
بسم الله الرحمن الرحيم

دانشگاه آزاد اسلامی

واحد علوم و تحقیقات

پایان نامه کارشناسی ارشد رشته مهندسی کامپیوتر- نرم‌افزار(M.Sc)

موضوع

یک روش برای مهندسی نیازمندیهای جنبه‌گرا

An Approach for Aspect-Oriented Requirements Engineering

استاد راهنما:

دکتر فریدون شمس

استاد مشاور:

دکتر میرعلی سیدی

نگارنده :

وحدت عبدل‌زاد

سال تحصیلی ۱۳۸۹-۱۳۸۸

سپاسگذاری

برای نگارش این پایان‌نامه از حضور استاد ارجمند جناب آقای دکتر فریدون شمس بهره بردم که این یادآوری، نمایانگر سپاس بـــی‌پایان من نسبت به کمک‌های بی‌دریغ ایشان است. زیرا بدون وجود ایشان نه تنها این تحقیق به نتیجه نمی‌رسید بلکه مطمئنا تحقیق در این حوزه مهندسی کامپیوتر دیگر برای من آنقدر مهم و ارزشمند نبود.

تقدیم به

پدر و مادر مهربانم که همواره در طول تحصیل از حمایت بی‌دریغ آنها بهره برده‌ام زیرا که بدون وجود آنها هیچ وقت اینجانب نمی‌توانستم این تحقیق را به پایان برسانم.

فهرست مطالب

فهرست جداول و لیست‌ها

فهرست شکل‌ها

چکیده

پیدایش برنامه‌نویسی جنبه‌گرا سبب بهبود پیمانه‌بندی دغدغه‌ها بخصوص دغدغه‌های مداخله‌ای در سطح پیاده‌سازی شد اما پیمانه‌بندی دغدغه‌های مداخله‌ای در این سطح از توسعه سیستم به تنهائی نمی‌تواند کارائی مهمی به ارمغان بیاورد. زیرا ضرورت دیدگاه سیستمی ایجاب می‌کند که این نوع فعالیت (پیمانه‌بندی) در مراحل اولیه توسعه سیستم نرم‌افزاری انجام شود. این مرحله از توسعه سیستم که به مهندسی نیازمندیهای جنبه‌گرا مشهور است مهمترین هدفش جداسازی دغدغه‌های مداخله‌ای به بهترین شکل ممکن و شناسائی جنبه‌ها بر اساس مستند نیازمندیها است. بنابراین باید مفهومی (ساختاری) که برنامه‌نویسی جنبه‌گرا درسطح پیاده‌سازی استفاده می‌کند و تحت عنوان جنبه از آن یاد می‌شود از همان مراحل اولیه توسعه شناسائی شود. در صورت عدم شناسائی جنبه‌ها در مراحل اولیه توسعه (فاز مهندسی نیازمندیها)، فقط در سطح کد می‌توان از مفهوم جنبه‌گرائی استفاده کرد.

تا به امروز روشهای مختلف و غیررسمی برای شناسائی دغدغه‌ها در مهندسی نیازمندیهای جنبه‌گرا ارائه شده‌اند. هر کدام از این روشها ساختارها، شیوه‌ها و ابزارهای خاص خود را برای پشتیبانی از شناسائی جنبه‌ها ارائه کرده‌اند و توانسته‌اند علاوه بر حل مشکل شناسائی جنبه‌ها به اهدافی چون ترکیب جنبه‌ها و مدیریت تداخل میان جنبه‌ها و مشخص کردن نقاط مصالحه[1] برای معماری جنبه‌گرا دست پیدا کنند. در زمینه رسمی‌سازی مفاهیم جنبه‌گرا، شیوه‌هایی در جهت بکارگیری مفهوم جنبه در شبکه پتری به عنوان یک زبان رسمی دنبال شده است که در آن فقط از مفهوم جنبه در قالب شبکه پتری (شبکه جنبه) برای کارهای امنیتی بهره‌برداری شده ولی هیچ روش رسمی برای شناسائی دغدغه‌ها در آنها ارائه نشده است.

در این تحقیق با هدف دستیابی به رسمی‌سازی، یک روش مبتنی بر شبکه پتری برای شناسائی جنبه‌ها ارائه شده است. روش پیشنهادی، از دو تعریفی مبتنی بر شبکه پتری که شبکه نیازمندی و شبکه دغدغه نامیده می‌شوند شروع شده و نهایتا با استفاده از اجرای مدل نهائی موجب شناسائی دغدغه‌ها می‌گردد. این روش پیشنهادی، اگرچه فقط شناسائی دغدغه‌ها را در بر می‌گیرد ولی تعاریف و ساختارهای آن می‌تواند از خصوصیات دیگر نیز حمایت کند که این نیازمند تحقیقات بیشتر پیرامون این حوزه است.

کلمات کلیدی

شناسائی جنبه‌ها، مهندسی نیازمندیهای جنبه‌گرا، شبکه پتری، شبکه دغدغه، شبکه نیازمندی، دغدغه‌های مداخله‌ای، برنامه‌نویسی جنبه‌گرا، جداسازی دغدغه‌ها

[1] Trade-off

فصل اول

معرفی

در این فصل ابتدا یک مقدمه پیرامون موضوع این تحقیق ارائه می‌شود سپس بخش تعریف مسأله را خواهیم داشت که در آن بیان می‌شود چه عاملی حاکی از ضرورت این کار تحقیقاتی است یا به بیان بهتر، در این تحقیق چه کاری باید صورت بپذیرد. با مشخص شدن صورت مسأله، سابقه کار تحقیقاتی پیرامون موضوع تحقیق ذکر می‌شود و عنوان می‌گردد که چه کارهایی از زمان پیدایش مفهوم مهندسی نیازمندیهای جنبه‌گرا صورت پذیرفته است. در بخش نتایج تحقیقاتی مورد انتظار، آنچه که امید می‌رود بعد از این تحقیق به آنها دست پیدا شود ذکر شده و نهایتا نیز ساختار این مستند تحقیقاتی (پایان‌نامه) ذکر می‌شود.

۱-۱- مقدمه

از سال ۱۹٦۸ (کنفرانس NATO) که توسعه سیستمهای نرم‌افزار به عنوان یک مشکل مهندسی مطرح شد کارهای فراوانی در جهت حل مشکلات پیش روی آن صورت پذیرفته و هم اکنون نیز در حال انجام است. این فعالیت‌ها بیشتر در جهت جداسازی بهتر مسأله (سیستم نرم‌افزاری) و پیمانه‌بندی بخشهای مختلف آن درون ماژول منسجم و مستقل هستند به طوری که این ماژولها کمتر هم پوشانی را با یکدیگر داشته باشند. این فعالیت‌های منسجم و طولانی مدت منجر به پیدایش شیوه‌ای برای توسعه سیستمهای نرم‌افزار شد که به آن شیءگرائی گفته می‌شود. این شیوه با در نظر گرفتن بخشهای مختلف یک سیستم نرم‌افزاری در قالب یک شیء توانست به اهداف مربوطه که همان جداسازی بهتر سیستم‌های نرم‌افزاری است دست پیدا کند. در این روش، دغدغه‌های[۱] سیستم (نیازمندیها، خواسته‌ها) در قالب شیء تعریف می‌شوند و از طریق واسطهای خوش تعریف با دیگر اشیاء ارتباط برقرار می‌کنند و نهایتا مجموعه‌ای از اشیاء که به صورت ساخت‌یافته در کنار هم قرار گرفته‌اند سیستم نهایی را پیاده‌سازی می‌کنند.

با به کارگیری توسعه نرم‌افزاری شیءگرا در کارهای تحقیقاتی و کاربردی مشخص شد که این شیوه در پیمانه‌بندی بعضی از دغدغه‌ها ناتوان است. این دغدغه‌ها، دغدغه‌های مداخله‌ای[۲] نامیده می‌شوند. دغدغه‌های مداخله‌ای دغدغه‌هایی هستند که نمی‌توانند در داخل یک ماژول (شیء) قرار بگیرند بنابراین مشکلاتی را برای طراحی، مدلسازی، پیاده‌سازی و نگهداری سیستم فراهم می‌کنند (مشکل درهم تنیدگی[۳] و پراکندگی[٤]). با گسترش تحقیقات در جهت حل این مشکل در شیءگرائی و همچنین تکامل سایر شیوه‌های مناسب برای توسعه سیستمهای نرم‌افزاری، در سال ۱۹۹۷ زبان برنامه‌نویسی تحت عنوان برنامه‌نویسی جنبه‌گرا معرفی شد. زبان برنامه‌نویسی جنبه‌گرا که نتیجه کار تحقیقاتی آقای کیک‌زالز[۵] و دوستانش در موسسه تحقیقاتی PARC[٦] بود بود مفهومی به نام جنبه[۷] را برای پیمانه‌بندی دغدغه‌های مداخله‌ای معرفی کرد. این مفهوم می‌توانست دغدغه‌های مداخله‌ای را در سطح پیاده‌سازی به بهترین شکل پیمانه‌بندی کند (از منظر پیاده‌سازی، جنبه ساختار برنامه‌نویسی است که دغدغه‌های مداخله‌ای را پیمانه‌بندی کرده و از بروز مشکلات درهم تنیدگی و پراکندگی کد جلوگیری می‌کند).

---

[1] Concerns
[2] Crosscutting concern
[3] Tangling
[4] Scattering
[5] Kiczales
[6] Palo Alto Research Center
[7] Aspect

با پیدایش زبان برنامه‌نویسی جنبه‌گرا و بکارگیری آن در پروژه‌های مختلف مشخص شد که استفاده از زبان برنامه‌نویسی جنبه‌گرا درست است که مشکل درهم تنیدگی و پراکندگی را حل می‌کند ولی به تنهایی کاربرد چندان مفیدی ندارد زیرا توسعه‌دهنده (یا برنامه‌نویس)، که سیستم را مبتنی بر مستندات تحلیل و طراحی شیءگرا برنامه‌نویسی می‌کند نیاز دارد که دوباره این مستندات را مورد تحلیل قرار دهد و از میان آنها جنبه‌ها را شناسائی کرده و سپس آن را پیاده‌سازی کند. این عمل شناسائی، یک کار وقت‌گیر، دشوار و خسته‌کننده است که بر دوش برنامه‌نویس قرار می‌گرفت. به عبارت دیگر در تولید سیستم تحلیل، طراحی و معماری به صورت شیءگرا انجام می‌شد و فقط پیاده‌سازی آن جنبه‌گرا بود. همچنین دغدغه‌های مداخله‌ای علاوه بر مشکل درهم تنیدگی و پیچیدگی که در سطح کد ایجاد می‌کردند، مشکلات مشابهی نیز برای دیگر فازها از جمله تحلیل، طراحی و مدلسازی سیستم فراهم می‌کردند. بنابراین، مفهوم جنبه‌گرائی در تمام فازهای توسعه سیستم نرم‌افزاری رخنه کرد که بتواند مشکلات مربوطه را حل کند.

ورود مفهوم جنبه‌گرائی در کلیه فازهای توسعه سیستم نرم‌افزاری منجر به پدیدار شدن توسعه نرم‌افزاری جنبه‌گرا شد. این فازها در توسعه نرم‌افزاری جنبه‌گرا با نام‌های مهندسی نیازمندیهای جنبه‌گرا، مدلسازی جنبه‌گرا، معماری جنبه‌گرا، پیاده‌سازی جنبه‌گرا، آزمایش و نگهداری جنبه‌گرا مورد ارجاع قرار می‌گیرند که هر کدام از این فازها علاوه بر اهداف قبلی خود اهداف دیگری را نیز در بر می‌گیرند.

