تاریخ دریافت مقاله: ۹۵/۰۸/۱۴
تاریخ پذیرش مقاله: ۹۵/۱۰/۱۳

بهبود معماری خودالتیامی در معماری سرویس‌گرا با استفاده از منطق فازی

جواد اسدی*
دانشجوی دکتری سیستم‌های نرم‌افزاری دانشگاه آزاد اسلامی واحد قزوین
پست الکترونیکی: j.asadi@qiau.ac.ir

اسلام ناظمی
دانشیار دانشکده مهندسی و علوم کامپیوتر دانشگاه شهید بهشتی
پست الکترونیکی: nazemi@sbu.ac.ir

چکیده

در سال‌های اخیر سیستم‌های نرم‌افزاری خود تطبیق مورد توجه قابل ملاحظه‌ای قرار گرفته است. یکی از فنون خود تطبیقی، خودالتیامی است به گونه‌ای که سیستم در صورت بیرون رفتن از وضعیت سرویس‌دهی بر اثر بروز اشکال، نقص یا خطا، بدون دخالت انسان خود را ترمیم نموده و به وضعیت عادی بر می‌گردد. در روش‌های ارائه شده قبلی غالبا سیستم منتظر خرابی می‌شد و بعد خرابی اقدام به التیام می‌نمود. در روش ارائه شده در این مقاله یک معماری پیشگویانه در نظر گرفته شده است که قبل از اتفاق افتادن خرابی اقدام به بررسی و رفع عوامل ایجاد خرابی می‌کند. در این مقاله یک معماری مبتنی بر فازی برای خودالتیامی نرم‌افزار در معماری سرویس‌گرا ارائه می‌شود. معماری ارائه شده به صورت لایه‌ای است و از سه بخش اصلی، شامل بخش سرویس‌گرا، میان‌افزار ارتباطی و بخش خودالتیامی تشکیل شده است. ارزیابی معماری پیشنهادی در یک مطالعه موردی مورد بحث و بررسی قرار گرفته است. نتایج به‌دست آمده نشان می دهد که معماری پیشنهادی اثرات قابل ملاحظه ای در خودالتیامی و ترمیم سیستم‌های نرم‌افزاری داشته به‌طوری که نسبت به سیستمی که از این معماری استفاده نمی‌کند به‌طور متوسط حدود ۲۱/۵ درصد تعداد خرابی‌های اتفاق افتاده کاهش داشته است.

واژه‌های کلیدی: معماری مبتنی بر فازی، خودالتیامی، معماری سرویس‌گرا، سرویس وب، منطق فازی ،VLSI.

* نویسندهٔ مسئول

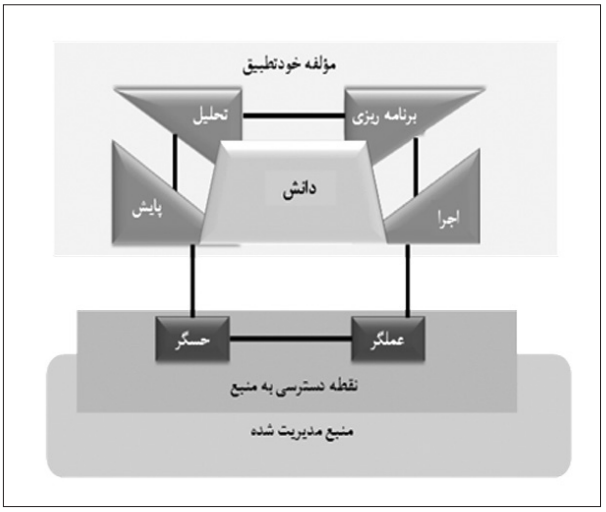
۱- مقدمه

معماری ســـرویس‌گرا، یک معماری مـــدرن برای ایجاد سیســـتم‌های نرم‌افزاری اســـت که امروزه مورد اســـتقبال قابل ملاحظه‌ای قرار گرفته اســـت. دلیل اصلی این استقبال را می‌توان در مزایای این معماری نســـبت به معماری‌های پیشین جســـتجو کرد. مستقل از بُن‌ســـازه بودن معماری سرویس‌گرا این امکان را به معمار می‌دهد که از موجودی‌ها و سیستم‌های موروثی در یک سازمان نهایت استفاده را به عمل آورد و از آن‌ها برای طراحی سیســـتم‌های جدید بهره گیری نماید. سرویس‌های وب ابزار اصلی پیاده‌سازی جهت ایجاد سیســـتم‌های نرم‌افزاری در سیستم‌های توزیع شده تحت وب هستند. ســـرویس‌های وب برای توسعه دهندگان به صورت مستقیم در سیستم‌های دیگر قابل استفاده مجدد هستند و مستقیما توسط کاربر قابل استفاده نیستند.

یک سیستم خودتطبیق سیستمی است که بدون دخالت کاربـــر بتواند بر روی مدیریت معمـــاری، عملکرد، کارایی و کنتـــرل خرابی خود تصمیم گیری کنـــد. در واقع در یک سیستم خودتطبیق معمولاً از سیستم‌های پشتیبانی تصمیم استفاده می‌شود. در یک سیستم خودتطبیق چرخه چهارگانه MAPK-E وجود دارد که عبارتند از پایش سیســـتم، تحلیل سیســـتم، برنامه ریزی و اجرا که همـــه این موارد در کنار یک پایگاه دانش انجام می‌شوند. این چرخه در شکل ۱ قابل مشاهده می باشد. [۲۰]

مهم‌ترین خصوصیات مطرح شده، چهار خصوصیتی اســـت که توسط آی‌بی‌ام به‌عنوان ابعاد اصلی سیستم‌های خـــود گردان مطرح شـــده اســـت. ایـــن ابعـــاد عبارتند از: خودپیکربندی، خود التیامی، خودبهینه سازی و حفاظت از خود[۱۸]

خـــود -* ها امـــروزه به‌عنوان یکـــی از نیازهای جدید نرم‌افزارها به‌شـــمار می‌رود. پژوهش‌های مختلفی در این باره انجام شـــده و مـــدل و معماری‌های مختلفی برای این سیستم‌ها ارائه شـــده اســـت. خودالتیام یکی از مهم‌ترین بخش‌های این سیســـتم‌ها به شـــمار می‌آید که در صورت



شکل ۱: چرخه خودتطبیقی[۲۰]

بروز نقص، خطا یا هر مشـــکل دیگری بدون دخالت انسان سیستم معیوب را به حالت عادی بر می‌گرداند.

ضرورت خودالتیامی در هر سیستم رایانه‌ای احساس می‌شود. به‌طور خاص معماری سرویس‌گرا که امروزه هم اقبال زیادی به آن وجود دارد و هم ســـازمان‌ها به ســـمت اســـتفاده از آن می‌روند بیش‌تر این نیاز مهم را احســـاس می‌کند. به همین دلیل ضرورت پرداختن به خودالتیامی در معماری سرویس‌گرا اجتناب ناپذیر است.

ار آنجایـــی که کاهش خرابی در سیســـتم‌های مختلف یکی از مهم‌ترین اهداف طراحان سیســـتم می‌باشـــد، هدف از ارائه این معمـــاری کاهش تعداد خرابی‌های اتفاق افتاده در سیســـتم اســـت. در روش‌های ارائه شده قبلی سیستم منتظر می‌شـــود تا خرابی اتفاق بیفتد و بعد از آن سعی در رفع مشکل یا خرابی دارد. در این حالت ممکن است توقف سیســـتم و عدم ســـرویس دهی آن اجتناب ناپذیر باشد و البتـــه قابلیت اطمینان و دســـترس‌پذیری آن کاهش خواهد یافت. بـــا توجه به این دلایل در این سیســـتم یک معماری ارائه می‌شود که به صورت پیشگویانه قبل از متوقف شدن روند طبیعی سیســـتم به برطرف کـــردن موانع موجود در سیســـتم می‌پردازد و سیســـتم در صورت امکان به حالت طبیعی خود برگردانده می‌شود.

بخش‌های بعدی این مقاله به صورت زیر ساختاربندی

شـــده اســـت. در بخش دوم مفاهیم و اصـــول و نیازهای معماری ســـرویس‌گرا معرفی خواهند شد. در بخش سوم معماری سرویس‌گرا، خودالتیامی و دلایل استفاده از آن ارائه می‌شود. در این بخش همچنین، کارهای مرتبط مرور خواهد شـــد. در بخش چهارم معماری فازی پیشـــنهادی ارائه شـــده و در بخش پنجم ایـــن معماری مورد ارزیابی قرار می گیرد. بخش ششم به جمع بندی نتایج و کارهای آتی اختصاص دارد .

۲- معماری سرویس‌گرا

معماری ســـرویس‌گرا رهیافتی اســـت برای ســـاخت سیســـتم‌های توزیع شـــده که کارکردهای نرم‌افزاری را در قالب سرویس ارائه می‌کند. این سرویس‌ها هم توسط دیگر نرم‌افزارها قابل فراخوانی هستند و هم برای ساخت ســـرویس‌های جدید مورد اســـتفاده قـــرار می‌گیرند، این رهیافت برای یکپارچه‌ســـازی فناوری‌هـــا در محیطی که انواع مختلفی از بُن‌ســـازه‌های نرم‌افزاری و سخت‌افزاری وجود دارد ایده‌آل است.

