دوازدهمین کنفرانس بین‌المللی انجمن کامپیوتر ایران
دانشگاه شهید بهشتی، دانشکده مهندسی برق و کامپیوتر ،تهران، ایران، ۱ تا ۳ اسفند ۱۳۸۵

استخراج جنبه از مدل طراحی نرم‌افزار

سید شبیر فخرایی*، سید حسن میریان حسین‌آبادی†

چکیده

در این مقاله راهکاری عملی برای استخراج جنبه از مدل طراحی نرم‌افزار با توجه به ریشه‌های جنبه در سطح طراحی و نیازمندی‌ها، در قالب یک فرآیند ارایه شده است. در این فرآیند ابتدا از کامل بودن مدل طراحی اطمینان حاصل می‌شود که برای این منظور دغدغه‌های غیرکارکردی وارد مدل موارد کاربرد شده، سپس مدل طراحی نرم‌افزار از دو دیدگاه ساختاری و رفتاری بررسی می‌گردد.

با بررسی دغدغه‌های متلاقی[1]، اطلاعات لازم از مدل موارد کاربرد استخراج می‌شود. در ادامه تلاقی در دو سطح موارد کاربرد و کلاس در نظر گرفته شده و کلاس‌هایی که دچار آمیختگی[2] شده‌اند مشخص می‌شوند. در مراحل بعدی نمودارهای ترتیبی مورد تحلیل قرار گرفته و الگوهایی جهت کشف جنبه و نوع آن، ارایه شده است. در نهایت جهت امکان مقایسه بهتر، خصوصیات بررسی شده به‌صورت کمّی بیان شده و فرمولی بر مبنای WMC پیشنهاد گردیده است.

کلمات کلیدی

جنبه گرایی، جنبه اولیه، استخراج جنبه، مدل طراحی، UML، دغدغه‌های متلاقی، فرآیند

Aspect Extraction from Software Design Model

Seyed-Shobeir Fakhraei, Seyed-Hassan Mirian-Hosseinabadi
{shobeir@fakhraei.com, hmirian@sina.sharif.edu}

Abstract

Aspect-Oriented programming was introduced by Gregor Kiczales in 1997 to handle concerns that could not be fully separated via Object-Oriented programming, which are called crosscutting concerns. Traditionally, aspect-oriented software development has focused on the software life cycle's implementation phase: developers identify and capture aspects mainly in code. But aspects are evident earlier in the life cycle, such as during requirements engineering and design level.

In this paper issues on different approaches for handling crosscutting concerns in requirement and design level were discussed. A practical process for identification and extraction of aspects in software design model was proposed. The process starts by checking the completeness of the model and adds non-functional requirements to UML's use case model and validates the model's relationships. Crosscutting concerns are identified in the design model during the next steps and then behavioral specifications of the model are analyzed with aspect identification perspective. Finally a formula for comparing different criteria based on WMC was proposed.

Keywords

Aspect Oriented, Early Aspect, Aspect Mining, Design Model, UML, Crosscutting Concern, Process

* کارشناس ارشد، دانشگاه صنعتی شریف، دانشکده مهندسی کامپیوتر، fakhraei@ce.sharif.edu
† عضو هیأت علمی دانشگاه، دانشگاه صنعتی شریف، دانشکده مهندسی کامپیوتر، hmirian@sina.sharif.edu

۱- مقدمه

برای کاهش پیچیدگی‌ها در چرخه تولید نرم‌افزار و کاهش فاصله بین شناخت اهداف سیستم و پیاده‌سازی آنها، جداسازی دغدغه‌ها[۳] به عنوان یک راه‌حل مناسب مطرح است. در جداسازی دغدغه‌ها، مساله-های بزرگ و پیچیده به مسایلی کوچکتر و ساده‌تر که هر یک مستقلاً قابل حل هستند، تقسیم شده و حل می‌شوند.

لیکن در بسیاری از موارد (مانند سازگاری و پردازش موازی)، دغدغه‌ها به‌کلی مستقل از یکدیگر نبوده و با استفاده از فناوری‌های موجود در پیاده‌سازی قابل جداسازی نمی‌باشند. در این حالت یک دغدغه، به عنوان دغدغه اصلی شناسایی شده و بقیه به عنوان دغدغه‌های فرعی در نظر گرفته می‌شوند. دغدغه‌های فرعی در میان قسمت‌های مختلف دغدغه‌های اصلی پخش شده و با آنها تلاقی پیدا می‌کنند، از اینرو به آنها «دغدغه‌های متلاقی» نیز گفته می‌شود.

در سال ۱۹۹۷ گرگور کیکزالس[۴] و همکارانش در مرکز تحقیقات پایلوآلتو، در مقاله‌ای با نام برنامه‌نویسی جنبه‌گرا[1] برای اولین بار نام جنبه‌گرایی را مطرح کرده و به دنبال آن هدایت تولید اولین زبان جنبه‌گرا، یعنی AspectJ را بر عهده گرفتند.

«جنبه نوعی دغدغه است که کارکرد آن توسط دغدغه‌های دیگر و در موقعیت‌های مختلف راه‌اندازی می‌شود [2] ». برای پیاده‌سازی جنبه‌ها در AspectJ سه مفهوم تعریف و به کارگرفته شده است:

- نقطه پیوست[۵]: هر نقطه پیوست، یک نقطه در هنگام اجرای برنامه است (مانند نقطه شروع اجرای یک رویه).
- محل برش[۶]: یک محل برش، شامل مجموعه‌ای از نقاط پیوست است.
- توصیه[۷]: شامل خصوصیات یک محل برش، قطعه‌ای کد که در هنگام رسیدن به آن اجرا می‌گردد و نیز یک نوع، می‌باشد. انواع توصیه عبارتند از قبل، بعد و پیرامون. نوع توصیه زمان اجرای آنرا در هنگام برخورد با محل برش مورد نظر مشخص می‌کند.

۲- جنبه در نیازمندی‌ها و طراحی

در مهندسی نرم‌افزار مراحل تحلیل و طراحی حداقل به اندازه مرحله پیاده‌سازی اهمیت داشته، حتی بسیاری از متخصصین نقش این مراحل را در موفقیت پروژه از مرحله پیاده‌سازی پررنگتر می‌دانند[2].

از آنجا که جنبه‌ها در مرحله پیاده‌سازی وارد شده و هم‌اکنون استفاده می‌شوند، بسیار غیرمنطقی است که طراحی سیستم به روش‌های دیگر صورت گرفته و پیاده‌سازی آن بصورت جنبه‌گرا باشد. از طرف دیگر در نظر گرفتن جنبه در فاز طراحی، طراحی را ساده‌تر خواهد کرد، چرا که طراحی به روش‌های موجود از جمله شیءگرایی ابزار مناسب جهت دربرگیری دغدغه‌های متلاقی را در اختیار نداشته، که این مساله موجب پیچیدگی در طراحی خواهد شد. لیکن در صورت ورود جنبه‌ها به سطح طراحی می‌توان دغدغه‌های متلاقی را جداگانه مورد بررسی قرار داده و از این پیچیدگی کاست. به‌همین دلیل در چند سال اخیر پژوهش‌های گسترده‌ای در این زمینه صورت گرفته است.

از جمله این تلاش‌ها می‌توان به فراهم‌سازی روش‌ها و ابزارهایی جهت طراحی جنبه‌گرا اشاره نمود، که به‌دلیل پرکاربرد بودن و نیز مقبولیت عمومی UML، اکثراً در جهت افزودن امکاناتی به UML برای دربرگیری و پشتیبانی از جنبه‌ها بوده و دستاوردهای چشم‌گیری را نیز به همراه داشته است[3]. روش کلارک[۸] [2] با عنوان Theme و روش استین[۹] [4] با عنوان AODM از جمله این راه‌کارها است.

پس از فراهم شدن نسبی ابزار طراحی جنبه‌ها، شناخت خصوصیات جنبه برای به‌کارگیری آن در مراحل مختلف مورد توجه قرار گرفت. گروهی از این تلاش‌ها بر استخراج جنبه از کد تولید شده به روش شیءگرایی، متمرکز شده‌اند. این روش‌ها خصوصیات پیاده‌سازی جنبه-ها را بررسی می‌کنند. از جمله این روش‌ها می‌توان به FEAT و Aspect Detector اشاره کرد [6][5].

گروهی دیگر بر شناخت جنبه در مراحل اولیه تولید نرم‌افزار تاکید کرده و سعی در بررسی خصوصیات آن در طراحی و نیازمندی‌ها دارند. جنبه در این مراحل بسیار به مفهوم دغدغه متلاقی نزدیک بوده و با عبارت جنبه اولیه[۱۰] شناخته می‌شود [7].

