# استخراج جنبه ها از متن زبان جریان کاری BPEL

سعید پارسا دانشگاه علم و صنعت ایران parsa@iust.ac.ir

مریم افرادی دانشگاه علم و صنعت ایران afradi@comp.iust.ac.ir

## چکیده

**روش های مختلفی برای استخراج جنبه از متن برنامه ها مطرح می باشد. در این مقاله این موضوع بررسی می شود که یک جنبه را می توان در قالب یک زیرگراف از گراف فراخوانی در نظر گرفت که یک نقطه دسترسی دارند که اگر فراخوانی ای از داخل این زیرگراف به خارج وجود داشته باشد، توابعی مورد فراخوانی قرار می گیرند که مشترک هستند و از مکان های مختلف به آنها ارجاع شده است. برای اثبات صحت این ادعا ، زبان BPEL را انتخاب و مبادرت به تعیین جنبه ها در برنامه های نوشته شده به زبان BPEL نموده ایم. برای اعتبارسنجی این مورد ، مدلی ارائه دادیم که ابزار استخراج جنبه نامیده شده است. این ابزار، جنبه‌های جریان کاری را از فرایند های BPEL جدا می‌کند.**

**واژه های کلیدی : جنبه ، برنامه نویسی جنبه گرا ، وب سرویس، استخراج جنبه ، BPEL**

## ۱- مقدمه

جنبه به بخشی از برنامه اطلاق می شود که هدف خاصی را دنبال نموده و از دیدگاه کاربر یک مفهوم قابل شناسایی است. برای نمونه ثبت سند یک جنبه از نرم افزار حسابداری است که در واقع یک جنبه را مشخص می کند. مشکل برنامه های شی گرا پراکندگی جنبه ها در قالب متد های کلاس ها است. در واقع یک جنبه درقالب یک دیاگرام همکاری قابل مدلسازی است. دیاگرام های همکاری چگونگی همکاری بین متد ها جهت رسیدن به اهداف جنبه را مشخص می کنند.

هدف در برنامه سازی جنبه گرا تشخیص و تفکیک دغدغه ها و جنبه های مختلفی است که در یک برنامه مطرح است. در روش های برنامه نویسی قدیمی کدهای مربوط به این دغدغه ها می توانست در پیمانه‌های مختلف به صورت پراکنده و درهم و تکراری پیاده سازی شود، زیرا مفهومی به نام جنبه یا دغدغه مطرح نبود و ساختار خاصی در این دسته از زبانها ایجاد نگردید. برای جداسازی چنین دغدغه ها ی تداخلی، چندین تکنیک و شیوه مطرح شده که برخی از مهمترین این شیوه ها به شرح زیر می باشند:

### ۱- ۱ تحلیل متد های تکراری ردیابی اجرایی

در این روش کاندیدهای جنبه را مبتنی بر الگوهای تکراری فراخوانی متدها کشف می‌کند. . این شیوه بر اجرای متدها تمرکز دارند. مشکلی که در آنها وجود دارد این است که وراثت و چندشکلی را به دقت بررسی نمی‌نمایند. [۳]

### ۲ -۱ تحلیل مفهوم رسمی در ردیابی‌های اجرایی

این روش یک تکنیک استخراج جنبه از طریق تحلیل مفهوم رسمی در ردیابی اجرایی می‌باشد. مفاهیم ایجاد شده، گروه هایی از اشیا هستند که مشخصات مشترک دارند. این شیوه نیز همانند روش قبلی، بر

ردیابی اجرا یی تمرکز دارد. مشکلی که در آن وجود دارد عدم بررسی دقیق وراثت و چندشکلی می باشد.[۴]

### ۳-۱ تحلیل مفهوم رسمی در نام کلاس ها و متدها

این تکنیک نیز مبتنی بر تحلیل مفهوم رسمی بود.در این شیوه ،عمل تحلیل شناسه انجام می شود که با استفاده از الگوریتم تحلیل مفهوم رسمی این کار انجام می گیرد. این شیوه بر اساس قراردادهای نام‌گذاری در کلاس‌ها و متدهای سیستم می باشد. یکی از مشکلات عمده‌ای که در این شیوه وجود دارد این است که به کدنویسی برنامه‌نویسان وابسته است و بطور دقیق قادر به شناسایی وراثت و چندشکلی نیست. وهمچنین به کندی کار می‌کند.[۵]