برای معماری ســـرویس‌گرا تعاریف متنـــوع و بعضا مختلفـــی ارائـــه شـــده که هـــر کـــدام از نگاهی بـــه تبیین خصوصیـــات آن پرداخته‌اند. بـــرای درک بهتر این مفهوم و آگاهی از کلیه برداشـــت‌ها و نگاه‌های موجود، در ادامه تعدادی از این تعاریف آورده شده است:

- چارچوبی وســـیع و استاندارد که ســـرویس‌ها در آن ســـاخته، استقرار و مدیریت می‌شوند و هدفش افزایش چابکی فناوری اطلاعات در جهت واکنش سریع به تغییرات در نیازهای کسب و کار است.[۱۹]
- ســـبکی از معماری سیســـتم‌های اطلاعاتی که هدف آن دستیابی به اتصال سست(چسبندگی کم) در ارتباطات بین مولفه‌های نرم‌افزاری است. اتصال سست (چسبندگی کم) بین مولفه‌های نرم‌افزاری باعث قابلیت استفاده مجدد آن‌ها می‌شـــود. ســـرویس در اینجا به معنای پیاده‌سازی نرم‌افزاری یک کارکرد کســـب و کار خوش تعریف اســـت که می‌تواند در فرآیندها یا نرم‌افزارهای مختلف دیگر مورد استفاده و فراخوانی قرار بگیرد.[۱۹]
- معماری ســـرویس‌گرا یـــک محصول نیســـت بلکه پلی اســـت بین حرفه و فنـــاوری به کمـــک مجموعه‌ای از سرویس‌های متکی بر فناوری که دارای قوانین، استانداردها و اصول طراحی مشخص هستند.[۱۶]
- مجموعه قوانیـــن، سیاســـت‌ها و چارچوب‌هایی که نرم‌افزارها را قادر می‌ســـازد تا عملکـــرد خود را از طریق مجموعه ســـرویس‌های مجزا و در عین حال مربوط به هم در اختیار ســـایر درخواســـت‌کنندگان قرار دهند تا بتوانند بدون اطلاع از نحوه پیاده‌ســـازی و تنها از طریق رابط‌های اســـتاندارد و تعریف شده، این ســـرویس‌ها را پیدا کرده و فراخوانی نمایند.[۱۸]
- معماری سرویس‌گرا روشی برای ساخت سیستم‌های توزیع شده‌ای است که در آن‌ها عملکرد سیستم به‌صورت ســـرویس در اختیار کاربران و یا ســـایر سرویس‌ها قرار می‌گیرد.. [۱۰،۱۲]

۳- سیستم‌های نرم افزاری خودتطبیق

در حـــال حاضر اکثر شـــرکت‌های نرم‌افـــزاری تیم‌های متخصص زیادی برای یافتن ریشه خرابی‌های سیستم‌های رایانـــه‌ای در اختیـــار دارند. برخی از خطاهایی که توســـط مشتریان گزارش می شود، ممکن است مدت‌ها یک تیم برنامه نویس را به خود مشغول کند. در بسیاری از موارد یک خطا به صورت مرموزی بدون هیچ اقدام خاصی محو می شود.

- یـــک سیســـتم خودتطبیـــق از طریق ابـــزاری مانند آزمون‌گر رگرسیون می‌تواند خطاهای محلی را تشخیص داده و ترمیم نماید. یک مؤلفه تشـــخیص خطا، با اســـتفاده از دانشـــی که در مورد سیستم کســـب می‌کند (با استفاده از ابزاری مانند شـــبکه بیزن) اطلاعات به دســـت آمده از فایل‌هـــای ثبت وقایع را درخواســـت می‌کند. به این منظور مؤلفه ممکن اســـت درخواست پایش‌های کمکی نموده و از تحلیل آن‌ها نیز برای یافتن مشـــکلات و خطاهای احتمالی

استفاده کند. سپس مؤلفه خودتطبیق اطلاعات حاصل شده را با تصحیح‌های نرم‌افزاری شـــناخته شده تطبیق داده و این تصحیح‌ها را برای رفع مشکل نصب می‌نماید.

۳-۱- خود التیامی

در سیستم‌های خودالتیام می‌توان از هوش مصنوعی بهره گرفت و گفت «سیســـتم‌های خودالتیام سیستم‌هایی هســـتند که مانند انســـان بهبود می‌یابنـــد». در واقع هدف سیســـتم‌های خودالتیام این است که نرم‌افزار بتواند مانند انسان مشکلات و خطاهای خود را برطرف کند.

در واقع یک سیســـتم زمانی خودالتیام نامیده می‌شود کـــه بتواند به شـــکل خودمختـــار مشـــکلات و خطاهای رخ‌داده در سیستم را تشـــخیص داده و اصلاح نماید. در سیســـتم‌های خودمختار و خودتطبیق توزیع شده علاوه بـــر اصلاحاتی که در هر عامل به شـــکل مجزا و انفرادی رخ می‌دهد، کل سیستم نیازمند اصلاحات ساختاری است که با همکاری تمام سیســـتم حاصل می‌شود. برای مثال هنگامی که در یک سیستم که از تعداد زیادی عامل تشکیل شده اســـت، عاملی از رده خارج می‌شود، بقیه عامل‌ها با همـــکاری یکدیگر باید ســـازوکاری را اتخاذ نمایند که اثر نبودن آن عامل در سیستم به نحوی جبران شود. این کار می‌توانـــد از طریق واگذاری نقش به یک عامل دیگر، تغییر وظایف عامل‌های موجود و غیره صورت پذیرد [۸].

۳-۲- انواع التیام

در حالت کلـــی می‌توان انواع التیام‌ها را در دو دســـته تقسیم‌بندی کرد:

- خودالتیامی: در این نوع التیام سیســـتم خود به خود و بدون نیاز به ابزار یا کمک خارجی مشکلات خود را رفع می‌کند.
- التیام به کمک دیگری: در این نوع التیام سیستم علاوه بـــر تلاش خود به محرک یا کمک کننده بیرونی برای التیام نیازمند اســـت. این سیســـتم‌ها بدون محرک یا کمک کننده خارجی نخواهند توانست مشکلات خود را برطرف کنند.

هـــر کدام از موارد فوق می‌تواننـــد از نظر زمانی هم با همدیگر متفاوت باشـــند. در مورد خودالتیامی نرم‌افزار به این نتیجه می‌توان رســـید که از نظر زمانی دو حالت ممکن است پیش آید:

- خودالتیامی سریع
- خودالتیامی با نیاز به زمان

البته باید گفت که التیام به کمک دیگری نیز این دسته‌بندی را دارد کـــه از موضوع بحث خودالتیامی خارج اســـت. در سیســـتم‌های نرم‌افزاری نیز خودالتیامی سریع یعنی بدون این‌که کاربران متوجه مشکل در سیستم شوند و با نیاز به زمان یعنی این‌که سیستم برای این‌که التیام یابد نیاز به زمان طولانی‌تری برای جایگزینی و ترمیم دارد که توقف سیستم و اطلاع کاربر اجتناب ناپذیر است. واضح است که سیستم‌های خودالتیام مطلوب مورد سیستم‌های سریع هستند اما گاهی ممکن است طولانی شدن اجتناب ناپذیر باشد.

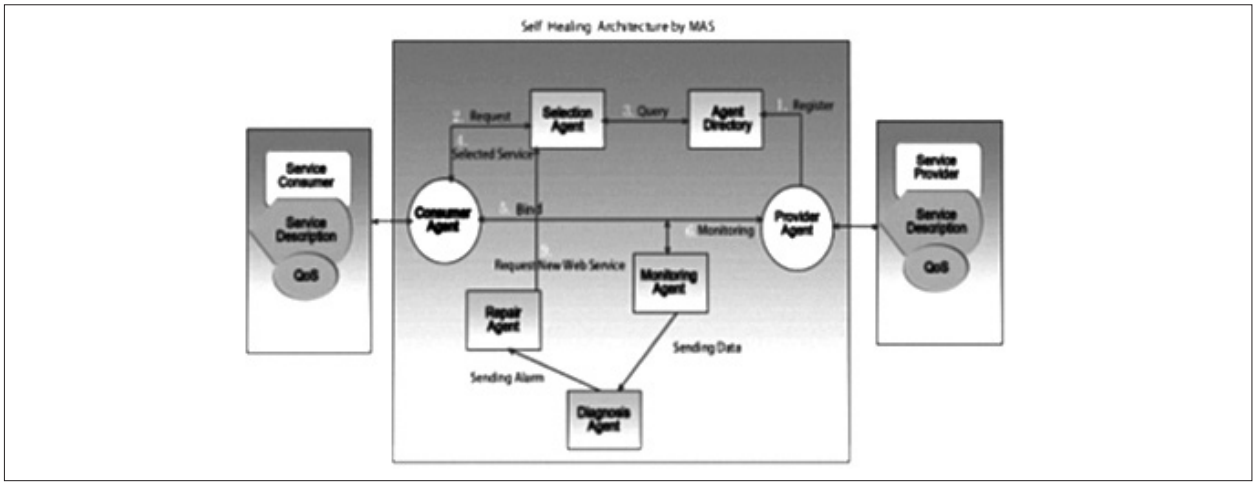
در هـــر کدام از موارد فوق ممکن اســـت بعـــد از التیام اثراتـــی از مشـــکل باقی مانده باشـــد یا هیچ اثـــری از آن نماند. در سیســـتم‌های خودالتیام بســـته به مشـــکل پیش آمده ممکن اســـت نیاز به اقدامات جبرانی باشـــد. به‌عنوان مثال اگر برداشـــت وجه از حســـاب و واریز آن به حساب دیگر در دو سیســـتم مجزا انجام شود بعد از این‌که مقدار برداشته شد سیستم دچار مشکل شود با توجه به خواص ACID[1] که در پایگاه داده جامعیت آن را تامین می‌کنند باید عمل بازیابی انجام شود و سیستم را به حالت قبل از مشکل برگرداند. بنابراین فقط التیام کافی نیست بلکه گاهی بعد از التیام نیز اقدامات جبرانی باید انجام شود.