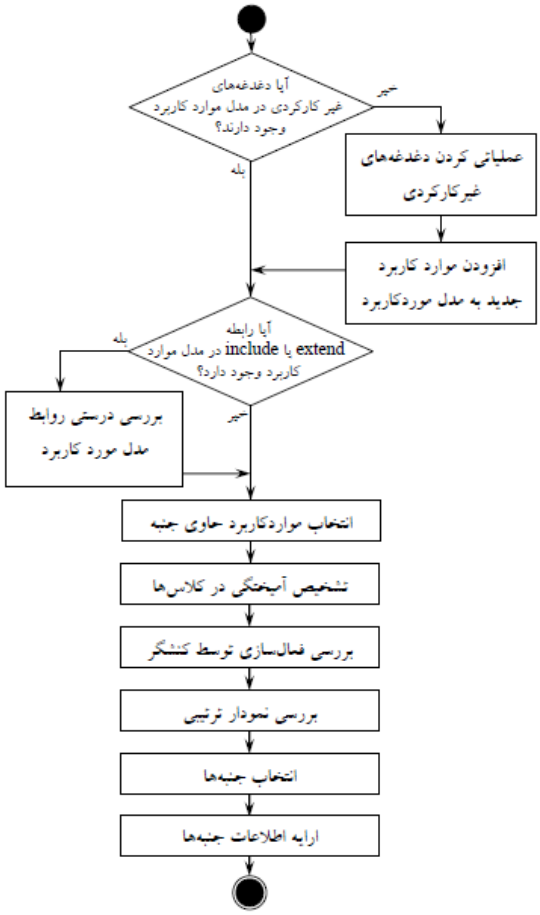
در این مقاله راهکاری عملی برای استخراج جنبه از مدل طراحی نرم-افزار با توجه به ریشه‌های جنبه در سطح طراحی و نیازمندی‌ها، در قالب یک فرآیند ارایه شده است.

۳- استخراج جنبه از مدل طراحی نرم‌افزار

در این فرآیند ابتدا دغدغه‌های متلاقی از مدل مورد کاربرد استخراج شده، سپس برای بدست‌آوردن اطلاعات دقیق‌تر ساختاری جنبه‌ها، از کلاس‌ها و مدل ترتیبی استفاده شده است. مراحل این فرآیند به‌صورت کلی در شکل(۱) آورده شده است.

در این فرآیند اطلاعات لازم برای استخراج جنبه‌ها در مراحل مختلف جمع‌آوری شده و در مراحل آخر با استفاده از این اطلاعات، کلاس‌ها رتبه‌بندی شده و کلاس‌های حاوی جنبه از آنها استخراج شده‌است. در برخی مراحل، معیارهایی برای انتخاب مواردکاربرد یا کلاس‌ها ارایه شده است. برای مقایسه دقیقتر اهمیت هر شاخص باید به صورت کمّی بیان شود تا در نهایت بتوان آنها را بر اساس امتیاز بدست آمده مرتب ساخت. برای این منظور در مراحل مختلف به خصوصیات مطرح شده یک متغیر و یک ضریب که نشان‌دهنده اهمیت و وزن آن خصوصیت است، نسبت داده شده است. فرمولی نیز مشابه با روش WMC[۱۱] به صورت $\sum w_i f_i$ برای محاسبه امتیاز هر کلاس ارایه شده، که در مراحل پایانی فرآیند، به انتخاب کلاس‌های حاوی جنبه کمک می‌کند.

این فرآیند جنبه‌ها را به عنوان جنبه پیشنهادی مطرح کرده و شاخص‌هایی برای رتبه‌بندی آنها ارایه می‌دهد، که در نهایت طراح سیستم باید از میان آنها بنا به نیازهای سیستم، انتخاب نهایی را انجام دهد.



شکل (۱): فرایند استخراج جنبه

۴- اطمینان از کامل بودن مدل

قبل از شروع به استخراج جنبه از مدل طراحی لازم است از کامل بودن مدل و درستی روابط آن اطمینان حاصل شود. زیرا ممکن است اطلاعاتی که حاوی جنبه می‌باشد در مدل آورده نشده و در آن وجود نداشته باشد. برای این منظور سه مرحله از فرآیند اختصاص یافته است، که در ادامه آورده شده‌است.

۴-۱- عملیاتی کردن دغدغه‌های غیرکارکردی[۱۲]

بسیاری از صاحب‌نظران مبحث دغدغه‌های متلاقی را همان جنبه در نظر گرفته‌اند[3]. ولی دغدغه مفهومی انتزاعی بوده و جنبه از پیاده-سازی آمده است. بنابراین راهکار کلی، یافتن دغدغه‌ها، سپس تبدیل آنها به جنبه است. دغدغه را می‌توان چنین تعریف نمود:

«هر چیزی است که مورد توجه هر ذینفع نرم‌افزار باشد. اعم از کاربر نهایی، حمایت کننده پروژه و یا سازنده نرم‌افزار[8] .»

به عنوان مثال دغدغه می‌تواند یک نیازمندی کارکردی و یا یک نیازمندی غیرکارکردی، یک محدودیت طراحی، حتی در سطح پایین مانند بافر کردن و یا استفاده از انباره باشد. برای شناخت دغدغه‌ها ابتدا باید به سوال زیر پاسخ داد:

« دغدغه‌ها در کجای مدل UML قرار گرفته‌اند؟»

مشهودترین مدلی که در آن می‌توان دغدغه‌ها را یافت، مدل مورد کاربرد است. یک مورد کاربرد به تعریف آقای جِیکوبسون که خود یکی از پدید آورندگان UML است، «زنجیره‌ای از عملیات است، که باید توسط سیستم، برای نمایش نتیجه‌ای قابل اهمیت برای کاربر، اجرا شود[8].» بنابراین موارد کاربرد پیمانه‌هایی هستند برگرفته از نیازمندی‌ها و دسته‌بندی شده با توجه به هر کاربر یا کنشگر.

«حال آیا می‌توان گفت که هر مورد کاربرد بطور عام یک دغدغه است؟»

از آنجا که ممکن است یک دغدغه کلی شامل چندین مورد کاربرد باشد، پاسخ منفی است. مثلاً در یک سیستم هتل‌داری دغدغه «مدیریت مسافرین» حاوی دو مورد کاربرد «پذیرش مسافر» و «ترخیص مسافر» می‌باشد. لیکن مورد کاربرد را می‌توان به عنوان دغدغه‌ای که نتیجه‌ای عینی در بردارد در نظر گرفت.

دغدغه‌های متلاقی از جنس دغدغه‌های غیرکارکردی در نظر گرفته شده‌اند. از دید جنبه‌گرایی در سطح طراحی دغدغه‌های کارکردی و غیرکارکردی مورد بحث می‌باشند. این بحث که در زمینه شناخت جنبه‌های اولیه مطرح می‌شود از این سوال نشات می‌گیرد که:

«آیا UML و به‌خصوص مدل مورد کاربرد همه دغدغه‌های حاوی جنبه و یا به اصطلاح دغدغه‌های متلاقی را در بر می‌گیرد؟ و یا اصلاً این دغدغه‌ها در مدل مورد کاربرد قابل مدل‌سازی هستند؟»

از آنجایی که عمده دغدغه‌های متلاقی، دغدغه‌های غیرکارکردی عنوان می‌شوند و موارد کاربرد پیمانه‌های عملیاتی یا کارکردی سیستم هستند، بسیاری از محققین بر این باورند که دغدغه‌های غیرکارکردی را نمی‌توان با ابزار کنونی به تنهایی، در مدل مورد کاربرد جای داده و این کار نیازمند ابزارهای دیگری است. تلاش‌هایی برای افزودن دغدغه-های غیرکارکردی، مثلاً به صورت «زمان پاسخ»، به مدل مورد کاربرد از این دیدگاه برگرفته شده‌اند.

در پاسخ به این ادعا می‌توان دو مطلب را بیان کرد؛ اول اینکه معمولاً هدف این روش‌ها در عمل، وارد ساختن دغدغه‌های غیرکارکردی به صورت یک عبارت کلی (مثلاً زمان پاسخ، امنیت، کارآیی)، در مدل مورد کاربرد است. این مساله مشکل بزرگ این دیدگاه می‌باشد، زیرا همان‌طوری که دغدغه‌ای مانند «مدیریت مسافرین» به موارد کاربرد کوچکتر تقسیم می‌شود، دغدغه‌هایی چون «امنیت» نیز نمی‌توانند بصورت کلی وارد مدل مورد کاربرد شده و باید به قسمت‌های کوچکتر تقسیم گردد.

مطلب دوم این است که همیشه دغدغه‌های غیرکارکردی در کد وارد نشده و در بسیاری از موارد دغدغه‌های غیرکارکردی، به تصمیمات معماری، تغییر در طراحی و یا ملاحظات سخت‌افزاری می‌انجامد. مثلاً دغدغه «زمان پاسخ» می‌تواند به استفاده از سرویس‌دهنده پرقدرت‌تر بیانجامد، حال اینکه مدل مورد کاربرد حاوی اطلاعات نرم‌افزاری با نگاه به کد است.

