گسترش ابزارهای خودکار شناسایی الگوهای طراحی با عملگر تصحیح برچسب

زینب اسمعیل پور[۱]، دانش آموخته کارشناسی ارشد، اشکان سامی[۲]، استادیار

۱- دانشکده مهندسی برق و کامپیوتر - دانشگاه شیراز - شیراز - ایران - esmailpoor1991@gmail.com

۲- دانشکده مهندسی برق و کامپیوتر - دانشگاه شیراز - شیراز - ایران - sami@shirazu.ac.ir

چکیده: الگوهای طراحی، راه‌حل‌های اثبات‌شده و قابل‌اطمینانی هستند که برای پاسخ به برخی از مسائل با رخداد مکرر در طراحی نرم‌افزار شیءگرا، ارائه شده‌اند. شناسایی آن‌ها در کد، به‌منزله بازیابی طرح و هدف طراح و سهولت در امر نگهداشت‌پذیری است. ازآنجاکه سهولت در نگهداشت‌پذیری سیستم بسیار مهم و اجتناب‌ناپذیر است، لذا تولید ابزارهای خودکار برای شناسایی الگوها، موردتوجه محققین قرارگرفته است. اکثر ابزارهای شناسایی کنونی درصد بازیابی بالایی دارند. اما در شناسایی الگوها، خصوصاً با ساختار و عملکرد مشابه، مثبت کاذب بالایی تولید می‌کنند. لذا تحقیقاتی در پالایش خروجی ابزارها، برای حذف مثبت‌های کاذب نیز شروع شده است. در این مقاله برای اولین بار یک روش «تصحیح برچسب» ارائه شده است. روش فوق، ابتدا مثبت‌های کاذب را شناسایی و سپس هویت صحیح آن‌ها را به کمک یک مجموعه معیارهای جدید تعیین می‌کند. خودکارسازی روش با داده‌کاوی است و نتایج حاصل، با دقت یادگیری ۹۸/۲٪ در دسته‌بندی «چندبرچسبه»، با مد ۹۸/۲٪ در دسته‌بندی «یکی در مقابل همه» و مد ۱۰۰٪ در دسته‌بندی «دوبه‌دو» خروجی ابزارها را تصحیح می‌کند.

واژه‌های کلیدی: الگوهای طراحی، ساختار مشابه، عملکرد مشابه، معیارها، داده‌کاوی، تصحیح برچسب.

Extending Automatic Design Pattern Detection Tools by Label Correcting Operation

Z. Esmailpour[1], A. Sami[2]

1, 2- Faculty of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran

Abstract: Design patterns (DP) are reliable and stable solutions to frequently occurring problems in object oriented software design. Detecting design patterns can reveal design purposes in code and facilitates maintainance. Thus automatic design pattern detection is an active area or research. Majority of automatic design pattern detection tools have high recall. However they suffer from high false positive rate since some of the design patterns are structureally and/or behaviorally similar. Hence another research direction is filtering the results of other automatic DP detection tools. In this paper, correcting the labels of these false positives is presented for the first time. Several metrics have been proposed that facilitates this process. Automatic label correcting is performed by data mining techniques. Results achieved with this process are as follows: accuracy about 98.2% in "multi-class classifications", 98.2% mode in "one versus all" classification and 100% mode in "pairwise" classification.

Keywords: Design patterns, similar structure, similar behavior, metrics, data mining, label correcting.

تاریخ ارسال مقاله: ۹۲/۰۹/۱۹

تاریخ اصلاح مقاله: ۹۲/۱۲/۰۴، ۹۳/۰۱/۳۰ و ۹۳/۰۵/۲۴

تاریخ پذیرش مقاله: ۹۳/۰۶/۲۵

نام نویسنده مسئول: اشکان سامی

نشانی نویسنده مسئول: شیراز - خیابان ملاصدرا - دانشگاه شیراز - دانشکده مهندسی برق و کامپیوتر

۱- مقدمه

طراحی یک نرم‌افزار شیءگرا دشواری‌های خاص خود را دارد. دشوارتر از آن طراحی یک نرم‌افزار شیءگرا با قابلیت استفاده مجدد است. الگوهای طراحی راه‌حل‌های اثبات‌شده و قابل‌اطمینانی هستند که به‌منظور حل برخی از مسائل که به‌طور مکرر در طراحی یک نرم‌افزار شیءگرا رخ می‌دهند، ارائه شده‌اند [۱]. الگوهای طراحی به نحوی پیکربندی شده‌اند که به‌کارگیری صحیح آن‌ها در توسعه یک نرم‌افزار شیءگرا، به‌طور چشم‌گیری کیفیت کد را ازنظر نگهداشت‌پذیری و قابلیت استفاده مجدد بهبود می‌دهد. هر الگو طراحی، هدف واحد و ساختار مشخص خودش را دارد. بنابراین با شناسایی الگوهای طراحی از کد یا طراحی، هدف و طرح نرم‌افزار بازیابی می‌شود [۲].

به‌طورکلی الگوهای طراحی به سه دسته ساختاری[۱]، ایجادکننده[۲]، و رفتاری[۳] تقسیم می‌شوند. دسته اول مسائل مربوط به ساختار (مسائل مربوط به نحوه برقراری صحیح ارتباط میان کلاس‌ها و اشیاء)، دسته دوم مسائل مربوط به نحوه تولید اشیاء و دسته سوم برخی از مسائل رفتاری (نحوه صحیح پاسخ به رویدادهای زمان اجرا) را حل می‌کنند.

مهم‌ترین مسئله نگهداشت‌پذیری سیستم‌های نرم‌افزاری، خصوصاً سیستم‌های قدیمی این است که فاقد سند کامل از طرح و اهداف سیستم هستند. لذا با توجه به خصوصیات ذکرشده از الگوهای طراحی، شناسایی و بازیابی خودکار یا نیمه‌خودکار آن‌ها از سیستم، سندسازی، نگهداشت‌پذیری، و قابلیت استفاده مجدد سیستم را تسهیل می‌کند. در زمینه شناسایی الگوهای طراحی، روش‌های خودکار و نیمه‌خودکار متعددی توسط محققان ارائه شده است. اما هیچ‌کدام نتوانسته‌اند یک خروجی مطمئن و بدون مثبت کاذب در اختیار توسعه‌دهندگان قرار دهند. به‌طورکلی شیوه‌های شناسایی الگوهای طراحی به دودسته تقسیم می‌شوند، آن‌هایی که بر اساس جنبه‌های ساختاری الگوها، شناسایی را انجام می‌دهند و آن‌هایی که از جنبه‌های رفتاری موجود در الگوها نیز جهت شناسایی بهره می‌گیرند [۲].

۱-۱- هدف قرار دادن جنبه‌های ساختاری

برخی از شیوه‌ها، برای شناسایی الگوها، تنها جنبه ساختاری آن‌ها را در نظر می‌گیرند. در این شیوه‌ها، ابتدا خصوصیات ساختاری هر کلاس موجود در کد منبع با خصوصیات هر کلاس تشکیل‌دهنده یک الگو مقایسه، و کاندیدهای هر جزء تشکیل‌دهنده یک الگو شناسایی می‌شود. سپس کاندیدهای اجزایی که می‌توانند به هم مرتبط شوند، ترکیب می‌شوند. درنهایت روابط بین کلاسی اجزاء مرتبط شده را بدون توجه به خصوصیات رفتاری، تجزیه‌وتحلیل، و با روابط بین اجزاء تشکیل‌دهنده الگوها مقایسه می‌کنند. روابط بین کلاسی شامل ارث‌بری، تعمیم[۴]، پیوند[۵] و ... هستند. به‌عنوان‌مثال SPOOL [۳]، DP++ [۴]، Osprey [۵] و [۶] به شیوه ساختاری فوق، الگوها را شناسایی کردند.

از موفق‌ترین ابزارهای کشف الگوهای طراحی با روش مقایسه ساختار، SSA[۶] است. درهرحال، برای بهبود نرخ شناسایی و کاهش مثبت‌های کاذب، علاوه بر خصوصیات ساختاری، تجزیه‌وتحلیل خصوصیات رفتاری نیز موردبررسی قرار گرفت.

۱-۲- هدف قرار دادن جنبه‌های رفتاری

شیوه‌های بحث‌شده در بخش قبل قادر به تشخیص الگوهایی که ازنظر ساختاری یکسان، اما رفتار متفاوتی دارند نظیر الگوی «استراتژی[۷]» در مقابل الگوی «وضعیت[۸]» نیستند. همچنین برخی از الگوها نیز نقاط رفتاری مشابهی دارند، که شیوه‌هایی که تنها بر ساختار تأکید دارند، نمی‌توانند آن‌ها را شناسایی کنند. روش‌هایی که بررسی جنبه‌های رفتاری را هدف قرار می‌دهند، سعی می‌کنند این مسئله را با استفاده از تحلیل پویا، تحلیل ایستا، یا ترکیب این دو حل کنند. روش‌های مختلف ایستا و پویا، در راستای شناسایی رفتارهای خاص الگوهای طراحی که منجر به تشخیص بهتر آن‌ها شوند به کار گرفته شده است. همان‌طور که مطرح شد الگوهایی وجود دارند که ساختاری مشابه ولی رفتاری متفاوت و یا برخی رفتارهای مشابهی دارند، که تنها جستجوی رفتارهای متمایزکننده می‌تواند آن‌ها را از هم جدا کند.

شیوه‌های تجزیه‌وتحلیل پویا سعی داشته‌اند از داده‌های ثبت‌شده‌ی مربوط به زمان اجرای برنامه، برای تشخیص جنبه‌های رفتاری الگوها استفاده کنند. تجزیه‌وتحلیل پویا به پوشش خوبی از داده آزمایش به‌منظور شناسایی هر مسیر اجرایی ممکن، نیازمند است. داده آزمایشی اغلب محدود بوده و حتی اگر داده آزمایش موجود باشد، نتایج زمان اجرا ممکن است هیچ کمک خاصی در شناسایی رفتارهای خاص الگوها نکند، چون این داده‌ها باهدف سنجش الگوها ثبت نشده‌اند. به دلیل نکات بیان‌شده روش‌های پویا در شناسایی خودکار الگوهای طراحی موفقیت خاصی را به دنبال نداشته‌اند [۲]. برای مثال، [۷] تنها از تجزیه‌وتحلیل پویا برای تشخیص الگوی «زنجیره‌ی مسئولیت‌ها[۹]» استفاده کرد اما دقیقاً به دلیل دو مشکل بیان‌شده نتیجه‌اش موفقیت‌آمیز نبود.

روش‌های مناسب‌تر و موجود شناسایی رفتاری در تشخیص الگو غالباً ایستا می‌باشند. تجزیه‌وتحلیل ایستای رفتار، در کد برنامه به جستجوی مشخصه‌هایی می‌پردازد که شاخص آن الگو است. بنابراین موفقیت روش‌های ایستا مبتنی بر مشخص کردن رفتارهایی هستند که دقت تشخیص را افزایش دهند. این مقاله دلیل اصلی مثبت‌های کاذب را در عدم جستجوی خصوصیت رفتاری مناسب می‌داند. با افزایش مشخصه‌های متمایزکننده، می‌توان دقت نتایج خروجی ابزارها را بهتر کرد.

در این کار، ابزارهای خودکار شناسایی الگوهای طراحی با دو عمل «پالایش» و «تصحیح برچسب» گسترش می‌یابند. عمل پالایش، مثبت‌های کاذب تولیدشده در خروجی ابزارهای خودکار را شناسایی می‌کند. درحالی‌که تصحیح برچسب، هویت صحیح نمونه مثبت کاذب

را (برحسب اینکه مثبت کاذب، به دلیل شباهت با کدام الگو مثبت کاذب شده است) با استفاده از معیارهای جدید تعریف‌شده در این کار و روش‌های داده‌کاوی شناسایی می‌کند. به‌طور مثال اگر نمونه شناسایی‌شده توسط ابزارهای خودکار، با برچسب «استراتژی» شناسایی شده است اما در حقیقت این نمونه استراتژی نبوده، با عملگر "تصحیح برچسب"، ابتدا به‌عنوان مثبت کاذب شناخته‌شده (پالایش)، سپس برچسب صحیح آن (برحسب مقادیر معیارها) تشخیص داده می‌شود.