با توجه به دســـته‌بندی فوق سیســـتم‌های نرم‌افزاری مطلوب سیســـتم‌هایی هســـتند که خودالتیام باشند، التیام سریع باشـــند و بعد از بهبود هیچ اثری از مشکل قبلی در سیستم باقی نماند.

۳-۳- ضرورت خودالتیامی در معماری سرویس‌گرا

در مقایسه با سایر سیستم‌های نرم‌افزاری، رفتارهای

1- Atomcity, Consistency, Isolation, Durability



شکل ۲: معماری ارائه شده[۱]

ناهنجـــار پیش‌بینـــی نشـــده بیشـــتری در زمـــان اجرا به سیســـتم‌های مبتنی بر سرویس تحمیل می‌شود. چند دلیل عمده زیر برای این مسئله وجود دارد:

- ماهیت توزیع‌شده محیط اجرای سرویس‌ها در بستر شبکه و تغییرات پیش‌بینی نشدهٔ آن
- اتصال و ترکیب پویای سرویس‌ها در هنگام اجرا
- مالکیت مختلف سرویس‌های تشکیل دهنده سرویس مرکب
- در معماری ســـرویس‌گرا، ســـرویس در زمان‌های مختلف قطع و وصل خواهد شد.
- سیستم‌های ســـرویس‌گرا معمولا به صورت توزیع شده بین چندین سازمان تحت اینترنت عمل می‌کنند.
- کشـــف، اتصال و ترکیب پویا در زمـــان اجرا باعث می‌شود در زمان اجرا خطاهای بیشتری رخ دهد.

در دهه اخیر، پژوهش‌های بسیاری برای غنی‌تر کردن ســـرویس‌گرایی با امکانـــات خودمدیریتی صورت گرفت و به دلیـــل اهمیتی که در بالا به آن اشـــاره شـــد، بخش عمده‌ای از این پژوهش‌ها یا پروژه‌های تحقیقاتی به جنبه خودالتیامی پرداخته‌اند[۶،۱۱].

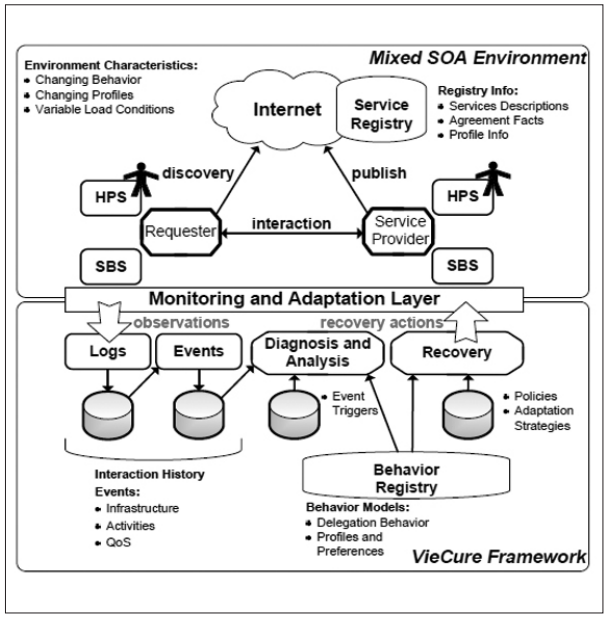
۴- پیشینه تحقیق

معماری‌های مختلفی بـــرای خودالتیامی برای معماری سرویس‌گرا ارائه شده اســـت. در معماری‌های ارائه شده در مقالات مختلف معماری‌هایـــی برای بخش‌های مختلف و برای رفع مشکلات مختلف در سیستم‌های سرویس‌گرا ارائه شده است.

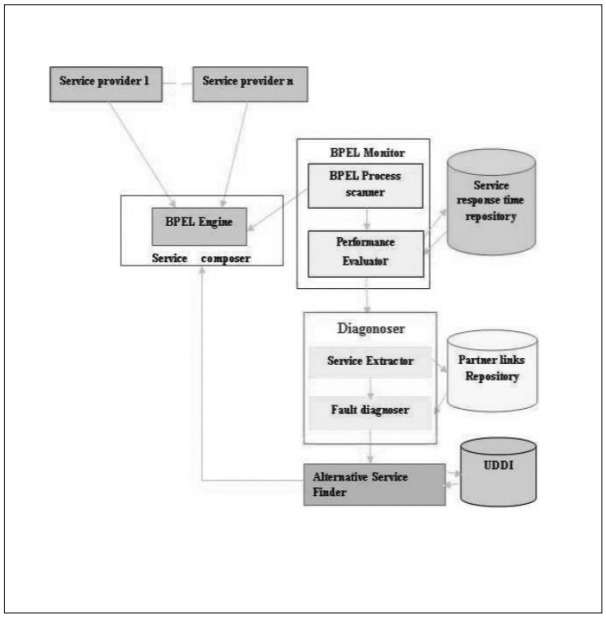
در [۱] که بر اســـاس معماری عامل‌گرا پایه‌گذاری شده اســـت بخش‌های مختلفی وجود دارد. عامل فراهم‌کننده و عامل مصرف‌کننده وظیفهٔ ارائه و اســـتفاده از سرویس‌ها را برعهده دارند. این کار به این صورت انجام می‌شود که فراهم کننده اطلاعات خود را در یک پوشـــه با نام پوشـــه عامـــل ثبت می‌کنـــد. عامل مصرف کننده نیز درخواســـت سرویس را به عامل انتخاب می‌دهد. عامل انتخاب در پوشه جســـتجو کرده و ســـرویس مورد نظر را پیدا می‌کند و به مصرف کننده می‌دهد. مصرف کننده نیز درخواست خود را برای استفاده به فراهم کننده تحویل می‌دهد.

بخش التیام این معماری نیز به این صورت عمل می‌کند که عامل نظارت بر تمام اتفاقات بین فراهم کننده و مصرف‌کننده نظارت می‌کند. عامل نظارت داده‌های خود را برای تشخیص مشکل به عامل تشخیص ارسال می‌کند. عامل تشخیص در صورت تشخیص مشکل هشـــداری به عامل ترمیم ارسال می‌کند. عامـــل ترمیم بـــرای جایگزینی ســـرویس معیوب درخواست خود را به عامل انتخاب ارسال می‌کند.

در [۲] معماری خودالتیامی برای معماری سرویس‌گرا ارائه شـــده است که از سه بخش اصلی تشکیل شده است. بخـــش اول بخـــش ســـرویس‌گرا که چرخه معروف ســـه



شکل ۳: معماری ارائه شده در [۲]



شکل ۴: معماری ارائه شده در [۳]

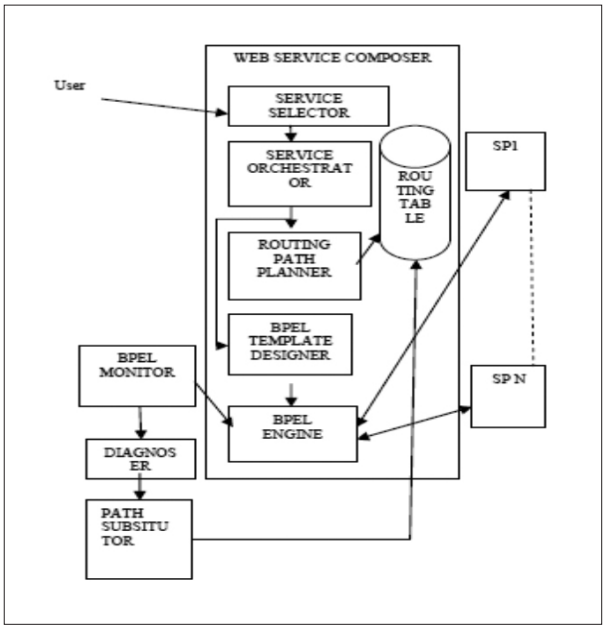
مرحله‌ای سرویس‌گرا را در بر می‌گیرد.

