سال چهاردهم، شماره‌های ۵۱ و ۵۲ ، بهار و تابستان ۱۴۰۱
صص: ۱۹۵_۲۰۸

دو فصلنامه علمی
فناوری اطلاعات و ارتباطات ایران

شناسایی تاکتیک‌های معماری در کد منبع بر اساس یک رویکرد معنایی

احسان شریفی* احمد عبدالله زاده بارفروش**

* دانشجوی دکتری، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، ایران

** استاد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، ایران

تاریخ دریافت: ۱۴۰۰/۰۳/۱۷ تاریخ پذیرش: ۱۴۰۰/۰۶/۰۶

نوع مقاله: پژوهشی

چکیده

سامانه‌های نرم‌افزاری تا زمانی که قابلیت اعمال تغییرات را داشته باشند زنده هستند و امکان استفاده از آن‌ها وجود دارد. اعمال تغییرات در کد منبع بدون توجه به تأثیرات آن می‌تواند باعث فرسایش معماری سامانه‌ی نرم‌افزاری شود. فرسایش معماری به‌مرور، امکان انجام تغییرات را غیرممکن می‌نماید و سامانه روبه‌زوال می‌رود. تصمیمات معماری در کد منبع معمولاً توسط تاکتیک‌های معماری محقق می‌شوند. تاکتیک‌ها تصمیم‌های ریزدانه‌ای هستند که برای تحقق یک ویژگی کیفیتی خاص اتخاذ می‌شوند. شناسایی تاکتیک‌ها در کد منبع این امکان را برای تولیدکنندگان فراهم می‌کند که اعمال تغییرات در کد منبع را با آگاهی از مکان پیاده‌سازی این تصمیم‌ها انجام دهند. لذا فرآیند فرسایش معماری کندتر شده و سامانه‌ی نرم‌افزاری دیرتر به سمت زوال حرکت می‌نماید. بدین منظور، در این مقاله یک رویکرد معنایی به‌منظور شناسایی تاکتیک‌های معماری در کد منبع معرفی می‌شود. بر اساس این رویکرد، مفهوم جدیدی به نام ریزتاکتیک معرفی می‌شود که امکان شناسایی تاکتیک‌های معماری را با استفاده از یک رویکرد معنایی مبتنی بر وب معنایی و آنتولوژی ارتقاء می‌بخشد. نتایج حاصل از ارزیابی رویکرد پیشنهادی نشان می‌دهد که امکان شناسایی تاکتیک‌ها در این روش با دقت و کیفیت بهتری نسبت به روش‌های مشابه انجام می‌شود.

واژگان کلیدی: تاکتیک معماری، ریزتاکتیک، آنتولوژی، مدل معنایی

۱. مقدمه

سامانه‌های نرم‌افزاری مانند یک موجود زنده به‌طور مستمر در حال تغییر هستند. بخش عمده‌ی فعالیت یک تیم توسعه، اعمال تغییر و ایجاد تکامل در محصولی است که بر اساس نیازمندی‌های جدید مشتری در حال رشد است. تا زمانی که امکان اعمال تغییرات در محصول نرم‌افزاری وجود دارد چرخه حیات محصول در جریان بوده و محصول زنده است.

هر زمان که امکان تکامل و تغییر فراهم نباشد یا به‌سختی انجام شود چرخه حیات محصول به اتمام رسیده است و محصول جدید جایگزین قبلی می‌شود. فرسایش معماری زمانی اتفاق می‌افتد که تیم توسعه بدون توجه به معماری کلی و به‌مرور، تغییراتی محلی را در سطح کد اعمال می‌کند و به دلیل ناآگاهی از تصمیمات معماری در سطح کلان

نویسنده مسئول: احمد عبدالله زاده بارفروش Ahmad@aut.ac.ir

باعث فرسایش معماری می‌گردد. مهم‌ترین تأثیر این فرسایش پایین آمدن قابلیت نگهداری نرم‌افزار است[۱].

فرسایش معماری[۱] زمانی اتفاق می‌افتد که تیم توسعه بدون توجه به معماری کلی و به‌مرور، تغییراتی محلی را در سطح کد اعمال می‌کند و به دلیل ناآگاهی از تصمیم‌های معماری در سطح کلان باعث فرسایش معماری می‌شود. مهم‌ترین تأثیر این فرسایش پایین آمدن قابلیت نگهداری نرم‌افزار است[۱].

الگوها و تاکتیک‌های معماری دو نمونه از تصمیم‌های معماری هستند که برای تحقق ویژگی‌های کیفیتی اتخاذ می‌شوند. الگوهای معماری راه‌حل‌های درشت‌دانه‌ای هستند که از مجموعه‌ای از تاکتیک‌های معماری و الگوهای طراحی تشکیل‌شده‌اند و طیف وسیعی از ویژگی‌های کیفیتی را تحت تأثیر قرار می‌دهند. در مقابل، تاکتیک‌های معماری تصمیم‌های طراحی ریزدانه‌ای هستند که هدف آن‌ها تحقق یک ویژگی کیفیتی مشخص است[۲]. به‌عنوان‌مثال، یکی از دغدغه‌های طراحی مرتبط با ویژگی کیفیتی امنیت، «نحوه جلوگیری از حمله[۲] به یک سیستم» است و یک تصمیم طراحی یا به عبارت صحیح‌تر تاکتیک معماری متناظر با آن «تائید اعتبار»[۳] کاربر است.

تاکتیک‌های معماری به دلیل ماهیت ریزدانگی که دارند مهم‌ترین اجزاء معماری هستند که امکان شناسایی در کد منبع نرم‌افزار رادارند. لذا شناسایی و مراقبت از تاکتیک‌های معماری در خلال فرآیند توسعه‌ی نرم‌افزار می‌تواند فرآیند فرسایش معماری را کند نماید. متأسفانه ماهیت تاکتیک‌های معماری با الگوهای طراحی تفاوت دارد. الگوهای طراحی با روش‌های تحلیل ساختاری قابل‌شناسایی هستند؛ اما تاکتیک‌های معماری به‌صورت نقش‌ها و تعاملات بین آن‌ها توصیف می‌شوند و ساختار استانداردی برای پیاده‌سازی ندارند[۳].

در این مقاله با معرفی مفهوم جدیدی به نام ریزتاکتیک[۴]، یک رویکرد معنایی بر پایه‌ی آنتولوژی و وب معنایی ارائه می‌شود که امکان شناسایی تاکتیک‌های معماری در کد منبع را فراهم می‌نماید. بر اساس این رویکرد، تاکتیک‌های معماری با استفاده از اجزاء ریزدانه‌تری به نام ریزتاکتیک مدل می‌شوند. این اجزاء با استفاده از آنتولوژی توصیف‌شده و از طریق پرس‌وجوهای وب‌معنایی و استفاده از یک رویکرد مبتنی بر مشابهت معنایی امکان شناسایی تاکتیک‌های معماری را فراهم می‌کنند.

اولین نوآوری این مقاله معرفی مفهوم ریزتاکتیک است که امکان شناسایی تاکتیک‌های معماری را در سطح کد تسهیل می‌نماید. دومین نوآوری ارائه یک راهکار معنایی مبتنی بر وب معنایی و آنتولوژی برای شناسایی تاکتیک‌های معماری است که تاکنون توسط سایر پژوهشگران موردتوجه قرار نگرفته است. به‌کارگیری رویکرد معنایی در این حوزه این امکان را فراهم می‌کند که شناسایی تاکتیک‌های معماری به ساختار نحوی پیاده‌سازی آن‌ها وابسته نبوده و نتایج به‌دست‌آمده دارای صحت و اعتبار بیشتری باشد.

ساختار مقاله در ادامه بدین‌صورت است. در بخش دوم پیشینه تحقیق به‌صورت اجمالی بررسی می‌شود. سپس در بخش سوم رویکرد پیشنهادی معرفی می‌شود. در بخش چهارم با انجام یک مطالعه موردی نتایج حاصل از رویکرد پیشنهادی مورد آزمایش قرار می‌گیرد. در بخش پنجم نتایج حاصل موردبحث و بررسی قرارگرفته و در بخش ششم نتیجه‌گیری انجام‌شده و فعالیت‌های آتی معرفی می‌شوند.

۲. پیشینه تحقیق

در حوزه‌ی تشخیص الگوهای طراحی پژوهش‌های متنوعی انجام‌شده است؛ اما در حوزه‌ی شناسایی تاکتیک‌های معماری پژوهش‌های انجام‌شده اندک است. در ادامه به معرفی چند پژوهش شاخص در این حوزه می‌پردازیم.

