورودی‌ها تغییر می‌کند یا در تعریف دیگر تغییر منابع و سرویس‌ها به تعداد دلخواه حتی در تعریف فرآیندها در صورت افزایش تعداد کاربران یا نرم‌افزارهای مختلف سرویس‌دهی دچار اختلال نشود [۱۷-۲۵].

- انعطاف‌پذیری

دو تعریف برای انعطاف‌پذیری وجود دارد. چگونه می‌توان بدون نوشتن یک خط کد اضافی در برنامه از یک کلاس در زمینه مشابه یا جدید استفاده کرد یا آنکه در تعریف دیگر چطور می‌توان این کلاس‌ها را افزایش دهد تا بتوانند در یک ساختار جدید یا یکنواخت بدون طراحی مجدد کلاس‌ها استفاده شوند بسته به نیاز مشتری

یا لایسنس یا خودمختاری: مؤلفه‌های مستقل تا چه حد خودمختاری ازنظر زبان و سیستم‌عامل و داده و سایر مؤلفه‌های مربوط به توسعه و استقرار دارد [۲۶،۳۰].

• کار آیی

کار آیی به معیارهای عملکردی معماری که قابل‌اندازه‌گیری هستند شامل قابلیت دسترسی، زمان پاسخ، ظرفیت کانال، زمان تأخیر، زمان تکمیل، زمان سرویس پهنای باند، توان عملیاتی تعریف می‌شود. به‌عبارت‌دیگر زمان پاسخگویی به کاربر بر اساس سطح خدمات تعریف‌شده یا اینکه درصورتی‌که سرویسی از کار بیافتد دیگر سرویس‌ها بتوانند به کار خود ادامه دهند [۳۱].

• استفاده مجدد

به عبارتی چگونگی وابستگی بین سرویس گونه باید به‌گونه‌ای باشد بتوان قابلیت استفاده مجدد را تعریف نمود که سرویس‌ها باید مستقل از هم بتوانند در دستگاه‌های دیگر یا حتی از ترکیبت آن‌ها با یکدیگر بتوان سرویس جدید را ایجاد یا استفاده نمود [۳۲-۳۸].

• داده محوری

داده محوری به در معماری سرویس گرا بیشتر به اهمیت پردازش و استناد به داده‌های سیستم برای تشخیص فرآیندها و شناخت رفتار و درگیر کردن از اطلاعات جهت بهره‌برداری بهتر از سرویس‌ها تعریف می‌شود

• پشتیبانی از ارتباط چند کانالی

پشتیبانی از ارتباط چند کانالی بودن با توجه به پیشرفت‌های اخیر در حوزه معماری سرویس گرا و کاربرد مفاهیم جدید در صنعت سرویس‌دهی به مشتری از قبیل سرویس‌دهی باز یا داده باز و یا نوآوری در این نوع خدمات و اینکه هر خدمتی را می‌توان از طریق کانالی شروع و در کانال دیگر به اتمام رساند معماری سرویس گرا را برداشته است که برای حفظ بقای خود رویکردی سرویس خود را به ارائه سرویس از طریق چند کانال با ارتباط یکپارچه تغییر دهد.

• مدیریت فرآیندها

مدیریت فرآیند کسب‌وکار در دهه نود به‌عنوان مهندسی مجدد فرآیند کسب‌وکار رایج بوده است. فرآیندهای کسب‌وکار موضوع تازه‌ای نمی‌باشند زیرا از زمان شکل‌گیری واحدهای اقتصادی مطرح هستند.

با توجه به خصوصیات و شاخص‌های تعریف‌شده به روشی نیاز داریم که در ادامه در بخش دوم روش‌های موجود را معرفی و چالش‌های آن‌ها را موردبررسی قرار می‌دهیم و در بخش سوم روش پیشنهادی مطرح و تشریح می‌گردد و در بخش چهارم روش پیشنهادی را مورد ارزیابی با توجه به خصوصیات مطرح‌شده قرار خواهد گرفت و در بخش پنجم نیز نتیجه‌گیری و ارزیابی روش بر اساس سناریو محور توضیح داده می‌شود.

۲- روش‌های موجود

امروزه فناوری اطلاعات به یک عامل ضروری برای پیشرفت و ارتقاء سازمان‌ها تبدیل‌شده است و هم‌زمان با رشد فناوری اطلاعات سازمان‌ها و حتی دولت نیز برای همگرا شدن با این پیشرفت پیشگام شده‌اند به‌طوری‌که بسیاری از سرویس‌های درون‌سازمانی خود را که تاکنون بسته و دور از دسترس بوده را نیز بازنموده و آن را با سطح امنیتی خاصی در اختیار توسعه‌دهندگان قرار داده‌اند. استفاده از معماری سرویس گرا و انتشار یکپارچه سرویس‌ها از گذرگاه سرویس و شکستن سرویس‌ها به سرویس‌های کوچک به‌عنوان یکی از فرآیندهای زنجیره تأمین و توسعه نرم‌افزار با کمترین هزینه و زمان و دسترسی از هر مکانی، جایگاه ویژه‌ای در مدیریت کسب‌وکار دارد. در یک نگاه کلی، فرآیند ایجاد و انتشار سرویس از مرحله شناسایی نیازمندی‌ها موردنظر توسعه‌دهندگان تا دریافت خدمات به‌عنوان یک سرویس برای رفع نیازمندی‌ها را شامل می‌شود. به دلیل ارتباطی که بین این فرآیند و دیگر فرآیندهای تهیه

سرویس وجود دارد، هرگونه بهبود در آن می‌تواند بر عملکرد کسب‌وکار و حتی کل زنجیره تأثیر بگذارد. درنتیجه بهبود کیفیت سرویس‌دهی بر مبنای معماری سرویس گرا مبتنی بر میکرو سرویس، کمکی بزرگی در جهت توسعه سرویس‌ها در زمان کوتاه‌تر و کیفیت بهتر است.

۲-۱ معماری نرم‌افزار

برای معماری نرم‌افزار، تعریفی که به‌طور عمومی پذیرفته‌شده باشد، وجود ندارد. افراد مختلف، معماری نرم‌افزار را به اشکال گوناگون تعریف کرده‌اند. این تعاریف، ازلحاظ ظاهری متفاوت‌اند ولی به مفهوم مشترکی اشاره می‌کنند.

معماری نرم‌افزار یک برنامه یا سیستم کامپیوتری، ساختار یا ساختارهایی از سیستم است که دربرگیرنده اجزاء، صفات قابل‌مشاهده آن اجزا و ارتباط بین آن‌ها باشد. همچنین معماری نرم‌افزار به‌صورت زیر تعریف‌شده است [٦] [٨]:

معماری نرم‌افزار، سازمان زیربنایی سیستم است که در قالب اجزا و روابط بین آن‌ها و همچنین روابط آن‌ها با محیط بیان‌شده است و برای طراحی و تکامل آن اصولی وجود دارد. در این نوع تعریف، فرآیند تولید معماری، بخشی از معماری در نظر گرفته‌شده است [۹-۱۰]. (زیرا قواعد و اصول طراحی و تکامل نیز به‌عنوان معماری مطرح‌شده‌اند.) معماری هر سیستم نرم‌افزاری می‌تواند بدون توجه به نحوه تولید آن مشخص و ارزیابی گردد.

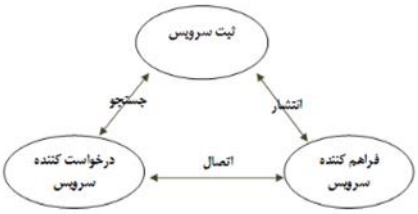
۲-۲ معماری سرویس گرا

سبکی از معماری سامانه‌های اطلاعاتی است که هدف آن دستیابی به اتصال سست در ارتباطات بین مؤلفه‌های نرم‌افزاری است.

معماریی است برای ساخت سامانه‌های توزیع‌شده که کارکردهای نرم‌افزاری را در قالب سرویس ارائه می‌کند. این سرویس‌ها هم توسط دیگر نرم‌افزارها قابل فراخوانی هستند و هم برای ساخت سرویس‌های جدید مورداستفاده قرار می‌گیرند. این معماری برای یکپارچه‌سازی فناوری‌ها در محیطی که انواع مختلفی از سکوهای نرم‌افزاری و سخت‌افزاری وجود دارد ایده آل است [٤-٥].

معماری سرویس گرا از دیدگاه‌های مختلف قابل‌بررسی است، هر فرد یا ذینفع بر طبق جایگاه خود تصویری از این معماری دارد، در ادامه این معماری ازنظر کارشناسان کسب‌وکار، معماران و طراحان سامانه‌های اطلاعاتی موردبررسی قرار می‌گیرد. [۷-۸].

کاری که به‌وسیله یک سرویس‌دهنده انجام می‌شود که ممکن است انجام یک درخواست کوچک روی‌داده مانند دریافت یا ذخیره اطلاعات باشد یا مربوط به انجام کاری پیچیده‌تر مانند چاپ یک تصویر باشد [۲۷].



شکل (۲-۲) مدل پایه معماری سرویس گرا [۱۲].

مهم‌ترین دستاورد معماری سرویس گرا انعطاف‌پذیری و چابکی فناوری اطلاعات در برابر تغییرات کسب‌وکار است که به‌عبارت‌دیگر [۸,۱٤].

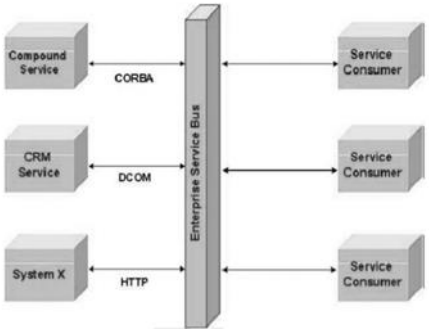
- حضور فعال‌تر و مسئولانه‌تر فناوری اطلاعات در سازمان‌ها.
- کاهش زمان چرخه تولید و توسعه سامانه‌های اطلاعاتی به خاطر استفاده از واحدهای قابل‌استفاده مجدد.
- کاهش پیچیدگی و هزینه نگهداشت.
- ارتقاء سامانه‌های اطلاعاتی موجود به‌جای جایگزینی یکجای آن‌ها.
- کاهش هزینه و زمان جهت پیکربندی مجدد.
- انعطاف‌پذیری کسب‌وکار به دلیل افزایش

دانه‌بندی از فرآیند به سرویس.

• قابلیت ایجاد سریع فرآیندهای جدید و ترکیب مؤلفه‌های نرم‌افزاری موجود جهت رقابت با تغییرات بازار.

۲-۳ گذرگاه سرویس سازمانی

سرویس باس یا گذرگاه سرویس مسیری است که از طریق آن، سرویس‌ها برای همه‌ی مشتریان یا توسعه‌دهندگان در دسترس قرار می‌گیرد و همچنین سرویس‌های پیاده‌سازی در یک سازمان همدیگر را از طریق سرویس باس یافته و فراخوانی می‌کنند و ارتباط مستقیم و دوبه‌دو ندارند. یکی از مزایای این مدل جلوگیری از پیچیدگی ارتباطات بین سامانه‌ها در صورت افزایش تعداد ماژول‌ها می‌باشد.



