برآورد کمی خصیصه‌های کارایی، قابلیت اطمینان و امنیت در سبک‌های داده مشترک، شیءگرا و لوله و صافی

هدی بانکی و سیدمرتضی بابامیر

چکیده: یک نرم‌افزار مطلوب باید قادر باشد خصیصه‌های کیفی مورد نیاز سیستم را علاوه بر نیازهای وظیفه‌مندی محقق کند. سبک‌های معماری نرم‌افزار علاوه بر توصیف نرم‌افزار و تجزیه آن به مؤلفه‌ها، تأثیر عمده‌ای بر روی خصیصه‌های کیفی نرم‌افزار طراحی‌شده دارند. تحلیل و ارزیابی کمی میزان این تأثیرگذاری سبب می‌شود مناسب‌ترین سبک برای طراحی معماری انتخاب گردد. در این مقاله برای ارزیابی کمی سه خصیصه کاندید موسوم به خصیصه‌های کیفی کارایی، قابلیت اطمینان و امنیت در سه سبک کاندید موسوم به سبک‌های معماری داده مشترک، شیءگرا و لوله و صافی، روشی مبتنی بر شبکه پتری رنگی را ارائه می‌دهیم که محدودیت روش‌های گذشته برای ارزیابی این خصیصه‌ها در سبک‌ها را ندارد. در این روش ابتدا سبک‌های کاندید را با استفاده از شبکه پتری رنگی مدل می‌کنیم. سپس با توجه به قواعدی که برای ارزیابی بیان می‌کنیم با ابزار CPNTools شبکه‌ها را تحلیل و مقدار خصیصه‌های کاندید را محاسبه می‌کنیم. در پایان با استفاده از رتبه‌بندی سبک‌ها از نظر میزان تحقق خصیصه‌های کیفی کاندید، بهترین سبک کاندید را برای پیاده‌سازی مشخص می‌کنیم. برای ارائه یک نمونه عملی در استفاده از روش پیشنهادی، سیستم خودپرداز را به عنوان یک مورد مطالعه انتخاب کردیم.

کلیدواژه: **ارزیابی کمی، امنیت، سبک‌های معماری نرم‌افزار، شبکه‌های پتری رنگی، قابلیت اطمینان، کارایی.**

۱- مقدمه

در روند توسعه نرم‌افزار، طراحی‌های معماری نقش مهمی را به عنوان پلی میان نیازمندی‌ها و پیاده‌سازی ایفا می‌کند. طراحی معماری نرم‌افزار در صورتی موفق خواهد بود که قادر باشد علاوه بر نیازهای وظیفه‌مندی سیستم، نیازهای غیر وظیفه‌مندی یا همان خصیصه‌های کیفی مورد نیاز را نیز فراهم نماید. سبک‌های معماری نرم‌افزار علاوه بر توصیف نرم‌افزار و تجزیه آن به مؤلفه‌ها، تأثیر عمده‌ای بر روی خصیصه‌های کیفی نرم‌افزار طراحی‌شده دارند. بنابراین انتخاب یک سبک معماری مناسب برای سیستم بسیار ضروری می‌باشد [۱] زیرا تصمیمات نادرست، علاوه بر این که مانع رسیدن به خصیصه‌های کیفی مطلوب می‌شود، سبب می‌شود هزینه و زمان زیادی که برای طراحی به کار رفته است، اتلاف گردد. عدم تحلیل کمی تأثیر سبک‌های معماری نرم‌افزار بر روی خصیصه‌های کیفی، مانع استفاده مؤثر از سبک‌های معماری می‌گردد زیرا عامل تعیین‌کننده در انتخاب سبک معماری نرم‌افزار، میزان حمایت آن سبک از صفات کیفی مورد نظر می‌باشد. بنابراین استفاده از روش یا مدلی برای ارزیابی کمی خصیصه‌های کیفی در سبک‌ها، طراحان سیستم را قادر می‌سازد با دقت بیشتری تصمیمات طراحی را اتخاذ نمایند.

تا کنون روش‌های مختلفی برای ارزیابی کمی خصیصه‌های کیفی در سبک‌های مختلف معماری ارائه شده است که دارای مشکلات و محدودیت‌هایی می‌باشند. در این مقاله به منظور رفع این محدودیت‌ها از شبکه پتری رنگی و ابزار CPNTools برای مدل‌سازی سبک‌ها و ارزیابی خصیصه‌های کیفی استفاده می‌کنیم. با استفاده از شبکه پتری رنگی فقط می‌توان خصیصه‌هایی را بررسی و ارزیابی نمود که مربوط به جنبه‌های رفتاری معماری باشند [۲]. در این مقاله سه خصیصه کیفی کاندید موسوم به خصیصه‌های کارایی، قابلیت اطمینان و امنیت را که مربوط به جنبه‌های رفتاری معماری نرم‌افزار می‌باشند، در سه سبک کاندید موسوم به سبک‌های داده مشترک، شیءگرا و لوله و صافی ارزیابی می‌کنیم. به این ترتیب امکان مقایسه سبک‌های کاندید را از نظر خصیصه‌های کاندید فراهم می‌کنیم. همچنین در پایان، سبک‌های کاندید را با توجه به خصیصه‌های کیفی کاندید رتبه‌بندی می‌کنیم و به این ترتیب مناسب‌ترین سبک برای پیاده‌سازی نرم‌افزار را مشخص می‌کنیم.

۲- کارهای مرتبط

هدف از ارزیابی معماری سیستم‌های نرم‌افزاری، تحلیل معماری برای شناسایی ریسک‌های احتمالی و حصول اطمینان از برآورده‌شدن نیازهای کیفی در طراحی آن است. بنابراین تصمیم‌گیری در مورد انتخاب یک معماری مناسب برای سیستم و زیرسیستم‌های مشابه آن در حین توسعه یک سیستم نرم‌افزاری بسیار ضروری است [۱]. روش‌های ارزیابی کمی معماری با جمع‌آوری معیارها و نمونه‌سازی و شبیه‌سازی معماری انجام می‌شود. یکی از راه‌های بهره‌گیری از معیارها، مدل‌سازی معماری نرم‌افزار با مدل‌های ریاضی است تا با استفاده از این مدل‌ها برای دستیابی به اطلاعات و آمارهای مربوط به معماری مانند میانگین زمان اجرای یک مؤلفه و ارزیابی خصیصه‌های کیفی عملیاتی مانند کارایی و قابلیت اطمینان استفاده شود. این مدل‌ها برای مشخص‌کردن مسیر اجرایی در یک سیستم نرم‌افزاری از سناریو استفاده می‌کنند و سپس مسیرها را با جزئیات بررسی می‌کنند [۳].

در [۴] خصیصه‌های کیفی به دو دسته اصلی طبقه‌بندی شده‌اند: ۱) خصیصه‌های مربوط به توسعه سیستم (مانند قابلیت نگهداری و انعطاف‌پذیری) و ۲) خصیصه‌های عملیاتی (مانند کارایی و قابلیت اطمینان). شبیه‌سازی و مدل‌های ریاضی برای خصیصه‌های کیفی عملیاتی مناسب می‌باشند. ارزیابی مبتنی بر مدل ریاضی نیازمند داده‌هایی از سابقه اجرای مؤلفه‌های نرم‌افزاری در گذشته می‌باشد. همچنین ارزیابی مبتنی بر مدل ریاضی برای سیستم‌های نرم‌افزاری مبتنی بر مؤلفه مناسب می‌باشد.

این مقاله در تاریخ ۳۰ شهریور ماه ۱۳۹۳ دریافت و در تاریخ ۲۹ مهر ماه ۱۳۹۴ بازنگری شد. این تحقیق توسط دانشگاه کاشان بر اساس قرارداد شماره ۴۰۵۲۹۵ پشتیبانی شده است.

هدی بانکی، دانشکده مهندسی برق و کامپیوتر، دانشگاه کاشان، کاشان، (email: banki@grad.kashanu.ac.ir).

سیدمرتضی بابامیر، دانشکده مهندسی برق و کامپیوتر، دانشگاه کاشان، کاشان، (email: babamir@kashanu.ac.ir).

تا کنون برای ارزیابی خصیصه‌های کیفی عملیاتی در معماری نرم‌افزار برخی از روش‌ها و مدل‌های ریاضی توسعه داده شده‌اند. روش‌های مختلفی برای ارزیابی قابلیت اطمینان ارائه شده که مبتنی بر حالت می‌باشند. در مدل‌های مبتنی بر حالت، رفتار معماری نرم‌افزار به صورت تعامل میان مؤلفه‌ها مدل می‌شود. برای مثال در [۵] با استفاده از مدل مارکوف زمان گسسته روشی برای ارزیابی قابلیت اطمینان ارائه شده که با توجه به قابلیت اطمینان هر مؤلفه و احتمال گذارها، قابلیت اطمینان معماری نرم‌افزار را محاسبه می‌کند. در [۶] نیز برای ارزیابی قابلیت اطمینان مدلی مبتنی بر مدل مارکوف زمان گسسته ارائه شده که در این مدل با توجه به احتمال گذارها، قابلیت اطمینان هر مؤلفه و تعداد دفعاتی که یک مؤلفه ملاقات و اجرا می‌شود، قابلیت اطمینان معماری نرم‌افزار محاسبه می‌شود. در [۷] برای ارزیابی کمی قابلیت اطمینان، روشی بر اساس شبکه بیزین پویا ارائه شده است. برای ارزیابی کمی کارایی در معماری نرم‌افزار نیز در [۸] روشی توسعه داده شده که بر اساس مدل مارکوف زمان گسسته عمل می‌کند و با توجه به زمان مصرف‌شده در هر یک از مؤلفه‌ها و تعداد دفعاتی که مؤلفه‌ها در طول اجرای برنامه ملاقات می‌شوند، کارایی معماری نرم‌افزار را محاسبه می‌کند. در [۹] برای ارزیابی کمی خصیصه‌های کیفی کارایی، قابلیت اطمینان و امنیت بر اساس مدل مارکوف زمان گسسته روشی ارائه شده که بر اساس میزان آسیب‌پذیری مؤلفه‌ها و تعداد دفعات ملاقات مؤلفه‌ها در طول اجرای برنامه، امنیت معماری نرم‌افزار را محاسبه می‌نماید. همچنین برای ارزیابی امنیت در [۱۰] بر اساس شبکه پتری آماری مدلی ارائه شده است. در [۲] نیز بر اساس شبکه‌های پتری رنگی برای ارزیابی کمی خصیصه‌های قابلیت اطمینان، کارایی و امنیت در معماری نرم‌افزار روشی ارائه شده است. در حالی که روش‌های کمی فوق‌الذکر سبک‌های معماری را در روش خود در نظر نمی‌گیرند، روش کمی ارائه شده در این مقاله، سبک‌های معماری را در ارزیابی خصیصه‌های کیفی مد نظر قرار می‌دهد.

یکی از مهم‌ترین گام‌ها در طراحی معماری نرم‌افزار، انتخاب یک سبک معماری است که برای طراحی نرم‌افزار مناسب باشد. انتخاب یک سبک معماری مناسب برای سیستم، کمک می‌کند تا ویژگی‌های مطلوب سیستم برآورده شوند. سبک‌های مختلف معماری، خصوصیات سیستمی متفاوتی دارند. به طور کلی سبک معماری مؤلفه‌ها و اتصالات مختلفی را تعریف می‌کند. همچنین قیودی را بیان می‌کند که این قیود مشخص می‌نماید چگونه این مؤلفه‌ها با استفاده از اتصالات با یکدیگر ترکیب شوند تا یک سیستم نرم‌افزاری را توسعه دهند. عامل تعیین‌کننده در انتخاب سبک معماری نرم‌افزار، میزان حمایت آن سبک از صفات کیفی مورد نظر می‌باشد. در جهت کمی‌سازی تأثیر سبک‌های معماری نرم‌افزار تا کنون تحقیقاتی انجام شده است. در این روش‌ها سبک‌های معماری نرم‌افزار با استفاده از معادلات و مدل‌های ریاضی مدل‌سازی می‌شوند.

بر اساس روش‌هایی که برای ارزیابی کمی خصیصه‌ها در معماری نرم‌افزار ذکر شد، روش‌هایی نیز برای ارزیابی کمی این خصیصه‌ها در سبک‌های معماری نرم‌افزار ارائه شده است. با استفاده از تاکتیک‌ها، مؤلفین این مقاله در [۱۱] به رتبه‌بندی سبک‌های لوله و صافی، لایه‌ای، تخته سیاه، کارگزار- مشتری و واسط در تحقق خصیصه‌های کیفی قابلیت دسترسی، امنیت و کارایی پرداخته‌اند. در [۱۲] بر اساس مدل مارکوف زمان گسسته، یک مدل تحلیلی برای ارزیابی قابلیت اطمینان معماری ناهمگن که متشکل از سبک‌های ترتیبی- دسته‌ای، فراخوانی- بازگشت، موازی و تحمل‌پذیر خطا می‌باشد، ارائه شده که با توجه به قابلیت اطمینان هر مؤلفه و احتمال گذارها، قابلیت اطمینان سبک‌های مورد نظر را محاسبه می‌کند. در [۱۳] از مدل مارکوف برای ارزیابی کمی خصیصه کیفی قابلیت اعتماد در یک سیستم صنعتی با مقیاس بزرگ شامل ۱۰۰ مؤلفه در ۹ زیرسیستم استفاده شده است. در [۱۴] نیز برای ارزیابی کارایی سبک‌های معماری که شامل سبک‌های ترتیبی- دسته‌ای، فراخوانی- بازگشت، موازی و تحمل‌پذیر خطا می‌باشد بر اساس مدل مارکوف زمان گسسته مدلی ارائه گردیده که با توجه به زمان مصرف‌شده در هر یک از مؤلفه‌ها و تعداد دفعاتی که مؤلفه‌ها در طول اجرای برنامه ملاقات می‌شوند، کارایی سبک‌های مورد نظر را محاسبه می‌کند.

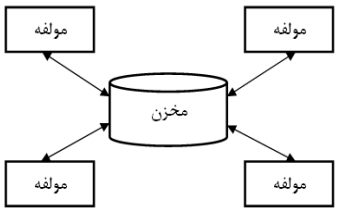
در این مقاله بر اساس روش ارائه‌شده در [۲]، سه خصیصه کاندید موسوم به خصیصه‌های کارایی، قابلیت اطمینان و امنیت را در سه سبک کاندید موسوم به سبک‌های داده مشترک، شیءگرا و لوله و صافی، ارزیابی کمی می‌کنیم. در [۲] با استفاده از شبکه پتری رنگی یک روش کلی برای ارزیابی معماری نرم‌افزار ارائه شده که ما در این مقاله از آن استفاده کرده‌ایم و پس از ایجاد تغییراتی در آن به منظور کامل‌تر شدن روش، آن را برای ارزیابی سبک‌های کاندید تعمیم داده‌ایم و این سبک‌ها را رتبه‌بندی کرده‌ایم. از آنجا که هر یک از این سبک‌ها از خصیصه‌های کیفی متفاوتی پشتیبانی می‌کنند، انتخاب یک سبک معماری مناسب برای سیستم، کمک می‌کند تا ویژگی‌های مطلوب سیستم برآورده شوند.

