ارائه یک سازنده کد ترکیبی و توسعه آن در قالب قابلیت‌های کاربردی با رهیافت معماری سرویس‌گرا

عرفان قندهاری[۱]، فاطمه سعادت‌جو[۲،*] و سید محمد صالح معافی[۳]

اطلاعات مقاله

دریافت مقاله: ۹۵/۰۱/۱۱
پذیرش مقاله: ۹۶/۰۷/۰۵

واژگان کلیدی:
معماری سرویس‌گرا،
سازنده کد،
توسعه نرم‌افزار،
وب‌سرویس،
SOG.

چکیده

معماری سرویس‌گرا یکی از زمینه‌های در حال توسعه مهندسی نرم‌افزار است که امکان برقراری ارتباط بین دو سیستم در یک شبکه ناهمگون با تکنولوژی‌های مختلف را امکان‌پذیر می‌سازد. ویژگی‌های این معماری شامل ارتباطات ساده، قابلیت استفاده مجدد، ساده‌سازی یک راهکار پیچیده در یک قالب ساده و چابکی در مقابل تغییرات آینده است. در بحث معماری سرویس‌گرا یکی از چالش‌های مهم و اساسی این است که علی‌رغم سازنده‌های کد متعددی که در این حیطه معرفی شده، هنوز کاستی‌هایی در فازهای مختلف این سازنده کدها وجود دارد. در این مقاله با توجه به نقاط ضعف سازنده‌های کد در فازهای مختلف و تأثیر به‌سزایی که این کاستی‌ها، می‌تواند در کیفیت، کارایی و تولید سیستم‌های نرم‌افزاری داشته باشد به ارائه راهکار ترکیبی برای معرفی سازنده کد از دو مرجع خواهیم پرداخت. بدین منظور با استفاده از مزایای دو سازنده کد فوق، یک سازنده کد ترکیبی (SOG) ارائه می‌گردد. همچنین به منظور افزایش کیفیت و کارایی در تولید سیستم‌های نرم‌افزاری با استفاده از سازنده کد ارائه شده، ویژگی‌های جدید در راستای افزایش امنیت کد تولید شده، پیش‌بینی سطح دسترسی، افزایش زبان‌های برنامه‌نویسی قابل پشتیبانی، تعدد قالب‌های انتقال اطلاعات بین کلاینت و سرور، اضافه شدن لایه رمزنگاری و تولید پنل مخصوص مدیریت پایگاه داده در وب ارائه شده است. بنابراین به طور خلاصه می‌توان گفت هدف اصلی در این مقاله ارائه سازنده کدی است که کدهای مربوط به معماری سرویس‌گرا را تولید می‌کند که علاوه بر داشتن ویژگی‌های سازنده کدهای مشابه دارای ویژگی‌های کاربردی نوینی می‌باشد. به منظور سنجش توانایی سازنده کد پیشنهادی، این سازنده کد بر اساس معیارهای مربوط به پروتکل‌های مختلف، پیش‌بینی امنیت در کد، پیش‌بینی سطح دسترسی در کد تولید شده، قالب‌های انتقال اطلاعات بین کلاینت و سرور و موارد مشابه در زمینه سازنده کد، مورد ارزیابی قرار گرفته است.

۱- مقدمه

معماری سرویس‌گرا شامل سیاست‌ها، تجارب و چارچوب‌هایی است که کارکردهای سیستمی را قادر می‌سازد به‌صورت مجموعه‌ای از سرویس‌های توزیع‌شده در اندازه‌های مختلف تعریف شوند. این سرویس‌ها با کمک تعریف یک واسط استاندارد از پیاده‌سازی مجزا می‌شوند [۱]. معماری سرویس‌گرا به منظور پاسخ‌گویی به نیازهای اساسی مهندسی نرم‌افزار مطرح شد [۲]. هدف اصلی

* پست الکترونیک نویسنده مسئول: saadatjou@sau.ac.ir
۱. کارشناسی ارشد، دانشکده مهندسی گروه کامپیوتر، دانشگاه علم و هنر یزد
۲. استادیار، دانشکده مهندسی گروه کامپیوتر، دانشگاه علم و هنر یزد
۳. کارشناسی ارشد، دانشکده مهندسی گروه کامپیوتر، دانشگاه علم و هنر یزد

معماری سرویس‌گرا ایجاد ساختار و امکاناتی برای ارائه سرویس‌هایی با امنیت و کارایی بالا است [۳]. از آن‌جا که سایر معماری‌های ارائه شده قادر به چابک‌سازی سیستم‌ها نبودند، نیاز مبرم به معماری جدید و هماهنگ با دیدگاه چابک به وجود آمد که باعث توسعه معماری سرویس‌گرا گردید [۴]. در بحث معماری سرویس‌گرا یکی از چالش‌های مهم و اساسی فقدان سازنده کدهای کامل، برای ایجاد سیستم‌های نرم‌افزاری می‌باشد. گرچه تعداد زیادی سازنده کد پیشنهاد شده است، با این حال تعداد کمی از آن‌ها به طور کامل همه فعالیت‌ها را پوشش می‌دهند و هیچ‌کدام از آن‌ها به‌طور کامل، نیازمندی‌های توسعه سیستم‌های نرم-افزاری را پشتیبانی نمی‌کنند. بنابراین در حال حاضر نیاز به توسعه سازنده کدها به منظور رسیدن به سازنده کدهای یکپارچه و جامع، کاملا ضروری به نظر می‌رسد [۹-۵]. در ادامه کارهای صورت گرفته در توسعه سازنده کدها که بیشترین اهمیت را در بین توسعه‌های صورت گرفته دارند را مورد بررسی قرار خواهیم داد.

Fertalej و همکاران سازنده کد خود را با عنوان یک سازنده کد منبع بر پایه مشخصات UML[1] ارائه نمودند. در این پژوهش سازنده کد ارائه شده با استفاده از نمودارهای UML مشخصات نرم‌افزاری را دریافت و کدهایی را به زبان‌های برنامه‌نویسی مختلف تولید می‌کند [۱۰]. Lazetic و همکاران نیز در سال ۲۰۱۲ یک سازنده برنامه-های تحت وب MVC[2] را ارائه نمودند که این سازنده کد ارائه شده بر پایه معماری سه لایه یک برنامه کاربردی تحت وب را تولید می‌کرد [۱۱]. توسعه دیگری که بر روی سازنده کد صورت گرفت توسط Imam و همکاران انجام شد که با استفاده از قوانین و تکنیک‌های دانش، سازنده کدی را ارائه نمودند که قابلیت استفاده در ساخت نرم‌افزارهایی با ساختار ثابت را دارا بود [۱۲]. Ries و همکاران نیز در قالب چارچوب محاسبات شبکه و همچنین رهیافت ساخت کد از مدل UML بهبودهایی را در زمینه سازنده کد به‌وجود آوردند [۱۳]. ارائه سازنده کد برای بهینه‌سازی Convex جاسازی شده در سال ۲۰۱۱ توسط Mattingley و همکارانش انجام شد. این ابزار توضیحات مسئله بهینه‌سازی مورد نظر از خانواده Convex را دریافت و کد قابل اعتمادی را به زبان برنامه‌نویسی C تولید می‌نماید [۱۴]. Siret و همکاران سازنده کد سخت‌افزار را از برنامه مبتنی بر جریان داده در سال ۲۰۱۰ ارائه نمودند. این سازنده کد، یک کد سخت‌افزاری با کیفیت را، به‌صورت سلسله مراتبی از یک برنامه مبتنی بر جریان داده تولید می‌کند [۱۵]. کارهای توسعه سازنده‌های کد، تنها به موارد گفته شده محدود نبوده است و کارهای توسعه‌ای بر روی سازنده‌های کد مبتنی بر معماری سرویس‌گرا نیز ادامه دارد. در این راستا Shan و همکاران سازنده کد خود را برای سیستم‌های اطلاعاتی نوشته شده، به زبان جاوا با معماری چهار لایه شامل Client، Business Layer، Service Layer و Database Layer ارائه نمودند. این سازنده کد قادر به ساخت همه توابع اساسی کار با پایگاه‌داده از جمله درج، حذف، جستجو و به‌روزرسانی بود. همچنین به علت استفاده از زبان اجرای فرآیندها برای مدل‌سازی فرآیند کسب و کار سیستم، انعطاف‌پذیری خوبی را در مقابل تغییرات دارد [۱۶]. Boian و همکاران سازنده کد خود را با عنوان WSSRAPPER ارائه نمودند که این سازنده کد با سه رویکرد، وب‌سرویس‌های استاندارد شده توسط کنسرسیوم جهانی وب و همچنین وب‌سرویس‌های مبتنی بر پروتکل معماری REST و نیز XML RPC کد مورد نظر را تولید می‌کند. همچنین قابلیت ساخت کد با چهار زبان PHP، Java، Python و #C را دارا می‌باشد [۱۷].

آن‌چه در توسعه این سازنده‌های کد و کارهای توسعه‌ای دیگر به عنوان چالش محسوب می‌شود این است که هنوز سازنده‌های کد توسعه‌یافته نمی‌توانند همه فعالیت‌های ایجاد سیستم‌های نرم‌افزاری را پوشش دهند و به منظور توسعه نسل بعدی سازنده‌های کد لازم است که کارهای تحقیقاتی بیشتری به منظور توسعه و بهبود سازنده‌های کد صورت گیرد تا در خلال آن بتوان با ایجاد همگرایی بیشتر در فازهای مختلف سازنده‌های کد، امکان تطبیق این سازنده‌های کد با سیستم‌های نرم‌افزاری را افزایش داد. با توجه به این چالش‌ها در این مقاله سعی می‌گردد ویژگی-های نوینی در راستای افزایش امنیت کد، پیش‌بینی سطح دسترسی، افزایش زبان‌های برنامه‌نویسی قابل پشتیبانی، تعدد قالب‌های انتقال اطلاعات بین کلاینت و سرور، اضافه شدن لایه رمزنگاری و تولید پنل مخصوص مدیریت پایگاه-داده در وب به سازنده کد ترکیبی به‌دست آمده اضافه گردد.

[1] Unified Modeling Language

[2] Model View Controller

این ویژگی‌ها به میزان قابل توجهی توانایی‌های سازنده کد پیشنهادی که از ترکیب سازنده کدهای [۱۶][۱۷] به‌دست می‌آید را افزایش خواهد داد و زمینه را برای توسعه نسل بعدی سازنده‌های کد فراهم می‌آورد.

در ادامه به منظور تفکیک رویکردهای شی‌گرا و سرویس‌گرا به بررسی تفاوت‌های سرویس‌ها و اشیا خواهیم پرداخت. سپس در بخش سوم به بررسی فازهای مختلف سازنده‌های کد سرویس‌گرا اشاره خواهیم نمود. در بخش چهارم به معرفی سازنده کد پیشنهادی و تشریح ویژگی‌های آن می‌پردازیم. در بخش پنجم به مقایسه سازنده کد بر اساس معیارهای پشتیبانی از پروتکل‌های مختلف، پیش‌بینی امنیت در کد، پیش‌بینی سطح دسترسی در کد تولید شده، قالب‌های انتقال اطلاعات بین کلاینت و سرور و موارد مشابه خواهیم پرداخت و در نهایت در بخش ششم به نتیجه‌گیری و راهکارهای آینده اشاره خواهیم نمود.