شکل (۲-۵) گذرگاه سرویس سازمانی (باس)

در حال حاضر ترکیب مدیریت فرآیند کسب‌وکار و معماری سرویس گرا بسیار رایج است و اکثر سامانه‌های مدیریت فرآیند کسب‌وکار که در دنیای امروزی مطرح‌اند از فناوری‌های معماری سرویس گرا استفاده می‌کنند.

• معماری سرویس گرا بدون مدیریت فرآیند کسب‌وکار: اگر معماری سرویس گرا در سازمان بزرگ کسب‌وکار به‌تنهایی به کار گرفته شود، سرویس‌هایش توانایی بهبود و بهینه‌سازی مداوم نیازمندی‌های کسب‌وکار را نخواهد داشت درنتیجه با داشتن مدیریت فرآیند کسب‌وکار، فاز بهینه‌سازی و بهبود فرآیندها به‌درستی صورت می‌گیرد و بهینه‌سازی سرویس‌ها نیز تسهیل می‌گردد. با این موضوع مدیریت فرآیند کسب‌وکار برای معماری سرویس گرا ضروری است.

• مدیریت فرآیند کسب‌وکار بدون معماری سرویس گرا: اگر مدیریت فرآیند کسب‌وکار در سازمان تجار به‌تنهایی به کار گرفته شود نمی‌توان سازمان تجاری را توسعه داد زیرا باوجود معماری سرویس گرا می‌توان پیاده‌سازی مدیریت فرآیند کسب‌وکار را در سطح سازمان ساده نمود و لایه کنترلی را فراهم کرد. معماری سرویس گرا می‌تواند نقش حیاتی در موفقیت مدیریت فرآیند کسب‌وکار داشته باشد و فناوری اطلاعات را به‌منظور تعریف و کنترل نحوه تعامل‌پذیری فرآیندهای کسب‌وکار با سامانه مهیا سازد و به استفاده مجدد، کنترل و تهیه اتصال سست بین ماژول‌ها کمک می‌کند.

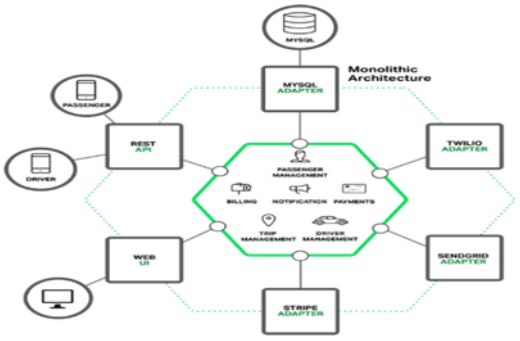
می‌توان نتیجه گرفت مدیریت فرآیند کسب‌وکار و معماری سرویس گرا بمانند دو طرف سکه هستند که در یک‌طرف معماری سرویس گرا مسیر را برای تکثیر و رشد مدیریت فرآیند کسب‌وکار هموار می‌کند و توسط ابزارهایی، فرآیندها را مدل می‌کند و سریعاً پیاده‌سازی می‌گردد. در طرف دیگر مدیریت فرآیند کسب‌وکار راه را برای رشد و ترقی معماری سرویس گرا هموار می‌کند و ارتباط بین فناوری اطلاعات و کسب‌وکار را نزدیک‌تر و پلی بین آنان ایجاد می‌نماید.

جدول (۲-٤) مقایسه بین مدیریت فرآیند کسب‌وکار و معماری سرویس گرا

معماری سرویس گرا	مدیریت فرآیند کسب‌وکار
تحریک‌کننده فناوری اطلاعات	تحریک‌کننده کسب‌وکار
رویکرد معماری پایین به بالا	رویکرد فرآیندی بالا به پایین
به‌کارگیری مجدد پیاده‌سازی سرویس‌ها	به‌کارگیری مجدد مدل فرآیند
زیر ساختار سازمان گرا	پروژه گرا
سنجش بر اساس شاخص‌های معماری، سازگاری، جامعیت و هزینه تغییر	سنجش بر اساس شاخص‌های راندمان و شاخص‌های کسب‌وکار

۲-٤ معماری مونولونیک

در این معماری بخش‌های مختلف برنامه سمت سرور ازجمله پردازش پرداخت آنلاین، مدیریت حساب‌ها، اعلان‌ها و سایر بخش‌ها همگی در یک واحد منفرد جمع شده‌اند. به عبارتی اگر برنامه تحت وب که در سرور قرار دارد به‌صورت یکجا با تمام متعلقات خود برای پاسخ به درخواست‌های سمت مشتری، کار با پایگاه داده و انجام سایر الگوریتم‌ها اجرا شود، این برنامه از معماری مذکور استفاده می‌کند.

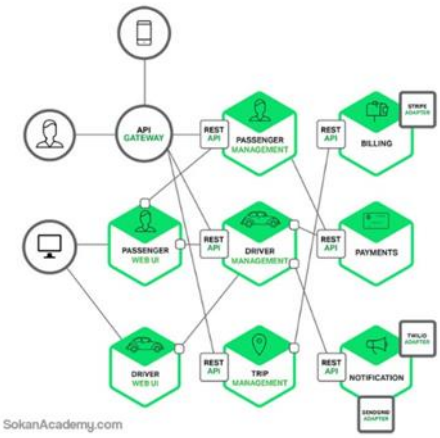


شکل (۲-۱) – معماری مونولیتیک

در معماری مذکور زمانی که ترافیک برنامه در سمت سرور افزایش پیدا می‌کند، باید برای پاسخگویی، اندازه را افزایش داد؛ یعنی باید برنامه تحت وب خود را بر روی سرورهای مختلف مجدداً اجرا نمود. بخشی به نام متعادل‌کننده بار، وظیفه توزیع درخواست‌ها را به سرورهای مختلف که بر روی هر یک، یک نسخه از برنامه در حال اجرا است، به عهده دارد. بر اساس توضیحی که از این معماری ارائه شد، در هر یک از این اجراها، کل برنامه با تمام متعلقاتی که دارد، فارغ از اینکه به همه آن‌ها نیاز است یا نه از منابع سرور استفاده می‌کند.

۲-۵ معماری میکروسرویس

میکروسرویس روشی به‌منظور تقسیم‌بندی کردن یک اپلیکیشن (نرم‌افزار) به بخش‌ها یا سرویس‌های کوچک، سبک، مستقل از یکدیگر و قابل مدیریت است. به‌عبارت‌دیگر، میکرو سرویسیک معماری توسعه نرم‌افزار به‌اصطلاح توزیع‌شده است.



یک رویکرد به توسعه‌ی نرم‌افزار است که در آن‌یک برنامه‌ی بزرگ، به‌صورت مجموعه‌ای از خدمات ماژولار ساخته‌شده است. هر ماژول یک هدف خاص را دنبال می‌کند و با استفاده از یک رابط کاربری ساده و مناسب با دیگر ماژول‌ها در ارتباط است. میکرو سرویس درواقع در پی پیاده‌سازی واقعی از معماری سرویس گرا ست که باگذشت زمان به سمت برنامه‌های مونولیتیک سوق پیداکرده بود. در معماری میکروسرویس هر میکرو سرویس طی یک فرآیند منحصربه‌فرد اجرا می‌گردد و پایگاه داده مختص به خود را مدیریت می‌کند. این امر نه‌تنها امکان توسعه‌ی نرم‌افزار با رویکرد غیرمتمرکز را ایجاد می‌کند، بلکه اجازه می‌دهد هر سرویس مستقلاً مستقر، اجرا و مدیریت گردد. به‌بیان‌دیگر در این سرویس‌ها حداقل مدیریت متمرکز وجود دارد و این بدین معنی است که هرکدام می‌توانند زبان برنامه‌نویسی مختلف نوشته‌شده و حتی دیتابیس ذخیره‌سازی متفاوتی داشته باشند.

جدول ۲-٦ چالش‌های موجود در معماری میکرو سرویس

چالش	مزیت
پیچیدگی سامانه توزیع‌شده است	ماژولار است برای تبدیل‌شدن به سامانه یکپارچه شدن بسیار ساده است
یکپارچگی	در استقرار، مستقل است
پیچیدگی عملیاتی	بسیار سریع است تغییرات به‌راحتی توسط استقرار مداوم هر یک از سرویس‌های مایکروسافت آزاد می‌شود

هنگامی‌که در مورد میکروسرویس‌ها صحبت می‌شود این سؤال مطرح می‌گردد که آیا این معماری همان معماری سرویس گرا است که یک دهه پیش مطرح گردید؟ علت شباهت این دو نوع معماری را می‌توان مشمول بودن مزایای موردعلاقه‌ی طرفداران معماری سرویس گرا در معماری میکروسرویس‌ها، دانست. مشکل اصلی این است که تعاریف بسیار متفاوتی از معماری سرویس گرا وجود دارد. معمولاً این نوع معماری با معماری‌های متشکل از چند سرویسیکپارچه اشتباه گرفته می‌شوند [۳۱, ۳۲].

نهایتاً مشکلات مدیریت متمرکز معماری سرویس گرا باعث شد تا طرفداران معماری میکرو سرویس معماری سرویس گرا را به‌طور کامل طرد کنند، هرچند که دیگران همچنان معماری میکرو سرویس را حالتی از معماری سرویس گرا می‌دانند [۳۲].

با توجه به ماهیت معماری میکرو سرویس استفاده از ابزارهایی متداول در حوزه مدیریت فرآیند مانند گذرگاه سازمانی گذرگاه سرویس باس و یا یک هماهنگ‌کننده مرکزی در این مدل با چالش‌هایی مواجه می‌شود. از طرفی دیگر مورد بسیار ضروری اندازه‌گیری شاخص تا در ابزار مدیریت فرآیندها که لازمه آن وجود یک هماهنگ‌کننده مرکزی بین سرویس‌ها است با ذات میکرو سرویس در تضاد است. دیدگاه‌ها و بحث‌های بسیاری در این حوزه در دنیای نرم‌افزار در جریان است و همانند بسیاری از مفاهیم دیگر نرم‌افزار هنوز هیچ برنده قطعی برای این مسئله پیدا نشده است [۳۳, ۳٦].

