معرفی روشی مبتنی بر نیازمندی‌ها جهت آزمون معماری نرم‌افزار

سیدمهران شرفی

چکیده: در این مقاله ضمن معرفی روش‌های متداول مبتنی بر سناریو در ارزیابی معماری نرم‌افزار و بیان نقاط ضعف و قوت آنها، رویکرد متفاوتی برای شناسایی نقایص معماری ارائه می‌شود. در روش پیشنهادی مشکلات تهدیدکننده سیستم توسط سهام‌داران فهرست می‌شوند و با تحلیل نقص‌های احتمالی که می‌توانند مسبب بروز آن مشکلات باشند، خطاهای موجود در سطوح مختلف به ویژه در سطح معماری نرم‌افزار کشف می‌گردند. نتایج به کارگیری عملی روش پیشنهادی نشان می‌دهد که این روش می‌تواند در آشکارنمودن نقص‌هایی که ممکن است از حوزه تأثیر روش‌های دیگر مصون مانده باشند، مؤثر باشد. لذا از این روش می‌توان هم برای آزمون معماری و هم به عنوان یک رویه تکمیلی در کنار روش‌های ارزیابی معماری نرم‌افزار جهت شناسایی نقایص و اصلاح معماری استفاده نمود. روش پیشنهادی و اجزای آن در یک قالب سیستماتیک معرفی شده و نتایج به کارگیری آن بر روی یک سیستم واقعی ارائه می‌گردد.

کلید واژه: **معماری نرم‌افزار، مشکل، نقص معماری، ارزیابی مبتنی بر سناریو.**

۱- مقدمه

معماری نرم‌افزار امروزه به عنوان یکی از مهم‌ترین شاخه‌های مهندسی نرم‌افزار مطرح است که شامل رویکردهایی جهت مدیریت پیچیدگی نرم‌افزار می‌باشد. معماری نرم‌افزار بنا به تعریف عبارت است از مؤلفه‌های اصلی تشکیل‌دهنده نرم‌افزار، اتصالات بین مؤلفه‌ها و رفتارهای بیرونی مؤلفه‌ها. ابعاد کیفی که طراحی کلان سیستم به دنبال خواهد داشت متأثر از نحوه ترکیب مؤلفه‌ها و اتصالات است. به عبارت دیگر معماری می‌تواند عامل یا مانع دستیابی به صفات کیفی نرم‌افزار باشد. امروزه ثابت شده که دستیابی به یک ویژگی کیفی وابستگی زیادی به تصمیماتی دارد که در سطح معماری گرفته می‌شود. به عنوان مثال اگر فاکتور کارایی[1] برای یک سیستم مهم باشد، ساختار کلی نرم‌افزار، ارتباط بین مؤلفه‌ها و زمان سرویس‌دهی هر مؤلفه و یا مؤلفه‌های موازی باید به گونه‌ای طراحی شوند که این نیاز را تأمین نمایند، یا مثلاً اگر قابلیت تغییر برای یک سیستم مهم باشد، توجه به خاصیت محصورسازی مؤلفه‌ها از اهمیت زیادی برخوردار است. حتی بعضی از خصوصیاتی که به نظر می‌رسد دسترسی به آنها بی‌ارتباط با معماری باشد مانند قابلیت کاربری[2]، امروزه ثابت شده است که ارتباط مستقیم با معماری دارد. به عنوان مثال طراحی مؤلفه‌های رفع و رجوع‌کننده استثنائات[3] و یا امکان لغو عملیات که در تقویت قابلیت کاربری مؤثر هستند از تصمیمات معماری به شمار می‌روند [۱].

از سوی دیگر توصیفات معماری نرم‌افزار این قابلیت را دارند که به کمک آنها می‌توان ویژگی‌های کیفی نرم‌افزار را پیش‌بینی نمود. ارزیابی معماری نرم‌افزار به دلیل این که امکان کشف نقایص را خیلی زودتر از مرحله پیاده‌سازی نرم‌افزار می‌دهد از صرف هزینه‌های زیاد تعمیرات در مراحل انتهایی چرخه حیات نرم‌افزار جلوگیری می‌کند. از نتایج حاصل از ارزیابی معماری می‌توان از جوانب مختلفی بهره‌برداری نمود. اولاً پس از ارزیابی معماری می‌توان در خصوص توانمندی معماری در جوابگویی به نیازهای وظیفه‌مندی و غیر وظیفه‌مندی قضاوت نمود، ثانیاً اگر دو یا چند معماری کاندید مورد ارزیابی قرار گرفته باشند می‌توان گزینه مناسب‌تر برای سیستم مفروض را انتخاب نمود و ثالثاً راهکارهایی برای اعمال تغییرات در معماری به منظور انطباق بیشتر آن با نیازهای سیستم آشکار می‌گردد [۲].

طراحی اولیه معماری و یا اعمال تغییر در معماری موضوعی بسیار حائز اهمیت است و به پارامترهای مختلفی از جمله مهارت و تجربه معمار بستگی دارد. اکثر ویژگی‌های کیفی نرم‌افزار با هم در تضاد هستند و یک تصمیم معماری که در تقویت یک ویژگی کیفی مؤثر است ممکن است باعث تضعیف ویژگی دیگر شود (مثلاً کارایی و دسترسی‌پذیری[4]). یکی از معروف‌ترین روش‌های سیستماتیک طراحی معماری نرم‌افزار روش ADD[5] [۱] است. ADD را می‌توان توسعه‌یافته روش دیگر تولید نرم‌افزار مانند RUP[6] [۳] قلمداد کرد [۱]. RUP دارای مراحل مختلفی است که به طراحی سطح بالای معماری منتهی می‌شوند و پس از آن، ادامه کار در طراحی تفصیلی و پیاده‌سازی دنبال می‌شود. به کار بردن ADD در این فرایند، مراحل مربوط به طراحی سطح بالا را ارتقا می‌بخشد و به آن فرم سیستماتیک می‌دهد [۳]. ADD روشی است که در آن ویژگی‌های کیفی، تعیین‌کننده نحوه تجزیه ماژول‌ها و نهایتاً ایجاد معماری نرم‌افزار هستند.

نکته مطرح این است که سهام‌داران پروژه و به خصوص مشتری هستند که از معنای واقعی یک ویژگی کیفی در یک دامنه خاص اطلاع لازم را دارند. روش‌های طراحی و یا ارزیابی معماری باید خواسته‌های سهام‌داران را مبنای تعریف ویژگی‌های کیفی قرار دهند چون در غیر این صورت ممکن است تصمیماتی اتخاذ گردد که انطباق لازم را با نظرات سهام‌داران نداشته باشد و ابعاد حساس سیستم را که سهام‌داران به آن وقوف لازم دارند، به طور کامل پوشش ندهد. در بیشتر روش‌های ارزیابی معماری مخصوصاً روش‌های مبتنی بر سناریو مانند ATAM[7] [۴] و CBAM[8] [۵] و [۶] سعی شده است نظر سهام‌داران در فرایند تعریف و پالایش ویژگی‌های کیفی لحاظ گردد. استفاده از نظر سهام‌داران در تعریف و پالایش ویژگی‌های کیفی، مهم‌ترین خصوصیت روش‌های مذکور است که باعث به وجود آمدن تعریف واضح‌تر و ایجاد درک مشترکی از ویژگی‌های کیفی بین اعضای تیم سازنده و دیگر سهام‌داران می‌گردد و موفقیت سیستم را تا حدود زیادی تضمین می‌نماید. اما با وجود

این مقاله در تاریخ ۱۶ آذر ماه ۱۳۹۰ دریافت و در تاریخ ۱۷ مهر ماه ۱۳۹۱ بازنگری‌شد.

سیدمهران شرفی، دانشکده مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد نجف آباد، اصفهان، (email: mehran_sharafi@iaun.ac.ir).

1. Performance
2. Usability
3. Exception Handler
4. Availability
5. Attribute-Driven Design
6. Rational Unified Process
7. Architecture Trade-off Analysis Method
8. Cost-Benefit Analysis Method

این که تاکنون روش‌های زیادی در ارزیابی معماری نرم‌افزار ارائه شده‌اند، در اکثریت قریب به اتفاق روش‌ها، در تعریف ویژگی‌های کیفی به "بایدها"ی پروژه توجه می‌شود اما کمتر روشی وجود دارد که به "نبایدها"یی که لازم است از آنها اجتناب شود، پرداخته باشد. به عبارت دیگر تحلیل مشکلات ناشی از عدم تأمین ویژگی کیفی در سیستم می‌تواند دید[1] متفاوت و مکملی برای شناسایی دقیق‌تر نیازمندی‌های کیفی به معمار و طراحان سیستم بدهد که در کمتر روشی به این رویکرد توجه شده است.