۱-۲- تعریف مسأله

با توجه به اینکه پیدایش جنبه‌گرائی در فاز پیاده‌سازی است نقطه شروع کار توسعه نرم‌افزاری جنبه‌گرا نیز این فاز می‌باشد ولی با گذشت زمان این نتیجه حاصل شد که نیاز است جنبه‌هایی که پیاده‌سازی می‌شوند از فازهای قبل پیاده‌سازی شناسائی شوند (نقطه شروع از فاز پیاده‌سازی به فاز نیازمندیها انتقال پیدا کند). در واقع به جای اینکه پیاده‌ساز (برنامه‌نویس) در این فاز شروع به استخراج جنبه‌ها کند این جنبه‌ها از فازهای قبل، مخصوصا فاز اولیه (مهندسی نیازمندیها) استخراج شوند و رفته رفته کاملتر شده و نهایتا با زبان برنامه‌نویسی جنبه‌گرا پیاده‌سازی شوند.

جنبه در فاز نیازمندیها دغدغه‌ای است که فرآورده‌های نیازمندیهای دیگر را میان‌بر[1] می‌کند. شناسائی و مدیریت جنبه‌های اولیه[2] به بهبود پیمانه‌بندی در سطح نیازمندیها کمک می‌کند و باعث شناسائی زود هنگام تداخل‌های ما بین دغدغه‌ها می‌شود. شناخت جنبه‌های نیازمندی به معمار کمک می‌کند که یک سیستم خوب طراحی کند. به عبارت دیگر شناسائی و مدیریت جنبه‌های اولیه باعث [۲]:

[1] Crosscut
[2] Early aspects

- افزایش سازگاری نیازمندیها با طراحی معماری و مخصوصا با پیاده‌سازی می‌شود.
- فراهم شدن یک دلیل[1] و قابلیت ردیابی برای جنبه‌ها در سرتاسر فعالیت‌های چرخه حیات می‌شود.
- کمک کردن به تضمین اینکه دغدغه‌های مداخله‌ای موجود و مشهود در فضای مسأله یا در فضای راه حل به عنوان یک جنبه در فاز پیاده‌سازی در نظر گرفته شده‌اند.

با توجه به مزایای مطرح شده برای شناسائی جنبه‌ها در مراحل اولیه توسعه سیستم نرم‌افزاری، فاز مهندسی نیازمندیهای جنبه‌گرا از حساسیت بیشتری برخودار می‌شود و نیاز به روشی برای دستیابی به اهداف آن است. این اهداف عبارتند از [۳] :

۱. فراهم کردن پشتیبانی بهبود یافته برای جداسازی خصوصیات وظیفه‌مندی و غیروظیفه‌مندی مداخله‌ای در روند مهندسی نیازمندیها و لذا ارائه قابلیت بهتر برای شناسائی و مدیریت تداخل‌های ناشی از نمایش‌های درهم‌تنیده.
۲. مشخص کردن تاثیر و نگاشت جنبه‌های سطح نیازمندیها بر روی فرآورده‌های گامهای بعدی توسعه و لذا بنا نهادن مصالحه‌های حیاتی قبل از حاصل شدن معماری

تا به امروز کارهایی نیز در این زمینه صورت پذیرفته است ولی آنچه که حائز اهمیت است و به عنوان صورت مسأله برای این تحقیق مطرح می‌شود فراهم کردن روشی رسمی برای شناسائی جنبه‌ها است به طوری که این روش رسمی قابلیت کاربردی نیز داشته باشد. یعنی روش با مفاهیم سطح پائین ریاضی مطرح نشود که کاربرد آن در عمل بسیار سخت باشد. به بیان ساده‌تر، هدف فراهم کردن روشی به منظور شناسائی جنبه‌ها مبتنی بر ابزار یا مفاهیم کاربردی که قابلیت تبدیل شدن به مفاهیم پایه ریاضی را داشته باشد (این مفهوم در این تحقیق شبکه پتری در نظر گرفته شده است).

۱-۳- سابقه کار تحقیقاتی

بعد از معرفی شدن مهندسی نیازمندیهای جنبه‌گرا در سال ۱۹۹۹ در مقاله آقای گراندی[2] [۴]، کارهای زیادی در این حوزه انجام شده است که نقطه شروع آنها مقاله‌ای بود که در کنفرانس مهندسی نیازمندیهای سال ۲۰۰۲ ارائه شد [۳]. این مقاله یک مدل عمومی برای مهندسی نیازمندیهای جنبه‌گرا پیشنهاد داد و باعث پدیدار شدن یک دید اولیه برای مهندسی نیازمندیهای جنبه‌گرا شد. با مشخص شدن اهداف و مزایای مهندسی نیازمندیهای جنبه‌گرا کارهای مختلفی در این زمینه صورت پذیرفت که مهمترین آنها به شرح زیر هستند:

[1] Rationale
[2] Grundy

۱. روش cosmos[1]، که در آن یک طرح مدلسازی چند منظوره برای دغدغه‌ها ارائه شده است [۵]. در این روش یک تقسیم‌بندی از انواع دغدغه‌ها صورت پذیرفته است که جنبه‌ها در یکی از این دسته‌بندی‌ها قرار داده شده‌اند.
۲. روش ARCaDe[2]، که در آن نویسندگان، مدل مطرح شده در [۳] را کاملتر کرده و روشهایی برای مشخص کردن شیوه ترکیب و حل تداخل‌های ما بین جنبه‌ها پیشنهاد داده‌اند. همچنین همراه این روش ابزاری نیز بدین منظور طراحی و پیاده‌سازی شده است.
۳. روش Theme/Doc، که در آن از Theme و تحلیل لغوی برای مشخص کردن جنبه‌ها استفاده شده است [۶]. این روش پیشنهادی علاوه بر ابزار، دارای نمادها و مفاهیمی است که به شناسائی جنبه‌های پنهان در سیستم کمک می‌کند. روش Theme همچنین شیوه‌ای برای تحلیل و طراحی ارائه می‌کند.
٤. روش جداسازی چند بعدی[3] [۷]، که در آن روش چند بعدی برای مشخص کردن دغدغه‌ها و جنبه‌ها پیشنهاد داده شده است. این روش علاوه بر پوشش اهداف مهندسی نیازمندیهای جنبه‌گرا، مکانیزمی برای مشخص کردن نقاط مصالحه برای فاز بعدی (معماری) ارائه کرده است.
۵. روش توسعه نرم‌افزاری جنبه‌گرا با موارد کاربری [۸] که در آن از قابلیت‌های موارد کاربری برای مشخص کردن جنبه‌ها استفاده شده است. در این روش، دو مفهوم تحت عنوان برش موارد کاربری[4] و ماژول موارد کاربری[5] معرفی شده‌اند که کلیه فرآیندهای مهندسی نیازمندیها و معماری جنبه‌گرا پیرامون این دو مفهوم حرکت می‌کنند.

علاوه بر روشهای ذکر شده که از اهمیت بیشتری برخوردار هستند روشهای دیگری در [۹، ۲، ۱۰، ۱۱، ۱۲، ۱۳، ۱۴، ۱۵، ۱۶، ۱۷، ۱۸] برای مهندسی نیازمندیهای جنبه‌گرا ارائه شده‌اند.

نوآوری این کار تحقیقاتی نسبت به کارهای ارائه شده در زمینه مهندسی نیازمندیهای جنبه‌گرا این است که کلیه شیوه‌های ارائه شده، یک روش غیررسمی برای شناسائی جنبه‌ها پیشنهاد داده‌اند که به نوعی الهام گرفته شده از روشهای موجود برای مهندسی نیازمندیهای سنتی است. همچنین در برخی از این روشها جنبه‌ها فقط به عنوان نیازهای غیروظیفه‌مندی در نظر گرفته شده‌اند. ولی کار تحقیقاتی حاضر یک روش رسمی مبتنی بر شبکه پتری برای شناسائی جنبه‌ها در فاز مهندسی نیازمندیهای جنبه‌گرا پیشنهاد می‌دهد. این روش پیشنهادی دغدغه‌ها

[1] Concern-space modeling schema
[2] Aspectual Requirements Composition and Decision
[3] Multi-dimensional
[4] Use-case slice
[5] Use-case module

را به عنوان موجودیت‌های رده اول[1] در ساختار خود در نظر می‌گیرد و آنها را محدود به نیازهای وظیفه‌مندی و غیروظیفه‌مندی نمی‌کند. همچنین با ارائه شیوه خاص، فعالیت‌های لازم در زمینه شناسائی جنبه‌ها را کاهش می‌دهد. این روش در ماهیت خود سه تعریف پایه نیز برای دغدغه، نیازمندی و ترتیب تحقق نیازمندیها ارائه می‌کند که می‌تواند کمک زیادی برای پشتیبانی از دیگر فعالیت‌های فاز مهندسی نیازمندیهای جنبه‌گرا فراهم نماید.

۱-۴- نتایج تحقیقاتی مورد انتظار

از آنجائی که این تحقیق به نوعی یک کار جدید در زمینه شناسائی جنبه‌ها با استفاده از یک روش رسمی است می‌تواند دست‌یافته‌ها و نتایج تحقیقاتی متعددی را به همراه بیاورد. با این حال نتیجه اصلی این تحقیق فراهم کردن یک روش رسمی مبتنی بر شبکه پتری برای شناسائی جنبه‌ها است.

در راستای این تحقیق یک تعریف جامع از مفهوم دغدغه ارائه شده است که به نوعی تعاریف پیشین را کاملتر می‌کند. همچنین سه تعریف برای دغدغه، نیازمندی و ترتیب تحقق نیازمندیها مبتنی بر شبکه پتری، تحت عنوان شبکه دغدغه، شبکه نیازمندی و ترتیب اجرا ارائه شده است که علاوه بر کمک کردن به شناسائی جنبه‌ها زیربنائی برای دیگر فعالیت‌های مهندسی نیازمندیهای جنبه‌گرا فراهم می‌کند (مثلا از بین بردن تداخها). اما آنچه که علاوه بر شناسائی جنبه‌ها حائز اهمیت است رسمی‌سازی اولین گامها برای مهندسی نیازمندیهای جنبه‌گرا است. با توجه به مطالب ذکر شده، یافته‌های اصلی این تحقیق را می‌توان در موارد زیر خلاصه کرد :

- ارائه کردن روشی برای شناسائی جنبه ها آن هم به صورت رسمی با استفاده از شبکه‌های پتری
- دست یابی به یک تعریف جامع از دغدغه
- دست یابی به یک تعریف رسمی مبتنی بر شبکه پتری برای دغدغه و نیازمندی
- افزایش میزان اطمینان به جنبه‌های شناسائی شده

با وجود یافته‌های مطرح شده، در این تحقیق به موضوعاتی چون نحوه ترکیب جنبه‌های شناسائی شده در سیستم نهائی، حل تداخل‌های حاصل از ترکیب جنبه‌ها و سایر اهداف جزئی که اخیرا (در سال ۲۰۰۹) مطرح شده‌اند پرداخته نشده است. زیرا هدف اصلی این تحقیق شناسائی دغدغه‌ها است ولی با توجه به قابلیت شبکه‌های پتری و همچنین نحوه چیدمان عناصر پایه‌ای برای روش پیشنهادی می‌توان به راحتی موضوعاتی چون ترکیب جنبه‌ها را نیز در روش پیشنهادی اعمال کرد. ولی با کلیه توضیحات ذکر شده راجع به روش

[1] First-class

پیشنهادی، تلاشهای زیادی در این حوزه تحقیقاتی مورد نیاز است تا بتوان کلیه فرآیند توسعه یک سیستم نرم‌افزاری جنبه‌گرا را مبتنی بر یک روش رسمی مانند شبکه‌های پتری به پیش برد.

۱-۵- ساختار پایان نامه

این تحقیق شامل هفت فصل است که ترتیب فصول و موضوعات مطرح شده به صورت کلی در زیر بیان شده است:

- **فصل اول: معرفی (همین فصل).**

در این فصل یک مقدمه از حوزه این تحقیق ارائه شده است و در آن مشخص شده که هدف از این تحقیق چیست و خواهان پاسخگوئی به چه سئوالی هستیم. در ادامه همین فصل یک توضیح مختصر از تحقیقات انجام شده در مورد موضوع این تحقیق ارائه شده و نهایتا نیز یک دید کلی از فصل‌های مختلف این تحقیق ذکر شده است.