در تحقیقاتی که تا کنون انجام شده است، سه خصیصه کاندید در سبک‌های معماری فقط با استفاده از مدل مارکوف مورد ارزیابی کمی قرار گرفته‌اند. به طور کلی ساخت مدل مارکوف برای مؤلفه‌های نرم‌افزاری به خصوص در برنامه‌هایی با مقیاس بزرگ نسبتاً مشکل می‌باشد [۱۵]. یکی دیگر از محدودیت‌های مدل مارکوف این است که این مدل برای مدل‌سازی با رفتار احتمالی مستقل از تاریخچه مناسب می‌باشد [۱۶] و به همین دلیل مدل‌سازی برنامه‌هایی که با توجه به تاریخچه اجرا مشخص می‌کنند که در مرحله بعد چه مؤلفه‌ای باید اجرا شود، امکان‌پذیر نیست. شبکه‌های پتری محدودیت‌های مدل مارکوف را ندارد و به این دلیل در این مقاله برای ارزیابی خصیصه‌های کیفی کاندید در سبک‌های کاندید از شبکه پتری رنگی استفاده کرده‌ایم.

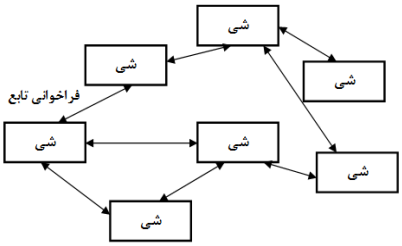
۳- مفاهیم پایه

۳-۱ سبک‌های معماری نرم‌افزار

یک سبک معماری مجموعه‌ای از مؤلفه‌ها و اتصالات به همراه مجموعه‌ای از قیود است که این قیود چگونگی ترکیب‌شدن عناصر را مشخص می‌کنند [۱۷]. سبک‌های معماری نرم‌افزار علاوه بر توصیف نرم‌افزار و تجزیه آن به مؤلفه‌ها، تأثیر عمده‌ای بر روی خصیصه‌های کیفی نرم‌افزار طراحی‌شده دارند و میزان پشتیبانی هر یک از سبک‌های معماری از خصیصه‌های کیفی، متفاوت می‌باشد. بنابراین برای رسیدن به خصوصیات کیفی مطلوب سیستم باید سبک معماری مناسب انتخاب گردد. به طور کلی تعاریف متفاوتی برای کیفیت وجود دارد. بر طبق استاندارد ISO ۹۱۲۶ [۱۸] خصیصه کیفی نرم‌افزار به این صورت تعریف می‌شود: "درجه‌ای که یک سیستم، یک مؤلفه و یا یک پردازه، انتظارات و نیازمندی‌های کاربر یا مشتری را فراهم می‌کند". یکی از مشکل‌ترین اعمال در معماری نرم‌افزار، طراحی یک سیستم نرم‌افزاری است که مجموعه‌ای از خصیصه‌های کیفی را برآورده کند. در این مقاله سه سبک رایج داده مشترک، شیءگرا و لوله و صافی که آنها را سبک‌های کاندید می‌نامیم از نظر خصیصه‌های کیفی کارایی، قابلیت اطمینان و امنیت که آنها را خصیصه‌های کاندید می‌نامیم، بررسی می‌کنیم.



شکل ۱: سبک مخزن.



شکل ۲: سبک شیءگرا.

- سبک‌های مخزن، شیءگرا و لوله و صافی

سبک مخزن یا داده مشترک (شکل ۱) دارای دو نوع مؤلفه است. یک انباره مرکزی به نام مخزن و مجموعه‌ای از مؤلفه‌های مستقل که بر روی انباره مرکزی عملیات انجام می‌دهند [۱۹]. در این سبک مؤلفه‌های مستقل با مخزن از طریق خواندن از/ نوشتن در/ مخزن، تعامل دارند. عناصر این سبک عبارتند از ۱) مخزن که شامل اطلاعاتی در مورد انواع داده‌های ذخیره‌شده، داده‌های توزیع‌شده و تعداد دسترسی‌های مجاز می‌باشد، ۲) مؤلفه دستیابی به داده و ۳) اتصالگر خواندن و نوشتن داده.

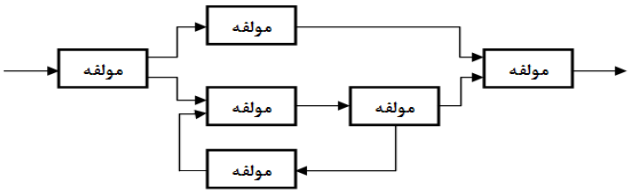
در سبک شیءگرا (شکل ۲)، نمایش داده و عملیاتی که با آنها در ارتباط هستند، در یک شیء محصور می‌شوند. مؤلفه‌های این سبک اشیا هستند که به وسیله فراخوانی توابع با یکدیگر در ارتباط می‌باشند [۱۷]. عناصر این سبک عبارتند از ۱) شیء، مؤلفه‌هایی که از طریق فراخوانی روال‌ها با یکدیگر تعامل می‌کنند و ۲) اتصال‌دهنده‌های فراخوانی-بازگشت که به وسیله یک شیء استفاده می‌شود تا روال‌های اشیای دیگر را فراخوانی کند.

سبک لوله و صافی (شکل ۳) از تعدادی مؤلفه تشکیل شده که هر یک در نقش یک صافی هستند.

داده‌ها از درگاه‌های ورودی صافی وارد شده، سپس پردازش می‌گردد و به وسیله درگاه‌های خروجی از طریق یک لوله به صافی بعدی انتقال داده می‌شود [۱۹]. عناصر این سبک عبارتند از ۱) صافی‌ها که داده را از درگاه ورودی دریافت و به داده قابل استفاده برای درگاه خروجی تبدیل می‌کنند و سپس آن را به درگاه خروجی ارسال می‌کنند و ۲) لوله‌ها که اتصالات بین صافی‌ها هستند.

۳-۲ خصیصه‌های کیفی نرم‌افزار

به طور کلی تعاریف متفاوتی برای ویژگی‌های کیفی نرم‌افزار وجود دارد. بر طبق استاندارد ISO۹۱۲۶ [۱۸]، ویژگی کیفی نرم‌افزار به این صورت تعریف می‌شود: "درجه‌ای که یک سیستم، یک مؤلفه و یا یک پردازه، انتظارات و نیازمندی‌های کاربر یا مشتری را فراهم می‌کند". این استاندارد دارای شش مشخصه است که ویژگی‌های کیفی نرم‌افزار که کمترین هم‌پوشانی را دارند، توصیف می‌کند. این شش مشخصه عبارتند از کارایی، قابلیت اطمینان، قابلیت استفاده، بهره‌وری، قابلیت نگهداری و قابلیت انتقال. حال به تعریف سه خصیصه کاندید از دید این استاندارد می‌پردازیم:



شکل ۳: سبک لوله و صافی.

- **کارایی:** کارایی دارای زیرخصیصه‌های زمان مصرفی و فضای مصرفی می‌باشد. زمان مصرفی به این معناست که نرم‌افزار زمان اجرای مناسب داشته باشد و بتواند در این زمان پاسخ را آماده نماید. فضای مصرفی به این معناست که نرم‌افزار از تعداد مناسبی از منابع استفاده کند و فضای مصرفی آن برای ذخیره و استفاده منابع به صورت کارآمد باشد.
- **امنیت:** به توانایی نرم‌افزار در حفاظت اطلاعات و داده اشاره دارد. بنابراین افراد و سیستم‌های غیر مجاز نمی‌توانند داده‌ها را بخوانند یا اصلاح کنند و برای افراد و سیستم‌های مجاز منعی برای دسترسی وجود ندارد [۱۹].
- **قابلیت اطمینان:** توانایی نرم‌افزار برای این که بتواند به صورت صحیح کار کند و دچار شکست نشود و این شکست ناشی از خطاهایی است که در نرم‌افزار به وجود می‌آید. همچنین نرم‌افزار بتواند داده‌ای را که به صورت مستقیم تحت تأثیر شکست قرار گرفته و آسیب دیده است ترمیم کند.

۳-۳ شبکه پتری رنگی

شبکه پتری رنگی به وسیله Jensen ارائه شد [۲۰] که یک زبان گرافیکی برای مدل‌سازی سیستم‌های هم‌روند و رخداد گسسته است و توانایی‌های شبکه پتری سطح پایین (غیر رنگی) را با توانایی‌های یک زبان برنامه‌نویسی سطح بالا (عموماً زبان تابعی ML) ترکیب می‌کند. پتری رنگی، یک ۹تایی است که به صورت $N=(P,T,A,\Sigma,C,N,E,G,I)$ تعریف می‌شود. مجموعه‌های P، T، A و Σ به ترتیب معرف مجموعه مکان‌ها، گذارها، قوس‌ها و الفبا هستند که الفبا مشتمل بر رنگ‌ها، عملیات و توابع مورد استفاده در شبکه است

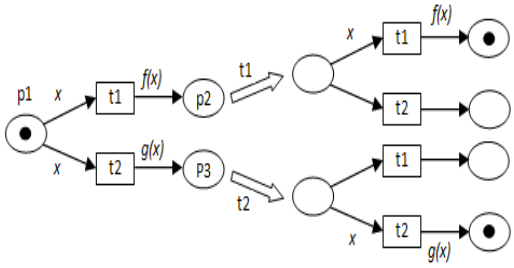
$$\begin{aligned} P &= \{p_1, p_2, \ldots, p_n\} \\ T &= \{t_1, t_2, \ldots, t_m\} \\ A &= \{a_1, a_2, \ldots, a_k\} \\ \Sigma &= \{\sigma_1, \sigma_2, \ldots, \sigma_j\} \end{aligned} \quad (۱)$$

توابع C، N، E، G و I به ترتیب معرف تابع رنگ، تابع گره، تابع عبارت، تابع محافظ و تابع مقداردهی اولیه به صورت زیر است که e یک عبارت، g یک شرط و i یک مقدار اولیه در یک مکان است

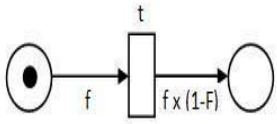
$$\begin{aligned} &C: P \to \Sigma\ ,\ E: A \to e\ ,\ N: A \to P\times T \cup T\times P\ , \\ &G: T \to g\ ,\ I: P \to i \end{aligned} \quad (۲)$$

در هر مکان یک یا تعدادی نشانه رنگی قرار می‌گیرد که هر رنگ معرف یک نوع داده است. توابع گره و عبارت اجازه می‌دهند تا بتوان تعدادی قوس را به یک جفت گره با عبارات متفاوت متصل کرد. خروجی تابع محافظ یک مقدار درست یا نادرست است.

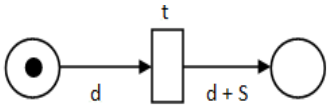
در شبکه پتری رنگی مکان‌ها به صورت دایره یا بیضی و گذارها به صورت مستطیل نمایش داده می‌شوند. مکان‌ها و گذارها به وسیله یال‌های مستقیمی به یکدیگر متصل می‌شوند. هر مکان دارای نوع و



شکل ۴: ارزیابی خصیصه‌های کیفی در شبکه پتری رنگی [۲].



شکل ۵: ارزیابی قابلیت اطمینان [۲].



d : مجموع احتمالات تفوذ به شبکه

S : احتمال تفوذ مخرب‌ها به گذار t در شبکه

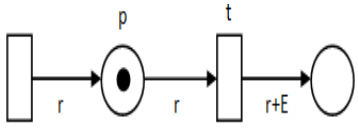
شکل ۶: ارزیابی امنیت شبکه [۲].

تعدادی نشانه است که نوع معرف نوع صحیح، اعشاری، حرف، رشته یا نوع تعریف‌شده به وسیله کاربر برای نشانه‌های مکان است. در حقیقت، نوع نشان می‌دهد که نشانه‌های یک مکان دارای چه محدوده‌ای از مقادیر داده‌ای می‌توانند باشند. این مقدار داده، رنگ نشانه نامیده می‌شود. همچنین گذارها می‌توانند شرطی باشند که در صورت تحقق شرط، گذار آتش می‌شود. برای آتش‌شدن یک گذار علاوه بر تحقق شرط آن، گذار باید توانا باشد. یک گذار وقتی توانا است که نشانه‌های موجود در مکان ورودی به آن گذار شروط مندرج روی یال بین مکان و گذار را محقق کند. شبکه پتری احتمالی یک شبکه پتری است که گذارهای توانا در آن با احتمال مشخصی آتش می‌شوند و شبکه پتری زمانی یک شبکه پتری است که در آن گذارهای توانا پس از سپری‌شدن مدت زمان معینی آتش می‌شوند.

دلیل استفاده از شبکه پتری رنگی این است که نشانه‌های موجود در یک مکان می‌تواند نوع‌های مختلفی داشته باشد. فرض کنید بخواهیم در یک معماری فراخوانی یک مؤلفه را توصیف کنیم، مؤلفه فراخوانی‌شده ممکن است توسط چندین مؤلفه فراخوانی شود و به صورت هم‌زمان اجرا شود، در این حالت، نقاط اجرا که به وسیله نشانه‌ها علامت‌گذاری می‌شوند برای هر فراخوانی‌کننده متفاوت است و مؤلفه فراخوانده‌شده باید اطلاعات مربوط به این که هر فراخواننده‌ای متناظر با چه نشانه‌ای است را نگهداری کند تا کنترل اجرا را به فراخواننده صحیح بازگرداند. برای مشخص‌کردن فراخوانی‌کننده باید شناسه مؤلفه به نشانه مربوطه پیوست شود [۲].

۳-۳-۱ ارزیابی خصیصه‌های کیفی با شبکه‌های پتری رنگی

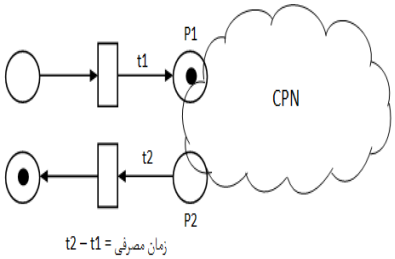
شکل ۴ چگونگی ارزیابی خصیصه‌های کیفی را با استفاده از پیوست‌کردن مقادیر به نشانه‌ها نشان می‌دهد. در این شکل مقدار x بر روی گذار نشان‌دهنده مقدار خصیصه کیفی است. دو احتمال برای آتش‌شدن وجود دارد: یکی آتش‌شدن گذار t_1 و دیگری آتش‌شدن گذار t_2. پس از آتش‌شدن گذار t_1، مقدار x به $f(x)$ به روز رسانی می‌شود و به این معناست که مقدار خصیصه کیفی طی یک مرحله پیشرفت در اجرا به مقدار $f(x)$ تغییر کرده است.



r : مجموع احتمالات تفوذ به حافظه‌ها

E : احتمال تفوذ به مکان p

شکل ۷: ارزیابی امنیت حافظه و فایل.



شکل ۸: ارزیابی کارایی [۲].