در این مقاله مکانیزمی معرفی می‌شود که در آن با نظر سهام‌داران ابعاد حساس و مشکلات بالقوه سیستم بر حسب اولویت شناسایی شده و به عنوان مبنای ارزیابی و اصلاح معماری در اختیار معمار قرار می‌گیرد. بدین ترتیب کلیه تصمیمات معماری در جهت تأمین ابعاد حساس سیستم اتخاذ می‌گردند و استفاده بهینه از بودجه و زمان به عمل می‌آید. از رویه پیشنهادی می‌توان هم برای ارزیابی معماری و هم به عنوان یک رویه تکمیلی جهت شناسایی نقایص و اصلاح معماری استفاده نمود. ادامه مطالب این مقاله به صورت ذیل سازماندهی شده‌اند: در بخش ۲ به کارهای انجام‌شده در زمینه ارزیابی معماری پرداخته می‌شود و نقاط ضعف و قوت آنها ارزیابی می‌گردد. در بخش ۳ ضمن بحث در مورد ضرورت بازنگری در روش‌های موجود، دیدگاه متفاوتی برای ارتقای روش‌های موجود مطرح می‌شود. در بخش ۴ رویه پیشنهادی معرفی و در بخش ۵ نتایج به کارگیری رویه بر یک نمونه مطالعاتی ارائه می‌گردد. بخش ۶ به نتیجه‌گیری و کارهای آینده اختصاص دارد.

۲- کارهای انجام‌شده

تا کنون روش‌های زیادی در ارزیابی معماری ارائه شده‌اند، بعضی از این روش‌ها از مدل‌های کاملاً رسمی برای توصیف معماری استفاده کرده‌اند و رویه‌ای جهت ارزیابی یا راستی‌آزمایی یک ویژگی کیفی یا وظیفه‌مندی ارائه نموده‌اند ([۷] تا [۹]). در [۱۰] مرور جامعی بر روش‌های رسمی استفاده‌شده در توصیف و ارزیابی معماری نرم‌افزار و مقایسه آنها صورت گرفته است. در مقابل، روش‌های ارزیابی دیگری وجود دارند که از جنبه‌های رسمی کمتری برخوردارند و برخی مبتنی بر سناریو هستند. این روش‌ها از انعطاف‌پذیری بالاتری نسبت به روش‌های رسمی برخوردارند. در این بخش به معرفی بعضی از روش‌های شناخته‌شده در این حوزه و ویژگی‌های آنها پرداخته می‌شود.

۲-۱ محدودیت روش‌های رسمی

روش‌های رسمی اگرچه از مزایای زیادی از جمله دقت بالا برخوردارند ولی به دلیل پیچیدگی‌هایی که در سطوح مختلف طراحی و خصوصاً معماری نرم‌افزار وجود دارد، ایجاد مدل‌های رسمی که بتوانند ابعاد مختلف این پیچیدگی‌ها را پوشش دهند بسیار مشکل و پرهزینه و گاهی غیر ممکن خواهد بود. در میان این روش‌ها به ندرت به ارزیابی توأم چند ویژگی کیفی پرداخته شده است و به مسایل مهم مربوط به تناقض ویژگی‌های کیفی و مصالحه بین آنها کمتر توجه می‌شود. محدودیت‌هایی از قبیل فوران فضای حالت، هزینه بالای یادگیری و به کارگیری باعث شده است که روش‌های رسمی کمتر در حوزه‌های عملی مورد استقبال مهندسین نرم‌افزار واقع شوند.

روش‌های غیر رسمی مبتنی بر سناریو اگرچه شامل فاکتورهای غیر قطعی و تا حدودی غیر دقیق و مبتنی بر تجربیات سازندگان هستند اما انعطاف‌پذیری بالاتری نسبت به روش‌های رسمی داشته و از اقبال بهتری در زمینه‌های کاربردی برخوردارند. در ادامه این بخش به معرفی اجمالی چند روش معروف در ارزیابی معماری پرداخته می‌شود و به نقاط ضعف و قوت آنها اشاره می‌گردد. روش‌های مورد بحث، شامل یک روش نیمه‌رسمی با عنوان FDAF[2] و روش‌های مبتنی بر سناریوی ATAM، FAAM، SARAH، CBAM و ALMA هستند.

۲-۲ FDAF

یکی از روش‌هایی که در آن سعی شده هم از جنبه‌های رسمی و هم غیر رسمی استفاده شود روش FDAF [۱۱] است. FDAF چارچوبی ارائه داده است که به کمک آن می‌توان ویژگی کارایی [۱۲] و امنیت [۱۳] را ارزیابی کرد. در این روش ویژگی کارایی به مجموعه‌ای از sub aspectها تقسیم می‌گردد. درFDAF ابتدا معماری بیان‌شده در UML توسعه‌یافته به زبان‌های توصیف معماری RAPide، Armani، Emilia، Tromellc و Alloy تفسیر[3] می‌گردند تا به کمک ابزارهای موجود وجوه کارایی و امنیت معماری محاسبه گردد. درFDAF به چگونگی تفسیر ویژگی‌های کیفی در یک دامنه خاص و نقش سهام‌داران در این پروسه اشاره مفصلی نشده است ضمناً نقاط تناقض ویژگی‌های کیفی و موضوع مصالحه[4] نیز مورد بحث قرار نگرفته است.

۲-۳ ATAM

ATAM یکی از جامع‌ترین روش‌های ارزیابی معماری است که در آن پیچیدگی‌های مربوط به ویژگی‌های کیفی نرم‌افزار و تأمین آنها در مرحله معماری به خوبی پوشش داده شده است. موضوع پرداختن به چند ویژگی کیفی با در نظر گرفتن تناقضات آنها از مسایل مهمی است که ATAM به آن پرداخته است.

در ATAM به نظرات سهام‌داران در تعریف ویژگی‌های کیفی توجه خاصی می‌شود. در این روش برای استخراج نیازهای کیفی از درخت سودمندی استفاده می‌شود که ریشه آن utility و سطح بعدی آن ویژگی‌های کیفی مانند کارایی، قابلیت نگهداری و قابلیت دسترسی است. سپس با نظر سهام‌داران این ویژگی‌ها پالایش می‌شوند، مثلاً کارایی به Data Latency و Transaction throughput تجزیه می‌شود. این پالایش ادامه می‌یابد تا به سناریوهای کیفی می‌رسد و بعد با اولویت‌بندی این سناریوها نقاط حساس، نقاط مصالحه و شرایط مخاطره[5] شناسایی شده و تصمیمات معماری در خصوص آنها اتخاذ می‌گردد.

۲-۴ FAAM

FAAM[6] [۱۴] روشی جهت ارزیابی معماری خانواده سیستم‌های اطلاعاتی است و بر دو ویژگی کیفی قابلیت تعامل و توسعه‌پذیری تمرکز دارد. FAAM فرایندی شامل راهنماها، معیارهای اندازه‌گیری و رویه‌ها جهت ارزیابی معماری‌های هم‌خانواده سیستم‌های اطلاعاتی ارائه می‌دهد.

در این روش از الگوهای موارد تغییر[7] برای مشخص‌کردن تغییرات

1. View
2. Formal Design Analysis Framework
3. Translate
4. Trade-Off
5. Risk
6. Family - Architecture Assessment Method
7. Change - Case

احتمالی با توجه به تعاملات لازم بین یک خانواده از سیستم‌ها و توسعه‌پذیری آنها استفاده می‌شود. FAAM مشابهت زیادی با ATAM دارد با این تفاوت که ATAM هنوز برای سیستم‌های گروهی استفاده نشده است.

۲-۵ CBAM

این روش معمولاً بعد از پروسه ATAM آغاز می‌شود و بیشتر بر هزینه و سود و موضوعات زمان‌بندی تصمیمات معماری متمرکز است. در این روش هزینه‌ها و بودجه نیز در زمره ویژگی‌های کیفی و حتی مهم‌تر از آنها محسوب می‌گردند که باید در مصالحه‌ها منظور شود و بر سطح عدم قطعیت تخمین‌های انجام‌شده نیز توجه ویژه‌ای دارد. این روش می‌تواند به سازمان‌ها در جهت تحلیل، ارزیابی و تصمیم‌گیری در مورد منابع موجود و تطبیق آنها با استراتژی‌های معماری کمک نماید به گونه‌ای که منافع و کارامدی[1] لازم حاصل گردد.

۲-۶ ALMA

این روش در ابتدا برای سیستم‌های اطلاعاتی تجاری ابداع شد و مورد استفاده قرار گرفت و بعداً برای سیستم‌های تعبیه‌شده نیز استفاده شد [۱۴]. ALMA[2] یک روش تحلیل مبتنی بر سناریو است که برای ارزیابی ویژگی تغییرپذیری معماری نرم‌افزار مفید می‌باشد ولی از دیگر ویژگی‌های مهم کیفی صرف نظر کرده است. ایجاد و ارزیابی سناریوهای تغییر، اساس کار این روش را تشکیل می دهند. در این روش به دو ویژگی اهمیت داده می‌شود، یکی پیش‌بینی هزینه‌های نگهداری و دیگری برآورد ریسک.