به‌عنوان‌مثال برچسب صحیح آن ممکن است الگوی «وضعیت» باشد. بنابراین برچسب استراتژی به وضعیت تغییر می‌کند. با استفاده از این معیارها و روش‌های داده‌کاوی علاوه بر نمونه‌های مثبت کاذب، نمونه‌های منفی کاذب نیز در برخی موقعیت‌ها کاهش می‌یابند. چون وقتی یک برچسب تصحیح می‌شود مثلاً از استراتژی به وضعیت، ممکن است این نمونه وضعیت جزء نمونه‌های شناسایی‌شده برای الگوی وضعیت توسط خود ابزار نباشد، بنابراین این نمونه وضعیت یک منفی کاذب بوده که ما با شناسایی آن توسط عملگر تصحیح برچسب، یک منفی کاذب را حذف کرده‌ایم.

با توجه به این‌که ابزارها و روش‌های پیشین، بیش‌ترین مثبت کاذب را در شناسایی الگوهای با ساختار و رفتار مشابه تولید می‌کنند، در این کار، سعی شده تا خروجی ابزارها با توانایی شناسایی چنین الگوهایی تصحیح گردد. لذا، تصحیح برچسب روی نمونه‌های استخراج‌شده الگوی استراتژی (الگو استراتژی به دلیل داشتن شباهت‌های ساختاری و رفتاری با الگوهای دیگر، طبق خروجی ابزارها، بیش‌ترین مثبت کاذب را دارد) انجام می‌شود. ابتدا بر اساس روش‌های عرف داده‌کاوی، با نمونه‌های مثبت کاذب و مثبت صحیح الگوی استراتژی استخراج‌شده توسط ابزارها، یک مجموعه داده تهیه شده است. سپس بر اساس مستندات موجود و همچنین بازبینی دستی، به‌منظور پیش‌بینی هویت صحیح هر نمونه، دو ستون در مجموعه داده تعیین شده است. یک ستون با «درست» و «نادرست» (درست، در صورت شناسایی صحیح توسط ابزارها و نادرست در صورت مثبت کاذب بودن) برچسب می‌خورد، و ستون دیگر با نام الگو صحیح آن نمونه یا در صورت ناشناس بودن، با «بدون الگو» برچسب می‌گیرد. سپس مقادیر معیارها یا پیش‌بینی کننده‌های استخراج‌شده در این کار، روی هر نمونه محاسبه می‌شوند و نهایتاً مجموعه داده در اختیار الگوریتم‌های داده‌کاوی جهت مدل‌سازی قرار می‌گیرند. در مدل‌سازی سعی می‌شود دانش موجود در داده‌ها، در قالب یک سری قوانین استخراج شوند. این قوانین برای شناسایی نمونه‌های ناشناخته (جدید) در مجموعه داده مورد استفاده قرار می‌گیرند.

آزمایش‌ها با استفاده از روش‌های طبقه‌بند، C5.0، J48، Boosting، Svm، وAdaBoostM1 انجام شده است. موجودیت‌های مجموعه داده، از خروجی ابزارها در شناسایی الگوها روی سه نرم‌افزار متن‌باز Jhotdraw [۸]، Jrefactory [۹] و Javaio [۱۰] به دست آمده‌اند.

این پروژه‌ها به علت‌های زیر انتخاب شده‌اند:

۱- آن‌ها حاوی تعداد زیادی از الگوهای طراحی معرفی‌شده در کتاب گاما [۱] هستند. بنابراین در ارزیابی ابزارها و بهبودهای انجام‌شده در شناسایی الگوهای طراحی بسیار مؤثر واقع می‌شوند.

۲- الگوهای طراحی موجود در این پروژه‌ها، توسط توسعه‌دهندگانشان سند شده‌اند. به این طریق ارزیابی نتایج تولیدشده توسط شیوه‌های ارائه‌شده را امکان‌پذیر می‌کنند.

۳- آن‌ها پروژه‌های متن‌باز هستند که به همراه کد منبعشان در دسترس عموم هستند [۱۱].

در این پروژه‌ها، برخی نمونه‌های الگوها وجود دارند که از ساختار پایه که برای آن‌ها در کتاب‌ها معرفی شده است، بسیار فاصله گرفته‌اند، بنابراین شناسایی چنین الگوهایی، دشواری‌های خاص خودش را دارد. به همین دلیل در اکثر تحقیق‌های شناسایی الگوهای طراحی به‌عنوان محک مورد استفاده قرار می‌گیرند.

در این کار عملیات تصحیح برچسب و پالایش، روی کاندیدهای الگوی طراحی «استراتژی» یافته توسط ابزارهای [۱۱] SSA و [۲] PINOT استفاده شده‌اند.

این دو ابزار هدف آزمایش قرار گرفته‌اند چون، این دو ابزار خصوصاً SSA سرعت بالایی دارد، در دسترس عموم هستند، به‌راحتی قابل‌استفاده هستند، تعداد زیادی از الگوها را شناسایی می‌کنند، درصد بازیابی بالایی[۱۰] دارند، و در شناسایی الگوها به‌جز آن‌هایی که ساختار و عملکرد مشابه دارند، دقت خوبی دارند.

به‌طورکلی غیر از الگوهایی که از پایه ساختار یکسانی دارند، بقیه الگوها نیز به دلیل انعطاف، ، وقتی از ساختار پایه دور می‌شوند، ساختار مشابه به برخی الگوهای دیگر پیدا می‌کنند. همچنین برخی از الگوها عملکرد مشابه به هم دارند. به‌طوری‌که در خیلی از ابزارها دیده شده است الگوی استراتژی با الگوی وضعیت، ملاقات‌کننده، کارخانه انتزاعی [۱۱]، و الگوی وضعیت با الگوی نماینده[۱۱] [۱۱]، و ... اشتباه گرفته شده‌اند. درواقع معیارهای استخراج‌شده در این کار با استفاده از شیوه‌های داده‌کاوی، برخی از نقص‌ها و کمبودهای دیده‌نشده در ابزارها را رفع می‌کنند.

در ادامه این بخش، به‌طور خلاصه بر برخی از تحقیقاتی که ازنظر هدف و یا شیوه کار شباهت بیش‌تری به تحقیق ما دارند مروری خواهیم داشت. بقیه این مقاله به بخش‌های زیر سازمان‌دهی می‌شود. در بخش سوم الگوهای موردمطالعه در این تحقیق و معیارهای شناسایی شرح داده می‌شوند. در بخش چهارم نگاهی بر شیوه‌های داده‌کاوی و مجموعه داده ایجادشده خواهیم داشت. در بخش پنجم نتایج ارائه خواهند شد و نهایتاً در بخش ششم نتیجه‌گیری و کارهای آتی پیشنهاد می‌شوند.

۲- کارهای مرتبط

تحقیقاتی که در این بخش بحث می‌شوند، به سه دسته قابل‌تقسیم هستند:

۱- آن‌هایی که به بررسی خصیصه‌های ساختاری تأکید دارند و جهت شناسایی الگوها با این روش، ابزار ارائه داده‌اند.

۲- آن‌هایی که بررسی خصیصه‌های رفتاری را در کنار خصوصیات ساختاری ضروری دانسته و ابزار شناسایی ارائه داده‌اند.

۳- دسته سوم که مورد تأکید این مقاله است، آن دسته از تحقیقاتی هستند که سعی می‌کنند خروجی بهترین ابزارهای موجود را بهبود بخشند.

۲-۱- شناسایی الگوها با جستجو خصیصه‌های ساختاری

این شیوه‌ها، سعی دارند خصیصه‌های ساختاری هر الگوی طراحی را با اطلاعات ساختاری بیرون کشیده‌شده از کد، به شیوه‌های متفاوت پرس‌وجو نویسی [۱۲]، مقایسه ماتریس‌های حاوی اطلاعات ساختاری از زیرگراف‌های سیستم [۱۱] و ... مقایسه کنند. یکی از اولین کارها برای خودکارسازی شناسایی الگوهای طراحی در سال ۱۹۹۶ [۱۲] ارائه شد. ابزار تولیدشده توسط [۱۲] پت نام دارد. پت، تنها الگوهای ساختاری را با پرس‌وجوی خصیصه‌های ساختاری شناسایی می‌کند. در [۱۲]، پالایش روی نتایج، برای تشخیص مثبت‌های کاذب به‌صورت دستی انجام گرفت. ابزار ارائه‌شده دارای ۴۰ درصد دقت است.

در سال ۱۹۹۸ [۱۳] یک روش با سه گام برای شناسایی الگوها ارائه داده شد. در گام اول، فضای جستجو را از طریق مقایسه مقادیر یک سری معیارهای (باهدف اندازه‌گیری ساختار) عام شیءگرا، نظیر شمارش تعداد صفات، تعریف‌ها و ... برای هر کلاس موجود در کد و هر نقش الگوی مورد جستجو تا حد زیادی کاهش دادند (از طریق حذف کلاس‌هایی که کاندید نشدند). در گام دوم نزدیک‌ترین مسیر بین کاندیدهایی که می‌توانند به هم مرتبط شوند، شناسایی و سپس هر ترکیب به‌دست آمده از این مرحله، به‌عنوان کاندیدهای الگوی مورد جستجو شناخته شد. در گام سوم، به دلیل وجود مثبت کاذب بسیار زیاد در مرحله دوم، یک محدودیت رفتاری برای حذف کاندیدهای نادرست، مورد بررسی قرار گرفت. بااین‌حال، روش [۱۳]، میزان مثبت کاذب بالایی دارد. عیب اصلی [۱۳]، در محوریت اصلی کارشان، استفاده از معیارهای عام شیءگرایی بود که از پایه باهدف سنجش الگوها استخراج نشده بودند.

در سال ۲۰۰۶ [۱۱] ابزاری به نام SSA ارائه شد. در SSA از یک الگوریتم نمره دهی که نمره مشابهت اطلاعات ساختار هر زیرگراف سیستم را با هر گراف الگو مورد جستجو محاسبه می‌کرد، استفاده شد. چون الگوریتم SSA خواص انتقال را در ارث‌بری و واگذاری مسئولیت[۱۲] در نظر می‌گیرد، تنوع پیاده‌سازی‌های یک الگو را که از ساختار پایه خود فاصله گرفته‌اند، نیز شناسایی می‌کند. SSA، درصد بازیابی بالایی دارد، تعداد زیادی از الگوها را شناسایی می‌کند، به‌راحتی قابل‌استفاده است، و همچنین سرعت بالایی دارد. اما به دلیل این‌که الگوریتم SSA تنها بر قواعد ساختاری تأکید دارد، الگوهایی که ساختار یکسانی دارند و تنها در رفتار متفاوت می‌شوند، و یا آن‌هایی که رفتار مشابهی دارند، توسط SSA قابل متمایز شدن نیستند و SSA آن‌ها را در یک گروه شناسایی می‌کند (مثل «وضعیت/ استراتژی»).

در سال ۲۰۱۳ [۱۴]، از یک روش مقایسه گراف برای شناسایی الگوهای طراحی استفاده کرد. در [۱۴]، از تئوری چندریختی زیرگراف استفاده شد. درواقع مفهوم عام مقایسه گراف مشخص می‌کند که آیا دو گراف یکسان هستند یا زیرگرافی از یکی در دیگری موجود است یا خیر. در [۱۴]، برای شناسایی الگوها سه حالت مقایسه ارائه شده است حالت اول اینکه، اگر گراف الگو با گراف مدل سیستم هم‌ریخت باشد، در این صورت الگو موجود است، حالت دوم، اگر زیرگراف هم‌ریخت از الگو در گراف مدل موجود باشد، در این حالت، به‌صورت تقریبی الگو موجود است و حالت سوم الگو زیرگراف هم‌ریخت از گراف مدل نیست که در چنین حالتی الگو موجود نیست. در [۱۴]، نتایجی ارائه نشده است، اما به‌طورکلی همان‌طور که در [۱۱] نیز ذکر شده است، استفاده از روش‌های مقایسه هم‌ریخت‌های زیرگراف یا گراف یک الگو نمی‌توانند روش مناسبی در شناسایی الگوها باشند، چون اول اینکه الگوها انعطاف‌پذیری بالایی دارند و به‌ندرت در پیاده‌سازی، معادل با ساختار پایه باقی می‌مانند. دوم، مسئله پیدا کردن کل گراف یا زیرگراف‌های هم‌ریخت یک گراف یک مسئله NP-Complete است و ممکن است هیچ هم‌ریختی پیدا نشود.