- **فصل دوم: ادبیات تحقیق.**

در این فصل یک اطلاعات کلی و مختصر از مفاهیم و ادبیات مطرح در زمینه مهندسی نیازمندیهای جنبه‌گرا و توسعه نرم‌افزاری جنبه‌گرا ارائه می‌شود.

- **فصل سوم: جایگاه مهندسی نیازمندیها.**

در این فصل به بررسی بخشهای مختلف مهندسی نیازمندیها پرداخته شده که در برگیرنده ماهیت بخشها و تکنیک‌های مورد استفاده در آنها است.

- **فصل چهارم: بررسی روشهای مهندسی نیازمندیهای جنبه‌گرا.**

در این فصل به بررسی روشهای موجود در زمینه مهندسی نیازمندیهای جنبه‌گرا پرداخته شده و نهایتا یک مقایسه اجمالی بر روی این روشها ارائه می‌شود.

- **فصل پنجم: روش پیشنهادی برای شناسائی جنبه‌ها.**

در این فصل روش پیشنهادی برای شناسائی جنبه‌ها به صورت کامل و جامع تشریح می‌شود.

- **فصل ششم: مطالعه موردی.**

در این فصل یک مطالعه موردی به نام سیستم مدیریت هتل با استفاده از روش پیشنهادی مورد بررسی قرار داده می‌شود تا بتوان نتیجه روش پیشنهادی را روی یک مطالعه موردی مشاهده کرد.

- **فصل هفتم: نتیجه‌گیری.**

در این فصل ابتدا یک خلاصه‌ای از کل فصل‌های این تحقیق ارائه می‌شود. سپس محدودیت‌های روش پیشنهادی و فرصت‌های تحقیقاتی آتی که ناشی از این کار تحقیقاتی است بیان می‌شود.

فصل دوم

ادبیات تحقیق

در این فصل هدف معرفی ادبیات تحقیق است. ادبیات تحقیق که همان مفاهیم کلیدی هستند نقش بسیار مهمی در درک و فهم یک حوزه تحقیق دارند. بدین منظور ابتدا مفاهیم پایه از جمله نیازمندی، دغدغه، دغدغه مداخله‌ای و چند مفهوم دیگر تعریف می‌شوند. سپس موضوعات منسجم‌تر و مهمتری مثل جداسازی دغدغه‌ها، خاصیت پیمانه‌ای، قابلیت نگهداری، قابلیت ردیابی و مفاهیم مرتبط مورد بررسی قرار می‌گیرند. علاوه بر مفاهیم اصلی، در این فصل حوزه‌هایی که این تحقیق به نوعی با آنها درگیر است مانند برنامه‌نویسی جنبه‌گرا تشریح خواهند شد و نهایتا شبکه‌های پتری به عنوان ابزاری اصلی این تحقیق تشریح می‌شود.

۲-۱- نیازمندیها

در مهندسی، نیازمندیها یک مستند منحصر به فرد از موارد لازم است که یک محصول یا سرویس باید دارای آن بوده یا آنها را باید انجام دهد. در واقع آن مستند ویژگیها، قابلیت‌ها، خصوصیات و کیفیت یک سیستم را در جهت اینکه سیستم مورد نظر ارزش و کاربرد لازم برای کاربر را دارد، مشخص می‌کند [۱۹]. در مهندسی نرم‌افزار نیازمندی عبارت است از توضیحاتی درباره مواردی (شروط یا قابلیت‌ها) که یک سیستم باید دارای آن باشد (آن را انجام دهد). در زمینه مهندسی نرم‌افزار، نیازمندیها به دو دسته نیازهای وظیفه‌مندی[۱] و نیازهای غیروظیفه‌مندی[۲] تقسیم می‌شوند. این نوع تقسیم‌بندی در مهندسی نرم‌افزار بسیار مفید است زیرا فقط نیازهای وظیفه‌مندی می‌توانند مستقیما در نرم‌افزار پیاده‌سازی شوند و نیازهای غیروظیفه‌مندی بوسیله دیگر جنبه‌های سیستم مورد کنترل قرار می‌گیرند.

۲-۱-۱- نیازهای وظیفه‌مندی

این نوع نیازمندیها موارد و موضوعاتی که یک سیستم نهایی باید قادر به انجام آن باشد را مشخص می‌کند. برای نمونه: قالب بندی یک متن در یک برنامه ویراستار یا ثبت نام در یک سیستم آموزش.

۲-۱-۲- نیازهای غیروظیفه‌مندی

این نوع نیازمندیها اغلب با نام‌های "نیازمندیهای کارائی[۳]"، "کیفیت سرویس نیازمندیها[٤]"، "محدودیت‌ها[٥]" نیز مورد ارجاع قرار می‌گیرند و موضوعاتی در ارتباط با خود سیستم و چگونگی خوب انجام دادن وظایف توسط سیستم را مشخص می‌کنند. نمونه‌هایی از نیازهای غیروظیفه‌مندی عبارت است از : قابلیت دسترسی[٦]، قابلیت آزمون‌پذیری[۷]، قابلیت نگهداری[۸] و استفاده آسان[۹].

نیازمندی غیروظیفه‌مندی می‌تواند با توجه به نوع نیازمندی از قبیل کارائی[۱۰]، قابلیت نگهداری، ایمنی[۱۱]، قابلیت اطمینان[۱۲] و یا دیگر نوع نیازمندیها در دسته‌های مختلف نیازمندی طبقه‌بندی شود و در بعضی از موارد

[1] Functional requirements (FR)
[2] Non-functional requirements (NFR)
[3] Performance requirements
[4] Quality of service requirements
[5] Constraints
[6] Availability
[7] Testability
[8] Maintainability
[9] Ease-of-use
[10] Performance
[11] Safety
[12] Reliability

نیازهای غیروظیفه‌مندی به داخل نیازهای وظیفه‌مندی پراکنده می‌شوند به عنوان مثال یک نیازمندی امنیتی می‌تواند به چندین نیازمندی در داخل نیازمندی وظیفه‌مندی تجزیه شود.

۲-۱-۳- خصوصیت یک فهرست نیازمندی خوب

مشخص کردن فهرست نیازمندیهای یک سیستم از اهمیت بسیار بالائی برخوردار است زیر اگر این فهرست نتواند نیازهای حوزه مسأله یا ذینفعان را به خوبی بیان کند سیستم نهایی فراهم شده ارزش و کاربرد لازم از دید کاربر خود را نخواهد داشت. لذا باید در بیان نیازمندیها دقت زیادی صورت پذیرد. ولی آنچه که در بیان این فهرست باید از آن اجتناب شود بیان این موضوع است که چگونه سیستم باید این نیازمندیها را پیاده‌سازی کند. حال در زیر تعدادی از خصوصیات یک فهرست نیازمندیهای خوب را که به صورت عمومی مورد قبول است ارائه می‌شود [۲۰، ۲۱، ۲۲، ۲۳].

جدول ۲-۱: خصوصیات یک فهرست نیازمندی مناسب

خصوصیت	توضیح
انسجام[1]	نیازمندی فقط و فقط یک چیز (موضوع) را مورد خطاب قرار دهد(آدرس‌دهی کند).
کامل بودن	نیازمندی یک بار به صورت منسجم بدون هیچ اطلاعات نادیده گرفته شده، بیان شود.
سازگاری	نیازمندیی با نیازمندی دیگر تناقض نداشته باشد و کاملا با مستندات خارجی معتبر، سازگار باشد.
صحت	نیازمندی تمام یا بخشی از حرفه مورد نیاز را که بوسیله ذینفعان بیان شده و معتبر است پوشش دهد.
قابل مشاهده خارجی	نیازمندی یک سری خصوصیاتی از محصول را مشخص می‌کند که به صورت خارجی توسط کاربر قابل مشاهده و آزمایش است و همچنین نیازمندیهایی که محدودیت‌ها را مشخص می‌کنند باید در بخش مربوطه به طور کامل مستند گردند.
امکان‌پذیری	نیازمندی بتواند در داخل محدودیت‌های پروژه پیاده‌سازی شود.
غیر مبهم	نیازمندی اجمالا بدون ارجاع به گفتارهای تکنیکی، اختصارات و دیگر گفتارهای مبهم بیان می‌شود.
الزام آور	نیازمندی خصوصیات تعریف شده توسط ذینفعان را ارائه می‌کنند که عدم وجود یکی از آنها موجب ناکارائی گشته و نمی‌تواند اصلاح گردد.
قابل بازبینی	پیاده‌سازی نیازمندی می‌تواند از طریق یکی از چهار روش نظارت، تحلیل، نمایش تجربی[2] و آزمایش تعیین گردد.

۲-۲- دغدغه[3]

بسیاری از توسعه‌دهندگان درک خیلی خوبی از مفهوم دغدغه دارند و هر یک تعریف خاصی را برای آن ارائه می‌دهند. اما نزدیک شدن به یک تعریف خوب برای دغدغه کار بسیار مشکلی است. بنابراین، برای رسیدن به یک تعریف خوب و همسو با تحقیق، ما ابتدا به بررسی این تعاریف پرداخته و بعد تعریفی که در این تحقیق محور کار ما خواهد بود را بیان خواهیم کرد.

[1] Cohesive
[2] Demonstration
[3] Concern

تعریف اول: در علوم کامپیوتر دغدغه یک مجموعه خاص از رفتارهای مورد نیاز برای یک برنامه کامپیوتری پنداشته شده است [۲٤].

تعریف دوم: دغدغه هر چیز مهم و ارزشمند برای ذینفعان است. ذینفعان می‌توانند کاربر، مدیر پروژه، توسعه‌گر و بانی[1] پروژه در نظر گرفته شوند. این دغدغه‌ها باید در فرآیند تولید نرم‌افزار مشخص، طراحی، پیاده‌سازی و آزمایش شوند [۸].

تعریف سوم: دغدغه یک نیازمندی و شاخص ضروری در یک سیستم است که با استفاده از یک ساختار کد پیاده‌سازی می‌شود [۲۵]. این تعریف این اجازه را می‌دهد که یک دغدغه محدود به سیستم‌های شی‌گراء نباشد و یک سیستم ساخته‌یافته نیز بتواند چنین مفهومی داشته باشد.

تعریف چهارم: دغدغه یک مستند بر روی (سرتاسر) واحدهای نرم‌افزار است [۲٦]. این تعریف باعث می‌شود که تعریف محدود به کد نباشد و به شکلی کل چرخه حیات را پوشش دهد.

تعریف پنجم: یک تعریف لغوی از دغدغه عبارت است از یک موضوع که باید مورد بررسی قرار گیرد [۲۷].

تعریف ششم: یک تعریف استاندارد IEEE، دغدغه را به صورت نکات مورد علاقه که وابسته به توسعه سیستم است و همچنین عملیات یا هر جنبه‌ای که برای ذینفعان مهم و حیاتی است، تعریف می‌کند [۲۸].

مطابق با تعاریف بالا می‌توان دریافت که دغدغه‌ها اساسا مفاهیمی هستند که نمی‌توانند در یک فرآورده گنجانده شوند. اگرچه فرآورده‌ها می‌توانند دغدغه‌ها را بیان کنند یا یک دغدغه در نظر گرفته شوند. همچنین دغدغه‌ها نیازمندیها هم نیستند. هر چند که نیازمندیها دغدغه‌ها را نمایش می‌دهند و در فرآیند توسعه نرم‌افزار نیز در بسیاری از نقاط، نوعی دغدغه محسوب می‌شوند. همچنین یک "مدل دغدغه[2]" در حالت عمومی یک "مدل حوزه[3]" نیست اگر چه یک مدل حوزه می‌تواند یک دغدغه را در خود به کار ببرد [۵]. پس مشاهده می‌شود که دغدغه‌ها از زوایای مختلف قابل بررسی و تعریف هستند ولی ما در ادامه تعریفی را ارائه خواهیم کرد که بیان‌گر، تقریبا همه ابعاد قابل تأمل در مورد دغدغه و تعاریف ذکر شده در بالا است.