۳-۳-۲ ارزیابی قابلیت اطمینان با شبکه پتری رنگی

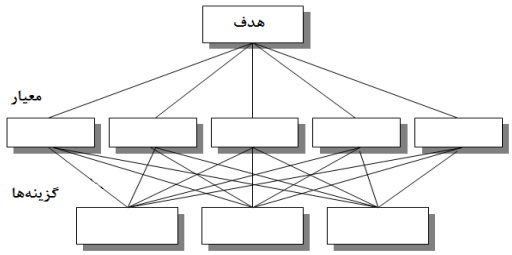
شکل ۵ نشان می‌دهد که چگونه قابلیت اطمینان در شبکه پتری رنگی محاسبه می‌شود. امکان شکست مانند مفقودشدن داده با احتمال F و احتمال عدم شکست (موفقیت) با $F-1$ نشان داده می‌شود. تجمع احتمال‌های موفقیت در طول مسیر محاسبه می‌شود و در متغیر f قرار می‌گیرد. این متغیر همراه با نشانه از یک محل به محل بعدی منتقل می‌شود. زمانی که اجرای شبکه پتری به پایان برسد، مقدار f در نشانه بیانگر مقدار قابلیت اطمینان مسیر اجراشده خواهد بود [۲].

۳-۳-۳ ارزیابی امنیت با شبکه پتری رنگی

بخشی از امنیت یک سیستم مربوط به امنیت شبکه‌ای می‌شود که سیستم مورد نظر از طریق آن به تبادل اطلاعات می‌پردازد (شکل ۶). همچنین بخشی از امنیت یک سیستم مربوط به امنیت حافظه و فایلی می‌شود که داده در آن قرار می‌گیرد (شکل ۷). نماد E میزان احتمال نفوذ به مکان p را نشان می‌دهد و مکان p متناظر با حافظه یا فایلی است که داده در آن ذخیره می‌شود. در زمانی که داده در حافظه یا فایل قرار دارد افراد مخرب می‌توانند بدون اجازه داده را کپی کنند و یا در آن تغییری ایجاد کنند. به همین دلیل مقدار امنیت بر اساس راحتی نفوذ عناصر خارجی به مکان p محاسبه می‌شود. بنابراین مقدار امنیت در این مورد برابر با جمع مقادیر در مکان‌هایی است که در آنها احتمال نفوذ عناصر خارجی وجود دارد [۲]. نماد S میزان احتمال نفوذ مخرب‌ها به شبکه را نشان می‌دهد و گذار t متناظر با مؤلفه یا اتصالی است که جنبه‌ای از یک شبکه را نشان می‌دهد. افراد مخرب می‌توانند زمانی که داده در گذار t قرار دارد داده را کپی کنند. مقدار امنیت بر اساس راحتی استراق سمع در t به دست آورده می‌شود. مقدار امنیت برابر با جمع مقادیر در گذارهایی است که در آنها احتمال دارد داده توسط عناصر خارجی کپی شود [۲].

۳-۳-۴ ارزیابی کارایی با شبکه‌های پتری رنگی

کارایی از نظر زمان لازم برای اجرای برنامه و همچنین فضای مصرفی مورد بررسی قرار می‌گیرد (شکل ۸). همان طور که در این شکل نشان داده شده است با استفاده از برچسب زمانی توکن‌ها می‌توان به راحتی محاسبه کرد که در آتش‌شدن و حرکت به مکان مقصد چه قدر زمان سپری می‌شود. فرض کنید در شکل ۸ می‌خواهیم کارایی (یعنی زمان اجرا



شکل ۹: مدل تصمیم سلسله‌مراتبی.

از P_1 تا P_2) را محاسبه کنیم. می‌توان با ارجاع به برچسب‌های زمانی و محاسبه $t_2 - t_1$، کارایی معماری را که واسطهای آن متناظر با P_1 و P_2 هستند محاسبه نمود [۲].

به منظور بررسی دقیق‌تر از لحاظ کارایی باید فضای مصرفی معماری نیز بررسی گردد و بنابراین ما در این مقاله برای ارزیابی کارایی، علاوه بر زمان مصرفی، حجم فضایی که برای ذخیره داده استفاده می‌شود را نیز مورد بررسی قرار داده‌ایم. از آنجایی که در شبکه پتری رنگی برای مدل‌کردن حافظه و فایلی که داده در آنها ذخیره می‌شود از مکان‌های مختلف استفاده می‌شود، باید تعداد و حجم مکان‌های استفاده‌شده برای ذخیره داده مورد بررسی قرار گیرند.

۳-٤ فرآیند تحلیل سلسله‌مراتبی

تحلیل یک سیستم هنگامی به کار گرفته می‌شود که تصمیم‌گیرنده بخواهد کارایی مجموعه‌ای از راهکارهای انتخابی را برای یک مسئله مشخص ارزیابی کند. در حوزه ارزیابی سبک‌های معماری، راهکارهای انتخابی بر اساس تعدادی معیار ارزیابی می‌شوند. از جمله این معیارها می‌توان نیازهای وظیفه‌مندی و نیازهای غیر وظیفه‌مندی، اولویت‌های معمار را نام برد که این معیارها لزوماً با یکدیگر همسو نیستند. به این معنا که ممکن است یک گزینه انتخابی، مجموعه‌ای از معیارها را برآورده کند ولی بقیه معیارها را برآورده نکند. پس مسئله این است که بتوان با در نظر گرفتن همزمان همه معیارها، اولویت گزینه‌ها را نسبت به یکدیگر مشخص کرد و گزینه ارجح را تعیین نمود. این امر انتخاب سبک معماری را به یک مسئله پیچیده تبدیل می‌کند. از میان روش‌های موجود که به مناسب‌ترین حالت به بررسی این نوع مسایل می‌پردازد، تصمیم‌گیری چندمعیاره نام دارد [۲۱]. از روش‌های تصمیم‌گیری چندمعیاره می‌توان به فرایند تحلیل سلسله‌مراتبی اشاره نمود.

فرایند تحلیل سلسله‌مراتبی روشی برای تصمیم‌گیری‌های چندمعیاره است که در سال ۱۹۸۰ توسط ساتی توسعه داده شده است [۲۲]. از مهم‌ترین مزایای ساختار سلسله‌مراتبی، فهم سریع و آسان آن است. در این روش یک ساختار سلسله‌مراتبی چندسطحی از اهداف، معیارها، زیرمعیارها و گزینه‌ها ساخته می‌شود و اساس کار آن بر پایه مقایسات جفتی می‌باشد و به تصمیم‌گیرندگان کمک می‌کند تا از عناصر مهم یک مسئله پیچیده چندمعیاری، یک ساختار سلسله‌مراتبی بسازند. در روش فرایند تحلیل سلسله‌مراتبی مسئله باید حداقل به سه سطح تقسیم شود:

- **سطح هدف:** در این سطح اهداف مورد نظر مشخص می‌شود. برای مثال یکی از اهداف می‌تواند انتخاب بهترین سبک معماری باشد که معمولاً در این سطح فقط یک هدف وجود دارد.
- **سطح معیار:** در این سطح معیارهای دخیل در تصمیم‌گیری مشخص می‌شوند. مثلاً در مسئله انتخاب سبک معماری، یکی از معیارها قابلیت اطمینان می‌باشد.
- **سطح گزینه**[1]: در این سطح راه حل‌های مختلف مطرح‌شده برای هدف، قرار دارند. برای مثال در مسئله انتخاب سبک معماری، انواع سبک‌های مطرح‌شده در این سطح قرار می‌گیرند.

شکل ۹ انتزاعی از یک ساختار سلسله‌مراتبی را نشان می‌دهد. خطوطی که هدف را به هر یک از معیارها متصل می‌کند به این معنا است که آن معیارها باید به صورت جفتی مقایسه شوند تا اهمیت و اولویت آنها با توجه به هدف مشخص گردد. به طور مشابه خطوطی که هر معیار را به گزینه‌ها متصل می‌کند به این معنا است که گزینه‌ها باید مقایسه جفتی شوند تا اولویت آنها با توجه به آن معیار مشخص گردد.

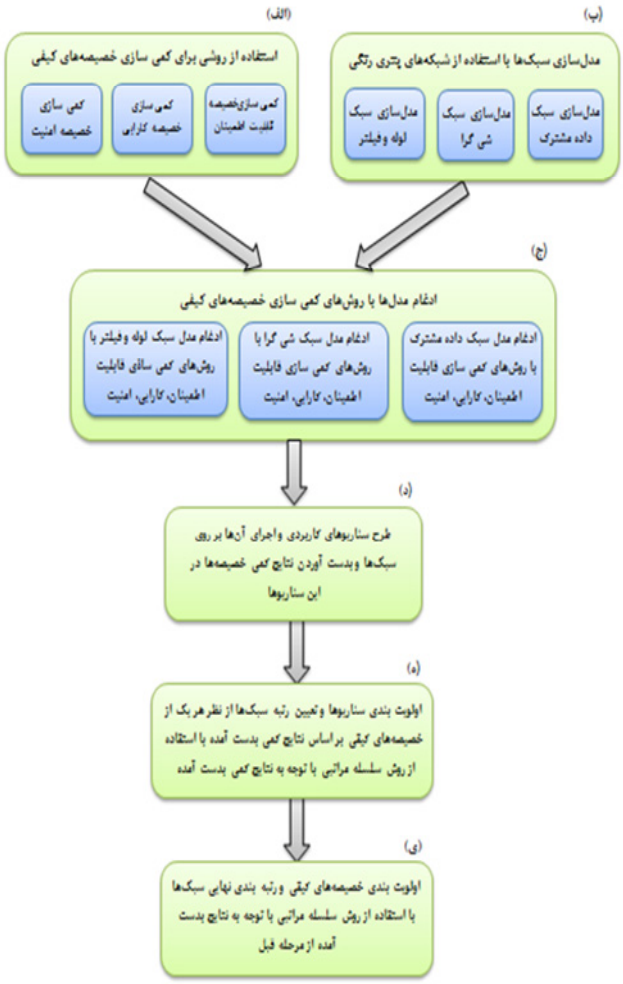
یکی از مهم‌ترین ویژگی‌های فرایند تحلیل سلسله‌مراتبی این است که نیاز به قضاوت کامل و قطعی نیست بلکه قضاوت‌های نسبی کافی هستند. دلیل منطقی این مسئله این است که اغلب، تولید عبارات مقایسه‌ای آسان‌تر از تولید یک عدد قطعی می‌باشد. مقیاس‌هایی که برای مقایسه‌های دو به دو به وسیله ساتی پیشنهاد شده است [۲۲] با اولویت‌های ۱، ۳، ۵، ۷ و ۹ مشخص می‌شود که به ترتیب متناظر با اولویت یکسان، اولویت بیشتر، اولویت بسیار بیشتر، اولویت بسیار بسیار بیشتر و اولویت به شدت بیشتر است. مقادیر زوج بین مقادیر فوق‌الذکر معرف مصالحه بین آن مقادیر است. معکوس اعداد ۳، ۵، ۷ و ۹ به ترتیب متناظر با اولویت کمتر، اولویت بسیار کمتر، اولویت بسیار بسیار کمتر و اولویت به شدت کمتر است. دلیل اصلی استفاده از فرایند تحلیل سلسله‌مراتبی برای ارزیابی سبک‌های معماری این است که این روش به صورت واضح خصوصیات کیفی مربوط به هر سبک و میزان تحقق آنها توسط آن سبک را بررسی می‌کند. لازم به ذکر است برای انجام محاسبات فرایند تحلیل سلسله‌مراتبی در این مقاله از نرم‌افزار Expertchoice [۲۳] استفاده شده است.

٤- روش پیشنهادی

در این بخش قصد داریم برای ارزیابی کمی خصیصه‌های کیفی کاندید در سبک‌های کاندید روشی را ارائه دهیم. همان طور که اشاره شد روش‌های مبتنی بر مدل ریاضی برای ارزیابی خصیصه‌های کیفی عملیاتی مناسب‌تر می‌باشند. در گذشته اغلب از روش مارکوف برای ارزیابی کمی خصیصه‌های کیفی در سبک‌ها استفاده شده است. ساخت مدل مارکوف برای مؤلفه‌های نرم‌افزاری به خصوص در برنامه‌هایی با مقیاس بزرگ، نسبتاً مشکل می‌باشد. یکی از مهم‌ترین مشکلات این است که با رشد تعداد مؤلفه‌های نرم‌افزار، فضای حالت با سرعت خیلی بیشتری رشد می‌کند [۱۶]. یکی دیگر از محدودیت‌های مدل مارکوف این است که این مدل برای مدل‌سازی با رفتار احتمالی مستقل از تاریخچه مناسب می‌باشد [۱۷]. اگر در طول زنجیره اجرا حداقل یک مؤلفه بیشتر از یک بار اجرا شود، نیاز به یک شاخص مجزا برای تفکیک نتیجه اثر تاریخچه اجرا می‌باشد و باید برای نمایش اجراهای مختلف مؤلفه، از حالت‌های مجزا استفاده شود. به همین دلیل مدل‌سازی بسیاری از نرم‌افزارها با استفاده از مدل مارکوف به راحتی امکان‌پذیر نیست. مخصوصاً برنامه‌هایی که با توجه به تاریخچه اجرا مشخص می‌کنند که در مرحله بعد چه مؤلفه‌ای باید اجرا شود. همچنین مدل مارکوف برای بیان مفاهیم موازی‌سازی و همزمانی توانایی زیادی ندارد [۲۴].

در این مقاله با الهام از روش ارائه‌شده در [۲] از شبکه پتری رنگی و ابزار CPNTools [۲۵] برای مدل‌سازی سبک‌ها و ارزیابی خصیصه‌های

1. Alternatives



شکل ۱۰: چارچوب پیشنهادی ارزیابی کمی سبک‌های معماری نرم‌افزار.

کیفی استفاده می‌کنیم. توصیف یک مسئله با شبکه پتری نسبت به مدل مارکوف پیچیدگی کمتری دارد و به درک طراح از توصیف سیستم نزدیک‌تر می‌باشد [۱۵]. همچنین برای غلبه بر کاستی‌های روش مارکوف می‌توان از شبکه پتری استفاده کرد [۲۴] زیرا شبکه پتری محدودیت‌های مدل مارکوف را ندارد. این روش برای نرم‌افزارهای وابسته به تاریخچه نیز مناسب است و محدودیتی در رشد تعداد مؤلفه‌های نرم‌افزار ندارد. در مدل مارکوف، تغییرات کوچک در طراحی سیستم باعث تغییرات زیادی در ساختار زنجیره مارکوف می‌شود و به همین دلیل توسعه سیستم‌هایی که از مدل مارکوف استفاده می‌کنند دشوار است، اما به راحتی می‌توان سیستم‌هایی که با شبکه پتری مدل می‌شوند را توسعه داد [۲۶] و [۲۷].