با توجه به ذات میکرو سرویس آن‌ها سرویس‌های هستند که به‌صورت پراکنده قرارگرفته‌اند سرویس‌ها نیاز به نگهداری وضعیت اجرا ندارند و درصورتی‌که در مدل فرآیند نیاز به نگهداری وضعیت آن‌ها است در معماری میکرو سرویس، سرویس‌ها بدون هماهنگ و کنترل‌کننده متمرکز با یکدیگر درگیر هستند

درصورتی‌که ذات مدیریت کسب‌وکار منظم و هماهنگ‌کننده است که این چالش‌های اصلی این دو رویکرد است و این سؤال مطرح است درصورتی‌که این دو با یکدیگر ترکیب شود چه ارزش‌افزوده می‌توانند داشته باشند [۳۷]. که مزایای استفاده از آن شرح داده‌شده است:

- هرکدام از قسمت‌های فرآیند می‌تواند یک زیرسیستم باشد که بتوان آن را پیاده‌سازی کرد.
- سامانه‌ها حساس به گرفتن سرویس‌های زیاد نمی‌باشند و فقط سرویس‌های موردنیاز را دریافت می‌نمایند
- مانیتورینگ سرویس‌ها و کنترل فرآیندها بسیار ساده است

مقایسه معماری‌های و کارهای قبلی

برای ارزیابی معماری همان‌طور که بیان شد از شاخص‌های کلیدی معرفی‌شده در بخش‌های بالا استفاده کردیم که همه معماری‌های از آن‌ها در ساختار خود استفاده کرده بودند و همین مبنا را قراردادیم تا آن‌ها را با معماری سرویس گرای موجود در سازمان بررسی نمودیم که در جدول ۷-۲ شرح داده‌شده است.

جدول ۲-۷ ارزیابی معماری‌های مبتنی بر سرویس بر اساس پارامترهای مورد ارزیابی‌شده

معماری	در دسترس پذیری	کارایی	ارتباطات چند کاناله	مقیاس پذیری	استفاده مجدد	داده محوری	انعطاف پذیری	مدیریت فرآیندها
معماری میکرو سرویس	*	-	*	*	*	-	*	-
معماری مونولیتیک	*	*	-	-	-	-	*	
معماری سرویس‌گرا	*	*	*	-	-	-	-	*
معماری سرویس گرای موجود	*	*	*	*	-	*	*	

همان‌طور که مشاهده می‌نمایید معماری سرویس گرای موجود در سازمان با معماری پیشین مورد ارزیابی قرار گفته است و مشخص‌شده است معماری‌های سرویس گرای موجود و پیشین در بخش‌های مدیریت فرآیندها و ترکیب سرویس‌های برای استفاده مجدد، داده محوری ضعف‌های را دارد یا در برخی پوشش داده نشده است که سازمان‌ها را بر بخش‌های برای ارائه سرویس باکیفیت دچار مشکل می‌نماید.

۳- روش پیشنهادی

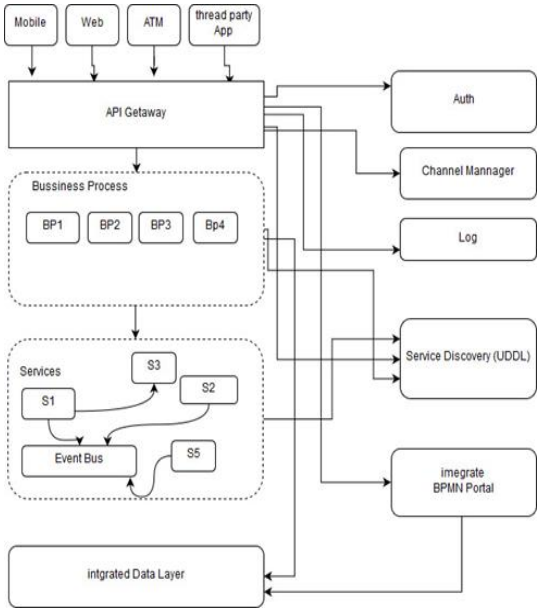
در بخش قبلی به اهمیت موضوعات مدیریت فرآیند کسب‌وکار، معماری سرویس گرا و معماری میکرو سرویس و بررسی نظریه و کارهای انجام‌شده در این دو زمینه پرداخته‌ایم. در این فصل می‌خواهیم با ارائه روشی ترکیب این دو معماری و فرآیند به‌عنوان سرویس در کنار معماری مطرح روشی را جهت بهینه‌سازی سرویس‌دهی در این نوع معماری‌های به مطرح می‌نماییم که علاوه بر یکپارچگی و جامعیت بهتر در این معماری‌ها سرویس‌دهی به توسعه‌دهندگان را بهبود می‌دهد.

برای بهینه کردن و پیاده‌سازی کردن معماری‌های مبتنی بر سرویس نیاز است که در آن ماژول‌های نرم‌افزار و سرویس‌های ارائه‌شده در آن در عین استقلال و در حالت جزیره‌ای بودن بتوانند کارنمایید (مفهوم میکرو سرویس) و به دیگر ماژول‌ها یا حتی سرویس‌ها بتوان سرویس ارائه نماید و در صورت ایجاد یک خدمت متمایز با دیگر خدمات ارائه‌شده بتوان با کمترین هزینه و ترکیب حتی چند سرویس باهم سرویس جدید را ایجاد و آن را در قالب یک سرویس جدید ارائه نمود. همچنین با پیدایش پلتفرم‌های جدید بتوان بدون تغییر در سرویس‌های ماژول‌ها از آن‌ها برای ارائه خدمات به پلتفرم‌های جدید استفاده نمود. درعین‌حال با تغییر محیط اجرای نرم‌افزار از سرورهای سنتی به ابر و یا از ابر به سرورهای سنتی بتوان نرم‌افزار را از حالت

توزیع‌شده به یکپارچه یا برعکس بدون تغییر ماژول انتقال داد.

تعدد درخواست‌های جدید از سمت مشتریان و توسعه‌دهندگان و نیز تعاریف جدید سرویس‌دهی همچون داده‌های باز، سرویس باز، نوآوری بازو نیز تحمیل باز هزینه‌ای تولید و نگهداری این درخواست‌ها در حال حاضر برای سازمان‌ها بسیار سنگین بوده و در برخی از اوقات به دلیل نبودن پشتیبانی مناسب سرویس‌های مشتریان و توسعه‌دهندگان آن‌ها ناراضی شده و یا حتی امکانی وجود ندارد که بتوان سرویس‌ها را به‌صورت باز با امنیت کافی در اختیار آن‌ها قرارداد و سازمان‌ها مجبور هستند این سرویس‌ها خود یا تنها توسط چند پیمان‌کار انجام دهند که این امر علاوه بر هزینه زیاد، آن نتیجه‌ای که باید حاصل شود عمدتاً حاصل نشده و از سرویس‌دهی و حتی رقابت با دیگر رقبا ممکن نیز جا بمانند این‌ها در صورتی است که سازمان‌ها از معماری‌های مبتنی بر سرویس در سازمان خود استفاده نمایند و با مفاهیم سرویس و سرویس گرایی و مدیریت فرآیندها آشنایی دارد.

روش پیشنهادی در معماری‌های مبتنی بر سرویسیا سرویس گرا بر اساس ساختار معماری‌های مجود صورت می‌پذیرد؛ که با اضافه کردن یک‌لایه به این نوع معماری‌ها به‌عنوان لایه مدیریت سرویس فرآیندی که در شکل ٦ زیر نمایش داده‌شده است می‌توان کیفیت سرویس‌دهی را تا حد مطلوبی افزایش داد. این لایه در معماری کمک بسیار زیادی به توسعه‌دهندگان و ارائه‌دهندگان می‌کند زیرا بجای آنکه سرویس‌های متعددی را در برنامه‌های کاربردی خود استفاده نمایند فقط سرویس اجرای فرآیند را در برنامه‌های ایجادشده خود، فراخوانی می‌نماید که این امر باعث ارائه خدمات در کمترین زمان به مشتریان استفاده‌کننده از این سرویس‌ها را می‌دهد و امنیت داده‌ای و فرآیندی نیز در این نوع درخواست‌های به دلیل اینکه ساختار فرآیند در لایه معماری و به‌دوراز محیط تولیدشده توسط توسعه‌دهندگان ایجاد گردیده بیشتر رعایت شده است؛ زیرا توسعه‌دهندگان نیازی به دانش و زیرساخت فرآیند ایجادشده نیست از فرآیندهای پیاده‌سازی شده و ارتباط سرویس‌ها آگاه نمی‌باشند و فقط با دسترسی مجاز به لیست سرویس‌های فرآیندی دسترسی دارند و دستور اجرای آن را صادر می‌نماید. البته نیاز است در اینجا بخشی از بار سیستم و مانیتورینگ سرویس‌ها بدوش سازمان‌ها ارائه‌دهنده سرویس قرار بگیرد که این با توجه اینکه این مباحث مانند داده باز و سرویس‌های باز مطرح است هزینه کمتری را ازنظر امنیتی نیز برای سازمان‌ها به همراه دارد.



همان‌طور که می‌دانیم معماری‌های مبتنی بر سرویسیا سرویس گرا از چندلایه اصلی استفاده می‌نمایند که این رهیافت پیشنهادی می‌توانند برای کلیه این معماری‌ها در نظر گرفته شوند که به شرح ذیل شرح داده‌شده است:

روند کلی این رهیافت به این صورت است که سازمان‌ها در لایه مخزن سرویس‌های ایجادشده در معماری، توسط یک‌لایه فرآیند کسب‌وکار، فرآیند موردنظر را با ترکیب سرویس‌ها موردنظر آن فرآیند

ایجاد می‌نماید و امکان فراخوانی آن‌ها را از مخزن سرویس فرآیندی مهیا می‌نماید و توسط یک گذرگاه سرویس که در ارتباط با یک مدیریت کانال است در معماری‌های مبتنی بر سرویس در اختیار توسعه‌دهندگان قرارمی گرد و توسعه‌دهندگان فقط دسترسی به اجرای سرویس فرآیندی آن‌هم بعداً از اینکه درخواست خود را فقط توسط این لایه به مدیریت کانال ارجاع داده و در صورت اعلام مجوز ورود به مخزن سرویس‌ها دسترسی دارد.