۲-۲-۱- تعریف دغدغه

به یک یا چند نیازمندی وابسته به ذینفعان و توسعه یک سیستم که قابلیت پیاده‌سازی با یک ساختار داده را داشته باشد دغدغه گفته می‌شود.

[1] Sponsor
[2] Concern model
[3] Domain model

در این تعریف بیان عبارت "یک یا چند" بیانگر این موضوع است که شاید یک نیازمندی تشکیل‌دهنده یک دغدغه باشد یا اینکه چندین نیازمندی، دغدغه‌ای را تعریف کنند. کلمه "نیازمندی" نشانگر رفتارهای مورد انتظار در یک سیستم یا برنامه است که منجر به دغدغه می‌شود (طبق تعریفی که از نیازمندیها داریم). ذکر کلمه "ذینفعان" حاکی از این است که این نیازمندیها فقط شامل نیازمندیهای مربوط به خود سیستم نیست و در برگیرنده نیازمندیهای ذینفعان نیز است. استفاده از کلمه "توسعه" نیز نشانگر این است که این تعریف محدود به یک بخش از فرآیند تولید نیست (منجمله کد) و استفاده از عبارت "قابلیت پیاده‌سازی با یک ساختار داده" وسعت بکارگیری این مفهوم را در بیشتر سیستمهای توسعه از جمله شی‌گراء، ساخت‌یافته و کلا هر سیستم توسعه‌ای که نهایتا با ساختارهای مربوط به پیاده‌سازی سروکار خواهد داشت افزایش می‌دهد.

۲-۲-۲- دغدغه هسته[1] و منطق حرفه[2]

در هر برنامه‌ای یک سری از دغدغه‌ها وجود دارند که جزء دغدغه‌های اصلی برنامه در نظر گرفته می‌شوند و برای عملکرد درست و صحیح سیستم لازم و ضروری هستند. هر یک از این دغدغه‌ها، "دغدغه هسته" نامیده می‌شود و مجموعه دغدغه‌های هسته، "منطق حرفه" را تشکیل می‌دهند در حالی که دیگر جنبه‌های برنامه (دغدغه‌ها) برای اجرای مناسب برنامه (کارائی بهتر) مورد نیاز بوده و جزئی از "منطق حرفه" واقعی برنامه نیستند. به عنوان مثال در نظر بگیرید اگر در حال نوشتن یک برنامه کاربردی برای مدیریت رکوردهای داروئی باشید سازماندهی و شاخص‌گذاری چنین رکوردهایی یک "دغدغه هسته" محسوب می‌شود در حالی که ثبت یک تاریخچه از تغییرات (واقعه‌نگاری[3]) در رکوردهای پایگاه‌داده، یا یک سیستم تصدیق، یک دغدغه عادی به شمار می‌رود که آن را دغدغه مداخله‌ای[4] می‌نامند (چون چندین بخش از سیستم را درگیر کار خود می‌کند). این نوع دغدغه‌ها در ادامه، در بخش سوم از همین فصل مورد بررسی قرار خواهند گرفت. نهایتا اینکه به عنوان یک مثال از دغدغه‌ها، یک دغدغه می‌تواند در یک حالت عمومی یک ارتباط پایگاه داده‌ای و در حالت خاص انجام یک عمل محاسبه در نظر گرفته شود که همه این نوع بیان‌ها (یعنی میزان دانه‌بندی در بیان دغدغه‌ها) به سطح گفتگوی بین توسعه‌دهندگان و برنامه‌ای که باید بر روی آن بررسی صورت پذیرد وابسته است.

به عنوان یک جمع‌بندی اینکه دغدغه‌ها نسبی و پویا هستند و به یک محصول نرم‌افزاری خاص یا یک واحد منفرد که در گذر زمان تغییر می‌کند، و به دید و هدف توسعه‌دهنده، کاربر یا دیگر ذینفعان که نرم‌افزار را مورد بررسی قرار می‌دهند، وابسته‌اند[۲۸، ۲۵، ۲۹، ۳۰].

[1] Core concern
[2] Business logic
[3] Logging
[4] Crosscutting concern

۲-۳- دغدغه مداخله‌ای

توسعه مبتنی بر مولفه شیوه‌ای است که به صورت گسترده برای ساخت سیستمهای پیچیده استفاده می‌شود. در واقع نیازمندیها به مولفه‌هایی از نوع کلاس، شیء و سرویس تخصیص داده می‌شوند ولی نیازمندیهایی وجود دارد که نمی‌توانند در یک مولفه متمرکز شوند و شاید بر مولفه‌های زیادی تاثیر بگذارد به این نوع نیازمندیهای میان‌بر[1] کننده مولفه‌ها، دغدغه‌های مداخله‌ای گفته می‌شود. در علوم کامپیوتر به دغدغه‌های مداخله‌ای جنبه‌هایی از برنامه گفته می‌شود که دغدغه‌های دیگر برنامه را تحت تاثیر خود قرار می‌دهند [۳۱]. این دغدغه‌ها اغلب نمی‌توانند به وضوح از دیگر بخشهای سیستم هم در طراحی و هم در پیاده‌سازی مجزا شوند و منجر به مشکل پراکندگی و درهم‌تنیدگی می‌شوند (این دو مفهوم در قسمت‌های بعدی شرح داده خواهند شد).

به عنوان مثالی از دغدغه‌های مداخله‌ای می‌توان به واقعه‌نگاری، امنیت ،اشکال‌زدایی و بازبینی‌کارائی[2] اشاره کرد. همچنین از بعد دیگر دغدغه‌های مداخله‌ای یک نوع افزونگی در سیستم ایجاد می‌کنند زیرا وقتی نتوان یک دغدغه را در یک مولفه متمرکز و پیاده‌سازی کرد در این صورت ناگزیریم که آن را در چندین مولفه بگنجانیم. در واقع قطعه کدهائی در هر یک از این مولفه وارد می‌شود تا اینکه بتوان دغدغه مربوطه را تحقق بخشید و هم اینکه با استفاده از این قطعه کدها بتوان ارتباط بین این مولفه‌ها را در جهت تحقق دغدغه مربوطه برقرار کرد. در واقع به عنوان یک نتیجه‌گیری از این تعاریف، برای دغدغه‌های مداخله‌ای می‌توان دو ویژگی اصلی ذکر کرد:

۱. عدم تجزیه واضح از دیگر بخش‌ها هم در طراحی و هم در پیاده‌سازی
۲. قرار گرفتن کد پیاده‌سازی در میان چندین مولفه

۲-۳-۱- تقسیم بندی دغدغه‌های مداخله‌ای

دغدغه‌های مداخله‌ای به دو نوع تقسیم می‌شوند و هر یک در برگیرنده دغدغه‌های خاصی از نرم‌افزار هستند که در زیر این دو نوع معرفی می‌شوند.

نوع اول: شامل دغدغه‌هایی هستند که نیازهای غیروظیفه‌مندی را برآورده می‌کنند و اصطلاحا به آنها "دغدغه پایه[3]" گفته می‌شود مانند امنیت، واقعه‌نگاری.

نوع دوم: شامل دغدغه‌هایی هستند که با نیازهای وظیفه‌مندی سروکار دارند مانند کلیه موارد کاربری. پس در واقع مواردکاربری نیز دغدغه‌های مداخله‌ای هستند اما در این نوع، دو حالت خاص از دغدغه‌های مداخله‌ای

[1] Cut across
[2] Performance monitoring
[3] Infrastructure concern

وجود دارند که اهمیت بیشتری دارند این دغدغه‌ها عبارتند از: "دغدغه همتا[1]" و "دغدغه گسترش[2]" که در زیر تعریف هر یک ارائه شده است.

دغدغه‌های همتا دغدغه‌هایی هستند که کاملا مجزا از یکدیگر هستند و هیچ دغدغه‌ای برتری نسبت به دیگر دغدغه‌ها ندارد اما هنگامی که به پیاده‌سازی این همتاها پرداخته می‌شود روی هم افتادگی‌هایی[3] میان این دغدغه‌ها پدیدار می‌شود. "دغدغه گسترش" دغدغه‌ای هستند که به عنوان یک دغدغه جدید بر روی دغدغه‌های اصلی تعریف می‌شود و سرویس‌ها و خصوصیت‌های جدید را ارائه می‌دهد.

۲-۴- جداسازی دغدغه‌ها[4]

"جداسازی دغدغه‌ها" یک اصل مهم طراحی در بسیاری از حوزه‌ها از قبیل برنامه‌ریزی شهری، معماری و طراحی اطلاعات و مهندسی است و هدف از آن این است که سیستمها طوری طراحی شوند که توابع موجود بتوانند به صورت مستقل از توابع دیگر بهینه‌سازی شوند و دلیل آن این است که خرابی یک تابع سبب خرابی تابع دیگر نشود و به طور کلی طراحی، فهم و مدیریت سیستمهای مستقل پیچیده را آسانتر کند. در علوم کامپیوتر "جداسازی دغدغه‌ها" یعنی فرآیند شکستن یک برنامه کامپیوتری به چندین ویژگی مجزا که این ویژگیها کمترین روی هم افتادگی را داشته باشند، در نظر گرفته شده است.

عبارت "جداسازی دغدغه‌ها" برای اولین بار در مقاله آقای دایجسترا در سال ۱۹۷٤ معرفی شد [۳۲] و هدف ایشان از بیان این عبارت تشریح فلسفه وجودی پیمانه‌بندی بود که به توسعه‌دهنده اجازه می‌دهد تا پیچیدگی سیستمی که باید طراحی شود کاهش دهد. ۱۵ سال بعد این عبارت به یک ایده قابل قبول تبدیل شد و در سال ۱۹۸۹ آقای کریس رید در کتابی تحت عنوان " عناصر برنامه‌نویسی تابعی" [۳۳] جداسازی دغدغه‌ها را شرح داد (برای علاقه‌مندان در این زمینه خواندن متن مقاله آقای دایجسترا توصیه می‌شود).

در پیشرفت رو به جلو در زمینه "جداسازی دغدغه‌ها"، جداسازی در روشهای سنتی از طریق پیمانه‌بندی و محصورسازی با کمک پنهان‌سازی اطلاعات فراهم می‌شد و امروزه نیز کلیه پارادایم‌های برنامه‌نویسی به توسعه‌دهندگان در فرآیند بهبود "جداسازی دغدغه‌ها" کمک می‌کنند. برای مثال زبانهای برنامه‌نویسی شیءگرا مثل C# می‌تواند دغدغه را جدا کرده و آنها را درون اشیاء قرار دهد و یک الگوی طراحی مثل MVC می‌تواند محتوایات را از آن چیزی که باید نمایش داده شود و همچنین مدل پردازش داده جدا کند. همین طور طراحی سرویس‌گرا می‌تواند دغدغه را به سرویس‌ها تبدیل کند، زبانهای برنامه‌نویسی مثل C و پاسکال دغدغه‌ها را از

[1] Peers concern
[2] Extension concern
[3] Overlaps
[4] Separation of concerns

هم جدا کرده و آنها را درون رویه‌ها قرار می‌دهند و زبانهای برنامه‌نویسی جنبه‌گرا دغدغه‌ها را با اشیاء و جنبه‌ها جداسازی می‌کنند.