با شبکه پتری رنگی فقط خصیصه‌هایی را می‌توان ارزیابی و بررسی نمود که مربوط به جنبه‌های رفتاری معماری باشند [۲]. در این مقاله سه سبک رایج داده مشترک، شیءگرا و لوله و صافی را با استفاده از روش پیشنهادی برای مدل‌سازی و ارزیابی کاندید کرده‌ایم و راهکارهایی برای ارزیابی کمی سه خصیصه کیفی کاندید موسوم به کارایی، قابلیت اطمینان و امنیت ارائه داده‌ایم. روش پیشنهادی، دقت بیشتری در ارزیابی سبک‌های کاندید نسبت به راهکارهای گذشته دارد زیرا برای بررسی خصیصه‌های کیفی کاندید، پارامترهای بیشتری را در نظر گرفته‌ایم. با استفاده از روش پیشنهادی، ارزیابی جداگانه خصیصه‌ها لازم نیست و با یک بار اجرای شبکه پتری مربوطه می‌توانیم مقادیر کمی خصیصه‌های مذکور را محاسبه کنیم. شکل ۱۰ مراحل روش پیشنهادی را نشان می‌دهد.

٤-١ روشی برای کمی‌سازی خصیصه‌های کیفی

بخش الف از چارچوب پیشنهادی در بخش ۳-۳-۱ که شامل آشنایی با مفاهیم بود شرح داده شد، با این تفاوت که در چارچوب پیشنهادی به منظور ارزیابی دقیق‌تر کارایی باید فضای مصرفی معماری نیز بررسی گردد و بنابراین حجم فضایی که برای ذخیره داده استفاده می‌شود را مورد بررسی قرار داده‌ایم. از آنجایی در شبکه پتری رنگی برای مدل‌کردن حافظه و فایلی که داده در آنها ذخیره می‌شود از مکان‌های مختلف استفاده می‌شود، باید تعداد و حجم مکان‌های استفاده‌شده برای ذخیره داده مورد بررسی قرار گیرند.

٤-٢ مدل‌سازی سبک‌ها با شبکه‌های پتری رنگی

در این قسمت به توضیح بخش ب از چارچوب پیشنهادی می‌پردازیم. اجزای اصلی معماری به سه دسته مؤلفه، اتصال و واسط تقسیم می‌شود [۱۷]. برای ارزیابی خصیصه‌های کیفی در سبک‌ها، ابتدا باید نرم‌افزار مورد نظر را که با سبک‌های مختلف پیاده‌سازی شده است با استفاده از شبکه پتری رنگی مدل کرد. به این منظور به ازای هر مؤلفه موجود در معماری یک گذار در نظر می‌گیریم و ارتباط و اتصال میان مؤلفه‌ها را به وسیله یال‌ها و واسط‌ها را با مکان مشخص می‌کنیم. خصیصه‌های کارایی، قابلیت اطمینان و امنیت را به ترتیب با متغیرهای r، p و s نشان می‌دهیم. سپس به منظور ارزیابی این خصیصه‌های کیفی یک نشانه در نظر می‌گیریم و مقادیر (r,s,p) را به آن نسبت می‌دهیم. با حرکت نشانه در طول شبکه با توجه به عباراتی که بر روی یال‌ها وجود دارد، این مقادیر تغییر می‌کنند به نحوی که در پایان اجرا، مقدار نهایی کارایی، قابلیت اطمینان و امنیت شبکه در متغیرها نمایش داده می‌شوند.

به منظور نگاشت سبک داده مشترک به شبکه پتری رنگی متناظر، گذارها، مکان‌ها و یال‌ها را به صورت زیر نگاشت می‌کنیم:

- **گذار:** سبک داده مشترک از دو نوع مؤلفه اصلی تشکیل شده است: ۱) مؤلفه مخزن و ۲) مؤلفه‌های دستیابی به داده. معمولاً علاوه بر این دو مؤلفه در این سبک، یک مؤلفه کنترل‌کننده مرکزی[1] نیز وجود دارد که مسئول ایجاد هماهنگی و کنترل نوبت دسترسی به مؤلفه مخزن است. مؤلفه مخزن دارای قابلیت خواندن و نوشتن داده می‌باشد. برای نشان‌دادن این قابلیت در شبکه پتری لازم است در گذار مخزن از زیرگذارهای خواندن و نوشتن استفاده شود. همان طور که اشاره شد به ازای هر یک از مؤلفه‌ها یک گذار در نظر گرفته می‌شود. بنابراین گذارهای تشکیل‌دهنده شبکه پتری متناظر با سبک داده مشترک به صورت گذار مخزن، گذار خواندن، گذار دستیابی به داده و گذار کنترل مرکزی است.
- **مکان:** در سبک داده مشترک، یک مکان برای ذخیره‌سازی داده در مخزن در نظر می‌گیریم. به طور کلی مؤلفه‌هایی که اجرای آنها وابسته به تاریخچه می‌باشد و ممکن است توسط چندین مؤلفه دیگر فراخوانی شوند باید اطلاعات مربوط به مؤلفه‌های فراخواننده را نگهداری کنند تا بتوانند کنترل را به مؤلفه مرتبط بازگردانند. بنابراین باید به ازای هر یک از این مؤلفه‌های فراخوانی‌شده، یک مکان در نظر گرفته شود که اطلاعات مربوط به مؤلفه فراخواننده را نگهداری کند. با توجه به موارد ذکرشده، مکان‌هایی را در شبکه پتری متناظر با سبک داده مشترک به صورت ۱) مکان ذخیره‌سازی داده در مخزن، ۲) مکانی برای مؤلفه‌هایی که توسط چندین مؤلفه

1. Master Control

دیگر فراخوانی می‌شوند، مانند گذارهای خواندن و نوشتن و (۳) مکان برای واسط میان مؤلفه‌ها در نظر می‌گیریم.

- **یال:** برای اتصال‌های این سبک نیز یال‌هایی بین گذارها در نظر می‌گیریم. برای مثال، یک شبکه پتری برای یک سیستم نمونه با سبک داده مشترک دارای یک مخزن مشترک، یک مؤلفه برای خواندن از مخزن و یک مؤلفه برای نوشتن در مخزن است. برای وضوح بیشتر و کمبود فضا شکل این شبکه را در پیوندی که برای اشکال پتری این مقاله در نظر گرفته شده است، قرار داده‌ایم (شکل ۱ در پیوند زیر):

http://ce.kashanu.ac.ir/babamir/petri-nets

به منظور نگاشت سبک شیءگرا به شبکه پتری رنگی متناظر، گذارها، مکان‌ها و یال‌ها را به صورت زیر نگاشت می‌کنیم:

- **گذار:** مؤلفه‌های سبک شیءگرا شامل اشیایی هستند که به وسیله فراخوانی توابع با یکدیگر در تعامل می‌باشند و بنابراین برای هر یک از اشیا یک گذار در نظر می‌گیریم. معمولاً در این سبک، یک مؤلفه کنترل‌کننده مرکزی نیز وجود دارد که مسئول ایجاد هماهنگی میان اشیا می‌باشد که برای این مؤلفه هم یک گذار در نظر می‌گیریم. هر یک از اشیا دارای توابعی هستند که با داده‌های درون اشیا در ارتباط هستند و عملیاتی را بر روی آنها انجام می‌دهند. برای نشان این توابع در شبکه پتری لازم است در گذاری که برای یک شیء در نظر می‌گیریم از زیرگذارهایی برای توابع نیز استفاده کنیم. بنابراین گذارهای تشکیل‌دهنده شبکه پتری متناظر با سبک شیءگرا را به صورت ۱) یک گذار برای هر شیء، ۲) یک گذار برای هر متد از هر شیء و ۳) گذار کنترل مرکزی در نظر می‌گیریم.
- **مکان:** در سبک شیءگرا برای هر متغیر ازهر شیء یک مکان در نظر می‌گیریم. به طور کلی مؤلفه‌هایی که اجرای آنها وابسته به تاریخچه می‌باشند و ممکن است توسط چندین مؤلفه دیگر فراخوانی شوند باید اطلاعات مربوط به مؤلفه‌های فراخواننده را نگهداری کنند تا بتوانند کنترل را به مؤلفه مرتبط بازگردانند. بنابراین باید برای هر یک از این مؤلفه‌های فراخوانی‌شده، یک مکان در نظر بگیریم تا اطلاعات مربوط به مؤلفه فراخواننده را نگهداری کند. همچنین به دلیل این که مؤلفه‌هایی که در طول یک بار اجرا مؤلفه‌های مختلفی را فراخوانی می‌کنند، رفتار وابسته به تاریخچه دارند باید برای هر یک از این مؤلفه‌ها نیز مکانی در نظر بگیریم تا کنترل اجرا با توجه به تاریخچه به صورت صحیح منتقل شود. با توجه به موارد ذکرشده، مکان‌هایی را که در شبکه پتری متناظر با سبک داده مشترک در نظر می‌گیریم به صورت ۱) مکان ذخیره‌سازی داده در شیء، ۲) مکان‌هایی برای مؤلفه‌هایی که توسط مؤلفه/ مؤلفه‌های دیگر، فراخوانی می‌شوند مانند توابع داخل اشیا، (۳) مکان‌هایی برای مؤلفه‌هایی که در طول یک بار اجرا، مؤلفه‌های مختلفی را فراخوانی می‌کنند مانند مؤلفه‌هایی که توابع مختلف یک شیء را فراخوانی می‌کنند و (۴) مکانی برای واسط میان مؤلفه‌ها هستند.
- **یال:** برای هر اتصال در این سبک یک یال بین دو گذار در نظر گرفته می‌شود. برای مثال، شبکه پتری برای یک سیستم با سبک شیءگرا دارای دو مؤلفه است و هر یک از مؤلفه‌ها دارای یک تابع داخلی برای دسترسی به متغیر محلی خود هستند. برای وضوح بیشتر و کمبود فضا، شکل این شبکه (شکل ۲) را در پیوندی که برای اشکال پتری این مقاله در نظر گرفته‌ایم (بخش ۴-۲) قرار داده‌ایم.

به منظور نگاشت سبک لوله و صافی به شبکه پتری رنگی متناظر، گذارها، مکان‌ها و یال‌ها را به صورت زیر نگاشت می‌شوند:

- **گذار:** برای هر یک از مؤلفه‌های اصلی که صافی نام دارند، یک گذار در نظر می‌گیریم. این سبک فاقد مؤلفه کنترل‌کننده مرکزی است و بنابراین گذاری برای این مؤلفه در نظر گرفته نمی‌شود.
- **مکان:** در سبک لوله و صافی، هر مؤلفه برای انجام عملیات، داده را در بافر خود ذخیره می‌کند. به همین دلیل برای هر صافی، یک مکان در نظر می‌گیریم.
- **یال:** برای هر اتصال این سبک یک یال بین دو گذار در نظر می‌گیریم.

برای مثال یک شبکه پتری برای یک سیستم با سبک لوله و صافی، دارای دو مؤلفه است. برای وضوح بیشتر و کمبود فضا، شکل این شبکه (شکل ۳) را در پیوندی که برای اشکال پتری این مقاله در نظر گرفته‌ایم (بخش ۴-۲)، قرار داده‌ایم.

٤-۳ ادغام مدل‌ها با کمی‌سازی خصیصه‌های کاندید

در این قسمت به توضیح بخش ج از چارچوب پیشنهادی می‌پردازیم. پس از این که معماری نرم‌افزار با سبک‌های معماری پیاده‌سازی شد، بر پایه روش‌هایی که در بخش ۴-۲ شرح دادیم، روش‌های بخش ۴-۱ را برای کمی‌سازی خصیصه‌های کیفی کاندید در سبک‌های کاندید به کار می‌گیریم تا به صورت کمی میزان تأثیر این خصیصه‌های کیفی را در معماری نرم‌افزار ارزیابی کنیم. به عبارت دیگر این بخش، ادغامی از قسمت‌های الف و ب چارچوب پیشنهادی است. در ادامه به ارزیابی کمی خصیصه‌های کیفی کاندید در سبک‌های کاندید می‌پردازیم.

٤-۳-۱ کارایی در سبک‌های کاندید

از آنجا که بر طبق استاندارد ISO ۹۱۲۶، خصیصه کارایی دارای زیرخصیصه‌های زمان و فضای مصرفی است، این خصیصه را با توجه این دو زیرخصیصه بررسی می‌کنیم [۱۸]. زمان مصرفی، زمانی پاسخ نرم‌افزار و فضای مصرفی، فضایی است که نرم‌افزار برای ذخیره و استفاده کارآمد از منابع لازم دارد.

در سبک داده مشترک، داده‌ها به صورت کارآمد استفاده می‌شوند زیرا یک مخزن داده واحد بین مؤلفه‌ها به اشتراک گذاشته می‌شود و دسترسی مؤلفه‌ها به داده‌های مشترک به صورت مستقیم و بدون نیاز به کپی‌کردن داده‌ها است. در نتیجه در مؤلفه‌ها فضای قابل توجهی برای استفاده از داده‌ها مصرف نمی‌شود.

سبک شیءگرا از لحاظ فضای مصرفی برای ذخیره داده‌ها نسبتاً به صورت کارامد عمل می‌کند ولی داده‌ها به صورت مستقیم بین مؤلفه‌ها به اشتراک گذاشته نمی‌شوند. در عوض هر مؤلفه دارای واسط‌هایی است که مؤلفه‌های دیگر تنها با فراخوانی این واسط‌ها می‌توانند به داده‌ها دسترسی پیدا کنند. بنابراین این سبک برای دسترسی به داده‌ها نیاز به فراخوانی توابع بیشتری نسبت به سبک داده مشترک دارد و این امر سبب می‌شود زمان بیشتری مصرف شود. اما چون در برخی از بخش‌ها، سبک داده مشترک تعداد مؤلفه‌های بیشتری نسبت به سبک شیءگرا را درگیر می‌کند و در نتیجه هر مؤلفه مسئول انجام وظایف کمتری است، زمان مصرفی در این مؤلفه‌ها کمتر است.

سبک لوله و صافی از لحاظ فضای مصرفی برای ذخیره داده‌ها به صورت ناکارامد عمل می‌کند زیرا هر صافی، همه داده‌ها را در درگاه خروجی‌اش کپی می‌کند و سپس داده‌ها به مؤلفه بعدی منتقل می‌شوند.

در نتیجه باید در هر مؤلفه برای ذخیره داده‌ها به اندازه سبک داده مشترک فضا در نظر گرفته شود. با توجه به زمان لازم برای کپی و انتقال داده‌ها به مؤلفه‌های بعدی، این سبک از لحاظ زمان مصرفی نسبت به سبک‌های داده مشترک و شیءگرا کارایی کمتری دارد. به طور کلی مؤلفه‌هایی که نیاز به انجام عملیات و محاسبات بیشتری دارند، نسبت به مؤلفه‌هایی که محاسبات کمتری انجام می‌دهند، زمان بیشتری مصرف می‌کنند.