رویکرد برخی از پژوهش‌های این حوزه، استخراج تاکتیک‌های معماری از الگوهای طراحی یا معماری است. یکی از پژوهش‌های مطرح در این حوزه یک رویکرد جدید برای استخراج تاکتیک‌های معماری مربوط به ویژگی کیفیتی امنیت از الگوهای شناخته‌شده معرفی نموده است[۴]. هسته‌ی اصلی این رویکرد، آزمودن الگوهای معماری شناخته‌شده در حوزه‌ی امنیت است. این آزمون به‌منظور بررسی مجموعه‌ای از شرایط است که در صورت احراز، الگوی موردنظر به‌عنوان یک تاکتیک معماری شناخته خواهد شد. ضعف این دیدگاه عدم توجیه نحوه تعیین شرایط مذکور است.

در پژوهش دیگر از تکنیک‌های داده‌کاوی برای یافتن نمونه‌های الگوهای طراحی در تاکتیک‌های معماری استفاده‌شده است[۵]. هدف از این پژوهش ایجاد «نقاط تغییری»[۵] است که امکان پیاده‌سازی یا شناسایی تاکتیک‌های معماری را بر اساس الگوهای طراحی مختلف فراهم نماید.

یکی دیگر از پژوهش‌های مطرح در این حوزه با استفاده از تکنیک‌های بازیابی اطلاعات و یادگیری ماشین تاکتیک‌های معماری را شناسایی نموده است. دیدگاه اصلی این پژوهش مبتنی بر یادگیری واژگانی است که در پیاده‌سازی‌های مختلف تاکتیک‌های معماری مورداستفاده قرار می‌گیرند[۶]. نقطه‌ضعف اصلی این رویکرد وابسته بودن به واژگانی است که قبلاً در پیاده‌سازی‌های مختلف تاکتیک‌ها

---

[۱] Architectural erosion

[۲] Attack

[۳] Authentication

[۴] Micro Tactic

[۵] Variability Points

مورداستفاده قرارگرفته‌اند. تغییر این واژگان یا استفاده از واژگان هم‌معنی می‌تواند دقت شناسایی این روش را به‌شدت کاهش دهد.
تحقیق دیگری در این حوزه ایده‌ی ایجاد یک موتور جستجو برای تاکتیک‌های معماری را مطرح نموده است[۷]. ایده اصلی این دیدگاه معرفی مفهوم «کلون‌های تاکتیک»[۱] به‌عنوان المان اصلی این موتور جستجو است. یک رویکرد مشابه نیز برای یافتن کدهای مهم ازنظر معماری پیشنهاد شده است[۸]. در این رویکرد از تکنیک‌های بازیابی اطلاعات و «تحلیل برنامه»[۲] برای شناسایی کدهای موردنظر استفاده می‌کند.
پژوهش دیگر در این حوزه مسئله‌ی شناسایی تاکتیک را به مسئله‌ی دسته‌بندی[۳] کردن متن نگاشت نموده و با استفاده از مدل‌های زبانی نظیر برت[۴] سعی نموده است که این مسئله را در حوزه پردازش زبان طبیعی حل نماید[۹].
در یک رویکرد دیگر برای ارزیابی معماری سامانه‌های نرم‌افزاری، از شناسایی الگوها و تاکتیک‌های پیاده‌سازی شده در آن‌ها استفاده شده است[۱۰]. این روش با استفاده از ابزاری به نام آرچی[۵] تاکتیک‌ها و الگوهای معماری را از کد منبع استخراج نموده و پس از توصیف آن‌ها در قالب مدل‌های معماری، میزان تأثیر آن‌ها بر ویژگی‌های کیفیتی را با استفاده از یک‌زبان توصیف نیازمندی هدف‌گرا[۶] بررسی می‌نماید.

۳. رویکرد پیشنهادی

در این بخش رویکرد ردیابی پیشنهادی معرفی می‌شود. در ابتدا برخی مفاهیم نظری[۷] مرتبط با این رویکرد تشریح می‌شوند. سپس فعالیت‌های مربوط به روش پیشنهادی با جزئیات تشریح می‌شوند.

۱,۳ مفاهیم نظری

در این بخش در ابتدا مفهوم ریزتاکتیک معرفی‌شده و در ادامه یک متامدل و آنتولوژی برای توصیف آن ارائه می‌شود. سپس یک آنتولوژی برای توصیف کدهای شیءگرا معرفی‌شده و در انتها نیز چارچوبی برای توصیف معنایی کد شیءگرا به نام کدآنتولوژی پلاس معرفی می‌شود.

۱,۱,۳ معرفی مفهوم ریزتاکتیک

شناسایی المان‌هایی در کد منبع که به‌طور بالقوه محقق کننده تاکتیک‌ها باشند می‌تواند فرآیند شناسایی تاکتیک‌های معماری در کد منبع را تسهیل نمایند. لذا می‌بایست به ازای هر تاکتیک معماری، ریزساختارهایی[۸] را که محقق کننده آن تاکتیک در کد منبع هستند شناسایی نمود. ریزساختارها، المان‌هایی از کد منبع هستند که یک موجودیت یکتا یا ارتباط بین دو یا چند موجودیت را در کد منبع تعریف می‌نمایند. مهم‌ترین ویژگی ریزساختارها، قابل‌شناسایی بودن آن‌ها به‌طور خودکار است و به دلیل برخورداری از این ویژگی استخراج آن‌ها از کد منبع با دقت و یادآوری بالا قابل انجام است[۱۱].
در این تحقیق مفهوم ریزتاکتیک برای اشاره به ریزساختارهایی انتخاب‌شده است که در محقق نمودن اهداف یک تاکتیک معماری نقش دارند. ایده اولیه ریزتاکتیک برگرفته از دو مفهوم در حوزه الگوهای طراحی است. اولین مفهوم «ریز الگو[۹]» است که برای الگوهای طراحی معرفی‌شده است [۱۲]. مفهوم بعدی «اثرهای الگوی طراحی[۱۰]» است [۱۳].
هر دو مفهوم فوق به‌عنوان یک ریزساختار قصد دارند که فرآیند شناسایی الگوهای طراحی در کد منبع را تسهیل نمایند. از این جنبه این دو ریزساختار شباهت معنایی به مفهوم ریزتاکتیک دارند. اما تفاوت اصلی ریزتاکتیک نسبت به دو مفهوم فوق در گستردگی و مقیاس‌پذیری آن است. درواقع می‌توان این‌گونه بیان کرد که ریزتاکتیک‌ها خود شامل ریز الگوها و اثرهای الگوی طراحی هستند، اما محدود به آن‌ها نبوده و المان‌های بیشتری را می‌توانند در برگیرند. تاکتیک‌های معماری زمان اجرا شامل نقش‌ها و تعاملات بین این نقش‌ها می‌باشند. لذا در وهله‌ی اول نقش‌های موجود در یک تاکتیک انتخاب مناسبی هستند که به‌عنوان ریزتاکتیک انتخاب شوند.

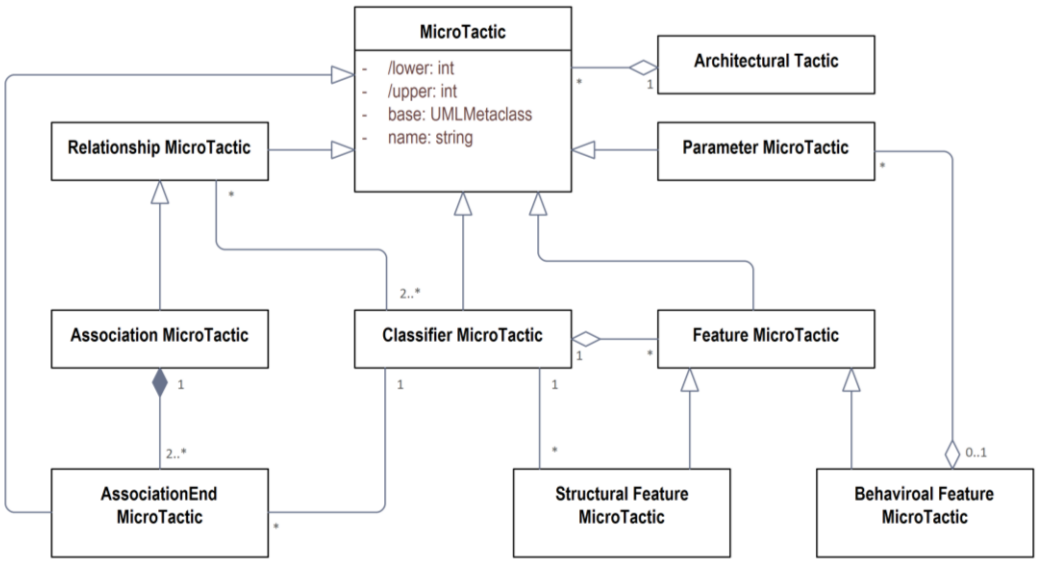
۲,۱,۳ معرفی متامدل ریزتاکتیک

به‌منظور تشریح دقیق ریزتاکتیک‌ها نیاز به ارائه یک متامدل داریم. یکی از معدود پژوهش‌های موجود تاکتیک‌های معماری را در قالب نقش‌ها و تعاملات بین آن‌ها و بر پایه‌ی زبان آربی‌ام‌ال[۱۱] توصیف نموده است [۱۴].
متامدل پیشنهادی ریزتاکتیک در این مقاله بر اساس متامدل زبان آربی‌ام‌ال ایجادشده است. در این متامدل اجزائی از کد منبع که می‌توانند در نقش ریزتاکتیک ظاهر شوند ارائه‌شده است.
شکل ۱ متامدل پیشنهادی ریزتاکتیک را نمایش می‌دهد. اجزای این متامدل، ریزتاکتیک‌های مرتبط با یک تاکتیک معماری را نمایش