بنابراین اولین مساله در استخراج جنبه از مدل طراحی، وارد ساختن دغدغه‌های غیرکارکردی در مدل مورد کاربرد، برای اطمینان از کامل بودن مجموعه عواملی است، که منجر به بروز جنبه در طراحی می‌شوند. این کار با استفاده از تقسیم کردن و عملیاتی کردن دغدغه‌های غیرکارکردی قابل اجرا است. یعنی یک دغدغه غیرکارکردی کلی باید به قسمت‌های کوچکتر عملیاتی یا کارکردی شکسته شده و وارد مدل طراحی گردد. برای این منظور می‌توان از رویکردهای مورد سوکاربرد [9] یا از SIG [10]، استفاده کرد.

پس از مشخص شدن موارد کاربرد جدید نمودار ترتیبی و کلاس‌های مربوط به آن مورد کاربرد باید توسط طراح سیستم، طراحی شده و در مستندات مدل اضافه شود.

به عنوان مثال اگر امنیت یک دغدغه غیرکارکردی سیستم باشد، می‌توان امنیت را به دو مورد کاربرد، Logging و Authentication تبدیل‌کرد.

بنابراین اولین مرحله‌ای که می‌توان برای فرآیند پیشنهادی جهت استخراج جنبه در نظر گرفت، «عملیاتی کردن دغدغه‌های غیرکارکردی» است. که البته هدف از این مرحله اطمینان از کامل بودن مدل طراحی و دربرگیری تمامی اطلاعات حاوی جنبه توسط آن است.

۴-۲- افزودن مواردکاربرد جدید به مدل مواردکاربرد

حال که موارد کاربرد مربوط به نیازمندی‌های غیرکارکردی در مرحله قبل مشخص شدند، مساله استفاده از یک نوع رابطه در این مدل برای ارتباط این موارد کاربرد با سایر موارد کاربرد پیش آمده و سوالی که در این قسمت مطرح می‌شود این است که:

«از کدام رابطه می‌توان برای ارتباط موارد کاربرد غیرکارکردی و سایر موارد کاربرد استفاده کرد؟»

عده‌ای از محققین این رابطه را با توجه به معنی رابطه، رابطه extend می‌دانند. لیکن بسیاری، روابط موجود در UML را برای نشان دادن این ارتباط کافی نمی‌دانند. پیشنهاد روابط با قالب‌هایی چون crosscut، WrappedBy و constrain، بر مبنی این باور نهاده شده‌اند[15][14].

عدم استفاده از رابطه extend عمدتاً بر پایه دو دلیل زیر بنا شده است[10]:

- extend معنی تلاقی نمی‌دهند.
- هنگام استفاده از extend نقطه extension point باید در مورد کاربرد پایه تعریف گردد، که این خلاف روش جنبه‌گرایی است. زیرا در جنبه‌گرایی، محلی که جنبه بر آن تاثیر می‌گذارد، از وجود جنبه مطلع نیست.

در پاسخ می‌توان گفت اولاً نیازی به نمایش تلاقی با استفاده از یک رابطه مخصوص وجود ندارد. زیرا تلاقی یک نوع رابطه بین دو مورد کاربرد نیست، بلکه بر مبنای آمیختگی و پراکندگی[۱۳] تعریف می‌شود [11]. این مطلب که یک مورد کاربرد با مورد کاربرد دیگری تلاقی دارد، نوع رابطه و اینکه چرا این دو به هم مرتبط هستند را مشخص نمی‌کند. بنابراین لزومی ندارد رابطه extend معنی تلاقی دهد، چون اصلاً تلاقی با توجه به مفاهیم موجود در مدل مورد کاربرد ، رابطه‌ای معنی‌دار نیست.

دوم اینکه که مورد کاربرد در شی‌گرایی نیز معادل کلاس نیست و در مرحله‌های بعد مانند طراحی ساختاری و پیاده‌سازی تغییر شکل داده و معمولاً به چند کلاس تبدیل می‌شود. همان‌طوری که مقایسه بین خصوصیات مورد کاربرد و خصوصیات کلاس درست نیست، مقایسه بین خصوصیات مورد کاربرد و خصوصیات پیاده‌سازی جنبه در سطح هم امری نادرست است. زیر این دو در سطوح متفاوتی قرار دارند. از اینرو گرچه نقاط extension point در مورد کاربرد پایه تعریف می‌شوند، لیکن می‌توان در هنگام طراحی و پیاده‌سازی، آنها را به جنبه‌ها منتقل نمود.

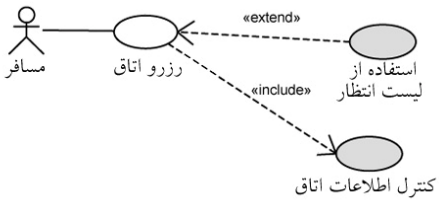
از آنجایی که مفهوم رابطه extend افزودن یک مورد کاربرد به مورد کاربرد دیگر است، به صورتی که به مورد کاربرد پایه یک کارآیی جدید اضافه می‌گردد، می‌توان از این رابطه جهت افزودن مورد کاربرد غیرکارکردی به‌دست آمده در مرحله قبل استفاده نمود. در نهایت دومین مرحله از فرآیند استخراج جنبه از مدل طراحی را می‌توان «افزودن موارد کاربرد جدید به مدل مورد کاربرد» در نظر گرفت.

۴-۳- بررسی درستی روابط مدل موارد کاربرد

به توجه به مفهوم رابطه extend که همان افزودن یک مورد کاربرد به مورد کاربرد دیگر است، این رابطه از اهمیت ویژه‌ای در بحث تشخیص دغدغه‌های متلاقی برخوردار است. از آنجایی که معمولاً در طراحی‌ها رابطه include و extend به جای یکدیگر استفاده می‌شوند، در این مرحله باید درستی روابط استفاده شده، کنترل گردد.

تفاوت رابطه extend با include چنین بیان شده است:

«برای تشخیص رابطه include از extend کافی است رابطه غیرپایه (extend شونده یا include شده) را از مدل حذف کنیم، اگر مورد کاربرد پایه مستقلاً دارای مفهوم بود، رابطه extend است و اگر نه رابطه include است[8] .»



شکل (۲): رابطه extend و include [8]

در شکل(۲) مثالی در این مورد آورده شده است. همان‌طور که در این شکل مشخص است، مورد کاربرد «رزرو اتاق»، مورد کاربرد پایه است، در صورت حذف مورد کاربرد «استفاده از لیست انتظار»، اجرای مورد کاربرد «رزرو اتاق» هنوز امکان پذیر بوده و مسافر می‌تواند به رزرو اتاق

بپردازد. در صورتی که، اگر مورد کاربرد «کنترل اطلاعات اتاق» از مدل حذف شود، رزرو اتاق دیگر معنی ندارد، زیرا در هنگام رزرو باید مشخص باشد که کدام اتاق رزرو می‌شود. بنابراین رابطه «استفاده از لیست انتظار» از نوع extend و رابطه «کنترل اطلاعات اتاق» از نوع include است.

این مرحله از فرآیند استخراج جنبه از مدل طراحی را می‌توان «بررسی درستی روابط مدل مورد کاربرد» در نظر گرفت. در ایم مرحله روابطی که اشتباه استفاده شده اصلاح می‌شوند.

۵- استخراج جنبه‌ها

مراحل قبل در اصل برای اطمینان از کامل و صحیح بودن مدل طراحی مطرح شده و هر مدلی که این مراحل را اجرا نکرده باشد، دارای نقص است. حال با توجه به اینکه نیازمندی‌های غیرکارکردی وارد مدل شده و روابط extend در مدل مشخص شده‌اند. می‌توان به مطرح کردن برخی از موارد کاربرد به عنوان موارد کاربرد حاوی جنبه‌های احتمالی پرداخت.

در این قسمت مراحل پیشنهادی برای شناخت، استخراج و ارایه جنبه-ها در شش بخش بیان شده است.

۵-۱- انتخاب موارد کاربرد حاوی جنبه

به عنوان اولین گروه از موارد کاربرد پیشنهادی حاوی جنبه‌ها، موارد کاربردی مطرح هستند که با رابطه extend به سایر موارد کاربرد مرتبط می‌شوند. می‌توان تمامی این موارد کاربرد را به نوعی حاوی جنبه دانست، زیرا رابطه extend نشان‌دهنده یک کارکرد جدید است که جزء کارکرد پایه نبوده و به آن افزوده می‌شود، بنابراین اگر جداسازی دغدغه‌های کارکردی و قابلیت استفاده مجدد، در اولویت باشد تمامی روابط extend با استفاده از جنبه قابل پیاده‌سازی است.