بخش بعدی بخش نظارت و تشـــخیص است که ارتباط بین بخش سرویس‌گرا و خودالتیامی را برقرار می‌کند.

در بخـــش ســـوم بخـــش خودالتیامی تمـــام حوادث و رویدادهای سیســـتم را ثبت می‌کنـــد و در صورت لزوم با واسطه بخش دوم التیام مورد نظر را به بخش سرویس‌گرا اعمال می‌کند.

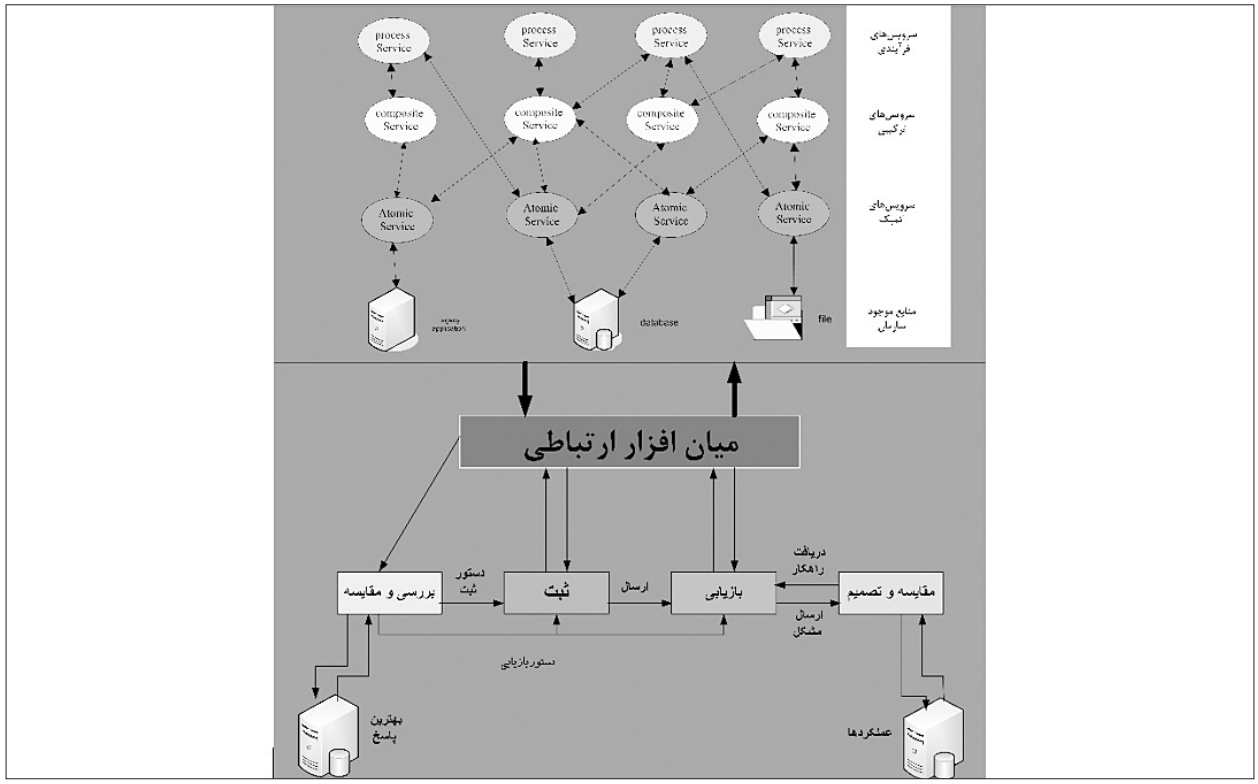
در [۳] نویســـندگان مقاله یک معماری برای جایگزینی سرویس‌های معیوب با سرویس‌های مشابه ارائه داده‌اند. در این معماری براســـاس زمان پاســـخ یک سرویس عمل می‌شود. به این صورت که اگر سرویسی در زمان میانگین ذخیره شده پاسخ نداد باید جایگزین شود. برای سرویس جایگزین بر اســـاس ورودی‌ها و خروجی‌های ســـرویس معیوب جستجو می‌شود. بعد از یافتن سرویس مورد نظر نشـــانی آن در موتور BPEL جایگزین ســـرویس معیوب می‌شـــود و سیســـتم با التیام انجام شده به کار خود ادامه می‌دهد.

در [٤] یک معماری برای التیام معماری ســـرویس‌گرا زمانی که خطایی در مســـیر شبکه اتفاق می‌افتد ارائه شده است. در حالتی که پاسخ در زمان مورد نظر از سرویسی



شکل ۵: معماری ارائه شده در [۴]

دریافت نشود بخشی از معماری مسیریابی را که اطلاعات از آن باید دریافت شـــود مورد بررســـی قرار می‌دهد. اگر این مسیریاب یا مسیریاب‌های همسایه دچار مشکل باشند مســـیر جدیدی را جستجو کرده و جایگزین مسیر معیوب می‌کند. مسیر جایگزین در سیستم به‌عنوان مسیر جدید آن سرویس جایگزین شده و تعاملات بعدی از این مسیر انجام خواهد شد.



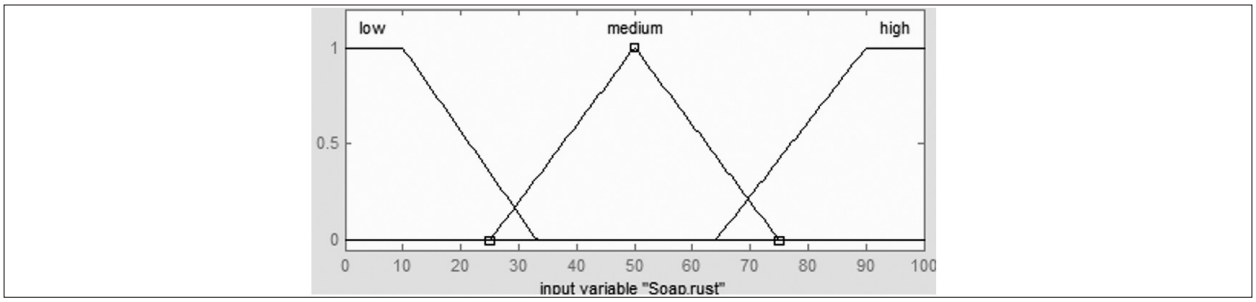
شکل۶: معماری ارائه شده برای خودالتیامی در معماری سرویس‌گرا

در معماری‌های ارائه شده قبلی سیستم خودالتیام منتظر می‌ماند تا اتفاقی در سیستم روی دهد و بعد از بروز این اتفاق اقدام به جایگزینی بخش آسیب دیده می‌کند. در صورتی که در برخی سیستم‌ها نمی‌توان کار سیستم را متوقف کرد تا مشکل برطرف شود. قابلیت اطمینان یک سیستم عبارت است از احتمال این‌که سیستم در زمان ۰ تا t درست کار کند به شرطی که در ابتدا سالم بوده است. با توجه به این تعریف اگر سرویس‌دهی سیستم متوقف شود قابلیت اطمینان آن کاهش خواهد یافت. در مورد دسترس‌پذیری نیز می‌توان گفت احتمال این‌که اگر سیستم در لحظه t قادر به پاسخگویی باشد در صورتی که در ابتدا درست بوده است. با توجه به نکات فوق دسترس‌پذیری سیستم در معماری‌های ارائه شده قبلی کاهش خواهد یافت. بنابراین برای جلوگیری از توقف سیستم در معماری ارائه شده یک روش پیشگویانه در نظر گرفته شده است که سعی می‌کند قبل از این‌که سیستم متوقف شود چاره‌ای برای مشکل پیدا کند و قبل از توقف آن را رفع کندکه این کار باعث افزایش دسترس‌پذیری و قابلیت اطمینان سیستم خواهد شد. در ادامه جزییات این معماری تشریح خواهد شد.

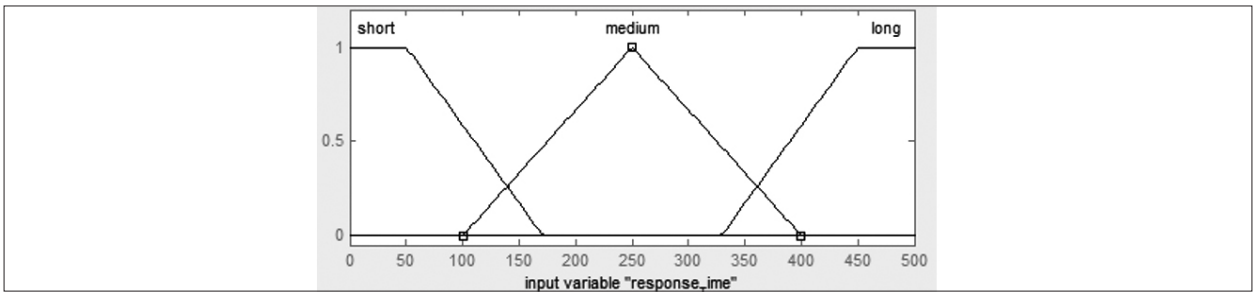
۵- معماری فازی پیشنهاد شده

در شکل ٦ معماری ارائه شده در این مقاله معرفی می‌شود. معماری ارائه شده به صورت لایه‌ای است و از سه بخش اصلی تشکیل شده است. بخش سرویس‌گرا، میان‌افزار ارتباطی و بخش خودالتیامی.