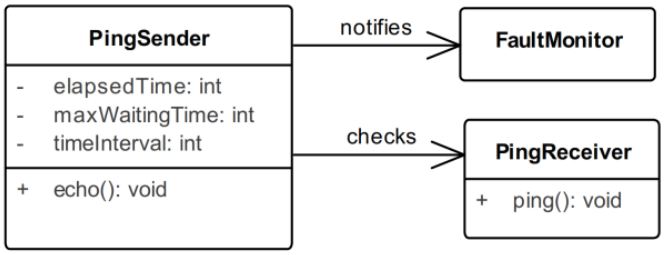
---

[۱] Tactical-clones
[۲] Program analysis
[۳] Classify
[۴] BERT
[۵] Archie
[۶] Goal-oriented
[۷] Theory
[۸] Microstructure
[۹] Micro Pattern
[۱۰] Design Pattern Clues
[۱۱] RBML

می‌دهند که امکان شناسایی خودکار آن‌ها در کد منبع یک پروژه وجود دارد.



شکل ۱. متامدل پیشنهادی ریزتاکتیک



شکل ۲. مدل ریزتاکتیک مربوط به تاکتیک پینگ/اکو

جدول ۱: ریزتاکتیک‌های مربوط به تاکتیک پینگ/اکو

ردیف	عنوان ریزتاکتیک	نوع ریزتاکتیک
۱	**PingSender**	دسته‌بند
۲	**FaultMonitor**	دسته‌بند
۳	**PingReceiver**	دسته‌بند
۴	**maxWaitingTime**	ویژگی ساختاری
۵	**timeInterval**	ویژگی ساختاری
۶	**elapsedTime**	ویژگی ساختاری
۷	**Ping**	ویژگی رفتاری
۸	**echo**	ویژگی ساختاری
۹	**notifies**	رابطه
۱۰	**checks**	رابطه

بر اساس این متامدل، ریزتاکتیک می‌تواند یک دسته‌بند[۱]، ویژگی، پارامتر یا رابطه باشد. شکل ۲ مدل مربوط به تاکتیک پینگ/اکو[۲] را که بر اساس متامدل شکل ۱ ایجادشده است نمایش می‌دهد. تاکتیک پینگ/اکو یکی از تاکتیک‌های «شناسایی خرابی»[۳] است که ذیل تاکتیک‌های مرتبط با ویژگی کیفیتی دسترس‌پذیری[۴] قرار دارد[۱۵]. اجزای این مدل نیز در جدول ۱ تشریح شده است.

۳،۱،۳ آنتولوژی ریزتاکتیک

تعاریف مرتبط با آنتولوژی در حوزه علوم کامپیوتر بسیار متفاوت و متنوع است. مشهورترین تعریف متعلق به استادر[۵] است که آنتولوژی را بدین‌صورت تعریف می‌کند: «آنتولوژی یک توصیف صوری[۶] و صریح[۷] از یک مفهوم‌سازی[۸] اشتراکی است»[۱۶].

در حوزه‌ی تاکتیک‌های معماری نیازمند چنین ساختاری هستیم تا از طریق توصیف معنایی دانش این حوزه، فرآیند شناسایی تاکتیک‌ها در کد منبع تسهیل شود. این آنتولوژی در وهله‌ی نخست می‌بایست ساختار ریزتاکتیک‌ها را بر اساس متامدل ریزتاکتیک توصیف نماید.

---

[۱] Classifier
[۲] Ping/Echo
[۳] Fault Detection
[۴] Availability
[۵] Studer
[۶] Formal
[۷] Explicit
[۸] Conceptualization

در وهله‌ی دوم نیز امکان توصیف مفاهیم و روابط موجود در حوزه‌ی تاکتیک‌های معماری را فراهم نماید. آنتولوژی در این مقاله، یک پنج‌تایی مرتب شامل مفاهیم (C)، روابط رده‌بندی (R_T)، روابط غیر رده‌بندی (R_N)، اصول (A) و دامنه (D) است که بدین شکل تعریف می‌شود[۱۷]:

$$O = (C, R_T, R_N, A, D) \quad (۱)$$

فرآیند ایجاد آنتولوژی ریزتاکتیک طی سه مرحله انجام می‌شود. در اولین مرحله یک «آنتولوژی پایه»[۱] ایجاد می‌شود که مفاهیم اصلی ریزتاکتیک را توصیف می‌نماید. در مرحله‌ی دوم یک «آنتولوژی حوزه»[۲] بر اساس مفاهیم و رده‌بندی‌های موجود در این حوزه ایجاد می‌شود. در مرحله‌ی سوم نیز یک آنتولوژی حوزه بر اساس منابع مرتبط با این حوزه ایجاد می‌شود.

در اولین مرحله و بر اساس ساختار متامدل ریزتاکتیک معرفی‌شده در شکل۱، آنتولوژی پایه ایجاد می‌شود. بدین منظور، قواعد زیر را تعریف می‌کنیم:

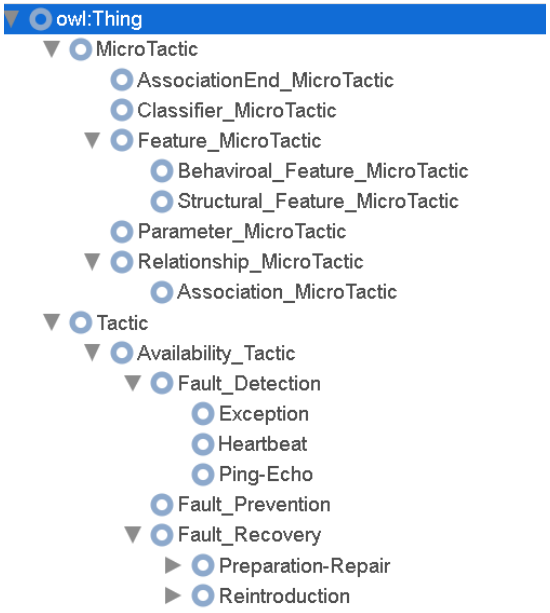
- قاعده ۱: هر دسته‌بند در متامدل به یک مفهوم در آنتولوژی نگاشت می‌شود.
- قاعده ۲: هر رابطه ارث‌بری در متامدل به یک رابطه سلسله مراتبی در آنتولوژی نگاشت می‌شود.
- قاعده ۳: هر رابطه تناظر[۳] در متامدل به یک رابطه غیر رده‌بندی یا دودویی در آنتولوژی نگاشت می‌شود.

شکل ۳ بخشی از آنتولوژی پایه را ذیل مفهوم «MicroTactic» نمایش می‌دهد. همان‌گونه که در متامدل شکل ۱ مشاهده می‌شود، «Classifier Microtactic» به‌عنوان یک دسته‌بند بر اساس قاعده‌ی ۱ به مفهوم «Classifier_Microtactic» در آنتولوژی پایه نگاشت شده است. از طرف دیگر بر اساس قاعده‌ی ۲ و با توجه به رابطه‌ی ارث‌بری آن با کلاس «MicroTactic»، یک رابطه‌ی سلسله مراتبی بین مفهوم «Classifier_Microtactic» و مفهوم «MicroTactic» در آنتولوژی ایجادشده است.