از آنجایی که جنبه‌ها حاوی کدهایی هستند که در موقعیت‌های مختلف و بعضاً متعدد، استفاده می‌شوند. معمولاً موارد کاربرد که تنها یک رابطه extend از آنها خارج شده، با استفاده از جنبه پیاده‌سازی نمی‌شوند. از اینرو تعداد روابط خروجی extend را می‌توان به عنوان یک شاخص ارزیابی برای رتبه‌بندی موارد کاربرد حاوی جنبه پیشنهادی مد نظر قرار داد. بدیهی است موارد کاربردی که تعداد بیشتری رابطه extend از آنها خارج شده باشد، به احتمال زیادتری حاوی جنبه هستند.

پس از پیدا کردن یک سری از موارد کاربرد به عنوان موارد کاربرد حاوی جنبه، باید کلاس‌های مرتبط با آن مورد کاربرد را به عنوان جنبه احتمالی پیشنهاد دهیم.

ارتباط بین مورد کاربرد و کلاس نیز با توجه به مدل ترتیبی مشخص می‌گردد. در نهایت خصوصیت تعداد رابطه extend بدست آمده از مدل مورد کاربرد، به کلاس‌ها منتقل شده و در جدول ارزیابی کلاس‌ها آورده می‌شود. بنابراین می‌توان این مرحله از فرآیند را تشکیل انتخاب موارد کاربرد حاوی جنبه دانست.

تعداد روابط extend را با e_i نشان می‌دهیم تا مراحل نهایی به همراه نتایج دیگر معیارهای بررسی شده، مورد استفاده قرار گیرد.

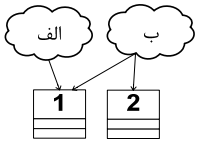
۵-۲- تشخیص آمیختگی در کلاس‌ها

اطلاعات دیگری که می‌توان به کمک مدل مورد کاربرد استخراج کرد، تلاقی دغدغه‌ها است. تلاقی دغدغه‌ها با استفاده از دو مفهوم زیر تعریف می‌شود[11]:

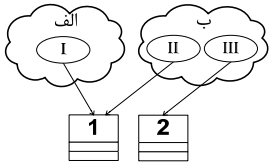
- آمیختگی: هنگامی که یک جزء حاوی چندین دغدغه است، دغدغه در این جزء دچار آمیختگی شده‌اند.
- پراکندگی: هنگامی که یک دغدغه در اجزا مختلف پراکنده شده است، این پدیده رخ می‌دهد.

حال چگونگی بروز تلاقی دغدغه‌ها در مدل طراحی مطرح است. بسیاری از محققین، تعدد رابطه‌های یک مورد کاربرد را دلیل بر متلاقی بودن آن می‌دانند. این دیدگاه، تلاقی را معادل با پراکندگی در نظر گرفته و آمیختگی را در حاشیه قرار می‌دهد. روشی که آقای بِرگ برای بیان تلاقی ارایه داده [11]، تلاقی را مربوط به یک سطح ندانسته و تلاقی را در صورت وجود پراکندگی و آمیختگی در دو سطح مختلف می‌داند. بِرگ برای اعمال این روش در عمل دو سطح دغدغه و کلاس را در نظر گرفته است.

اگرچه این روش یکی از نزدیک‌ترین روش‌ها به تعریف تلاقی است، لیکن انتخاب دغدغه‌ها به عنوان یک سطح از روش، خود ایجاد مشکل می‌کند. این مشکل از آنجا ناشی شده که استخراج دغدغه‌ها راهکاری نداشته و به تجربه طراح باز می‌گردد.



شکل (۳): تلاقی دو دغدغه در یک کلاس بنا بر نظر بِرگ



شکل (۴): تقسیم دغدغه‌ها به موارد کاربرد

از آنجایی که مورد کاربرد به عنوان دغدغه جزیی مطرح شد، می‌توان روش بِرگ را در بین دو سطح مورد کاربرد و کلاس آزمود. بِرگ وجود تلاقی را منوط به وجود توام پراکندگی و آمیختگی می‌داند. در شکل(۳) مثالی در این باره آورده شده است. در این شکل دغدغه (الف) در کلاس (۱) و دغدغه (ب) در کلاس‌های (۱) و (۲) پیاده‌سازی شده-اند. بدلیل وجود پراکندگی و آمیختگی در کلاس (۱)، بنابر نظر بِرگ تلاقی رخ داده است. حال در صورتی که دغدغه الف با استفاده از مورد کاربرد I ، و دغدغه ب با استفاده از موارد کاربرد II و III همان‌طوری

که در شکل(۴) آورده شده، مدل شده باشند، دیدگاه برگ قابل اعمال در دو سطح مورد کاربرد و کلاس نیست. زیرا این دیدگاه هیچ تلاقی را نشان نمی‌دهد.

واضح است که پراکندگی به تنهایی نمی‌تواند نشان‌دهنده تلاقی باشد، زیرا تقسیم یک مورد کاربرد به چند کلاس در هنگام پیاده‌سازی امری رایج بوده و مشکلی ایجاد نمی‌کند. پس می‌توان گفت تلاقی وابستگی مستقیم با آمیختگی داشته و به عبارتی، آمیختگی شرط لازم برای تلاقی است. برای تشخیص آمیختگی در کلاس‌ها، استفاده از جدول(۱) پیشنهاد می‌شود.

جدول (۱): آمیختگی و پراکندگی در کلاس‌ها

	مورد کاربرد ۱	مورد کاربرد ۲	مورد کاربرد ۳	...	مورد کاربرد n	آمیختگی
کلاس ۱	×					۱
کلاس ۲	×	×	×			۳
کلاس ۳					×	۱
کلاس ۴					×	۱
...						
کلاس m		×				۱
پراکندگی	۲	۲	۱		۲	

در این جدول به سادگی می‌توان دید که در کلاس ۲ آمیختگی رخ داده و این کلاس می‌تواند به عنوان جنبه پیشنهاد گردد. حال اینکه مورد کاربرد n با اینکه در کلاس‌های ۳ و ۴ پراکنده شده، با هیچ کلاس دیگری آمیختگی نداشته و دچار تلاقی نشده است، بلکه تنها یک تقسیم بندی ساده در آن صورت گرفته است.

مرحله پنجم فرآیند پیشنهادی را با توجه به مطالب بالا «تشخیص آمیختگی کلاس‌ها با استفاده از جدول تلاقی» عنوان می‌کنیم.

تعداد موارد کاربرد در آمیختگی در این مرحله با t_i نشان داده شده که یکی از خصوصیات دیگر انتخاب جنبه‌ها است.

۵-۳- بررسی فعال‌سازی توسط کنشگر

در مراحل قبل اطلاعات مربوط به مورد کاربرد استخراج شد. حال خصوصیات کلاس‌ها که برای استخراج جنبه مهم است، باید مد نظر قرار گیرد.

یکی از خاصیت‌های جنبه‌ها این است که «بر مبنای کد در حال اجرا فعال می‌شوند.» از اینرو می‌توان خصوصیت مهم جنبه‌ها را عدم ارتباط مستقیم با کاربر دانست. چون جنبه‌ها توسط کاربر فراخوانی نشده و فعال نمی‌گردند. در [12] این ارتباط در سطح مورد کاربرد بررسی شده است. از آنجایی که ممکن است یک کلاس از یک مورد کاربرد حاوی جنبه بوده و کلاس‌های دیگر آن مورد کاربرد حاوی جنبه نباشند، ممکن است یک مورد کاربرد دارای ارتباط با کنشگر بوده، لیکن کلاسی در آن مورد کاربرد حاوی جنبه باشد. بنابراین بررسی ارتباط کاربر و جنبه باید در سطح کلاس انجام شده و در انتخاب جنبه‌ها این معیار باید به صورت نسبی در نظر گرفته شود.

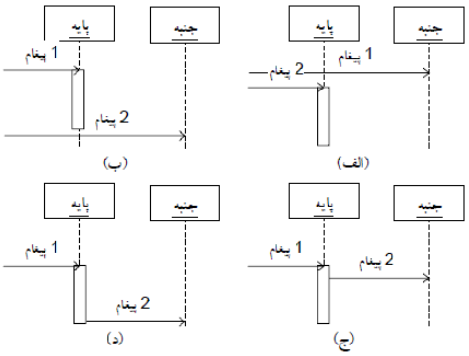
با توجه به نمودار ترتیبی می‌توان عدم ارتباط مستقیم کلاس و کاربر یا همان کنشگر مدل UML را مورد بررسی قرار داد. که مرحله بعدی روند کاری پیشنهادی را تشکیل می‌دهد.

در این مرحله متغیر a که نشان دهنده تعداد فراخوانی کلاس توسط کنشگر می‌باشد، مقدار دهی شده و ضریب آن به صورت منفی در نظر گرفته می‌شود.