۲-۷ SARAH

در بین روش‌های ارزیابی معماری روش SARAH[3] [۱۵] با بهره‌مندی بهره‌مندی از سناریوهای شکست و وزن‌دهی و اولویت‌بندی آنها نقاط حساس سیستم را از نظر قابلیت اطمینان[4] کشف می‌نماید. این روش بر اساس FMEA[5] است [۱۶] که برای محاسبه قابلیت اطمینان سخت‌افزار از آن استفاده می‌شود. در این روش، تعریف سناریوهای شکست توسط سهام‌داران مبنای کار است. پس از این که این سناریوها تعریف شدند در قالب درخت‌های خطا[6] درآورده می‌شود و در خصوص درجه اهمیت آنها و پیداکردن مؤلفه‌های حساس نرم‌افزار که ریسک بیشتری را متوجه سیستم می‌نمایند، قضاوت می‌گردد. روش SARAH از این نظر که یک روال مبتنی بر نظرات سهام‌داران را جهت یافتن مشکلات سیستم محور ارزیابی خود

قرار می‌دهد به روش پیشنهادی در این مقاله شباهت‌هایی دارد اما در SARAH فقط به ویژگی قابلیت اطمینان پرداخته می‌شود و بحثی در مورد ویژگی‌های کیفی دیگر نرم‌افزار نمی‌شود. روش پیشنهادی به جای این که تعاریف کلاسیک ویژگی‌های کیفی را مبنای کار قرار دهد، سعی در ارائه مکانیزمی دارد که نقایص معماری را در هر زمینه از خصوصیات وظیفه‌مندی و غیر وظیفه‌مندی که باشند کشف نماید.

۳- دیدگاهی متفاوت برای ارتقای روش‌های موجود

همان طور که اشاره شد، ATAM یکی از جامع‌ترین روش‌های ارزیابی معماری است و بسیاری از روش‌های ارزیابی مبتنی بر سناریو از مکانیزم‌های معرفی‌شده در این روش استفاده می‌نمایند. در مراحل مختلف این روش به نظر سهام‌داران توجه زیادی می‌شود. به عنوان مثال در مراحل انتهایی فرایند ATAM که به سناریوهای طوفان مغزی[7] اختصاص دارد از سهام‌داران خواسته می‌شود که نیازهای خود را در قالب سناریوها بیان نمایند و سناریوهای ایجادشده توسط سهام‌داران با سناریوهایی که در فرایند درخت سودمندی ایجاد شده است مقایسه می‌گردد. این مکانیزم می‌تواند در درک دقیق‌تر نیازهای کیفی و شناسایی نقایص احتمالی بسیار مؤثر باشد. اما با وجود همه توانمندی‌های ATAM به نظر می‌رسد می‌توان روش‌های مکملی در کنار ATAM نیز به کار گرفت که توانایی آن را ارتقا دهد. روال بالا به پایینی که در پالایش درخت سودمندی اتخاذ می‌شود دریافت دقیق معنای نیاز کیفی از دید سهام‌دار را محدود می‌نماید.

به عنوان مثال وقتی گره Performance بعد از Utility قرار داده می‌شود به این معنا است که ما در نرم‌افزار به ویژگی Performance نیاز داریم اما واقعیت این است که ما شاید دقیقاً به Performance نیاز نداریم بلکه به یک چیزی نیاز داریم که معنای دقیق آن را کاربر و سهام‌داران می‌دانند. پس بهتر است مستقیماً سراغ نظر سهام‌دار برویم و بعد نتیجه‌گیری کنیم که خواست او Performance است یا چیز دیگر، چون مبنا قرار دادن Performance و بعد پالایش آن می‌تواند ما را از نظر واقعی کاربر دور سازد و یا حداقل می‌توان گفت که راه مطمئنی برای کشف دقیق منظور کاربر از ویژگی کیفی نمی‌باشد. به عنوان مثال در یک سیستم فرض کنید منظور کاربر بازیافت سریع سیستم در کمتر از یک دقیقه پس از توقف آن است. سؤال این است که بر اساس درخت سودمندی آیا از شاخه Performance به این موضوع خواهیم رسید؟ یا از شاخه Reliability یا از Availability یا حتی از Usability و یا اصلاً تضمینی وجود دارد که هر یک از شاخه‌های تعریف‌شده در ساختار درخت سودمندی ما را به این مطلب برساند؟

لذا برای طراحی و یا اصلاح معماری به مکانیزمی نیاز است که با اختصاص‌دادن نقش اساسی‌تری به کاربر و سهام‌داران دیگر، خواسته‌های کیفی مورد نظر آنها به طور دقیق‌تری دریافت شوند و سپس در قالب ویژگی‌های کیفی مورد نیاز تفسیر گردند و نهایتاً با به کارگیری تاکتیک‌های مرسوم در ارضای آن خواسته‌ها اقدام گردد.

از سوی دیگر در اکثر روش‌های موجود برای شناسایی نیازهای کیفی به "بایدها"ی پروژه توجه می‌شود اما کمتر روشی وجود دارد که به "نبایدها" پرداخته باشد. حال آن که نقص در ارضای یک ویژگی کیفی است که می‌تواند باعث بروز مشکلات و نارضایتی سهام‌داران گردد. در روش پیشنهادی در این تحقیق با تحلیل مشکلات بالقوه ناشی از عدم تأمین ویژگی کیفی در سیستم و کشف نقاط ضعف سیستم، دید متفاوت و مکملی برای شناسایی دقیق‌تر و برآورده کردن نیازمندی‌های کیفی به معمار و طراحان سیستم داده می‌شود که در کمتر روشی به این رویکرد توجه شده است.

رویه پیشنهادی در این مقاله یک روال متفاوت را در جهت درک ویژگی‌های کیفی از دید سهام‌داران ارائه می‌دهد که علاوه بر تعیین حدود

1. Efficiency
2. Architecture-Level Modifiability Analysis
3. Software Architecture Reliability Approach
4. Reliability
5. Failure Modes and Effects Analysis Method
6. Fault Tree
7. Brainstorming Scenarios

و مشخصه‌های ویژگی‌های کیفی، تشخیص نقاط حساس سیستم را نیز میسر می‌سازد. شناسایی نقاط حساس سیستم می‌تواند به معمار در اولویت‌بندی فعالیت‌های معماری و تمرکز منابع جهت رفع نقایص مهم‌تر یاری رساند. این روش را می‌توان به طور مستقل و یا به عنوان بخشی از فرایند ATAM جهت ارتقای فاز سناریوهای طوفان مغزی به کار برد با این امتیاز که از این روش می‌توان در شناسایی نقایص مربوط به ویژگی‌های وظیفه‌مندی سیستم نیز در سطح معماری استفاده نمود.

٤- رویه پیشنهادی

در این قسمت به معرفی رویه پیشنهادی پرداخته می‌شود. برای طراحی یک مدل جهت ارزیابی کیفیت سیستم قبل از هر چیز بر حسب نیاز، معنای واژه‌هایی چون Error، Defect، Fault، Failure و Problem در محدوده بحث مشخص می‌شوند. از آنجایی که بسیاری از محققین مجبور به تعریف مجدد این واژه‌ها بوده‌اند، تا کنون تعاریف مختلف و بعضاً متناقضی از آنها در تحقیقات وجود دارد [۱۷]. لذا برای جلوگیری از تکثر مفاهیم در این مقاله سعی می‌شود حتی‌الامکان از تعریف‌های استاندارد IEEE در [۱۸] استفاده شود. در رویه پیشنهادی در این مقاله، دو مفهوم اساسی مورد استفاده قرار می‌گیرد، یکی مشکل[1] و دیگری نقص[2]. در [۱۸] تعریف‌هایی که از این دو مفهوم شده است به قرار زیر است:

مشکل: (الف) دشواری و یا عدم قطعیت تجربه‌شده توسط یک یا چند شخص که نتیجه وقوع یک ناهنجاری در سیستم در حال استفاده است و (ب) یک وضعیت نامطلوب که باید برطرف شود.

نقص: اشکال در یک محصول که مانع از پاسخ‌گویی مطلوب به نیازمندی‌ها شود.

تعریفی که از نقص شده است منطبق با مفهوم مورد نظر در این مقاله است اما تعریف بالا از واژه problem محدود به اشکالاتی شده است که در کاربری سیستم مشاهده می‌شود (به دلیل عبارت "سیستم در حال استفاده"). از آنجایی که در رویه پیشنهادی ما واژه مشکل به مفهوم وسیع‌تری اطلاق می‌شود که به عدم رضایت هر یک از سهام‌داران سیستم می‌تواند مربوط شود لذا در اینجا مفهوم problem را با اندکی تغییر نسبت به آنچه در [۱۸] آمده است استفاده می‌نماییم و آن را با مثال‌های ذیل تشریح می‌کنیم.