در مرحله‌ی دوم، آنتولوژی حوزه مربوط به تاکتیک‌های معماری ایجاد می‌شود. آنتولوژی حوزه، معانی واژگان موجود در یک حوزه را تشریح نموده و امکان توصیف بهتر مفاهیم و روابط بین آن‌ها در یک فیلد خاص را فراهم می‌کند[۱۸]. در این مرحله، از رده‌بندی ارائه‌شده توسط بَس و همکاران برای ایجاد اولین بخش از آنتولوژی حوزه‌ی تاکتیک استفاده می‌شود[۱۵]. این رده‌بندی، تاکتیک‌های معماری مطرح در حوزه‌ی نرم‌افزار را بر اساس ویژگی‌های کیفیتی مرتبط با آن‌ها دسته‌بندی نموده است. شکل ۳ بخشی از آنتولوژی حوزه‌ی تولیدشده بر اساس رده‌بندی فوق را ذیل مفهوم «Tactic» نمایش می‌دهد. این بخش از آنتولوژی به‌صورت دستی و توسط نویسنده اول مقاله ایجادشده است. اما ایجاد آنتولوژی حوزه به روش دستی یک فرآیند پرهزینه، زمان‌بر و مستعد خطاست. خودکارسازی عملیات ساخت آنتولوژی که از آن با عنوان یادگیری آنتولوژی[۴] یاد می‌شود، هزینه‌ی ساخت و میزان خطا را کاهش می‌دهد.

در بخش سوم و به‌منظور تکمیل آنتولوژی حوزه، از رویکردی که توسط نویسنده اول این مقاله استفاده می‌کنیم[۱۹]. هدف اصلی این پژوهش جمع‌آوری «توده‌ی متون»[۵] جامع برای تولید یک آنتولوژی فازی[۶] است[۲۰]. توده متون شامل کلیه‌ی منابعی است که به‌طور مستقیم یا غیرمستقیم به حوزه‌ی موردنظر مرتبط بوده و مفاهیم و روابط موجود در حوزه با دقت قابل قبولی از این توده متون قابل‌استخراج است. لذا جامعیت توده‌ی متون تأثیر مستقیم بر کیفیت آنتولوژی حوزه‌ی نهایی دارد. با استفاده از این رویکرد، امکان جمع‌آوری خودکار توده متون جامع در حوزه‌ی تاکتیک‌های معماری فراهم می‌شود.

بخش سوم از آنتولوژی حوزه‌ی تاکتیک با استفاده از این توده متون و بر اساس رویکرد [۱۹] ایجاد می‌شود.



شکل ۳. بخشی از آنتولوژی ریزتاکتیک

آنتولوژی نهایی از درون‌برد[۷] سه آنتولوژی تولیدشده ایجاد می‌شود. شکل ۳ بخشی از این آنتولوژی نهایی را نمایش می‌دهد. این آنتولوژی به‌صورت آنلاین در دسترس است[۸].

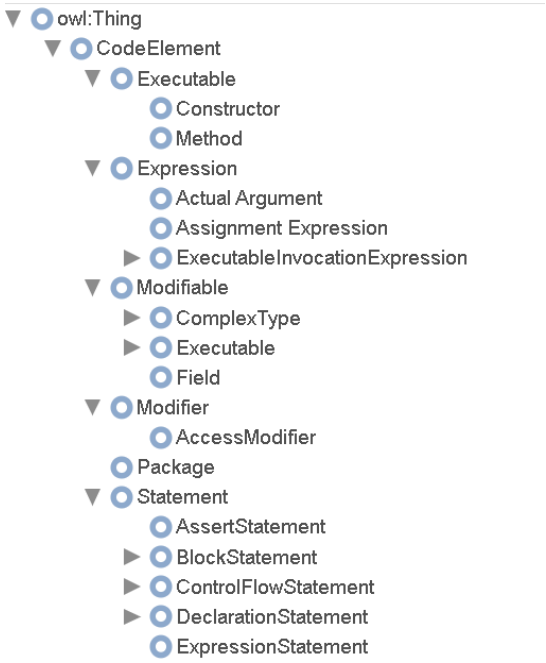
---

[۱] Base ontology
[۲] Domain ontology
[۳] Association
[۴] Ontology Learning
[۵] Corpus
[۶] Fuzzy
[۷] Import
[۸] http://islab.ceit.aut.ac.ir/mt

۴،۱،۳ آنتولوژی کد شیءگرا

به‌منظور ایجاد مدل معنایی از کد منبع نیازمند وجود یک آنتولوژی در حوزه‌ی زبان‌های شیءگرا هستیم.

در این پژوهش از یک آنتولوژی که برای برای مدل‌سازی معنایی زبان‌های برنامه‌نویسی شیءگرا معرفی‌شده است استفاده می‌کنیم[۲۱]. این آنتولوژی به زبان اُول۲[۱] و با ابزار *پروتج*[۲] ایجادشده است و شامل ۶۵ کلاس، ۸۶ ویژگی شیء[۳] و ۱۱ ویژگی داده[۴] است. در این آنتولوژی موجودیت‌های ساختاری مرتبط با تمامی زبان‌های برنامه‌نویسی شیءگرا نظیر کلاس‌ها، متدها، متغیرها، جملات[۵] و عبارت‌ها[۶] را در قالب کلاس‌های مجزا[۷] بازنمایی می‌شوند. طراحی این آنتولوژی بر اساس «الگوهای طراحی آنتولوژی»[۸] انجام‌شده است. این آنتولوژی به‌صورت آنلاین در دسترس است[۹]. شکل ۴ بخشی از این آنتولوژی را نمایش می‌دهد.



شکل ۴. بخشی از آنتولوژی کد شیءگرا

۵،۱،۳ چارچوب کدآنتولوژی پلاس[۱۰]

افزایش روزافزون حجم پروژه‌های نرم‌افزاری نیازمند راهکاری است که امکان مدیریت تغییر، بازنگری[۱۱] و بازآرایی[۱۲] کد را فراهم نماید.

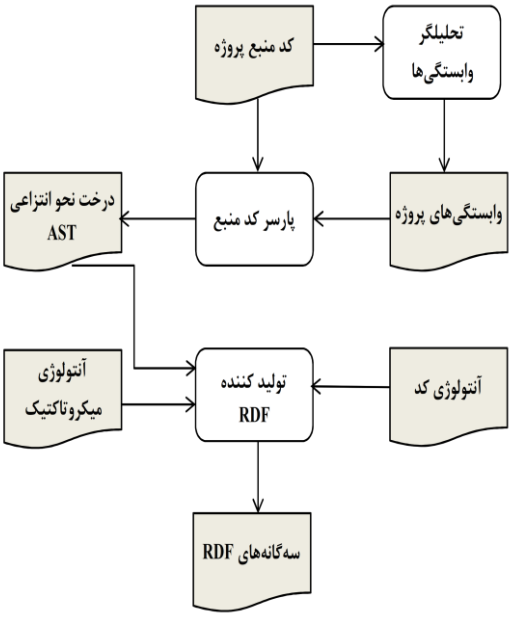
لذا امکان انجام پرس‌وجو بر روی کد منبع یکی از چالش‌های مهم در حوزه‌ی مهندسی نرم‌افزار است.

پشته‌ی فناوری وب معنایی یک مجموعه استاندارد برای بازنمایی و پرس‌وجوی اطلاعات ساخت‌یافته ارائه می‌نماید. کدآنتولوژی چارچوبی بر مبنای وب معنایی است که امکان تبدیل کد منبع پروژه به یک مدل معنایی را فراهم می‌کند[۲۱]. این ساختار معنایی امکان انجام پرس‌وجوهای متنوع بر روی کد منبع پروژه را با استفاده از زبان قدرتمند اسپارکل[۱۳] فراهم می‌کند. ورودی این چارچوب، کد منبع پروژه به زبان جاوا است و خروجی آن‌یک ساختار معنایی به فرمت آردی‌اف[۱۴] است. این چارچوب فرآیند تولید مدل آردی‌اف از کد منبع را طی سه مرحله انجام می‌دهد. در مرحله‌ی اول پروژه به‌منظور دانلود وابستگی‌هایش مورد تحلیل قرار می‌گیرد. سپس در مرحله‌ی دوم یک «درخت نحوی انتزاعی»[۱۵] از کد منبع ایجاد می‌شود. سپس در مرحله‌ی سوم این درخت مورد پردازش قرارگرفته و با استفاده از آنتولوژی کد که در بخش ۴.۱.۳ معرفی شد سه‌گانه‌های آردی‌اف ایجاد می‌شوند.