۲-۲- شناسایی الگوها با جستجوی خصیصه‌های رفتاری در کنار خصوصیات ساختاری

با توجه به عدم توانایی روش‌های ساختاری در شناسایی الگوهای با ساختار و رفتار مشابه، شیوه‌هایی ارائه شدند که در کنار خصیصه‌های ساختاری، ضرورت بررسی خصیصه‌های رفتاری مناسب را نیز محور توجه قرار دادند.

در سال ۲۰۰۹ [۲] ابزاری به نام PINOT[۱۳] ارائه داده شد. در [۲] سعی کردند برای شناسایی الگوها، تعاریف عملی مربوط به هدف هر الگو را، به‌عنوان محدودیت‌های اجباری (رفتاری و ساختاری)، با آنالیز ایستا، از کد پیاده‌سازی آن الگو جستجو کنند. PINOT، درصد بازیابی بالایی دارد، تعداد زیادی از الگوها را شناسایی می‌کند، سرعت کم‌تری نسبت به SSA دارد، و توانایی ضعیفی در متمایز کردن الگوها با ساختار و عملکرد مشابه را دارد. چون در بیرون کشیدن تعاریف عملی از اهداف الگوها، تنوع کاملی از پیاده‌سازی‌ها و انعطاف‌های آن‌ها را ارائه ندادند و تنها بر اساس ساختار پایه الگوها عمل کردند.

در سال ۲۰۱۲ [۱۵] ابزاری با نام DPJF[۱۴] ارائه شد. در [۱۵]، برای برخی از الگوها، تعدادی محدودیت یا الزام رفتاری و ساختاری جهت شناسایی‌شان، ارائه داده شد. برخی از محدودیت‌های [۱۵]، ثبات و قدرت متمایزکنندگی بالایی دارند. ابزار DPJF ادعا می‌کند، صد

درصد دقت دارد. اما ابزار DPJF تنها الگوی یگانه[۱۵]را با چنین دقتی شناسایی می‌کند. مثلاً در مورد دقت شناسایی الگوی نماینده، برخی نمونه‌های شناسایی‌شده توسط این ابزار بعد از بررسی و ارتباطات با این گروه مشخص شد که الگوی وضعیت هستند و نه الگوی نماینده. همچنین این ابزار هنوز برای شناسایی الگوهای با ساختار و عملکرد مشابه مثل استراتژی و وضعیت پیاده‌سازی نشده است.

در سال ۲۰۱۳ [۱۶]، از گام شناسایی کاندیدهای هر نقش در یک الگو، تا مرحله شناسایی کاندیدهای هر الگو (ترکیب صحیح نقش‌ها) از تعدادی معیار سطح کلاس عام (ساختاری و رفتاری) و داده‌کاوی بهره گرفتند. در [۱۶]، برای تعیین برچسب «صحیح» یا «ناصحیح» کاندیدهای الگوها قبل از مدل‌سازی روی کاندیدها توسط الگوریتم‌های داده‌کاوی، از چهار ابزار خودکار شناسایی الگوهای طراحی استفاده کردند. به‌طوری‌که اگر حداقل دو ابزار در رابطه با یک کاندیدا رأی مثبتی داشتند، آن کاندید به‌عنوان نمونه صحیح در نظر گرفته می‌شد. با توجه به اینکه در داده‌کاوی مهم‌ترین عناصر، صفات توصیف‌کننده موجودیت‌ها (معیارها) و صحت برچسب‌زنی قبل از مدل‌سازی روی مجموعه داده است، ضعف کار [۱۶] در درجه اول، در استفاده از تعداد زیاد معیارهایی است که باهدف سنجش الگوها استخراج نشده بودند، و دوم برای برچسب‌زنی (تعیین صحت یا عدم صحت کاندیدها) اولیه مجموعه داده، استفاده از ابزارهای موجود شناسایی الگوهای طراحی است، که دقت کاملی ندارند و بدیهی است که در فرایند برچسب‌زنی دچار اشتباه شوند. روش [۱۶] با دقت نسبتاً پایینی متوسط «۴۹ ٪» روبرو شد.

در سال ۲۰۱۳ [۱۷]، یک روش پرس‌وجوی الگوهای طراحی از پایگاه داده ارائه کرد. محور اصلی [۱۷]، به‌کارگیری شیوه‌ی اصولی برای سست کردن پرس‌وجوها بود، پرس‌وجوها به دودسته تقسیم می‌شدند، خاصوعام. خاص یعنی تبدیل خصوصیات الگو به یک پرس‌وجوی سخت برای شناسایی الگوهایی که از ساختار پایه فاصله نگرفته‌اند و عام، تبدیل خصوصیات الگوها به یک پرس‌وجوی عام‌تر (با بهره‌گیری از تنوع پیاده‌سازی یک الگو) برای نمونه‌هایی که از ساختار پایه فاصله گرفته‌اند، به‌طوری‌که بتوان تنوع نمونه‌های قابل‌قبول الگوهای طراحی را پیدا کرد. نتایج این کار تنها برای الگوی «یگانه» ارائه شده است. مقایسه نتایج با دو ابزار PINOT و DPJF نشان‌دهنده بهبود کارشان نسبت به آن دو روش در تشخیص الگوی یگانه است. اما نتایج ابزار SSA که در کارشان آورده نشده است از هر سه بهتر است.

۲-۳- پالایش ابزارهای خودکار شناسایی الگوها

با توجه به اینکه برخی از ابزارهای خوب، مثل SSA درصد بازیابی بالایی دارند و در شناسایی الگوها، به‌جز الگوهایی که ساختار و رفتارشان با بقیه الگوها مشابه است، نتیجه خوبی تولید می‌کنند، برخی از محققان به‌جای ابداع یک ابزار جدید، به سمت تولید یک خروجی مطمئن از ابزارهای موجود روی آوردند، که نتیجه‌های خوبی نیز گرفتند. به‌طور مثال در سال ۲۰۰۵ [۱۸]، یک روش پالایش خروجی ابزار ارائه داده شد. ورودی پالایش، خروجی ابزار ۲۰۰۳ [۱۹] بود که میزان مثبت کاذب بالایی داشت. در [۱۸]، بر اساس انعطاف‌پذیری‌ها و تنوع پیاده‌سازی‌های یک الگو، یک سری معیار استخراج کردند تا با بررسی معیارها در کد کاندیدهای استخراج‌شده توسط ابزار [۱۹]، درستی یا نادرستی هر کاندید را معین کنند. کار [۱۸] انجام شد تا، نمونه‌های مثبت کاذب ابزار، شناسایی و از خروجی حذف شوند. پالایش [۱۸]، خروجی [۱۹] را تا حد زیادی بهبود داد، اما معیارهای [۱۸]، تنها بر اساس تنوع پیاده‌سازی‌های یک الگو استخراج شده بودند و ساختار و عملکرد مشابه الگوهای دیگر در تولید معیارها در نظر گرفته نشده بودند. از طرفی بااینکه حذف نمونه‌های مثبت کاذب خروجی مطمئن‌تری را در اختیار توسعه‌دهنده قرار می‌دهد، در همان حال، خیلی از اطلاعات را نیز از بین می‌رود.

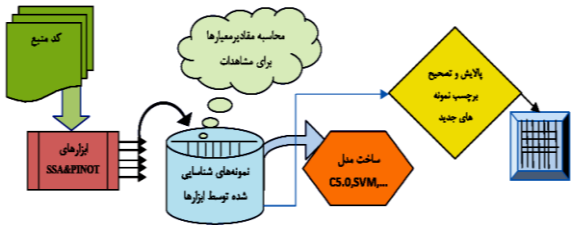
در سال ۲۰۱۲ [۲۰]، از ساختارهای ریز[۱۶] (ساختار داخلی و خارجی یک الگو را مشخص می‌کنند و با آنالیز کد به دست می‌آیند) به‌عنوان پیش‌گویی کننده‌ها (معیارها) به کمک شیوه‌های داده‌کاوی برای پالایش خودکار ابزاری بنام مارپل، استفاده کردند. در [۲۰] ذکر شده است که به دلیل قواعد رفتاری سخت‌گیرانه الگوهای با ساختار و عملکرد مشابه، در تجزیه‌وتحلیل ایستا نتوانسته‌اند ساختارهای ریزی جهت پالایش آن‌ها پیدا کنند.

برای بهبود خروجی روش‌های ارائه‌شده، استفاده از داده‌کاوی و استخراج معیارهایی جدیدی که بتواند نقص‌های موجود در ابزارها را رفع کند لازم است. علت استفاده از داده‌کاوی این است که می‌توان هرگونه معیاری را (رفتاری و ساختاری) به‌راحتی باهم ترکیب کرد و با تولید مقادیر این معیارها و تجزیه‌وتحلیل آن‌ها توسط شیوه‌های داده‌کاوی، قانون‌های قدرتمندی را برای پالایش و تصحیح خروجی ابزارهای تقریباً قوی مثل SSA ارائه داد و قدرت آن را در شناسایی تمامی الگوها تکمیل کرد. پالایش به کمک استخراج معیارهای خاص سنجش الگوها در [۱۸] و کارهای مشابه مثل [۲۰، ۲۱] انجام شده است. اما هیچ‌کدام معیارهایی استخراج نکرده‌اند که، با ترکیب آن‌ها عملیات تصحیح خروجی (بدون از دست دادن اطلاعات) قابل انجام باشد. ما در این کار قصد داریم، به‌جای شناسایی و حذف مثبت‌های کاذب ابزارها، نقص‌های موجود در خروجی بهترین ابزارها را از بین ببریم و خروجی را تصحیح کنیم.

۳- شیوه پیشنهادی (تصحیح برچسب خروجی)

همان‌طور که در بخش قبل توضیح داده شد، محققان در زمینه شناسایی الگوهای طراحی به سمت پالایش کردن خروجی ابزارهای موجود سمت‌وسو پیدا کرده‌اند. بر اساس بررسی‌های ما، این مقاله اولین کاری است که نه‌تنها مثبت‌های کاذب را شناسایی (پالایش) می‌کند بلکه برچسب آن‌ها را هم تصحیح می‌کند.

در این کار، به‌منظور پالایش و تصحیح خروجی ابزارهای خودکار متداول با قابلیت شناسایی الگوهای با ساختار و عملکرد مشابه، یک مجموعه معیارهای جدید جهت استخراج، ارائه شده است. این معیارها صرفاً باهدف سنجش الگوها و اندازه‌گیری پارامترهای ثابت و متمایزگر هر الگو از میان پارامترهای مشابه، استخراج شده‌اند. درواقع این معیارها، عملکردها و ساختارهای مشابه را در نظر می‌گیرند، و تفاوت‌ها را به‌عنوان مرز، بین مشاهدات بیرون‌کشیده‌شده توسط ابزارها می‌کشند، و خروجی‌شان را تصحیح می‌کنند. فرایند تصحیح برچسب در شکل (۱) نشان داده شده است. در ادامه، به بیان فلسفه استخراج معیارها پرداخته خواهد شد.



شکل (۱): شمای کلی عملیات پالایش و تصحیح برچسب

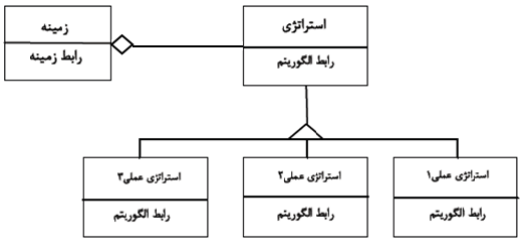
۳-۱- فلسفه استخراج معیارها

معیارهای استخراج‌شده در این تحقیق، با بررسی و مشاهده دقیق مجموعه‌ای تقریباً کامل از انعطاف‌پذیری‌های هر الگوی طراحی، از پیاده‌سازی‌های واقعی استخراج شده‌اند. در استخراج این معیارها علاوه بر بررسی انعطاف‌های یک الگو، ابزارهای خودکار مختلف در شناسایی آن الگو نیز مورد بررسی قرارگرفته‌اند. بررسی و بازبینی خروجی ابزارها نشان می‌دهد که اکثر مثبت‌های کاذب یک الگو، در میان الگوهایی با ساختار و عملکرد مشابه با آن هستند.