بخش‌های مختلف معماری ارائه شده در ادامه به تفصیل توضیح داده خواهند شد اما نکته حائز اهمیت این است که در این معماری برای تصمیم‌گیری در مورد عملی که بخش التیام انجام خواهد داد از منطق فازی استفاده خواهد شد. در منطق فازی از دانش بشری برای طراحی سیستم استفاده خواهد شد. به همین دلیل منطق فازی به علت استفاده از دانش بشری نزدیکی بیشتری به نظر کاربران خواهد داشت که پیاده‌سازی آن نیز در بخش‌هایی از مقاله توضیح داده خواهد شد.



شکل ۷: صحت بسته‌های SOAP



شکل ۸: زمان پاسخ

۵-۱- بخش سرویس‌گرا

بخش سرویس‌گرا چهار بخش سرویس‌های فرآیندی، سرویس‌های ترکیبی، سرویس‌های اتمیک و منابع موجود سازمان را در خود جای داده است. همان‌طور که می‌دانیم این بخش‌ها زیرســاخت معماری ســرویس‌گرا را تشکیل می‌دهنــد. در واقع معماری ســرویس‌گرا با این ســه لایه نمود پیدا می‌کند و منظور از منابع موجود سازمان فایل‌ها، پایگاه داده‌ها و نرم‌افزارهای موروثی که از قبل در سازمان وجود دارد است.

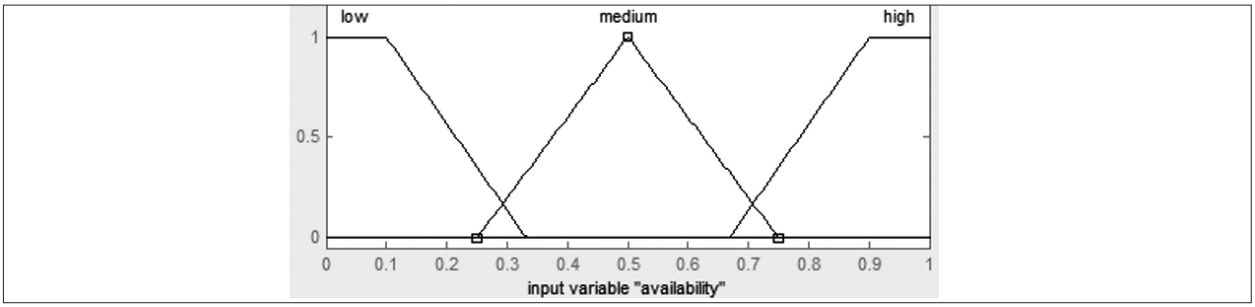
۵-۲- بخش میان‌افزار ارتباط با بخش مبتنی با سرویس

بخش میان‌افزار در واقع واســط بین دو قسمت اصلی معماری ارائه شــده خواهد بود. این بخش از یک طرف با لایه ســرویس‌گرا ارتباط خواهد داشت و از طرف دیگر با لایــهٔ التیام در ارتباط خواهد بــود. وظیفه بخش میان‌افزار بررســی کیفیت انجام کار در لایهٔ سرویس‌گرا خواهد بود. این بخــش به یک موتور فــازی مجهز اســت و اطلاعات دریافتی را به لایه التیام و به‌طور خاص به بخش بررســی وضعیت تحویل می‌دهد. علاوه بر این اعمال موارد دیگری که از طریق لایهٔ التیام به لایه ســرویس اعمال خواهد شد نیز به واســطهٔ لایهٔ میان‌افزار انجام خواهد شد. میان‌افزار ارتباطی موارد زیر را بررسی می‌کند:

- میزان صحت بسته‌های SOAP [۲] ارسالی و دریافتی
- زمان دریافت پاسخ از همنواساز
- دسترس‌پذیری یک وب سرویس
- تعداد تلاش برای ارتباط با همنواساز
- زمان ذخیره و بازیابی اطلاعات در بانک اطلاعاتی

از آنجایی که هر کدام از موارد فوق به دلیل استفاده از نظرات فرد خبره مقادیری به صورت فازی دارند متغیرهای فــازی برای آن‌هــا در نظر گرفته خواهد شــد. هر کدام از موارد فوق سه متغیر زبانی فازی دارند. صحت بسته‌های SOAP بر اســاس درصد صحت آن‌ها درنظر گرفته شده است و سه مقدار (low, medium, high) خواهد داشت. زمان دریافت پاسخ از همنواساز نیز براساس میلی‌ثانیه با حداکثر ۵۰۰ در نظر گرفته شده است و متغیرهای آن مقدار (short, medium, long) خواهند داشــت. مقدار دسترس‌پذیری یک وب سرویس کرانه ۰ تا ۱۰ خواهد داشت که سه مقدار (low, medium, high) برای آن وجود خواهد داشت. تعداد تلاش

2 -Simple Object Access Protocol



شکل ۹: دسترس‌پذیری

برای ارتباط با همنواساز (خطای عدم ارتباط) که مقدار بین ۰ و ۱۰ دارد بـا مقادیر (low, medium, high) نشـان داده خواهد شد. نهایتا برای ذخیره و بازیابی اطلاعات از بانک اطلاعاتی با کرانه ۰ تا ۵۰۰ و مقادیر (short, medium, long) اسـتفاده شده است. مقادیر فوق با توجه به نظر فرد خبره انتخاب شده است.[۷]

برای خروجـی و نتیجه حاصل از موارد فوق بازه ۰ تا ۱ و چهار مقـدار (safe, fine, log, recovery) در نظر گرفته شـده اسـت. در مجموع ۲۴۳ قانون فازی در این سیستم استفاده شده (در پیوست A بخشی از آن نشان داده شده) که توسط افراد خبره تعیین شده است. الگوی قوانین فازی برای پنج ویژگی فهرست شده به شکل زیر می باشد:

If Soap_trust is A and response_Time is B and Availibility is C and connection_error is D and register_Time is E then sys_act is F

کـه A و B و C و D و E و F اصطلاحات زبان فازی را نمایش می‌دهند. بعد از اسـتنتاج فازی نتایج به‌دست آمده عملی را که سیسـتم انجام خواهد داد را مشـخص خواهد کرد. خروجی‌های سیستم براساس وروردی‌های مختلف در شکل ۱۲ نشان داده شده است.

۵-۳- بخش خودالتیامی

بخـش خودالتیامی در بردارندۀ مهم‌ترین قسـمت این معماری برای ایجاد خودالتیامی در معماری سـرویس‌گرا می‌باشـد. این بخش از چهار مولفه اصلی و دو پایگاه داده تشکیل شده است که با همکاری هم و با واسطه میان‌افزار فـازی، خودالتیامی را بـه معماری سـرویس‌گرا اضافه می‌کنند. مولفه‌های تشـکیل دهنده شامل این بخش شامل بررسـی وضعیت و مقایسـه، ثبت، بازیابی و مقایسـه و تصمیم اسـت که در ادامه هرکدام از این بخش‌ها توضیح داده خواهند شد.

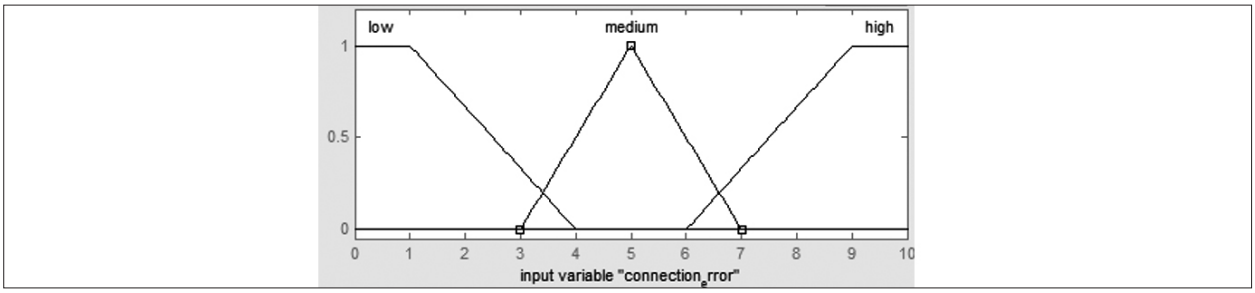
۵-۳-۱- مولفه بررسی وضعیت فعلی و مقایسه

در ایـن مولفه کـه مهم‌ترین مولفه بخـش خودالتیامی می‌باشـد اطلاعات دریافت شده از میان‌افزار مورد ارزیابی قرار گرفته و براسـاس این داده‌ها تصمیمات مختلفی اتخاذ خواهد شد.[۵] این مولفه با توجه به الگوریتم زیر عمل می‌کند