مثال‌هایی از انواع مشکل: "به وجود آمدن ازدحام در یک منطقه از شهر در اثر عملکرد نامطلوب سیستم مدیریت ترافیک"، "تغییر اطلاعات کتاب با بیش از ۵ ثانیه تأخیر (بیش از زمان قابل قبول)"، "اصابت موشک به یک هدف اشتباه در یک سیستم دفاعی"، "عدم امکان فروش مجزای مؤلفه‌های سیستم"، "اشکال در جایگزینی ماژول پایگاه داده مرکزی با محصول مشابهی از یک شرکت دیگر"، "هزینه بالای لازم برای کشف و رفع یک خطا در زیرسیستم استنتاج (بیش از دو نفر ساعت)"، "صرف مدت ۲ ماه (بیش از حد قابل قبول) جهت آموزش یک کارمند با سابقه فروش تا بتواند کارکردن با سیستم ثبت سفارشات را یاد بگیرد".

برای هر مشکل، دو صفت خاصه تعریف می‌نماییم: هزینه و احتمال وقوع. هزینه مشکل عبارت است از خسارتی که به دنبال خواهد داشت. هزینه می‌تواند وابسته به کاربرد نرم‌افزار و محیط‌های عملیاتی مختلف باشد و باید با استفاده از نظرات سهام‌داران مشخص گردد. احتمال وقوع مشکل، عددی بین صفر و یک است که احتمال بروز آن مشکل را بیان می‌دارد. با به کارگیری رویه پیشنهادی، احتمالات وقوع مشکلات به تدریج کاهش می‌یابند.

همان طور که ذکر گردید هر مشکل، نتیجه وجود حداقل یک نقص در سیستم است و بسیاری از مشکلات خصوصاً مشکلات مربوط به ویژگی‌های کیفی ناشی از نقایص معماری نرم‌افزار هستند [۱]. در این رویه برای هر نقص یک درجه اهمیت اختصاص داده می‌شود. درجه اهمیت نقص وابسته به تأثیر آن نقص در به وجود آوردن مشکلات پرهزینه‌تر است.

٤-١ رابطه بین مشکلات و نقص‌ها

هر مشکل می‌تواند توسط یک یا چند نقص به وجود آید و متقابلاً هر نقص نیز می‌تواند موجب بروز یک یا چند مشکل گردد. ارتباطی که بین یک مشکل و یک نقص می‌توان تعریف نمود، میزان تأثیر نقص در به وجود آمدن مشکل است که می‌تواند به صورت یک رابطه فازی بیان شود که در ادامه معرفی خواهد شد. به عنوان مثال در زیرسیستمی از یک سیستم حساس که قابلیت دسترسی در آن از اهمیت بالایی برخوردار است، عدم وجود مؤلفه پشتیبان با تأثیر (درجه عضویت) x و عدم عملکرد صحیح مؤلفه voter با درجه با تأثیر (درجه عضویت) y می‌تواند موجب توقف آن سیستم شود که یک مشکل به حساب می‌آید. به عنوان مثال دیگر، در یک سیستم حساس به زمان[3] عدم وجود مؤلفه کمکی هم‌زمان[4] هم‌زمان[4] با تأثیر x و عدم کنترل تواتر نمونه‌برداری[5] با تأثیر y می‌تواند می‌تواند باعث کندشدن سیستم (که یک مشکل است) گردد.

٤-٢ نقص‌های پیشنهادی

اولین مرحله در رویه پیشنهادی، شناسایی مشکلات بالقوه برای سیستم می‌باشد. در ادامه کار با مشارکت سهام‌داران و به ویژه مشتری، به هر یک از مشکلات هزینه‌ای نسبت داده می‌شود. این کمیت، بیانگر هزینه‌ای است که وقوع مشکل می‌تواند به طور مستقیم و یا غیر مستقیم به پروژه تحمیل نماید. ویژگی دیگری که برای یک مشکل منظور می‌نماییم احتمال وقوع مشکل است، احتمالات وقوع همه مشکلات در ابتدا ۱ در نظر گرفته می‌شود که در خلال اجرای رویه، به تدریج از آن احتمالات کاسته خواهد شد.

در مرحله بعدی، سهام‌داران و خصوصاً اعضای تیم توسعه نرم‌افزار باید به ازای هر مشکل، نقص یا نقایصی را پیشنهاد کنند که ممکن است باعث به وجود آمدن آن مشکل شوند. این مرحله از اهمیت ویژه‌ای برخوردار است و تهیه یک فهرست جامع از مشکلات، اساس ارزیابی و شناسایی اشکالات معماری نرم‌افزار را تشکیل می‌دهد. مشکلات ممکن است ناشی از اختلال در پاسخ‌گویی به نیازهای مختلف وظیفه‌مندی و یا غیر وظیفه‌مندی باشند و مشارکت کلیه سهام‌داران در تهیه این فهرست بسیار حائز اهمیت است زیرا یک مشکل می‌تواند توسط انواع نقایص و خطاها ایجاد شود، از جمله نقص‌های مربوط به تحلیل نیازها، معماری، طراحی و خطاهای برنامه‌نویسی. بنابراین در این مرحله هر یک از اعضای تیم توسعه نرم‌افزار نقایص را با توجه به دیدگاه خود و وظیفه‌ای که در پروژه به عهده دارد پیشنهاد می‌نماید. به عنوان مثال مشکلی را در نظر بگیرید با این عنوان: "نفوذ غیر مجازی به سیستم مالی صورت گرفته که غیر قابل رد گیری است". هر یک از افراد تیم سازنده با توجه به تخصص

1. Problem
2. Defect
3. Time Sensitive
4. Concurrent
5. Sampling

خود، این مشکل را تفسیر نموده و نقایص مولد احتمالی را پیشنهاد می‌نماید. به عنوان مثال نقصی را که طراح سیستم ممکن است پیشنهاد نماید این است که "رویه رد ممیزی برای سه روز متوقف بوده است" و این توقف اصلاح نشده و پیغام لازم نیز به مدیر سیستم داده نشده است. شایان ذکر است که مورد اخیر به نوعی به دسترسی پذیری و به نوعی به قابلیت کاربری مربوط می‌شود که تأثیرش را در امنیت[1] نشان داده است. نقصی را که برنامه‌نویس ممکن است پیشنهاد نماید این است که یک خطا در روتین رد ممیزی باعث شده است که بعضی از تراکنش‌های خاص ثبت نشوند و یا نقص در روتین مقایسه الگوهای ترافیک، مانع از شناسایی به موقع رفتار مهاجم شده است. نقصی را که طراح سیستم ممکن است پیشنهاد نماید این است که سیستم در موقع تعریف یک رمز عبور ضعیف توسط کاربر، هشدار لازم جهت اصلاح آن را به کاربر نمی‌دهد. لذا سیستم در سنجش هویت کاربر دچار اشکال شده و مقصر اصلی قابل شناسایی نیست (این می‌تواند مربوط به قابلیت کاربری باشد که در امنیت تأثیر گذاشته است).

اساس روش پیشنهادی بر این موضوع استوار است که تعاریف کلاسیک بیان‌شده در ادبیات موضوع، نمی‌توانند معنای دقیق ویژگی کیفی مورد نظر سهام‌داران را منعکس نمایند. لذا هدف اصلی این روش آن است که به جای استفاده از این تعاریف و یا ابزارهایی مانند درخت سودمندی، از مکانیزم خاص خود استفاده نماید که در آن به نگرانی‌ها و دغدغه‌های سهام‌داران در خصوص سیستم اولویت داده می‌شود، سوای این که نام آنها یا تعاریف کلاسیک آنها چه باشد. لذا مکانیزم پیشنهادی نگاهی تفکیک‌شده به ویژگی‌های کیفی ندارد و هدف آن کشف نقص‌های موجود در زمینه‌های مختلف وظیفه‌مندی و حتی غیر وظیفه‌مندی است.

فاز پیشنهاد نقص از رویه، اطلاعات متنوع و مهمی را در اختیار تیم ارزیابی معماری قرار می‌دهد که می‌توانند از آن برای کشف و رفع نقایص معماری استفاده نمایند (در شرح مراحل رویه به طور مفصل به چگونگی استفاده از این اطلاعات اشاره خواهد شد). از طرف دیگر بعضی از اطلاعات به دست آمده به نقایصی اشاره دارند که کمتر جنبه معماری دارند و بیشتر به نقایص طراحی تفصیلی و یا خطاهای برنامه‌نویسی یا دیگر انواع خطاها اشاره دارند. این نوع اطلاعات بهتر است مستندسازی شده و در اختیار تیم‌های ارزیابی و آزمون مربوطه قرار گیرند تا از آنها جهت تولید نمونه‌های آزمون و تدوین معیارهای ارزیابی استفاده نمایند. هرچه میزان مشارکت طراحان سطح بالا و معماران در فرایند پیشنهاد نقص بیشتر باشد می‌توان انتظار داشت که خطاهای کشف‌شده بیشتر جنبه معماری داشته باشند.

در رویه پیشنهادی برای هر نقص یک درجه اهمیت محاسبه می‌شود. نقصی دارای اهمیت بیشتر است که تأثیر بیشتری در تولید مشکلات پرهزینه‌تر داشته باشد. فرمول محاسبه درجه اهمیت نقص در بخش بعدی ذکر می‌گردد.