با بررسی ابزارها، خطاها و کمبودهای هر ابزار را جستجو و با استخراج معیارها سعی بر رفع آن می‌کنیم. در ادامه، الگوی استراتژی و یک سری معیارها جهت شناسایی آن شرح داده شده‌اند. بعدازآن، الگوهایی که به دلیل ساختار و عملکرد مشابه با الگوی استراتژی، باعث تولید مثبت کاذب برای آن شده‌اند، بررسی و معیارهایی جهت شناسایی و متمایزسازی آن‌ها از الگوی استراتژی، آورده شده است.

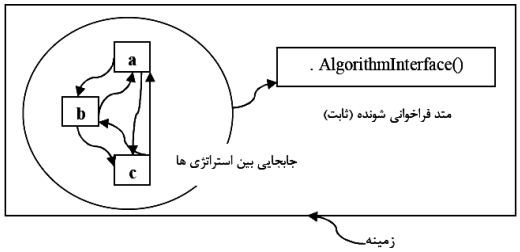
۳-۱-۱- الگوی استراتژی

هدف الگوی استراتژی، تعریف خانواده‌ای از الگوریتم‌ها برای یک عمل (مثل الگوریتم‌های quik sort- buble sort برای عمل sort)، در پوشینه[۱۷] قرار دادن هرکدام و ایجاد امکانی برای تعویض آن‌ها در یک زمینه خاص و در زمان اجرا است [۱]. این الگو طبق شکل (۲) از سه نقش تشکیل می‌شود: استراتژی، استراتژی‌های عملی، و زمینه. استراتژی، رابط یا یک کلاس منتزع است که از یک روش که اسکلت مشترک این خانواده را معین می‌کند تشکیل شده است. این روش توسط هر یک از کلاس‌های استراتژی عملی برای در برگرفتن یکی از تنوع الگوریتم‌ها، پیاده‌سازی می‌شود. زمینه، کلاسی است که از الگوریتم‌ها استفاده می‌کند و شرایطی را ایجاد می‌کند که بتوانیم استراتژی‌ها یا الگوریتم‌ها را برحسب شرایط مختلف در زمان اجرا تعویض کنیم.



شکل (۲): الگوی استراتژی

در ساده‌ترین شکل همان‌طور که در شکل (۲) نشان داده شده است، الگوی استراتژی یک رابط مشترک را مثل «رابط الگوریتم» برای پیاده کردن خانواده‌ای از الگوریتم‌های مرتبط به هم استفاده می‌کند. به‌طوری‌که هر استراتژی عملی در چارچوب این رابط مشترک، یک تنوع از الگوریتم را پیاده‌سازی می‌کند. درواقع مشترک بودن رابط (بین گونه‌های مختلف الگوریتم) این امکان را فراهم می‌کند که در یک زمینه خاص با فراخوانی‌های ثابت، اشیاء استراتژی بتوانند در زمان اجرا جابجا شوند. شکل (۳) را مشاهده کنید.



شکل (۳): نیاز به رابط مشترک در ساختار فراخوانی ثابت بین اشیاء متغیر

مشترک بودن رابط باعث می‌شود که الگوریتم‌ها بتوانند به‌راحتی با یکدیگر در زمان اجرا، در زمینه‌ای خاص و در چارچوب فراخوانی ثابت مبادله شوند. اگر یک استراتژی واقعی این پایه را حفظ کند، تشخیص الگوی استراتژی سختی زیادی ندارد.

ازجمله معیارهایی که در کارهای پیشین برای پالایش کاندیدهای الگوی استراتژی ارائه شده‌اند، در ادامه مورد نقد و بررسی قرار گرفته‌اند.

۱- بررسی حضور رابطه‌ی پیوند بین زمینه و استراتژی.

۲- شمارش تعداد فرزندان استراتژی با محدودیت حداقل دو فرزند.

۳- شمارش تعداد فرزندان و والد زمینه با محدودیت صفر فرزند و صفر والد.

۴- سنجش حضور الگوریتم در هر یک از کلاس‌های وارث استراتژی، با شمارش تعداد حلقه‌ها، بازگشت‌ها، و ساختارهای شرطی.

۵- بررسی حضور خصوصیت "حفظ فیلد[۱۸]" (استراتژی‌ها، علاوه بر اینکه در لحظه‌ی ساخت نمونه از زمینه و از طریق پارامتر سازنده زمینه، قابل‌تعیین هستند، باید در هرلحظه دیگر در زمان اجرا نیز قابل‌تعیین و جابجایی باشند) در زمینه برای جابه‌جایی بین استراتژی‌ها.

در نقد و تحسین هر یک از معیارهای فوق، موارد زیر قابل‌بحث هستند:

۱- حضور رابطه پیوند: نیازمند بررسی است، اما جزء معیارهای مشابه با برخی الگوهای دیگر است.

۲- تعداد فرزندان استراتژی: یک استراتژی حقیقی می‌تواند تک‌فرزند هم باشد زیرا باهدف گسترش در آینده برحسب نیازمندی‌ها ساخته شده است و هنوز تکمیل نشده است.

۳- تعداد فرزندان و والد زمینه: زمینه مکانی است که الگوریتم‌ها در آن مورد استفاده قرار می‌گیرند، بنابراین زمینه، هر سلسله مراتبی را می‌تواند داشته باشد.

۴- خصوصیت الگوریتمیک: در هر استراتژی وارث، این معیار،کاملاً بجا، و نیازمند بررسی است. اما در رابطه با روش انجام آن‌که شمارش تعداد حلقه‌ها، بازگشت‌ها، و ساختارهای شرطی است، برای همه الگوریتم‌ها صدق نمی‌کند. یک الگوریتم ممکن است از یک خط مثل "return a+b;" تا هزاران خط شامل حلقه و بازگشت‌ها باشد. از طرفی خیلی از الگوها برای برخی از پیاده‌سازی‌های خود از چنین ساختارهایی استفاده می‌کنند.

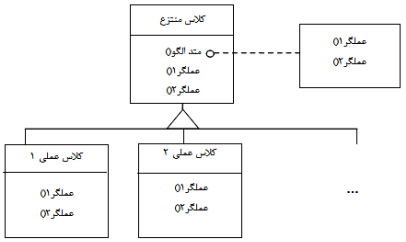
۵- حفظ فیلد [۱۸، ۱۵]: یک خصیصه لازم (به‌منظور ایجاد امکان تعویض الگوریتم‌ها در هرلحظه‌ای از اجرا توسط کلاینت) در زمینه است. اما حضور آن در زمینه، لزوماً نشان‌دهنده این نیست که سلسله‌مراتب مرتبط‌شده به آن یک استراتژی است.

اگر به شکل (۶) که یکی از نمونه‌های واقعی استراتژی در نرم‌افزار jhotdraw است دقت کنید، از استاندارد بیان‌شده کمی فاصله دارد. توجه کنید، این نمونه واقعی از استراتژی، فقط یک پیاده‌سازی دارد، بدیهی است که برحسب نیازمندی‌ها در آینده پیاده‌سازی‌های دیگر آن گسترش خواهند یافت. بر طبق شکل (۶) این نمونه استراتژی چهار روش دارد. این نمونه توسط معیارهای قبلی قابل‌شناسایی نیست. حال به بیان معیارهای پیشنهادشده جهت تصحیح برچسب و پالایش الگوی استراتژی می‌پردازیم. تا با پیدا کردن این معیارها در کد، بتوانیم دقت شناسایی خروجی ابزارها را افزایش دهیم. لازم به ذکر است که معیارهای پیشنهادشده، همه بر مبنای تصحیح برچسب «استراتژی اشتباه شناسایی‌شده» تعریف و از کد استخراج می‌شوند.

بر اساس این هدف، «پیاده کردن خانواده‌ای از الگوریتم‌ها در کنار هم برای یک عمل مشترک، و ایجاد امکانی برای جابجایی بین آن‌ها در یک زمینه‌ی خاص با ساختار فراخوانی ثابت»، و بر اساس جمله «هر الگوریتم مجموعه‌ای از دستورالعمل‌هاست که طبق یک ترتیب خاص اجرا می‌شوند و مسئله‌ای را حل می‌کنند» [۲۲]، حتماً باید یک چارچوب یا اسکلت مشترک وجود داشته باشد که بخش‌های مختلف یک الگوریتم را تعریف، و ترتیب آن را مشخص کند، و آن چارچوب به‌عنوان ساختار ثابت فراخوانی توسط زمینه مورد استفاده قرار گیرد (خصوصاً برای استراتژی‌های دارای چند روش). در این کار، این اصل به‌عنوان اولین معیار (استراتژی ۱) در نظر گرفته شده است و به یکی از چهار روش زیر شناسایی می‌شود.

استراتژی ۱:

روش اول: گاهی دستورالعمل‌های الگوریتم با یک ترتیب مشخص، در کلاس استراتژی منتزع به شکل یک «روش الگو[۱۹]» تعریف می‌شوند. روش الگو یک ترتیب ثابت را برای گونه‌های مختلف الگوریتم فراهم می‌کند. این روش در هیچ‌یک از کلاس‌های وارث رونویسی نمی‌شود و توسط زمینه به‌عنوان ساختار فراخوانی ثابت، فراخوانی می‌شود. شکل (۴) یک نمونه روش الگو را نشان می‌دهد.



شکل (۴): روش الگو

روش دوم: مثل شکل (۵) زمینه ترتیب ثابتی از دستورالعمل‌های الگوریتم‌ها را در یکی از روش‌های خودش تعریف می‌کند.

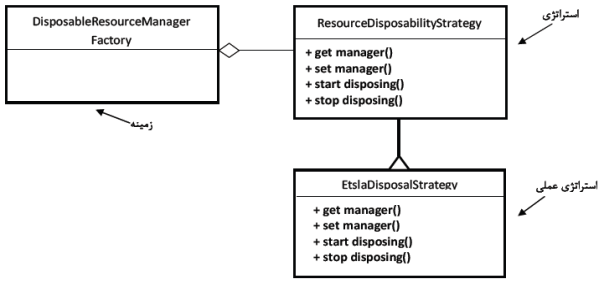
```
Class Context {
Run()
{
Oparation1()
Operation2()
}}
```

شکل (۵): ترتیب ثابتی از دستورالعمل‌های الگوریتم‌ها در روش Run کلاس زمینه

روش سوم: برای داشتن ترتیب‌های متنوع بین دستورالعمل‌ها، این اسکلت‌بندی در هر کلاس وارث استراتژی، در قالب یک روش مشترک بین استراتژی‌ها تشکیل می‌شود و آن روش توسط زمینه فراخوانی می‌گردد.

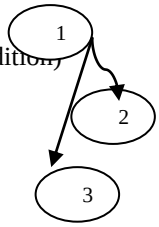
روش چهارم: روش‌های استراتژی منتزع در هر کلاس وارث به شکل یک زنجیر یکدیگر را فراخوانی می‌کنند، به‌طوری‌که سر زنجیر بین همه استراتژی‌های وارث، مشترک است. زمینه سر زنجیر را فراخوانی می‌کند.

این معیار می‌تواند شاخص خوبی برای تشخیص استراتژی‌ها، خصوصاً استراتژی‌ها با بیش از یک رابط و کم‌تر از دو پیاده‌سازی در کنار معیارهایی که در ادامه آمده‌اند، باشد. شکل (۷) اسکلت تعریف الگوریتم شکل (۶) را به روش سوم نشان می‌دهد.