### ۴-۱ پردازش زبان طبیعی

این تکنیک با استفاده از نام‌گذاری و قوانین رمزگذاری پیاده‌سازی شده است. این روش به عنوان ورودی، مجموعه‌ای از کلمات را می‌گیرد و خروجی آن نیز مجموعه ای از کلمات می‌باشند که با هم وابستگی محکم دارند. بدین‌ترتیب ، به الگوریتمی برای بررسی مفهومی این واژه‌ها نیاز است. این روش نیز به درستی قادر به شناسایی وراثت و چندشکلی نمی‌باشد.[۶]

با بررسی تکنیک های موجود به این نتیجه دست می یابیم که تمامی تکنیک ها دارای ضعف ها و قدرت هایی هستند و اغلب آنها وراثت و چند شکلی را پشتیبانی نمی کنند. ما در طرح پیشنهادی خود جنبه را دنباله ای از فراخوانی ها در نظر می گیریم که در داخل مجموعه خود یک سری توابع و یا متد ها رافراخوانی می کنند ولی هیچ فراخوانی به خارج از مجموعه خود ندارند مگر به یک سری توابع خاص ویا سیستمی. ولی از جاهای دیگر این مجموعه مورد فراخوانی قرار می گیرند. پس این مفهوم که جنبه یک موجودیت مستقل و منفرد در برنامه است در این روش پشتیبانی می گردد.اکنون در نظر داریم این روش تشخیص جنبه را بر روی زبان جریان کاریBPEL ، اعمال و جنبه ها را از متن آن استخراج کنیم.

## ۲- برنامه نویسی جنبه گرا

روش های ساخت یافته وشی گرا روش های متداول جداسازی دغدغه های موجود درنرم افزار وبسته بندی آنها درقالب موجودیت های جداگانه و مستقل می باشند. عیب موجود در این روش ها، این است که آنها تنها خواسته های عملکردی[1] سیستم را درنظر می گیرند وتنها آنها را درقالب پیمانه هایی، بسته بندی می کنند ولی به خواسته های غیرعملکردی اهمیت چندانی نمی دهند.[۱] این مساله باعث می شود که مواردی که به بعد غیرعملکردی ارتباط دارند، درقالب واحدهای مستقلی بسته بندی نشود ودرعرض سایر مفاهیم، پراکنده گردد. درصورتی که نرم افزار با روش شی گرا به قطعات کوچک ترتقسیم شود، مشکلاتی مانند هرج و مرج در کد برنامه و همچنین پراکندگی مفاهیم در کد برنامه به وجود می آید که باعث پیچیدگی کلاس ها و همچنین کاهش قابلیت استفاده مجدد کلاس ها می گردد و همچنین

اثرات مختلفی بر روی طراحی وساخت برنامه می گذارند که عبارتند از :

- بهره وری پایین : پیاده سازی چندین دغدغه به طور همزمان ، باعث می شود تا برنامه نویس نتواند به درستی بر روی دغدغه های اصلی برنامه متمرکز شود و حواسش متوجه دغدغه های جانبی دیگر می گردد. بدین ترتیب بهره وری پایین می آید.
- کاهش میزان استفاده مجدد از کد :در سیستم های دیگر که همین عملیات تکراری را انجام می دهند، بدلیل وجود چندین دغدغه در هر پیمانه ، نمی توان از این پیمانه ها استفاده مجددکرد.
- کیفیت پایین کد : در هم پیچیدگی و پراکندگی کد باعث می شود که مشکلات و عیب ها در کدها پنهان شوند. همچنین وقتی برنامه نویس در یک زمان به پیاده سازی چندین دغدغه بپردازد، ممکن است به برخی از آنها توجه لازم را نکند.

برای اینکه قابلیت استفاده مجدد دریک نرم افزار بالا باشد، باید بتوان آن را تغییرداد ودرنتیجه اجزای مختلف نرم افزار می توانند درمحیط های مختلف، پیکره بندی شوند وباید بتوان، به خواسته های غیرعملکردی نیزقابلیت استفاده مجدد را اعمال کرد. وباید توجه داشته باشیم که اشکالات بیان شده تنها درکد برنامه رخ نمی دهد، بلکه ازمرحله تحلیل تا مرحله پیاده سازی وجود دارند ودرتمامی این مراحل با مشکل پراکندگی برخی دغدغه ها درعرض سایر دغدغه ها مواجه خواهیم بود. برای رفع مشکلات ذکرشده، kiczales درسال ۱۹۹۷، تکنیک جدیدی را تحت عنوان برنامه نویسی جنبه گرا ارائه داد. با استفاده از این رویکرد نرم افزارهایی ایجاد خواهند شد که قابلیت استفاده مجدد آنها ومیزان تحمل پذیری آنها نسبت به تغیرات افزایش خواهند یافت. این رویکرد درکنار رویکرد شی گرا مورد استفاده قرارمی گیرد ودرآن مفهوم جدیدی به نام جنبه تعریف می شود. پس جنبه یک واحد بسته بندی جدید است که مفاهیمی را که درعرض سایر مفاهیم پراکنده است را بسته بندی می کند. [۲,۱]جنبه ها، به عنوان یک واحد کپسوله سازی جدید معرفی می شوند که مفاهیمی که درعرض سایرمفاهیم پراکنده هستند را بسته بندی می کند یک جنبه دارای قسمت های زیراست :