۵-۴- بررسی نمودار ترتیبی

در این مرحله یک خصیصه مهم دیگر در مورد جنبه باید بررسی گردد و آن هم‌پوشان، ابطال کننده و یا دربرگیرنده بودن جنبه است. برای استخراج این اطلاعات باید از نمودار ترتیبی استفاده کرده و رابطه کلاس‌های مد نظر به عنوان جنبه را به سایر کلاس‌ها بررسی نمود. «جنبه هم‌پوشان»، قبل یا بعد از اجرای کد پایه و یا در ابتدا یا انتهای آن اجرا می‌شود. بنابراین پیغام‌های نمودار ترتیبی به کلاس حاوی جنبه باید قبل از اجرای کلاس پایه و یا بعد از آن ارسال شوند.

در شکل(۵) قسمت‌های (الف) و (ب) نشان‌دهنده حالتی است، که هم شیء پایه و هم شیء جنبه توسط یک شیء دیگر راه‌اندازی شده‌اند. در حالت (الف) جنبه، قبل از شیء پایه فراخوانی شده بنابراین جنبه قبل، است. در حالت (ب) جنبه، بعد از شیء پایه فراخوانی شده، بنابراین جنبه بعد، است. در حالت‌های (ج) و (د)، جنبه توسط خود شیء پایه فراخوانی شده است. در حالت (ج)، جنبه قبل و در حالت (د)، جنبه بعد، رخ داده است. تمامی این حالات نشان دهنده احتمالی نوع هم-پوشان هستند.



شکل (۵): حالت‌های مختلف جنبه هم‌پوشان

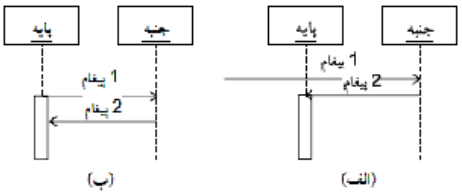
«جنبه ابطال کننده» جایگزین کد اصلی می‌گردد. معمولاً این نوع جنبه در طراحی اولیه سیستم کمتر وجود داشته و در صورت اعمال تغییرات در سیستم اولیه بروز می‌کند. در نمودار ترتیبی می‌توان چنین جنبه‌هایی را با استفاده از پیغامی که از طرف شیء جنبه به شیء پایه بازگردانده می‌شود، شناسایی کرد. زیرا این پیغام می‌تواند حاوی متغیری باشد که با توجه به آن کلاس یا شیء پایه اجرا شده یا نشود. دراین صورت جنبه باید حاوی کد جایگزین برای شیء اصلی بوده و یا حداقل مسیر جدیدی را در شیء پایه فعال کند.

برای اطمینان از درنظر گرفتن تمامی حالات، سناریوهای مختلف یک مورد کاربرد باید بررسی شده و در صورتی که یک کلاس یا شیء به عنوان تصمیم گیرنده در مسیر اجرای سناریو وجود داشته باشد، این کلاس می‌تواند به عنوان جنبه در نظر گرفته شود. در این حالت،

اطلاعات مشابهی در نمودار Statechart هم وجود داشته و می‌توان با استفاده از آن نمودار این اطلاعات را استخراج کرد.

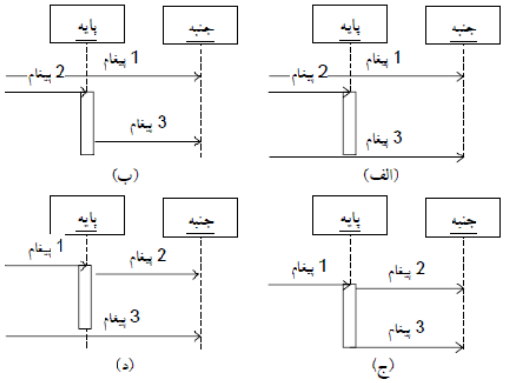
شکل(۶) حالت‌های احتمالی جنبه ابطال‌کننده را نشان می‌دهد. در حالت (الف) جنبه توسط شیء دیگری فعال شده و پیغامی را جهت کنترل مسیر اجرا به شیء پایه ارسال می‌کند، در حالت (ب) جنبه توسط خود شیء پایه فعال شده و این شیء با توجه به پاسخ جنبه مسیر اجرای خود را مشخص می‌کند.

حالت‌های دیگری نیز برای این نوع جنبه متصور است. بطور کلی هرگاه در هنگام اجرای شی پایه پیغام جنبه، دریافت شود، امکان وجود جنبه ابطال کننده وجود دارد، حتی در صورتی که این پیغام توسط یک یا چندین واسطه رد و بدل شود.



شکل (۶): حالت‌های مختلف جنبه ابطال کننده

«جنبه دربرگیرنده»، شیء پایه را احاطه می‌کند و در نمودار ترتیبی هم قبل و هم بعد از شیء پایه به آن پیغام ارسال شده است. شکل(۷) حالت‌های احتمالی جنبه دربرگیرنده را نشان می‌دهد. در حالت (الف) شروع و پایان فعالیت شیء پایه، توسط شیء دیگری به جنبه اطلاع داده شده و در حالت (ب) شروع، توسط شی دیگر و پایان، توسط خود شی پایه مشخص شده است.در حالت (ج) شروع و پایان را خود شی پایه به جنبه خبر داده و در حالت (د)، شروع توسط شی پایه و پایان توسط شی دیگری مشخص شده‌است.

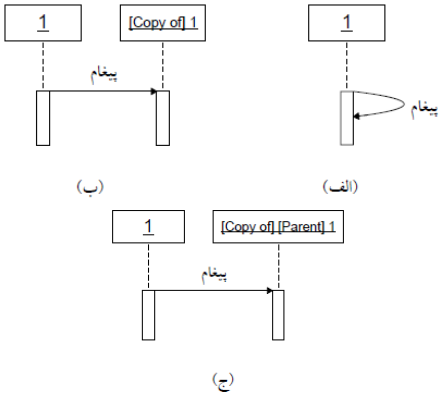


شکل (۷): حالت‌های مختلف جنبه دربرگیرنده

این مرحله را «بررسی جنبه‌ها در نمودار ترتیبی» نام گذاری می‌کنیم. نکته حایز اهمیت در اینجا عدم توجه به زمان اجرای شی پیشنهادی به عنوان جنبه و تنها توجه به پیغام‌های رد و بدل شده می‌باشد. این مساله به دلیل تفاوت در طراحی‌ها و مشخص نبودن زمان اجرای جنبه است. مثلاً یک جنبه دربرگیرنده که باید قبل و بعد از اجرای کد پایه اجرا شود، ممکن است در طراحی اولیه در تمام مدت زمان اجرای کد پایه فعال بوده و زمان اجرایی معادل با زمان اجرای کد پایه داشته‌باشد.

مساله مهم دیگر در بررسی نمودار ترتیبی، توجه به پیغام‌هایی است، که شیء برای خودش ارسال می‌کند. در شکل(۸) قسمت (الف) این نوع پیغام نمایش داده شده است. در این شکل شیء (۱) یک پیغام بازگشتی ارسال کرده است.

برای اعمال روش بیان شده در این قسمت بر پیغام‌های بازگشتی، باید این نوع پیغام‌های را به‌صورت یک پیغام بین دو شیء در نمودار ترتیبی تصور کرد، شیء اول ارسال کننده پیغام و شیء دوم نیز به‌عنوان یک شیء فرضی در نمودار ترتیبی آورده شده می‌شود. در شکل(۸) قسمت (ب) نمودار ترتیبی تغییر یافته نشان داده شده‌است.



شکل (۸): پیغام بازگشتی

بررسی رابطه ارث‌بری نیز در این نمودار اهمیت بسیاری دارد، چرا که معمولاً خصوصیات مشترک که ممکن است حاوی جنبه باشند در یک کلاس در سطح بالاتر جمع شده و به کلاس‌های پایین‌تر به ارث می‌رسد. برای در نظر گرفتن این مساله بهتر است در نمودار ترتیبی هر مورد کاربردی که به کلاس پدر مربوط می‌شود، کلاس پدر به‌جای کلاس فرزند آورده شود. در صورتی که این مساله در هنگام ساختن مدل رعایت نشود، طراح در هنگام انتخاب کلاس‌های حاوی جنبه باید به رابطه ارث‌بری بین کلاس‌ها توجه داشته باشد، و خصوصیات مشترک چند کلاس فرزند را مربوط به کلاس پدر ببیند.

در مورد مدل سازی پیغام‌های بازگشتی، در صورتی که پیغام بازگشتی برای رویه‌ای در شیء یا کلاس پدر ارسال شده بود، شیء فرضی اضافه شده در نمودار ترتیبی، باید شیء پدر باشد. این مساله در شکل(۸) قسمت (ج) نشان داده شده است.

در این مرحله تعداد هرکدام از الگوهای مطرح شده برای هر کلاس مشخص می‌گردد. تعداد الگوهای فراخوانی هم‌پوشان در متغیر l، تعداد الگوهای فراخوانی ابطال کننده در متغیر r و تعداد الگوهای فراخوانی دربرگیرنده در متغیر p قرار می‌گیرند.