۲- مقایسه رویکرد شی‌گرا و سرویس‌گرا

معماری سرویس‌گرا تکاملی از معماری شی‌گرا است. به عبارت دیگر می‌توان گفت سرویس‌ها شباهت‌هایی با اشیا دارند [۱۸]. Hezavehi و همکاران در سال ۲۰۱۲ [۱۹] به مقایسه معماری سرویس‌گرا و شی‌گرا در پنج سطح قابلیت تغییر و قابلیت اصلاح، قابلیت استفاده مجدد، امنیت، کارایی و قابلیت اطمینان پرداختند که در ادامه برخی از تفاوت‌ها بیان شده است. الف) معماری سرویس‌گرا از سیستم‌های سازگار در محیط‌های ناهمگون و در حال تغییر به خوبی پشتیبانی می‌کند، در حالی که اعمال تغییر در نرم‌افزارهای مبتنی بر معماری شی‌گرا به سهولت صورت نمی‌گیرد. ب) با توجه به این‌که سرویس‌ها به کمک یک واسط استاندارد و مستقل تعریف می‌شوند، به راحتی می‌توان از آن‌ها در نرم‌افزارهای دیگر استفاده نمود، در حالی که در سیستم‌های شی‌گرا، با توجه به این‌که واحدهای نرم‌افزاری به‌صورت مجدد مورد استفاده قرار می‌گیرند و ممکن است اندازه‌های متفاوتی داشته باشند، این امکان وجود ندارد. پ) به‌طور کلی در سیستم‌های مبتنی بر سرویس، مشخصات امنیتی استفاده از سرویس، در توافق‌نامه بین سرویس‌دهنده و سرویس‌گیرنده بیان شده است. این مشخصات شامل محدودیت در دسترسی، جلوگیری از دست‌کاری داده‌ها، استراق سمع[1]، رمزگذاری، محرومیت از سرویس[2]، انطباق با استانداردهای امنیتی و پشتیبانی است. در حالی که در سیستم‌های شی‌گرا یکی از چالش‌های مهم، حفظ امنیت در پایگاه‌داده‌ها است. ت) در سیستم‌های سرویس‌گرا، کل زمان پردازش یک سرویس درخواست شده، از زمانی که درخواست ارسال می‌شود، تا زمانی که سرویس ارائه می‌گردد به عنوان معیاری مناسب برای ارزیابی کارایی سیستم در نظر گرفته می‌شود، در حالی که در سیستم‌های شی‌گرا ممکن است مشکلات مربوط به کارایی حتی زمانی که پیاده‌سازی انجام شده است، باز هم حل نشده باقی بماند چرا که دارای تعریف روشنی نیست. ث) در رویکرد سرویس‌گرا، سرویس‌ها از طریق پروتکل‌های ارتباطی با یکدیگر تعامل برقرار می‌کنند و در واقع همین برقراری تعامل می‌تواند یکی از نقاط شکست سیستم محسوب شود، در حالی که در رویکرد شی‌گرا از متد ارث‌بری استفاده می‌شود که منجر به کاهش خطا و تلاش برای رفع آن‌ها می‌شود. البته با توجه به این‌که سرویس‌ها به نوعی از اشیا مشتق شده‌اند، می‌توان گفت شباهت‌هایی نیز بین سرویس‌ها و اشیا وجود دارد. با وجود این تفاوت‌ها و شباهت‌ها می‌توان پارامترهایی از هر دو رویکرد را به هم نگاشت کرد، جدول ۱ یک نگاشت نوعی بین رویکرد شی‌گرا و سرویس‌گرا را نشان می‌دهد.

جدول۱- نگاشت بین رویکردهای شی‌گرا و سرویس‌گرا [۱۹]

رویکرد شی‌گرا	رویکرد سرویس‌گرا
کلاس انتزاعی	نقش کلی[3]
کلاس	نقش مخصوص یک قلمرو[4]
متد	قابلیت
نمونه‌سازی	سرویس خاص
فرخوانی متد	تبادل پیغام

بر اساس جدول ۱ می‌توان گفت رویکرد سرویس‌گرا برای بیشتر قابلیت‌های رویکرد شی‌گرا راهکار ارائه نموده است، این راهکار برای تحلیل و طراحی سیستم‌های مبتنی بر سرویس مناسب هستند.

۳- بررسی و مقایسه فازهای مختلف سازنده‌های کد سرویس‌گرا

در این قسمت قصد داریم به معرفی و مقایسه کارهای مهم

[1] Eavesdropping
[2] Service Deprivation
[3] Generic Role
[4] Domain-Specific Role

در زمینه سازنده کد سرویس‌گرا، که از آن‌ها در سازنده کد ترکیبی استفاده شده است بپردازیم. از بین کارهای انجام شده در زمینه سازنده کد، که قسمتی از آن‌ها را برای شناخت حیطه فعالیت سازنده کد بررسی کردیم، دو رویکرد با در نظر گرفتن بیشترین جنبه از فعالیت‌های سازنده کد در زمینه سرویس‌گرایی ارائه شده است، که باعث تمایز آن-ها از سایر کارهای انجام شده، می‌باشد. به همین دلیل این دو رویکرد به عنوان مرجع برای رسیدن به سازنده کد ترکیبی انتخاب شده‌اند. رویکرد اول توسط Shan و همکارانش با عنوان " توسعه سازنده کد برای سیستم‌های اطلاعاتی بر پایه معماری سرویس‌گرا" در سال ۲۰۰۹ [۱۶] و رویکرد دوم توسط Boian و همکاران با عنوان "یک سازنده وب سرویس جهانی" در سال ۲۰۱۰ [۱۷] انجام شده است. در ادامه این بخش، ویژگی‌های هر یک از آن‌ها را شرح داده و نقاط قوت و ضعف آن‌ها را مورد بحث و بررسی قرار خواهیم داد.

- کارهای انجام شده
- تحقیق و توسعه سازنده کد برای سیستم‌های اطلاعاتی بر پایه معماری سرویس‌گرا

Shan و همکاران، سازنده کد خود را بر پایه استاندارد تعریف شده در کنسرسیوم جهانی وب[1] برای وب سرویس‌ها تعریف کرده‌اند، که وب سرویس‌های ساخته شده ابتدا توسط زبان اجرای فرآیندها[2] در نمودارهای جریان کار مدل می‌شوند، و سپس سازنده کد برای سیستم‌های اطلاعاتی نوشته شده با زبان جاوا با معماری شکل (۱) عمل می‌کند. از جمله موارد و ویژگی‌های این سازنده کد می‌توان به ساخت همه توابع اساسی کار با پایگاه داده از جمله درج، حذف، جستجو و به‌روزرسانی اشاره نمود. همچنین به علت استفاده از زبان اجرای فرآیندها برای مدل سازی فرآیند کسب و کار سیستم، می‌توان انعطاف پذیری خوبی را در مقابل تغییرات از سیستم انتظار داشت.

سازنده کد Shan را می‌توان مختص به سیستم‌های اطلاعاتی با معماری مشخص و مدل شده در قالب BEPL دانست که با توجه به این موضوع، این سازنده کد به‌صورت بسیار محدود و برای سیستم‌هایی با ویژگی‌های ذکر شده قابل استفاده است هرچند اختصاص یک سازنده کد به یک نوع نرم‌افزار خاص با معماری خاص می‌تواند قابلیت‌های آن سازنده کد را برای کارکرد در معماری‌های دیگر کم‌رنگ کند اما می‌توان این انتظار را داشت که این سازنده کد برای معماری هدف خود، بسیار دقیق و بهینه عمل نماید.

از این رو است که هر نرم‌افزار در طول چرخه حیات خود به آزمودن نیز احتیاج دارد و برای مشخص شدن وضعیت سیستم در حالات مختلف به نظارت نیاز پیدا می‌کند. این سازنده کد علاوه بر کدهای مربوط به کار با پایگاه داده برای آزمودن، در حیطه نظارت و رابط کاربری نرم‌افزار نیز کدهایی را تولید می‌کند که جزئی از ویژگی‌های این سازنده کد به شمار می‌رود.

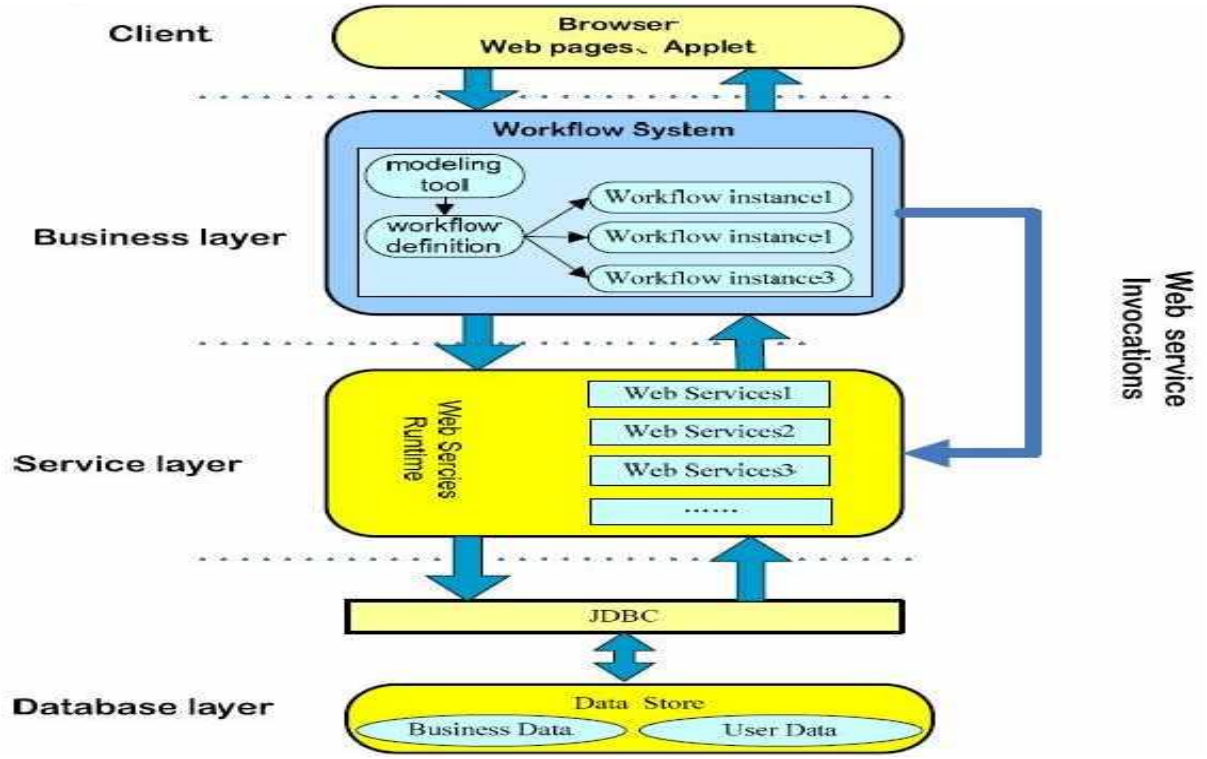
- **یک سازنده وب سرویس جهانی**

Boian و همکاران سازنده کد خود را با عنوان WSWRAPPER ارائه دادند که این سازنده کد با سه رویکرد وب سرویس‌های استاندارد شده توسط کنسرسیوم جهانی وب و همچنین وب سرویس‌های مبتنی بر REST و نیز XML RPC کد مورد نظر را تولید می‌کند. همچنین قابلیت ساخت کد با چهار زبان Python, Java, PHP و C# را دارا است، که این قابلیت‌ها با توجه به هدف وی برای ساخت یک سازنده کد جهانی و همه جانبه است. اما برخلاف سازنده کد قبلی، توجهی به ایجاد کد آزمودن و نظارت نداشته است.