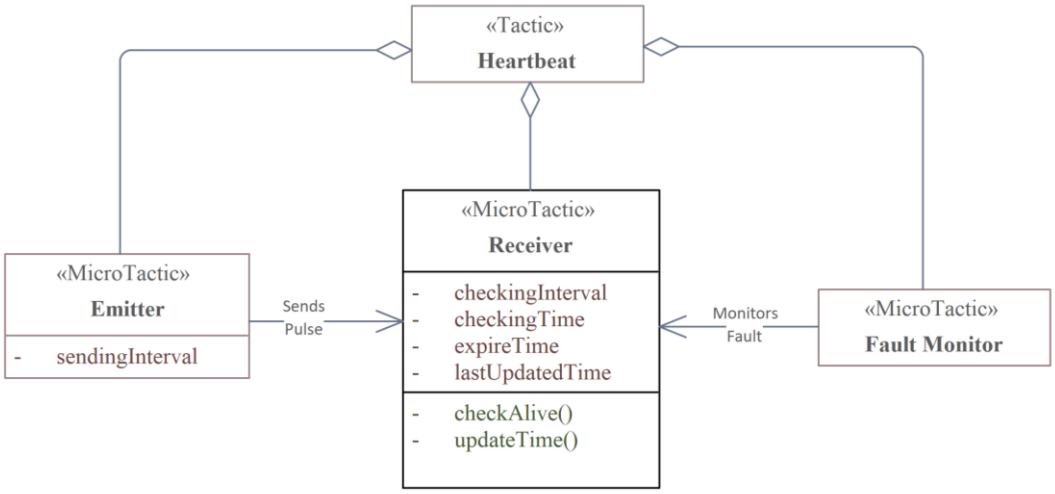
به‌منظور به‌کارگیری چارچوب کدآنتولوژی در این مقاله، تغییراتی در این چارچوب ایجادشده است و ازاین‌پس با عنوان کدآنتولوژی پلاس به آن رجوع می‌نماییم. شکل ۵ چارچوب کدآنتولوژی پلاس را نمایش می‌دهد. بدین منظور در حین پردازش درخت نحوی انتزاعی مربوط به کد منبع، بین «موجودیت‌های نامدار»[۱۶] موجود در کد منبع و مفاهیم موجود در آنتولوژی ریزتاکتیک که در بخش ۳.۱.۳ معرفی‌شده است پیوندهایی برقرار می‌شود. ایجاد این پیوندها امکان شناسایی معنایی ریزتاکتیک‌ها را در کد منبع فراهم می‌کند.

---

[۱] OWL۲
[۲] Protégé
[۳] Object property
[۴] Data property
[۵] Statement
[۶] Expression
[۷] Disjoint
[۸] Ontology Design Patterns
[۹] http://doi.org/۱۰،۵۲۸۱/zenodo.۵۷۷۹۳۹
[۱۰] Codeontology+
[۱۱] Review
[۱۲] Refactor
[۱۳] SPARQL
[۱۴] RDF
[۱۵] Abstract Syntax Tree (AST)
[۱۶] Named Entity



شکل ۵. معماری کدآنتولوژی پلاس



شکل ۶. مدل ریزتاکتیک ضربان قلب

۲,۳ تشریح رویکرد پیشنهادی

با توجه به مفاهیم و پیش‌نیازهای معرفی‌شده در بخش ۱.۳ فعالیت‌های رویکرد پیشنهادی در این بخش تشریح می‌شوند. این فعالیت‌ها شامل مدل‌سازی معنایی کد منبع، ایجاد مخزن معنایی ریزتاکتیک، پیشنهاد تاکتیک و شناسایی تاکتیک است.

۱,۲,۳ مدل‌سازی معنایی کد

اولین فعالیت در رویکرد پیشنهادی، تبدیل کد منبع پروژه به مدل معنایی است که امکان تحلیل و استنتاج کد منبع را فراهم نموده و دستیابی به اجزاء کد را تسهیل نماید. برای این منظور از چارچوب کدآنتولوژی‌پلاس که در بخش ۵.۱.۳ معرفی شد استفاده می‌کنیم.

نحوه‌ی ایجاد مدل معنایی وابسته به رویکرد موردنظر است. اگر هدف شناسایی تاکتیک‌های معماری در بخش مشخصی از کد منبع نظیر یک کلاس، مؤلفه یا زیرسیستم است، می‌توان فقط همان بخش از کد را به مدل معنایی تبدیل نمود. در این صورت فرآیند شناسایی تاکتیک‌های معماری با سرعت و کارایی بیشتری انجام خواهد شد. در غیر این صورت می‌بایست تمامی کد منبع به ساختار معنایی تبدیل شود که نیازمند زمان و هزینه بیشتری است.

مدیریت تغییرات نیز بدین‌صورت انجام می‌شود که با تغییر هر بخش از کد به‌طور خودکار یک «رویداد تغییر»[۱] ایجادشده و مدل معنایی جدیدی از کد ارسال‌شده[۲] در مخزن کد ایجاد می‌شود.

[۱] Change Event

[۲] Commit

۲,۲,۳ ایجاد مخزن معنایی ریزتاکتیک

گام دوم رویکرد پیشنهادی ایجاد مخزن معنایی ریزتاکتیک است. در این مخزن مدل‌های ریزتاکتیک مربوط به هر تاکتیک معماری در قالب داده‌های معنایی ذخیره می‌شوند. آردی‌اف، مدل داده‌ی معنایی مورداستفاده در این مقاله است. شمای داده نیز با استفاده از آنتولوژی ریزتاکتیک که در بخش ۳,۱,۳ معرفی شد توصیف می‌شود. هر مدل ریزتاکتیک موجود در مخزن، اجزاء یک تاکتیک و روابط بین آن‌ها را توصیف می‌کند.

توصیف مدل‌های ریزتاکتیک نیازمند شناسایی ریزتاکتیک‌های موجود در هر تاکتیک معماری است. در این بخش با استفاده از رویکرد معرفی‌شده توسط کیم و همکاران به‌منظور توصیف تاکتیک‌های معماری، مدل‌های معنایی ریزتاکتیک ایجاد و در مخزن معنایی ذخیره می‌شوند[۱۴].

شکل ۶ مدل ریزتاکتیک مرتبط با تاکتیک «ضربان قلب»[۱] را بر اساس متامدل پیشنهادی ریزتاکتیک نمایش می‌دهد. تاکتیک ضربان قلب یکی از تاکتیک‌های مرتبط با ویژگی کیفیتی دسترس‌پذیری است. جدول ۲ نیز بخشی از سه‌گانه‌های آردی‌اف مربوط به مدل معنایی تاکتیک ضربان قلب را نمایش می‌دهد.

نکته قابل‌توجه در خصوص مخزن معنایی ریزتاکتیک امکان تکامل آن توسط آکسیوم‌ها[۲] و قوانین[۳] است. با افزودن آکسیوم‌ها و قوانین جدید به مخزن معنایی ریزتاکتیک، امکان استنتاج بر روی داده‌های موجود و به‌تبع آن تولید سه‌گانه‌های جدید آردی‌اف وجود دارد. این سه‌گانه‌های جدید امکان غنی‌تر شدن مخزن معنایی تاکتیک را در طول زمان فراهم می‌کنند.

مخزن معنایی ریزتاکتیک علاوه بر داده‌های معنایی در قالب آردی‌اف شامل پرس‌وجوهای معنایی در قالب اسپارکل است. این پرس‌وجوها بر اساس متامدل ریزتاکتیک و با استفاده از آنتولوژی ریزتاکتیک به‌منظور شناسایی اجزایی از کد منبع که مشابهت ساختاری به یک تاکتیک معماری دارند تعریف می‌شوند.

جدول ۲. بخشی از مدل معنایی مربوط به تاکتیک ضربان قلب

مدل آردی‌اف ضربان قلب		
Subject	**Predicate**	**Object**
Last_Updated_Time	type	Structural_Feature_MicroTactic
Emitter	Has_Structural_Feature	Sending_Interval
Emitter	type	Classifier_MicroTactic
Update_Time	type	Behaviroal_Feature_MicroTactic
Receiver	Has_Structural_Feature	Last_Updated_Time
Receiver	Has_Structural_Feature	Checking_Time
Receiver	Has_Beahvioral_Feature	Update_Time
Receiver	Has_Beahvioral_Feature	Check_Alive
Receiver	type	Classifier_MicroTactic

۳,۲,۳ پیشنهاد تاکتیک

در این بخش با استفاده از پرس‌وجوهای معنایی ایجادشده در بخش ۲,۲,۳، کلاس‌هایی از کد منبع که احتمال پیاده‌سازی بخشی از تاکتیک‌های معماری توسط آن‌ها وجود دارد شناسایی می‌شوند. متأسفانه برخی از کلاس‌های شناسایی‌شده در این مرحله دارای «مثبت کاذب»[۴] هستند و ممکن است بیانگر یک پیاده‌سازی از تاکتیک یا ریزتاکتیک معماری نباشند. لذا خروجی این مرحله به‌عنوان نامزدهایی[۵] برای پیاده‌سازی بخشی از تاکتیک‌های معماری مطرح است. در این مرحله یک الگوریتم برای پیشنهاد تاکتیک‌های معماری معرفی می‌شود. بر اساس این الگوریتم، هر یک از پرس‌وجوهای توصیف‌شده در بخش ۲,۲,۳ بر روی مدل معنایی کد منبع اجراشده و نتایج به‌دست‌آمده مجموعه‌ای از تاکتیک‌های نامزد را ارائه می‌نماید.

```
semSrc = Semantic Source Code;
tSRepo = Repository of SPARQL queries;
tCandidList = Tactic Candidate List = {};
foreach query tS in repository tSRepo  {
 c = execSparql(tS , semSrc);
 if (c != Null)
 {
  add c to tCandidList;
 }
}
return tCandidList;
```

شکل ۷. شبه کد الگوریتم پیشنهاد تاکتیک

شکل ۷ شبه کد الگوریتم پیشنهاد تاکتیک را نمایش می‌دهد. بر اساس این الگوریتم، پرس‌وجوهای مرتبط با هر تاکتیک معماری بر روی مخزن معنایی کد منبع اجراشده و نتایج در قالب مجموعه تاکتیک‌های نامزد به بخش بعدی که شناسایی تاکتیک است ارسال می‌شود.