۵-۵- انتخاب جنبه‌ها

حال می‌توان کلاس‌ها را در جدول(۲) قرار داده و بر اساس خصوصیات بدست آمده در مراحل قبل آنها را رتبه‌بندی کرده، و از میان آنها جنبه‌های مورد نظر را انتخاب نمود.

برای ارزیابی جنبه‌ها کافی است، این جدول با توجه به اهمیت هر کدام از خصوصیت‌ها، توسط طراح سیستم و با توجه به نیازمندی‌های سیستم، مرتب شود. سپس به راحتی از میان کلاس‌هایی که رتبه بالاتری را اتخاذ نموده‌اند، کلاس‌هایی جهت تبدیل به جنبه انتخاب شوند.

جدول (۲): خصوصیات انتخاب جنبه‌ها

ردیف	خصوصیت	کلاس ۱	کلاس ۲	...	کلاس n
۱	تعداد رابطه extend در مورد کاربرد				
۲	تعداد مواردکاربرد در آمیختگی				
۳	تعداد فعال‌سازی توسط کنشگر				
۴	تعداد هم‌پوشانی				
۵	تعداد ابطال کنندگی				
۶	تعداد دربرگیرندگی				

حال خصوصیات موجود در این جدول را به صورت یک فرمول ریاضی درآورده و برای هر خصوصیت یک ضریب در نظر می‌گیریم. این فرمول که بر مبنای WMC یا Weighted Method per Class بنا شده‌است، بدین صورت می‌باشد:

$$CAR_i = \sum w_{x_i} f_{x_i} | x \in \{e,t,a,l,r,p\} \quad [۱۴]$$

و یا به عبارت دیگر

$$CAR_i = w_{e_i} f_{e_i} + w_{t_i} f_{t_i} + w_{a_i} f_{a_i} + w_{l_i} f_{l_i} + w_{r_i} f_{r_i} + w_{p_i} f_{p_i}$$

در این فرمول متغیرهای مربوط به هر مرحله، برای هر کلاس و w_{x_i} که ضریب‌های آنها است و بنا بر اهمیت هر خصوصیت به آن اختصاص می‌یابد، گنجانده شده است. سپس عدد CAR که نشان دهنده امتیاز کلی هر کلاس است، برای آن محاسبه می‌گردد. در نهایت می‌توان با استفاده از مرتب‌سازی همین عدد به انتخاب جنبه‌ها پرداخت.

مقادیر متغیرهای هر مشخصه در فرمول باید به عددی بین صفر و یک تبدیل شود تا در هنگام جمع کردن آنها با یکدیگر تنها ضریب متغیرها، نشان‌دهنده وزن آن متغیر بوده و بزرگی مقدار تاثیری در اهمیت متغیر نداشته باشد. برای این کار باید مقادیر متغیرها را بر بزرگترین مقدار موجود در این سری متغیرها تقسیم کرد. بدین صورت:

$$f_{x_i} = \frac{x_i}{Max(x_i)}$$

ضرایب w_{x_i} در این فرمول باید توسط طراح سیستم یا روش‌های موجود در شبکه‌های عصبی تعیین شوند. استفاده از روش‌های شبکه‌های عصبی برای تعیین ضرایب، نیازمند نمونه‌های زیادی از طراحی شی‌گرایی تبدیل شده به جنبه‌گرا است، که متاسفانه چنین نمونه‌هایی در حال حاضر در دست نیست.

در حالت کلی می‌توان خصوصیت دوم یعنی تعداد آمیختگی را دارای اهمیت بالاتری دانسته و خصوصیت سوم یعنی تعداد رابطه مستقیم با کاربر را به عنوان یک خصوصت منفی بیان کرد، بدین صورت که هرچه تعداد آمیختگی بیشتر باشد، احتمال جنبه بودن کلاس هم بیشتر است، این کلاس نباید توسط کاربر فراخوانی شود. لیکن در حالت‌هایی که طراحی اولیه سیستم نامناسب بوده و تعداد آمیختگی در کلاس زیاد است ممکن است، ارتباط مستقیم بین کلاس و کاربر نیز وجود داشته باشد، که در هنگام تعیین معیار جهت استخراج جنبه، فعال‌سازی توسط کنشگر در مدل، حتماً باید از نظر معنی ارتباط توسط طراح سیستم بازنگری شود.

خصوصیت اول یعنی تعداد رابطه extend در مورد کاربرد، که از بررسی مدل مورد کاربرد در این قسمت آورده شده با خصوصیت دوم هم‌سو بوده و تنها برای اطمینان آورده شده است. بنابراین می‌توان به آن اهمیتی در حد متوسط داد.

خصوصیت چهارم تا هفتم، که نوع جنبه را مشخص می‌کنند، در [12] به عنوان مشخصه اصلی جنبه‌ها آورده شده است. دلیل این امر، ثابت بودن نحوه اجرای جنبه‌ها بیان شده است. یعنی کلاسی که به عنوان جنبه در نظر گرفته می‌شود باید همیشه با یک الگو فراخوانی شود. در غیر این‌صورت جنبه نیست. در روش ما این مساله بدین شکل استفاده نشده، و دلیل آن نیز وجود عملگر «یا» در هنگام تعریف محل برش است. با استفاده از این عملگر می‌توان چندین نقطه پیوست را تعریف نمود. توصیه‌ها نیز می‌توانند به طرق مختلف در یک جنبه تعریف شوند. بنابراین یک جنبه می‌تواند با الگوهای مختلف فراخوانی و اجرا شود. مساله مهم در اینجا تعداد الگوهای مشابه است که به طراح کمک می‌کند تا انتخاب مناسبی داشته باشد.

۵-۶- ارایه اطلاعات جنبه‌ها

در آخر پس از انتخاب جنبه‌ها آنها را در جدولی مانند جدول(۳) وارد ساخته تا به راحتی بتوان از آنها در هنگام طراحی مجدد و نیز پیاده‌سازی استفاده نمود.

جدول (۳): اطلاعات جنبه‌ها

نام جنبه		
۱	نام مورد کاربرد اصلی	
۲	نام موارد کاربرد مورد تاثیر	
۳	نام کلاس اصلی	
۴	نام کلاس‌های مورد تاثیر	
۵	نوع جنبه	
۶	اهمیت جنبه	

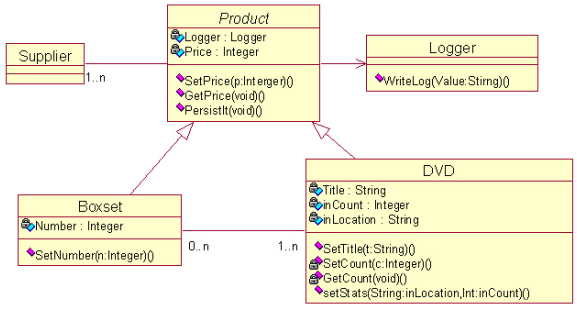
در این جدول ردیف اول، حاوی نامی است که طراح برای جنبه انتخاب خواهد کرد. در ردیف دوم نام مورد یا موارد کاربردی است که جنبه از آن نشات گرفته است. ردیف سوم نام مورد کاربردهایی که جنبه بر آنها تاثیر گذاشته را در بر می‌گیرد. در ردیف چهارم نام کلاسی که جنبه در آن وجود دارد درج می‌شود. ردیف پنجم نام کلاس‌هایی که جنبه بر آنها تاثیر می‌گذارد را مشخص می‌کند. در ردیف ششم نوع جنبه به عنوان هم‌پوشان، ابطال‌کننده و یا دربرگیرنده که در قسمت‌ها قبل مشخص شده، بیان می‌گردد. در نهایت در ردیف هفتم اهمیت جنبه توسط طراحی سیستم مشخص شده تا در صورت تاثیر دو جنبه بر یک نقطه و یا ناسازگاری احتمالی، جنبه مهم‌تر غالب باشد.

۶- مثال استخراج جنبه از مدل طراحی

در این قسمت کارایی روش پیشنهادی در عمل بر روی یک سیستم نرم‌افزاری نشان داده شده است. به عنوان نمونه از سیستم یک فروشگاه دیسک نوری که در [13] و [11] مورد استفاده قرار گرفته است، بهره می‌گیریم. قسمتی از مشخصات سیستم مذکور بدین شکل است:

- در این سیستم چندین دیسک نوری که نام و تعداد آنها باید در سیستم نگهداری شود، وجود دارند.
- مجموعه‌ای از دیسک‌های نوری با عنوان بسته ارایه شده است.
- قیمت هر محصول در سیستم باید نگهداری شود.
- تغییرات قیمت که کاربران مجاز در سیستم اعمال می‌کنند، باید ثبت گردد.
- زمان دسترسی کاربران به سیستم باید ثبت گردد.

نمودار کلاس‌های این سیستم به‌صورتی که در شکل(۹) نشان داده شده، معرفی شده است[11][13].



شکل (۹): نمودار کلاس‌های سیستم فروش دیسک نوری

با اجرای فرآیند استخراج جنبه در این مثال و جمع آوری اطلاعات بدست آمده، جدول(۴) به‌دست می‌آید.