یکی از خواصی که در سازنده کد Boian وجود دارد استفاده از این سازنده کد به صورت کتابخانه در برنامه‌های کاربردی دیگر است که این کار می‌تواند باعث استفاده از این سازنده کد در مراحل اجرای یک نرم افزار شود در حالی‌که استفاده از سازنده‌کد در اغلب اوقات توسط برنامه-نویسان و در طول طراحی و توسعه نرم‌افزار استفاده می‌شود. از آن‌جا که Boian در سازنده کد خود جنبه‌های بسیار مختلفی از یکپارچگی را در نظر گرفته است، می‌توان کار او را به عنوان یک شروع برای رسیدن به یک زبان برنامه نویسی و یا چارچوب سرویس‌گرا دانست تا آن‌جا که بر روی تعریف انواع داده‌های مختلف مانند عدد صحیح و رشته‌ها نیز پیش رفته است، هرچند برای رسیدن به چنین زبان برنامه نویسی یا چارچوبی هنوز موارد زیاد دیگری نیز بایستی در نظر گرفته شود.

[1] W3C

[2] BPEL4WS



شکل ۱- معماری سازنده کد Shan و همکاران

- **مقایسه کارهای انجام شده**

در این قسمت به مقایسه کارهای Shan و Boian از دیدگاه انتقال داده، پروتکل سرویس‌ها، ساخت کدهای جانبی و زبان‌های برنامه‌نویسی پشتیبانی شده خواهیم پرداخت و نقاط قوت و ضعف هر یک را مورد بررسی قرار می‌دهیم.

- **انتقال داده**

در سازنده کد Shan انتقال اطلاعات بر پایه استاندارد SOAP صورت می‌گیرد که نوعی XML محسوب می‌شود اما Boian در سازنده کد خود انواع داده‌ای مختلفی را تعریف کرده است نظیر WSInt,WSString و WSArray که با توجه به تعاریف صورت گرفته انتقال داده‌ها در سازنده کد او بر پایه این انواع صورت می‌گیرد. با توجه به سازنده کد Boian و رویکردهای SOAP و REST می‌توان قابلیت انعطاف‌پذیری در فرمت‌های انتقال اطلاعات را برای این سازنده کد مناسب‌تر ارزیابی کرد هرچند که سازنده کد Shan هدف اصلی یعنی ساخت سیستم‌های اطلاعاتی را برآورده می‌سازد. اما در مقام مقایسه، سیستم Boian را می‌توان انعطاف پذیرتر ارزیابی نمود.

در مجموع با توجه به این‌که SOAP یک استاندارد انتقال اطلاعات مخصوص به خود را دارد، ممکن است اطلاعات دریافت شده از یک سرویس، توسط درخواست کننده آن نیازمند فرمت خاصی باشد که در این صورت بهتر است که تعدادی از این فرمت‌ها توسط سازنده کد قابل ساخت باشند بدین معنی که سرویس ساخته شده قابلیت ارسال اطلاعات در فرمت‌های مختلفی را داشته باشد، به عنوان مثال می‌تواند فرمت‌های XML, JSON, Simple Text را پوشش دهد.

- **پروتکل سرویس‌ها**

shan سازنده کد خود را فقط بر پایه وب سرویس‌هایی با پروتکل SOAP طراحی کرده است در حالی که Boian در سازنده کد خود از سه روش XML RPC, SOAP, REST استفاده نموده است، هرچند که SOAP به تنهایی برای داشتن یک سیستم با معماری سرویس‌گرا کافی است اما پشتیبانی از روش‌های دیگر باعث یک جامعیت نسبی در سازنده کد Boian شده است.

- **ساخت کدهای جانبی**

هر سازنده کد با توجه به استفاده و قالبی که دارد می‌تواند دارای ویژگی‌های منحصر به فردی برای تولید کد نهایی باشد و همچنین می‌تواند کدهایی به‌جز کدهای مورد نیاز برای اجرا در برنامه اصلی را تولید نماید که کدهای اضافه نامیده می‌شود. کدهای اضافه بیشتر مربوط به رفتارهای

غیرعملکردی و رفتارهای کیفی سیستم می‌شوند البته در بسیاری از مواقع بدون وجود این‌گونه کدها سیستم بلااستفاده بوده و یا استفاده از آن مضر است، به عنوان مثال نبود امنیت در یک سیستم جامعه مجازی ممکن است موجب دسترسی به اطلاعات شخصی کاربران شده و عواقب جبران ناپذیری را به همراه داشته باشد. یک سازنده کد با پشتیبانی و در نظر گرفتن تولید خودکار این‌گونه کدهای اضافه می‌تواند کمک به‌سزایی در تولید سیستم نهایی نماید. shan در سازنده کد خود مواردی مانند تولید کد برای نظارت و آزمودن سیستم را در نظر گرفته است و این در حالی است که Boian در رابطه با ساخت کدهای اضافه موردی را ارائه نکرده است. اگرچه Shan در این رابطه کار مناسبی را برای ساخت کد آزمودن و نظارت سیستم ارائه کرده است اما بحثی که بیشتر در رابطه با سیستم‌های مبتنی بر معماری سرویس‌گرا مهم است بحث امنیت و سطح دسترسی به سرویس‌های ارائه شده در سیستم است، در سیستم‌های سرویس‌گرا به علت دسترسی همگان به URL هر سرویس، این دسترسی بایستی در سیستم محدود شود و البته این مخصوص سرویس‌هایی است که به همگان ارائه نمی‌شوند اما در سیستم‌هایی که به صورت عمومی ارائه می‌شوند نیازی به تعریف سطح دسترسی نیست. از آن‌جا که در اکثر سیستم‌ها نیاز به اعمال سطح دسترسی وجود دارد، مناسب است که سازنده کد، راهکاری را برای تولید کدهای مربوط به سطح دسترسی در نظر بگیرد و اجازه دسترسی به هر سرویس را برای افراد غیرمجاز محدود کند. این ویژگی در سازنده کد ارائه شده در این مقاله در نظر گرفته شده است.

- **زبان‌های برنامه نویسی پشتیبانی شده**

در سازنده کد Shan فقط ساخت کد با زبان Java پیش بینی شده است، اگرچه حیطه کاری این زبان زیاد است و برای تولید بسیاری از سیستم‌ها مناسب است اما محدوده کار این سازنده کد را، برای تولید سیستم‌های با زبانی به‌جز Java محدود می‌کند اما در سازنده کد ارائه شده توسط Boian پشتیبانی از زبان‌های برنامه‌نویسی PHP، Java Python و C# در نظر گرفته شده است و از این حیث سازنده کد ارائه شده برای استفاده در تولید سیستم‌های با زبان‌های برنامه‌نویسی بیشتری قابل استفاده است.

- **جمع‌بندی کلی از مقایسه دو سازنده کد**

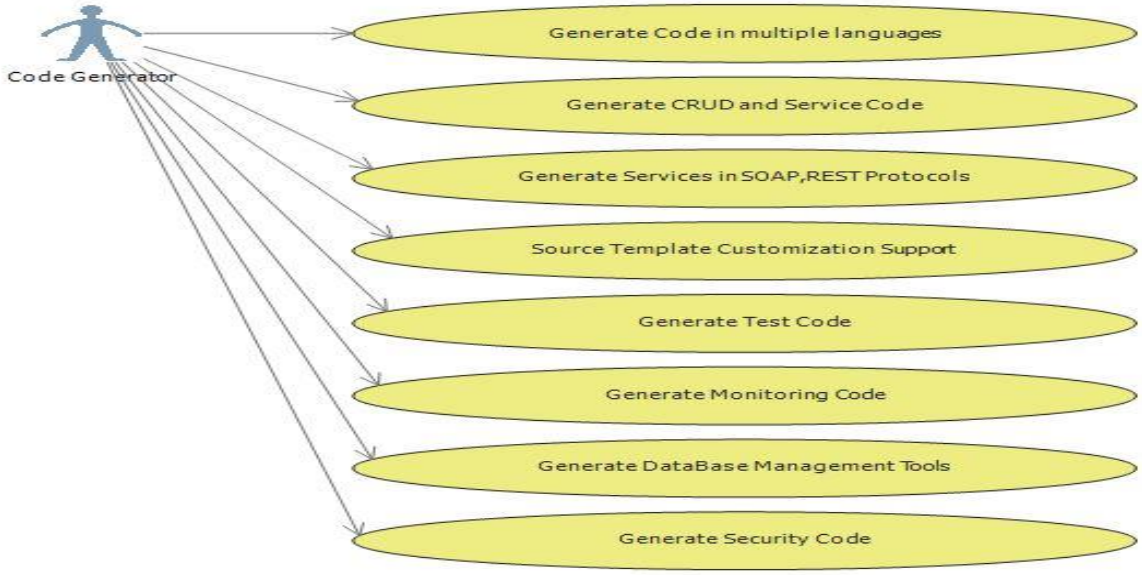
با در نظر گرفتن کارهای Shan و Boian در صورتی که هدف، استفاده از زبان Java و پروتکل SOAP برای ایجاد یک سیستم اطلاعاتی باشد می‌توان سازنده کد Shan را یک کاندید اصلی دانست، در غیر این صورت نمی‌توانیم از این سازنده کد استفاده کنیم. اما سازنده کد Boian با توجه به گستردگی بسیار بیشتر از زبان‌های برنامه‌نویسی(PHP، Java، Python، C#) و همچنین چند پروتکل سرویس-گرایی مختلف(SOAP, REST,XML RPC) قابلیت بسیار بیشتری نسبت به سازنده کد Shan دارد و می‌توان از آن در دایره وسیع‌تری از سیستم‌ها استفاده نمود.