درنتیجه ایجاد یک فرآیند دیگر برای تولید محصولی جدید بنا به درخواست توسعه‌دهندگان یا مشتریان سازمان‌ها می‌تواند گاها از ترکیب فرآینده‌های جدید که خود نیز قابلیت پیاده‌سازی را دارد در کمترین زمان حتی گاها بدون نیاز به برنامه‌نویسی نیز می‌توان یک فرآیند جدید را در قالب یک سرویس جدید نیز ارائه نمود. برای اینکه بتوان سرویس‌دهی را بهینه نمود با اضافه کردن لایه سرویس دیسکاوری به این لایه‌ها در خصوص دسترسی به سرویس فرآیندها ابتکاری صورت گرفته است که بین دولایه فرآیند و سرویس اتصال محکمی را ایجاد نموده است در این رهیافت پیشنهادی در لایه سرویس از میکروسرویس‌ها استفاده‌شده است. اصلی‌ترین عناصر روش پیشنهادی در معماری‌های مبتنی بر سرویس عبارت‌اند از:

- لایه سرویس: در لایه سرویس با توجه به اهداف، وظایف و فرآیندهای سازمانی و نیز سامانه‌ها و برنامه‌های کاربردی، سرویس‌های مورد استخراج و طراحی می‌شود و همچنین توسعه می‌یابند و توضیحات مربوط به هر سرویسیا سرویس وب در یک مخزن سرویس تحت عنوان UDDI قرار می‌گیرد.
- لایه فرآیندهای کسب‌وکار: در این لایه فرآیندهای سرویس‌های قابل‌ارائه سازمانی تعریف و با استفاده از ابزارها و فناوری‌های رایج مانند BPML مدل‌سازی و تحلیل می‌شود.
- لایه امنیت: وظیفه این لایه‌ایجاد ارتباط امن با استفاده از فناوری‌های احراز هویت، تعیین دسترسی، رمزنگاری و ردیابی در تمام لایه‌های معماری است
- لایه کنترل: این لایه کلیه فرآیندهای اتفاق افتاده در سیستم را ثبت و بررسی می نمای دو امکان رسیدگی به خطاها را فراهم می‌نماید.
- لایه دیتا: این لایه دربردارنده‌ی کلیه منابع اطلاعاتی سازمان اعم نرم‌افزارها و برنامه‌های کاربردی سازمان‌ها است که درواقع پایه اصلی تشکیل سرویس‌ها است.
- لایه داده در این رهیافت برای سرویس‌های عمودی استفاده‌شده در لایه سرویس نیست و این سرویس‌ها آزادی استفاده از دیتابیس خود را دارا می‌باشند. این لایه داده پیشنهادی برای این است که ما در مدل کلاسیک سرویس‌های فرآیندی احتیاج داریم نمای کلی و کاملی از همه داده‌های، فرآیندها داشته باشیم، اما همچنان که در رهیافت پیشنهادی این مشاهده می‌کنید به‌جای یک سیستم متمرکز که مجموعه‌ای از فرآیندها باشد، هر یک از فرآیندهای ما به شکل واحدهای مستقل نصب و راه‌اندازی می‌شوند و درواقع تفاوت ماهیتی با سرویس‌های عمودی استفاده‌شده در لایه سرویس ندارند.

استفاده از ابزارهای انبار داده و یا ابزارهایی مانند آپاچی اسپارک برای فراهم کردن اطلاعات تحلیل‌شده در این لایه توصیه می‌شود چون در این حالت می‌توان پرتال یکپارچه‌ای که نمای کل داده‌های سیستم، شاخص‌های کلیه فرآیندهای لایه سرویس فرآیند و سایر داده‌های تحلیلی را نمایش دهد تولید کرد که در کنار فراهم کردن امکان مانیتورینگ مورداستفاده مدیران ارشد و تحلیلگران فرآیندها برای تحلیل کلان وضعیت سیستم قرار بگیرند.

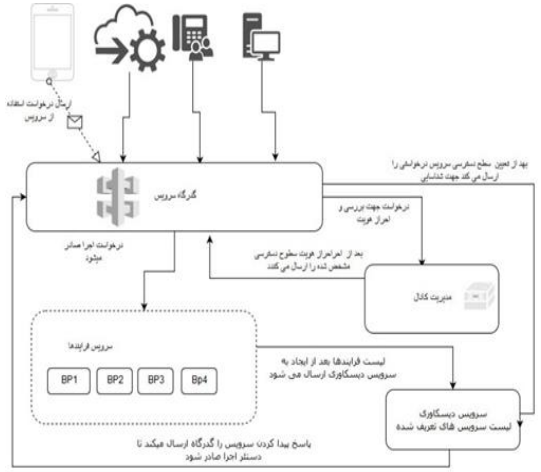
تشریح معماری پیشنهادی

در این رهیافت ارائه‌شده در معماری‌های مبتنی بر سرویسیا سرویس گرا سعی شده است که مفهوم سرویس‌های مدیریت فرآیند که معمولاً طراحی فرآیند با نشانه‌گذاری به نام BPMN و اجرای آن‌ها بر بستر BPMS است با مفاهیم معماری میکرو سرویس ادغام شود تا بتوانیم از مزایای هر دو معماری بیشترین استفاده را داشته باشیم. تفاوت‌های اصلی رهیافت پیشنهادی با روش‌های موجود در معماری کلاسیک سامانه‌های مدیریت فرآیند به شرح ذیل است:

- **نصب هر یک از سرویس‌های فرآیند به‌عنوان یک سیستم مستقل**

در حالت کلاسیک مجموعه سرویس‌های فرآیند به شکل یکپارچه بر بستر یک سامانه یا نرم‌افزار نصب و در دسترس قرار می‌گیرند اما در رهیافت پیشنهادی مطابق با دستورالعمل میکرو سرویس هر یک از فرآیندها در یک بستر مستقل نصب و اجرا می‌شود. ترجیح بر این است که سرویس‌های فرآیند از BPMS استفاده کنند اما اجباری در این مسئله نیست و هر یک از سرویس‌های فرآیند می‌تواند از زبان و فنّاوری دلخواه خود برای طراحی و اجرای فرآیندها استفاده کند. نقش مجموع این سرویس‌ها همانند سیستم کلاسیک مدیریت فرآیند است و وظیفه هماهنگ‌سازی سرویس‌های درگیر در فرآیند و پیش برد مسیر فرآیند را بر عهده‌دارند. برای اینکه این یکپارچگی از نگاه مشتری نهایی ایجاد شود بخش‌های زیر به معماری اضافه‌شده است:

- **گذرگاه سرویس که همان‌طور که بالاتر گفته شد تمامی درخواست‌های دسترسی به سرویس‌ها از این لایه گذر می‌کند و پس از انجام و پایش‌های لازم برای امنیت و بازرسی توسط مدیریت کانال با استفاده از سرویس دیسکاوری سرویس فرآیند موردنظر را یافته و درخواست‌ها را به سمت آن هدایت می‌کند که در شکل ۱۱ نمایش داده است. گذرگاه سرویس باید به نحوی طراحی شود که داشتن نمونه‌های اجرایی متعدد از آن و استفاده از ابزارهای کنترل بار بین سرورها بدون مشکل خاصی امکان‌پذیر باشد تا مقیاس‌پذیری سیستم دچار مشکل نشود. گذرگاه سرویسیک‌لایه توصیه‌شده در معماری میکرو سرویس است که کلاینت نهایی را دچار چالش یافتن سرویس‌های پراکنده این معماری نمی‌کند و نمایی یکپارچه از سیستم نشان می‌دهد.**



شکل شماره ۳-۳- نمونه از ارتباط درخواست سرویس تا زمان اجرای آن

- **سرویس دیسکاوری یا UDDI: به دلیل تعدد سرویس‌ها و گستردگی مکان نصب آن‌ها در رویکرد میکرو سرویس، پیدا کردن سرویس‌ها برای یکدیگر تبدیل به چالش می‌شود. لذا ابزارهایی برای این کار توسط نت فلیکس و آمازون و سایر سردمداران معماری میکرو سرویس تولیدشده است که کار ثبت سرویس‌ها و مسیر دسترسی بر آن‌ها، کنترل کردن فعال یا غیرفعال بودن سرویس‌ها و پاسخ به درخواست‌های یافتن سرویس را بر عهده دارد. این ابزارها مسیرهای مختلفی که یک سرویس نصب‌شده است را به کلاینت برمی‌گرداند و ابزارهایی نیز سمت سرویس درخواست‌کننده وجود دارد که بتواند به شکل لود بالانس شده درخواست‌ها را توزیع کند. این ابزارها از پروتکل‌های متعددی مانند http یا dns برای ثبت و**

بازیابی سرویس‌ها پشتیبانی می‌کند.

با توجه به تعاریفی که تابه‌حال مطرح‌شده است ممکن است این‌طور به نظر برسد که گذرگاه سرویس نقشی مانند سرویس باس را در معماری سرویس گرا بازی می‌کند اما این لایه تفاوت‌های ماهیتی اساسی با سرویس باس که معمولاً گذرگاه تنگ کار آیی معماری سرویس گرا است، دارد. باس سازمانی برای جلوگیری از پیچیدگی ارتباط زیرسیستم‌ها به شکل مستقیم با یکدیگر است به این شکل که به‌جای اینکه هر سرویسی با هر سرویس دیگر مستقیم تعامل کند درخواست خود را به باس می‌دهد و باس سرویس موردنظر فراخوانی می‌کند. سرویس باس می‌تواند سینک یا آسینک باشد و معمولاً نقش تبدیل بدنه درخواست سرویس مبدأ به درخواست موردپذیرش سیستم مقصد را هم بر عهده دارد. درواقع کارکردهای سرویس باس یافتن سرویس‌های مختلف و همچنین انجام تبدیلات لازم برای این تعامل است؛ که در زیر تفاوت‌های عمده آن‌ها باهم شرح داده‌شده است.

در این بخش با روش پیشنهادی در معماری‌های مبتنی بر سرویسیا سرویس گرا آشنا شدیم. ابتدا دلایل ارائه روش پیشنهادی برای افزایش سطح سرویس‌دهی یکپارچه بیان شد و پیش مدل پیشنهادی با عناصر و مؤلفه‌های آن و همچنین فناوری‌های مورداستفاده درروش پیشنهادی مطرح تشریح گردید. مدل یا روش پیشنهادی و همچنین فناوری‌های مورداستفاده در آن با توجه به نقاط قوت و ضعف معماری‌های موجود و همچنین اهداف موردنظر در پایان‌نامه بوده است.

برای بهینه‌سازی سرویس‌دهی از حذف گذرگاه سرویس و استفاده از گذرگاه سرویس مطرح و با اضافه کردن اضافه کردن لایه سرویس فرآیند و گذرگاه رویدادها و میکرو سرویس در این معماری‌ها برای یکپارچگی ببین سرویس‌ها و بهینه کردن سرویس‌دهی استفاده‌شده است.

بنابراین مهم‌ترین ویژگی‌های رهیافت پیشنهادی عبارت‌اند از:

- ایجاد یک‌لایه سرویس باس و جایگزینی آن با لایه سرویس.
- ایجاد لایه مانیتورینگ و خطاها برای کنترل و مدیریت بهتر.
- ایجاد لایه سرویس فرآیند و اتصال آن به لایه کشف سرویس جهت شناسایی سرویس‌ها.
- تغییر مسیر درخواست سرویس به درخواست فرآیند.
- استفاده از میکرو سرویس بجای سرویس در لایه سرویس‌ها.
- ایجاد یک‌لایه رویداد برای دسترسی برای ایجاد یک‌لایه میانی بین سرویس و فرآیند.
- پشتیبانی از مفاهیم Omni channel و چند کانال ارتباطی.

همان‌طور که در فصل بعدی بررسی خواهد شد رهیافت مذکور می‌تواند موجب بهبود سرویس‌دهی یکپارچه در معماری‌های مبتنی بر سرویسیا سرویس گرا شود.

٤- ارزیابی رویش پیشنهادی

رهیافت پیشنهادی باید از دو جنبه مورد ارزیابی قرار گیرد، در مرحله اول جامعیت و کلیت روش پیشنهادی ارزیابی می‌شود و در مرحله دوم فناوری‌های استفاده‌شده در این رهیافت مورد ارزیابی و بررسی قرار می‌گیرد.