٤-۳-۲ قابلیت اطمینان در سبک‌های کاندید

بر طبق استاندارد ISO ۹۱۲۶ خصیصه قابلیت اطمینان دارای زیرخصیصه‌های بلوغ و قابلیت ترمیم است. بنابراین باید قابلیت اطمینان را از نظر این دو زیرخصیصه بررسی کنیم. بلوغ قابلیتی از نرم‌افزار است که بتواند صحیح کار کند و دچار شکست نشود. قابلیت ترمیم به معنای ترمیم داده‌هایی است که به صورت مستقیم تحت تأثیر شکست قرار گرفته و آسیب دیده‌اند [۱۸]. بنابراین یکی از اهداف مدل‌سازی قابلیت اطمینان نرم‌افزار، ارزیابی تعداد رخداد خطا در برنامه است و یکی از عواملی که بر روی قابلیت اطمینان مؤلفه‌ها تأثیر می‌گذارد، میزان پیچیدگی پیاده‌سازی آنها است. به طور کلی مؤلفه‌های دارای پیچیدگی بیشتر، درک و فهم مشکل‌تری نسبت به مؤلفه‌های دارای پیچیدگی کمتر دارند. همچنین احتمال بروز خطا و نقص در آنها بیشتر است. یکی از معیارهای مهم پیچیدگی، تعداد خطوط برنامه است. هرچه تعداد خطوط یک مؤلفه بیشتر باشد، پیچیدگی آن مؤلفه بیشتر است [۲۸]. در نتیجه مؤلفه‌هایی که محاسبات بیشتری انجام می‌دهند و تعداد خطوط بیشتری دارند، نسبت به مؤلفه‌هایی که عملیات کمتری انجام می‌دهند و تعداد خطوط کمتری دارند، قابلیت اطمینان کمتری دارند.

در سبک داده مشترک به دلیل این که مؤلفه‌ها به صورت مستقل از یکدیگر عملیات پردازش را انجام می‌دهند، اگر مؤلفه‌ای دچار شکست شود از انتشار خطا در مؤلفه‌های دیگر جلوگیری می‌شود. بنابراین تحمل‌پذیری خطا در این سبک نسبت به دو سبک دیگر بیشتر است [۲۹]. همچنین در این سبک، هر مؤلفه مسئول انجام عملیات بیشتری نسبت به سبک شیءگرا است. در نتیجه مؤلفه‌ها در این سبک درشت‌دانه هستند و تعداد خطوط آنها نیز بیشتر است. به همین دلیل پیچیدگی مؤلفه‌ها بیشتر و به این ترتیب احتمال بروز خطا در مؤلفه‌های این سبک بیشتر است. بنابراین قابلیت اطمینان مؤلفه‌های مختلف در سبک داده مشترک نسبت به قابلیت اطمینان مؤلفه‌ها در سبک شیءگرا کمتر است.

در سبک شیءگرا چون مؤلفه‌ها نسبتاً از یکدیگر مستقل هستند، تأثیر مثبتی بر قابلیت اطمینان دارد و سبب می‌شود با شکست یک مؤلفه، کل سیستم دچار شکست نشود [۲۹]. از طرف دیگر چون تعداد مؤلفه‌های بیشتری درگیر هستند و هر مؤلفه مسئول انجام وظایف کمتری است، مؤلفه‌ها ریزدانه هستند و تعداد خطوط آنها کمتر است و در نتیجه پیچیدگی مؤلفه‌ها نیز کمتر است و همین امر سبب می‌شود قابلیت اطمینان هر یک از مؤلفه‌ها در این سبک نسبت به سبک داده مشترک بیشتر باشد. ولی با توجه به نتایج به دست آمده، در کل قابلیت اطمینان سبک داده مشترک بیشتر از سبک شیءگرا است.

در سبک لوله و صافی باید همه صافی‌ها یا مؤلفه‌ها به صورت صحیح اجرا شوند تا سیستم موفق گردد و در غیر این صورت کل سیستم دچار شکست می‌شود ولی در سایر سبک‌ها اگر یک مؤلفه شکست بخورد و به صورت صحیح اجرا نشود، ممکن است مؤلفه‌های دیگر بتوانند به کار خود ادامه دهند و نتایج صحیح تولید کنند [۲۹]. در این سبک چون باید هر صافی همه داده‌ها را در درگاه خروجی‌اش کپی کند، مؤلفه‌ها در این سبک دارای ساختار پیچیده‌تری هستند و هر مؤلفه باید قابلیت انتقال داده را داشته باشد و به همین دلیل احتمال رخداد خطا در مؤلفه‌های این سبک بیشتر از سبک‌های دیگر است. از طرف دیگر به دلیل این که در این سبک بر خلاف سبک‌های داده مشترک و شیءگرا داده‌ها در یک مکان متمرکز نیستند، اگر داده‌ها دچار آسیب شوند ترمیم آنها مشکل است. به این ترتیب قابلیت اطمینان مؤلفه‌ها در این سبک کمتر از قابلیت اطمینان در سبک‌های داده مشترک و شیءگرا است.

٤-۳-۳ امنیت در سبک‌های کاندید

بر طبق استاندارد ISO ۹۱۲۶، امنیت عبارت است از توانایی نرم‌افزار در حفاظت اطلاعات و داده. به این ترتیب افراد و سیستم‌های غیر مجاز نمی‌توانند داده‌ها را بخوانند و یا اصلاح کنند و برای افراد و سیستم‌های مجاز منعی برای دسترسی وجود ندارد [۱۸]. بخشی از امنیت یک سیستم به امنیت شبکه‌ای مربوط است که سیستم مورد نظر از طریق آن به تبادل اطلاعات می‌پردازد و بخشی دیگر امنیت یک سیستم مربوط به امنیت حافظه‌ای است که داده‌ها در آن قرار دارند. به طور کلی مؤلفه‌هایی که در شبکه قرار گرفته‌اند و داده‌ها را در طول شبکه انتقال می‌دهند، نسبت به مؤلفه‌هایی که در شبکه قرار ندارند، آسیب‌پذیری بیشتر و در نتیجه امنیت کمتری دارند.

در سبک داده مشترک، چون همه داده‌ها در یک مکان منفرد به نام مخزن طبقه‌بندی و ذخیره می‌شوند، کنترل دسترسی به داده‌ها آسان است. از این رو می‌توانیم با اعمال تدابیر امنیتی بر روی مخزن مانند اعطای دسترسی به کاربران و بررسی نقص‌های امنیتی، مانع از دسترسی‌های غیر مجاز شویم. به همین دلیل داده‌ها در سبک داده مشترک دارای امنیت مناسبی هستند.

در سبک شیءگرا، داده‌ها در داخل یک کلاس قرار می‌گیرند و این کلاس دارای واسطهایی است که مؤلفه‌های دیگر تنها با فراخوانی این واسطها می‌توانند به داده‌ها دسترسی پیدا کنند. بنابراین به دلیل کپسوله‌سازی داده‌ها در این سبک، داده‌ها از امنیت زیادی برخوردار هستند و از دسترسی عناصر غیر مجاز به داده‌ها جلوگیری می‌شود. در نتیجه، امنیت مؤلفه‌ها در این سبک نسبت به سبک داده مشترک بیشتر است.

در سبک لوله و صافی داده‌ها در یک مکان متمرکز نیستند و امکاناتی برای ایجاد امنیت وجود ندارد. به همین سبب داده‌ها در این سبک بیشتر در معرض دسترسی عناصر غیر مجاز هستند و امنیت این سبک نسبت به سبک‌های داده مشترک و شیءگرا کمتر است.

٤-٤ سناریوهای کاربردی

در این قسمت به توضیح بخش د از چارچوب پیشنهادی می‌پردازیم. به این منظور ابتدا چندین سناریوی کاربردی را مطرح می‌کنیم. برای محاسبه مقدار کمی نهایی خصیصه‌های کاندید در سبک‌های کاندید به صورت زیر عمل می‌کنیم:

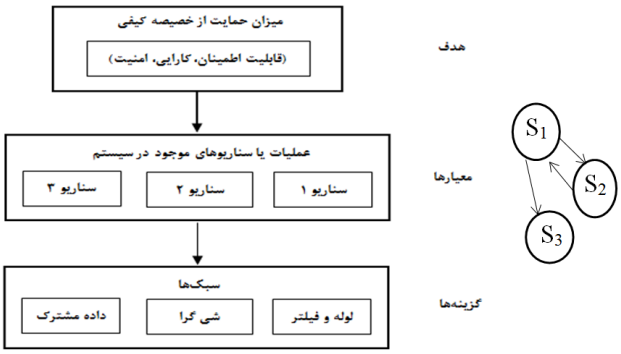
۱) مسیر اجرای سناریوها را در شبکه پتری متناظر با هر یک از سبک‌های کاندید دنبال می‌کنیم.

۲) با توجه به بخش ۳-۳-۱، مؤلفه‌های مؤثر در سناریو را در نظر می‌گیریم.

۲-۱) تعداد دفعات اجرای هر مؤلفه را به دست می‌آوریم.

۲-۲) مدت زمان اجرای هر مؤلفه و قابلیت اطمینان و آسیب‌پذیری آنها را از سابقه اجرای مؤلفه‌ها به دست می‌آوریم.

لازم به ذکر است در سبک‌هایی که به صورت فراخوانی- بازگشت هستند، ممکن است در حین اجرای یک مؤلفه، قبل از انتقال کامل کنترل



شکل ۱۱: سبک فراخوانی- بازگشت و ساختار سلسله‌مراتبی انتخاب سبک.

اجرا به مؤلفه دیگر، سرویس‌هایی درخواست شود که توسط مؤلفه‌های دیگر فراهم شده است. در این صورت پس از انجام درخواست توسط مؤلفه فراخوانی‌شده، کنترل اجرا به مؤلفه فراخواننده باز می‌گردد و از جایی که آن مؤلفه ترک شده بود، دستورات اجرا می‌شود. برای مثال، فراخوانی $S_۲$ به وسیله $S_۱$ در طرف راست شکل ۱۱ را در نظر بگیرید. مؤلفه $S_۱$ قبل از واردشدن به $S_۲$ (بدون در نظر گرفتن این که چند بار $S_۲$ اجرا شود)، فقط یک بار ملاقات می‌شود. بنابراین برای محاسبه قابلیت اطمینان شبکه باید میزان قابلیت اطمینان $S_۱$ فقط یک بار در نظر گرفته شود. جزئیات عملی در یک مطالعه موردی در بخش ۵ بیان می‌شود.

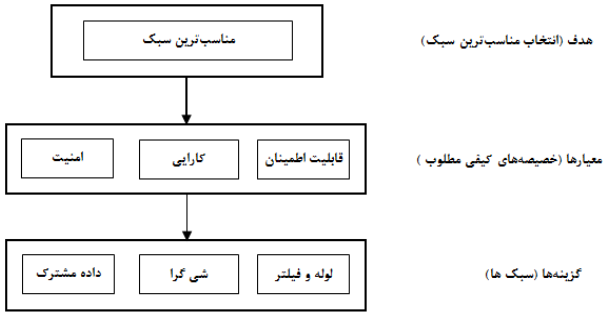
در این قسمت به توضیح بخش ه از چارچوب پیشنهادی می‌پردازیم. در این بخش ابتدا سناریوهای مطرح‌شده در بخش قبل را به ترتیب اهمیت اولویت‌بندی می‌کنیم. سپس با توجه به نتایج کمی به دست آمده از ارزیابی خصیصه‌های کاندید در هر سناریو (که در بخش قبل محاسبه شده‌اند)، از روش تحلیل سلسله‌مراتبی که در بخش مقدمات بیان شد، استفاده می‌کنیم. به این ترتیب سبک‌های کاندید را از نظر میزان حمایت از هر یک از خصیصه‌های کاندید رتبه‌بندی می‌کنیم. در طرف چپ شکل ۱۱، انتزاعی از ساختار سلسله‌مراتبی را برای تعیین میزان حمایت سبک‌های کاندید از خصیصه‌های کیفی کاندید نشان داده‌ایم. جزئیات عملی در یک مطالعه موردی در بخش ۵ بیان می‌شود.

٤-۵ اولویت خصیصه‌های کیفی و رتبه‌بندی سبک‌ها

در این قسمت به توضیح بخش ی از چارچوب پیشنهادی می‌پردازیم. ابتدا با روش سلسله‌مراتبی خصیصه‌های کیفی کاندید سیستم را اولویت‌بندی می‌کنیم و سپس از نسبت آنها به یکدیگر استفاده می‌کنیم. پس از اولویت‌بندی خصیصه‌های کاندید از روش تحلیل سلسله‌مراتبی استفاده می‌کنیم و در آن با توجه به رتبه سبک‌های کاندید از نظر میزان حمایت از هر یک از خصیصه‌های کیفی کاندید، رتبه نهایی سبک‌های کاندید را مشخص می‌کنیم. نمایش انتزاعی ساختار سلسله‌مراتبی مورد استفاده برای تعیین رتبه کلی و نهایی سبک‌های کاندید را از نظر خصیصه‌های کیفی کاندید در شکل ۱۲ نشان داده‌ایم. جزئیات عملی در یک مطالعه موردی در بخش ۵ بیان می‌شود.

۵- مطالعه موردی، دستگاه خودپرداز

در این بخش سیستم نرم‌افزاری دستگاه خودپرداز بانک را به عنوان مطالعه موردی انتخاب می‌کنیم. می‌خواهیم هر یک از سبک‌های کاندید را برای این سیستم ارزیابی کمی کنیم و در نهایت رتبه کلی آنها را برای حمایت از این سیستم به دست آوریم. به منظور تعیین سبک مناسب برای طراحی معماری این سیستم از نظر خصیصه‌های کیفی کاندید، ابتدا



شکل ۱۲: نمایش انتزاعی ساختار سلسله‌مراتبی برای تعیین رتبه سبک‌ها.

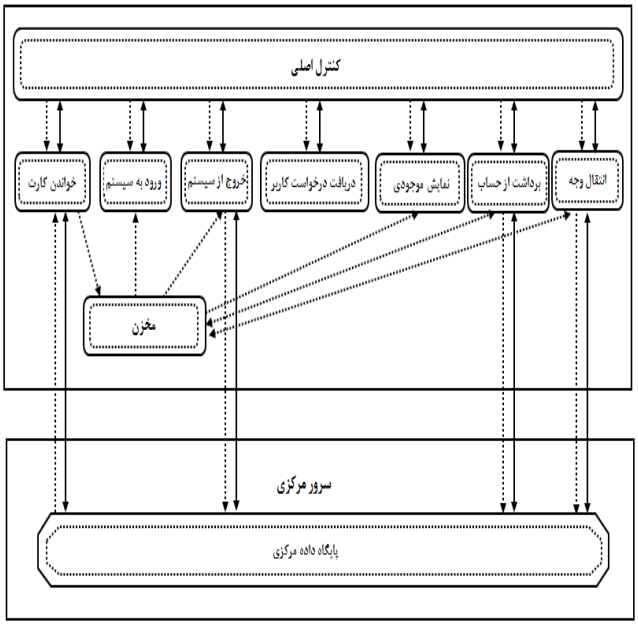
معماری سیستم خودپرداز را با هر یک از این سبک‌های کاندید توصیف می‌کنیم. سپس روش پیشنهادی را به کار می‌بریم که در آن ۱) توصیف هر سبک کاندید را به شبکه پتری متناظر نگاشت می‌کنیم، ۲) با استفاده از نرم‌افزار CPNTools، شبکه‌های پتری را اجرا می‌کنیم تا خصیصه‌های کیفی کاندید را به صورت کمی ارزیابی کنیم و ۳) با توجه به نتایج به دست آمده از این ارزیابی کمی، از روش فرآیند سلسله‌مراتبی برای رتبه‌بندی سبک‌های کاندید استفاده می‌کنیم تا بتوانیم مناسب‌ترین سبک کاندید را برای طراحی معماری سیستم خودپرداز مشخص کنیم.