۴- معرفی سازنده کد ترکیبی پیشنهادی (SOG[10])

SOG یک سازنده کد توسعه نرم‌افزاری سرویس‌گرا است که به‌صورت ترکیبی از دو سازنده کد Shan و Boian و همچنین قابلیت‌های پیشنهادی در راستای افزایش امنیت کد تولید شده، پیش‌بینی سطح دسترسی، افزایش زبان‌های برنامه‌نویسی قابل پشتیبانی، تعدد قالب‌های انتقال اطلاعات بین کلاینت و سرور، لایه رمزنگاری و تولید پنل مخصوص مدیریت پایگاه‌داده در وب به دست آمده است. در بهبود و توسعه SOG دو ایده اصلی مورد استفاده قرار گرفته است که عبارتند از: الف) تصور و درک از سرویس و همچنین امنیت، اهداف، برنامه‌ریزی‌ها و موارد مشابه در قابلیت‌های سازنده کد از تحلیل اولیه تا طراحی مورد استفاده قرار گرفته است. ب) یکی از خواسته‌های اساسی که در سازنده کد SOG مورد توجه قرار گرفته است، ساخت لایه دسترسی به پایگاه داده است که، این لایه با تغییر در قالب کد منبع این سازنده کد قابلیت ساخت کد به هفت زبان برنامه‌نویسی از جمله: c#, java, PHP, python, c++, visual basic, Delphi را داراست. استفاده از چنین ساختاری در سازنده کد SOG باعث افزایش چشم‌گیر قابلیت انعطاف آن شده است بدین معنی که استفاده‌کننده می‌تواند لایه دسترسی به پایگاه داده را به‌گونه‌ای که خودش در نظر دارد با استفاده از این سازنده کد تولید نماید و SOG به‌هیچ‌وجه استفاده‌کننده را مجبور به استفاده از

[10] Service Oriented Generator

ساختارهای خاص و غیرقابل تغییر نمی‌کند و از آن‌جا که در نحوه کار با پایگاه داده سلایق متفاوتی وجود دارد و به شیوه‌های گوناگونی این امر صورت می‌گیرد، وجود قابلیت انعطاف زیاد سازنده کد در این قسمت بسیار مورد نیاز است و وجود این امکان در SOG یکی از برتری‌های مهم آن نسبت به سازنده کدهای ارائه شده در حیطه سرویس‌گرا است. تحلیل، طراحی و پیاده‌سازی سرویس‌گرا مجموعه زیادی از مفاهیم را شامل می‌شود و این موضوع فهم همه جنبه‌های تحلیل، طراحی و پیاده‌سازی را از دیدگاه خاص مشکل می‌سازد، به همین دلیل در سازنده کد SOG، تعدادی قابلیت که روی جنبه‌های مختلف تأکید دارند، تعریف شده است. این قابلیت‌ها جنبه‌های مختلفی را در برمی‌گیرند، اما به تنهایی کامل نیستند، بنابراین با کنار هم قرار دادن آن‌ها یک دیدگاه کامل و قابل فهم نسبت به سازنده کد حاصل می‌شود. در انتخاب قابلیت‌های سازنده کد SOG از یک چارچوب ارزیابی که جنبه‌های مختلف یک سازنده کد را پوشش می‌دهد استفاده شده است. بر اساس این چارچوب و پارامترهای آن میزان پشتیبانی هر یک از سازنده‌های کد مورد بررسی، ارزیابی شده است. نتایج ارزیابی نشان می‌دهد که قابلیت‌های اضافه شده به SOG مفاهیم سرویس‌گرایی را تا حد زیادی پوشش می‌دهد و این قابلیت‌ها هیچ‌گونه هم پوشانی با یکدیگر ندارد. با توجه به مواردی که بیان شد مدل UML سازنده کد SOG در قالب شکل (۲) نشان داده شده است.



شکل ۲- خواسته‌های سازنده کد ارائه شده در قالب UML

در ادامه بر اساس مشخصات نرم‌افزاری مشخص شده در شکل (۲) به تشریح ویژگی‌ها و قابلیت‌های مختلف سازنده کد SOG خواهیم پرداخت. بدین منظور هر کدام از فازهای سازنده کد SOG تشریح می‌گردد.

برای ساخت لایه دسترسی به پایگاه داده در سازنده کدSOG، زیرسیستمی با نام crudgenerator در نظر گرفته شده است. این زیرسیستم شامل قطعات مختلفی بوده و همچنین خواسته‌های خاصی را در بر می‌گیرد. در ادامه در مورد بخش‌های مختلف این زیرسیستم و همچنین کد لایه بحث خواهیم کرد.

در ابتدا بهتر است در مورد امکان ساخت لایه دسترسی به پایگاه داده به هفت زبان برنامه‌نویسی بحث کنیم. همان‌گونه که قبلاً گفته شد وجود قابلیت ساخت کد به زبان‌های برنامه‌نویسی مختلف قابلیت انعطاف سازنده کد را افزایش می‌دهد و استفاده از آن را، در طیف وسیعی از سیستم‌ها ممکن می‌سازد. از این رو در سازنده کد ارائه شده این نیاز به‌عنوان یکی از نیازهای سیستم به شمار رفته و برآورده شدن آن بر عهده زیرسیستم crudgenerator گذاشته شده است.

اما در مورد ساخت توابع درج، حذف، به‌روزرسانی و دریافت داده از پایگاه داده نیز باید گفت که به‌طور معمول هر پایگاه داده شامل جداولی است و هر جدول دارای ستون‌هایی با نام‌های مختلف، که می‌تواند دارای صفر تا تعداد زیادی رکورد باشد. در لایه دسترسی به پایگاه داده قابلیت درج،

حذف، به‌روزرسانی و دریافت اطلاعات از هر جدول بایستی به زبان برنامه‌نویسی مورد استفاده در سیستم، کدنویسی شود که در سازنده کدSOG، این کدها به‌صورت خودکار توسط این زیرسیستم نوشته می‌شوند.

این زیرسیستم برای رسیدن به خواسته‌های خود نیازمند اطلاعاتی در مورد ساختار پایگاه داده و زبان برنامه‌نویسی هدف، است.

اطلاعات مربوط به ساختار پایگاه داده به شرح زیر هستند:

- تعداد و نام جدول
- تعداد، نام و نوع ستون‌های هر جدول

بنابراین این زیرسیستم قبل از شروع به ساخت کد بایستی به پایگاه داده موردنظر وصل شده و اطلاعات موردنیاز را از آن دریافت نماید. پس از دریافت، این اطلاعات درون متغیرهای خاصی قرار داده خواهند شد و سازنده کد در حین ساخت کد می‌تواند از آن‌ها استفاده کند.

حال که این اطلاعات دریافت شده‌اند برای ساخت کد به زبان برنامه‌نویسی هدف، نیازمند به یک ساختار عمومی (قالب ساخت کد) هستیم. این قالب می‌تواند توسط توسعه‌دهنده سیستم هدف به سازنده کد داده شود و یا از قالب پیش‌فرض ساخت کد استفاده شود. این قالب‌ها متون ثابتی هستند که با قرار گرفتن اطلاعات مربوط به پایگاه داده در میان آن‌ها تبدیل به کدهای لایه دسترسی به پایگاه داده خواهند شد. در لایه دسترسی به پایگاه داده برای هر جدول فایل مخصوصی در نظر گرفته شده است که کدهای مربوط به کار با آن جدول در آن قرار می‌گیرد.

کد درج در پایگاه داده به شکل یک پرس و جو[11] قابل‌اجرا خواهد بود. این پرس‌وجو بایستی توسط سازنده کد ساخته شود و توسط کد ساخته‌شده به زبان برنامه‌نویسی هدف، به سمت پایگاه داده ارسال شود. کد درج برای هر جدول بایستی مقادیر ستون‌های آن جدول را دریافت نماید و آن‌ها را در پایگاه داده قرار دهد. برای ساخت کد درج ابتدا پرس و جوی آن را باید شناخت.

پرس و جوی درج در هر جدول به شکل زیر است:

```
INSERT INTO table_name
(column1,column2,column3,…);
VALUES (value1,value2,value3,…);
```

با توجه به آن‌که قبل از ساخت کد توسط crudgenerator نام ستون‌های هر جدول مربوط به پایگاه داده را دریافت کرده‌ایم و نام جداول به شکل T_i و نام ستون‌های هر جدول C_i، است، بنابراین شبه کد مربوط به ساخت کد درج به شکل زیر خواهد بود:

```
Prepare Connection String for database
Foreach Table Ti
        Write Class Name for Ti
        Write Insert Function Name for Tiwith
V1 to Vn Input Parameters
        Write Connection String Variable
        //Insert Query Maker
        Write INSERT INTO Ti_Name
Foreach Columns in Ti Write Ci
Write VALUES
Foreach Columns in Ti WriteVi //Insert Query
Made
Write Open Connection to Database By
 ConnectionString Variable Defined
Write Execute Insert Query on Database
Write Close Connection
```

کد حذف نیز در پایگاه داده مانند کد درج دارای یک پرس‌وجو است، اما به‌جای نیاز به دانستن کلیه ستون‌های هر جدول کافی است که یک رکورد یکتا از هر جدول را بدانیم و از آن‌جا که به‌طور معمول هر جدول دارای یک فیلد کلید اصلی است و این فیلد نمی‌تواند دارای مقادیر تکراری در یک جدول باشد دانستن مقادیر این فیلد برای پرس و جوی حذف کافی است و نیازی به دانستن بقیه اطلاعات مربوط به ساختار جدول برای نوشتن این پرس‌وجو وجود ندارد.

پرس و جو حذف در پایگاه داده به شکل زیر است:

```
DELETE FROM table_name
WHERE some_column=some_value;
```

توجه شود که نام ستون فیلد کلید اصلی به شکل Ci_p در نظر گرفته می‌شود، بنابراین شبه کد مربوط به ساخت کد حذف به شکل زیر است:

```
Prepare Connection String for database
Foreach Table Ti
        Write Class Name for Ti
        Write DELETE Function Name for
TiWith Vi_p Input Parameter
        Write Connection String Variable
        // DELETE Query Maker
        Write DELETE FROM Ti_Name
        Write WHERE Ci_p = Vi_p // Delete
Query Made
```

[11] Query

است:

```
SELECT column_name,column_name
FROM table_name;
```

شبه کد مربوط به ساخت کد دریافت (به‌صورت پیش‌فرض) به شکل زیر می‌باشد:

```
Prepare Connection String for database
Foreach Table Ti
        Write Class Name for Ti
        Write Select Function Name for TiWith C1 to Cn Input Parameters
        Write Connection String Variable
        //Select Query Maker
        Write SELECT
        Foreach Columns in Ti Write Ci
        Write From Ti_Name //Select Query Made
        Write Open Connection to Database By ConnectionString Variable Defined
        Write Execute Select Query on Database
        Write Close Connection
```

پس‌ازآن‌که یک سازنده کد بتواند لایه دسترسی به پایگاه داده را تولید کند می‌تواند این خدمات را در لایه دیگری به نام لایه سرویس ارائه کند البته لایه سرویس نیازمند ساخت کدهای مجزا از لایه دسترسی به پایگاه داده می‌باشد. در لایه سرویس مواردی از جمله پروتکل‌های مورد استفاده در هر سرویس و خدمات قابل ارائه و نحوه دسترسی به این خدمات بایستی در نظر گرفته شوند. استفاده از پروتکل‌های مختلف در لایه سرویس معمول نیست و به‌صورت معمول در هر نرم‌افزار برای ساخت لایه سرویس و ارائه خدمات توسط سرویس‌ها از یک پروتکل استفاده می‌شود اما زمانی که بحث سازنده کد به میان می‌آید، دیگر معلوم نیست که این سازنده کد چه لایه سرویسی را تولید خواهد کرد. ممکن است این سازنده کد درجایی مورد استفاده قرار گیرد که پروتکل مورداستفاده آن REST باشد و یا درجایی مورداستفاده قرار گیرد که پروتکل هدف SOAP باشد. وظیفه تولید سرویس‌ها و ارائه آن‌ها در قالب معماری سرویس گرا بر عهده زیرسیستم Servicegenerator می‌باشد. در ادامه در مورد بخش‌های مختلف این زیرسیستم و همچنین سرویس‌گرایی بحث خواهیم کرد.

ارائه سرویس در قالب پروتکل‌هایSOAP و REST: با توجه به آن‌چه گفته شد امکان ایجاد هر دو پروتکل SOAP و REST در سازنده کد به این زیرسیستم سپرده شده

```
        Write Open Connection to Database ByConnectionString Variable Defined
        Write Execute Delete Query on Database
        Write Close Connection
```

برای ساخت کد مربوط به پرس و جوی به‌روزرسانی بایستی هم اطلاعات مربوط به مقادیر جدید یک رکورد دریافت شوند و هم رکورد مربوطه مشخص شود. اطلاعات جدید مربوط به یک رکورد و اطلاعات مربوط به رکورد هدف برای به‌روزرسانی، به‌صورت ورودی گرفته می‌شوند.