برای ارزیابی جامعیت و کلیت رهیافت پیشنهادی با معرفی تعدادی شاخص و بررسی چند سناریو کاربردی به ارزیابی کلیت آن می‌پردازیم و برای فناوری‌های مورداستفاده با توجه به اینکه فناوری‌های میان روش‌های آزموده شده است مورد ارزیابی قرارگرفته است که پیش‌تر درستی آن توسط محققان به اثبات رسیده است.

٤-١ شاخص‌های ارزیابی

پشتیبانی از سرویس‌دهی یکپارچه: تمامی کلاینت‌ها از

طریق هر کانالی که متصل شوند نقطه ورودشان از گذرگاه سرویس است که برای تمام کانال‌ها از مجموعه سرویس‌های مشترک استفاده می‌کند. درواقع بااینکه به نظر می‌رسد مجموعه‌ای از سرویس‌ها به شکل جزیره وجود دارد اما این جزایر به شکل یه واحد کل به کاربر نمایانده می‌شود و مانند روش‌هایی بسیار قدیمی که کلاینت‌ها هر یک به سرویس مستقل خود متصل می‌شدند و ارتباط داده‌ای بین سرویس‌ها برقرار نبود نیست.

• پشتیبانی از چندکاناله بودن: بازهم با توجه به معماری و نحوه اتصال کلاینت‌ها به مجموعه سرویس‌ها کاربر می‌تواند درخواست‌ها و تعاملش را از هر کانالی که برایش مقدور بود انجام دهد و با گسستگی و ناهماهنگی داده‌ها در کلاینتهای مختلف وب و موبایل مواجه نشود.

• خودمختار بودن سرویس‌ها: همین‌طور که توضیح داده شد لایه سرویس و لایه سرویس فرآیند هر دو با رویکرد میکرو سرویس طراحی‌شده‌اند و سرویس‌ها در مورد سرویس مربوط به خود و نحوه انجامش خودمختاری دارند. هر یک از سرویس‌ها می‌توانند در تعداد لازم تکثیر شوند و هر سرویسی انتخاب می‌کند که با کدام سرویس‌ها تعامل داشته باشد. درواقع معماری پیشنهادشده مجموعه‌ای از سرویس‌ها را با استفاده از هر دو مکانیزم choregraphy و orchestration به شکل یک سیستم یکپارچه و هماهنگ در اختیار کلاینت‌ها قرار می‌دهد.

• امنیت: در این معماری تمامی سرویس‌های داخلی می‌تواند پشت فایروال داخلی قرار بگیرد. تمامی درخواست‌های ارسالی به هر یک از سرویس‌ها باید یک توکن jwt که قبلاً از سیستم auth دریافت شده است را در هدر تمام درخواست‌ها به هر یک از سرویس‌ها ارسال کند.

• مدیریت فرآیندها: مدیریت فرآیندها در سرویس‌دهی، راه‌حل جامع برای مدل‌سازی، طراحی، اجرا، تحویل و بهینه‌سازی سازمان است. در این معماری سعی بر این شده است که سرویس‌ها در قالب فرآیند طراحی و مدیریت شود و کلیه رویدادهای که در این فرآیندها رخ می‌دهد جهت کنترل‌های لازم ذخیره و نگه‌داری می‌شود؛ و حتی فرآیندها قادر به این هستند که هریک به‌تنهایی گسترش یابد.

٤-٢ سناریوی کاربردی

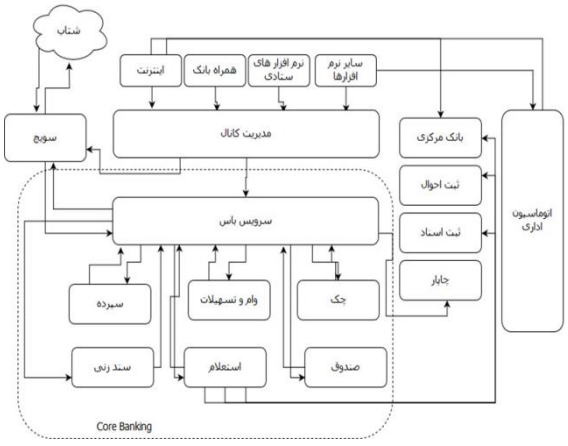
برای ارزیابی معماری پیشنهادی و بررسی امکان دسترسی به شاخص‌ها معرفی‌شده در بخش قبلی از یک سناریو بر اساس معماری پیشنهادی تشریح گردد و در پایان اهداف انجام سناریو و میزان تحقق آن بررسی گردد. لازم به ذکر است به دلیل وجود محدودیت‌های بسیار در پیاده‌سازی معماری پیشنهادی در معماری مطرح‌شده در سازمان (بانک ایران‌زمین) ارزیابی آن امکان‌پذیر نبوده و به همین دلیل سناریو، تمرکز بر لایه سرویس و مدیریت فرآیند بین آن‌ها که به‌عنوان محور و موضوع اصلی این تحقیق مطرح‌شده صورت گرفته است.

سازمان موردنظر یا سازمان خصوصی است به نام بانک ایران‌زمین که شامل تعداد زیادی سامانه‌های اطلاعاتی از قبیل سامانه بانکداری متمرکز، اینترنت بانک، سامانه‌های حسابداری، سامانه‌های کارکنان، خودکارسازی اداری، سامانه‌های آموزشی و چند سرویس داخلی و خارجی است. درنتیجه دارای حجم زیادی از اطلاعات بانکی و داخلی می‌باشد؛ که معماری آن به شرح توضیح داده می‌شود.

معرفی معماری و نمای استقرار فعلی در سازمان موردمطالعه.

سامانه‌های بانکی، سامانه‌هایی با چندین و چند مؤلفه هستند و تعاملات بسیاری نیز با نهاد و سازمان‌های

خارج از بانک دارد. در حال حاضر معماری بروزترین سامانه‌های بانکی ایران به این شکل است که یک سامانه مرکزی که اصطلاحاً به آن کر بنکیگ می‌گویند وجود دارد که شامل چندین زیرسامانه می‌شوند. در مدل معماری بانک ایران‌زمین این زیرسامانه‌ها هر یک به شکل یک وب اپلیکیشن مستقل قابل پیاده‌سازی و گسترش هستند. این زیرسامانه‌ها از طریق یک گذرگاه سرویس باس با یکدیگر تعامل می‌کنند. برای تعاملات بین‌بانکی مبتنی بر کارت بر روی شبکه شتاب نیز یک سامانه سوییچ وجود دارد که گاها در کنار کر بنکیگ و گاهی به‌عنوان یک زیرسامانه از آن نصب می‌شود. بر روی این مجموعه کر بنکیگ و سوییچ یک سامانه مدیریت کانال وجود دارد که مسئولیت احراز هویت کاربر درخواست دهنده و همچنین کنترل سطوح دسترسی این کاربر از طریق این کانال بر روی سرویس درخواستی را بر عهده دارد و درصورتی‌که مشکلی در احراز هویت و کنترل دسترسی وجود نداشته باشد سامانه مدیریت کانال سرویس را از طریق گذرگاه سرویس باس یا سوییچ فراخوانی می‌کند و سپس نتیجه را به کلاینت برمی‌گرداند.



شکل ٤-١- نمای استقرار فعلی سامانه در سازمان

به‌منظور حل چالش سازمان مذکور معماری پیشنهادی مورداستفاده قرار گرفت که در فصل قبلی در مورد معماری آن بحث گردید و نحوه اجرای سناریو بر اساس آن پیاده‌سازی گردیده است که ابتدا سناریو با وضع فعلی شرح داده‌شده است و سپس با معماری پیشنهادی که این خود می‌تواند نمونه‌ای از کار آیی یا ارزیابی معماری پیشنهادی باشد.

اجرای سناریو

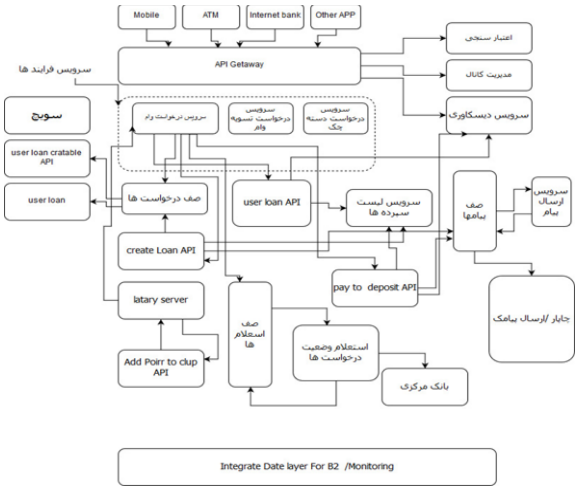
ازجمله اقدامات و عملیاتی که در این سازمان انجام‌شده است آماده کردن چند فرآیند مانند درخواست تسهیلات، درخواست دسته‌چک، درخواست تسویه وام است. منظور از فرآیندهای مذکور مراحلی و فرآیندی است که یک مشتری برای انجام دادن آن باید به محل شعبه مراجعت نماید و اقداماتی را انجام دهد که خود شامل سرویس‌های از قبیل جستجوی مشتری، سوابق عملکردی در شعبه، میزان تعهدات، سرویس‌های استعلام از منابع خارجی و اخذ تأییدیه‌های مختلف از مجموعه‌های مختلف است.

تهیه چنین فرآیندهای می‌تواند یک چالش شود زیرا عملیات جستجوی مشتری و درخواست وام یا هر درخواست دیگری متناسب با درخواست ورودی که خود از میان انبوهی از سرویس‌ها با وظیفه‌های مشابه باید انتخاب شود صورت گیرد. علاوه بر آن ذخیره اطلاعات کار آیی لازم را نخواهد داشت زیرا بخشی از عملیات در حال حاضر به‌صورت دستی باید مقایسه شود و دقت قابل قبولی وجود ندارد و همچنین تهیه گزارش‌ها نیز با توجه به آنکه دخالت کارکنان را دارد ممکن است دچار خطا شود و تضمینی برای صحت و راست سنجی آن وجود ندارد؛ و عملاً این کارها هر بار به‌صورت تکراری انجام‌شده و وقت بسیاری هزینه می‌شود تا خدماتی به مشتریان داده شود.

با توجه به محدودیت بسیار در پیاده‌سازی مطالعه موردی مطرح‌شده در این بخش چند مورد از کاربرد معماری پیشنهادی که در بانک ایران‌زمین از بین سناریوهای متعددی که در یک سامانه بانکی پیش می‌آید

در این پایان‌نامه چند مورد که ازنظر فرآیندی ساختار پیچیده‌تری داشت را در این بخش معرفی می‌کنیم تا سپس بتوانیم نحوه مدل و اجرای این سناریوها در معماری پیشنهادی را بررسی کنیم.