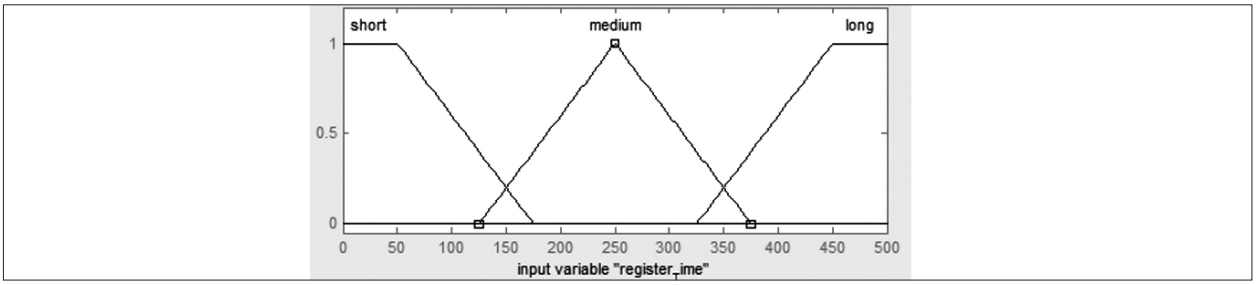
```
If (fuzzy_output is safe)
 system is safe and store new Threshold;
Else if (fuzzy_output is fine)
 System is fine but not favorite;
Else if (fuzzy_output is log)
 Create log file for future use;
Else if (fuzzy_output is recovery)
 Start recovery before system fault;
Else
 System failed;
```

برای هر کدام از مواردی که توسط میان‌افزار بررسی می‌شـود الگوریتمی مشـابه الگوریتم فوق وجود خواهد داشت که براساس این الگوریتم عمل خواهد شد.

زمان‌های آسـتانه بر اساس کارکردهای قبلی سیستم به‌دسـت می‌آید که می‌تواند میانگینی از زمان‌های اجرای مطلوب، خوب، متوسط و نه چندان خوب باشد. این زمان‌ها در پایگاه داده‌ای به نام بهترین پاسـخ ذخیره می‌شود. اگر خروجی سیسـتم safe باشد سیستم به‌درستی کار می‌کند و هیچ مشـکلی وجود ندارد و البته زمان پاسـخ جدید یک زمان جدید و در واقع یک رکورد جدید اسـت که در پایگاه داده بهترین پاسخ ذخیره می‌شود. اگر خروجی fine باشد



شکل ۱۰: تعداد تلاش برای ارتباط



شکل ۱۱: زمان ذخیره و بازیابی

سیستم درست کار می‌کند ولی حالت سیستم مطلوب نیست. اگر سیستم خروجی log را بدهد سیستم به کار خود ادامه می‌دهد و برای پیشگیری از خطرات احتمالی بعدی شروع به log گرفتن از سیستم می‌کند. نهایتا وقتی خروجی recovery است سیستم برای رفع مشکل سیستم اقداماتی انجام می‌دهد و قبل از توقف سیستم، شروع به رفع مشکل آن خواهد کرد. نکته حائز اهمیت معماری پیشنهادی این است که سیستم خودالتیام منتظر نمی‌شود تا خرابی اتفاق بیفتد و بعد از آن شروع به التیام کند بلکه قبل از این‌که سیستم به آن نقطه برسد شروع به بازیابی سیستم می‌کند و حالت پیشگویانه دارد که این کار می‌تواند با جایگزینی یک سرویس، جایگزینی یک سیستم، هشدار به کاربر سیستم و کارهای مختلف دیگر انجام شود.

۵-۳-۲- مولفه ثبت

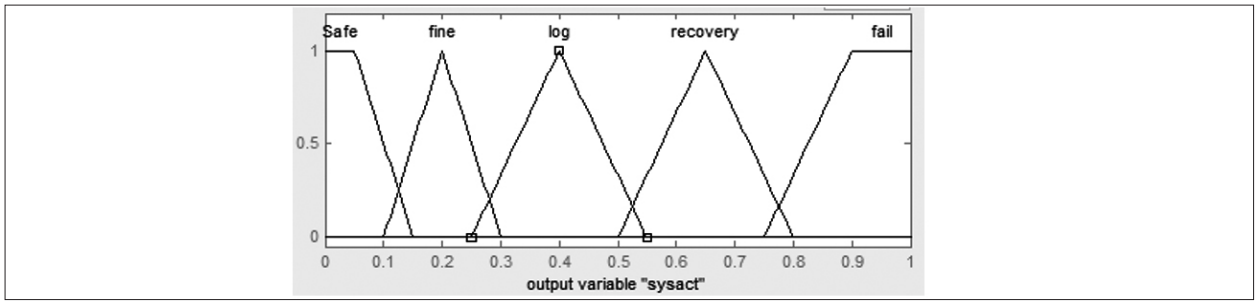
این مولفه وظیفه ثبت رویدادهای اتفاق افتاده در سیستم را دارد. وقتی سیستم خودالتیام احساس خطر می‌کند نیاز می‌بیند که وضعیت سیستم را برای بازیابی احتمالی ذخیره کند به همین دلیل به مولفه ثبت دستور می‌دهد با کمک میان‌افزار شروع به ثبت وضعیت سیستم کند تا در صورت بروز مشکل احتمالی از اطلاعات ثبت شده استفاده شود. اطلاعات ثبت شده این بخش در صورت ارسال دستور به مولفه بازیابی برای بازیابی سیستم برای آن مولفه ارسال خواهد شد.

۵-۳-۳- مولفه بازیابی

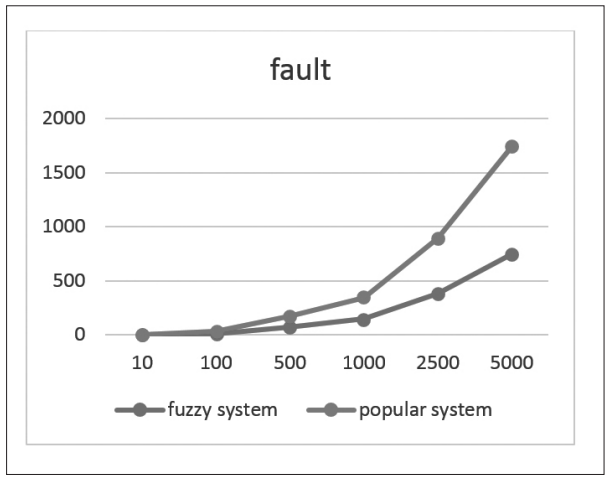
این مولفه در صورت بروز مشکل در سیستم آن را به حالت امن برخواهد گرداند. بعد از این‌که وضعیت در بخش بررسی وضعیت و مقایسه مورد ارزیابی قرار می‌گیرد در صورت نیاز با توجه به الگوریتم ممکن است دستور بازیابی به این بخش ارسال شود. این مولفه اطلاعات ثبت شده در بخش ثبت را دریافت کرده و برای دریافت راهکار اجرایی مشکل در حال اتفاق افتادن را به بخش مقایسه و تصمیم ارسال می‌کند. پس از دریافت راهکار با واسطه میان‌افزار ارتباطی اعمال بازیابی را روی سیستم سرویس‌گرا اعمال می‌کند.

۵-۴- مولفه مقایسه و تصمیم

در این معماری با توجه به مشکلات روی داده تصمیمات مختلفی اتخاذ می‌شود. راهکارهای پیش‌بینی شده برای مشکلات مختلف در پایگاه داده عملکردها ذخیره شده است. بخش بازیابی مشکلات دریافتی از بخش بررسی وضعیت را برای بخش مقایسه و تصمیم ارسال می‌کند و این بخش با مقایسه آن‌ها با اطلاعات موجود در



شکل ۱۲: خروجی فازی



نمودار ۱: مقایسه تعداد خرابی در دو سیستم

پایگاه داده راهکار اجرایی مورد اجرا را اســتخراج کرده و برای بخش بازیابی ارسال می‌کند.

۶- آزمایش‌ها

برای بررســی این‌که سیســتم ارائه شــده تا چه حد توانسته اســت وضعیت سیســتم را بهبود بخشد روش ارائه شــده در این مقاله در نرم‌افزار متلب نســخهٔ ۲۰۰۹ و با اســتفاده از جعبه ابزار فازی متلب در سیســتمی با پردازنده Intel corei5/2.67GHz و RAM 4G شبیه‌ســازی شده اســت. برای بررسی این‌که آیا تعداد خرابی‌هایی که در این سیســتم اتفاق می‌افتد تا چه حد کاهش یافته است در این سیســتم داده‌های ورودی را با ۱۰۰، ۵۰۰، ۱۰۰۰، ۲۵۰۰ و ۵۰۰۰ حالــت مورد بررســی قــرار داده‌ایم. در این حالت‌ها سیســتم با مقادیر تصادفی برای ورودی‌ها آزمایش شده‌اند. تعداد خرابی‌ها با سیستم معمولی که به صورت پیشگویانه عمل نمی‌کند بدون استفاده از معماری ارائه شده در این مقاله با معماری ارائه شده در این مقاله با همدیگر مقایسه شده‌اند. در این مقایسه هدف این بوده اســت که تعداد خرابی‌ها در دو سیستم در حالت یکسان مورد ارزیابی قرار گیرد و مقایســه شود که آیا سیستم ارائه شده توانســته است از خرابی در سیستم پیشگیری کند؟ نتایج به‌دســت آمده در نمودار ۱ نشــان داده شده است.