پرس و جو به‌روزرسانی در پایگاه داده به شکل زیر می‌باشد:

```
UPDATE table_name
SET column1=value1,column2=value2,…
WHERE some_column=some_value;
```

شبه کد مربوط به ساخت کد به‌روزرسانی در لایه دسترسی به داده به این صورت است:

```
Prepare Connection String for database
Foreach Table Ti
        Write Class Name for Ti
        Write Update Function Name for TiWith V1 to Vn and C1 to Cn and Ci_p=Vi_p Input
        Write Connection String Variable
        //Update Query Maker
        Write UPDATE Ti_Name
        Write SET
        Foreach Columns in Ti Write SET Ci=Vi
        Write WHERE Ci_p=Vi_p //Update Query Made
        Write Open Connection to Database By ConnectionString Variable Defined
        Write Execute Update Query on Database
        Write Close Connection
```

کد دریافت مانند سایر عملیات گفته‌شده در لایه دسترسی به پایگاه داده دارای یک پرس و جوی قابل اجرا در پایگاه‌داده می‌باشد، اما برخلاف عملکردهای قبل این پرس‌وجو می‌تواند دارای پارامترهای متعددی باشد. از آن‌جا که نمی‌توان همه حالات مختلف در این پرس‌وجو را برای ساخت کد در نظر گرفت این قسمت فقط به‌عنوان یک قالب پیش‌فرض در سازنده کد نوشته می‌شود و توسعه‌دهنده هرسیستم با توجه به نیاز سیستم خود می‌تواند با تغییر قالب سازنده کد، سایر پرس‌وجوهای دریافت را تولید کند. ساختار کلی پرس و جو دریافت از پایگاه داده به شکل زیر

است.

ارائه سرویس‌ها به صورتی که توسط AJAX قابل فراخوانی باشد: استفاده از AJAX در بسیاری از زیر سیستم‌ها و سیستم‌های خارجی با توجه به زیاد شدن حجم صفحات اینترنتی یک نیاز محسوب می‌شود. به همین جهت سرویس‌های تولید شده توسط این زیرسیستم بایستی قابلیت فراخوانی توسط AJAX را دارا باشد.

امکان ارسال اطلاعات سرویس‌ها در ساختارهای داده‌ای مختلف: از آن‌جا که در سیستم‌های مختلف داده‌ها با فرمت‌ها و ساختارهای متفاوتی رد و بدل می‌شوند لازم است که این زیرسیستم ساختارهای داده‌ای مختلفی را برای انتقال داده‌های سرویس‌ها پشتیبانی کند.

این زیرسیستم برای برآورده کردن خواسته‌های تعریف‌شده می‌بایستی تعدادی از عملکردهای سیستم را به‌عنوان سرویس‌های قابل ارائه در نظر بگیرد و آن‌ها را توسط پروتکل REST و SOAP ارائه نماید. سپس با استفاده از ساختارهای داده‌ای مشخص‌شده این عملکردها را ارائه کند. عملکردهایی که در این زیرسیستم به‌عنوان سرویس ارائه خواهند شد قسمتی از عملکردهای داخلی سیستم می‌باشند. به‌طور پیش‌فرض در سازنده کد SOG این عملکردها همان عملکردهای تولید شده توسط زیرسیستم Crudgenerator می‌باشند.

پیش از آن‌که نحوه کارکرد و ساخت سرویس‌ها توضیح داده شود، ابتدا لازم است انواع فرمت‌های داده‌های متنی مورد استفاده در سازنده کد SOG مشخص شود. برای انتقال داده‌ها، فرمت‌های متنی زیادی وجود دارد که توسط افراد مختلف ارائه و استاندارد شده‌اند. دو فرمت مورد استفاده که در سازنده کد SOG پیش‌بینی شده‌اند به شکل زیر است:

- فرمت XML
- فرمت JSON

سرویس‌هایی که با پروتکل REST ارائه می‌شوند می‌بایستی توسط متدهای HTTP با فرمتی خاص درخواست شوند. ابتدا به فرمت خاص این متدها اشاره می‌کنیم و نحوه عملکرد سرویس ساخته‌شده را نشان می‌دهیم سپس به نحوه ساخت این سرویس توسط سازنده کد خواهیم پرداخت. در مورد یک درخواست HTTP باید توجه داشت که این درخواست شامل پارامترهایی است که می‌توان با آن‌ها نوع سرویس درخواستی را مشخص نمود.

برخی از پارامترهای مورد استفاده در یک درخواست HTTP به شرح هستند:

- URL
- Body
- HTTPMethod

سرویس‌های ساخته‌شده هریک با توجه به متد HTTP، متناظر هستند و توسط کدهای از قبل ساخته شده بر روی پایگاه داده اجرا می‌شوند. جدول ۲ عملکرد متدهای HTTP درمقابل پرس‌وجوی SQL را نمایش می‌دهد.

جدول ۱- عملکرد متدهای HTTP در مقابل پرس و جوی SQL

HTTP Method	SQL Query Type
HTTP GET	SELECT
HTTP POST	INSERT
HTTP PUT	UPDATE
HTTP DELETE	DELETE

پاسخی که این سرویس خواهد داد با توجه به ساختار داده درخواست شده یکی از حالات بیان شده خواهد بود:

- ساختار داده‌ای XML
- ساختار داده‌ای JSON
- متن ساده

درخواست ارسال شده ابتدا نام هرجدول را به‌عنوان قسمت ابتدای URL می‌نویسد سپس مابقی URL را به همراه پروتکل HTTP ارسالی مشخص می‌کند که درخواست چه نوع سرویسی ارسال‌شده است و چه عملیاتی باید صورت گیرد.

درخواست‌های دریافت داده از پایگاه داده می‌توانند پارامتر نوع داده برگشتی را توسط یک‌رشته پرس‌وجو (Querystring) مشخص کنند و در این حالت داده‌های برگشتی می‌توانند دارای هر یک از فرمت‌های XML، JSON، JSArray و یا Simpletext باشند. البته سازنده کد امکان اضافه کردن فرمت‌های دلخواه دیگر را نیز دارد که در این حالت می‌توان با تغییرات لازم در سورس برنامه فرمت‌های داده‌ای دیگر را به آن اضافه نمود.

در ادامه عملکردهای دسترسی به پایگاه داده از طریق سرویس‌های REST به‌طور کامل با جزئیات بیشتر توضیح

داده شده است همچنین نحوه ساخت این سرویس‌ها مورد بررسی قرار گرفته است.

سرویس‌هایی که با استفاده از پروتکل REST ساخته می‌شوند دارای یکURL می‌باشند. برای دریافت داده‌های یک جدول از پایگاه داده توسط یک سرویسREST بایستی درخواستی به شکل زیر با پروتکل HTTPGET ارسال شود.

http://mycodeGenerator.1developer.ir/
REST/API/Default.ashx?t=tableName

در آدرس فوق متغیر t نام جدولی را مشخص می‌کند که اطلاعات آن باید برگردانده شود و متد HTTPGET به سرویس می‌فهماند که درخواست ارسالی نیازمند رکوردهای جدول مشخص‌شده می‌باشد و بایستی دستور SELECT متناظر آن فراخوانده شود، بنابراین پاسخ این سرویس کلیه رکوردهای مربوط به جدول مشخص‌شده در پایگاه داده خواهد بود.

جزئیات این درخواست در جدول ۳ مشخص‌شده است.

جدول ۲- مشخصات عملکرد درخواست دریافت با پارامترهای URL

HTTP Method	GET
Parameters Mode	URL
HTTP Body	None
URL Format	/Default.ashx? t=tableName&format=json
SQL Operation	SELECT
SQL Parameters	tableName
SQL Query	SELECT * FROM tableName
Return	All Table Records

درصورتی‌که نیاز به ستون‌های خاصی از جدول باشد این ستون‌ها می‌توانند در قالب پارامترهای سرویس توسط URL برای سرویس ارسال شوند که در این صورت درخواست سرویس مطابق جدول ۴ خواهد بود.

جدول ۳- مشخصات عملکرد درخواست دریافت با پارامترهای URL و ستون‌های خاص

HTTP Method	GET
Parameters Mode	URL
HTTP Body	None
URL Format	../Default.ashx?t=tableName&format=json&columns=$c_1,c_2,...,c_n$
SQL Operation	SELECT
SQL Parameters	tableName c_1 to c_n columns' names
SQL Query	SELECT $c_1,c_2,...,c_n$ FROM tableName
Return	All Table Records

همان‌طور که ملاحظه می‌شود برای عملکرد دریافت داده‌ها یعنی SELECT از متد GET از پروتکل HTTP استفاده شده است.

برای ارسال داده به یک جدول از پایگاه داده توسط یک سرویسREST بایستی درخواستی به شکل زیر با پروتکل HTTPPOST ارسال شود.

http://mycodeGenerator.1developer.ir/
REST/API/Default.ashx?t=tableName

در آدرس فوق متغیر t نام جدولی را مشخص می‌کند که اطلاعات آن باید برگردانده شود و متد HTTPPOST به سرویس می‌فهماند که درخواست ارسالی حاوی داده‌هایی است که باید در جدول مشخص‌شده قرار گیرند و بایستی دستور INSERT متناظر آن فراخوانده شود. پاسخ این سرویس، داده‌های رکوردی است که در جدول مربوط در پایگاه داده قرار گرفته است. جزئیات این درخواست در جدول ۵ مشخص‌شده است:

در درخواست قبل داده‌هایی که باید به سرویس ارسال شوند در بدنه درخواست POST قرار می‌گیرند ولی می‌توان این داده‌ها را در URL درخواست نیز قرار داد که در واقع نوع پارامترها به گزینه URL تغییر می‌کند که در این صورت درخواست به صورت جدول ۶ در می‌آید. به منظور به‌روزرسانی داده یک جدول از پایگاه داده توسط یک سرویسREST، بایستی درخواستی به شکل زیر با پروتکل HTTPPUT ارسال شود.

http://mycodeGenerator.1developer.ir/REST/API/Default.ashx?t=tableName&c_{i_p}=v_{i_p}

در آدرس فوق متغیر t نام جدولی را مشخص می‌کند که اطلاعات آن باید برگردانده شود و C_{i_p} ستون کلید اصلی جدول مربوطه است و متد HTTPPUT به سرویس می‌فهماند که درخواست ارسالی حاوی داده‌هایی است که باید جدول مشخص‌شده را به‌روزرسانی کنند و بایستی دستور UPDATE متناظر آن فراخوانده شود. پاسخ این سرویس داده‌های رکوردی است که در جدول مربوط در پایگاه داده، به‌روزرسانی شده‌اند.

جدول ۴- مشخصات عملکرد درج با پارامترهای HTTP BODY

HTTP Method	POST
Parameters Mode:	Body
HTTP Body	$columnName_1$=$value_1$& $columnName_2$=$value_2$& $columnName_n$=$value_n$
URL Format	../Default.ashx?t=tableName&format=json
SQL Operation	INSERT
SQL Parameters	tableName c_1 to c_n columns' names and v_1 to v_n columns' values
SQL Query	INSERT INTO tableName(c_1,c_2,...,c_n) VALUES (v_1,v_2,...,v_n)
Return	Inserted Record

جدول ۵: مشخصات عملکرد درج با پارامترهای URL

HTTP Method	POST
Parameters Mode	URL
HTTP Body	None
URL Format	../Default.ashx?t=tableName&format=json &$columnName_1$=$value_1$ &$columnName_2$=$value_2$ &$columnName_n$=$value_n$
SQL Operation	INSERT
SQL Parameters	tableName c_1 to c_n columns' names and v_1 to v_n columns' values
SQL Query	INSERT INTO tableName(c_1,c_2,...,c_n) VALUES (v_1,v_2,...,v_n)
Return	Inserted Record

جزئیات این درخواست در دو نوع پارامترهای URL و پارامترهای بدنه در جدول ۷ مشخص‌شده است. می‌توان این داده‌ها را در URL درخواست نیز قرار داد که در واقع نوع پارامترها به گزینه URL تغییر می‌کند که در این صورت درخواست به صورت جدول ۸ در می‌آید.