شکل (۶): نمونه‌ی واقعی از الگوی استراتژی و رابط‌های آن

```
SetManager()
{
If(GetManager()== Condition)
StopDisposing();
else
StartDisposing();
}
```



شکل (۷): اسکلت مشترک ترتیب فراخوانی دستورات الگوریتم شکل (۶)، در روش SetManager

استراتژی ۲. آیا کاندید استراتژی دارای یک ساختار منتزع- عملی است؟ این معیار به‌جای محاسبه تعداد فرزندان استراتژی پیشنهاد شده است. تعداد فرزندان استراتژی از یک تا چندین پیاده‌سازی متغیر هستند. بنابراین به‌جای در نظر گرفتن تعداد فرزندان، وجود ساختار منتزع- عملی (برای پیاده‌سازی گونه‌های مختلف الگوریتم، روش‌ها باید رونویسی شوند، بنابراین وجود ساختار منتزع، الزامی است) را معیار قرار می‌دهیم.

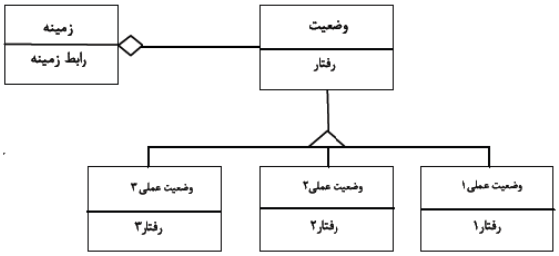
استراتژی ۳. آیا رابط‌ها با دسترسی عمومی، در استراتژی منتزع، بین همه استراتژی‌های عملی رونویسی شده‌اند؟ لازم است که روش‌های عمومی رونویسی‌شده بین استراتژی‌های وارث برای قابل تعویض ساختن آن‌ها در یک ساختار فراخوانی ثابت، مشترک باشند. به عبارتی روش جدید با دسترسی عمومی نباید در یک استراتژی موجود باشد و در دیگری نباشد.

استراتژی ۴. آیا والد استراتژی و زمینه مشترک است؟ سلسله‌مراتب استراتژی و زمینه استفاده‌کننده از آن، باید از هم مستقل باشند. در غیر این صورت زمینه نیز یک استراتژی است و باید حاوی مشخصات الگوریتمیک باشد و استقلال بین الگوریتم‌ها و زمینه از بین می‌رود.

استراتژی ۵. حضور خصیصه «حفظ فیلد» در زمینه، به‌منظور این‌که کلاینت در هرلحظه‌ای از اجرا بتواند برحسب شرایط از یک استراتژی به استراتژی دیگر جابجا شود، حضور حداقل یک روش با دسترسی عمومی (به‌علاوه روش سازنده) که استراتژی‌ها را به شکل پارامتر از کلاینت بگیرد و جابجایی بین آن‌ها را انجام دهد، لازم است. اگر این روش در زمینه استفاده‌کننده از استراتژی موجود نباشد مفهوم اصلی این الگو، قابلیت جابجا شدن الگوریتم‌ها در زمان اجرا، حفظ نشده است.

استراتژی ۶. آیا همه رابط‌هایی که از سمت زمینه استفاده‌کننده از استراتژی فراخوانی می‌شوند، در کلیه استراتژی‌های عملی موجود هستند؟ روش عمومی جدیدی که در یکی از استراتژی‌ها تعریف شده ولی در دیگری موجود نباشد، نباید از سمت زمینه استفاده‌کننده از استراتژی مورد فراخوانی قرار گیرد. در غیر این صورت اصل جابجایی با یک فیلد مشترک را از بین می‌برد. ما این فراخوانی را 'فراخوانی خارج از محدوده' می‌نامیم.

۳-۱-۲- الگوی وضعیت [۲۰]



شکل (۸): الگوی طراحی وضعیت

گاهی نیازمند یک شیء با چندین وضعیت هستیم، به‌طوری‌که با حرکت از یک وضعیت به وضعیت دیگرش، رفتار مناسبی را از خود نشان دهد [۱]. همچنین می‌خواهیم در یک زمینه خاص و در یک‌لحظه، تنها در یک وضعیت قرار گیرد و به‌راحتی وضعیتی اضافه یا حذف شود. الگوی وضعیت نیازمندی‌های فوق را برآورده می‌کند. الگوی وضعیت ساختاری کاملاً مشابه با الگوی استراتژی دارد و تنها در رفتار با الگوی استراتژی متفاوت است. این شباهت باعث شده که در خیلی از موقعیت‌ها، ابزارهای خودکار شناسایی الگوهای طراحی، نتوانند الگوی وضعیت را از استراتژی تشخیص دهند. جابجایی بین کلاس‌های مشخص‌کننده وضعیت‌ها یا به‌صورت داخلی و یا از خارج انجام می‌گیرد. درواقع الگوی وضعیت به‌صورت آشکارا مشخص نمی‌کند که در چه مکانی و چه زمانی تغییر وضعیت رخ می‌دهد. بنابراین هر الگوی وضعیت می‌تواند تعدادی رابط را پیاده‌سازی کند که به رابط‌های کلاس وضعیت دیگر هیچ ارتباطی نداشته باشد.

معیارهای متمایزکننده الگوی وضعیت از الگوی استراتژی در ادامه آمده است. به‌طورکلی حالت‌های امکان‌پذیر برای جابجایی بین وضعیت‌ها (شکل (۸)) در الگوی وضعیت، یا از طریق زمینه قابل انجام است، یا به‌صورت داخلی و توسط وضعیت‌ها. معیارهای زیر انواع حالت‌های امکان‌پذیر برای جابجایی بین وضعیت‌ها را بررسی می‌کنند.

وضعیت ۱. تعداد استراتژی‌های عملی (دلیل استفاده از واژه‌های استراتژی و زمینه در تمام معیارها این است که، این معیارها می‌خواهند نمونه‌های استراتژی را تصحیح برچسب کنند و ابزارها در خروجی خود، آدرس استراتژی و زمینه را داده‌اند) که حاوی ایجاد نمونه‌ای از همزاد خود هستند به‌طوری‌که، این نمونه‌ها بعد از ایجاد در یک فیلد مشترک مستقر شده‌اند (یعنی بین آن‌ها با دستورات شرطی جابجایی رخ داده است). گاهی زیرکلاس‌های وضعیت، همزادی از خود را ایجاد و وضعیت جانشین را به‌منظور جابجایی تعیین می‌کنند.

وضعیت ۲. تعداد نمونه‌های استراتژی که در زمینه ایجاد شده‌اند، به‌طوری‌که این نمونه‌ها در یک فیلد مشترک مستقر شده‌اند (یعنی بین آن‌ها با دستورات شرطی جابجایی رخ داده است).

گاهی زمینه مجموعه‌ای از وضعیت‌ها را ایجاد و امکانی برای جابجایی آن‌ها فراهم می‌کند.

وضعیت ۳. آیا زمینه استفاده‌کننده از استراتژی خود را به‌عنوان پارامتر به استراتژی می‌فرستد؟ ممکن است زمینه خود را به وضعیت‌ها بفرستد تا وضعیت‌ها هنگام جابجایی، زمینه را مطلع کرده و سپس زمینه، جابجایی را با وضعیت ارسال‌شده از طرف وضعیت‌ها انجام دهد.

وضعیت ۴. تعداد نمونه‌های استراتژی که در زمینه ایجاد می‌شوند و فیلدی که در آن قرار می‌گیرند مشترک نیست.

اگر دو یا چند وضعیت، در یک زمینه و در فیلدهای غیرمشترک ایجاد شوند، در این صورت الگوی موردنظر نمی‌تواند الگوی وضعیت باشد چون یک زمینه در هر زمان تنها می‌تواند در یک وضعیت قرار بگیرد.

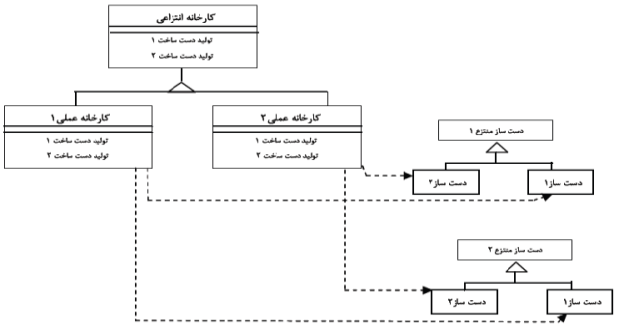
وضعیت ۵. تعداد فراخوانی‌هایی که از عملگرهای زمینه در استراتژی می‌شود. در حالت وضعیت ۳، برای تنظیم وضعیت، زمینه از سمت وضعیت‌ها باید فراخوانی شود.

وضعیت ۶. آیا والد استراتژی و زمینه مشترک است؟ سلسله‌مراتب وضعیت و زمینه استفاده‌کننده از آن برای حفظ استقلال باید از هم جدا باشند.

۳-۱-۳- الگوی کارخانه انتزاعی[۲۱]

اگر مجموعه‌ای از اشیاء را داشته باشیم که به سبک‌های مختلف قابل‌تولید شدن باشند، هنگام جابجایی از یک سبک به سبک دیگر، چه باید کرد؟ باید از هزاران شیء به هزاران شیء (از همان نوع اما با سبک متفاوت)، جابجا شویم؟ الگوی کارخانه انتزاعی این امکان را می‌دهد که در چنین مواقعی تنها با جابجایی یک شیء، از یک سبک به سبک دیگر جابجا شویم. هدف این الگو ایجاد یک رابط برای تولید خانواده‌ای از اشیا مرتبط یا وابسته، بدون معین کردن کلاس عملی‌شان است [۱].

حتی با مقایسه تعریف دو الگوی کارخانه انتزاعی (شکل (۹)) و استراتژی، شباهت‌های موجود در پیاده‌سازی و عملکرد آن‌ها مشخص است. الگوی استراتژی امکان جابجایی بین خانواده‌ای از الگوریتم‌ها را در یک زمینه خاص فراهم می‌کند، درحالی‌که کارخانه انتزاعی نیز، امکان جابجایی بین استانداردهای مختلف تولید مجموعه‌ای از اشیاء را در یک زمینه خاص فراهم می‌سازد. برای فراهم‌سازی این جابجایی در یک زمینه خاص نیازمند چارچوب مشترک ذکرشده در بخش استراتژی و ساختار فراخوانی ثابت هستیم.



شکل (۹): الگوی طراحی کارخانه انتزاعی

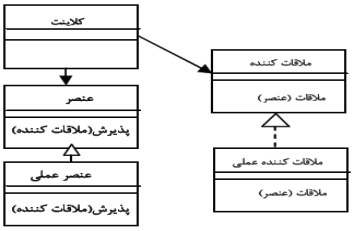
معیارهای متمایزکننده الگوی کارخانه انتزاعی به جهت مشخص کردن هدف هر عملکرد و ساختار مشابه (الزام حضور رابط‌های مشترک برای جابجایی خانواده‌ای از اشیاء در یک مکان مشترک) با الگوی استراتژی در زیر آمده‌اند.

کارخانه انتزاعی ۱. تعداد روش‌های عمومی که بین تمام استراتژی‌های وارث، رونویسی شده‌اند. روش‌های عمومی رونویسی‌شده باید در تمام تولیدکننده‌ها مشترک باشند (به‌منظور ایجاد قابلیت جابجایی بین استانداردهای موجود، در یک زمینه خاص با ساختار فراخوانی ثابت).

کارخانه انتزاعی ۲. کم‌ترین تعداد روش‌های رونویسی‌شده عمومی که شیء جدیدی را ایجاد و برمی‌گردانند (بین کلاس‌های وارث استراتژی) بدون اینکه پیاده‌سازی‌ای را در بر بگیرند. هر تولیدکننده عملی در کارخانه انتزاعی، تنها برای تولید مجموعه‌ای از اشیاء مرتبط یا وابسته استفاده می‌شود و پیاده‌سازی دیگری را در بر ندارند. «کم‌ترین» استفاده می‌شود چون، همه زیرکلاس‌های تولیدکننده باید رابط‌هایی که اشیایی را برمی‌گردانند و هیچ پیاده‌سازی‌ای ندارند، داشته باشند. مقدار این معیار باید برابر با تعداد روش‌های رونویسی‌شده مشترک بین همه تولیدکننده‌ها باشد.