۲-۵- خاصیت پیمانه‌ای[1]

مفهوم خاصیت پیمانه‌ای در نرم‌افزار کامپیوتر برای چند دهه است که مورد حمایت قرار گرفته و به معنی این است که نرم‌افزار به مولفه‌های مجزایی با نام‌های متفاوت تقسیم می‌شود که اغلب پیمانه‌ها نامیده می‌شوند و با یکدیگر مجتمع می‌شوند تا نیازهای نرم‌افزار را برآورده نمایند [۳۴]. در اصل، خاصیت پیمانه‌ای یک صفت واحد از نرم‌افزار است که به برنامه امکان می‌دهد تا از نظر مفهومی قابل مدیریت باشد. از دید آقای بوچ، "خاصیت پیمانه‌ای" خصوصیتی از یک سیستم است که به مجموعه‌ای از پیمانه‌های منسجم با اتصال سست تجزیه شده است (پیمانه‌ها تجریدها را در داخل واحدها مجزا از هم بسته‌بندی می‌کنند) و پیمانه‌بندی نیز یعنی شکستن یک برنامه به پیمانه‌هایی که این پیمانه‌ها بتوانند به صورت جدا از هم کامپایل شده و نیز اتصالات لازم با دیگر پیمانه‌ها را داشته باشند [۳۵]. همچنین پیمانه‌ها جداسازی دغدغه‌ها را نشان می‌دهند و بوسیله اعمال محدودیت‌های منطقی بین مولفه‌ها باعث بهبود قابلیت نگهداری می‌شوند.

پیمانه‌ها از طریق واسط‌هایشان در داخل برنامه مورد استفاده قرار می‌گیرند و یک واسط پیمانه عناصری را که آن پیمانه فراهم می‌کند یا مورد نیاز برای پیمانه دیگر است ارائه می‌کند و این عناصر که در واسط‌ها تعریف شده‌اند قابل مشاهده توسط دیگر پیمانه‌ها هستند ولی پیاده‌سازی داخل پیمانه‌ها که نشانگر کد اجرائی مربوط به عناصر است از دید دیگر پیمانه‌ها مخفی است.

اما اگر نرم‌افزار به تعداد دفعات بیش از حدی تقسیم شود (تعداد پیمانه‌ها زیاد باشد) فعالیت لازم برای توسعه‌ی آن تا حد قابل صرف نظر کردن، کوچک می‌شود. متأسفانه در این میان نیروهای دیگری نیز وارد صحنه می‌شوند (یکپارچگی) و باعث می‌شوند این نتیجه‌گیری تا حد ناراحت کننده‌ای نادرست باشد. به همین دلیل باید از پیمانه‌بندی کمتر از حد و بیش از حد اجتناب کرد. اما اصولی که در پیمانه‌ای کردن سیستم نرم‌افزاری باید به آن توجه شود این است که هر پیمانه بر حسب دغدغه‌ای که آن، پیاده‌سازی می‌کند منسجم باشد و واسط‌های بین این پیمانه‌ها نیز ساده در نظر گرفته شوند.

۲-۶- موارد کاربری

مدتها در فرآیندهای شیء گرا و غیر شیء گرا از سناریوها برای کمک به فهم نیازمندیهای سیستم استفاده می‌شد اما این سناریوها با اینکه همیشه تولید می‌شدند بندرت مستندسازی و نگهداری می‌شدند تا زمانیکه "ایوار

[1] Modularity

جکیبسون" وضعیت را با ارائه متدولوژی شیءگرا دگرگون نمود. او در متدولوژی خود نام این سناریوها را مورد کاربری گذاشت و آنرا محور اصلی کار خود قرار داد. ویژگی بارز این روش جمع‌آوری نیازمندیها از دیدگاه غیرتکنیکی است یعنی در این روش به سیستم از بیرون نگاه می‌شود و نیازمندیها از این دید مورد بررسی قرار می‌گیرد پس با این مقدمه می‌توانیم مورد کاربری را به شکل زیر تعریف کنیم: مورد کاربری دنباله‌ای از عملیات است که یک سیستم انجام می‌دهد تا یک نتیجه قابل مشاهده و ارزشمند برای فرد استفاده کننده از سیستم فراهم نماید.

موارد کاربری مانند هر شیوه دیگر برای خود مدلسازی دارند که به آن مدلسازی موارد کاربری گفته می‌شود و شامل ساختن دیدهای خارجی از سیستم به صورت مجموعه‌ای از عوامل و موارد کاربری مرتبط هستند. در این مدلسازی دو رابطه مهم وجود دارد که بسیار پرکاربرد هستند (روابط "توسعه به[1]" و "استفاده از[2]"). اتصال "توسعه به" یک مورد کاربری را به یک مورد کاربری مشابه اما اندکی محدودتر (همان رابطه وراثت) وصل می‌کند و اتصال "استفاده از" وقتی رفتار مشترک بین دو مورد کاربری وجود دارد برای جلوگیری از تکرار آن در مورد کاربری استفاده می‌شود و در اصل عمل فاکتور گیری را انجام می‌دهد. موارد کاربری به دلیل اینکه ابزاری سیستماتیک و در عین حال شهودی برای دریافت نیازهای وظیفه‌مندی تلقی می‌شوند و کل فرآیند توسعه را به جلو می‌راند می‌توانند ابزار بسیار خوبی برای توسعه نرم‌افزار جنبه‌گرا شوند.

۲-۷- دیدگاه‌ها[3]

ساختار یک مدل یا توصیف پیچیده، عاملهای[4] زیادی را درگیر خود می‌کند. این عاملها دورنماها[5] و دیدهای[6] مختلفی از فرآورده‌ها و یا سیستمی که آنها سعی دارند آن را توصیف یا مدل کنند، دارند. دورنماها و دیدها توضیحات جزئی (ناکامل) هستند که دلیل آن ناشی از مسئولیت‌ها یا نقشهای مختلفی است که به عاملها تخصیص داده شده‌اند. این مسئولیت‌ها و نقشها می‌توانند به صورت سازمان‌یافته تعریف شوند، از بعضی ساختارهای پایه‌ای فرآورده‌ها یا سیستم پیروی کنند و یا اینکه می‌توانند مدلسازی یا قابلیت‌های توصیفی مختلفی را بازتاب کنند.

به ترکیبی از عامل و دید که عامل آن را نگهداری می‌کند دیدگاه گفته می‌شود [۳۶]. کار بر روی دیدگاه‌ها عموما مربوط به مهندسی نیازمندیها است. اما محدود به آن نیست زیرا دیدگاه‌ها در سرتاسر فرآیند توسعه

[1] Extends
[2] Uses
[3] Viewpoints
[4] Agents
[5] Perspectives
[6] Views

سیستم به کار گرفته می‌شوند. برای مثال، روشهای طراحی مختلفی دیدگاه‌های ضمنی دارند بدین صورت که آنها پیشنهاد می‌کنند چندین مدل از سیستم مانند جریان-داده، مدل موجودیت-رابطه و غیره ایجاد شود. همچنین به عنوان یک روشی که مبتنی بر دیدگاه‌ها است می‌توان به PREView اشاره کرد که یکی از روشهای مهندسی نیازمندیها است [۳۷]. در این روش یک دیدگاه PREView، اطلاعات جزئی پیرامون سیستم را محصور کرده و نیازمندیها را بر حسب دیدگاه‌ها سازماندهی می‌کند.

۲-۸- قابلیت نگهداری[1]

درجه‌ای که یک سیستم می‌تواند به سادگی تغییر پیدا کند قابلیت نگهداری نامیده می‌شود (یا تلاش‌هایی برای اینکه سیستم بتواند تغییر پیدا کند) [۳۸]. قابلیت نگهداری به عنوان یکی از شش خصوصیت مدل کیفیت ISO/IEC 9126-1 در سال ۲۰۰۱ تعریف شده است. قابلیت نگهداری باید به عنوان بخشی از نیازمندی غیروظیفه‌مندی یک سیستم در نظر گرفته شود.

یک عبارت مترادف با قابلیت نگهداری، قابلیت وفق‌پذیری[2] است که به صورت توانائی یک سیستم نرم‌افزاری برای اجازه دادن به تغییرات، تعریف می‌شود. نمونه‌ای از تغییرات که یک سیستم می‌تواند داشته باشد عبارت است از [۳۹]:

- برطرف کردن خطاها
- تحقق بخشیدن به نیازمندیهای جدید
- سهل کردن نگهداری در آینده
- تلاشهای مرتبط با تغییرات محیط

۲-۹- قابلیت ردیابی[3]

توانائی برای تعیین آسان اینکه، چگونه یک تکه از یک فرآورده نرم‌افزاری (از قبیل نیازمندی، طراحی وکد) دیگر فرآورده‌ها را تحت تاثیر قرار می‌دهد را قابلیت ردیابی گویند [۱۱]. قابلیت ردیابی یک مفهوم مهم در توسعه نرم‌افزار است و این امکان را فراهم می‌کند که وقتی یک تغییر در نیازمندیها انجام شد و به دنبال آن بخشهایی از طراحی و کد تحت تاثیر آن تغییر قرار گرفتند، بتوان آن تاثیرات را مشخص کرد.

[1] Maintainability
[2] Adaptability
[3] Traceability

در اصل قابلیت ردیابی، عناصر موجود در مدل سطح بالا را به عناصر موجود در مدل سطح پائین مرتبط می‌کند [۸]. این ارتباط‌دهی کمک می‌کند که بتوان تشخیص داد که آیا کلیه نیازمندیها پیاده‌سازی شده‌اند و چه تاثیری، ناشی از تغییرات در نیازمندیها است. به طور کلی، تغییرات در عناصر سطح بالا را که منجر به تغییرات بر روی عناصر سطح پائین می‌شود می‌توان با این قابلیت مشخص کرد. همچنین اگر مدلهای سطح بالا و سطح پائین از اصول ساختاری مختلفی پیروی کنند در این صورت باید تلاش زیادی برای نگهداری قابلیت ردیابی بین عناصر انجام شود.

۲-۱۰- مشکل درهم تنیدگی[1]

اگر مولفه‌ای شامل پیاده‌سازی باشد که این پیاده‌سازی، پیاده‌سازی دغدغه‌های مختلفی از سیستم در حال توسعه را در بر گیرد در واقع یک "درهم پیچیدگی" اتفاق افتاده است [۸] و به این معنی است که توسعه‌گر یک مولفه، نیاز به این دارد که دغدغه‌های مختلف سیستم را درک کند تا بتواند آن مولفه را پیاده‌سازی کند و مولفه به جای برآورده کردن یک دغدغه منفرد، درگیر چندین دغدغه می‌شود و از قابل فهم بودن جلوگیری می‌کند. "درهم پیچیدگی" ناشی از ماهیت دغدغه‌های مداخله‌ای است. با توجه به اینکه این نوع دغدغه‌ها با چندین بخش از سیستم برای عملکرد خود درگیر هستند، بعضی از این دغدغه‌ها با بخش‌هایی از سیستم اشتراک پیدا می‌کنند که این بخش مشترک در پیاده‌سازی خود باید کلیه ساختارهایی را که مورد نیاز برای آن دغدغه‌ها است در نظر بگیرد. این عمل منجر به مشکل "درهم پیچیدگی" می‌شود. اما نکته‌ای که در این مورد وجود دارد این است که توسعه‌دهنده این عمل را به عنوان یک قابلیت استفاده مجدد در نظر نگیرد زیرا استفاده مجدد دلالت بر این دارد که کد و رفتارهای یکسان، در زمینه‌های مختلف قابل استفاده باشد. در حالی که این عمل بیانگر این موضوع نیست چون این مولفه خاص دارای کدی است که متعلق به چندین مولفه است و لزوما در زمینه دیگر یا سیستم دیگر به کار گرفته نشده است.

۲-۱۱- مشکل پراکندگی[2]

اگر کدهای پیاده‌سازی یک دغدغه در داخل چندین مولفه قرار گیرند به این مفهوم پراکندگی گویند [۸]. در این صورت اگر نیازمندیهای مربوط به یک دغدغه تغییر پیدا کند، یا اگر طراحی آن تغییر کند در این صورت چندین مولفه برای اعمال این تغییرات باید به روزرسانی شوند و این یک مشکل مهم در توسعه یک سیستم است. در واقع پیاده‌سازی آن پیمانه‌ای نیست و کاملا مشخص است که این مشکل ناشی از پیاده‌سازی یک

[1] Tangling
[2] Scattering

دغدغه مداخله‌ای است. چون کد آن باید به دلیل ماهیت این نوع دغدغه یعنی درگیری با کلاسهای مختلف سیستم، در میان مولفه‌های مختلف پخش شود.