جدول ۶: مشخصات عملکرد به‌روزرسانی با پارامترهای HTTP BODY

HTTP Method	PUT
Parameters Mode	Body
HTTP Body	$columnName_1$=$value_1$& $columnName_2$=$value_2$& $columnName_n$=$value_n$
URL Format	../Default.ashx?t=tableName&format=json&c_{i_p}=v_{i_p}
SQL Operation	UPDATE
SQL Parameters	tableName c_1 to c_n columns' names and v_1 to v_n columns' values
SQL Query	UPDATE tableName SET c_1=v_1,c_2=v_2,...,c_n=v_n WHERE c_{i_p}=v_{i_p}
Return	Updated Record

جدول ۷- مشخصات عملکرد بروز رسانی با پارامترهای URL

HTTP Method	PUT
Parameters Mode	URL
HTTP Body	None
URL Format	../Default.ashx?t=tableName&format=json&c_{i_p}=v_{i_p} &$columnName_1$=$value_1$ &$columnName_2$=$value_2$ &$columnName_n$=$value_n$
SQL Operation	INSERT
SQL Parameters	tableName c_1 to c_n columns' names and v_1 to v_n columns' values
SQL Query	UPDATE tableName SET c_1=v_1,c_2=v_2,...,c_n=v_n WHERE c_{i_p}=v_{i_p}
Return	Updated Record

برای حذف داده ازیک جدول از پایگاه داده توسط یک سرویسREST بایستی درخواستی به شکل زیر با پروتکل HTTPDELETE ارسال شود.

http://mycodeGenerator.1developer.ir/REST/API/Default.ashx?t=tableName&c_{i_p}=v_{i_p}

در آدرس فوق متغیر t نام جدولی را مشخص می‌کند که اطلاعات آن باید برگردانده شود و C_{i_p} ستون کلید اصلی جدول مربوطه است و متد HTTPDELETE به سرویس می‌فهماند که درخواست ارسالی خواستار حذف رکوردی از پایگاه داده است و بایستی دستور DELETE متناظر آن فراخوانده شود. پاسخ این سرویس مقدار کلید اصلی رکوردی است که از جدول مربوط در پایگاه‌داده حذف شده است.جزئیات این درخواست در جدول ۹ نمایش داده شده است.

ساخت سرویس‌هایSOAP با توجه به گستردگی این پروتکل و همچنین استاندارد بودن آن بسیار ضروری است این پروتکل نسبت به پروتکل REST دارای سابقه بیشتری بوده و هنوز نیز در بسیاری از سیستم‌های سرویس‌گرا کاربرد زیادی دارد. سرویس‌های دارای معماریSOAP دارای ساختار مبتنی بر XML هستند و همچنین قبل از ارسال درخواست به این نوع سرویس‌ها می‌توان با نگاه کردن فایلWsdl مخصوص به آن‌ها از نام توابع موجود در آن‌ها و همچنین پارامترهای ورودی و خروجی آن‌ها مطلع شد.

جدول ۸- مشخصات عملکرد حذف با پارامترهای URL

HTTP Method	DELETE
Parameters Mode	URL
HTTP Body	None
URL Format	../Default.ashx?t=tableName &c_{i_p}=v_{i_p}
SQL Operation	DELETE
SQL Parameters	tableName Primary Key column c_{i_p} Primary Key value v_{i_p}
SQL Query	DELETE FROM tableName WHERE c_{i_p}=v_{i_p}
Return	Deleted ID

با توجه به امکان ساخت کد این نوع سرویس‌ها بایستی قابلیت پشتیبانی از فرمت‌های گفته‌شده را در این سرویس-ها دارا باشند اما این پشتیبانی با توجه به اینکه درخواست-های ارسالی در این پروتکل دارای ساختار XML هستند، دچار مشکلاتی می‌شود. ازاین‌رو برای انجام این کار در مواردی که بین درخواست و پاسخ‌های این سرویس با فرمت‌های داده‌ای گفته‌شده، امکان بروز اختلال است، می‌توان با رمزنگاری و یا کدگذاری داده‌ها از این اختلال پیشگیری نمود. البته امکان رمزنگاری در این سازنده کد به‌صورت مجزا پیش‌بینی‌شده است که در همین بخش به آن خواهیم پرداخت.

بخش TestGenerator وظیفه ساخت کدهایی را بر عهده دارد که سایر کدهای تولیدشده در بخش‌های قبل از جمله بخش crudgenerator،servicegeneratorرا مورد تست قرار می‌دهند. این کدها پس از اجرا شدن مشخص می‌کنند که آیا توابع بخش‌های قبل همان‌گونه که انتظار می‌رود عمل می‌نمایند یا خیر.

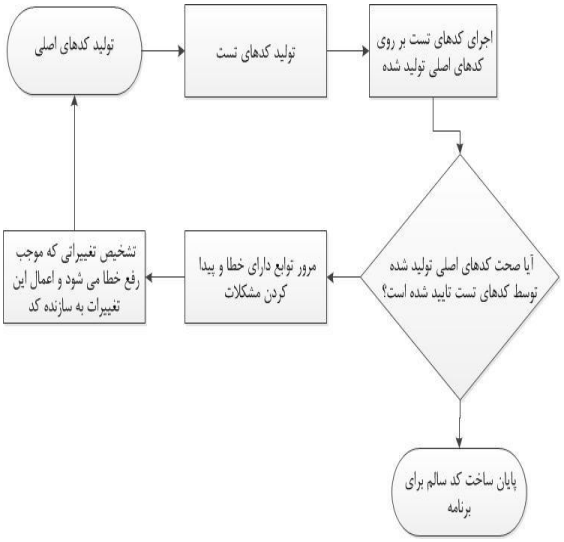
منطق این کدها به شکلی است که می‌توانند خطاهای ممکن را بررسی و در صورتی‌که کدهای قبل دارای خطایی باشند توسعه‌دهندگان را از آن با خبر کنند. توسعه‌دهندگان می‌توانند پس از اجرای کدهای تولیدشده در این بخش نتیجه صحت بخش‌های قبل را به‌صورت جداگانه مشاهده نمایند و برای هر تابع و یا سرویس نتیجه صحت و یا عدم صحت مشخص می‌شود.

درصورتی‌که یک تابع با خطایی مواجه شود این خطا توسط این کدها مشخص می‌شود و در برابر نام آن تابع، گزینه وجود خطا قرار خواهد گرفت و توسعه‌دهندگان می‌توانند با مرور کدهای تولیدشده برای آن تابع هرگونه ایراد را تشخیص داده و راهکاری متناسب با آن ارائه نمایند. راهکار ارائه‌شده پس از اعمال در قالب سازنده کد نیاز به اجرای دوباره فرآیند ساخت کد دارد.

پس از آن‌که ایراد تشخیص داده شد و فرآیند ساخت کد نیز با اصلاحات صورت گرفته انجام شد، دوباره کدهای مربوط به تست تولید می‌شوند و دوباره این کدها برای اطمینان از صحت سیستم اجرا می‌شوند و این فرآیند آن‌قدر تکرار می‌شود تا کدهای تست، صحت همه توابع سیستم را اعلام نمایند. این صحت هم شامل توابعی از سیستم است که به‌صورت سرویس ارائه می‌شوند و هم شامل توابع سایر بخش‌های سیستم که می‌توان منطقی را برای اطمینان از صحت آن‌ها در نظر گرفت و کد مورد نظر آن را تولید کرد.

شکل زیر فرآیند تولید کد توسط سازنده کد و تست کدهای تولید شده و سایر مراحل اصلاحات تا رسیدن به کدهای سالم تست شده را نمایش می‌دهد. این فرآیند صرفاً به‌عنوان

یک فرآیند پیشنهادی ارائه شده و الزامی برای تبعیت از آن وجود ندارد و توسعه‌دهندگان می‌توانند با توجه به مدل و فرآیند توسعه نرم‌افزار خودشان از کدهای تست تولیدشده توسط سازنده کد استفاده نمایند.



شکل ۳- نحوه عملکرد Test Generator

شکل (۳) نحوه عملکرد Test Generator در سازنده کد SOG را نمایش می‌دهد.

در ادامه به بررسی وظایف زیر سیستم MonitoringGenerator می‌پردازیم.

زیرسیستم MonitoringGenerator وظیفه ساخت کدهایی برای نظارت بر سیستم را، بر عهده دارد. با توجه به این‌که نظارت در سیستم‌های مختلف مفاهیم مختلفی دارد، این قسمت برخی عملکردهای پیش‌فرض را برای مانیتورینگ سیستم تولید می‌کند. برخی از عملکردهایی که این زیرسیستم به‌صورت پیش فرض تولید می‌کند به شرح زیر است:

- تعداد پرس‌وجوهای انجام شده بر روی پایگاه داده در یک بازه‌ی زمانی
- آخرین پرس و جوی انجام شده بر روی پایگاه داده
- میانگین درصد استفاده سیستم از CPU در مدت زمان تعیین‌شده
- میانگین RAM اشغال شده توسط سیستم

در ادامه زیرسیستم SecurityGenerator را مورد بررسی قرار خواهیم داد.

زیرسیستم SecurityGenerator وظیفه ساخت کدهای مربوط به امنیت نرم‌افزارتولید شده را بر عهده دارد. این زیرسیستم کدهایی تولید می‌کند که اجازه دسترسی به سرویس‌های ساخته شده را فقط به درخواست‌های مجاز صادر می‌کند. در سیستم‌های مختلف نرم‌افزاری از مکانیسم‌های مختلفی برای تعیین سطح دسترسی کاربران و ورود و خروج آن‌ها استفاده می‌شود و مقوله‌های مختلف امنیتی از جمله Authorization وAuthentication در سیستم‌های مختلف به‌صورت یکسانی پیاده‌سازی نمی‌شوند. از همین رو ارائه یک نوع خاص از تأمین امنیت در یک سازنده کد ممکن است موجب اجبار توسعه‌دهندگان به استفاده از آن نوع مکانیسم امنیتی خاص شود و در صورت عدم علاقه توسعه‌دهندگان به آن نوع مکانیسم خاص موجب تولید کدهای بی‌مصرف در این زمینه شود. از این رو یک سازنده کد برای ساخت کدهای امنیتی سیستم لازم است راهکاری را مورداستفاده قرار دهد که علاوه بر تأمین امنیت لازم برای سرویس‌های تولید شده، امکان استفاده از مکانیسم‌های امنیتی مختلف را برای توسعه‌دهندگان فراهم کند. برای ارائه چنین راهکاری سازنده کد ارائه شده باید قابلیت انعطاف زیادی در برابر استفاده از مکانیسم‌های مختلف داشته باشد. برای این منظور دو رویکرد عمومی وجود دارد:

- سازنده کد همه راهکارهای موجود را در خود داشته باشد و توسعه‌دهندگان با توجه به نیاز خود راهکار موردنظر را انتخاب نمایند.
- سازنده کد کدهای امنیتی سیستم را به‌گونه‌ای تولید کند که توسعه‌دهندگان بتوانند مکانیسم امنیتی مخصوص به خودشان را در آن جای دهند.