در صورتی که خواص ردیف اول و دوم را مهم و بسیار مهم و خواص سایر ردیف‌ها را با اهمیت متوسط در نظر بگیریم. و مقادیر ضرایب نشان‌دهنده وزن هر مشخصه بدین شکل مشخص گردند.

«بسیار مهم=۱ ، مهم=۰/۷۵ ، اهمیت‌متوسط=۰/۵ و کم‌اهمیت=۰/۲۵»

جدول (۴): خصوصیات کلی کلاس‌ها

ردیف	خصوصیت	Logger	Product	DVD	Supplier	Boxset
۱	تعداد رابطه extend موردکاربرد	3	5	3	0	0
۲	تعداد مواردکاربرد در آمیختگی	0	5	3	0	0
۳	تعداد فعال‌سازی توسط کنشگر	0	0	1	0	0
۴	تعداد هم‌پوشانی	6	3	0	0	0
۵	تعداد ابطال کنندگی	0	0	0	0	0
۶	تعداد دربرگیرندگی	0	2	0	0	0

مقادیر CAR را برای کلاس‌ها این چنین محاسبه می‌کنیم:

$$CAR_{product} = 0.75(1) + 1(1) - 0.5(0) + 0.5(0.5) + 0.5(0) + 0.5(1) = 2.5$$
$$CAR_{Logger} = 0.75(0.6) + 1(0) - 0.5(0) + 0.5(1) + 0.5(0) + 0.5(0) = 0.95$$
$$CAR_{DVD} = 0.75(0) + 1(0.6) - 0.5(1) + 0.5(0) + 0.5(0) + 0.5(1) = 0.1$$

واضح است که دو کلاس Logger و Product حاوی جنبه هستند. در مرحله آخر اطلاعات هر جنبه، در جدول مربوط به ارایه اطلاعات جنبه‌ها جمع‌آوری می‌شود.

جدول (۵): اطلاعات جنبه تایید هویت

Authenticate User		
۱	نام مورد کاربرد اصلی	Authentication
۲	نام مورد کاربردهای مورد تاثیر	Enter Product Price Read Product Price Sell DVD
۳	نام کلاس اصلی	Product
۴	نام کلاس‌های مورد تاثیر	Product DVD
۵	نوع جنبه	Overlapping
۶	اهمیت جنبه	High

جدول (۶): اطلاعات جنبه ثبت تغییرات قیمت

Log Price Change		
۱	نام مورد کاربرد اصلی	Log Changes
۲	نام مورد کاربردهای مورد تاثیر	Enter Product Price Read Product Price Sell DVD Authentication
۳	نام کلاس اصلی	Logger
۴	نام کلاس‌های مورد تاثیر	Product DVD
۵	نوع جنبه	Overlapping
۶	اهمیت جنبه	Average

در جدول(۵) اطلاعات مربوط به جنبه تایید هویت آورده شده است. در جدول(۶) اطلاعات مربوط به جنبه ثبت تغییرات قیمت و در جدول(۷) اطلاعات مربوط به جنبه ثبت زمان دسترسی نشان داده شده‌است.

این مثال در [11] نیز مورد بررسی قرار گرفته و جنبه‌ها به روش پیشنهادی آن استخراج شده‌اند. در این منبع تنها ثبت تغییرات قیمت به عنوان جنبه پیشنهاد شده‌است. هیچ کدام از جزییات ارایه شده در جدول(۶) نیز در این روش مورد بررسی قرار نگرفته است.

جدول (۷): اطلاعات جنبه ثبت زمان دسترسی

Log Access Time		
۱	نام مورد کاربرد اصلی	Log Changes Enter Product Price Sell DVD
۲	نام مورد کاربردهای مورد تاثیر	Enter Product Price Sell DVD
۳	نام کلاس اصلی	Product Logger
۴	نام کلاس‌های مورد تاثیر	Product DVD
۵	نوع جنبه	Wrapping
۶	اهمیت جنبه	Average

این مثال نشان می‌دهد، حتی در سیستم‌هایی که به درستی طراحی نشده‌اند نیز فرآیند پیشنهادی این مقاله کارایی مناسبی دارد. مثلاً در سیستم ارایه شده ثبت زمان دسترسی و تایید هویت کاربران باید در کلاس دیگری قرار می‌گرفتند، که به دلیل طراحی نادرست، تمامی این موارد در کلاس product گنجانده شده بودند. با این حال اعمال روش پیشنهادی، منجر به استخراج این جنبه‌ها گردید.

۷- ارزیابی روش ارایه شده

در قسمت قبل نشان داده شد که روش پیشنهادی اعمال شده بر روی مثال مشترک با [11]، جنبه‌های بیشتری را با جزییات بیشتر، نسبت به روش ارایه شده در [11] مشخص می‌کند. حال برای ارزیابی بهتر روش پیشنهادی، که از این پس با نام روش «استخراج جنبه از مدل طراحی» به آن اشاره می‌کنیم، به مقایسه این روش با روش‌های مشابه خواهیم پرداخت.

۷-۱- معیارهای مقایسه

معیارها را می‌توان در دو سطح بررسی نمود. یک سطح بررسی دغدغه-ها در نیازمندی‌ها است و دیگری بررسی جنبه‌ها در سطح طراحی است.

در سطح نیازمندی‌ها اولین معیاری که می‌توان مطرح کرد، «بررسی مدل موارد کاربرد» است. عناصر این مدل به دغدغه‌ها نزدیک‌تر بوده و بررسی این مدل در اصل بررسی دغدغه‌ها است. دومین معیار در این سطح «توجه به رابطه extend» است. این رابطه مفهوم افزودن یک کاربرد به کاربرد دیگر را می‌رساند که مفهومی نزدیک به جنبه است. معیار سوم «در برگیری دغدغه‌های غیرکارکردی» است. بیشتر جنبه‌ها از این نوع از دغدغه‌ها نشات می‌گیرند.

در سطح طراحی «بررسی الگو فراخوانی جنبه‌ها» اولین معیاری است که می‌توان به آن اشاره کرد. زیرا الگوی فراخوانی جنبه‌ها یکی از مهمترین خصوصیات جنبه‌ها می‌باشد که در نمودار ترتیبی نمایان است. در بررسی این الگو، «بررسی رابطه ارث‌بری» بین کلاس‌ها و «بررسی پیغام‌های بازگشتی در نمودار ترتیبی» اهمیت دارد. زیرا معمولاً خصوصیات مشترک که حاوی جنبه هستند یک کلاس قرار گرفته و بقیه از آن ارث می‌برند. در مورد پیغام‌های بازگشتی هم در صورتی که جنبه در سطح رویه وجود داشته باشد و در خود کلاس گنجانده شده باشد، بررسی این پیغام‌ها لازم است. «بررسی تلاقی در سطح کلاس‌ها» از جهت کشف آمیختگی یکی از مراحل مهم کشف جنبه است. «تشخیص نوع جنبه» در روش، اعم از هم‌پوشان، ابطال کننده یا دربرگیرنده نیز مورد مقایسه قرار گرفته است. «ابزار طراحی جنبه‌ها» و وارد ساختن جنبه‌های استخراج شده در مدل طراحی نیز اگرچه به مراحل بعد از استخراج جنبه‌ها مربوط می‌شود، به عنوان یک معیار جنبی در روش‌ها مورد مقایسه قرار گرفته است.

۷-۲- روش‌های انتخاب شده برای مقایسه

روش‌هایی که در این قسمت برای مقایسه با روش «استخراج جنبه از مدل طراحی» انتخاب شده‌اند، همگی نسبتاً دارای راهکارهای عملی هستند. این روش‌ها از لحاظ معیارهای بیان شده در قسمت قبل، با روش «استخراج جنبه از مدل طراحی» قابل مقایسه هستند. روش‌های انتخاب شده بدین شرح است:

۱. روش «شناخت تلاقی در طراحی نرم‌افزار» [11]
۲. روش «تشخیص جنبه‌های اولیه در پشتیبانی از طراحی» [12]
۳. روش «Theme» [2]
۴. روش «Early-AIM» [16]
۵. روش «جداسازی دغدغه‌های متلاقی در نیازمندیها و طراحی» [10]

روش‌های ۱،۲و۵ مشخصاً به استخراج جنبه از مدل طراحی پرداخته و روش ۴ مستندات نیازمندی‌ها را بررسی می‌کند. روش ۳ مشخصاً بر استخراج جنبه‌ها تمرکز نداشته، و به‌خاطر جامع بودن روش و نیز وجود روشی برای بررسی مستندات نیازمندی‌ها در نظر گرفته شده است.

در جدول(۸) این روش‌ها به ترتیب شماره‌های بالا آورده شده و «روش استخراج جنبه از مدل طراحی» با شماره ۶ مشخص شده است.