۴,۲,۳ شناسایی تاکتیک

در این مرحله از بین موارد پیشنهادشده در مرحله قبل تاکتیک یا تاکتیک‌های نهایی شناسایی می‌شوند. رویکرد این بخش استفاده از تکنیک‌های مشابهت معنایی به‌منظور مقایسه‌ی تاکتیک‌های نامزد

[۱] Heartbeat
[۲] Axioms
[۳] Rules
[۴] False Positive
[۵] Candidate

شناسایی‌شده در بخش ۳،۲،۳ با سه‌گانه‌های آردی‌اف موجود در مخزن تاکتیک است.

در یک کد شیءگرا، کلاس کوچک‌ترین کپسول ماژولاری است که امکان پیاده‌سازی بخشی از یک تاکتیک معماری را دارد. ما در این بخش کلاس را در قالب یک چهارگانه به شکل زیر تعریف می‌کنیم:

$$C_i = (N_i, A_i, M_i, P_i) \quad (۲)$$

در این چهارگانه N_i بیانگر نام کلاس i، A_i مجموعه ویژگی‌های کلاس i، M_i مجموعه متدهای کلاس i و P_i مجموعه پارامترهای مربوط به متدهای کلاس i است. همچنین هر تاکتیک معماری را در قالب یک چندگانه به شکل زیر تعریف می‌کنیم:

$$T_i = (TN_i, TC_i, TSF_i, TBF_i, TP_i, TR_i) \quad (۳)$$

در این چندگانه TN_i نام تاکتیک i، TC_i مجموعه کلاس‌های تاکتیک i، TSF_i مجموعه خصیصه‌های ساختاری تاکتیک i، TBF_i مجموعه خصیصه‌های رفتاری تاکتیک i، TP_i مجموعه پارامترهای مربوط به خصیصه‌های رفتاری تاکتیک i و TR_i مجموعه روابط بین کلاس‌های تاکتیک i را نمایش می‌دهد.

ایده اصلی الگوریتم شناسایی تاکتیک بر اساس محاسبه‌ی میزان شباهت معنایی هر کلاس C_i از کد منبع با تاکتیک‌های T_i موجود در مخزن تاکتیک است. بدین منظور، شباهت معنایی المان‌های متناظر موجود در چندگانه‌های معرفی‌شده در روابط (۲) و (۳) با یکدیگر محاسبه می‌شود.

الگوریتم‌های متنوعی برای محاسبه‌ی میزان شباهت دو کلمه یا جمله معرفی‌شده است. مشابهت معنایی در این مقاله بر اساس رابطه‌ی (۴) محاسبه می‌شود[۲۲].

$$Sim(C_i, T_j) = \alpha_1 * SSim(C_i, T_j) + \alpha_2 * BSim(C_i, T_j) + \alpha_3 * CSim(C_i, T_j) \quad (۴)$$

در رابطه (۴) میزان مشابهت یک کلاس با یک تاکتیک بر اساس سه مؤلفه‌ی مشابهت ساختاری، رفتاری و کلاسی محاسبه می‌شود. این سه مؤلفه مشابهت در روابط (۵) تا (۷) تعریف‌شده است.

$$SSim(C_i, T_j) = semSim(A_i, TSF_j) \quad (۵)$$

$$BSim(C_i, T_j) = \alpha_{21} * semSim(M_i, TBF_j) + \alpha_{22} * semSim(P_i, TP_j) \quad (۶)$$

$$CSim(C_i, T_j) = semSim(C_i, TC_j) \quad (۷)$$

در رابطه (۸) نیز نحوه محاسبه‌ی $semSim$ مشاهده می‌شود.

$$semSim(X_i, Y_j) = \frac{\sum_{x_m \in X_i} \sum_{y_n \in Y_j} semSim_T(x_m, y_n)}{|X_i||Y_j|} \quad (۸)$$

در رابطه‌ی (۸) $semSim_T$ بیانگر شباهت معنایی بر اساس وردنت[۱] یا هر آنتولوژی از واژگان نظیر ویکی‌پدیا است.

شکل ۸ شبه کد مربوط به الگوریتم انتخاب تاکتیک برای کلاس c را نمایش می‌دهد.

```
۱.  c = A Class of Source Code;
۲.  tCandidList = Candidate List of Tactics;
۳.  tFinalList = Final List of Tactics = {};
٤.  foreach tactic t in tCandidList {
٥.   s = semSim(c, t);
٦.   if (s >= threshold)
۷.   {
۸.    add t to tFinalList;
۹.   }
۱۰. }
۱۱.
۱۲. if (tFinalList.length >۰)
۱۳.  return tFinalList;
۱٤. else
۱٥.  return tCandidList;
```

شکل ۸. شبه کد الگوریتم انتخاب تاکتیک

تعیین حد آستانه و وزن‌های α در الگوریتم فوق می‌تواند بر میزان دقت و یادآوری رویکرد پیشنهادی تأثیرگذار باشد. انتخاب این حد آستانه به‌صورت تجربی انجام می‌شود.

۴. مطالعه موردی

در این بخش با انجام یک مطالعه موردی نحوه عملکرد راهکار پیشنهادی موردبررسی و ارزیابی قرار می‌گیرد. در ادامه و در ابتدا مقدمات مطالعه موردی تشریح می‌شود. سپس مطالعه موردی انجام می‌شود.

۱،۴ تشریح مقدمات انجام مطالعه موردی

در این بخش مقدمات انجام مطالعه‌ی موردی شامل معرفی سنجه‌های ارزیابی، کد منبع انتخاب‌شده، تاکتیک‌های معماری موردنظر و دیتاست مورداستفاده معرفی می‌شوند.

۱،۱،۴ معرفی سنجه‌های ارزیابی

نتایج این مطالعه موردی بر اساس سه سنجه‌ی استاندارد حوزه‌ی بازیابی اطلاعات مورد ارزیابی قرار می‌گیرد.

$$Recall = \frac{|RelevantCode \cap RetrievedCode|}{|RelevantCode|} \quad (۹)$$

$$Precision = \frac{|RelevantCode \cap RetrievedCode|}{|RetrievedCode|} \quad (۱۰)$$

$$F-measure = \frac{۲* Precision * Recall}{Precision + Recall} \quad (۱۱)$$

سنجه‌ی یادآوری[۲] بیانگر نسبت تعداد کلاس‌های مرتبط با تاکتیک‌های معماری در کلاس‌های شناسایی‌شده به تعداد کل

[۱] WordNet

[۲] Recall

کلاس‌های مرتبط است. سنجه‌ی دقت[۱] بیانگر تعداد کلاس‌های مرتبط با تاکتیک‌های معماری در کلاس‌های شناسایی‌شده به تعداد کل کلاس‌های بازیابی شده است. سنجه‌ی-اف[۲] نیز یک «میانگین همساز»[۳] از دقت و یادآوری است.

۲،۱،۴ انتخاب کد منبع

در این مرحله یک پروژه نرم‌افزاری متن‌باز که قرار است رویکرد پیشنهادی برای شناسایی تاکتیک‌های معماری بر روی آن آزمایش شود انتخاب می‌شود. در این مقاله چارچوب متن‌باز «آپاچی هدوپ»[۴] برای این منظور انتخاب‌شده است. آپاچی هدوپ مجموعه‌ای متشکل از ابزارهای متن‌باز است که برای غلبه بر مشکلات مرتبط با «کلان داده‌ها»[۵] از آن استفاده می‌شود. هدوپ یک پروژه سطح بالای آپاچی است که توسط گستره وسیعی از مشارکت‌کنندگان پشتیبانی و استفاده می‌شود و از زبان برنامه‌سازی جاوا استفاده می‌کند. این پروژه به‌صورت متن‌باز و در گیت‌هاب در دسترس است[۶].