۲-۱۲- برنامه‌نویسی جنبه‌گرا

برنامه‌نویسی جنبه‌گرا یک پارادایم برای جداسازی دغدغه‌ها و پیمانه‌بندی کردن دغدغه‌های مداخله‌ای در داخل موجودیت‌های نرم‌افزاری خوش تعریف به نام جنبه‌ها است [٤٠]. برنامه‌نویسی جنبه‌گرا به عنوان یک روش جدید جایگزینی برای برنامه‌نویسی شیءگرا نیست بلکه زبانهای برنامه‌نویسی شیءگرا را به وسیله پیمانه‌بندی دغدغه‌های مداخله‌ای کامل می‌کند تا برنامه‌هایی با ساختار خوب و واضح ایجاد شود (برنامه‌نویسی جنبه‌گرا در جهت تبدیل شدن به یک گسترش برای برنامه‌نویسی شیءگرا حرکت می‌کند). بنابراین در برنامه‌های کاربردی پیاده‌سازی شده با برنامه‌نویسی جنبه‌گرا، برنامه‌نویس هنوز هم با موضوعاتی همچون کلاس، عملیات و ویژگیهای مربوط به کلاس سروکار دارد. در اصل برنامه‌نویسی جنبه‌گرا با تعریف بعد جدید (دوم) از پیمانه‌بندی، مفهوم کلاسها را با استفاده از جنبه‌ها کامل می‌کند تا امکان پیاده‌سازی دغدغه‌هایی فراهم شود که قبلا از طریق کلاسها به صورت مناسب قابل پیمانه بندی نبودند.

برنامه‌نویسی جنبه‌گرا در نیمه سال ۱۹۹۰ در شرکت PARC[1] به عنوان پروژه‌ای تعریف شد که ریشه آن بر می‌گشت به کارهایی که در جهت پیمانه‌بندی کد و تسهیل استفاده مجدد و نگهداری صورت گرفته بود [٤٠]. در سال ۱۹۹۷ آقای Gregor Kiczals و اعضای گروه آن از شرکت PARC موفق شدند مفهوم جنبه را تعریف کنند و آن را در کنفرانس OOPSLA معرفی کردند [٤١]. حاصل تلاشهای این گروه علاوه بر معرفی برنامه‌نویسی جنبه‌گرا منجر به یک پیاده‌سازی از زبان، تحت عنوان AspectJ شد که هدف از آن ایجاد یک متدولوژی از برنامه‌نویسی جنبه‌گرا بود تا برای تعداد زیادی از توسعه‌دهندگان قابل دسترس باشد.

(برای داشتن یک دید اولیه از محیط‌های برنامه‌نویسی جنبه‌گرا و یک مطالعه موردی پیرامون زبان برنامه‌نویسی جنبه‌گرا به [۱] مراجعه شود)

۲-۱۲-۱- مفاهیم پایه و تعاریف

هر پارادایم جدید برنامه‌نویسی یک مجموعه از تعاریف و مفاهیم را معرفی می‌کند که برای درک، فهم و به کارگیری پارادایم لازم است. این مفاهیم برای روش برنامه‌نویسی رویه‌ای، پیمانه و رویه است و برای شیوه شیءگرا نیز دربرگیرنده محصورسازی، وراثت و چندریختی است. قابل مشاهده است که هر پارادایم جدید با تغییرات مهمی در مفاهیم و ساختار همراه است (از دیدگاه فنی کاملا مشخص است که تغییرات در ساختار

[1] Palo Alto Research Center

برنامه‌ها به صورت رادیکالی است). بنابراین چنین تغییراتی در روند توسعه نرم‌افزاری برای بدست آوردن پارادایم‌های جدید مورد نیاز است. برنامه‌نویسی جنبه‌گرا مشابه پارادایم‌های پیش از خود مفاهیم جدیدی را به وجود می‌آورد که یادگیری این مفاهیم جدید دربرگیرنده هزینه است اما با این وجود، بهبود در کیفیت برنامه‌های کاربردی و افزایش پیمانه‌بندی که ناشی از به کارگیری برنامه‌نویسی جنبه‌گرا است هزینه یادگیری این مفاهیم جدید را کاهش می‌دهد.

۲-۱۲-۱-۱- جنبه

جنبه یک واحد برنامه‌نویسی است که برای تسخیر[1] عملکردی که برنامه کاربردی را میان‌بر می‌کند طراحی شده است. در اصل واحدی است که از پراکندگی کد عملیات مداخله‌ای جلوگیری می‌کند و اغلب نیز به صورت یک ساختار مداخله‌ای تشریح می‌شود.

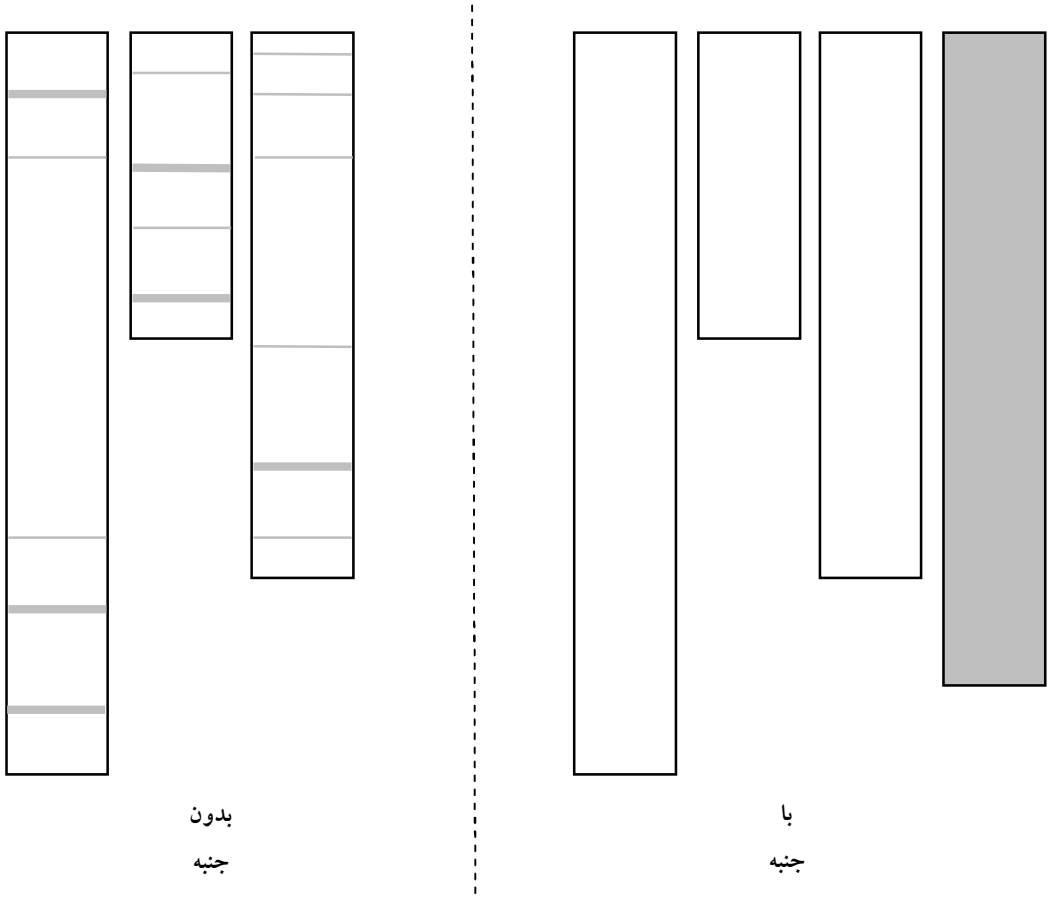
برنامه‌های کاربردی که با استفاده از زبان برنامه نویسی جنبه‌گرا ایجاد می‌شوند از کلاسها و جنبه‌ها تشکیل شده‌اند. وجود کلاس‌ها و جنبه‌ها در برنامه کاربردی نشان دهنده این است که پیمانه‌بندی یک برنامه از دو بعد صورت گرفته است: عملیاتهای پایه‌ای که به وسیله کلاسها پیاده‌سازی می‌شوند (این بعد می‌تواند بعد ساختاری نامیده شود) و عملیات مداخله‌ای که به وسیله جنبه‌ها پیاده‌سازی شده‌اند (این بعد می‌تواند بعد عملیاتی تلقی شود). بنابراین در یک برنامه کاربردی جنبه متفاوت از کلاس است زیرا جنبه یک عملکرد مداخله‌ای را پیاده‌سازی می‌کند. این موضوع در شکل ۲-۱ به تصویر کشیده شده است. در شکل ۲-۱ سمت چپ برنامه‌ای را نشان می‌دهد که از کلاسها تشکیل شده است و خطوط افقی نشان دهنده قسمتهایی از کد مربوط به هر کلاس است که در ارتباط با یک عملکرد مداخله‌ای (مانند واقعه‌نگاری) ایجاد شده است. این عملکرد، برنامه کاربردی را به دلیل اینکه کلیه کلاسهای را تحت تاثیر قرار داده است میان‌بر می‌کند. همچنین سمت راست شکل ۲-۱ همان برنامه کاربردی پیاده‌سازی شده با جنبه را نشان می‌دهد (مستطیل خاکستری) که در آن، کلیه کدهای مربوطه به عملکرد مداخله‌ای در داخل جنبه قرار گرفته شده است و کلاسها عاری از کدهای پراکنده هستند.

از نظر ساختاری یک جنبه متشکل از دو بخش کد توصیه[2] و محل برش[3] است. توصیه دربرگیرنده قطعه کدی است که باید اجرا شود درحالی که محل برش نقطه یا نقاط‌هایی را مشخص می‌کند که کد توصیه در آن باید به اجرا درآید. کاملا واضح است که کد توصیه یا کد جنبه به عملیاتی که می‌خواهد پیاده‌سازی کند وابسته است. کد جنبه می‌تواند عملیات دغدغه‌های وظیفه‌مندی و غیروظیفه‌مندی باشد ولی در بیشتر موارد، عملیاتهای

[1] Capture
[2] Advice code
[3] Pointcut

پیاده‌سازی شده غیروظیفه‌مندی هستند. همچنین دو نکته بسیار مهم در مورد جنبه وجود دارد که عبارت است از:

- یک جنبه به صورت مستقیم عملیات مداخله‌ای را پیاده‌سازی نمی‌کند بلکه از API اختصاصی برای عملکرد خود استفاده می‌کند.
- جنبه همانند شیء یک مفهوم انتزاعی است که در زبانهای مختلف می‌تواند به کار گرفته شود.