همان‌طور که در نمودار مشخص است معماری ارائه شده بهبود بســیار زیادی در کاهش خرابی‌های سیستم داشته است. علت این بهبود این است که در معماری ارائه شــده با log و Recovery از رســیدن سیســتم به مرحله Failure پیشگیری می‌شود و سیستم کمتر با خرابی مواجه می‌شــود. این به این علت است که تا سیستم در مرحله‌ای قــرار می‌گیرد که ممکن اســت به ســمت خرابی برود با Recovery جلوی این کار گرفته شــده و سیستم به حالت مطلوب بر می‌گردد. به‌طور متوسط معماری ارائه شده در این مقاله حدود ۲۱/۵ درصد تعداد خرابی‌های اتفاق افتاده را کاهش می‌دهد.

۷- نتایج

روش‌های ارائه شده در مقالات مختلف هر کدام بخشی از خودالتیامی در معماری سرویس‌گرا را در بر دارند. این روش‌ها معمولا نمی‌توانند مرجع کاملی برای خودالتیامی در معماری سرویس‌گرا باشند. از این رو سعی شد کمبودها و نواقص این روش‌ها مورد ارزیابی قرار گرفته و بر اساس رویکرد رفع نواقص در این روش‌ها گام برداشته شود.

نتایج به‌دست آمده نشان می دهد که معماری پیشنهادی اثرات قابل ملاحظه ای در خودالتیامی و ترمیم سیستم‌های نرم‌افزاری داشـــته به‌طوری که نســـبت به سیستمی که از این معماری اســـتفاده نمی‌کند به‌طور متوسط حدود ۲۱/۵ درصد تعداد خرابی‌های اتفاق افتاده کاهش داشته است.

به‌عنوان کار آتی که برای معماری سرویس‌گرا می‌توان انجـــام داد ارائه یـــک معماری برای خـــود-* در معماری سرویس‌گرا می‌باشـــد که تمام موارد از جمله خودالتیامی را در بـــر می‌گیرد. این معماری می‌توانـــد به‌عنوان مرجع خود-* در معماری سرویس‌گرا مطرح باشد.

۸- منابع

1) Mahmoud Hossein Zadeh, Mir Ali Seyyedi," A Self-Healing Architecture for Web Services based on Failure Prediction and a Multi Agent System", 2011 IEEE

2) Harald Psaier, Florian Skopik, Daniel Schall, Schahram Dustdar, Behavior Monitoring in Self-healing Service-oriented Systems, 2010 IEEE 34th Annual Computer Software and Applications Conference

3) S.Poonguzhali, R.Sunitha, Dr.G.Aghila," Self-Healing in Dynamic Web Service Composition", International Journal on Computer Science and Engineering (IJCSE), Vol. 3 No. 5 May 2011

4) S.Poonguzhali, L.JerlinRubini, S.Divya, "A Self-Healing Approach for Service Unavailability in Dynamic Web Service Composition", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014

5) Amal Alhosban and Erfan Najmi and Khayyam Hashmi and Zaki Malik, "Automated Self-healing Framework for Service-Oriented Systems", 2013 IEEE

6) Mohammad Hadi Valipour, Bavar Amirzafari, Khashayar Niki Maleki and Negin Daneshpour, "A Brief Survey of Software Architecture Concepts and Service Oriented Architecture", 2009

7) Hei-Chia Wang, Chang-Shing Lee, Tsung-Hsien Ho, "Combining subjective and objective QoS factors for personalized web service selection", 2007

8) Romina Torres and Hern´an Astudillo, Rodrigo Salas, "Self-adaptive Fuzzy QoS-driven Web service discovery", 2011

9) Ali Yavari, Maryam Musavi, Hossein Momeni, Mahnaz Hamzehnia, "Measuring the Failure Rate in Service-Oriented Architecture Using Fuzzy Logic", 2013

10) Vassiliki Diamadopoulou, Christos Makrisa, Yannis Panagis, Evangelos Sakkopoulos, "Techniques to support Web Service selection and consumption with QoS characteristics", 2008

11) Dunlu Peng Qingkui Chen Huan Huo,"A Fuzzy Partial Ordering Approach for QoSbased Selection of Web Services", 2010

12) Dunlu Peng Qingkui Chen Huan Huo,"A Fuzzy Partial Ordering Approach for QoSbased Selection of Web Services", 2010

13) Hongxia Tong Shensheng Zhang, "A Fuzzy Multi-Attribute Decision Making Algorithm for Web Services Selection Based on QoS", 2006

14) N. Hema Priya, S.Chandramathi,"QoS Based Optimal Selection of Web Services Using Fuzzy Logic", 2012

۱۵- استوارت راســل و پیتر نورویگ، کتاب هوش مصنوعی، ترجمه مهندس عین الله جعفرنژاد قمی، ویراست سوم ۲۰۱۰

۱۶- فریدون شـــمس، امیـــر مهجوریـــان، «معرفی اصول، مبانـــی و روش‌های معماری سازمانی سرویس‌گرا»، مرکز چاپ و انتشارات دانشگاه شهید بهشتی، ۱۳۸۹

۱۷- امیر رضا مهجوریان، «مقاله معماری ســـازمانی ســـرویس‌گرا چیســـت؟»، بهمن ۸۷

۱۸- زهرا نخعی، «خودالتیامی در معماری ســـرویس‌گرا»، ســـمینار کارشناسی ارشد دانشگاه شهید بهشتی تهران، ۱۳۹۰

۱۹- جواد اسدی، «ارائه مدلی برای بهبود عملکرد دولت الکترونیک با استفاده از معماری ســـرویس‌گرا» پایان نامه کارشناســـی ارشد، دانشـــگاه آزاد اسلامی واحد قزوین ۱۳۹۱

۲۰- امین شکری، «مقاله خود التیامی در سیستم های خودتطبیق»، پاییز ۱۳۹۱

پیوست A

	Soap_trust	response_Time	availibility	connection_error	register_Time	result
1	Low	Short	Low	Low	Short	log
2	Low	Short	Low	Low	Medium	log
3	Low	Short	Low	Low	Long	recovery
4	Low	Short	Low	Medium	Short	log
5	Low	Short	Low	Medium	Medium	recovery
6	Low	Short	Low	Medium	Long	recovery
7	Low	Short	Low	High	Short	recovery
8	Low	Short	Low	High	Medium	recovery

9	Low	Short	Low	High	Long	failure
10	Low	Short	Medium	Low	Short	fine
11	Low	Short	Medium	Low	Medium	fine
12	Low	Short	Medium	Low	Long	log
13	Low	Short	Medium	Medium	Short	log
14	Low	Short	Medium	Medium	Medium	log
15	Low	Short	Medium	Medium	Long	recovery
16	Low	Short	Medium	High	Short	log
17	Low	Short	Medium	High	Medium	recovery
18	Low	Short	Medium	High	Long	recovery
19	Low	Short	High	Low	Short	safe
20	Low	Short	High	Low	Medium	fine
21	Low	Short	High	Low	Long	log
22	Low	Short	High	Medium	Short	fine
23	Low	Short	High	Medium	Medium	fine
24	Low	Short	High	Medium	Long	log
25	Low	Short	High	High	Short	log
26	Low	Short	High	High	Medium	log
27	Low	Short	High	High	Long	recovery
28	Low	Medium	Low	Low	Short	log
29	Low	Medium	Low	Low	Medium	recovery
30	Low	Medium	Low	Low	Long	recovery
31	Low	Medium	Low	Medium	Short	log
32	Low	Medium	Low	Medium	Medium	recovery
33	Low	Medium	Low	Medium	Long	failure
34	Low	Medium	Low	High	Short	recovery
35	Low	Medium	Low	High	Medium	failure
36	Low	Medium	Low	High	Long	failure
37	Low	Medium	Medium	Low	Short	log
38	Low	Medium	Medium	Low	Medium	log
39	Low	Medium	Medium	Low	Long	recovery
40	Low	Medium	Medium	Medium	Short	log
41	Low	Medium	Medium	Medium	Medium	recovery
42	Low	Medium	Medium	Medium	Long	recovery
43	Low	Medium	Medium	High	Short	recovery
44	Low	Medium	Medium	High	Medium	recovery
45	Low	Medium	Medium	High	Long	failure
46	Low	Medium	High	Low	Short	fine
47	Low	Medium	High	Low	Medium	fine