نقطه پیوند[۲]: نقطه ای درقسمت اجرایی برنامه است. مانند فراخوانی یک متد.
نقطه برش[۳] : مکانیزمی برای ارجاع به تعدادی نقطه پیوند.
توصیه[۴]: بیانگر کدی است که شامل خواسته های غیر عملکردی سیستم است که ازکلاسها جدا شده ودرون جنبه قرارمی گیرد و درنقطه پیوند اجرا می شود.
بافتن[۵] : باید دریک مرحله ، مجدداً جنبه ها درون کلاسها، اعمال شوند به این عمل ادغام مجدد جنبه درنقاط پیوند مشخص شده برنامه، بافتن گفته می شود برای این کار ابزاری به نام بافنده وجود دارد که وظیفه اش، یافتن نقاط پیوند درون برنامه اصلی واعمال توصیه به آن قسمت ها است.

## ۳- وب سرویس

همانطور که می دانیم وب سرویس ها، به عنوان یک تکنولوژی جهانی برای ترکیب برنامه های توزیع شده بر روی اینترنت ایجاد شده اند. برای ساده کردن پردازش های تجاری، برنامه های غیر متمرکز باید با یکدیگر ارتباط داشته باشند و از داده های اشتراکی یکدیگر استفاده کنند. وب سرویس ها، نرم افزارهایی هستند که از XML برای انتقال اطلاعات استفاده می کنند. جداسازی دغدغه ها، یک هدف اساسی و مهم در معماری وب سرویس است. متاسفانه این اصل، اغلب در پیاده سازی وب سرویس ها گم می شود. برخی از دغدغه های موجود در وب سرویس ها که باید جداسازی شوند عبارتند از امنیت، مدیریت تراکنش [۶]، ثبت اطلاعات [۷]، کارایی[۸] و.... این خواسته ها در WSDL تعریف می شود. بنابر این کاربران می توانند ویژگی هایی را که به آنها نیاز دارند بر مبنای این تعریف ها انتخاب کنند. با جداسازی جنبه ها نگهداری سیستم راحت تر شده و ساختار برنامه می توانند به خوبی از هم جدا شوند. پس این جنبه ها می توانند قابلیت استفاده مجدد داشته باشند و در ساخت سرویس های دیگر مورد استفاده قرار گیرند. بنابراین هزینه و تلاش کاهش می یابد. ضمنا قابلیت انعطاف پذیری سرویس ها افزایش می یابد. وقتی سرویس گیرنده، خواسته ای را انتخاب می کند یک سری کد تولید می شود که این کدها در بعضی مواقع شامل جنبه ها می شوند که در هنگام کامپایل این اطلاعات و کدها به فایل WSDL اضافه می شوند تا در صورت نیاز مجدد به این کد ها با رجوع به سند WSDL آن جنبه ها را دوباره مورد استفاده قرار دهیم. بنابر این طراحان، دیگر به طراحی مراحل تولید جنبه ها کاری ندارند و فقط بر روی خواسته های عملیاتی تمرکز می کنند و در نتیجه کیفیت کارشان بالاتر می رود. [۸,۷]

### ۳-۱ ترکیب وب سرویس ها

عبارت ترکیب سرویس های وب ، به منظور ترکیب وب سرویس های موجود براساس الگوهایی برای حل یک مسئله پیچیده و یا دستیابی به یک هدف کاری و یا ایجاد یک سرویس جدید می باشد. مهمترین مسائل در رابطه با ترکیب سرویس های وب، استفاده مجدد از آنها و کاهش پیچیدگی آنها می باشد. با افزایش تعداد وب سرویس های قابل استفاده، استفاده مجدد از وب سرویس های موجود جالب تر از نوشتن مجدد آنهاست.در بعضی موارد، نمی توان از وب سرویسی استفاده مجدد کرد. وقتی که برای اولین بار وب بوجود آمد ، تنها یک محیطی برای انتشار داده ها بود ولی اکنون وب بصورت محیطی سرویس گرا برای تولید و دستیابی نه تنها صفحات ایستا ، بلکه برای تولید سرویس های توزیع شده ، توسعه پیدا کرده است . جهت ایجاد و توسعه برنامه های کاربردی بزرگتر نیاز به ترکیب وب سرویس ها می باشد و بررسی چگونگی ترکیب آنها از اهمیت ویژه ای برخوردار است