شکل ۲-۱: مفهوم جنبه از دید کلاسها

۲-۱۲-۱-۲- نقطه اتصال

در بخش قبل مشاهده شد که جنبه یک موجودیت نرم‌افزاری است که عملکرد مداخله‌ای را پیاده‌سازی می‌کند. این تعریف جنبه تکیه بر مفهومی به نام نقطه اتصال[1] دارد. نقطه اتصال، نقطه‌ای در روند اجرای برنامه (جریان کنترل) است که در آن محل، یک یا چندین جنبه می‌تواند اعمال شود بنابراین نقطه اتصال یک مفهوم کلی است. یک برنامه می‌تواند نقاط اتصال زیادی داشته باشد اما کلیه این نقاط توسط برنامه‌نویسی جنبه‌گرا قابل

[1] Jointpoint

استفاده نیستند. نقطه اتصالهای یک سیستم در انواع مختلف طبقه‌بندی می‌شوند و فقط یک زیر مجموعه‌ای از این انواع بوسیله زبان جنبه‌گرا پشتیبانی می‌شوند. بعلاوه مفهوم نقطه اتصال به شدت وابسته به یک اجرای خاص یا ویژه از برنامه است یعنی اینکه اجراهای مختلف از یک برنامه می‌تواند منجر به نقطه اتصالهای متفاوتی شود. با وجود اینکه تعریف جنبه در زمان اجرا اتفاق می‌افتد ولی این تعریف مبتنی بر ساختار برنامه (کلاسها، متدها، فیلدها و غیره) است. بنابراین متناسب با ساختار برنامه، نقطه اتصال می‌تواند در انواع مختلف وجود داشته باشد که در زیر این گونه‌ها که مستقل از نوع پیاده‌سازی هستند معرفی می‌شوند:

- **متدها:** با توجه به اینکه برنامه می‌تواند به صورت یک ترتیبی از فراخوانی و اجرای متدها در نظر گرفته شود بنابراین فراخوانی و اجرای متد دو نقطه اتصال هستند.
- **سازنده:** با توجه به اینکه سازنده‌ها مهمترین موجودیتهای استفاده شده برای ایجاد اشیاء در برنامه کاربردی هستند و همچنین با در نظر گرفتن سازنده‌ها به صورت یک متد، فراخوانی و اجرای یک سازنده، نوعی از نقطه اتصال است.
- **استثناءها:** یک استثناء وقتی اتفاق می‌افتد که یک موقعیت غیر عادی در زمان اجرا را گزارش دهد و دلیلش این است که در آن زمان یک رفتار خاصی را انجام دهد. این دو اتفاق یعنی رخ دادن استثناء و اجرای رفتار مورد انتظار، دو نقطه اصلی در اجرای یک برنامه کاربردی هستند و می‌توانند به عنوان نقطه اتصال در نظر گرفته شوند.
- **فیلدها:** با توجه به اینکه بسیاری از جنبه‌ها با داده‌ها سروکار دارند و فیلدها، عناصر اصلی در سطح کد برای پیاده‌سازی داده‌ها هستند عملیات خواندن و نوشتن بر روی فیلدها می‌توانند به عنوان نقطه اتصال در نظر گرفته شوند.

در میان این گروهها، متدها جزء بیشترین نقطه‌های مورد استفاده در زبان جنبه‌گرا هستند. نهایتا اینکه حتی برنامه‌های ساده نیز دارای نقطه اتصال هستند و تعداد آنها می‌تواند خیلی زیاد باشد پس این وظیفه توسعه‌دهنده جنبه است که بهترین نقطه اتصال برای پیاده‌سازی جنبه مورد نظر را انتخاب کند.

۲-۱۲-۱-۳ محل برش

نقطه اتصالهایی که در روند برنامه‌نویسی جنبه‌گرا تعریف می‌شوند برای تعیین اینکه کدام نقطه اتصال به کدام جنبه وابسته است کافی و مناسب نیستند. برای این امر موجودیتی مورد نیاز است که این نقاط اتصال را شرح دهد. این موجودیت در قالب محل برش تعریف می‌شود.

محل برش مجموعه‌ای از نقطه اتصالها است که یک جنبه در آن نقاط اعمال می‌شود (استفاده می‌شود). یک جنبه با استفاده از یک محل برش بیان می‌شود زیرا یک محل برش تعدادی از نقطه اتصالها را گروه‌بندی می‌کند که در کد منبع مختلف برای عملکرد جنبه مورد نیاز هستند. البته نقطه اتصالها به صورت تصادفی انتخاب نمی‌شوند بلکه یک سری اطلاعات در مورد اینکه در کدام بخش از برنامه قرار گرفته‌اند که یک جنبه خاص باید در آنجا اعمال شود به اشتراک می‌گذارند. در واقع محل برش مشخص می‌کند که جنبه کجاست (در کجا باید قرار گیرد) اما اینکه جنبه چیست؟ توسط کد توصیه مشخص می‌شود.

اما نکته مهم اینکه محل برش وابسته به برنامه کاربردی است و زمانی که لازم است یک جنبه در یک برنامه دیگر دوباره مورد استفاده قرار گیرد باید محل برش مورد نیاز برای جنبه، با محلهای جدید مطابقت داشته باشد. نکته دیگر اینکه مفهوم محل برش با مفهوم نقطه اتصال پیوند خورده است اما ماهیت این دو با هم متفاوت هستند نقطه اتصالها موجودیتهای خوش تعریف زمان اجرا هستند در حالی که محلهای برش نمی‌توانند به یک برش زمانی یا ساختار ویژه تعلق داشته باشند. در واقع محل برش تعریف شده است تا به عنوان یک عنصر ساختاری کد، در تعریف یک جنبه شرکت کند.

۲-۱۲-۱-۴- کد توصیه

با توجه به موضوعات مطرح شده کاملا مشخص است که جنبه متکی بر محل برش است و محل برش بیان می‌کند که جنبه در کجا باید اعمال شود. مفهوم کد توصیه نیز مشخص می‌کند که در داخل جنبه چه دستوراتی وجود دارد. در واقع کد توصیه تعریفی از رفتار یک جنبه است. کد توصیه همراه با محل برش عملکردهای مداخله‌ای را پیاده‌سازی می‌کنند و دستوراتی که در داخل کد توصیه تعریف شده‌اند در کلیه نقاط اتصال مشخص شده در محل برش اجرا می‌شوند. یک کد توصیه درست مانند یک متد دارای بدنه دستورات است ولی یک تفاوت اصلی بین متد و کد توصیه وجود دارد و آن عدم امکان فراخوانی مستقیم کد توصیه است. کد توصیه به جای فراخوانی مستقیم، به داخل نقطه اتصالهای مشخص شده در محل برش وارد می‌شود (با آنها بافته می‌شود).

کد توصیه‌ها نیز مانند محل برش‌ها دارای انواع متفاوتی هستند که سه نوع اصلی آن به شرح زیر هستند:

- قبل از[1] : کد توصیه قبل از نقطه اتصال اجرا می‌شود.
- بعد از[2] : کد توصیه بعد از نقطه اتصال اجرا می‌شود.
- پیرامون[3] : کد توصیه قبل و بعد نقطه اتصال اجرا می‌شود.

[1] Before
[2] After
[3] Around

۲-۱۲-۲- مزایای برنامه‌نویسی جنبه‌گرا

برنامه‌نویسی جنبه‌گرا با فراهم کردن راه‌حلهایی برای حل مشکلات دغدغه‌های مداخله‌ای، برنامه‌نویسی شیءگرا را کامل می‌کند. این زبان با تعریف یک لایه جدید بر روی مکانیزم تجزیه مبتنی بر داده روشهای برنامه‌نویسی شیءگرا و عملکردهای آنها را بهبود می‌دهد به طوری که لایه جدید، مرتبط با کارکردهای عملیاتی است که روشهای برنامه‌نویسی شیءگرا قادر به یکپارچه‌کردن (پیمانه‌بندی) مستقیم آنها نیست.

برنامه‌نویسی جنبه‌گرا با تعریف لایه جدید، مفاهیم نقطه اتصال، محل برش، کد توصیه و جنبه را تعریف می‌کند که امکان این را فراهم می‌کنند که بتوان دغدغه‌های مداخله‌ای را در داخل جنبه‌ها منسجم کرده و از پراکندگی کد مربوط به آنها جلوگیری کرد. از طرفی هم، با توجه به اینکه کلیه روشهای موجود در توسعه سیستمها به نوعی از تجزیه یک بعدی استفاده می‌کنند این امر باعث می‌شود که کنترل تغییرات، قابلیت استفاده مجدد مولفه‌ها و قابلیت ردیابی در سیستمها با مشکلاتی مواجه شود. اما با معرفی تجزیه چند بعدی این مشکلات تا حد زیادی بهبود یافتند [۲۶]. جداسازی چند بعدی دغدغه‌ها بیان می‌کند که در هر سطح یا فاز توسعه سیستم، باید فرآورده‌های آن را از چندین بعد مورد ارزیابی قرار داده و نهایتا نیز کلیه توصیفات و مشخصات هر فرآورده را در قالب یک واحد در نظر گرفت.

این نوع عملکرد باعث شود کلیه داده‌ها، متدها و کلاس‌های مربوط به یک دغدغه در داخل یک واحد قرار داده شوند. پس جداسازی چند بعدی این قابلیت را فراهم می‌کند تا بتوانیم سیستم را به بهترین شکل پیمانه‌بندی کنیم. اما این روش نیازمند یک مکانیزم تجزیه و ترکیب است تا بتواند این عمل جداسازی و ترکیب را انجام دهد. برنامه‌نویسی جنبه‌ها با فراهم کردن مکانیزم تجزیه و ترکیب این امکان را فراهم می‌آورد که از جداسازی چند بعدی که در سطح مدلسازی یا معماری قابل بکارگیری است در سطح کد نیز استفاده شود. با پشتیبانی از جداسازی چند بعدی توسط برنامه‌نویسی جنبه‌گرا تا حدی زیادی تاثیر موجی تغییرات در سیستم کاهش پیدا کرده و مولفه‌هایی حاصل خواهد شد که قابلیت استفاده مجدد و توسعه موازی بالائی دارند.

۲-۱۲-۳- معایب برنامه‌نویسی جنبه‌گرا

برنامه‌نویسی جنبه‌گرا مانند هر تکنولوژی دیگر دارای معایب است. یکی از مهمترین ایرادات مربوطه که ناشی از تفکر بعضی از اشخاص است قدرت و توانائی زبان برنامه‌نویسی جنبه‌گرا در ایجاد کدهای غیرقابل اشکال‌زدائی است. زیرا از دیدگاه این اشخاص این زبان به صورت بالقوه بازگشتی به سمت کدهای اسپاگتی دارد که در زمانهای قدیم از دستورGo to ناشی می‌شد [۴۲].

یک برنامه‌نویس برای اینکه بتواند خطاهای برنامه را برطرف کرده یا از آن جلوگیری کند باید کد منبع را خوانده و آن را درک کند اما برنامه نویسی جنبه‌گرا سعی می‌کند که جزئیات فراخوانی متدها و رفتارهای خود

شمول[1] را نادیده بگیرد و این امر باعث می‌شود که برنامه‌نویس خوب کد منبع را جهت خطایابی درک نکند. از طرفی ایده اینکه جنبه‌ها بعدا بتوانند وارد سیستم شوند و رفتارهای جدیدی را به کد اضافه کند از دیگر موارد مختل کننده این موضوع است. از دیگر معایب یا موضوعات بحث برنگیز در مورد برنامه‌نویسی جنبه‌گرا می‌توان به میزان امنیت ترکیب جنبه‌ها در زمان اجرا، کامپایل یا بارگذاری و تاثیرات جانبی ناشی از تغییر نام یک تابع اشاره کرد که نهایتا منجر به این می‌شود که نتوان از جنبه مرتبط با تابع مورد نظر زیاد استفاده کرد.

۲-۱۳- توسعه نرم‌افزاری جنبه‌گرا[2]

همراه با برنامه‌نویسی جنبه‌گرا نیاز به روشی برای توسعه سیستم‌های نرم‌افزاری با جنبه‌ها از مرحله نیازمندیها تا تحلیل، طراحی و پیاده‌سازی است تا از این طریق بتوان نیازهای وظیفه‌مندی، غیروظیفه‌مندی و وابسته به سکو[3] را به بهترین شکل ممکن پیمانه‌بندی کرد. توسعه نرم‌افزاری جنبه‌گرا یک تکنولوژی توسعه نرم‌افزاری است که روشهای جدید برای پیمانه‌بندی سیستمهای نرم‌افزاری فراهم می‌گرداند. توسعه نرم‌افزاری جنبه‌گرا این امکان را می‌دهد که دغدغه‌های چندگانه یا مداخله‌ای به صورت جداگانه بیان شده و به صورت خودکار با سیستم (در حال اجرا) یکپارچه[4] شوند. در واقع تمرکز اصلی توسعه نرم‌افزاری جنبه‌گرا بر روی شناسائی، ذکر مشخصات، بیان (معرفی) و پیمانه‌بندی دغدغه‌های مداخله‌ای در داخل واحدهای عملیاتی مجزا بعلاوه ترکیب خودکار آنها در داخل سیستم است.