۳-۱-۴- الگوی ملاقات‌کننده[۲۲]

هرگاه بخواهیم یک یا چند عملگر مشترک (مثلاً draw()) را به مجموعه‌ای از عناصر (مثل دایره، مثلث، مستطیل، و ...) اضافه کنیم، مجبور هستیم این عملگر را در تک‌تک عناصر، به شیوه‌های مختلف پیاده‌سازی کنیم. اگر نیاز به تغییر بخشی از عملگر داشته باشیم، به تعداد عناصر، نیازمند تغییر در کلاس عناصر هستیم. الگوی ملاقات‌کننده (شکل (۱۰)) امکانی را فراهم می‌کند که یک یا چند عملگر جدید را به مجموعه‌ای از عناصر اضافه کنیم، بدون اینکه نیاز به ویرایش کل کلاس عناصر داشته باشیم [۲۳].



شکل (۱۰): الگوی طراحی ملاقات کننده

الگوی ملاقات‌کننده این امکان را با جداسازی بخش پیاده‌سازی عملگرها از عناصر، فراهم می‌کند. به‌طوری‌که تمام پیاده‌سازی‌های یک عملگر برای عناصر مختلف در یک کلاس قرار می‌گیرند. عناصر هستند که خودشان را به پیاده‌سازی‌ها می‌فرستند و عملگرها، عنصر خاص خودشان را شناسایی و عمل موردنظر را انجام می‌دهند. شباهت بین الگوی استراتژی و ملاقات‌کننده در این نکته قرار دارد که به بخش پیاده‌سازی ملاقات‌کننده به ازای اضافه شدن هر عملگر روی کلیه عناصر، یک کلاس اضافه می‌شود. بنابراین یک عنصر، برای شناسایی عملگرهای خودش باید یک شناسه مشترک بین کل عملگرها داشته باشد. ملاقات‌کننده، این شناسه را با قرار دادن یک رابط مشترک برای هر عنصر بین عملگرهای مختلف موردنیازش فراهم می‌کند.

معیارهای شناسایی الگوی ملاقات‌کننده روی نمونه‌های مثبت کاذب استراتژی در ادامه آمده‌اند.

ملاقات‌کننده ۱. تعداد روش‌هایی که در استراتژی منتزع حداقل یک پارامتر هم نوع می‌گیرند. اصولاً تعداد روش‌هایی که در الگوی ملاقات‌کننده برای عمل روی ساختار عناصر تعریف می‌شوند، نسبت به تعداد عناصری که روی آن‌ها عمل می‌کنند برابر یا بزرگ‌ترند. در ملاقات‌کننده برای ملاقات کردن هر عنصر یک روش تعریف می‌شود.

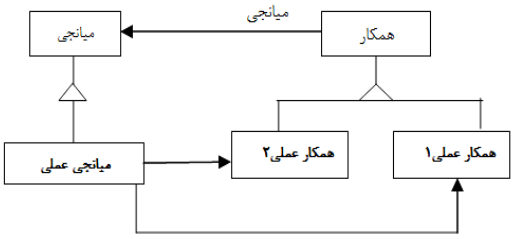
ملاقات‌کننده ۲. تعداد کلاس‌های استراتژی که به سمت زمینه فرستاده می‌شوند. در الگوی ملاقات‌کننده، همه عملگرها خودشان را به عناصر مشخص به‌عنوان پارامتر می‌فرستند و عناصر نیز آن‌ها را می‌پذیرند.

ملاقات‌کننده ۳. تعداد کلاس‌های ساختار زمینه که خودشان را به سمت استراتژی می‌فرستند. در الگوی ملاقات‌کننده همه عناصر خودشان را به سمت عملگرهای مشخص به‌صورت پارامتر می‌فرستند و عملگرها آن‌ها را ملاقات می‌کنند.

ملاقات‌کننده ۴. تعداد فرزندان زمینه (پیاده‌سازی‌ها) که حاوی فیلد عمومی از استراتژی (عناصر) هستند. اصولاً هر عملگر از الگوی ملاقات‌کننده یک عنصر همکار را به‌صورت پارامتر می‌گیرد و برعکس. بنابراین نیازی به تعریف فیلدی از عناصر که در کل عملگر قابل‌استفاده باشد ندارند. اصولاً مقدار این معیار برای این الگو صفر است.

۳-۱-۵- میانجی[۲۳]

گاهی مجموعه‌ای از اشیاء داریم که به‌طور دائم با یکدیگر در ارتباط هستند. درواقع به‌جای این‌که چند شیء جهت انجام ارتباطاتشان باهم پیوند مستقیم برقرار کنند، الگوی میانجی اتخاذی می‌بینند که همه اشیاء تنها به یک شیء میانی پیوند داشته باشند و آن شیء عملیات ارتباطی همه آن‌ها را حل‌وفصل کند. به این طریق اشیاء مرتبط، می‌توانند مستقل از هم کار کنند و هم‌زمان ارتباط داشته باشند، گاهی میانجی ارتباطات بین وضعیت‌های الگوی وضعیت را فراهم می‌کند. درواقع به دلیل حضور ساختار وضعیت، این الگو (شکل (۱۱)) بین مثبت‌های کاذب الگوی استراتژی قرار گرفته شده است. معیارهای تشخیص الگوی میانجی از الگوهای بحث‌شده در بخش‌های قبل، در ادامه شرح داده شده‌اند.



شکل (۱۱): الگوی طراحی میانجی

میانجی ۱. تعداد زیرکلاس‌های ساختار استراتژی که پارامتر سازنده‌شان از یک نوع می‌باشند (از نوع زمینه). در الگوی میانجی، همه عناصر (عناصری که به‌واسطه میانجی رابطه برقرار می‌کنند) میانجی را به شکل پارامتر از سازنده می‌گیرند.

میانجی ۲. تعداد زیرکلاس‌های استراتژی که در کلاس هم نوع با پارامتر سازنده‌شان (زمینه)، حاوی فیلد وضعیت هستند. همه عناصر همکار که میانجی آن‌ها را به هم مرتبط می‌کند در میانجی یک فیلد وضعیت از ساختارشان دارد.

میانجی ۳. حداقل تعداد فراخوانی از سلسله‌مراتب استراتژی در ساختار زمینه. میانجی برای برقراری ارتباط میان عناصر نیاز به فراخوانی عملگرهای هر عنصر به درخواست عنصر مرتبط دارد.

۴- فرایند یادگیری

در داده‌کاوی با دو مجموعه داده مواجه هستیم، داده آموزشی و داده آزمایشی. صفات داده آموزشی را مجموعه معیارهایی تشکیل می‌دهند که هویت موجودیت‌های قرارگرفته در رکوردها را پیش‌گویی می‌کنند. موجودیت‌های داده‌ی آموزشی، مشاهداتی هستند که از قبل هویتشان شناسایی شده است. داده‌ی آموزشی حاوی یک ستون پیش‌گویی است. مقادیر این ستون، با برچسب‌هایی پر می‌شوند که هویت اصلی موجودیت‌ها را نشان می‌دهد (مثلاً درست یا غلط). داده آزمایشی حاوی مشاهداتی است که هویت اصلی‌شان شناخته شده نیست.

با تجزیه‌وتحلیلی که به‌واسطه الگوریتم‌های داده‌کاوی روی داده‌ی آموزشی صورت می‌گیرد مدل‌هایی ساخته می‌شود. مدل‌سازی، دانش موجود در مشاهدات داده آموزشی را در قالب یک سری قوانین استخراج می‌کند. داده آزمایشی برای ارزیابی دقت پیش‌گویی مدل

ساخته‌شده روی داده آموزشی به کار برده می‌شود. درواقع پیش‌گویی یک فرایند دومرحله‌ای دارد، فاز یادگیری و فاز دسته‌بندی.

در فاز یادگیری بر اساس مجموعه داده‌ی آموزشی، مدل طبقه‌بند ساخته می‌شود و در فاز طبقه‌بندی بر اساس مدل ساخته‌شده در فاز قبل، مجموعه داده جدید که در فاز یادگیری استفاده نشده است (مجموعه داده آزمایشی) دسته‌بندی می‌شود (پیش‌گویی می‌شود که مشاهدات جدید چه برچسبی بگیرند).

نمونه مشاهدات داده آموزشی از سه پروژه‌ی Javaio ,Jhotdraw و Jrefactory توسط دو ابزار SSA و PINOT فراهم شده‌اند. به‌طورکلی مراحل یادگیری داده به شرح زیر است:

۱- محاسبه مقادیر پیش‌بینی‌کننده‌ها (معیارها) برای چندین نمونه مثبت کاذب و مثبت صحیح شناسایی‌شده توسط ابزارها.

۲- در این گام با استفاده از الگوریتم‌های داده‌کاوی مدل‌های تصمیم‌گیری جهت پیش‌بینی روی مجموعه داده ساخته می‌شوند. این مدل‌ها می‌توانند با هر ابزار خودکار شناسایی الگو جهت بهبود در شناسایی ترکیب شوند.

۳- بررسی صحت یا عدم صحت نمونه‌های شناسایی‌شده. به‌طوری‌که در این کار، اگر نمونه‌ی موردنظر مثبت صحیح باشد، برچسب استراتژی و اگر مثبت کاذب باشد، با بررسی و شناسایی الگوی صحیح موجود در آن با «هویت صحیح الگو» برچسب می‌خورد. نهایتاً درصورتی‌که هیچ الگویی در آن مشاهده نشود با «بدون الگو» برچسب می‌خورد.

روش‌های دسته‌بندی

از پنج الگوریتم داده‌کاوی برای استخراج دانش از معیارهای بحث‌شده در بخش قبل استفاده شده است. هر پنج مورد، مدلی را برای تصمیم‌گیری فراهم می‌کنند.

طبقه‌بند C5.0

این طبقه‌بند درواقع بر اساس تقسیم مبتنی بر نمونه روی صفتی (معیار) که بیش‌ترین سود اطلاعاتی را با خود دارد، کار می‌کند. سپس هر زیرنمونه تعریف‌شده با اولین تقسیم، دوباره تقسیم می‌شود (معمولاً بر اساس یک صفت متفاوت). این فرایند تکرار می‌شود تا اینکه هیچ زیرنمونه قابل‌تقسیم نداشته باشیم. سرانجام پایین‌ترین سطح تقسیم‌ها دوباره بررسی می‌شوند. آن‌هایی که تأثیر قابل‌توجهی بر مقدار مدل ندارند حذف یا هرس می‌گردند [۲۴].

طبقه‌بند SVM

یک طبقه‌بند و الگوریتم رگرسیون است که از تئوری یادگیری ماشین با حداکثر دقت پیش‌بینی، بدون «اتصال بیش از حد معیارها به داده آموزشی[۲۴]» استفاده می‌کند. این روش از یک تبدیل غیرخطی بر داده‌های یادگیری استفاده می‌کند. سپس با جستجو برای تساوی‌های رگرسیون در داده‌های تبدیل‌شده، کلاس‌ها (اهداف پیش‌بینی) را جدا می‌کند. SVM، خصوصاً برای آنالیز داده‌ها با تعداد زیادی از فیلد های پیش‌گویی‌کننده مناسب است [۲۵]. در این تحقیق از SMO، یک نوع SVM تطبیق‌یافته برای داده‌های حاوی بیش از دو هویت قابل پیش‌بینی استفاده می‌شود.

طبقه‌بند J48 و ADABOOSTM1 (J48)

طبقه‌بند j48 درخت‌های تصمیم‌گیری را از مجموعه داده‌ی دارای برچسب می‌سازد. این طبقه‌بند هر صفت را با تقسیم داده‌ها به مجموعه‌های کوچک‌تر به‌عنوان یک تصمیم‌گیرنده در نظر می‌گیرد.

AdaBoostM1 یک الگوریتم یادگیری ضعیف را به‌عنوان پایه در نظر می‌گیرد و کارایی آن را افزایش می‌دهد. این الگوریتم برای دسته‌بندی بیش‌تر از دو هدف جهت پیش‌بینی نسبت به Boosting بهبود یافته است [۲۶]. در این کار با الگوریتم j48 استفاده شده است.

طبقه‌بند BOOSTING

یک شیوه کلی برای بهبود کارایی الگوریتم‌های یادگیری (الگوریتم‌هایی که به‌آرامی به سمت دسته‌بندی صحیح میل می‌کنند) است. به‌صورت تکراری الگوریتم یادگیری ضعیف را روی توزیع‌های مختلف داده‌ی آموزشی اجرا می‌کند. سپس طبقه‌بندهای تولیدشده با الگوریتم یادگیری ضعیف را ترکیب می‌کند و یک طبقه‌بند ترکیبی ایجاد می‌کند [۲۶].