دستگاه خودپرداز در یک زمان، فقط به یک مشتری سرویس می‌دهد. از مشتری درخواست می‌شود که کارت اعتباری خود را در دستگاه وارد کند و رمز عبور را وارد نماید. شماره کارت و رمز عبور اعتبارسنجی می‌شود، سپس مشتری می‌تواند یک یا تعداد بیشتری تراکنش انجام دهد و تا پایان انجام آخرین تراکنش، کارت باید در دستگاه باقی بماند. دستگاه خودپرداز قادر است خدمات ۱) مانده حساب، ۲) برداشت، ۳) انتقال وجه به حساب دیگر و ۴) خروج از سیستم را به مشتری ارائه نماید.

دستگاه خودپرداز برای انجام هر تراکنش با سرور مرکزی ارتباط برقرار می‌کند. اگر مشخص شود که رمز عبور مشتری نامعتبر است از کاربر درخواست می‌شود که رمز عبور را مجدداً وارد کند. اگر مشتری بیش از سه بار رمز عبور را نادرست وارد کند کارت در دستگاه باقی می‌ماند.

۵-۱ توصیف معماری نرم‌افزار دستگاه خودپرداز

در این بخش به توصیف دستگاه خودپرداز با سبک‌های داده مشترک، شیءگرا و لوله و صافی می‌پردازیم. در این توصیف، عناصر معماری (مانند مؤلفه‌ها) به صورت گره و ارتباطات (مانند اتصالات) میان عناصر معماری به صورت پیکان میان گره‌ها نمایش داده می‌شوند [۳۰]. برای مثال، یک مؤلفه نرم‌افزاری یک رویه فعال است و فقط زمانی اجرا می‌شود که فراخوانی شود. در توصیف بصری، داده (داده با خطوط بریده) و کنترل (با خطوط ممتد) از هم متمایز می‌شود. یک مؤلفه کنترلی، یک مؤلفه قابل اجرا است و یک مؤلفه داده به معنی قابلیت ذخیره‌سازی است. ارتباطات یا اتصالات میان عناصر معماری با پیکان میان گره‌ها مشخص می‌شود که پیکان‌های توپر، بریده و ترکیب توپر و بریده به ترتیب معرف جریان کنترل، جریان داده و جریان توأم کنترل و داده است. اگر یک اتصال بتواند کنترل را از یک مؤلفه به مؤلفه دیگر انتقال دهد و سبب اجرای آن شود، آن اتصال خاصیت کنترلی دارد. اگر اتصالی یک داده را از یک مؤلفه به مؤلفه دیگر منتقل کند، آن اتصال خاصیت داده‌ای دارد. اتصالات داده با خطوط نقطه‌چین و اتصالات کنترلی با خطوط ممتد نمایش داده می‌شوند. در جدول ۱ رخدادها (در سبک داده مشترک)، متدها (در سبک شیءگرا) و عملیات (در سبک لوله و صافی) را بیان کرده‌ایم. این جدول را در اولویت‌بندی سبک‌ها در سناریوهای دستگاه خودپرداز استفاده می‌کنیم.



شکل ۱۳: توصیف معماری دستگاه خودپرداز با سبک داده مشترک.

شکل ۱۳ توصیف دستگاه خودپرداز را با سبک داده مشترک نشان می‌دهد. بر اساس این شکل و قواعد بخش ۴-۲ شبکه پتری رنگی سبک داده مشترک را ساختیم. برای مدل‌سازی و ارزیابی شبکه پتری رنگی از نرم‌افزار CPNTools استفاده کردیم و مقادیر اولیه خصیصه‌های کارایی، قابلیت اطمینان و امنیت (متغیرهای r، p و s، بخش ۴-۲) را با نشانه‌ای در اولین مکان شبکه به نام Initial Value مشخص کردیم. برای وضوح بیشتر و کمبود فضا در این مقاله، این شبکه همراه با شرح گذارها و مکان‌ها را در شکل ۴ در پیوندی که برای اشکال پتری این مقاله در نظر گرفته‌ایم (بخش ۴-۲)، قرار دادیم. شکل‌های ۵ و ۶ در این پیوند، زیرشبکه مربوط به گذارهای مخزن و پایگاه داده مرکزی را نشان می‌دهد.

همان طور که در بخش ۴-۲ اشاره شد برای مدل‌سازی سبک داده-مشترک نیاز به گذار مخزن، گذار دسترسی به داده و گذار کنترل مرکزی داریم. در مدل‌سازی که انجام دادیم، گذار Repository را برای گذار مخزن در نظر گرفتیم. برای هر گذار مخزن، دو زیرگذار خواندن و نوشتن را در نظر گرفتیم. یک گذار به نام Master Control برای کنترل مرکزی مخزن در نظر گرفتیم. در این مدل‌سازی، رخدادهای گذارهای اصلی را که برای دسترسی به داده‌ها تعریف کردیم بر اساس جدول ۱ به صورت دنباله زیر است

$$a_{۶} - a_{۵} - a_{۲} - a_{۱۱} - a_{۸} - a_{۹} - a_{۱۰} \qquad (۳)$$

طبق بخش ۴-۲ برای مدل‌سازی سبک داده مشترک سه نوع مکان لازم داریم: ۱) مکانی به نام store برای ذخیره‌سازی داده در مخزن، ۲) مکانی به نام Turn برای کنترل فراخوانی مؤلفه کنترل مرکزی و ۳) مکان‌های Read-Turn و Write-Turn به ترتیب برای کنترل فراخوانی گذارهای خواندن و نوشتن. دیگر مکان‌های استفاده‌شده در این مدل را برای واسط بین گذارها در نظر گرفتیم.

در سبک شیءگرا مؤلفه‌ها نمی‌توانند به صورت مستقیم به رکورد حاوی اطلاعات حساب دسترسی داشته باشند و رکورد حساب در یک کلاس قرار می‌گیرد که این کلاس دارای توابعی (متدهای بارگذاری، دریافت پین‌کد، مشخص‌کردن نشان خورده‌شدن کارت، دریافت موجودی، نوشتن موجودی و ذخیره) است که امکان کار با رکورد حساب را فراهم

جدول ۱: رخدادها/ عملیات/ متدها در گذارهای سبک‌های کاندید.

$a_{۱}$	کنترل مرکزی
$a_{۲}$	خواندن کارت به وسیله دستگاه
$a_{۳}$	اتصال به شبکه
$a_{۴}$	انتقال داده بین مخزن و شبکه
$a_{۵}$	ورود به سیستم
$a_{۶}$	درخواست و دریافت اطلاعات از کاربر
$a_{۷}$	بازگشت به دستگاه
$a_{۸}$	نمایش موجودی
$a_{۹}$	دریافت وجه
$a_{۱۰}$	انتقال وجه به حساب دیگر
$a_{۱۱}$	خروج از سیستم
$a_{۱۲}$	خواندن اطلاعات از مخزن
$a_{۱۳}$	نوشتن اطلاعات در مخزن
$a_{۱۴}$	تعامل با پایگاه داده
$a_{۱۵}$	بارگیری اطلاعات حساب از مخزن
$a_{۱۶}$	دریافت پین‌کد
$a_{۱۷}$	تشخیص نشان خوردگی کارت
$a_{۱۸}$	دریافت موجودی
$a_{۱۹}$	اصلاح موجودی
$a_{۲۰}$	ذخیره اطلاعات رکورد

می‌کنند. برای وضوح بیشتر و کمبود فضا در این مقاله، توصیف معماری دستگاه خودپرداز با سبک شیءگرا را در شکل ۷ در پیوندی که برای اشکال این مقاله در نظر گرفته‌ایم (بخش ۴-۲) قرار دادیم. بر اساس این شکل و قواعد بخش ۴-۲، شبکه پتری رنگی سبک شیءگرا برای خودپرداز را ساختیم. برای وضوح بیشتر و کمبود فضا، شکل این شبکه همراه با شرح گذارها و مکان‌ها را در شکل ۸ در پیوندی که برای اشکال پتری این مقاله در نظر گرفته‌ایم (بخش ۴-۲) قرار دادیم. شکل ۹ در این پیوند جزئیات گذار پایگاه داده مرکزی را نشان می‌دهد. همان طور که در بخش ۴-۲اشاره شد، در مدل‌سازی سبک شیءگرا باید برای هر یک از اشیا یک گذار جداگانه و همچنین باید برای هر یک از متدهای اشیا نیز یک گذار جداگانه در نظر بگیریم. در این مدل‌سازی، گذار Master Control را متناظر با گذار کنترل مرکزی در نظر گرفتیم. بر اساس بخش ۴-۲ برای مدل‌سازی سبک شیءگرا چهار نوع مکان لازم است. بر این اساس، ۱) مکان Account Record برای متغیری در اشیا که اطلاعات حساب کاربری را نگهداری می‌کند، ۲) مکان Turn برای کنترل مؤلفه‌هایی که در طول اجرا چندین بار فراخوانی می‌شوند مانند کنترل مرکزی، ۳) مکان‌هایی برای کنترل مؤلفه‌هایی که در طول اجرا مؤلفه‌های مختلفی را فراخوانی می‌کنند مانند دریافت و انتقال وجه و ۴) مکان‌هایی برای واسطی میان گذارها در نظر می‌گیریم.

در سبک لوله و صافی مؤلفه کنترل‌کننده مرکزی وجود ندارد و مؤلفه‌ها به صورت متوالی اجرا می‌شوند. همچنین رکورد حاوی اطلاعات حساب از مؤلفه‌ای به مؤلفه دیگر منتقل می‌شود تا به این ترتیب همه مؤلفه‌ها به داده‌های مورد نیاز دسترسی داشته باشند. برای وضوح بیشتر و کمبود فضا در این مقاله، توصیف معماری دستگاه خودپرداز با لوله و صافی را در شکل ۱۰ در پیوندی که برای اشکال این مقاله در نظر گرفته‌ایم (بخش ۴-۲) قرار دادیم.

جدول ۲: زمان مصرفی گذارها/ مؤلفه‌ها در سبک داده مشترک.

گذار	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
زمان مصرفی	t_2	t_5	t_2	t_6	t_5	t_2	t_1	t_5	t_6	t_7	t_2	t_1	t_1	t_5	t_6	t_4	t_4	t_4	t_4	t_4

جدول ۳: زمان مصرفی گذارها/ مؤلفه‌ها در سبک شیءگرا.

گذار	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{19}	a_{20}
زمان مصرفی	t_2	t_5	t_2	t_6	t_2	t_2	t_1	t_2	t_2	t_3	t_2	t_1	t_1	t_5	t_6	t_4	t_4

جدول ۴: زمان مصرفی مؤلفه‌ها در سبک لوله و صافی.

گذار	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{14}
زمان	t_5	t_2	t_6	t_5	t_2	t_1	t_5	t_6	t_7	t_2	t_5
یال ورودی به گذار	a_5	a_6	a_8	a_9	a_{10}	a_{11}					
زمان مصرفی	t_6	t_6	t_6	t_6	t_6	t_6					

بر اساس این شکل و قواعد بخش ۴-۲، شبکه پتری رنگی را برای سبک لوله و صافی ساختیم. برای وضوح بیشتر و کمبود فضا، شکل این شبکه همراه با شرح گذارها و مکان‌ها را در شکل ۱۱ در پیوندی که برای اشکال پتری این مقاله در نظر گرفته‌ایم (بخش ۴-۲) قرار دادیم. شکل ۱۲ در این پیوند جزئیات گذار پایگاه داده مرکزی را نشان می‌دهد. همان طور که در بخش ۴-۲اشاره شد، برای مدل‌سازی سبک لوله و صافی برای هر یک از مؤلفه‌های اصلی که نقش صافی را دارند، یک گذار در نظر می‌گیریم. وابسته به هر صافی عملی وجود دارد که صافی مربوطه پس از عمل روی داده، آن را به صافی بعدی برای انجام عمل بعدی ارسال می‌کند. بنابراین برای هر یک از عملیات (کادرهای با خطوط پررنگ در شکل ۱۰ پیوند)، یک صافی جداگانه در نظر می‌گیریم.

۵-۲ ارزیابی کمی خصیصه‌ها

بر اساس بخش ۴-۴ برای هر یک از خدمات نمایش موجودی حساب، برداشت از حساب و انتقال وجه، یک سناریو در نظر می‌گیریم. سپس بر اساس بخش ۴-۵ اولویت این سناریوها را نسبت به یکدیگر مشخص می‌کنیم (اولویت‌های ۰٫۵، ۰٫۴ و ۰٫۱ به ترتیب برای نمایش موجودی، برداشت از حساب و انتقال وجه). با ارزیابی میزان خصیصه‌های کیفی کاندید هر یک از این سه سناریو در سه سبک کاندید و با در نظر گرفتن اولویت سناریوها، مقدار نهایی این خصیصه‌ها را در این سبک‌ها محاسبه می‌کنیم.

۵-۲-۱ تعیین زمان مصرفی (تأخیر زمانی) مؤلفه‌ها در سبک‌ها

در جدول ۱ از بخش ۵-۱، رخدادها/ متدها/ عملیاتی که در سبک‌ها انجام می‌شود را مشخص کردیم. برای آنها، هفت زمان t_1 تا t_7 را به عنوان زمان مصرفی در نظر می‌گیریم

$$t_1=۱۰۰\ ,\ t_2=۲۰۰\ ,\ t_3=۲۲۰\ ,\ t_4=۴۰۰$$
$$t_5=۵۰۰\ ,\ t_6=۱۰۰۰\ ,\ t_7=۱۱۰۰ \qquad (۴)$$

با توجه به موارد بیان‌شده در بخش ۴-۳ و اطلاعات سابقه اجرای مؤلفه‌های نرم‌افزار، مقادیر t_1 تا t_7 را برای زمان مصرفی در سبک‌های کاندید در سیستم خودپرداز محاسبه می‌کنیم. بر اساس زمان‌های t_1 تا t_7 فرض می‌کنیم تأخیر زمانی مؤلفه‌های مختلف در سبک‌های کاندید به صورت جداول ۲ (برای سبک داده مشترک)، ۳ (برای سبک شیءگرا) و ۴ (برای سبک لوله و صافی) باشد. لازم به ذکر است چون در این سبک‌ها داده بین مؤلفه‌ها انتقال می‌یابد، برای یال‌هایی که مؤلفه‌های اصلی را به یکدیگر متصل می‌کنند نیز تأخیر زمانی در نظر می‌گیریم. گذارهای a_1 تا a_{20} از جداول ۲ تا ۴ در جدول ۱ شرح داده شده‌اند.