### BPEL ۳-۲

تا کنون چندین زبان جریان کاری[۹] برای ترکیب وب سرویس ها طراحی شده است. که یکی از این زبان ها که به عنوان استاندارد زبان جریان کاری در ترکیب سرویس های وب مطرح شد، BPEL نام دارد. ترکیب در BPEL بازگشتی[۱۰] است. یعنی نتیجه ترکیب، نیز یک وب سرویس است. BPEL 1.0 در جولای سال 2002 توسط microsoft , BEA , IBM به عنوان نتیجه الحاق دو زبان ترکیب مطرح شد. زبان جریان وب سرویس مربوط به IBM به نام WSFL و زبان XLANG مربوط به Microsoft در آوریل سال 2003 ، BPEL 1.1 برای استانداردسازی در OASIS ثبت شد. و سپس BPEL به WS-BPEL2.0 تغییر نام یافت.

مشکل اصلی زبان های جریان کاری کنونی مانند BPEL که در ترکیب سرویس های وب به کار می رود عدم جداسازی ویژگی های مربوط به عمل ترکیب است . یعنی خواسته های غیر عملکردی که ما به آن جنبه می گوییم در وب سرویس های ترکیب شده قابل جداسازی نیستند . بدون پشتیبانی از جداسازی دغدغه های تداخلی ، تعاریف آنها در پردازش ها ، بصورت پراکنده و پیچیده در آمده و این نگهداری و توسعه ترکیب وب سرویس ها را مشکل تر می کند . مثلا هنگامی که یک تغییر در سطح ترکیب نیاز باشد ، چندین مکان تحت تاثیر قرار می گیرد که این یک فرایند پر هزینه و حاوی خطا می باشد . که اعمال روش جنبه گرا ، باعث می شود که کدهای مربوط به دغدغه های تداخلی موجود که همان جنبه ها[۱۱] هستند در واحدهای جداگانه بسته بندی شوند و تنها یکبار طراحی شوند و چندین بار در نقاط اتصال مختلف بکار می روند که در هنگام بروز تغییر در عمل

ترکیب ، فقط همان تکه کد مربوط به آن جنبه تغییر مـی یابـد و ایـن باعث کاهش هزینه و زمان و جلوگیری از بروز خطا می شود .

## ۴- طرح پیشنهادی

در طرح پیشنهادی ما که استخراج جنبه ها از متن BPEL است، بعد از اضافه نمودن شرکت کننده هـایBPEL یعنـی وب سـرویس هـا و فرایندها ،بعد از اینکه هدف مشخص شد، متـدهای وب سـرویس هـای شریک مورد نظردر BPEL مورد بررسـی قـرار گرفتـه وگـراف جریـان فراخوانی استخراج می گردد. اگر متـدی از یـک شـریک توسـط چنـد فرایند، فراخوانی شود، در کد فرایند ،فعالیت مربوط به این فراخوانی را پیدا کرده و به عنوان کاندیدای جنبه ذخیره می نماید و سپس بررسی می شود که یک موجودیت مـستقل و منفـرد در برنامـه اسـت. سـپس جنبه با این ویژگی ها استخراج شده و به فرم AO4BPEL تبدیل می گردد.

مثلا اگر سیستم پردازش کارت اعتباری در بانک را در نظـر بگیـریم که از تکنولوژی مدیریت جریان کاری برای انجام برخی از پردازش های کاری خود استفاده می کند، و کاربران مختلف از سـرویس هـای ایـن سیستم استفاده کنند، با اعمال روش مطرح شده می توان عمل ثبت را که یک جنبه محسوب می شود را از داخـل فراینـدهای جریـان کـاری استخراج نمود. زیرا وب سرویس ثبت زیرگرافی اسـت در داخـل گـراف جریان فراخوانی که متـدی از آن توسـط چنـد فراینـد مختلـف ماننـد فرایند مجوز دهی و برداشت پول از حساب ،فراخوانی شده است و کـد مربوطه در هر دو فرایند تکرار شده است.