نتایج

مهم‌ترین عناصر در فرایند داده‌کاوی، در درجه اول قدرت متمایزکنندگی معیارهایی است که به‌عنوان پیش‌گویی‌کننده‌های یک مجموعه داده قرار می‌گیرند و دوم صحیح برچسب زدن نمونه‌هایی که مدل داده آموزشی روی آن‌ها ایجاد می‌شود. درواقع طبقه‌بندها دانش موجود در معیارها را شناسایی می‌کنند. در این بخش نتایج ارزیابی دقت یادگیری مدل‌های ایجادشده از شیوه‌های مختلف داده‌کاوی بر اساس معیارهای پیشنهادی ارائه شده است.

در این کار، یک بازبینی دستی به ازای تمام نمونه‌های یافت‌شده توسط ابزارها انجام می‌گیرد. سپس یک دسته‌بندی چندتایی (چندتایی یعنی چند هدف قابل پیش‌بینی (استراتژی، وضعیت و...) موجود است)، یک دسته‌بندی یکی در مقابل بقیه (در این دسته‌بندی، هر بار یک الگو درست و بقیه با نادرست برچسب می‌خورند) و یک دسته‌بند دوبه‌دو (الگوها دوبه‌دو در نظر گرفته می‌شوند، به‌عنوان‌مثال، یک‌مرتبه نمونه‌های استراتژی و وضعیت را به‌عنوان مجموعه داده در نظر می‌گیریم، به‌طوری‌که یک الگو با درست و دیگری با نادرست برچسب می‌خورد و یک‌مرتبه استراتژی با میانجی و به همین ترتیب) برای هر نمونه فراهم می‌گردد. در دسته‌بندی چندتایی نمونه‌ها برچسب‌های «نام هر الگو» و در صورت موجود نبودن الگو به‌صورت «بدون الگو» دسته‌بندی می‌شوند.

دقت برچسب دهی «یکی در مقابل همه»، قدرت متمایزسازی معیارها را در جداسازی یک الگو از الگوهای دیگر نشان می‌دهد، درحالی‌که دقت روش دوبه‌دو قدرت جداسازی یک الگو در مقابل یک الگوی دیگر را نشان می‌دهد و دسته‌بندی چند برچسبه دقت جداسازی همه الگوها را در کنار یکدیگر می‌سنجد. ازآنجاکه در این تحقیق، هدف به‌دست آوردن معیارهایی است که بتوانند الگوهای با ساختار و عملکرد مشابه را به‌درستی از یکدیگر جدا کنند، مقایسه دسته‌بندهای مختلف (SVM,C5.0,...)، سنجش آن‌ها با شیوه‌های دسته‌بندی متفاوت («یکی در مقابل همه»، دو در دو، و چند برچسبه)، نزدیکی نتایج آن‌ها، و همچنین دقت بالای نتایج نشان می‌دهد که استفاده از معیارهای استخراج‌شده در این تحقیق، در تشخیص و جداسازی الگوهای طراحی مناسب بوده و می‌تواند توسط سایر محققان نیز مورداستفاده قرار گیرد.

نمونه‌ای از مجموعه داده‌ی تهیه‌شده در جدول ۱ و ۲ آمده است.

جدول (۱): بخش کوچکی از مجموعه داده برای عمل دسته‌بندی یکی در مقابل همه

Context and strategy	M1[۲۸]	M2[۲۷]	M3[۲۶]	M4[۲۵]	...	پالایش
Locator handle-locator	۱	۰	۱	۰		صحیح
Tool button-tool	۱	۰	۰	۰		ناصحیح
PrettyPrintVisitor-node	۱	۱	۱	**۸۶**		ناصحیح
Text figure- collection factory	۱	**۷**	۱	۰		ناصحیح
Draw applet- tool	۱	۰	۰	۰		ناصحیح
....					...	

جدول (۲): بخش کوچکی از مجموعه داده برای عمل تصحیح برچسب

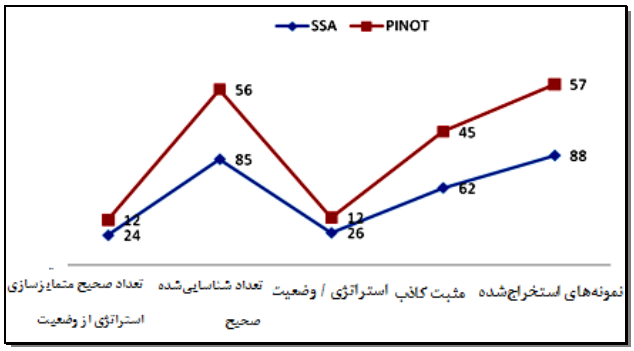
Context and strategy	M1	M2	M3	M4	...	تصحیح برچسب
Locator handle-locator	۱	۰	۱	۰	...	استراتژی
Tool button-tool	۱	۰	۰	۰		وضعیت
PrettyPrintVisitor-node	۱	۱	۱	**۸۶**		ملاقات‌کننده
Text figure- collection factory	۱	**۷**	۱	۰		کارخانه انتزاعی
Draw applet- tool	۱	۰	۰	۰		میانجی
....			...			

ترکیب مقادیر معیارها، قوانین واضح و با ثباتی را ایجاد کرده‌اند و دسته‌بندها توانسته‌اند آن قوانین را استنباط و استخراج کنند. همان‌طور که بهبود روی ابزارهای SSA و PINOT در شکل (۱۲) نشان داده شده است. شکل (۱۲) پنج آمار را مرتبط به نتایج به‌دست‌آمده روی نمونه‌هایی که جز الگوهای مورد بررسی در این تحقیق بوده‌اند، نشان می‌دهد. اولین آمار، تعداد کاندیدهای استخراج‌شده برای الگوی استراتژی توسط ابزارهای SSA و PINOT است. آمار دوم، مربوط به تعداد مثبت کاذبی است که در خروجی ابزارهای خودکار SSA و PINOT برای شناسایی الگوی استراتژی، به‌صورت دستی و بر اساس سندهای موجود برای این سه پروژه متن‌باز (چه سندهای داخلی در خود پروژه‌ها و چه سند خارجی ایجادشده برای شرح این پروژه‌ها)، مشاهده شده است. آمار سوم، تعداد نمونه‌هایی است که توسط ابزارها به‌صورت «استراتژی/ وضعیت» شناسایی شده است، یعنی مشخص نشده که کدام‌یک استراتژی و کدام‌یک وضعیت هستند.

آمار چهارم مشخص می‌کند از کل نمونه‌های استخراج‌شده توسط ابزارها، چندتای آن‌ها توسط معیارها و به کمک الگوریتم‌های داده‌کاوی به‌درستی با نام الگوی صحیحشان شناسایی شده‌اند. آمار آخر بیان می‌کند که از تعداد نمونه‌هایی که ابزارها تحت یک گروه و به شکل «استراتژی / وضعیت» نشان داده‌اند، چند تای آن‌ها را معیارها، از یکدیگر متمایز کرده‌اند.

آمار آخر نشان می‌دهد که اکثر وضعیت‌ها از استراتژی‌ها قابل متمایز شدن هستند. به دلیل این‌که این دو ابزار درصد بازیابی بالایی دارند، کاهش منفی کاذب با تعداد خیلی کم مشاهده شد. اما برخی از الگوها مثل کارخانه انتزاعی و میانجی، که توسط برخی از ابزارها مثل SSA قابل‌شناسایی نیست، از روی نمونه‌های استراتژی بازیابی شده‌اند. پس از تشکیل داده‌ها، از نرم‌افزار آماری کلمن‌تاین و وکا جهت ارزیابی دقت یادگیری از شیوه‌ی اعتبارسنجی ضربدری با ده بخش[۲۹] استفاده شده است (داده آموزشی به ده قسمت تقسیم می‌شود به‌طوری‌که طی ده مرتبه، هر مرتبه یک بخش به‌عنوان داده‌ی آزمایشی و نه بخش به‌عنوان داده آموزشی در نظر گرفته می‌شود و دقت مدل به این طریق سنجیده می‌شود).

نتایج، با سه روش دسته‌بندی یکی در مقابل همه[۳۰]، دوبه‌دو و چندبرچسبه ارائه شده‌اند. در این تحقیق، دقت یادگیری از طریق دو معیار Accuracy و TPR[۳۱] سنجش شده است. Accuracy، برابر با درصد کل نمونه‌هایی است که به‌طور صحیح برچسب خورده‌اند، درحالی‌که TPR، تنها درصد نمونه‌های مثبتی که به‌طور صحیح برچسب خورده‌اند را مشخص می‌کند. نتایج در جدول‌های ۳ تا ۷ آورده شده است.



شکل (۱۲): بهبود به‌دست آمده روی دو ابزار SSA و PINOT

جدول (۳): دقت و نرخ مثبت صحیح یادگیری یک الگو در مقابل بقیه نمونه‌ها

یکی در مقابل بقیه	استراتژی	وضعیت	میانجی	ملاقات‌کننده	کارخانه انتزاعی	بدون الگو
C5.0 (Accuracy)	٪۹۴/۷	٪۹۸/۲	٪۹۸/۲	٪۹۸/۲	٪۱۰۰	٪۹۴/۷
C5.0 (TPR)	٪۹۳/۳	٪۸۸/۷	٪۱۰۰	٪۱۰۰	٪۱۰۰	٪۹۰
C5.0 Boosting (Accuracy)	٪۹۸/۲	٪۹۸/۲	٪۱۰۰	٪۱۰۰	٪۱۰۰	٪۹۶/۴
C5.0 Boosting (TPR)	٪۱۰۰	٪۸۸/۷	٪۱۰۰	٪۱۰۰	٪۱۰۰	٪۸۸/۸
Adaboostm1 J48 (Accuracy)	٪۹۱/۲	٪۹۴/۷	٪۹۶/۴	٪۹۶/۴	٪۱۰۰	٪۸۴/۲
Adaboostm1 J48 (TPR)	٪۸۰	٪۷۱/۴	٪۸۰	٪۱۰۰	٪۱۰۰	٪۸۳/۳
SVM (Accuracy)	٪۹۴/۷	٪۹۱/۲	٪۱۰۰	٪۹۴/۷	٪۹۴/۷	٪۸۲/۴
SVM (TPR)	٪۹۳/۳	٪۲۸/۶	٪۸۰	٪۱۰۰	٪۲۵	٪۴۴/۴
J48 (Accuracy)	٪۹۲/۹	٪۸۹/۴	٪۹۲/۹	٪۹۶/۴	٪۱۰۰	٪۸۴/۲
J48(TPR)	٪۹۳/۳	٪۷۱/۴	٪۲۰	٪۱۰۰	٪۱۰۰	٪۶۱/۱

نتایج جداول ۷-۳، توانایی معیارهای ارائه‌شده را، ازنظر میزان متمایزسازی الگوهای مجموعه داده از یکدیگر، موردسنجش قرار می‌دهند.

جدول شماره ۳، نتایج ارزیابی معیارها را با روش برچسب‌دهی یکی در مقابل همه و به کمک الگوریتم‌های مختلف دسته‌بندکننده، به ازای هر الگو نشان می‌دهد. با فرض آزمایش جداسازی الگوی A از پنج الگوی F, E, D, C و B در روش «یکی در مقابل همه» نمونه مشاهدات از نوع A را در طبقه T و نمونه مشاهدات از انواع F, E, D, C، B را در طبقه F قرار می‌دهیم. سپس بر اساس مقادیر معیارها روی مشاهدات، الگوریتم‌های داده‌کاوی قوانینی را استخراج می‌کنند که بتوان گروه T را از گروه F جدا کرد. الگوریتم‌ها در صورتی در فرایند جداسازی موفق می‌شوند و دقت بالایی دارند که، گروه T مشخصه‌های متمایزکننده بدون ابهامی را روی مقادیر مشاهدات T نشان دهند. بدیهی است که در روش دوبه‌دو جداسازی دو کلاس A از B در مجموعه داده کوچک‌تر، تلاش کم‌تری را نسبت به جداسازی A از دسته F، E، D، C و B می‌طلبد.