پس از محاسبه درجه اهمیت هر نقص، لیست نقص‌ها بر اساس درجه اهمیت به صورت نزولی مرتب می‌شود. سپس از ابتدای لیست شروع کرده و وجود یا عدم وجود هر نقص در معماری مورد ارزیابی قرار می‌گیرد. اگر نقص مورد نظر در معماری وجود نداشته باشد از احتمال وقوع مشکل(های) مربوطه کاسته می‌شود و اگر نقص وجود داشته باشد نیز پس از رفع نقص، از احتمال وقوع مشکل(های) مربوطه به تناسب کاسته می‌شود. این کار برای تمام نقص‌ها انجام می‌شود. پس از آن مجدداً لیست مشکلات بر اساس حاصل‌ضرب هزینه در احتمال وقوع مرتب می‌گردد و مراحل مذکور تکرار می‌شوند.

در رویه پیشنهادی معیاری جهت خاتمه‌دادن به فرایند ارزیابی معماری ارائه شده است که در ادامه معرفی خواهد شد.

۴-۳ بیان رسمی‌تری از رویه پیشنهادی

با توجه به مطالب مذکور در ذیل رویه پیشنهادی با یک بیان سیستماتیک و رسمی تشریح می‌گردد. انجام رویه پیشنهادی به ۸ مرحله تقسیم شده است که عبارتند از:

۱) با مشارکت کلیه سهام‌داران و در جلسات طوفان مغزی، فهرستی از مشکلات نرم‌افزار تهیه شود. هر مشکل به صورت $p=(c,r)$ تعریف می‌شود که c و r به ترتیب هزینه و احتمال وقوع مشکل p هستند. مجموعه مشکل‌ها به صورت $P=\{p_i \mid i=1,2,\ldots,n\}$ در نظر گرفته می‌شود.

۲) با مشارکت سهام‌دارن پروژه و خصوصاً مشتری، هزینه‌ای را که هر مشکل می‌تواند به دنبال داشته باشد مشخص و مقدار اولیه ۱ به احتمال وقوع هر مشکل اختصاص داده شود. این احتمال برای هر مشکل در خلال اجرای رویه به تدریج کاهش می‌یابد.

۳) لیست مشکلات بر اساس هزینه به صورت نزولی مرتب شود.

۴) با مشارکت سهام‌داران پروژه خصوصاً معمار و طراحان نرم‌افزار، به ازای هر یک از مشکلات، فهرستی از نقص‌هایی که می‌توانند مولد مشکل مربوطه باشند تهیه شود. در این قسمت کلیه افراد تیم توسعه نرم‌افزار مشارکت دارند و هر یک از دیدگاه تخصصی خود نقص‌هایی را که ممکن است باعث بروز مشکل شوند پیشنهاد می‌کنند. در صورت لزوم می‌توان یک نقص کلی را به نقص‌های جزئی‌تری تجزیه کرد و سلسله مراتبی از نقص‌ها بر اساس رابطه علت و معلولی به وجود آورد. مجموعه نقص‌های پیشنهادی را با $D=\{d_j \mid j=1,2,\ldots,m\}$ نمایش می‌دهیم. همان طور که ذکر گردید هر مشکل می‌تواند توسط یک یا چند نقص به وجود آید و متقابلاً هر نقص نیز می‌تواند موجب بروز یک یا چند مشکل گردد. ارتباطی که بین یک مشکل و یک نقص می‌توان تعریف نمود، میزان تأثیر نقص در به وجود آمدن مشکل است که می‌تواند به صورت یک رابطه فازی بیان شود.

بنا به تعریف [۱۹] اگر X و Y دو مجموعه غیر تهی از اشیا باشند، یک رابطه فازی R نگاشتی از فضای حاصل‌ضرب دکارتی $X\times Y$ به بازه $[0,1]$ است به طوری که استحکام نگاشت با تابع عضویت μ_R بیان می‌شود. به عبارت دیگر

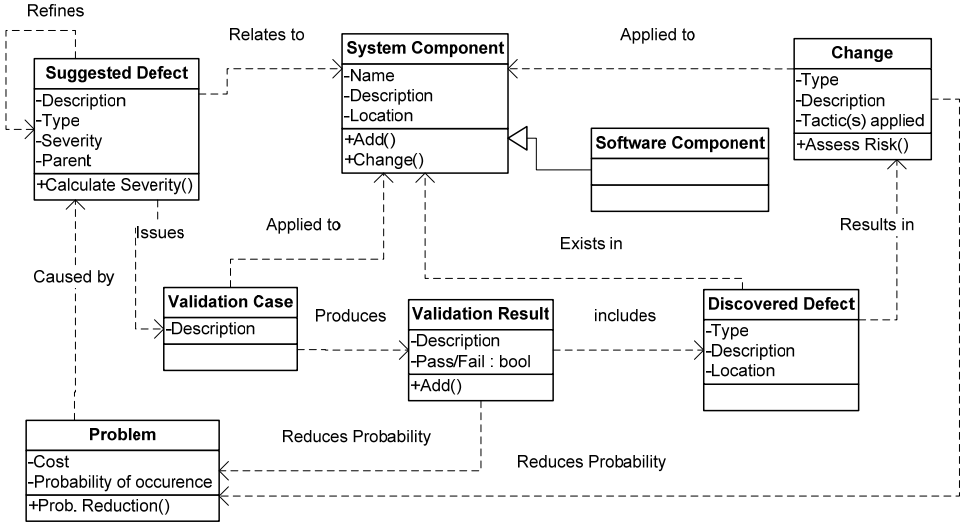
$$R=\{((x,y),\mu_R(x,y)) \mid 0\le \mu_R(x,y)\le 1, x\in X, y\in Y\} \quad (1)$$

که $\mu_R(x,y)$ درجه عضویت زوج (x,y) در رابطه R است.

در رویه پیشنهادی، میزان تأثیر نقص در به وجود آمدن مشکل را به صورت رابطه فازی R از مجموعه P به مجموعه D بیان می‌کنیم به طوری که $\mu_R(p_i,d_j)$ درجه عضویت زوج (p_i,d_j) در R می‌باشد که میزان تأثیر نقص d_j در به وجود آمدن مشکل p_i را نشان می‌دهد. میزان‌های تأثیر (درجات عضویت) باید توسط پیشنهادکننده نقص و با اتفاق نظر دیگر اعضای تیم شرکت‌کننده در انجام رویه و بر اساس تجربیات کسب‌شده از پروژه‌های قبلی تعیین گردند.

۵) در این رویه برای هر نقص یک درجه اهمیت محاسبه می‌شود. به

1. Security



شکل ۱: نمودار کلاس مربوط به اجزای رویه پیشنهادی.

جدول ۱: مشکلات فهرست‌شده برای سیستم مدیریت پارکینگ و هزینه‌ها.

	مشکل	هزینه
۱	سرقت وسایل نقلیه	۶۵
۲	ایجاد ترافیک در محدوده خارج و یا داخل پارکینگ	۹۰
۳	واردآمدن صدمه به افراد یا وسایل نقلیه در حین ورود یا خروج	۹۵
۴	اشتباه در اعلام هزینه	۴۰
۵	سوء استفاده مالی از سیستم	۷۰

طور کلی نقصی از اهمیت بیشتر برخوردار است که تأثیر بیشتری در تولید مشکلات پرهزینه‌تر داشته باشد. بنابراین درجه اهمیت نقص d_j که با $S(d_j)$ نمایش داده می‌شود، طبق (۲) محاسبه می‌شود

$$S(d_j)=\sum_{i=1}^{n} p_i.c\times\mu_R(p_i,d_j) \qquad (۲)$$

۶) نقص‌ها بر اساس درجه اهمیت به صورت نزولی مرتب شوند.

۷) از ابتدای لیست نقص‌ها شروع نموده، بر پایه نوع نقص پیشنهادشده و مؤلفه‌هایی که می‌توانند نقص مربوطه را در بر داشته باشند، آزمون‌هایی جهت کشف آن طراحی و اجرا گردند. در انجام آزمون‌ها اگر نقص مورد نظر در معماری وجود نداشته باشد از احتمال وقوع اثر(های) نامطلوب مربوطه کاسته شود و اگر نقص وجود داشته باشد نیز پس از رفع نقص، از احتمال وقوع اثر(های) نامطلوب مربوطه به تناسب کاسته شود. این کار برای تمام نقص‌ها انجام شود.

۸) ζ که معیار پیشنهادی برای خاتمه رویه است از (۳) محاسبه شود

$$\zeta=\frac{\sum_{i=1}^{n} p_i.c\times p_i.r}{n} \qquad (۳)$$

اگر ζ به اندازه کافی کوچک شده باشد (تشخیص این موضوع بستگی به زمان و بودجه اختصاص داده شده به فرایند ارزیابی معماری و نظر تیم ارزیابی معماری دارد) می‌توان رویه را خاتمه داد و در غیر این صورت برای هر p_i عبارت $p_i.c\times p_i.r$ محاسبه شده و مشکلات بر این اساس به صورت نزولی مرتب می‌شوند. مشکلات انتهایی فهرست به دست آمده که خطر کمتری دارند را می‌توان با توافق اعضای تیم ارزیابی حذف نمود. سپس به ازای هر مشکل مجدداً فهرستی از نقایص تعریف‌شده و اجرای رویه از مرحله ۴ تکرار می‌گردد.