ارزیابی کمی کارایی: همان طور که در بخش ۲ اشاره کردیم، ارزیابی مبتنی بر مدل ریاضی نیازمند داده‌هایی از سابقه اجرای مؤلفه‌های نرم‌افزاری در گذشته است. در ادامه به منظور ارزیابی خصیصه‌های کیفی کاندید، برای هر یک از مؤلفه‌ها اطلاعاتی در نظر گرفتیم که فرض می‌شود این اطلاعات با توجه به سابقه اجرای مؤلفه‌ها در گذشته به دست آمده است. چند نمونه از مسیر اجرای هر یک از سناریوها را با هر یک از سبک‌ها ارائه می‌دهیم. به منظور محاسبه کارایی در سبک‌های کاندید، سه عامل ۱) گذارهای/ مؤلفه‌های مؤثر در اجرای سناریوها، ۲) تعداد دفعات اجرای هر گذار/ مؤلفه و ۳) زمان مصرفی گذارها/ مؤلفه‌ها را در سبک‌های مختلف در نظر می‌گیریم و زمان اجرای سناریوهای نمایش موجودی حساب (S_1)، برداشت از حساب (S_2) و انتقال وجه (S_3) را در سبک‌های کاندید محاسبه می‌کنیم. بر اساس ترتیب اجرای گذارها یا مؤلفه‌های این سناریوها در سبک‌های کاندید، زمان اجرای آنها در این سبک‌ها با در نظر گرفتن مقادیر t_1 تا t_7 و جداول ۲ تا ۴ به دست می‌آید.

مسیر گذارهای مؤثر برای سناریوهای S_1، S_2 و S_3 در سبک داده مشترک بر اساس جدول ۱ به ترتیب عبارتند از

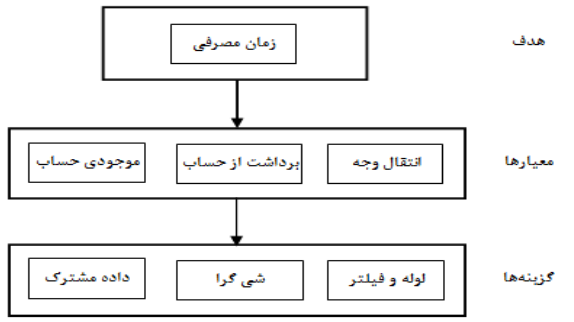
$$\begin{aligned}
S_1 &: a_1 a_2 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_8 \\
S_2 &: a_1 a_2 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_9 a_{12} a_3 a_{19} a_7 \\
S_3 &: a_1 a_2 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_{10} a_{13} a_3 a_{19} a_7
\end{aligned} \qquad (۵)$$

کارایی این سبک برای سناریوهای S_1، S_2 و S_3 به ترتیب عبارتند از

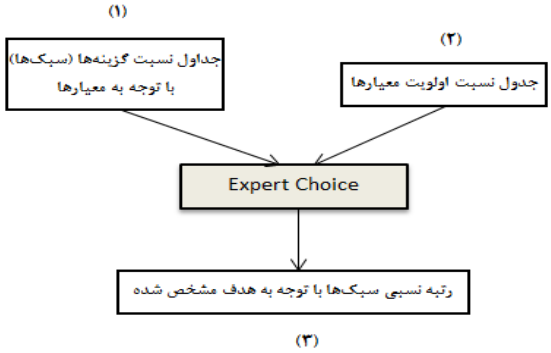
$$\begin{aligned}
S_1 &: t_2+t_5+t_2+t_5+t_6+t_2+t_1+t_2+ \\
&\quad t_1+t_5+t_2+t_2+t_2+t_1+t_5=۵۵۰۰ \\
S_2 &: t_2+t_5+t_2+t_5+t_6+t_2+t_1+t_2+t_1+t_5+ \\
&\quad t_2+t_2+t_2+t_1+t_6+t_1+t_2+t_4+t_1=۶۸۰۰ \\
S_3 &: t_2+t_5+t_2+t_5+t_6+t_2+t_1+t_2+t_1+t_5+ \\
&\quad t_2+t_2+t_2+t_1+t_7+t_1+t_2+t_4+t_1=۶۹۰۰
\end{aligned} \qquad (۶)$$

به روش فوق مسیر گذارهای مؤثر و کارایی (زمان مصرفی یا زمان اجرا) را برای سناریوهای S_1، S_2 و S_3 در سبک‌های شیءگرا و لوله و صافی محاسبه کردیم و نسبت زمان اجرای سناریوها را به دست می‌آوریم. برای نمونه جدول ۵ نسبت زمان اجرای سناریو انتقال وجه در سبک‌های کاندید را نشان می‌دهد.

با توجه به زمان اجرای هر سناریو و همچنین اولویت سناریوها



شکل ۱۴: ساختار سلسله‌مراتبی برای رتبه‌بندی سبک‌ها از نظر زمان اجرا.



شکل ۱۵: الگوی اجرایی محاسبه رتبه سبک‌های کاندید.

(بخش ۵-۲) و با استفاده از روش فرایند تحلیل سلسله‌مراتبی سبک‌های کاندید را از نظر زمان اجرا رتبه‌بندی می‌کنیم. بر اساس مدل تصمیم سلسله‌مراتبی (شکل ۱۰ از بخش ۳-۴)، ساختار سلسله‌مراتبی سیستم خودپرداز را در شکل ۱۴ مشخص کرده‌ایم. همان طور که در بخش ۳-۴ اشاره شد، محاسبات لازم جهت تحلیل ساختار سلسله‌مراتبی را با نرم‌افزار ExpertChoice انجام داده‌ایم (شکل ۱۵).

در استفاده از الگوی شکل ۱۵ برای محاسبه رتبه سبک‌های کاندید از نظر زمان اجرا، جداول ورودی برای مورد ۱ را جداول نسبت گزینه‌ها یعنی سبک‌ها (که نمونه آن در جدول ۵ آمده است) و جدول ورودی برای مورد ۲ را جدول نسبت اولویت معیارها (یعنی اولویت سناریوها) بر اساس مقادیری که در بخش ۵-۲ آمده است، در نظر می‌گیریم (جدول ۶). بر این اساس در خروجی، رتبه نسبی سبک‌ها را از نظر زمان اجرا به دست می‌آوریم (ستون ۲ از جدول ۷). در ستون‌های ۳ تا ۵ از جدول ۷، نسبت زمان‌های اجرا در سبک‌های کاندید را نیز بر اساس ستون ۱ مشخص کرده‌ایم.

برای محاسبه کارایی نهایی سبک‌های کاندید، علاوه بر زمان مصرفی، بایستی میزان فضای مصرفی برای ذخیره داده‌ها را در هر سبک کاندید نیز مشخص کنیم. بر اساس بخش ۲-۴، فضای مصرفی در سبک‌های داده مشترک یا شیءگرا برابر با تعداد مکان‌های ذخیره‌سازی داده در مخزن یا تعداد مکان‌های ذخیره‌سازی اشیا در سیستم خودپرداز است که برابر با یک است. بر اساس بخش ۲-۴ فضای مصرفی در سبک لوله و صافی برابر با تعداد مکان‌های ذخیره‌سازی داده به ازای هر یک از صافی‌ها برای سیستم خودپرداز است که برابر با ۷ است. در ادامه جداول، فضاهای مصرفی در سبک‌های کاندید و نسبت این فضاها را نشان داده‌ایم.

برای تعیین رتبه نهایی کارایی سبک‌های کاندید با توجه به زمان و فضای مصرفی، ابتدا فرایند سلسله‌مراتبی شکل ۱۵ را تشکیل می‌دهیم (این شکل را بر اساس مدل سلسله‌مراتبی شکل ۱۰ در بخش ۳-۴ مشخص کرده‌ایم) و سپس الگوی شکل ۱۴ را برای محاسبه رتبه نهایی

جدول ۵: نسبت زمان اجرای سناریوی انتقال وجه در سبک‌ها.

لوله و صافی	شیءگرا	داده مشترک	برداشت از حساب
۶۸۰۰/۹۷۰۰	۶۸۰۰/۷۴۰۰	۱	داده مشترک
۷۴۰۰/۹۷۰۰	۱	۷۴۰۰/۶۸۰۰	شیءگرا
۱	۹۷۰۰/۷۴۰۰	۹۷۰۰/۶۸۰۰	لوله و صافی

جدول ۶: نسبت اولویت سناریوها بر اساس جدول ۲.

انتقال وجه	برداشت از حساب	موجودی حساب	اولویت سناریو
۵/۱	۵/۴	۱	موجودی حساب
۴/۱	۱	۴/۵	برداشت از حساب
۱	۱/۴	۱/۵	انتقال وجه

جدول ۷: رتبه سبک‌های کاندید از نظر زمان اجرا و نسبت آنها.

سبک	رتبه	نسبت به داده مشترک	نسبت به شیءگرا	نسبت به لوله و صافی
داده مشترک	۲۸۲	۱	۲۸۲/۲۹۷	۲۸۲/۴۲۱
شیءگرا	۲۹۷	۲۹۷/۲۸۲	۱	۲۹۷/۴۲۱
لوله و صافی	۴۲۱	۴۲۱/۲۸۲	۴۲۱/۲۹۷	۱

جدول ۸: فضای مصرفی در سبک‌های کاندید و نسبت آنها.

سبک	فضای مصرفی	نسبت به داده مشترک	نسبت به شیءگرا	نسبت به لوله و صافی
داده مشترک	۱	۱	۱	۱/۷
شیءگرا	۱	۱	۱	۱/۷
لوله و صافی	۷	۷	۷	۱

جدول ۹: نسبت معیارها با توجه به اولویت.

فضای مصرفی	زمان مصرفی	اولویت معیارها
۷/۳	۱	زمان مصرفی
۱	۳/۷	فضای مصرفی

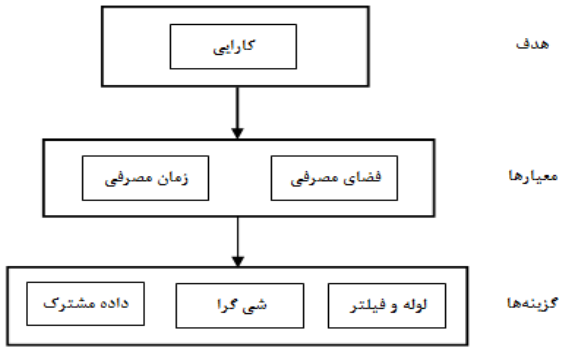
جدول ۱۰: رتبه نهایی کارایی سبک‌های کاندید.

سبک	رتبه نسبی از لحاظ کارایی
داده مشترک	۴۰۷
شیءگرا	۳۹۴
لوله و صافی	۱۹۹

کارایی سبک‌ها به کار می‌گیریم که ورودی ۱ نسبت گزینه‌ها در دو جدول ۷ و ۸ است و ورودی ۲ جدول ۹ است. پس از اجرا، نتیجه محاسبه خروجی (مورد ۳ در شکل ۱۵) را در جدول ۱۰ نشان داده‌ایم.

ارزیابی کمی قابلیت اطمینان: با توجه به موارد بیان‌شده در بخش ۴-۳ و سابقه اجرای گذارها/ مؤلفه‌ها، پارامترهای قابلیت اطمینان را برای r_1 تا r_9 با مقادیر ۹۵ درصد تا ۹۹/۵ درصد با فاصله ۰/۵ درصد فرض می‌کنیم. قابلیت اطمینان ۹۹٪ برای مثال به این معناست که از هر ۱۰۰ بار اجرا، گذار/ مؤلفه یک بار دچار شکست می‌شود. مشابه با جداول ۲ تا ۴، قابلیت اطمینان گذارها/ مؤلفه‌ها در سبک‌های کاندید را محاسبه می‌کنیم.

پس از محاسبه قابلیت اطمینان گذارها/ مؤلفه‌ها در سبک‌های کاندید، میزان قابلیت اطمینان حاصل از ارزیابی شبکه‌های پتری را برای سناریوهای نمایش موجودی حساب (S_1)، برداشت از حساب (S_2) و انتقال وجه (S_3) در هر یک از سبک‌های کاندید محاسبه می‌کنیم.



شکل ۱۶: ساختار سلسله‌مراتبی برای رتبه‌بندی سبک‌ها از نظر کارایی.

مسیر گذارهای مؤثر برای سناریوهای S_1، S_2 و S_3 در سبک داده مشترک بر اساس جدول ۱ به ترتیب عبارتند از

$$\begin{aligned} S_1 &: a_1 a_7 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_8 \\ S_2 &: a_1 a_2 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_9 a_{13} a_3 a_{19} a_7 \\ S_3 &: a_1 a_2 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_{10} a_{13} a_3 a_{19} a_7 \end{aligned} \quad (۷)$$

قابلیت اطمینان این سبک برای سناریوهای S_1، S_2 و S_3 به ترتیب عبارتند از

$$\begin{aligned} S_1 &: r_9 \times r_4 \times r_8 \times r_8 \times r_9 \times r_9 \times r_9 \times \\ &\quad r_4 \times r_9 \times r_8 \times r_9 \times r_9 \times r_6 = ۰/۸۶۸ \\ S_2 &: r_9 \times r_4 \times r_8 \times r_8 \times r_9 \times r_9 \times r_9 \times r_4 \times r_9 \times \\ &\quad r_8 \times r_9 \times r_9 \times r_5 \times r_9 \times r_8 \times r_8 \times r_8 = ۰/۸۳۰ \\ S_3 &: r_9 \times r_4 \times r_8 \times r_8 \times r_9 \times r_9 \times r_9 \times r_4 \times r_9 \times \\ &\quad r_8 \times r_9 \times r_9 \times r_4 \times r_9 \times r_8 \times r_8 \times r_8 = ۰/۸۲۵ \end{aligned} \quad (۸)$$

به روش فوق مسیر گذارهای مؤثر را برای سناریوهای S_1، S_2 و S_3 در سبک‌های شیءگرا و لوله و صافی محاسبه کردیم. سپس قابلیت اطمینان سناریوها در سبک‌های کاندید را به دست آوردیم. با توجه به میزان قابلیت اطمینان هر سناریو و اولویت سناریوها، با استفاده از روش فرایند تحلیل سلسله‌مراتبی، سبک‌های کاندید را از نظر قابلیت اطمینان رتبه‌بندی کردیم. فرایند تحلیل سلسله‌مراتبی برای قابلیت اطمینان مانند شکل ۱۴ است که به جای زمان مصرفی، قابلیت اطمینان قرار می‌گیرد. برای اجرای این فرآیند از الگوی سلسله‌مراتبی شکل ۱۵ استفاده کردیم که در آن برای ورودی ۱، نسبت قابلیت اطمینان سبک‌ها و برای ورودی ۲، اولویت سناریوها در بخش ۵-۲ را در نظر گرفتیم. خروجی حاصل از اجرا رتبه سبک‌ها از نظر قابلیت اطمینان است.