به‌عنوان‌مثال نتایج عددی در ستون شماره ۲ با عنوان «استراتژی» از جدول ۳، میزان متمایزسازی الگوی استراتژی را در کنار الگوهای دیگر نشان می‌دهد. به‌طوری‌که اعداد ستون ۲ که در ستون ۱ با عنوان TPR در کنار هم هستند، نرخ پیش‌بینی صحیح نمونه‌های استراتژی را نسبت به دیگر الگوها نشان می‌دهند و اعداد از ستون ۲ که در ستون ۱ با عنوان Accuracy در کنار هم هستند، نرخ صحیح پیش‌بینی را برای دو برچسب درست و غلط در کنار هم نشان می‌دهند. نتایج نشان می‌دهد در روش یکی در مقابل همه، معیارها، الگوها را خصوصاً با کمک الگوریتم‌های C5.0 و Boosting که اثبات شده در یادگیری دقت بالاتری دارند، در حدود ۹۰ درصد مواقع، به‌خوبی شناسایی می‌کنند.

جدول (۴): دقت یادگیری روش برچسب دهی دوبه‌دو با شیوه SVM

	وضعیت	ملاقات‌کننده	میانجی	کارخانه انتزاعی	بدون الگو
استراتژی	٪۹۰/۹	٪۹۵/۶	٪۱۰۰	٪۹۴/۷	٪۹۳/۹
وضعیت	-	٪۸۶/۶	٪۱۰۰	٪۸۱/۸	٪۸۰
ملاقات‌کننده	-	-	٪۱۰۰	٪۹۱/۶	٪۸۴/۶
میانجی	-	-	-	٪۱۰۰	٪۹۳/۹
کارخانه انتزاعی	-	-	-	-	٪۸۶/۳
بدون الگو	-	-	-	-	-

جدول (۵): دقت یادگیری روش برچسب دهی دوبه‌دو با شیوه C5.0

	وضعیت	ملاقات‌کننده	میانجی	کارخانه انتزاعی	بدون الگو
استراتژی	٪۹۵/۴	٪۱۰۰	٪۱۰۰	٪۱۰۰	٪۹۳/۹
وضعیت	-	٪۱۰۰	٪۹۱/۶	٪۱۰۰	٪۹۶
ملاقات‌کننده	-	-	٪۱۰۰	٪۱۰۰	٪۹۶/۱
میانجی	-	-	-	٪۱۰۰	٪۹۵/۶
کارخانه انتزاعی	-	-	-	-	٪۱۰۰
بدون الگو	-	-	-	-	-

جدول (۶): دقت یادگیری روش برچسب دهی دوبه‌دو با شیوه Boosting C5.0

	وضعیت	ملاقات‌کننده	میانجی	کارخانه انتزاعی	بدون الگو
استراتژی	٪۱۰۰	٪۱۰۰	٪۱۰۰	٪۱۰۰	٪۹۶/۹
وضعیت	-	٪۱۰۰	٪۹۱/۶	٪۱۰۰	٪۱۰۰
ملاقات‌کننده	-	-	٪۱۰۰	٪۱۰۰	٪۱۰۰
میانجی	-	-	-	٪۱۰۰	٪۱۰۰
کارخانه انتزاعی	-	-	-	-	٪۱۰۰
بدون الگو	-	-	-	-	-

جداول شماره ۴، ۵ و ۶ نتایج ارزیابی معیارها را در متمایزسازی نمونه‌های هر دو الگو در کنار هم، به کمک الگوریتم‌های مختلف دسته‌بندی کننده SVM، C5.0، و Boosting نشان می‌دهند. به‌عنوان‌مثال در ردیف ۲ که با عنوان استراتژی همراه شده است، نتیجه عددی هر سلول، نتیجه دقت (Accuracy) پیش‌بینی هر کاندید از الگوی استراتژی را نسبت به یک الگوی دیگر نشان می‌دهد. نتایج این سه جدول نشان می‌دهد که دقت معیارها با کمک الگوریتم‌های C5.0 و Boosting، برابر با میانگین ۹۸/۶ درصد است که دقت بالایی را نشان می‌دهد.

جدول (۷): دقت یادگیری روش برچسب دهی چندتایی

Multi Classification	SMO	J48	Adaboostm1 J48	C5.0	Boosting(c5.0)
Accuracy	٪۷۵/۴	٪۸۷/۷	٪۸۷/۷	٪۹۲/۹	٪۹۸/۲

جدول شماره ۷، که مهم‌ترین جدول در سنجش نتایج است، دقت متمایزسازی الگوها را، به کمک معیارها و الگوریتم‌های دسته‌بند مختلف، در جداسازی شش کلاس متفاوت در کنار هم می‌سنجد. در عمل پیش‌بینی توسط الگوریتم‌های داده‌کاوی، هر چه حجم داده‌ها جهت تحلیل و تعداد کلاس‌ها (برچسب‌ها) جهت پیش‌بینی بیش‌تر باشند، به‌ویژه در حالتی که خصوصیات کلاس‌ها به هم نزدیک باشند، کار پیش‌بینی بسیار مشکل می‌شود. در چنین حالتی حضور ویژگی‌ها (معیارهای پیش‌بینی‌کننده) با توان

متمایزسازی بالا است که می‌تواند منجر به نتیجه با ثبات و با دقت بالا شود. جدول ۷، به‌خوبی بیانگر توان بالای متمایزسازی معیارهای ارائه‌شده در این تحقیق است.

۵- نتیجه‌گیری

در این کار، علاوه بر این‌که دقت شناسایی ابزارهای خودکار شناسایی الگوهای طراحی بهبود داده شده است (شناسایی مثبت‌های کاذب)، به‌علاوه خروجی ابزارهای خودکار، در شناسایی الگوهای با ساختار و عملکرد مشابه تصحیح شد. در تصحیح خروجی علاوه بر این‌که مثبت‌های کاذب به میزان قابل قبولی کاهش می‌یابند، تعدادی از منفی‌های کاذب نیز، قابل کاهش هستند. ابزارهای شناسایی الگوهای طراحی به‌واسطه اضافه شدن عملگر تصحیح برچسب، می‌توانند به حد قابل قبولی بهبود یابند.

مراجع

[1] E. Gamma, R. Helm, R. Johnson and J. Vlisside, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Pub Co, 1995.

[2] N. Shi and R. A. Olsson, "Reverse engineering of design patterns from java source code," in ASE'06, pp. 123–134, 2006.

[3] R. Keller, R. Shauer, S. Robitaille and P. Pag´e, "Pattern based reverse-engineering of design components," In Proc. Of the 21st International Conference on Software Engineering, pp. 226–235, 1999.

[4] J. Bansiya, "Automating design-pattern identification –DP++ is a tool for C++ programs," Dr. Dobbs Journal, 1998.

[5] A. Asencio, S. Cardman, D. Harris and E. Laderman, "Relating expectations to automatically recovered design patterns," In WCRE, pp. 87–96, 2002.

[6] M. Vok´a˘c, "An efficient tool for recovering design patterns from C++ code," Journal of Object Technology, March-April 2006.

[7] K. Brown, "Design Reverse Engineering and Automated Design Pattern Detection in SmallTalk," Master's thesis, North Carolina State University, 1998.

[8] http://www.javaworld.com/javaworld/jw-02-2001/jw-0216-jhotdraw.html.

[9] http://jrefactory.sourceforge.net/, 2006.

[10] http://sewiki.iai.unibonn.de/research/dpd/benchmarks/start.

[11] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides and S. T. Halkidis, "Design pattern detection using similarity scoring," IEEE TSE, vol. 32, no. 11, pp. 896–909, 2006.

[12] C. Kramer and L. Prechelt, "Design recovery by automated search for structural design patterns in object-oriented software," In Proc. Working Conf. Reverse Engineering, pp. 208–215, November 1996.

[13] G. Antoniol, R. Fiutem, and L. Cristoforetti, "Using metrics to identify design patterns in object-oriented software," In Proceedings of the Fifth International Symposium on Software Metrics (METRICS98), pp. 23–34, 1998.

[14] R.S. Rao and M. Gupta, "Design pattern detection by sub graph isomorphism technique," International Journal of Engineering and Computer Science, 2013.

[15] A. Binun and G. Kniesel, "Joining forces for higher precision and recall of design pattern detection," in proceeding of the 16th conference on software maintenance and reengineering (CSMR2012), IEEE Computer society, pp. 27-30, 2012.

[16] S. Alhusain, S. Coupland, R. John and M. Kavanagh, "Towards machine learning based design pattern Recognition," Computational Intelligence (UKCI), 2013.

[17] P. W. Egrzynowicz and K. Stencel, "Relaxing queries to detect variants of design patterns," Computer Science and Information Systems (FedCSIS), 2013.

[18] R. Ferenc, A. Besz´edes, L. Fulop and J. Lele, "Design pattern mining enhanced by machine learning," In ICSM, pp. 295–304, 2005.

[19] Z. Balanyi and R. Ferenc, "Mining design patterns from C++ source code," In Proceedings of the 19th International Conference on Software Maintenance (ICSM 2003), pp. 305–314, Sept. 2003.

[20] M. Zanoni, "Data mining techniques for design pattern detection," PhD's thesis, Università degli Studi di Milano Bicocca, 2012.

[21] S. Uchiyama, H. Washizaki, Y. Fukazawa and A. Kubo, "Design pattern detection using software metrics and machine learning," Fifth International Workshop on Software Quality and Maintainability (SQM'11), pp. 38-47, 2011.

[22] http://courses.cs.vt.edu/cs2604/spring04/Notes/C04.AlgorithmAnalysis.pdf.

[23] J. Cooper, *Design Patterns Java Companion*, Addison -Wesley, 1998.

[24] Clementine 12, copyright(c) integral solutions ltd, help-modeling nodes, (1994-2007)

[25] J. Han, M. Kamber and J. Pei, "Data mining third edition: concepts and techniques," M. Kaufmann pub, 2011.

[26] Y. Freund and R.E. Schapire, "Experiments with a New Boosting Algorithm," 1996.

زیرنویس‌ها

1 Structural
2 Creational
3 Behavioral
4 Generalization
5 Association
6 Similarity scoring algorithm

۷ Strategy، امکان داشتن خانواده‌ای از الگوریتم‌ها را در کنار هم و تعویض آن‌ها در زمان اجرا فراهم می‌کند. این الگو ساختاری کاملاً مشابه با الگوی State دارد. شباهت آن‌ها برحسب رابطه‌ها، نقش‌ها و برخی از خصوصیات داخلی است.

۸ State، هرگاه نیاز به یک شیء داریم که وضعیت‌های متعددی دارد به‌طوری‌که تحت شرایط مختلف باید از یک وضعیت به وضعیت دیگر جابجا شود، و با هر تغییر وضعیت رفتار خاصی اجرا کند، از الگوی State استفاده می‌کنیم (در بخش سه توضیح داده شده است).

۹ زنجیره‌ای از کلاس‌ها را برای مدیریت یک تقاضا فراهم می‌کند.

۱۰ درصد نمونه‌هایی که بازیابی شده است.

۱۱ الگویی که دستیابی به یک شیء پرهزینه را کنترل و تسهیل می‌کند.

12 Delegation
13 Pattern INference and recOvery Tool
14 Detection of Patterns by joining forces

۱۵ هرگاه بخواهیم در کل نرم‌افزار از یک کلاس، تنها مجوز ایجاد یک نمونه داشته باشیم، از الگوی یگانه بهره استفاده می‌کنیم.

16 Micro structure
17 Encapsulate
18 Field Maintenance

۱۹ هدف روش الگو، تعریف اسکلت ثابت از یک الگوریتم در یک روش است.

20 State
21 Abstract Factory
22 Visitor
23 Mediator
24 Over fitting, generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

۲۵ معیار شماره یک میانجی

[۲۶] معیار شماره دو استراتژی
[۲۷] معیار شماره دو از کارخانه منتزع
[۲۸] معیار شماره یک از استراتژی

[29] Ten fold cross validation
[30] One per all
[31] True positive rate