در شکل ۱ مدل کلاس مربوط به اجزای رویه پیشنهادی آورده شده است. این مدل علاوه بر این که عناصر رویه پیشنهادی و ارتباطات بین آنها را تشریح می‌کند، می‌تواند به عنوان زیربنای طراحی یک ابزار برای مدیریت و اجرای خودکار رویه پیشنهادی استفاده شود که بخشی از این کار انجام شده و در کارهای آینده به ادامه آن پرداخته می‌شود.

۵- به کارگیری رویه پشنهادی بر یک مطالعه موردی و ارزیابی نتایج

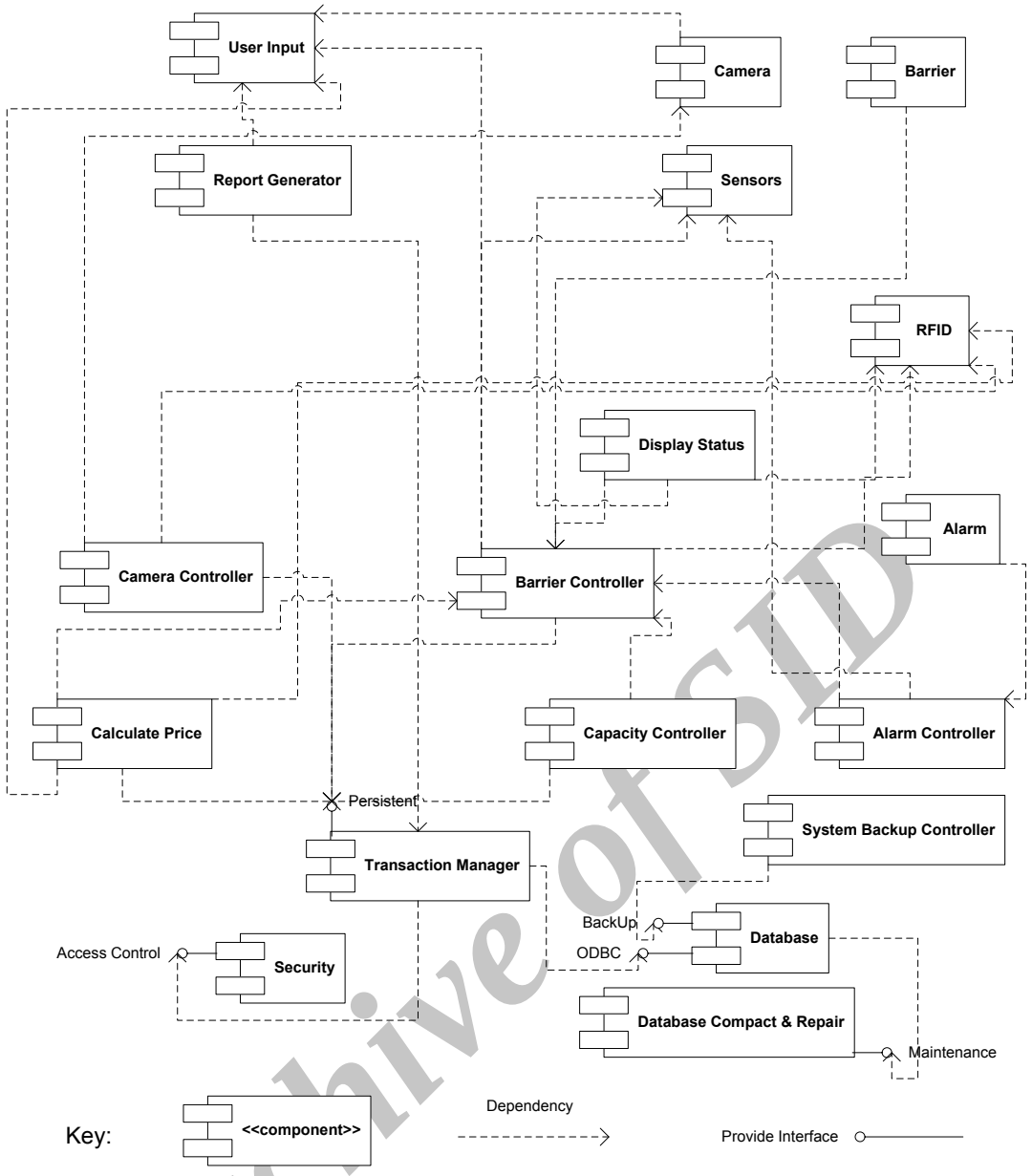
در این بخش نتایج اجرای رویه پیشنهادی بر روی یک سیستم واقعی ارائه می‌گردد تا کارامدی رویه در کشف و رفع نقایص معماری به صورت واضح‌تری بیان گردد. این سیستم یک سیستم اطلاعاتی کنترل متمرکز پارکینگ‌های شهری (سمپا) است که جهت مدیریت بهتر و سریع‌تر، افزایش امنیت پارکینگ‌ها و سامان‌دهی به ترافیک شهری در نواحی مرکزی طراحی شده است. این سیستم ورود و خروج وسایل نقلیه، جایابی و هدایت رانندگان به محل‌های پارک، محاسبات و تراکنش‌های مالی، اعلان وضعیت لحظه به لحظه به مراکز حفاظتی و اعلان ظرفیت پارکینگ‌های تحت کنترل را در پانل‌های الکترونیکی سطح شهر برای اطلاع و راهنمایی مردم به طور خودکار انجام می‌دهد. شکل ۲ نمودار UML مؤلفه نرم‌افزار مربوطه را نشان می‌دهد.

طی جلسات طوفان مغزی با مشارکت کلیه سهام‌داران سیستم، مشکلات تعیین گردیدند و هزینه‌های هر مشکل با بررسی شرایط مختلف محیطی (اجتماعی، قانونی و ...) از ۱ تا ۱۰۰ وزن‌دهی شد که در جدول ۱ آورده شده است.

در مرحله پیشنهاد نقص‌های مولد، برای هر مشکل به طور متوسط بیش از ۳۰ نقص مولد توسط تیم سازنده ارائه شد که به دلیل مفصل‌بودن نتایج به ذکر بعضی از نقص‌های مولد مربوط به مشکل شماره دو یعنی "ایجاد ترافیک در محدوده خارج و یا داخل پارکینگ" می‌پردازیم.

همان طور که در بخش ۴ اشاره گردید، با توجه به این که نقص‌ها را می‌توان در سطوح مختلفی از جزئیات بیان کرد لذا امکان بیان سلسله مراتبی یک نقص وجود دارد به این معنی که در صورت نیاز به بیان واضح‌تر یک نقص می‌توان آن نقص را به زیرنقص‌هایی تجزیه کرد و این تجزیه را به صورت سلسله مراتب علت و تأثیر[1] تا هر سطحی که لازم باشد ادامه داد. در جداول ۲ و ۳ بعضی از نقص‌های پیشنهادی، تست‌های

1. Cause - Effect



شکل ۲: نمودار مؤلفه نرم‌افزار مدیریت پارکینگ.

لازم برای تشخیص نقص‌ها، تغییرات انجام‌شده و مؤلفه‌های مربوط و همچنین روند کاهشی احتمال وقوع مشکل مذکور آورده شده‌اند (نقص‌ها و آزمون‌های انجام‌شده برای ارجاعات بعدی کدگذاری شده‌اند). نقص‌های پیشنهادی که در جداول آورده نشده‌اند به همراه کدهای مربوطه عبارتند از: "خطا در نمایش اطلاعات پانل‌های سطح شهر" (۲-۳)، "اعلام اشتباه محل پارک وسیله نقلیه واردشده" (۲-۴) و "نقص اطلاعات دریافت‌شده از سنسورهای ورودی و خروجی" (۲-۷). همچنین برای بعضی از نقص‌های پیشنهادی مانند "توقف سیستم" به دلیل مفصل‌بودن تست‌های انجام‌شده و کمبود فضا، فقط تست‌های منتخب در جدول آورده شده‌اند. در این جدول بر اساس این که نقص از نوع نقص سخت‌افزاری، برنامه‌نویسی، طراحی و یا معماری نرم‌افزار است به ترتیب با حروف H، P، D و A مشخص شده است.