۳،۱،۴ انتخاب تاکتیک‌های معماری

مرحله بعدی در انجام این مطالعه موردی انتخاب تاکتیک‌هایی است که قرار است شناسایی آن‌ها توسط رویکرد پیشنهادی مورد ارزیابی قرار گیرد. تاکتیک‌های ضربان قلب، نقطه‌ی چک[۷]، آربَک[۸] و زمان‌بندی[۹] تاکتیک‌های انتخاب‌شده در این مقاله هستند.

جدول ۳. لیست تاکتیک‌های معماری انتخاب‌شده

ردیف	نام تاکتیک	نام ویژگی کیفیتی مرتبط
۱	ضربان قلب	دسترس‌پذیری
۲	نقطه‌ی چک	دسترس‌پذیری
۳	آربَک	امنیت
۴	زمان‌بندی	کارایی

دو تاکتیک معماری ضربان قلب و نقطه‌ی چک مرتبط با ویژگی کیفیتی دسترس‌پذیری هستند. آربَک نیز تاکتیک معماری مرتبط با ویژگی کیفیتی امنیت و زمان‌بندی مرتبط با ویژگی کیفیتی کارایی هستند. جدول ۳ تاکتیک‌های انتخاب‌شده را نمایش می‌دهد.

۴،۱،۴ دیتاست

در این بخش کدهای چارچوب هدوپ به‌منظور شناسایی تاکتیک‌های موردنظر موردبررسی دستی قرار می‌گیرد. این شناسایی با مراجعه به مقالات مرتبط با این چارچوب و بررسی کد منبع توسط نویسنده اول مقاله انجام‌شده است. جدول ۴ نتایج این بررسی را نمایش می‌دهد.

جدول ۴. نمونه‌های تاکتیک‌های معماری در آپاچی هدوپ

نام تاکتیک	نام پکیج در هدوپ	تعداد کلاس
ضربان قلب	HDFS	۲۷
نقطه‌ی چک	MapReduce, HDFS	۳۲
آربَک	HDFS, MapReduce, Security	۳۹
زمان‌بندی	Common & MapReduce	۸۸

۲،۴ انجام مطالعه موردی

در این بخش مطالعه‌ی موردی بر اساس موارد تعیین‌شده در بخش ۴-۱ انجام می‌شود. بدین منظور و در ابتدا مدل معنایی مربوط به پروژه آپاچی هدوپ ایجاد می‌شود. سپس بر اساس پرس‌وجوهای اسپارکل مربوط به تاکتیک‌های معرفی‌شده در جدول ۴، لیست کلاس‌های شناسایی به‌عنوان تاکتیک‌های نامزد بر اساس الگوریتم پیشنهاد تاکتیک محاسبه می‌شود. در انتها نیز بر اساس الگوریتم شناسایی تاکتیک لیست تاکتیک‌های شناسایی‌شده نهایی استخراج می‌شود.

۵. بحث

جدول ۵ نتایج حاصل از محاسبه‌ی سنجه‌های معرفی‌شده در بخش ۱.۱.۴ را بر روی نتایج به‌دست‌آمده از رویکرد پیشنهادی نمایش می‌دهد. جدول ۶ نیز نتایج حاصل از محاسبه‌ی این سنجه‌ها را بر روی نتایج به‌دست‌آمده از رویکرد تِد نمایش می‌دهد[۶].

در شکل ۹ نیز میزان سنجه‌ی-اف در این دو رویکرد در قالب یک نمودار نمایش داده‌شده است. مقایسه نتایج حاصل از رویکرد پیشنهادی با نتایج حاصل از تِد به‌عنوان مهم‌ترین رویکرد موجود در این حوزه بیانگر این واقعیت است که رویکرد معنایی پیشنهادشده در این مقاله نتایج قابل قبولی را ارائه نموده است.

جدول ۵. نتایج به‌دست‌آمده از رویکرد تِد

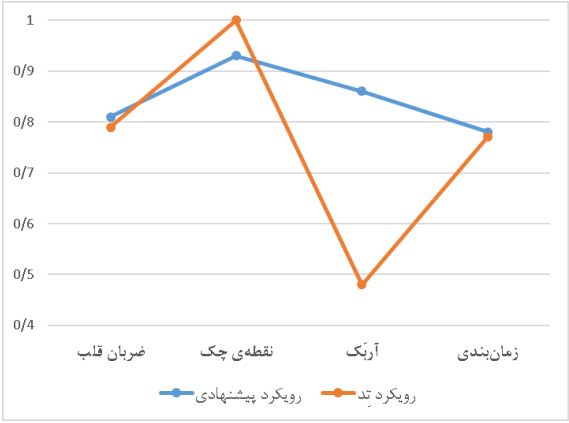
نام تاکتیک	دیدگاه تِد		
	دقت	یادآوری	سنجه‌ی‌اف
ضربان قلب	۰،۶۶	۱	۰،۷۹
نقطه‌ی چک	۱	۱	۱
آربَک	۰،۳۱	۰،۹۷	۰،۴۸
زمان‌بندی	۰،۶۵	۰،۹۴	۰،۷۷

---

[۱] Precision
[۲] F-measure
[۳] Harmonic mean
[۴] Apache Hadoop
[۵] Big Data
[۶] https://github.com/apache/Hadoop
[۷] Checkpoint
[۸] RBAC
[۹] Scheduling

جدول ۶. نتایج به‌دست‌آمده از رویکرد پیشنهادی

نام تاکتیک	دیدگاه پیشنهادی		
	دقت	یادآوری	سنجه‌ی‌اف
ضربان قلب	۰,۹۳	۰,۷۱	۰,۸۱
نقطه‌ی چک	۰,۸۸	۱	۰,۹۳
آربَک	۰,۷۷	۰,۹۷	۰,۸۶
زمان‌بندی	۰,۸۳	۰,۷۳	۰,۷۸



شکل ۹. مقایسه سنجه‌ی اف در رویکرد پیشنهادی و تِد

همان‌گونه که در شکل ۹ مشاهده می‌شود در خصوص تاکتیک‌های ضربان قلب، آربَک و زمان‌بندی، رویکرد پیشنهادی در این مقاله نتایج بهتری ارائه نموده است؛ اما در خصوص تاکتیک نقطه‌ی چک رویکرد تِد وضعیت بهتری دارد. نکته قابل‌توجه در خصوص روش تِد در این نمودار، مشاهده تغییرات شدید بین میزان سنجه‌ی-اف برای تاکتیک‌های نقطه‌ی چک و آربَک است. همان‌گونه که مشاهده می‌شود تغییر شدید در مقدار این سنجه در رویکرد پیشنهادی وجود ندارد.

درروش تِد تمرکز اصلی بر روی یادگیری واژگان استفاده‌شده در پیاده‌سازی‌های مختلف از یک تاکتیک است. لذا میزان سنجه‌ی-اف به الگوریتم یادگیری و دیتاست مربوطه وابسته است. همچنین پیاده‌سازی‌های جدید از یک تاکتیک نیازمند فرآیند یادگیری مجدد است که وجود این نقص از کارایی این روش می‌کاهد.

۶. نتیجه‌گیری

در این مقاله یک رویکرد معنایی به‌منظور شناسایی خودکار تاکتیک‌های معماری از کد منبع معرفی‌شده است. بدین منظور در ابتدا یک متامدل برای تاکتیک‌های معماری معرفی‌شده و به‌عنوان متامدل برای مدل‌های تاکتیک مورداستفاده قرارگرفته است. همچنین بر اساس این متامدل یک آنتولوژی ایجادشده که به‌عنوان شمای مدل‌های معنایی تاکتیک مورداستفاده قرارگرفته است. سپس هر تاکتیک توسط یک مدل معنایی با ساختار آردی‌اف توصیف و در مخزن ذخیره می‌شود. از طرف دیگر، کد منبع پروژه نیز به ساختار آردی‌اف تبدیل می‌شود. سپس فرآیند شناسایی تاکتیک طی دو مرحله و بر اساس شباهت معنایی و ساختاری انجام می‌شود.