ارزیابی کمی امنیت: با توجه به مواردی که در بخش ۴-۳ بیان شد و اطلاعاتی که از سابقه اجرای مؤلفه‌های نرم‌افزار در گذشته در دسترس است، مقادیر آسیب‌پذیری گذارها/ مؤلفه‌های مرتبط با داده‌ها را در سبک‌های کاندید مشخص می‌کنیم (مشابه جداول ۲ تا ۴). مقادیر پارامترهای آسیب‌پذیری v_1، v_2، v_3 و v_4 را به ترتیب ۱، ۳، ۱۰ و ۱۵ در نظر گرفته‌ایم. سپس میزان امنیت حاصل از ارزیابی شبکه‌های پتری را برای سناریوهای نمایش موجودی حساب (S_1)، برداشت از حساب (S_2) و انتقال وجه (S_3) در هر یک از سبک‌های کاندید محاسبه می‌کنیم. برای محاسبه امنیت سبک‌های کاندید، میزان آسیب‌پذیری آنها را با توجه به میزان امنیت گذارها/ مؤلفه‌ها در برابر نفوذ عناصر خارجی، محاسبه می‌کنیم.

مسیر گذارهای مؤثر برای سناریوهای S_1، S_2 و S_3 در سبک داده مشترک بر اساس جدول ۱ به ترتیب عبارتند از

جدول ۱۱: نسبت خصیصه‌های کیفی در سبک‌های کاندید.

لوله و صافی	شیءگرا	داده مشترک	خصیصه کیفی	
۴۰۷/۱۹۹	۴۰۷/۳۹۴	۱	کارایی	
۳۴۲/۳۲۱	۳۴۲/۳۳۷	۱	قابلیت اطمینان	داده مشترک
۳۶۰/۲۲۹	۳۶۰/۴۱۱	۱	امنیت	
۳۹۴/۱۹۹	۱	۳۹۴/۴۰۷	کارایی	
۳۳۷/۳۲۱	۱	۳۳۷/۳۴۲	قابلیت اطمینان	شیءگرا
۴۱۱/۲۲۹	۱	۴۱۱/۳۶۰	امنیت	
۱	۱۹۹/۳۹۴	۱۹۹/۴۰۷	کارایی	
۱	۳۲۱/۳۳۷	۳۲۱/۳۴۲	قابلیت اطمینان	لوله و صافی
۱	۲۲۹/۴۱۱	۲۲۹/۳۶۰	امنیت	

$$\begin{aligned} S_1 &: a_1 a_7 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_8 \\ S_2 &: a_1 a_2 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_9 a_{12} a_3 a_{19} a_7 \\ S_3 &: a_1 a_2 a_3 a_{14} a_4 a_{15} a_{13} a_1 a_{12} a_5 a_1 a_6 a_1 a_{12} a_{10} a_{13} a_3 a_{19} a_7 \end{aligned} \quad (۹)$$

امنیت این سبک برای سناریوهای S_1، S_2 و S_3 به ترتیب عبارتند از

$$\begin{aligned} S_1 &: ۰+۰+v_4+۰+v_4+۰+v_2+۰+ \\ &\quad v_2+۰+۰+۰+۰+v_2+۰ = ۳۹ \\ S_2 &: ۰+۰+v_4+۰+v_4+۰+v_2+۰+v_2+۰+ \\ &\quad ۰+۰+۰+v_2+۰+v_2+v_4+۰+v_4 = ۷۲ \\ S_3 &: ۰+۰+v_4+۰+v_4+۰+v_2+۰+v_2+۰+ \\ &\quad ۰+۰+۰+v_2+۰+v_2+v_4+۰+v_4 = ۷۲ \end{aligned} \quad (۱۰)$$

به روش فوق مسیر گذارهای مؤثر را برای سناریوهای S_1، S_2 و S_3 در سبک‌های شیءگرا و لوله و صافی محاسبه کردیم. سپس میزان امنیت سبک‌های کاندید در اجرای سناریوهای کاندید را به دست آوردیم. با توجه به میزان امنیت هر سناریو و اولویت سناریوها (بخش ۵-۲) با استفاده از روش فرایند تحلیل سلسله‌مراتبی، سبک‌های کاندید را از نظر امنیت رتبه‌بندی کردیم. بر اساس مدل سلسله‌مراتبی (شکل ۱۰ در بخش ۳-۴)، ساختار سلسله‌مراتبی متناظر برای امنیت را مشابه شکل ۱۴ مشخص کردیم که در آن به جای زمان مصرفی، آسیب‌پذیری قرار می‌گیرد. برای اجرای این ساختار از الگوی سلسله‌مراتبی شکل ۱۵ استفاده کردیم که در آن ورودی ۱، نسبت آسیب‌پذیری سبک‌ها و ورودی ۲، اولویت سناریوها از بخش ۵-۲ است. خروجی حاصل از اجرای این ساختار سلسله‌مراتبی (مورد ۳ از شکل ۱۶)، رتبه سبک‌ها از نظر امنیت است.

۵-۲-۲ رتبه‌بندی نهایی سبک‌های کاندید

برای تعیین مناسب‌ترین سبک کاندید با توجه به خصیصه‌های کیفی کارایی، قابلیت اطمینان و امنیت، ساختار سلسله‌مراتبی شکل ۱۰ از بخش ۳-۴ را در نظر می‌گیریم و از شکل ۱۶ برای الگوی اجرا استفاده می‌کنیم. در استفاده از این الگو، ورودی ۱، جدول ۱۱ و ورودی ۲، جدول ۱۲ (نسبت اولویت خصیصه‌های کیفی به صورت دو به دو) است و با اجرای این الگو، خروجی ۳ را به دست آوردیم (جدول ۱۳). از جدول ۱۳ نتیجه می‌گیریم که با توجه به شرایط سیستم خودپرداز به ترتیب سبک‌های شیءگرا، داده مشترک و لوله و صافی برای طراحی معماری این سیستم مناسب است.

یکی از قابلیت‌های روش تحلیل سلسله‌مراتبی این است که می‌توان با تغییر اولویت خصیصه‌های کیفی، نتیجه تأثیر آن را در رتبه‌بندی سبک‌ها مشاهده کرد زیرا در الگوی اجرایی مدل سلسله‌مراتبی در شکل ۱۵ یکی از ورودی‌ها، نسبت اولویت معیارها است.

جدول ۱۲: نسبت اولویت خصیصه‌های کیفی.

امنیت	اطمینان	کارایی	اولویت خصیصه‌ها
۱/۵	۱/۲	۱	کارایی
۱/۳	۱	۲	اطمینان
۱	۳	۵	امنیت

جدول ۱۳: رتبه کلی سبک‌ها از لحاظ کارایی + اطمینان + امنیت.

رتبه	سبک
۳۹۲	شیءگرا
۲۴۶	داده مشترک
۳۶۲	لوله و صافی

٦- نتیجه‌گیری

سبک‌های معماری تأثیر مستقیم بر روی میزان تحقق خصیصه‌های کیفی در سیستم‌های نرم‌افزاری دارند. کارایی، قابلیت اطمینان و امنیت سه خصیصه مهم و کاربردی هستند که در موفقیت نرم‌افزار تأثیر به سزایی دارند. بنابراین بررسی میزان حمایت سبک‌های معماری از این سه خصیصه به منظور انتخاب مناسب‌ترین سبک بسیار سودمند است. در این مقاله برای ارزیابی کمی خصیصه‌های کیفی در سبک‌های معماری نرم‌افزار روشی مبتنی بر شبکه‌های پتری رنگی ارائه شد که محدودیت روش‌های گذشته را ندارد و با دقت بیشتری خصیصه‌های مورد نظر را ارزیابی می‌کند. در روش پیشنهادی با طراحی نرم‌افزار مورد نظر با سبک‌ها و مدل‌سازی سبک‌ها با شبکه پتری رنگی و استفاده از قوانین بیان‌شده برای ارزیابی خصیصه‌ها و همچنین مشخص‌کردن سناریوهای پرکاربرد در سیستم نرم‌افزاری می‌توان به صورت دقیق میزان کارایی، قابلیت اطمینان و امنیت هر سبک را تعیین کرد.

با استفاده از مدل‌سازی سبک‌های معماری با شبکه‌های پتری رنگی خصیصه‌های کیفی کارایی، قابلیت اطمینان و امنیت در سه سبک داده مشترک، شیءگرا و لوله و صافی به صورت کمی مورد ارزیابی قرار گرفت و مشخص شد که به ترتیب سبک‌های داده مشترک، شیءگرا و لوله و صافی بیشترین کارایی را دارند. همچنین به ترتیب سبک‌های داده مشترک، شیءگرا و لوله و صافی بیشترین قابلیت اطمینان را دارند. از نظر امنیت نیز به ترتیب سبک‌های شیءگرا، داده مشترک و لوله و صافی بیشترین امنیت را دارند.

مراجع

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

[2] K. Fukuzawa and M. Saeki, "Evaluating software architectures by coloured petri nets," in *Proc. 14th Int. Conf. on Software Engineering and Knowledge Engineering*, pp. 263-270, Ischia, Italy, 15-19 Jul.2002.

[3] B. Roy and T. C. N. Graham, *Methods for Evaluating Software Architecture: A Survey*, School of Computing, TR 545, p. 82, 2008.

[4] J. Bosch and P. Molin, "Software architecture design: evaluation and transformation," in *Proc. IEEE Conf. and Workshop on Engineering of Computer-Based Systems*, pp. 4-10, Mar. 1999.

[5] V. S. Sharma and K. S. Trivedi, "Quantifying software performance, reliability and security: an architecture-based approach," *J. of Systems and Software,* vol. 80, no. 4, pp. 493-509, Apr. 2007.

[6] S. S. Gokhale and K. S. Trivedi, "Reliability prediction and sensitivity analysis based on software architecture," *Proc. 13th Int. Symp. on Software Reliability Engineering*, pp. 64-75, 2002.

[7] R. Roshandel, N. Medvidovic, and L. Golubchik, "A Bayesian model for predicting reliability of software systems at the architectural level," in *Proc. Quality of Software Architectures 3rd int. Conf. on Software Architectures, Components, and Applications*, pp. 108-126, Medford, MA, USA, 11-13 Jul. 2007.

[8] S. S. Gokhale, *et al.*, "An analytical approach to architecture-based software performance and reliability prediction," *Performance Evaluation*, vol. 58, no. 4, pp. 391-412, Dec. 2004.

[9] V. S. Sharma and K. S. Trivedi, "Quantifying software performance, reliability and security: an architecture-based approach," *J. of Systems and Software*, vol. 80, no. 4, pp. 493-509, Apr. 2007.

[10] N. Yang, H. Yu, H. Sun, and Z. Qian, "Quantifying software security based on stochastic petri-nets," *J. of Computational Information Systems*, vol. 6, no. 9, pp. 3049-3056, Sep. 2010.

[11] S. Tahmasebipour and S. M. Babamir, "Ranking of common architectural styles based on availability, security and performance quality attributes," *J. of Computing and Security*, vol. 1, no. 2, pp. 83-93, 2014.

[12] W. L. Wang, D. Pan, and M. H. Chen, "Architecture-based software reliability modeling," *J. of Systems and Software*, vol. 79, no. 1, pp. 132-146, Jan. 2006.

[13] H. Koziolek, B. Schlich, and C. Bilich, "A large-scale industrial case study on architecture-based software reliability analysis," in *Proc. 21st IEEE Int. Symp. on Software Reliability Engineering*, pp. 279-288, Nov. 2010.

[14] S. M. Sharafi, G. A. Ghazvini, and S. Emadi, "An analytical model for performance evaluation of software architectural styles," in *Proc. 2nd Int. Conf. on Software Technology and Engineering*, pp. 394-398, 2010.

[15] M. L. Yin, C. L. Hyde, and L. E. James, "A petri-net approach for early-stage system-level software reliability estimation," in *Proc. Int. Symp. on Product Quality and Integrity.*, pp. 100-105, 24-27 Jan. 2000.

[16] W. L. Wang and M. H. Chen, "Heterogeneous software reliability modeling," in *Proc. 13th Int. Symp. on Software Reliability Engineering*, pp. 41-52, 12-15 Nov. 2002.

[17] R. Pressman, *Software Engineering: A Practioner's Approach*, McGraw-Hill, 8th Edition, 2014.

[18] ISO, *ISO/IEC 9126: Information Technology-Software Product Evaluation-Quality Characteristics and the Guidelines for Their Use*, 1991.

[19] P. Clements and R. Nord, *Documenting Software Architectures: Views and Beyond*, Addison-Wesley, 2010.

[20] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, Springer, 2009.

[21] E. Triantaphyllou, B. Kovalerchuk, L. Mann, and G. M. Knapp, "Determining the most important criteria in maintenance decision making," *J. of Quality in Maintenance Engineering*, vol. 3, no. 1, pp. 16-28, Mar. 1997.

[22] T. L. Saaty, "How to make a decision: the analytic hierarchy process," *European J. of Operational Research*, vol. 48, no. 1, pp. 9-26, Sep. 1990.

[23] www.expertchoice.com.

[24] S. S. Gokhale, "Architecture-based software reliability analysis: overview and limitations," *IEEE Trans. on Dependable and Secure Computing*, vol. 4, no. 1, pp. 32-40, Jan. 2007.

[25] www.cpntools.org.

[26] N. Yang, H. Yu, Z. Qian, and H. Sun, "Modeling and quantitatively predicting software security based on stochastic petri-nets," *Mathematical and Computer Modelling*, vol. 55, no. 1, pp. 102-112, Jan. 2012.

[27] N. Yang, H. Yu, H. Sun, and Z. Qian, "Modeling UML sequence diagrams using extended petri nets," in *Proc. Int. Conf. on Information Science and Applications*, 8 pp., 21-23 Apr. 2010.

[28] S. Yu and S. Zhou, "A survey on metric of software complexity," in *Proc. 2nd IEEE Int. Conf. on Information Management and Engineering*, pp. 352-356, 16-18 Apr. 2010.

[29] J. Bosch, "Design and use of industrial software architectures," in *Proc. Conf. on Technology of Object-Oriented Languages and Systems*, pp. 404-404, Jun. 1999.

[30] H. Zhu, *Software Design Methodology: From Principles to Architectural Styles*, Butterworth-Heinemann, 2005.

هدی بانکی تحصیلات خود را در مقطع کارشناسی در سال ۱۳۸۷ از دانشگاه آزاد، واحد تهران مرکزی در رشته مهندسی نرم‌افزار و در مقطع کارشناسی ارشد در سال ۱۳۹۱ از دانشگاه کاشان به پایان رساند. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: معماری نرم‌افزار، مهندسی نرم‌افزار و سیستم‌های تطبیقی.

سید مرتضی بابامیر تحصیلات خود را در مقطع کارشناسی از دانشگاه فردوسی مشهد و در مقاطع کارشناسی ارشد و دکتری از دانشگاه تربیت مدرس دریافت نمود. نام‌برده در

سال‌های ۱۳۶۵ تا ۱۳۷۱ به‌عنوان کارشناس سیستم‌های نرم‌افزاری در صنایع هواپیمایی ایران فعالیت نمود. دکتر بابامیر در حال حاضر دانشیار گروه مهندسی کامپیوتر دانشگاه کاشان و مدیر مسئول مجله علمی ترویجی محاسبات نرم است. زمینه‌ها و فعالیت‌های تحقیقاتی وی، معماری نرم‌افزار، مهندسی نرم‌افزار، سیستم‌های توزیعی و محاسبات ابری است.