48	Low	Medium	High	Low	Long	log
49	Low	Medium	High	Medium	Short	fine
50	Low	Medium	High	Medium	Medium	log
51	Low	Medium	High	Medium	Long	recovery
52	Low	Medium	High	High	Short	log
53	Low	Medium	High	High	Medium	recovery
54	Low	Medium	High	High	Long	recovery
55	Low	Long	Low	Low	Short	recovery
56	Low	Long	Low	Low	Medium	recovery
57	Low	Long	Low	Low	Long	failure
58	Low	Long	Low	Medium	Short	recovery
59	Low	Long	Low	Medium	Medium	failure
60	Low	Long	Low	Medium	Long	failure
61	Low	Long	Low	High	Short	failure
62	Low	Long	Low	High	Medium	failure
63	Low	Long	Low	High	Long	failure
64	Low	Long	Medium	Low	Short	log
65	Low	Long	Medium	Low	Medium	recovery
66	Low	Long	Medium	Low	Long	recovery
67	Low	Long	Medium	Medium	Short	recovery
68	Low	Long	Medium	Medium	Medium	recovery
69	Low	Long	Medium	Medium	Long	failure
70	Low	Long	Medium	High	Short	recovery
71	Low	Long	Medium	High	Medium	failure
72	Low	Long	Medium	High	Long	failure
73	Low	Long	High	Low	Short	log
74	Low	Long	High	Low	Medium	log
75	Low	Long	High	Low	Long	recovery
76	Low	Long	High	Medium	Short	log
77	Low	Long	High	Medium	Medium	recovery
78	Low	Long	High	Medium	Long	recovery
79	Low	Long	High	High	Short	recovery
80	Low	Long	High	High	Medium	recovery
81	Low	Long	High	High	Long	failure
82	Medium	Short	Low	Low	Short	fine
83	Medium	Short	Low	Low	Medium	fine
84	Medium	Short	Low	Low	Long	log
85	Medium	Short	Low	Medium	Short	log
86	Medium	Short	Low	Medium	Medium	log

87	Medium	Short	Low	Medium	Long	recovery
88	Medium	Short	Low	High	Short	log
89	Medium	Short	Low	High	Medium	recovery
90	Medium	Short	Low	High	Long	recovery
91	Medium	Short	Medium	Low	Short	safe
92	Medium	Short	Medium	Low	Medium	fine
93	Medium	Short	Medium	Low	Long	log
94	Medium	Short	Medium	Medium	Short	fine
95	Medium	Short	Medium	Medium	Medium	fine
96	Medium	Short	Medium	Medium	Long	log
97	Medium	Short	Medium	High	Short	log
98	Medium	Short	Medium	High	Medium	log
99	Medium	Short	Medium	High	Long	recovery
100	Medium	Short	High	Low	Short	safe
101	Medium	Short	High	Low	Medium	safe
102	Medium	Short	High	Low	Long	fine
103	Medium	Short	High	Medium	Short	safe
104	Medium	Short	High	Medium	Medium	fine
105	Medium	Short	High	Medium	Long	log
106	Medium	Short	High	High	Short	fine
107	Medium	Short	High	High	Medium	log
108	Medium	Short	High	High	Long	log
109	Medium	Medium	Low	Low	Short	log
110	Medium	Medium	Low	Low	Medium	log
111	Medium	Medium	Low	Low	Long	recovery
112	Medium	Medium	Low	Medium	Short	log
113	Medium	Medium	Low	Medium	Medium	recovery
114	Medium	Medium	Low	Medium	Long	recovery
115	Medium	Medium	Low	High	Short	recovery
116	Medium	Medium	Low	High	Medium	recovery
117	Medium	Medium	Low	High	Long	failure
118	Medium	Medium	Medium	Low	Short	fine
119	Medium	Medium	Medium	Low	Medium	log
120	Medium	Medium	Medium	Low	Long	log
121	Medium	Medium	Medium	Medium	Short	log
122	Medium	Medium	Medium	Medium	Medium	log
123	Medium	Medium	Medium	Medium	Long	recovery
124	Medium	Medium	Medium	High	Short	log
125	Medium	Medium	Medium	High	Medium	recovery

126	Medium	Medium	Medium	High	Long	recovery
127	Medium	Medium	High	Low	Short	safe
128	Medium	Medium	High	Low	Medium	fine
129	Medium	Medium	High	Low	Long	log
130	Medium	Medium	High	Medium	Short	fine
131	Medium	Medium	High	Medium	Medium	fine
132	Medium	Medium	High	Medium	Long	log
133	Medium	Medium	High	High	Short	log
134	Medium	Medium	High	High	Medium	log
135	Medium	Medium	High	High	Long	recovery
136	Medium	High	Low	Low	Short	log
137	Medium	High	Low	Low	Medium	recovery
138	Medium	High	Low	Low	Long	recovery
139	Medium	High	Low	Medium	Short	recovery
140	Medium	High	Low	Medium	Medium	recovery
141	Medium	High	Low	Medium	Long	failure
142	Medium	High	Low	High	Short	recovery
143	Medium	High	Low	High	Medium	failure
144	Medium	High	Low	High	Long	failure
145	Medium	High	Medium	Low	Short	log
146	Medium	High	Medium	Low	Medium	log
147	Medium	High	Medium	Low	Long	recovery
148	Medium	High	Medium	Medium	Short	log
149	Medium	High	Medium	Medium	Medium	recovery
150	Medium	High	Medium	Medium	Long	recovery
151	Medium	High	Medium	High	Short	recovery
152	Medium	High	Medium	High	Medium	recovery
153	Medium	High	Medium	High	Long	failure
154	Medium	High	High	Low	Short	fine
155	Medium	High	High	Low	Medium	log
156	Medium	High	High	Low	Long	log
157	Medium	High	High	Medium	Short	log
158	Medium	High	High	Medium	Medium	log
159	Medium	High	High	Medium	Long	recovery
160	Medium	High	High	High	Short	log
161	Medium	High	High	High	Medium	recovery
162	Medium	High	High	High	Long	recovery
163	High	Short	Low	Low	Short	fine
164	High	Short	Low	Low	Medium	fine

165	High	Short	Low	Low	Long	log
166	High	Short	Low	Medium	Short	fine
167	High	Short	Low	Medium	Medium	log
168	High	Short	Low	Medium	Long	log
169	High	Short	Low	High	Short	log
170	High	Short	Low	High	Medium	log
171	High	Short	Low	High	Long	recovery
172	High	Short	Medium	Low	Short	safe
173	High	Short	Medium	Low	Medium	safe
174	High	Short	Medium	Low	Long	fine
175	High	Short	Medium	Medium	Short	safe
176	High	Short	Medium	Medium	Medium	fine
177	High	Short	Medium	Medium	Long	log
178	High	Short	Medium	High	Short	fine
179	High	Short	Medium	High	Medium	log
180	High	Short	Medium	High	Long	log
181	High	Short	High	Low	Short	safe
182	High	Short	High	Low	Medium	safe
183	High	Short	High	Low	Long	safe
184	High	Short	High	Medium	Short	safe
185	High	Short	High	Medium	Medium	safe
186	High	Short	High	Medium	Long	fine
187	High	Short	High	High	Short	safe
188	High	Short	High	High	Medium	fine
189	High	Short	High	High	Long	log
190	High	Medium	Low	Low	Short	fine
191	High	Medium	Low	Low	Medium	log
192	High	Medium	Low	Low	Long	log
193	High	Medium	Low	Medium	Short	log
194	High	Medium	Low	Medium	Medium	log
195	High	Medium	Low	Medium	Long	recovery
196	High	Medium	Low	High	Short	log
197	High	Medium	Low	High	Medium	log
198	High	Medium	Low	High	Long	recovery
199	High	Medium	Medium	Low	Short	safe
200	High	Medium	Medium	Low	Medium	fine
201	High	Medium	Medium	Low	Long	log
202	High	Medium	Medium	Medium	Short	fine
203	High	Medium	Medium	Medium	Medium	log
204	High	Medium	Medium	Medium	Long	log

205	High	Medium	Medium	High	Short	log
206	High	Medium	Medium	High	Medium	log
207	High	Medium	Medium	High	Long	recovery
208	High	Medium	High	Low	Short	safe
209	High	Medium	High	Low	Medium	safe
210	High	Medium	High	Low	Long	fine
211	High	Medium	High	Medium	Short	safe
212	High	Medium	High	Medium	Medium	fine
213	High	Medium	High	Medium	Long	log
214	High	Medium	High	High	Short	fine
215	High	Medium	High	High	Medium	log
216	High	Medium	High	High	Long	log
217	High	High	Low	Low	Short	log
218	High	High	Low	Low	Medium	log
219	High	High	Low	Low	Long	recovery
220	High	High	Low	Medium	Short	log
221	High	High	Low	Medium	Medium	recovery
222	High	High	Low	Medium	Long	recovery
223	High	High	Low	High	Short	recovery
224	High	High	Low	High	Medium	Recovery
225	High	High	Low	High	Long	Failure
226	High	High	Medium	Low	Short	Fine
227	High	High	Medium	Low	Medium	Log
228	High	High	Medium	Low	Long	Log
229	High	High	Medium	Medium	Short	Log
230	High	High	Medium	Medium	Medium	Log
231	High	High	Medium	Medium	Long	Recovery
232	High	High	Medium	High	Short	Log
233	High	High	Medium	High	Medium	Recovery
234	High	High	Medium	High	Long	Recovery
235	High	High	High	Low	Short	Fine
236	High	High	High	Low	Medium	Fine
237	High	High	High	Low	Long	Log
238	High	High	High	Medium	Short	Log
239	High	High	High	Medium	Medium	Log
240	High	High	High	Medium	Long	Log
241	High	High	High	High	Short	Log
242	High	High	High	High	Medium	Log
243	High	High	High	High	Long	Log