نمای معماری و مدل استقرار معماری پیشنهادی برای سناریوهای معرفی‌شده



شکل ۲-٤- نمای معماری سرویس گرا با معماری پیشنهادی با توجه به سناریوی معرفی‌شده

همین‌طور که در شکل ٤-۲ مشاهده می‌کنید تغییرات معماری پیشنهادی در معماری مبتنی بر سرویسیا سرویس گرا نسبت به معماری فعلی به ترتیب زیر است:

- ارتباط مستقیم با مدیر کانال حذف‌شده و درخواست‌کننده سرویس به گذرگاه سرویسیا سرویس‌گیرنده متصل می‌شوند.
- فراخوانی مدیریت کانال توسط گذرگاه سرویس صورت می‌گیرد و تنها برای کنترل دسترسی‌ها نه برای فراخوانی سرویس توسط مدیر کانال.
- یک سامانه سرویس دیسکاوری که به گذرگاه سرویس و سایر سرویس‌ها اجازه می‌دهد خود را در آن ثبت کنند و آدرس سرویس‌های موردنظر را استعلام کنند اضافه‌شده است.
- لایه گذرگاه سرویس باس حذف‌شده و گذرگاه سرویس سرویس‌های موردنیاز را توسط سرویس دیسکاوری یافته و سپس درخواست‌ها را به سمت آن‌ها هدایت می‌کند.
- به‌جای یک سامانه یکپارچه خودکارسازی، سرویس‌های فرآیندی متعدد برای هر یک از سناریوها اضافه‌شده است، این سرویس‌ها قابلیت دیپلوی مستقل از هم رادارند.
- به‌جای زیرسامانه‌هایی که از طریق گذرگاه سرویس باس با یکدیگر تعامل می‌کردند اکنون سرویس‌های بیزینسی و سرویس‌های سامانه‌ی از دو طریق سرویس دیسکاوری و یا ایونت باس های مبتنی بر rabbitmq با یکدیگر تعامل می‌کنند.

هرگاه که عملیات فراخوانی یک سرویس به اتمام رسید کنترل به سرویس فرآیندی برای ادامه کار و فراخوانی سرویس بعدی بازمی‌گردد؛ اما درعین‌حال فراخوانی یک سرویس مانند لیست وام‌های کاربر ممکن است به فراخوانی سرویس جزییات سپرده به شکل مستقیم توسط خود سرویس لیست وام‌ها منجر شود. همچنین سرویس‌ها هم با یکدیگر و هم با سرویس‌های فرآیندی ارتباطات اسینک از طریق ایونت باس دارند مثلاً سرویس پرداخت بعد از انجام پرداخت رویدادی از نوع اطلاع‌رسانی در صف نوتیفایر قرار می‌دهد تا سرویس نوتیفایر با توجه به زمان‌بندی خود آن را برای کاربر ارسال کند.

مدیریت فرآیندها و سرویس فرآیند

سرویس‌های دیده‌شده در لایه سرویس فرآیند می‌توانند مبتنی بر bpmn طراحی و در بستر bpms اجرا شوند. به این طریق انعطاف‌پذیری و سرعت عمل در تغییر رفتار فرآیندها بسیار بیشتر از حالتی است که همگام‌سازی عملیات در لایه میکرو سرویس به شکل choregraphy انجام شود و یا اینکه به شکل هارد کد در اپلیکیشن تعبیه شود.

رسیدگی به خطاها

در این معماری به دلیل استفاده از فنّاوری bpms و همچنان رویکرد میکرو سرویس از ابزارهای غنی و متنوعی در مانیتورینگ سیستم در سطوح مختلف ازنظر KPI و سرویس‌دهی، نمایش بالا و پایین بودن سرویس‌ها، آمار میزان موفق یا ناموفق بودن ارتباط بین سرویس‌ها در فراخوانی‌های یکدیگر بهره می‌برد. علاوه بر این وجود یک‌لایه داده که برای عملیات BI و Data mining با استفاده از ابزارهای big data processing در نظر گرفته‌شده است امکان نمایش انواع اطلاعات تحلیلی و آماری را بر روی عملکرد کلی سیستم می‌دهد.

نمایش نتایج

به‌منظور نمایش نتایج، خروجی به‌دست‌آمده آر مراحل قبل ارائه می‌شود. این خروجی‌ها عبارت‌اند از روش‌های مستند شده، مجموعه از سناریوها در فرآیندهای آماده‌شده در حوزه فناوری اطلاعات، خصوصیات کیفی، ریسک‌های کشف‌شده، نحوه سرویس‌دهی.

روش‌های معمارانه استفاده‌شده در رهیافت پیشنهادی بدین‌صورت می‌باشد. سبک انتخابی برای ارائه رهیافت پیشنهادی مدل توزیع‌شده می‌باشد که برای دستیابی به این منظور از ترکیب معماری‌های مبتنی بر سرویسیا سرویس گرا با معماری میکرو سرویس سود برده شده است. همچنین برای پیکربندی معماری خصوصیات رهیافت پیشنهادی در قالب اضافه کردن لایه‌های معماری و استفاده از نرم‌افزارهای مدیریت کسب‌وکار و سرویس دیسکاوری که در قسمت‌های قبلی توضیح داده است به این معماری‌های اضافه‌شده است. سناریوهای و فرآیندهای مستند شده در مراحل ارزیابی به ترتیب اولویت در جدول (٤-١) در زیر ارائه‌شده است.

جدول ٤-١ سناریو و فرآیندهای مس تندشده در مراحل ارزیابی رهیافت پیشنهادی

سناریو	توضیحات
سناریو ١	در صورت تغییر یک پلتفرم و یا به وجود آمدن یک پلتفرم جدید، در کمترین زمان بتوان ارائه خدمات را روی پلتفرم شروع کرد.
سناریو ٢	درصورتی‌که ازکارافتادن یک فرآیند دیگر فرآیندها از کار نیافتند.
سناریو ٣	در صورت افزایش تعداد کاربران یا نرم‌افزارهای مختلف سرویس‌دهی دچار اختلال نشود
سناریو ٤	وابسته به سخت‌افزار خاصی نباشد و در هر محیطی بتوان از سرویس‌ها استفاده نمود
سناریو ٥	قادر به پشتیبانی از مفاهیم چندکاناله و omni channel در بازار رقابت‌های دیجیتالی باشد.

سناریو ٦	قادر باشد تجربیات حاصل از اتفاقات را ذخیره نماید تا بتوان در صورت لزوم از دانش به‌دست‌آمده استفاده نمود.
سناریو ٧	زمان پاسخ‌دهی در حالت عادی و در زمان افزایش بار، تغییری احساس نشود
سناریو ٨	روش پیشنهادی باید بجای فراخوانی سرویس از لایه سرویس باس، سروش را از لایه سرویس دیسکاوری پیدا و سپس قادر باشد فرآیند را اجرا کند نه سرویس را.
سناریو ٩	روش پیشنهادی کلیه عملیات رخ‌داده را ثبت و امکان مانیتورینگ را باید داشته شد
سناریو ١٠	لایه سرویس باید بجای استفاده از مخزن سرویس از میکروسرویس‌ها استفاده نماید و ارتباط سرویس‌ها نیز باید مشخص باشد.

همان‌طور که در جدول ٤-١ نشان داده‌شده است سناریوهایی مطرح در رهیافت پیشنهادی در معماری‌های سرویس گرا باید قادر به پوشش آن‌ها باشد.

جدول ٤-٢ خصوصیات کیفی معماری

خصوصیات کیفی رهیافت پیشنهادی	در دسترس‌پذیری	ارائه خدمات روی هر پلتفرم نرم‌افزاری
	نگه‌داری	سازگاری روش پیشنهادی با تغییرات احتمالی در محیط
	مقیاس‌پذیری	تغییر منابع و سرویس‌ها به تعداد دلخواه حتی در تعریف فرآیندها
	انعطاف‌پذیری	کم‌وزیاد کردن ماژول‌های در معماری بسته به نوع درخواست‌ها
	کار آیی	قابلیت تعریف فرآیندهای جدید و انتشار آن روی گذرگاه سرویس و ادامه کار در صورت ازکارافتادن فرآیند یا سرویس خاصی.
	زمان سرویس‌دهی	با اضافه شدن درخواست و بار سیستم زمان پاسخ‌دهی تغییری ننماید
	داده محوری	کلیه اتفاقات باید در سیستم ذخیره‌سازی شود تا در صورت نیاز بتوان از آن استفاده نمود

	omni channel	سرویس‌ها باید قابلیت اجراشدن روی تمامی کانال‌ها با رعایت سطح دسترسی از هر کانال را داشته باشد
	مدیریت فرآیندها	با اضافه شدن مدیریت فرآیندها روی میکرو سرویس مدیریت آن‌ها از سطح به سطح فرآیند تغییریافته و سرعت اجرای آن بهینه‌تر خواهد شد

همان‌طور که در جدول شماره ٤-٢ مشاهده می‌نماید رهیافت پیشنهادی ارائه‌شده خصوصیات موردنظر را پوشش می‌دهد و طبق روش‌های معمارانه ی استفاده‌شده و نیز سناریوهای تشریح شده نیازمندی‌های سیستم را به‌طرف می‌نماید و همچنین با این رهیافت پیشنهادی درصورتی‌که فرآیندی از سرویس‌دهی خارج شود دیگر فرآیندها به کار خود ادامه داده و سیستم پایدار می‌ماند و با مدیریت گزارش‌گیری که در این روش پیشنهادی اضافه‌شده است تمام فرآیند در حال رصد شدن می‌باشند و کوچک‌ترین تغییرات نیز اطلاع‌رسانی می‌گردد. مهم‌ترین هدف این روش پیشنهای پایدار نگه‌داشتن سرویس‌دهی برای سرویس‌دهی بهتر در کمترین زمان با کمترین هزینه می‌باشد.

جدول ٤-٣ پارامترهای ارزیابی معماری

در دسترس‌پذیری	سرویس‌ها با توجه به SLA در دسترس باشند درصورتی‌که ازکارافتادن یک فرآیند دیگر فرآیندها از کار نیافتند.
مقیاس‌پذیری	تغییر منابع و سرویس‌ها به تعداد دلخواه حتی در تعریف فرآیندها در صورت افزایش تعداد کاربران یا نرم‌افزارهای مختلف سرویس‌دهی دچار اختلال نشود
انعطاف‌پذیری	کم‌وزیاد کردن ماژول‌های در معماری بسته به نیاز مشتری و لایسنس قابلیت تعریف فرآیندهای جدید و انتشار آن روی گذرگاه خودمختاری: مؤلفه‌های مستقل تا چه حد خودمختاری ازنظر زبان و سیستم‌عامل و داده و سایر مؤلفه‌های مربوط به توسعه و استقرار دارد
کار آیی	زمان پاسخگویی به کاربر بر اساس SLA باشد (زمان پاسخگویی می‌تواند با توجه به لود شبکه تعیین شود) ادامه کار در صورت ازکارافتادن فرآیند یا سرویس خاصی.
استفاده مجدد	وابستگی سرویس‌ها تا چه حد قابلیت استفاده مجدد دارند ارتباط و وابستگی بین سرویس‌ها به چه سبکی است؟ آیا می‌توان از سرویس‌ها به شکل مستقل در سیستم دیگری استفاده کرد؟ ریزدانگی سرویس‌ها سرویس‌ها تا چه حد تخصصی و مربوط به یک تسک مشخص هستند؟
داده محوری	آیا داده‌های سیستم سالم هستند؟ (با یکدیگر و با قوانین کلی سیستم سازگاری دارند)

	قادر باشد داده حاصل از اتفاقات را به نحوی ذخیره کند که در آینده بتواند برای تحلیل داده کاربرد داشته باشد.
omni channel	قادر به پشتیبانی از مفاهیم چندکاناله و omni channel در بازار رقابت‌های دیجیتالی باشد.
مدیریت فرآیندها	بانک باید قادر باشد که بتواند با ترکیب سرویس‌ها فرآیندهای جدید بسازد و بعد از مشاهده متریک های KPI هایی که تعریف‌شده است در صورت لزوم مسیر گردش کار را تغییر دهد.