نتایج حاصل از ارزیابی رویکرد پیشنهادی بیانگر کارایی قابل‌قبول این روش است. رویکرد معنایی پیشنهادی از دو جنبه‌ی مهم دارای تأثیرات مثبت است. اولین جنبه قابل‌گسترش بودن آن است. در این رویکرد امکان افزودن مدل‌های معنایی مرتبط با تاکتیک‌های معماری با یافتن پیاده‌سازی‌های جدید از هر تاکتیک به‌سادگی امکان‌پذیر است. هر پیاده‌سازی جدید از یک تاکتیک با افزودن آکسیوم‌های مربوطه به مدل معنایی قابل‌استفاده خواهند بود. جنبه مثبت دیگر رویکرد پیشنهادی عدم نیاز به فرآیند یادگیری است که یک فرآیند زمان‌بر است و نیازمند دیتاست‌های مربوط به خود است. همچنین به‌کارگیری رویکرد معنایی این امکان را فراهم می‌کند که فرآیند شناسایی یک تاکتیک به واژگان خاصی حساس نباشد.

پژوهش‌های آتی در این حوزه شامل افزودن سایر تاکتیک‌های معماری به مخزن معنایی خواهد بود. همچنین با افزودن امکان شناسایی روابط بین تاکتیک‌ها الگوریتم‌های شناسایی و انتخاب تاکتیک بهبود خواهد یافت.

۷. مراجع

[۱] M. Mirakhorli, “Preventing erosion of architectural tactics through their strategic implementation, preservation, and visualization,” in *۲۰۱۳ ۲۸th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. ۲۰۱۳, pp. ۷۶۲–۷۶۵. doi: ۱۰,۱۱۰۹/ASE.۲۰۱۳,۶۶۹۳۱۵۲.

[۲] N. B. Harrison and P. Avgeriou, “How do architecture patterns and tactics interact? A model and annotation,” *Journal of Systems and Software*, vol. ۸۳, no. ۱۰, pp. ۱۷۳۵–۱۷۵۸, Oct. ۲۰۱۰, doi: ۱۰,۱۰۱۶/j.jss.۲۰۱۰,۰۴,۰۶۷.

[۳] M. Mirakhorli, A. Fakhry, A. Grechko, M. Wieloch, and J. Cleland-Huang, “Archie: A Tool for Detecting, Monitoring, and Preserving Architecturally Significant Code,” in *Proceedings of the ۲۲Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, New York, NY, USA, ۲۰۱۴, pp. ۷۳۹–۷۴۲. doi: ۱۰,۱۱۴۵/۲۶۳۵۸۶۸,۲۶۶۱۶۷۱.

[۴] J. Ryoo, P. Laplante, and R. Kazman, “A Methodology for Mining Security Tactics from Security Patterns,” in *۲۰۱۰ ۴۳rd Hawaii International Conference on System Sciences*, Jan. ۲۰۱۰, pp. ۱–۵. doi: ۱۰,۱۱۰۹/HICSS.۲۰۱۰,۱۸.

[۵] M. Mirakhorli, P. Mäder, and J. Cleland-Huang, “Variability points and design pattern usage in architectural tactics,” in *Proceedings of the ACM SIGSOFT ۲۰th International Symposium on the Foundations of Software Engineering*,

[۱۴] S. Kim, D.-K. Kim, L. Lu, and S. Park, "Quality-driven architecture development using architectural tactics," *Journal of Systems and Software*, vol. ۸۲, no. ۸, pp. ۱۲۱۱–۱۲۳۱, Aug. ۲۰۰۹, doi: ۱۰,۱۰۱۶/j.jss.۲۰۰۹,۰۳,۱۰۲.

[۱۵] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, ۳rd edition. Upper Saddle River, NJ: Addison-Wesley Professional, ۲۰۱۲.

[۱۶] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data & Knowledge Engineering*, vol. ۲۵, no. ۱, pp. ۱۶۱–۱۹۷, Mar. ۱۹۹۸, doi: ۱۰,۱۰۱۶/S۰۱۶۹-۰۲۳X(۹۷)۰۰۰۵۶-۶.

[۱۷] A. Barforoush and A. Rahnama, "Ontology Learning: Revisited," *Journal of Web Engineering (JWE)*, vol. ۱۱, pp. ۲۶۹–۲۸۹, Dec. ۲۰۱۲.

[۱۸] N. Noy and D. Mcguinness, "Ontology Development ۱۰۱: A Guide to Creating Your First Ontology," *Knowledge Systems Laboratory*, vol. ۳۲, Jan. ۲۰۰۱.

[۱۹] S. Ehsan and D. Mahmood, "AN APPLICABLE METHOD FOR COMPREHENSIVE CORPUS GATHERING NEEDS FOR FUZZY DOMAIN ONTOLOGY GENERATION," *ELECTRONIC INDUSTRIES*, vol. ۷, no. ۱, pp. ۸۹–۱۰۳, Jan. ۲۰۱۶.

[۲۰] R. Y. K. Lau, D. Song, Y. Li, T. C. H. Cheung, and J.-X. Hao, "Toward a Fuzzy Domain Ontology Extraction Method for Adaptive e-Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. ۲۱, no. ۶, pp. ۸۰۰–۸۱۳, Jun. ۲۰۰۹, doi: ۱۰,۱۱۰۹/TKDE.۲۰۰۸,۱۳۷.

[۲۱] M. Atzeni and M. Atzori, "CodeOntology: RDF-ization of Source Code," in *The Semantic Web – ISWC ۲۰۱۷*, Cham, ۲۰۱۷, pp. ۲۰–۲۸. doi: ۱۰,۱۰۰۷/۹۷۸-۳-۳۱۹-۶۸۲۰۴-۴_۲.

[۲۲] S. Paydar and M. Kahani, "A semantic web enabled approach to reuse functional requirements models in web engineering," *Autom Softw Eng*, vol. ۲۲, no. ۲, pp. ۲۴۱–۲۸۸, Jun. ۲۰۱۵, doi: ۱۰,۱۰۰۷/s۱۰۵۱۵-۰۱۴-۰۱۴۴-۴.

New York, NY, USA, Nov. ۲۰۱۲, pp. ۱–۱۱. doi: ۱۰,۱۱۴۵/۲۳۹۳۵۹۶,۲۳۹۳۶۵۷.

[۶] M. Mirakhorli and J. Cleland-Huang, "Detecting, Tracing, and Monitoring Architectural Tactics in Code," *IEEE Transactions on Software Engineering*, vol. ۴۲, no. ۳, pp. ۲۰۵–۲۲۰, Mar. ۲۰۱۶, doi: ۱۰,۱۱۰۹/TSE.۲۰۱۵,۲۴۷۹۲۱۷.

[۷] D. E. Krutz and M. Mirakhorl, "Architectural clones: toward tactical code reuse," in *Proceedings of the ۳۱st Annual ACM Symposium on Applied Computing*, New York, NY, USA, Apr. ۲۰۱۶, pp. ۱۴۸۰–۱۴۸۵. doi: ۱۰,۱۱۴۵/۲۸۵۱۶۱۳,۲۸۵۱۷۸۷.

[۸] I. J. Mujhid, J. C. S. Santos, R. Gopalakrishnan, and M. Mirakhorli, "A search engine for finding and reusing architecturally significant code," *Journal of Systems and Software*, vol. ۱۳۰, pp. ۸۱–۹۳, Aug. ۲۰۱۷, doi: ۱۰,۱۰۱۶/j.jss.۲۰۱۶,۱۱,۰۳۴.

[۹] J. Keim, A. Kaplan, A. Koziolek, and M. Mirakhorli, "Does BERT Understand Code? – An Exploratory Study on the Detection of Architectural Tactics in Code," in *Software Architecture*, Cham, ۲۰۲۰, pp. ۲۲۰–۲۲۸. doi: ۱۰,۱۰۰۷/۹۷۸-۳-۰۳۰-۵۸۹۲۳-۳_۱۵.

[۱۰] B. Milhem and H. A. Ismail, "Evaluating Software Architecture Based on Their Implemented Patterns and Tactics," Thesis, Université d'Ottawa / University of Ottawa, ۲۰۲۰. doi: ۱۰,۲۰۳۸۱/ruor-۲۵۰۵۷.

[۱۱] M. Zanoni, F. Arcelli Fontana, and F. Stella, "On applying machine learning techniques for design pattern detection," *Journal of Systems and Software*, vol. ۱۰۳, no. Supplement C, pp. ۱۰۲–۱۱۷, May ۲۰۱۵, doi: ۱۰,۱۰۱۶/j.jss.۲۰۱۵,۰۱,۰۳۷.

[۱۲] J. (Yossi) Gil and I. Maman, "Micro Patterns in Java Code," in *Proceedings of the ۲۰th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, New York, NY, USA, ۲۰۰۵, pp. ۹۷–۱۱۶. doi: ۱۰,۱۱۴۵/۱۰۹۴۸۱۱,۱۰۹۴۸۱۹.

[۱۳] S. Maggioni, "Design pattern detection and software architecture reconstruction: an integrated approach based on software micro-structures," Università degli Studi di Milano-Bicocca, ۲۰۱۰.