استفاده از راهکار اول باعث حجیم شدن نرم‌افزار سازنده کد و استفاده از معماری‌های متفاوت و بعضاً متناقضی در درون آن خواهد شد و از سوی دیگر ارائه همه مکانیسم‌های موجود در سازنده کد ممکن نیست چون راهکارهای مختلف تعداد محدود و ثابتی ندارند و نیز در برخی سیستم‌ها از مکانیسم‌های ابتکاری استفاده می‌شود که خاص آن سیستم است. به دلایل گفته شده استفاده از رویکرد اول عملاً غیرممکن است بنابراین در سازنده کد پیشنهادی از رویکرد دوم استفاده شده است. این رویکرد به این شکل ارائه‌شده

است که سازنده کد یک تابع با سطح دسترسی عمومی تولید می‌نماید و از آن در قسمت‌های موردنیاز برای تأمین سطح دسترسی و امنیت سرویس‌ها استفاده می‌کند سپس بدنه تابع را به توسعه‌دهندگان واگذار می‌کند؛ بنابراین توسعه‌دهندگان با پیاده‌سازی بدنه این تابع می‌توانند مکانیسم امنیتی موردنظر خودشان را اعمال کنند.

زیرسیستم Encryptiongenerator مسئول تولید کد برای رمزنگاری داده‌های انتقالی بین سرویس‌دهنده و سرویس‌گیرنده را بر عهده دارد، از آن‌جایی‌که پروتکل‌های انتقالی سرویس‌ها عموماً متنی می‌باشد امکان شنود اطلاعات برای یک sniffer وجود دارد که در این صورت امکان خواندن اطلاعات یک درخواست‌کننده مجاز برای sniffer به‌وجود می‌آید. ممکن است بتوان برای جلوگیری از این مشکل از راهکارهایی مانند SSL[۱۲] استفاده نمود اما بااین‌همه وجود یک لایه Encryption می‌تواند امنیت داده‌ها را چند برابر نماید.

نحوه کار کدهای رمزنگاری به شکلی است که داده‌های سرویس موقع ارسال به سرویس‌گیرنده رمزنگاری می‌شوند و سرویس‌گیرنده فقط با داشتن کلید رمز می‌تواند آن‌ها را بازگشایی کند این عمل باعث می‌شود تا sniffer حتی با گوش دادن به خط ارتباطی و عبور از سایر موانع شنود بازهم نتواند داده‌های انتقالی بین سرویس‌دهنده و سرویس‌گیرنده را بخواند.

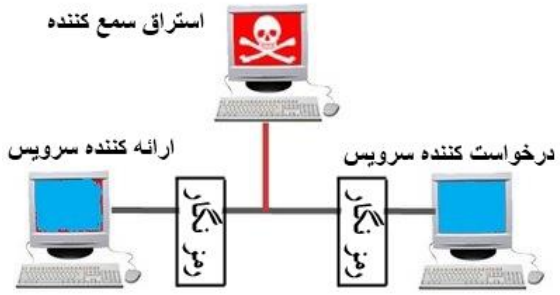
با توجه به اینکه الگوریتم‌های رمزنگاری متفاوتی در این زمینه وجود دارند سازنده کد بایستی به‌گونه‌ای طراحی شود تا بتواند دست توسعه‌دهندگان را برای استفاده از انواع مختلف الگوریتم‌های رمزنگاری باز بگذارد در این صورت توسعه‌دهندگان می‌توانند با اعمال الگوریتم موردنظر در قالب سازنده کد از الگوریتم رمزنگاری موردنظرشان برای رمزنگاری و رمزگشایی استفاده کنند. در شکل‌های (۴) و (۵) نحوه کارکرد استراق سمع کننده و نحوه جلوگیری از آن نشان داده شده است.

بعد از بیان فازهای مختلف کد SOG، به بیان نحوه ساخت این کد پرداخته سپس توضیحات تکمیلی آورده می‌شود.

- **نحوه ساخت کد توسط SOG**

به طور خلاصه می‌توان گفت نحوه ساخت کد توسط **SOG** بر پایه ساختار یک پایگاه داده با پروتکل سرویس‌گرایی **SOAP** است و همین مراحل برای پروتکل **REST** نیز تکرار خواهند شد.

شکل ۴- تبادل داده بدون رمزنگار



شکل ۵- تبادل داده با استفاده از رمزنگار

اگر که یک پایگاه داده را شامل **n** جدول در نظر بگیریم شبه الگوریتم **SOG** به صورت کلی بدین شکل خواهد بود:

foreach T_n in Database

{

Generate Service T_n_name

Generate AuthorizationMethod$_i$ for T_n_name

Generate All WorkWithTableMethod$_i$ T_n_name

Generate EncryptionMethod$_i$ to Return Data

}

- **توضیحات تکمیلی**

ساخت سرویس: SOG بر اساس هریک از اشیاء پایگاه داده سرویسی را با نام همان شئ تولید می‌نماید و در داخل این سرویس دستورات و توابع کلی کار با آن شئ پایگاه داده قرار داده می‌شود.

[12] Secure Sockets Layer

ساخت متد سطح دسترسی: از آنجا که تمام افراد مجاز نیستند از وب سرویس‌های یک سیستم استفاده نمایند بنابرین SOG با تولید متدی تحت عنوان "متد بررسی سطح دسترسی" این قابلیت را ایجاد می‌کند تا توسعه دهنده سیستم بتواند با استفاده از یک متد کلیه دسترسی‌ها به وب سرویس‌های سیستم را کنترل نماید.

ساخت لایه دسترسی به داده(DAL[۱۳]): کلیه کدهای دسترسی به پایگاه داده و انجام عملیات مختلف بر روی آن در این قسمت توسط SOG تولید می‌شوند. این کدها توسط قسمت سرویس تولید شده SOG مورد استفاده قرار می‌گیرند.

ساخت رمزنگار: SOG برای رمزنگاری اطلاعات، زیر سیستمی را تولید می‌کند که کار رمزنگاری اطلاعات را انجام می‌دهد. این زیرسیستم قابلیت رمزنگاری اطلاعات را توسط سرور دارا می‌باشد و کلاینتی که می‌خواهد از این زیر سیستم استفاده کند می‌بایستی بتواند این اطلاعات را توسط کلید مخصوص دریافت شده از SOG بازگشایی نماید.

جهت نشان دادن خصوصیات سازنده کد ارائه شده در این مقاله، ویژگی‌ها و قابلیت‌های نوین سازنده کد SOG در ادامه آورده شده است.

الف) پشتیبانی از پروتکل‌های سرویس‌گرایی

SOG از پروتکل‌های اصلی سرویس‌گرایی مانند **SOAP** و **REST** به خوبی پشتیبانی می‌کند و کد نهایی تولید شده می‌تواند در هریک از این پروتکل‌ها به کاربران سرویس ارائه دهد.

ب) قالب‌های داده‌ای قابل ارسال

داده‌ها در معماری سرویس‌گرا پس از آماده سازی به درخواست کننده سرویس ارسال می‌شوند، این داده‌ها بسته به نوع پروتکل استفاده شده می‌توانند قالب‌های مختلفی داشته باشند برای مثال در پروتکل **SOAP** داده ها در قالب **XML** اختصاصی تعریف شده در این پروتکل ارسال می‌شوند اما در بسیاری از اوقات لازم است تا این داده‌ها در قالب دیگری به غیر از قالب ارسال شده استفاده شوند و در این حالت نیاز است تا دریافت کننده داده آن‌ها را به قالب مورد نیاز خود تبدیل نماید اما در صورتی که درخواست کننده داده بتواند قالب مورد نیاز خود را به ارسال کننده داده بگوید و ارسال کننده، داده‌ها را با همان قالب ارسال نماید دیگر نیازی به تبدیل داده‌ها نیست و این موضوع در **SOG** پیش بینی شده است.

SOG از چندین قالب کلی داده برای ارسال داده مانند **XML, JSON, Java Script Array, Plain Text** پشتیبانی می‌کند و همچنین امکان اضافه کردن قالب‌های جدید به آن نیز وجود دارد.

پ) زبان‌های برنامه نویسی پشتیبانی شده

SOG از زبان‌های برنامه نویسی **C#, PHP, Python, JAVA, Delphi, Visual Basic, C++** پشتیبانی می‌کند که در مقابل سازنده کدهای دیگر بسیار قدرتمندتر بشمار می‌رود.

ت) سطح دسترسی به سرویس‌ها

به صورت پیش فرض در هر سرویس نوشته شده در یک سیستم با معماری سرویس‌گرا دسترسی برای همگان آزاد است مگر آنکه سازنده سرویس به شکلی این دسترسی را محدود کند. حال زمانی که ما از یک سازنده کد برای ساخت تعداد زیادی سرویس در یک سیستم با معماری سرویس‌گرا استفاده می‌کنیم به طور قطع در برخی اوقات نیازمند این خواهیم بود تا سطح دسترسی به برخی از سرویس‌ها به نوعی محدود شود و در صورتی که سازنده کد این کار را برای ما انجام ندهد نیازمند انجام این کار به صورت دستی خواهیم شد با این همه در صورتی که این کار به صورت خودکار صورت بگیرد ساختار و معماری سیستم با تمام خواصی که استفاده از سازنده کد دارد و در بخش‌های قبل به آن‌ها اشاره کردیم، تولید خواهد شد.

نکته دیگر در مورد ایجاد سطح دسترسی و قرار دادن محدودیت برای استفاده از سرویس‌ها این است که سیستم-های مختلف برای ایجاد سطح دسترسی از روش‌های مختلفی استفاده می‌کنند و در صورتی که سازنده کد برای این کار از یک روش خاص استفاده نماید ممکن است این روش با روشی که سیستم نهایی مایل به استفاده از آن است همخوانی نداشته باشد و سیستم تولید شده دچار مشکلاتی شود. برای حل این مشکل سازنده کد **SOG** سطح دسترسی را به گونه‌ای تعریف می‌کند که تولید کننده سیستم نهایی از هر روشی که می‌خواهد برای بررسی سطح دسترسی کاربر استفاده کند و بتواند آن را در سرویس

[13] Data Access Layer

نوشته شده بگنجاند در حالی که کد کلی سطح دسترسی به هر سرویس توسط سازنده کد نوشته شده است.
بدین منظور سازنده کد برای ساخت هر سرویس از معماری زیر استفاده می‌نماید که در آن تابع مربوط به بررسی سطح دسترسی پارامتر، نام سرویس را دریافت می‌کند و بر آن اساس تصمیم می‌گیرد که آیا کاربر اجازه دسترسی به سرویس مورد نظر را دارد یا خیر و در صورتی که این اجازه را ندارد خطای عدم دسترسی را بر میگرداند

نام سرویس
{
تابع مربوط به بررسی سطح دسترسی(نام سرویس)
کدهای مربوط به سرویس( )
}

بدنه تابع مربوط به بررسی سطح دسترسی توسط خود کاربر نوشته می‌شود و از آنجا که این تابع برای تمامی سرویس-های ساخته شده توسط سازنده کد فراخوانی می‌شود بنابرین سطح دسترسی بر اساس روش انتخابی طراح سیستم نهایی اعمال شده اما سازنده کد آن روش را در ساخت کدهای خود لحاظ کرده است.