برای ارزیابی روش پیشنهادی با به کارگیری روش ATAM بر روی معماری سیستم سمپا نتایج حاصل از آن با نتایج روش پیشنهادی مقایسه گردید که در جدول ۴ خلاصه این مقایسه آورده شده است که نقص ۱ فقدان مؤلفه‌ای جهت تشخیص زودهنگام توقف پایگاه داده و اعلان هشدار، نقص ۲ عدم وجود مؤلفه یا راهکار معماری جهت مدیریت صحیح ذخیره و بازیابی اطلاعات در وضعیت‌های بارکاری متفاوت و نقص ۳ عدم تدارک مؤلفه مجزا جهت مدیریت امور بلادرنگ و واکنشی سیستم است. این مقایسه فقط در مورد نقص‌های معماری کشف‌شده یا کشف‌نشده توسط دو روش انجام شد. در این مقایسه، روش پیشنهادی، تقریباً کلیه نقص‌هایی که توسط روش ATAM کشف شد را مشخص نمود. در مقابل، نقص "فقدان مؤلفه جهت مدیریت ذخیره و بازیابی اطلاعات در بارهای کاری مختلف" توسط روش ATAM آشکار نشد. با کشف این نقص توسط روش پیشنهادی، معمار با بررسی‌های انجام‌شده به ضرورت اضافه‌کردن ماژول Compact & Repair پی برد (جدول ۲). دلیل این موضوع این بود که هیچ یک از گره‌های درخت سودمندی ایجادشده حتی گره Performance به سناریویی که باعث شناسایی این نقص شود پالایش نگردید و در مرحله سناریوهای طوفان مغزی نیز هیچ یک از سهام‌داران سناریویی ارائه ندادند که دقیقاً این نقص را آشکار نماید.

٦- نتیجه‌گیری و کارهای آینده

در بیشتر روش‌های ارزیابی معماری سعی شده است نظر سهام‌داران در فرایند تعریف و پالایش ویژگی‌های کیفی لحاظ گردد. استفاده از نظر

جدول ۲: نقص‌های پیشنهادی، تست‌های لازم برای تشخیص نقص‌ها، تغییرات انجام‌شده و بخشی از روند کاهشی احتمال وقوع، مربوط به مشکل شماره ۲.

نقص پیشنهادشده	نقص‌های پالایش‌شده سطح اول (refined defects-level1)	میزان تأثیر $\mu_R(p_{2,}d_j)$	آزمون‌های انجام‌شده	نتیجه ارزیابی	تغییر انجام‌شده/ تاکتیک به کار گرفته شده	مؤلفه یا مؤلفه‌ها	نوع اصلاح	احتمال وقوع مشکل $(P_{2,}r)$
کندی عملکرد سیستم، ۲-۱	کندی انتقال اطلاعات، ۲-۱-۱	۰/۸	ارزیابی قابلیت اطمینان شبکه انتقال اطلاعات، ۲-۱-۱-۱	شبکه انتقال اطلاعات از پهنای باند مناسب و اطمینان قابل قبولی برخوردار است. در صورت لزوم می‌توان یک ماژول کنترل‌کننده و هشداردهنده هنگام قطعی و یا کندی شبکه تعبیه نمود.	اضافه‌کردن مکانیزم heartbeat یا ping/echo جهت اطمینان از عملکرد شبکه انتقال اطلاعات.	NETWORK	A, H	۰/۵
			اطمینان از پهنای باند شبکه، ۲-۱-۱-۲					
	کندی ذخیره و بازیابی اطلاعات، ۲-۱-۲	۰/۳	وجود مؤلفه‌ای جهت مدیریت صحیح ذخیره و بازیابی اطلاعات در وضعیت‌های مختلف بار کاری، ۲-۱-۲-۱	مؤلفه مشخصی برای این منظور وجود ندارد و رویه خاصی هم تدارک دیده نشده است. وقتی رکوردی حذف می‌شود فضای مربوط به فیلد تصویر به طور کامل آزاد نمی‌گردد و منجر به افزایش بی‌رویه حجم بانک اطلاعاتی و کندی عملکرد آن به مرور زمان می‌گردد.	ماژولی جهت مدیریت حافظه جداول اطلاعاتی اضافه گردید: ماژول Compact & repair.	DB, TM	A	۰/۳
	عدم مدیریت صحیح منابع سیستم، ۲-۱-۳	۰/۲	ارزیابی الگوی زمان‌بندی و نرخ خواندن اطلاعات سنسورها و دریافت و ثبت تصاویر دوربین‌ها، ۲-۱-۳-۱	حداقل تواتر لازم برای خواندن اطلاعات سنسورها و دوربین‌ها محاسبه شدند.	به جهت استفاده بهینه از پهنای باند و افزایش کارایی تواتر خواندن اطلاعات دوربین‌ها کاهش داده شد.	SNS, CAM	A, H, P	۰/۲۲
نمایش اشتباه ظرفیت باقیمانده، ۲-۲	اشکال در عملکرد روتین محاسبه ظرفیت در ماژول CAP، ۲-۲-۱	۰/۲	محتوای اولیه جدول Places فیلد Capacity صحیح است یا خیر، ۲-۲-۱-۱	پس از ارزیابی انجام‌شده خطایی کشف نشد.	–	CAP	P, D	۰/۱۸
			صحت عملکرد روتین Calculate Capacity، ۲-۲-۱-۲	پس از ارزیابی انجام‌شده خطایی کشف نشد.	–	CAP		۰/۰۹
			صحت Query، remainder (۱ Query)، ۲-۲-۱-۳	پس از ارزیابی انجام‌شده خطایی کشف نشد.	–	CAP		۰/۰۷
عملکرد اشتباه یا خارج از کنترل راه‌بند، ۲-۵	خطای انتقال اطلاعات (RS۴۸۵)، ۲-۵-۱	۰/۹	بررسی ارتباط نرم‌افزار با سخت‌افزار راه‌بند که فرامین درست منتقل شوند، ۲-۵-۱-۱	در انتقال اطلاعات خطا پیدا نشد اما مسیر انتقال اطلاعات از یک محیط نویزی عبور می‌کند که امکان ایجاد اختلال در تبادل اطلاعات را دارد.	تغییر مسیر خطوط انتقال اطلاعات از کامپیوتر مرکزی به راه‌بندها.	BAR, BC	A, D, H	۰/۰۳
	خطای نرم‌افزار در انتشار فرمان به راه‌بندها، ۲-۵-۲	۰/۹	تست بازکردن Barrier-Port در ابتدای برنامه، ۲-۵-۲-۱	پس از ارزیابی انجام‌شده خطایی کشف نشد.	جهت اطمینان دستور العمل بازگشایی port در ابتدای برنامه قرار داده شد.	BAR, BC	P	۰/۰۱
			صحت روتین‌های Open-Barrier Close-Barrier در ماژول Barrier (P و D) در ماژول Barrier، ۲-۵-۲-۲	پس از ارزیابی انجام‌شده خطایی کشف نشد.	–	BAR, BC	D, P	۰/۰۰۰۸
توقف سیستم، ۲-۶	عدم برپایی مجدد به موقع پایگاه داده پس از توقف، ۲-۶-۱	۰/۹	وجود ماژول Recovery & Backup جهت برگرداندن سریع سیستم به حالت عملیاتی، ۲-۶-۱-۱	چنین ماژولی در طراحی اولیه دیده نشده است.	اضافه‌کردن ماژول Recovery & Backup که با مکانیزم Ping/echo توقف را تشخیص داده و سریعاً سیستم هشداردهنده را فعال می‌نماید.	SBC, DBR	A	۰/۰۰۰۵

جدول ۳: نقص پیشنهادی، تست لازم برای تشخیص نقص و تغییرات انجام‌شده مربوط به مشکل شماره ۳.

نقص پیشنهادشده	نقص‌های پالایش‌شده سطح اول (refined defects-level1)	میزان تأثیر $\mu_R(p_{۳},d_j)$	آزمون‌های انجام‌شده	نتیجه ارزیابی	تغییر انجام‌شده/ تاکتیک به کار گرفته شده	نوع اصلاح	احتمال وقوع مشکل $(P_{۳},r)$
عملکرد با تأخیر راه‌بند پس از تشخیص مانع، ۳-۱	تأخیر نرم‌افزار در انتشار فرمان توقف راه‌بند، ۳-۱-۳	۰/۸	وجود مؤلفه‌ای مجزا جهت انجام امور واکنشی و بلادرنگ سیستم، ۳-۱-۳-۱	انجام امور بلادرنگ در محدوده ماژول‌های عملیاتی مربوط به هر مورد کاربری انجام می‌شود و ماژول تفکیک‌شده‌ای برای این منظور وجود ندارد.	تدارک ماژول مجزا برای انجام واکنش‌های بلادرنگ سیستم.	A	۰/۵

جدول ۴: مقایسه عملکرد روش پیشنهادی با ATAM در کشف نقص‌های معماری در مطالعه موردی.