همان‌طور که در جدول شماره ٤-٣ نشان داده‌شده است، برای ارزیابی معماری پیشنهادی نسبت به معماری فعلی پارامترهای متداول در حوزه مقایسه سرویس‌ها به‌عنوان معیار ارزیابی معرفی‌شده‌اند. به این معیارها مورد داده محوری و مدیریت فرآیندها و omnichannel به دلیل نیاز شدید سیستم بانکی به این موارد اضافه‌شده است.

خلاصه مطالب تشریح شده در این فصل در جهت ارائه یک ارزیابی و مقایسه کیفی بین دو سیستم موجود و سیستم پیشنهادی در جدول ٤-٤ نشان داده‌شده است.

جدول ٤-٤ ارزیابی و مقایسه کیفی دو معماری موجود و پیشنهادی ازنظر در دسترس‌پذیری:

معیار	در دسترس‌پذیری
معماری موجود	١- استفاده از ابزارهای مانیتورینگ وضعیت سرور و سامانه عامل ٢- استفاده از ابزارهای مانیتورینگ وضعیت پایگاه داده
امتیاز معماری موجود	B
امتیاز معماری پیشنهادی	A+
معماری پیشنهادی	١- استفاده از ابزارهای مانیتورینگ وضعیت سرور و سامانه عامل ٢- استفاده از ابزارهای مانیتورینگ وضعیت پایگاه داده ٣- استفاده از پنل های ابزارهای سرویس دیسکاوری برای مشاهده بالا بودن سرویس‌ها ٤- استفاده از داشبوردهای مانند هیستریکس برای نمایش میزان و وضعیت فراخوانی سرویس‌ها توسط سایر سرویس‌ها ٥- نصب و راه‌اندازی سرویس‌ها می‌تواند مستقل از یکدیگر انجام شود و همین مسئله موجب داون تایم پایین‌تر شود. ٦- به دلیل استقلال دیتا و کانتینر هر سرویس مسائلی مانند مشکلات حافظه یا اتصال به دیتابیس یک سرویس موجب اختلال و از دسترس خارج شدن همه سرویس‌ها نمی‌شود

جدول ٤-٥ ارزیابی و مقایسه کیفی دو معماری موجود و پیشنهادی ازنظر مقیاس‌پذیری:

معیار	مقیاس‌پذیری
معماری موجود	١- مقیاس‌پذیری افقی به‌راحتی انجام نمی‌شود و معمولاً ابتدا به سمت مقیاس‌پذیری عمودی حرکت می‌شود.

	۲- استفاده از لود بالانسر در سمت سرور ۳- هر یک از ماژول‌ها که مجموعه زیادی از سرویس‌ها و جابهای بک‌گراند می‌باشند باید بر روی چندین سرور نصب شود. ٤- حتی اگر تنها یک سرویس لود بالایی داشته باشد به دلیل استفاده از esb به غیر از ماژول دارنده سرویس ماژول esb که ماژول پیچیده و سنگینی است باید در چندین کانتینر نصب شود
امتیاز معماری موجود	C
امتیاز معماری پیشنهادی	A+
معماری پیشنهادی	۱- مقیاس‌پذیری افقی بخش مهمی از پارادایم این معماری می‌باشد. ۲- به دلیل توزیع بار در بین چندین سرویس مستقل به‌راحتی در وضعیتی که برای مدیریت لود احتیاج به توزیع داشته باشد قرار نمی‌گیرد. ۳- تنها ماژولی که به شکل متداول در این معماری احتیاج به لود بالانس دارد api-gateway است که به دلیل سبکی و نداشتن وابستگی به دیتابیس به‌راحتی انجام می‌شود. ٤- ابزارهای سرویس دیسکاوری این پارادایم به شکل پیش فرض توزیع‌شده و سبک هستند. ٥- امکان نصب چندین نمونه از تنها یک سرویسیا یک مجموعه کوچک از سرویس‌ها ٦- استفاده از لود بالانسر در سمت سرور ۷- استفاده آسان از لود بالانسرهای سمت کلاینت ۸- استفاده از کانتینرهای سبک مانند داکر و همچنین ابزارهای CI و Agile Delivery در پارادیم میکرو سرویس نصب سرویس‌های متعدد به شکل مستقل را آسان کرده است

جدول ٤-٦ ارزیابی و مقایسه کیفی دو معماری موجود و پیشنهادی ازنظر انعطاف‌پذیری:

معیار	انعطاف‌پذیری
معماری موجود	۱- چون در این معماری ابزاری مشابه سرویس دیسکاوری وجود ندارد معمولاً اضافه کردن ماژول جدید علاوه بر توسعه و نصب خود ماژول ممکن است احتیاج به تغییراتی در ESB هم داشته باشد. ۲- اضافه کردن سرویس جدید در ماژول موجود احتیاج به استقرار مجدد ماژول و احتمالاً esb دارد. ۳- استفاده از زبان و سکویی و سامانه عامل متناسب با نوع سرویس امکان‌پذیر است اما با دو پیش‌شرط: a. احتیاج به ارتباط با مؤلفه‌های وابسته به سکویی خاص (مانند EJB) نداشته باشد. b. ممکن است این سرویس لازم باشد به شکل یک ماژول مستقل دیپلوی شود که درواقع در این حالت حرکتی به سمت میکرو سرویس در حال انجام است.

	٤- حذف کردن یک ماژول ممکن است به شرطی که وابستگی محکمی بین این ماژول و ماژول دیگری وجود نداشته باشد. حذف یک سرویس پیچیدگی بیشتر از ماژول دارد.
امتیاز معماری موجود	B
امتیاز معماری پیشنهادی	A+
معماری پیشنهادی	١- افزودن راحت سرویس و ماژول جدید بی‌نیاز به تغییر در هیچ‌یک از سرویس‌های موجود به دلیل ابزارهای سرویس دیسکاوری وapi-gateway ٢- اضافه کردن سرویس فرآیندی جدید بدون نیاز به redeploy تمام سرویس‌های فرآیندی ٣- استفاده از زبان و سکویی و سامانه عامل متناسب با نوع سرویس (مثلاً برای سرویسی که احتیاج پردازش تصویر دارد می‌توان از سی یا پایتون استفاده کرد(٤- هر چه تعداد سرویس‌ها در ماژول‌های قابل‌نصب مستقل کمتر باشد امکان نصب انتخابی سرویس‌ها بیشتر می‌شود. ٥- با توجه به اینکه در پارادیم این معماری توصیه به ارتباط غیر هم‌زمان و مبتنی بر SAF بین سرویس‌ها می‌باشد بالا نبودن دائم یا موقت یک سرویس کارکرد سرویس دیگر را مختل نمی‌کند.

جدول ٤-٧ ارزیابی و مقایسه کیفی دو معماری موجود و پیشنهادی ازنظر کار آیی

معیار	کار آیی
معماری موجود	١- درصورتی‌که در لود معمول نتواند کار آیی مورد انتظار در قرارداد را تأمین کند معمولاً ابتدا با روش‌های افزودن منابع سامانه‌ی سرور و سپس استفاده از روش‌های مقیاس‌پذیری باید این کار آیی تأمین شود. ٢- به دلیل استفاده همه سرویس‌ها از منابع مشترک مانند دیتابیس و حافظه گاها عدم کار آیی مناسب یک سرویس موجب افت کیفیت پاسخگویی همه سرویس‌ها می‌شود.
امتیاز معماری موجود	A
امتیاز معماری پیشنهادی	A
معماری	کار آیی سامانه در این پارادایم تیغ دو لبه است.

پیشنهادی	۱- به دلیل استقلال سرویس‌ها و راحتی مقیاس‌پذیری می‌توان کار آیی موردنظر بر طبق SLA را فراهم کرد و به آن وفادار ماند. همچنین به همین دلیل امکان مدیریت لود بالا فراهم است. ۲- اما از طرفی با توجه به استقلال داده و کانتینر نصب سرویس‌ها ارتباط از طریق حافظه مشترک یا دیتا بیس مشترک در این معماری امکان‌پذیر نیست و ممکن زمان پاسخ یک سرویس در حالت لود معمولی از همان سرویس در معماری موجود کندتر باشد. ۳- همچنین در این معماری برای حفظ کار آیی تنظیم اولیه منابع هر کانتینر و ارتباطات شبکه‌ای بین آن‌ها. ضروری است.

جدول ٤-٨ ارزیابی و مقایسه کیفی دو معماری موجود و پیشنهادی ازنظر استفاده مجدد

معیار	استفاده مجدد
معماری موجود	استفاده مجدد در حد ماژول ممکن است اما در حد فقط یک سرویس به شکل مستقل در یک سامانه دیگر پیچیدگی دارد.
امتیاز معماری موجود	B
امتیاز معماری پیشنهادی	A+
معماری پیشنهادی	استفاده مجدد از سرویس‌های پایه‌ای و بیزینسی در این معماری به دلیل وابستگی سست بین آن‌ها و تمرکز هر سرویس بر وظیفه خاص خودش به‌راحتی ممکن است. تعریف یک سرویس فرآیندی جدید با توجه به ابزار قوی bpms ممکن است به‌صرفه تر از اعمال تغییرات در یک فرآیند موجود باشد.

جدول ٤-٩ ارزیابی و مقایسه کیفی دو معماری موجود و پیشنهادی از داده محوری

معیار	داده محوری
معماری موجود	۱- برای تولید گزارش‌ها پیچیده تحلیلی و آماری ممکن است احتیاج به ابزارهای انبار داده باشد اما بخش زیادی از گزارش‌ها متداول سامانه‌ی از طریق اجرای Query های متداول ممکن است. ۲- به دلیل اینکه قابلیت‌های transactional در این معماری پررنگ‌تر است تأمین data consistency راحت‌تر است
امتیاز معماری موجود	A

امتیاز معماری پیشنهادی	B
معماری پیشنهادی	۱- به دلیل استقلال دیتابیس‌های سرویس‌ها و همچنین تنوع انواع دیتابیس مورداستفاده اجرای بسیاری از Query ها متداول برای تولید گزارش ممکن نیست. ۲- در معماری پیشنهادی لایه‌ای برای تجمیع داده‌ها استفاده‌شده است که از ابزارهایی مانند آپاچی اسپارک برای تجمیع داده‌ها و تولید نتایج تحلیلی در لحظه بهره می‌برد. ۳- به دلیل نبود ماهیت transactional در ارتباط بین سرویس‌ها تأمین سازگاری داده با روش‌های متداول به‌راحتی ممکن نیست و معمولاً به سمت روش‌هایی مانند CQRS و Event Sourcing حرکت می‌کنند.