جدول (۸): مقایسه روش‌های استخراج جنبه

معیارها	(۱)	(۲)	(۳)	(۴)	(۵)	(۶)
بررسی مدل موارد کاربرد	✓	✓	×	×	✓	✓
توجه به رابطه extend	×	✓	×	×	×	✓
در برگیری دغدغه‌های غیرکارکردی	×	✓	✓	✓	✓	✓
بررسی الگو فراخوانی جنبه‌ها	×	✓	×	×	✓	✓
بررسی رابطه ارث‌بری	×	×	×	×	×	✓
بررسی پیغام‌های بازگشتی در نمودار ترتیبی	×	×	×	×	×	✓
بررسی تلاقی در سطح کلاس‌ها	✓	×	×	×	×	✓
تشخیص نوع جنبه	×	×	×	×	✓	✓
ابزار طراحی جنبه‌ها	×	×	✓	×	×	×

روش (۱) بر بررسی تلاقی در دو سطح مختلف تکیه داشته و به جزییات جنبه‌ها توجهی ندارد. روش (۲) و (۵) نیز مدل موارد کاربرد و نمودار ترتیبی را تک تک مورد بررسی قرار داده و مانند روش (۱) تلاقی را در دو سطح بررسی نکرده‌اند. در ضمن این روش‌ها در هنگام بررسی نمودارهای ترتیبی، رابطه ارث بری و نیز پیغام‌های بازگشتی را مد نظر قرار نمی‌دهند. روش (۴) نیز از راهکارهای پردازش زبان استفاده کرده و به بررسی مستندات نیازمندی‌ها می‌پردازد. در (۳) نیز بیشتر ارایه راهکاری جهت انجام طراحی جنبه‌ها مطرح بوده و استخراج جنبه‌ها هدف اصلی نیست. این روش ابزاری جهت بررسی مستندات نیازمندی‌ها ارایه کرده است.

۸- خلاصه مطالب و فعالیت‌های آینده

در مراحل فرآیند پیشنهادی، ابتدا در مورد مدل‌سازی دغدغه‌های غیرکارکردی بحث شده و روش‌های [10]، [14] و [15] که روابط جدیدی را برای مدل‌سازی دغدغه‌های غیرکارکردی پیشنهاد می‌کنند، بررسی شده‌اند. سپس در مورد اشکالات این روش‌ها بحث شده و استفاده از راهکارهایی چون SIG و موارد سوکاربرد پیشنهاد گردیده است. در ادامه، بکارگیری رابطه extend برای افزودن کارکردهای جدید به مدل موارد کاربرد، استفاده شده و پیشنهاد شده که روابط این مدل با استفاده از راهکار موجود در [8] اصلاح گردد. در ادامه معیارهایی برای ارزیابی احتمال وجود جنبه در هر کلاس ارایه شده،

که در نهایت به‌صورت یک فرمول جمع‌بندی شده‌است. این فرمول به‌صورت مجموع متغییرهای وزن‌دار بیان شده‌است.

اولین معیار ارایه شده، تعداد روابط extend در مدل مورد کاربرد است که به موارد کاربرد نسبت داده می‌شود. سپس طراح سیستم مشخص می‌کند که کدام مورد کاربرد در کدام کلاس‌ها پیاده‌سازی شده و از این طریق تعداد روابط extend از موارد کاربرد به کلاس‌ها منتقل می‌شود.

سپس به مبحث تلاقی پرداخته و روش آقای بِرگ [11]، که در دو سطح دغدغه و کلاس اعمال می‌شود، بیان شده است. با استفاده از راهکار وی، روش جدیدی در دو سطح مواردکاربرد و کلاس پیشنهاد شده و مورد استفاده قرار گرفته است. در ادامه، بررسی فعال‌سازی توسط کنشگر مطرح شده و راهکار موجود در [12] که این مساله را در مدل مورد کاربرد مورد بررسی قرار داده، بهینه شده است. سپس بررسی فعال‌سازی توسط کنشگر، در سطح کلاس و با توجه به نمودار ترتیبی انجام شده است. در مرحله بعدی الگوهای فراخوانی و فعال‌شدن جنبه‌ها در نمودار ترتیبی پیشنهاد شده و نوع جنبه در این نمودارها مشخص شده است. برای بررسی پیغام‌های بازگشتی و رابطه ارث‌بری نیز راهکارهایی پیشنهاد شده است. در آخرین مرحله جدولی برای نگهداری اطلاعات جنبه پیشنهاد شده و در نهایت مقایسه‌ای بین روش‌های دیگر و روش ارایه شده در این مقاله صورت گرفته است.

در ادامه فعالیت‌های ارایه شده در این مقاله می‌توان ضرایب فرمول ارایه شده را با استفاده از روش‌های موجود در شبکه‌های عصبی مشخص نمود. در صورت مشخص شدن ضرایب فرمول برای استخراج جنبه‌ها می‌توان دخالت طراح سیستم را در روش، به حداقل رساند.

همچنین می‌توان یک روش طراحی بهینه جنبه‌گرا انتخاب و یا ارایه نمود. سپس برای تبدیل این جدول به طراحی جنبه‌گرا روشی پیشنهاد نمود، که همان عمل Refactoring است. در نهایت می‌توان به مساله تداخل جنبه‌ها پرداخته و با استفاده از راهکارهای موجود مانند توصیف رسمی، تداخل جنبه‌ها را بررسی کرده و راهکارهایی برای رفع آن ارایه نمود.

مراجع

[1] Kiczales Gregor, Lamping John, Mendhekar Anurag, Maeda Chris, Videira Lopes Cristina, Loingtier Jean-Marc, Irwin John. *Aspect-Oriented Programming.* Springer-Verlag LNCS 1241, P. 220-242, 1997.

[2] Clarke Siobhan, Baniassad Elisa. *Aspect-Oriented Analysis and Design: The Theme Approach.* Addison Wesley, 2005.

[3] Chitchyan Ruzanna et al, *Survey of Aspect-Oriented Analysis and Design Approaches.* http://www.aosd-europe.net, 2005.

[4] Stein D., Hanenberg St., Unland. R. *Designing Aspect-Oriented Crosscutting in UML.* AOSD-UML Workshop at AOSD' 02 Enschede, the Netherlands, 2002.

[5] Robillard M., Murphy. G. *Concern graphs: Finding and describing concerns using structural program dependencies.* International Conference on Software Engineering.P.406-416, 2002.

[6] Tellander David. *Aspect Detector: A tool for automatic aspect detection in Java code.* Master Thesis, Vaxjo University, Sweden, 2005.

[7] Baniassad E., Clements P., Araújo J., Moreira A., Rashid A., Tekinerdogan B., *Discovering Early Aspects.* IEEE Software, Special Issue on Aspect-Oriented Programming, P. 61-70, January/February 2006.

[8] Jacobson Ivar, Pan-Wei Ng, *Aspect-Oriented Software Development with Use Cases*. Addison Wesley, 2004.

[9] Alexander Ian. *Misuse Cases: Use Cases with Hostile Intent.* IEEE Software, 20(1), P. 58-66, January/February 2003.

[10] Sousa G., Soares S., Borba P., and Castro J. *Separation of crosscutting concerns from requirements to design: Adapting the use case driven approach.* Early Aspects Workshop at AOSD 2004, P. 96-106, March 2004.

[11] Van den Berg Klaas et al, *Identification of Crosscutting in Software Design.* 8th International Workshop on Aspect-Oriented Modeling at AOSD'06, Bonn, Germany, March 2006.

[12] Keuler Thorsten, Naab Matthias, *Supporting Architectural Design by Early Aspects Identification.* 8th International Workshop on Aspect-Oriented Modeling at AOSD'06, Bonn, Germany, March 2006.

[13] Gradecki Joseph D., Lesiecki Nicholas. *Mastering AspectJ Aspect-Oriented Programming in Java.* Wiley. 2003.

[14] Araújo J., Moreira A., *An Aspectual Use Case Driven Approach.* VIII Jornadas de Ingeniería de Software y Bases de Datos (JISBD), Alicante, Spain, 12-14 November 2003.

[15] Araújo J., Moreira A., I. Brito, A. Rashid, *Aspect-Oriented Requirements with UML*. Workshop on Aspect-oriented Modeling with UML, UML 2002, Dresden, Germany, October 2002.

[16] Sampaio A., Rashid A., Rayson P., *Early-AIM: An Approach for Identifying Aspects in Requirements.* Requirements Engineering Conference, Paris, France, 2005.

[17] Pressman R.S., *Software Engineering: A Practitioner's Approach, 5'Th Edition*, McGraw-Hill, 2003.

زیرنویس‌ها

[1] Crosscutting Concern
[2] Tangling
[3] Separation of Concerns
[4] Gregor Kiczales
[5] Joinpoint
[6] Pointcut
[7] Advice
[8] Siobhan Clarke
[9] Dominik Stein
[10] Early Aspect
[11] Weighted Method per Class (WMC) [17]
[12] Non-functional
[13] Scattering
[14] Class Aspect Ration