ث) رمزنگاری داده‌های ارسالی و دریافتی

یکی از موارد دیگری که ممکن است در یک سیستم با معماری سرویس‌گرا مورد استفاده قرار گیرد استفاده از روشی برای رمزنگاری داده‌های دریافتی و ارسالی است. از آنجا که داده‌های هر سرویس بر پایه متن ارسال و دریافت می‌شوند بنابرین به راحتی قابل شنود هستند، به عنوان مثال فرض کنید در سیستم، سرویسی وجود دارد که لیست کل کاربران به همراه تمام جزییات را ارائه می‌کند و فقط برای مدیران ارشد یک سیستم قابل استفاده است، حال اگر یک مدیر ارشد این سرویس را درخواست نماید و در این میان یک استراق سمع کننده وجود داشته باشد بنابرین می‌توان مطمئن بود که استراق سمع کننده به این لیست دسترسی پیدا کرده است، اینجاست که نیاز به رمزنگاری داده‌ها بوجود می‌آید. درصورتی که داده‌ها قبل از ارسال رمز نگاری شوند نگرانی شنود آن‌ها توسط استراق سمع-کنندگان از بین خواهد رفت. در بخش قبل در شکل‌های (۴) و (۵) نحوه ایجاد رمزنگار در **SOG** نشان داده شده است.
SOG می‌تواند لایه رمزنگاری را به صورت خودکار تولید نماید و دیگر نیازی به نگرانی برای شنود داده‌ها وجود نخواهد داشت و همچنین از آن‌جا که این لایه به صورت خودکار تولید می‌شود نیازی به صرف وقت و هزینه برای تولید آن وجود ندارد.

۵- ارزیابی SOG نسبت به دو سازنده کد شان و بووان

ویژگی‌های SOG که دربخش قبل بیان شده، هریک می‌تواند به عنوان پارامتری مفید در جهت طراحی یک سازنده کد کاربردی برای توسعه نرم‌افزارهای گوناگون باشد. با ارزیابی سازنده کدSOG در مقایسه با سایر سازنده کدها مشخص می‌شود که این سازنده‌کداز هر دو پروتکل REST و SOAP پشتیبانی می‌کند. همچنین قابلیت‌های ساخت کد نظارت، ساخت کد آزمودن، ساخت لایه رمزنگاری، مدل‌سازی کد تولید شده و مدیریت پایگاه داده را بر خلاف سازنده‌کدهای ارائه شده پوشش می‌دهد. برای ارزیابی سازنده‌کد پیشنهادی، از روش پرسشنامه‌ای استفاده شده است. پرسش‌ها توسط ۱۴ نفر خبره و کارشناس در زمینه معماری سرویس‌گرا پاسخ داده شده است. به منظور رفع مشکل عدم همگرایی موجود در نظرات از روش دلفی‌فازی [۲۰] استفاده شده و نتایج پس از نرمال شدن بیان شده است. ارزیابی سازنده کدها بر اساس معیارهای امنیتی، برنامه‌نویسی‌های تحت پشتیبانی، قالب انتقال اطلاعات و موارد مشابه صورت گرفته است چرا که این معیارها، از مهمترین معیارهای ارزیابی سازنده کد از دیدگاه افراد متخصص و خبره بوده است.

۶-نتیجه گیری و راهکارهای آینده

در این مقاله یک سازنده کد ترکیبی ارائه و در قالب قابلیت-های نوین مانند پیش‌بینی سطح دسترسی، تعدد قالب‌های انتقال اطلاعات بین کلاینت و سرور، اضافه شدن لایه رمزنگاری و تولید پنل مخصوص مدیریت پایگاه‌داده توسعه یافت. سازنده کد ارائه شده می‌تواند سرویس‌های لازم را تولید و در توسعه نرم‌افزارهای مبتنی بر سرویس بکار ببرد. علاوه بر موارد بیان شده، استفاده از سازنده کد **SOG** باعث عدم درگیر شدن با چالش‌های طراحی سرویس‌ها خواهد شد. سرویس‌های ساخته شده در این نرم‌افزار از پروتکل‌های**SOAP**و**REST** پشتیبانی می‌کنند. این پروتکل‌ها از بیشترین پروتکل‌های مورد استفاده درمعماری سرویس‌گرا هستند. این سازنده‌کد علاوه بر ساخت وب سرویس‌های لازم، کد مورد نیاز جهت امنیت این وب

سرویس‌ها را فراهم می‌کند. این کد امنیتی خللی در استفاده از ساختار تعیین سطح دسترسی‌ها در هر سیستم را به‌وجود نمی‌آورد بلکه به‌گونه‌ای طراحی شده است که در هر سیستم با توجه به نوع سطح دسترسی‌ها بتواند پاسخگوی نیاز آن سیستم باشد. علاوه بر ساختار امنیتی مربوط به سطح دسترسی به سرویس‌ها در این سازنده کد امکان رمزنگاری داده‌های قابل انتقال نیز پیش بینی شده است. این سازنده کد امکان تولید کد به هفت زبان برنامه نویسی را میسر می‌سازد و می‌تواند داده‌های قابل انتقال را به چهار فرمت مورد استفاده معمول، ارائه کند. استفاده از راهکار ترکیبی در سازنده کد پیشنهادی باعث شد که دو هدف عمده‌ی استفاده از استانداردهای کاری و تعریف مجدد بلوک‌های اصلی برآورده شود. سازنده کد پیشنهادی، در قالب پارامترهای ارزیابی در حیطه سرویس‌گرایی، دارای پشتیبانی مناسب‌تری از پارامترهای ساخت لایه رمزنگاری، مدل‌سازی سازنده کد، مدل‌سازی کد تولید شده و تولید کد مدیریت پایگاه‌داده نسبت به دو سازنده کد شان و بووان است.

کارهای آینده در زمینه سازنده‌های کد سرویس‌گرا می‌تواند در دو بخش متمرکز شود، بخش اول ارزیابی سازنده‌های کد توسعه یافته با شیوه‌ای تجربی یا مقایسه‌ای است و بخش دوم توسعه سازنده‌های کد با استفاده از راهکار ترکیبی و روش ادغام متدهاست. این تحقیقات می‌تواند زمینه را برای معرفی نسل بعدی سازنده‌های کد فراهم آورد.

مراجع

[1] White, L. J., Reichherzer, T., Coffey, J., Wilde, N., & Simmons, S. “Maintenance of service oriented architecture composite applications: static and dynamic support”, Journal of Software: Evolution and Process, Vol.25, No.1, 2013, pp.97-109.

[2] Wang, S., Liu, Z., Sun, Q., Zou, H., & Yang, F. “Towards an accurate evaluation of quality of cloud service in service-oriented cloud computing” Journal of Intelligent Manufacturing, Vol. 25, No. 2, 2014, pp.283-291.

[3] Oliveira, P., & Matos, P. J., “BLEGen—A Code Generator for Bluetooth Low Energy Services”Lecture Notes on Software Engineering, Vol. 4, No.1, 2016.

[4] Vrba, P., Marik, V., Siano, P., Leitão, P., Zhabelova, G., Vyatkin, V., & Strasser, T. “A review of agent and service-oriented concepts applied to intelligent energy systems”, IEEE Transactions on Industrial Informatics, Vol. 10, No. 3, 2014, pp. 1890-1903.

[5] Biehl, M., El-Khoury, J., & Törngren, M. “High-Level specification and code generation for service-oriented tool adapters”, 2012 12th International Conference on Computational Science and Its Applications (ICCSA), 2012, pp. 35-42.

[6] Jongmans, S. S. T., Santini, F., Sargolzaei, M., Arbab, F., & Afsarmanesh, H. “Automatic code generation for the orchestration of web services with Reo”, European Conference on Service-Oriented and Cloud Computing, Berlin, 2012, pp. 1-16.

[7] Naujokat, S., Traonouez, L. M., Isberner, M., Steffen, B., & Legay, A. ,“Domain-specific code generator modeling: a case study for multi-faceted concurrent systems”, International Symposium on Leveraging Applications of Formal Methods, Verification and Validation , Springer Berlin Heidelberg, 2014, pp. 481-498

[8] Jongmans, S. S. T., Santini, F., Sargolzaei, M., Arbab, F., & Afsarmanesh, H.,”Orchestrating web services using Reo: from circuits and behaviors to automatically generated code” Service Oriented Computing and Applications, Vol. 8, No.4, 2014, pp. 277-297.

[9] Ringert, J. O., Roth, A., Rumpe, B., & Wortmann, A.,”Code Generator Composition for Model-Driven Engineering of Robotics Component & Connector Systems”, 1st International Workshop on Model-Driven Robot Software Engineering, York, Great Britain, Volume 1319, 2015.

[10] Fertalj, K., & Brcic, M., “A source code generator based on uml specification”,International journal of computers and communications, Vol. 2, No.1, 2008, pp.10-19.

[11] Lazetic, S., Savic, D., Vlajic, S., & Lazarevic, S.,"A generator of MVC-based web applications", World of Computer Science and Information Technology Journal (WCSIT), Vol.2, No.4, 2012, pp.147-156.

[12] Imam, A. T., Rousan, T., & Aljawarneh, S., "An expert code generator using rule-based and frames knowledge representation techniques", 5th International Conference on Information and Communication Systems (ICICS), IEEE, 2014, pp. 1-6.

[13] Ries, C. B., & Grout, V., "Code generation approaches for an automatic transformation of the unified modeling language to the Berkeley open infrastructure for network computing framework", International Conference on Soft Computing and Software Engineering (SCSE), 2013.

[14] Mattingley, J., & Boyd, S. "CVXGEN: a code generator for embedded convex optimization", Optimization and Engineering, Vol.13, No.1, 2012, pp.1-27.

[15] Siret, N., Wipliez, M., Nezan, J. F., & Rhatay, A.,"Hardware code generation from dataflow programs", Conference on Design and Architectures for Signal and Image Processing (DASIP),IEEE, 2010, pp.113-120.

[16] Shan, J., Ma, D., Zhang, B., & Luo, W. (2009, September),"Research and implementation of code generator for information system based on SOA", Eighth International Conference on Embedded Computing, IEEE, 2009, pp.143-147.

[17] Boian, F., Chinces, D. I. A. N. A., Ciupeiu, D., Homorodean, D., Jancso, B., & Ploscar, A. D. I. N. A.,"WSWRAPPER—A UNIVERSAL WEB SERVICE GENERATOR",Studia Universitatis Babes-Bolyai Series Informatica, Vol.55, No.4, 2010, pp.59-69.

[18] Baker, S., & Dobson, S. (2005),"Comparing service-oriented and distributed object architectures", On the Move to Meaningful Internet Systems, 2005: CoopIS, DOA, and ODBASE, Springer Berlin Heidelberg, 2005, pp.631-645.

[19] Perepletchikov, M., Ryan, C., & Frampton, K. (2005),"Comparing the impact of service-oriented and object-oriented paradigms on the structural properties of software", On the Move to Meaningful Internet Systems 2005: OTM Workshops, Springer Berlin Heidelberg, 2005, pp.431-441.

[20] Hsu, Y. L., Lee, C. H., & Kreng, V. B. (2010),"The application of Fuzzy Delphi Method and Fuzzy AHP in lubricant regenerative technology selection", Expert Systems with Applications, Vol.37, No.1, 2010, pp.419-425.