پس می توان گفت کـه روش پیـشنهادی مـا دارای دقـت و سـرعت بیشتری می باشد زیرا برخلاف روش های دیگر استخراج جنبه که بعد از استخراج کدهای تکراری باید طی مرحله دیگر مفاهیم معتبـر جنبـه شناسایی گردد و در برخی موارد به کدنویسی برنامـه نویـسان بـستگی دارد، روش ما مستقل از زبان و بر پایه مفهـوم مـی باشـد کـه ویژگـی ارزنده ای محسوب می شود. مابا استفاده از روش پیشنهادی خـود نـرم افزاری را پیاده سازی نمـوده ایـم کـه جنبـه هـای موجـود را از مـتن BPEL استخراج کرده و به فرم جنبه بـه زبـان AO4BPEL تبـدیل می نماید.

برای تبدیل جنبه شناسایی شده به فرم زبان AO4BPEL باید نقطه ای که فعالیت جنبه از آن مکان ها صدا زده می شود، مشخص گردد و در داخل المان pointcut نظیم شود . سپس فعالیت های مربوط به فعالیت جنبه مورد نظر که با تعامل با یکدیگر باعث به وجود آمدن آن سرویس شده اند، را به عنوان توصیه در تگ Advice ، قرار می دهیم. در مثال مطرح شده در مورد پردازش کارت اعتباری ، همانطور که گفتیم فعالیت ثبت به دلیل اینکه از مکان های مختلف فراخوانی شده و دارای ویژگی جنبه است، به عنوان جنبه مشخص می شود. برای تبدیل این جنبه به فرم AO4BPEL ابتدا باید نقطه برش این جنبه که همان جایی است که این جنبه از فرایندهای مختلف از آن فعالیت ها فراخوانی شده ، در المان pointcut در تگ مربوطه قرار گیرد.برای مثال داریم:

```
<pointcut>
//invoke[@operation ="credit"]
//invoke[@operation ="debit"]
</pointcut>
```

سپس باید عملیات مربوط به فعالیت ثبت که خود شامل فعالیت های دیگر است به صورت توصیه در قالب تگ Advice قرار گیردو همینطور لینک شرکا و متغیر هایی که در تمامی فرایند های BPEL در ایجاد این جنبه آمده اند به المان Aspect اضافه شود.بدین ترتیب جنبه موجود در متن زبان جریان کاریBPEL استخراج شده و به فرم زبان AO4BPELتبدیل می گردد.

## ۵- نتیجه

در این مقاله روشهای مختلف برای یافتن جنبه ها در داخل کد برنامـه ها و نقاط قوت وضعف آنها مطرح شد. بیـشتر تکنیـک هـا، حـداقل تـا حدی، مکمل بنظر می‌رسند، دراین مقاله همچنین زبان جریـان کـاری BPEL و محدودیت های موجود در این زبان را بررسی نمودیم . سپس با اعمال روش پیشنهادی برروی BPEL مشاهده کردیم که جنبـه هـا بهتر و با دقت بـالاتری تـشخیص داده مـی شـوند. زیـرا در ایـن روش فعالیت مشترک و تکراری کشف می شود کـه دارای ویژگـی منفـرد و مستقل است و جایی را به غیر از مجموعه خود فراخوانی نمی کند بلکه فرایندهای دیگر به آن وابسته اند و آن را فراخوانی می نمایند.

## مراجع


[1] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin .Aspect-Oriented rogramming. European Conference on Object-Oriented Programming (ECOOP),Finland. Springer-Verlag LNCS 1241. June 1997

[2] Aspect-Oriented software developement. http://www.aosd.net

[3] J. Hannemann and G. Kiczales, "*Overcoming the prevalent decomposition in legacy code*", Workshop on Advanced Separation of Concerns, International Conference on Software Engineering (ICSE), 2001.

[4] P. Tonella and M. Ceccato, "*Aspect mining through the formal concept analysis of execution traces*", Working Conference on Reverse Engineering (WCRE), 2004.

[5] T. Tourw´e and K. Mens, "*Mining aspectual views using formal concept analysis*", Source Code Analysis and Manipulation Workshop (SCAM), 2004.

[6] D. Shepherd, T. Tourw´e and L. Pollock, "*Using language clues to discover crosscutting concerns*", Workshop on the Modeling and Analysis of Concerns, 2005.

[7] Guadalupe Ortiz, Juan Hernández , Pedro J. Clemente , Pablo A. Amaya. How to Model Aspect-
Oriented Web Services . *Quercus Software Engineering Group University of Extremadura Computer Science Department*

[8] Mbuamuh Unice Gwe , " ASPECT ORIENTED PROGRAMMING FOR WEB SERVICES" , Fachgebiet Software Technik Fachbereich Informatik Technische Universitaet Darmstadt, 2004.

[1] functional
[2] Join point
[3] Pointcut
[4] Advice
[5] Weaving
[6] Transaction management
[7] Logging
[8] Performance
[9] workflow language
[10] recursive
[11] Aspect