جدول ٤-١٠ ارزیابی و مقایسه کیفی دو معماری موجود و پیشنهادی ازنظر Omnichannel:

معیار	Omnichannel
معماری موجود	اگر همه سرویس‌ها به شکل صحیح و مطابق قاعده معماری از طریق esb در اختیار کانال‌های مختلف قرار بگیرد دستیابی به omni channel به‌راحتی امکان‌پذیر است.
امتیاز معماری موجود	A
امتیاز معماری پیشنهادی	A+
معماری پیشنهادی	ارائه و اکسپوز سرویس‌های متنوع و جدید برای تمام کانال‌ها ممکن است. به دلایل اتصال همه کانال‌ها به api-gateway تجربه developer ی و کاربری یکسانی به توسعه‌دهنده و کاربر نهایی داده می‌شود.

جدول ٤-١١ ارزیابی و مقایسه کیفی دو معماری موجود و پیشنهادی ازنظر مدیریت فرآیندها

معیار	مدیریت فرآیندها
معماری موجود	از bpms استفاده نمی‌کند و فرآیندها باید به شکل دستی تعریف شوند.
امتیاز معماری موجود	B
امتیاز معماری پیشنهادی	A+

معماری پیشنهادی	در این معماری مجموعه‌ای از سرویس‌های فرآیندی که هر یک نسخه‌ای از bpms است که به شکل مستقل نصب می‌شود توسعه و مقیاس‌پذیری این سرویس‌ها را آسان می‌کند. برای ارائه پنل های مانیتورینگ و نمایش شاخص‌های مجتمع احتیاج به کد نویسی و سفارشی‌سازی دارد.

۵- نتیجه‌گیری

برای سنجش میزان کار آیی سرویس‌دهی رهیافت پیشنهادی، چند نمونه مطالعه موردی و پیاده‌سازی فرآیندی انجام شد که مدل‌های آن نمایش داده شد. برای ارزیابی روش پیشنهادی، سناریوهایی با توجه به نیازمندی‌های حرفه و نیازمندی‌های کیفی در معماری مطرح شدند. سپس روش‌های که برای برآورد کردن ای نیازمندی‌ها در نظر گرفته‌شده بود بیان شدند. در پایان مجموعه از ترکیب لایه مختلف در معماری‌های سرویس گرا روش پیشنهادی شرح داده و برای کار آیی سیستم چند نونه از فرآیند در معماری قدیم و روش پیشنهادی باهم مود بررسی قرار گرفت. البته این نکته حائز اهمیت است بایستی توجه ویژه‌ای به تولید فرآیندها به نحوه پیاده‌سازی روش پیشنهادی در معماری‌های سرویس گرا داشت. درنهایت روش پیشنهادی پویایی لازم و کاهش زمان خدمات‌رسانی و بهبود سرویس‌دهی در این‌گونه معماری‌ها را به ارمغان می‌آورد. همچنین، با توجه به سناریوها و راهکارهای پیشنهادی نشان داده شد دسترسی‌پذیری سرویس‌ها و نیز کار آیی، نگهداری، مقایس پذیری را با استفاده از روش پیشنهادی در معماری‌های سرویس گرا، برآورد می‌نماید.

جدول ۵-۱ ارزیابی و مقایسه کلی دو معماری موجود و پیشنهادی:

	در دسترس پذیری	کار ایی	ارتبا طات چند کانال ه	مقید اس پذیر ی	استف اده مجدد	داد ه محو ری	انع طاف پذیر ی	مدیر یت فرآیند ها
معماری پیشنهادی	A+	A	A+	A+	A+	B	A+	A+
معماری موجود	B	A	B	C	B	A	B	B

مراجع

[1] Szydło, T. and Zieliński, K., "Model Driven Adaptive Quality Control in Service Oriented Architectures," Applications and experiences of quality control, INTECH, pp. 381–396, 2011.
[2] Zieliński, K., Szydło, T., Szymacha, R., Kosinski, J., Kosinska, J. and Jarzab, M., "Adaptive soa solution stack," IEEE Transactions on Services Computing, 99 (PrePrints), 2011.
[3] Boehm, B.W., Software risk management: principles and practices. Software, IEEE, 1991. 8(1): p. 32-41.
[4] Kircher, M. and P. Jain, Pattern-Oriented Software Architecture, Patterns for Resource Management. Vol. 3. 2013: John Wiley & Sons.
[5] Hanmer, R., Pattern-oriented software architecture for dummies. 2012: John Wiley & Sons.
[6] Schmidt, D.C., et al., Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects. Vol. 2. 2013: John Wiley & Sons.
[7] Bird, A., et al., Survey of Architectural Styles.
[8] Fu, Y., M. Li, and F. Chen, Impact propagation and risk assessment of requirement changes for software development projects based on design structure matrix. International Journal of Project Management, 2012. 30(3): p. 363-373.

[9] Medvidovic, N. and R.N. Taylor. Software architecture: foundations, theory, and practice. in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2. 2010. ACM.
[10] Rosen, M., et al., Applied SOA: service-oriented architecture and design strategies. 2012: John Wiley & Sons
[11] Rotem-Gal-Oz, A., E. Bruno, and U. Dahan, SOA patterns. 2012: Manning.
[12] Network, M.D. Chapter 3: Architectural Patterns and Styles. 2014; Available from: http://msdn.microsoft.com/enus/library/ee658117.aspx.
[13] Erl, T., et al., SOA with REST: Principles, Patterns &Constraints for Building Enterprise Solutions with REST. 2012: Prentice Hall Press.
[14] Pautasso, C., RESTful web services: principles, patterns, emerging technologies, in Web Services Foundations. 2014, Springer. p. 31-51.
[15] Gregor Hohpe, Bobby Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. s.l. : AddisonWesley, 2012
[16] Becoming a BPM Professional. [Online] http://www.abpmp.org/displaycommon.cfm?an=1&subarticlenbr=216.
[17] John Jeston, Johan Nelis. Business process management: practical guidelines to successful implementations. s.l. : Routledge, 2012.
[18] Allweyer, Thomas. Bpmn 2.0. s.l. : Books on Demand ,2012
[19] Oliveira, B., Belo, O.: BPMN patterns for ETL conceptual modeling and validation. In:Chen, L., Felfernig, A., Liu, J., Ra, Z. (eds.) Foundations of Intelligent Systems, Lecture Notes in Computer Science, vol. 7661, pp. 445–454. Springer Berlin Heidelberg ,2012
[20] OMG: Business process model and notation (BPMN) version 2.0. Tech. rep,2011
[21] Avram, A., Docker: Automated and Consistent Software Deployments, March 27, 2013, Available:https://www.infoq.com/news/2013/03/Docker.
[22] Balalaie, A., Heydarnoori, A.& Jamshidi, P.,Migrating to CloudNative Architectures Using Microservices: An Experience Report, 2015.
[23] Docker, What is Docker?,Retrieved September 2, 2016. Available: https://www.docker.com/whatdocker,
[24] Fowler, M.,Microservice Trade-offs, July 1, 2015, Available:http://martinfowler.com/articles/microservice-tradeoffs.html#deployment,
[25] Fowler, M., ServiceOrientedAmbiguity, July 1, 2005, Available:http://martinfowler.com/bliki/ServiceOrientedAmbiguity.html
[26] Lauret, A., Writing OpenAPI (Swagger) Specification Tutorial–Part 1 –Introduction, March 2, 2016, Available:https://apihandyman.io/writing-openapi-swagger-specificationtutorial-part-1-introduction.
[27] Miri, I., Microservices vs. SOA ,July 5, 2016. Available:https://dzone.com/articles/microservices-vs-soa-2
[28] Richards, Microservices vs. Service-Oriented Architecture, Sebastopol, USA: O´Reilly Media,2016.
[29] Richardson, C., Service Discovery in a Microservices Architecture, October 12, 2015 Available:https://www.nginx.com/blog/servicediscovery-in-a-microservices-architecture,
[30] Miina Koskinen, MICROSERVICES AND. CONTAINERS, Aug 23, 2016 .
[31] Rajasekar, A., Wan, M., Moore, R., & Schroeder, W ,Micro-Services: A Service-Oriented Paradigm for. Data Intensive Distributed Computing. In Challenges and Solutions for Large-scale Information Management 2012.
[32] Singer, R , Agent-Based Business Process Modeling and Execution: Steps Towards a Compiler-Virtual Machine Architecture. In Proceedings of the 8th International Conference on Subject-oriented Business Process Management,2016.
[33] IBM Business Process Manager V8.5.6 documentation. Retrieved May 7, 2017 from: http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.6/com.ibm.wb pm.wid.bpel.doc/topics/cprocess_transaction_micro.htm
[34] Lanthaler, M., & Gütl, C.On using JSON-LD to create evolvable RESTful services. In Proceedings of the Third International Workshop on RESTful Design 2012
[35] Sam Newman.Building Microservices. "O'Reilly Media, Inc.", 2015
[36] Soa,Fabrizio Montesi, Claudio Guidi, and Gianluigi Zavattaro. Service-Oriented Programming with Jolie. In Web ServicesFoundations, pages 81–107. Springer, 2014.
[37] OASIS. Web Services Business Process Execution Language. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.
[38] Open Service Oriented Architecture. Service Component Architecture Transaction Policy Specification, (SCA version 1.00). http://www.osoa. org/download/attachments/35/SCA_TransactionPolicy_V1.0.pdf, December 3, 2007
[39] Open Service Oriented Architecture. Service Component Architecture Final Version 1.0 Specifications.

http://www.osoa.org/display/Main/ Service+Component+Architecture+Specifications, 2007-2009.
[40] World of Web Services, IBM Forum, www.IBM.com
[41] Oracle Corporation, Available: www.oracle.com
[42] Forrester Research, Inc. Available: www.forrester.com
[43] Scalability:http://www.linfo.org/scalable.html
[44] Scalability:https://www.techopedia.com/definition/9269/scalability
[45] Flexibility: https://informativearchitecture.wordpress.com/2011/10/04/softwareflexibility/
[46] Performance: https://en.m.wikipedia.org/wiki/Computer_performance
[47] Reusability: http://journals.ecs.soton.ac.uk/java/tutorial/beans/whatis/softwarecomponents.html

[48] Bc. Jiří Mach, Comparison of BPM Suites and Their Application in Enterprise Architecture, 2012