	روش پیشنهادی					ATAM					
	مشکل مربوط	نقص‌های پیشنهادی	آزمون مسبب کشف نقص	سهام‌داران) پیشنهاددهنده مشکل	سهام‌داران) پیشنهاددهنده نقص (نقایص)	آشکارشدن نقص	ذکر نیازمندی مربوط در Business Drivers	وجود رویکرد معماری جهت رفع مشکل قبل از ارزیابی	حضور در سناریوها: درخت سودمندی	حضور در سناریوها: طوفان مغزی	شرایط رویکرد ارزیابی‌شده	آشکارشدن نقص
نقص معماری ۱	۲	۲-۶-۱	۲-۶-۱-۱	مشتری	طراح تفصیلی	✔	✔	–	✔	–	Risk	✔
نقص معماری ۲	۲	۲-۱-۲	۲-۱-۲-۱	مشتری	طراح داده	✔	✔	–	–	–	–	–
نقص معماری ۳	۳	۳-۱-۳	۳-۱-۳-۱	مدیر کیفیت	معمار	✔	✔	–	–	✔	Trade-off	✔

سهام‌داران در تعریف و پالایش ویژگی‌های کیفی، موضوع مهمی است که باعث به وجود آمدن تعریف واضح‌تر و ایجاد درک مشترکی از ویژگی‌های کیفی بین اعضای تیم سازنده و دیگر سهام‌داران می‌گردد و موفقیت سیستم را تا حدود زیادی تضمین می‌نماید. اما با وجود این که تاکنون روش‌های زیادی در ارزیابی معماری نرم‌افزار ارائه شده‌اند، در اکثریت قریب به اتفاق این روش‌ها، در تعریف ویژگی‌های کیفی به "بایدها"ی پروژه توجه می‌شود اما کمتر روشی وجود دارد که به "نبایدها"یی که لازم است از آنها اجتناب شود پرداخته باشد. تحلیل مشکلات ناشی از عدم تأمین ویژگی کیفی در سیستم می‌تواند دیدگاه متفاوت و مکملی به معمار و طراحان سیستم بدهد که برای شناسایی دقیق‌تر نیازمندی‌های کیفی و شناسایی نقص‌های معماری در پاسخگویی به آن نیازها بسیار ضروری است.

در روش پیشنهادی در این مقاله مکانیزمی استفاده می‌شود که در آن با نظر سهام‌داران ابعاد حساس و مشکلات بالقوه سیستم بر حسب اولویت شناسایی شده و به عنوان مبنای ارزیابی و اصلاح معماری در اختیار معمار قرار می‌گیرد. بدین ترتیب کلیه تصمیمات معماری در جهت تأمین ابعاد حساس سیستم اتخاذ می‌گردند و استفاده بهینه از بودجه و زمان به عمل می‌آید. در این مقاله یک توصیف الگوریتمی از رویه پیشنهادی مطرح شد و مؤلفه‌های اصلی روش و ارتباطات آنها در قالب یک مدل کلاس ارائه گردید که به عنوان زیربنای طراحی یک ابزار برای مکانیزه‌سازی رویه استفاده می‌شود. رویه پیشنهادی بر روی یک سیستم واقعی به کار گرفته شد و نتایج مربوطه ارائه گردید تا نحوه عملکرد رویه به طور واضح‌تری تشریح شود.

پیچیدگی‌های مربوط به ارزیابی معماری و این که در اکثر مواقع نرم‌افزار اصلی در هنگام ارزیابی معماری آن هنوز وجود ندارد باعث می‌شود که روش‌های ارزیابی از پارامترهای غیر قطعی که از سیستم‌های مشابه قبلی و یا تجارب اعضای تیم سازنده حاصل می‌شود استفاده نمایند. این موضوع تا حدودی دقت نتایج را تحت تأثیر قرار می‌دهد. رویه معرفی‌شده در این تحقیق نیز به لحاظ ماهیت از این محدودیت مبرا نیست. تهیه یک فهرست ناقص از مشکلات و یا نقص‌های پیشنهادی می‌تواند عملکرد رویه را تضعیف نماید، لذا بازبینی دقیق لیست‌های مذکور لازمه موفقیت کار است. از سوی دیگر تعیین فاکتورهایی مانند هزینه مشکل و یا میزان تأثیر نقص در تولید مشکل با استفاده از اطلاعات حاصل از سیستم‌های مشابه قبلی و تجربیات معمار و دیگر اعضای تیم سازنده انجام می‌شود که ممکن است به نوبه خود دچار خطا باشند. با وجود محدودیت‌های ذکرشده، رویه پیشنهادی می‌تواند در شناسایی نقص‌هایی که ممکن است از حوزه تأثیر روش‌های دیگر در امان مانده باشند بسیار مؤثر باشد. لذا از این رویه می‌توان هم برای ارزیابی معماری و هم به عنوان یک رویه تکمیلی در کنار روش‌های دیگر جهت شناسایی نقایص و اصلاح معماری استفاده نمود.

دادن وجوه رسمی‌تر به رویه پیشنهادی با تطبیق آن با توصیفات رسمی و یا زبان‌های توصیف معماری و ایجاد یک ابزار برای خودکارسازی این رویه، از کارهای در حال انجام است و در آینده کامل خواهد شد.

۷- سپاس‌گزاری

از اعضای تیم پروژه سمپا به ویژه مدیریت پروژه، آقای مهندس سیدمهدی موسوی به خاطر همکاری در انجام مطالعه موردی این مقاله تقدیر و تشکر می‌گردد.

مراجع

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd Edition, Addison-Wesley, 2003.

[2] S. T. Albin, *The Art of Software Architecture: Design Methods and Techniques*, John Wiley & Sons, 2003.

[3] P. Kruchten, *The Rational Unified Process: An Introduction*, 2nd Edition, Addison-Wesley, 2000.

[4] R. Kazman, M. Klein, and P. Clements, *ATAM: Method for Architecture Evaluation*, Technical Report, CMU/SEI-2000-TR-004, ESC-TR-2000-004, 2000.

[5] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley, 2002.

[6] "CBAM: Cost Benefit Analysi Method", http://www.sei.cmu.edu/ata/products_services/cbam.

[7] J. J. Li and J. R. Horgan, "Applying formal description techniques to software architectural design," *J. of Computer Communications*, vol. 23, no. 12, pp. 1169-1178, Jul. 2000.

[8] X. He, H. Yu, T. Shi, J. Ding, and Y. Deng, "Formally analyzing software architectural specifications using SAM," *The J. of Systems and Software*, vol. 71, no. 12, pp. 11-29, 2004.

[9] J. Xu, "Evaluating and balancing reliability and performance properties of software architecture using formal modeling techniques," in *Proc. 30th Annual IEEE/NASA Software Engineering Workshop*, pp. 212-222, Apr. 2006.

[10] J. J. P. Tsai and K. Xu, "A comparative study of formal verification techniques for software architecture specifications," *Annals of Software Engineering*, vol. 10, no. 1-4, pp. 207-223, 2000.

[11] L. Dai, Formal Design Analysis Framework: An Aspect-Oriented Architectural Framework, Ph.D. Dissertation, the University of Texas at Dallas, 2005.

[12] K. Cooper, L. Dai, and Y. Deng, "Performance modeling and analysis of software architectures: an aspect-oriented UML based approach," *Science of Computer Programming*, vol. 57, no. 1, pp. 89-108, Jul. 2005.

[13] L. Dai and K. Cooper, "Using FDAF to bridge the gap between enterprise and software architectures for security," *Science of Computer Programming*, vol. 66, no. 1, pp. 87-102, Apr. 2007.

[14] T. J. Dolan, *Architecture Assessment of Information-System Families*, Ph.D. Thesis, Department of Technology Management, Eindhoven-University of Technology, Feb. 2002.

[15] B. Tekinerdogan, H. Sozer, and M. Aksit, "Software architecture reliability analysis using failure scenarios," *J. of Systems and Software*, vol. 81, no. 4, pp. 558-575, Apr. 2008.

[16] J. B. Dugan, "Software system analysis using fault trees," in Lyu, M. R. (Ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, pp. 615-659, Chapter 15, 1996.

[17] N. E. Fenton and M. Neil, "A critique of software defect prediction models," *IEEE Trans. on Software Engineering*, vol. 25, no. 5, pp. 675-689, Sep. 1999.

[18] IEEE Std. 1044-2009, IEEE Standard Classification for Software Anomalies, pp. C1-15, 2010.

[19] J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control, and Information*, Prentice-Hall, Upper Saddle River, NJ, 1999.

سیدمهران شرفی تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد مهندسی کامپیوتر گرایش نرم‌افزار به‌ترتیب در سال‌های ۱۳۷۵ و ۱۳۷۸ از دانشگاه آزاد اسلامی واحد نجف آباد و در مقطع دکتری کامپوتر گرایش نرم‌افزار در سال ۱۳۸۶ از دانشگاه آزاد اسلامی واحد علوم و تحقیقات تهران به پایان رسانده است و هم‌اکنون استادیار دانشکده مهندسی کامپیوتر دانشگاه آزاد اسلامی واحد نجف آباد می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: مهندسی نرم‌افزار، معماری نرم‌افزار، روش‌های رسمی، مدل‌سازی و ارزیابی کارایی و ارزیابی و راستی‌آزمایی نرم‌افزار.