توسعه نرم‌افزاری جنبه‌گرا، دیگر پارادایم‌های توسعه نرم‌افزاری از قبیل برنامه نویسی شیءگرا، توسعه نرم‌افزاری مبتنی بر مولفه، معماری سرویس‌گرا و توسعه نرم‌افزاری مدل‌رانه را به وسیله فراهم کردن تجرید[5] واضح‌تر در جهت بدست آوردن نیازمندیها، موارد کاربری و دیگر خصیصه‌هایی که با استفاده از مکانیزم‌های تجزیه پایه‌ای زبان‌ها قابل پیمانه‌بندی نیست کامل می‌کند. توسعه نرم‌افزاری جنبه‌گرا فقط برنامه‌نویسی جنبه‌گرا نیست. بلکه یک محدوده گسترده‌ای از تکنیک‌ها را که به پیمانه‌بندی بهتر کمک می‌کند در بر می‌گیرد. این تکنیک‌ها شامل شیءگرائی، توسعه مبتنی بر مولفه، طراحی الگوها و چارچوب‌های[6] شیءگرا ازقبیل J2EE , Net. و غیره هستند. بنابراین توسعه نرم‌افزاری جنبه‌گرا با دیگر تکنیک‌ها رقابت نمی‌کند بلکه بر روی آنها ساخته می‌شود.

[1] Container-supplied
[2] Aspect-oriented software development
[3] Platform specifics
[4] Integrate
[5] Abstract
[6] Framework

۲-۱۳-۱- مهندسی نیازمندی جنبه‌گرا

مهندسی نیازمندی جنبه‌گرا که با نام جنبه‌های اولیه[1] نیز مورد ارجاع قرار داده می‌شود بر روی شناسائی، ذکر خصوصیات، استدلال درباره وابستگیها و معرفی دغدغه‌های مداخله‌ای در سطح نیازمندیها متمرکز است. مهندسی نیازمندی جنبه‌گرا انتزاع و مکانیزمهای ترکیب جدیدی برای جدا کردن دغدغه‌های مداخله‌ای فراهم می‌کند. مهندسی نیازمندیهای جنبه‌گرا همچنین از طریق مشخصات ترکیب[2]، این امکان را فراهم می‌کند که وابستگی دغدغه‌های مداخله‌ای با دیگر دغدغه‌ها بدست آید [٤٠]. علاوه بر این، مشخصات ترکیب می‌توانند برای تحلیل تداخل‌ها و تعاملات بالقوه استفاده شوند.

۲-۱۳-۲- معماری سیستم جنبه‌گرا

هدف از معماری سیستم جنبه‌گرا شناسائی و مشخص کردن دغدغه‌های مداخله‌ای در طراحی‌های معماری است. دغدغه‌های مداخله‌ای با توجه به ویژگیهای خود قابل پیمانه‌بندی در سطح معماری نیستند (در روشهای مرسوم)، از طرفی اگر این دغدغه‌های مداخله‌ای به وسیله تعریف مجدد معماری نرم‌افزار با توجه به مفاهیم (انتزاع‌های) روشهای معماری مرسوم پیمانه‌بندی شوند نتیجه مطلوبی نخواهد داشت [٤٣]. بدین منظور نیاز به روشهای جدید در سطح معماری است که از پیمانه‌بندی دغدغه‌های مداخله‌ای حمایت کند. این روشها که با نام معماری جنبه‌ای معرفی می‌شوند همانند مفهوم جنبه در سطح برنامه‌نویسی، دغدغه‌های مداخله‌ای را با نام جنبه‌های معماری در سطح معماری مورد توجه قرار می‌دهند. در واقع زبانهای معماری سیستم جنبه‌گرا مکانیزمهای روشنی برای شناسائی، بیان مشخصات و ارزیابی جنبه‌ها در سطح معماری پیشنهاد می‌دهند.

۲-۱۳-۳- طراحی و مدلسازی جنبه‌گرا

طراحی جنبه‌گرا اهدافی مشابه با اهداف فعالیت طراحی نرم‌افزار دارد (به عنوان مثال توصیف و مشخص‌کردن رفتار و ساختار سیستم نرم‌افزاری بخشی ار فعالیت‌های طراحی و مدلسازی است). طراحی جنبه‌گرا منحصرا بر این حقیقت متکی است که، دغدغه‌هایی که ضرورتا در شیوه‌های طراحی و مدلسازی مرسوم پراکنده و درهم تنیده می‌شدند قابل پیمانه‌بندی هستند. چنین شیوه‌هایی عموما شامل فرآیند و زبان خاص خود هستند که در آن، فرآیند نیازمندیها را به عنوان ورودی دریافت می‌کند و مدل طراحی را تولید می‌کند. مدل طراحی شده، دغدغه‌های مجزا را به همراه رابطه بین آنها نمایش می‌دهد. زبان نیز ساختارهایی فراهم می‌کند که از طریق آنها امکان نمایش عناصر موجود در طراحی و رابطه‌های ممکن بین آنها امکان‌پذیر می‌شود. به خصوص اینکه،

[1] Early aspects
[2] Composition specifications

ساختارها به این منظور فراهم شده‌اند تا از پیمانه‌بندی دغدغه‌ها و مشخصات ترکیب دغدغه‌ها با توجه به تداخل‌های ممکن حمایت کنند.

۲-۱۳-۴- برنامه‌نویسی جنبه‌گرا

اطلاعات پیرامون برنامه‌نویسی جنبه‌گرا در بخش ۲-۱۳ ذکر شده است.

۲-۱۳-۵- آزمایش برنامه‌های جنبه‌گرا

برنامه‌های تولید شده با استفاده از زبانهای جنبه‌گرا همانند دیگر برنامه‌ها نیاز به آزمایش دارند ولی با توجه به اینکه زبانهای جنبه‌گرا ویژگیهای جدید ارائه می‌کنند (دارا هستند) امکان بکارگیری روشهای آزمایش موجود برای این برنامه‌ها وجود ندارد (مناسب نیست). در واقع این امر ناشی از خصوصیات جنبه‌ها است که این خصوصیات به شرح زیر هستند[٤٤] :

- جنبه‌ها از نظر محتوا با دیگرکلاسها متفاوت هستند.
- جنبه‌ها به شدت به کلاسهایی که در آنها پوشیده شده‌اند وابسته هستند.
- هنگام تحلیل کد منبع کلاسها و جنبه‌ها وابستگیهای داده و کنترل واضح نیست.
- منابع ممکن زیادی برای یک شکست در سیستم وجود دارد.

با توجه به این خصوصیات جنبه‌ها، آزمایش برنامه‌های جنبه‌گرا با سئوالات زیادی همراه است که در آزمایش باید به آنها پاسخ داده شود. نمونه‌های از روشهای سیستماتیک برای آزمایش جنبه‌گرا در [٤٥] به اختصار بیان شده است.

۲-۱۴- شبکه‌های پتری

شبکه پتری یک روش ریاضی برای مدلسازی و ارزیابی محصولات نرم‌افزار است که برای اولین بار در سال ۱۹۶۲ توسط آقای کارل آدام پتری[1] معرفی شد. شبکه پتری مفاهیم دقیق و واضح، نماد گرافیکی قابل درک، و بسیاری تکنیک‌ها و ابزارها برای تحلیل خودش فراهم می‌کند. اساس شبکه‌های پتری، گرافها هستند در واقع ایده‌ی گرافها باعث شد تا کارل آدام پتری به مدل شبکه های پتری دست یابد.

۲-۱۴-۱- تعریف شبکه‌های پتری

شبکه‌های پتری، چند گراف‌های جهت‌دار دو قسمتی هستند.

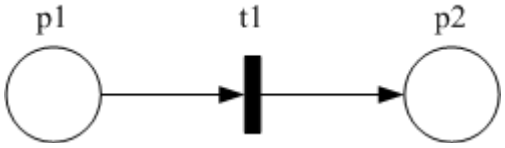
- دو قسمتی: گره‌های موجود در شبکه پتری دو نوع هستند:

[1] Carl adam petri

۱. گره دایره‌ای یا مکان[1]

۲. گره میله‌ای یا گذر[2]

- جهت‌دار: در شبکه پتری کمان‌هایی که دو گره را به یکدیگر متصل می‌کنند، جهت دارند. کمان‌ها فقط می‌توانند مکان‌ها را به گذرها متصل کنند و یا اینکه گذرها را به مکان‌ها متصل نمایند (شکل ۲-۲).



شکل ۲-۲: نحوه اتصال کمان‌ها و گذرها

- چند گراف: در شبکه‌های پتری می‌توان تعدادی کمان از یک مکان به یک گذر یا از یک گذر به یک مکان داشت.

گذرها برای نشان دادن مواردی مانند پردازنده، رویداد، مرحله محاسباتی یا الگوریتم، وظیفه یا کار و عبارت منطقی به کار می‌رود. مکان‌های ورودی برای نشان دادن بافرها، پیش شرط‌ها، داده ورودی، منابع مورد نیاز، شرایط، سیگنال‌های ورودی مناسب هستند. مکان‌های خروجی نیز برای نمایش بافرها، پیش شرط‌ها، داده خروجی، منابع آزاد شده، نتایج و سیگنالهای خروجی استفاده می‌شود. یک کمان از یک مکان به یک گذر می‌تواند به عنوان نمونه معانی زیر را تداعی کند:

- فرآیندی نیاز به منبعی در بافر دارد.
- رویدادی برای رخ دادن به صحت پیش شرط نیاز دارد.
- الگوریتمی به این داده ورودی نیاز دارد.
- وظیفه‌ای به این منبع نیاز دارد.
- این یک شرطی برای این عملیات منطقی است.

یک کمان از یک گذر به یک کمان می‌تواند در موراد زیر استفاده شود:

- فرآیندی یک منبع را در یک بافر خروجی تولید می‌کند.
- رویدادی، وقتی که اتفاق می‌افتد، این شرط را تولید می‌کند.

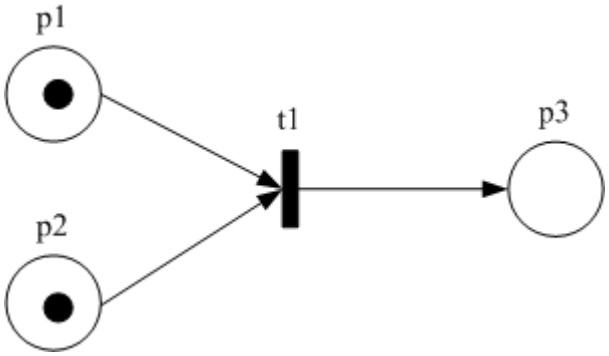
[1] Place
[2] Transition

- الگوریتمی، این داده خروجی را تولید می‌کند.
- وظیفه‌ای این منبع را تولید می‌کند.
- این نتیجه یک عبارت منطقی است.

۲-۱۴-۲- شبکه پتری علامت‌گذاری شده

یک شبکه پتری علامت‌گذاری شده شامل نشانه‌هایی[1] است. نشانه‌ها به صورت گرافیکی به وسیله نقاط سیاه رنگ تو پر در درون مکان‌ها مشخص می‌شوند. یک علامت‌گذاری از یک شبکه پتری نگاشتی است که یک عدد صحیح غیر منفی (تعداد نشانه‌ها) را به یک مکان از شبکه اختصاص می‌دهد. علامت‌گذاری، وضعیت شبکه پتری را مشخص می‌کند.

در شبکه پتری همه‌ی نشانه‌ها یکسان هستند و هیچگونه تمایزی بین آنها وجود ندارد. وجود یک یا بیشتر نشانه، در دسترس بودن منبع یا برقرار شدن یک شرط را نشان می‌دهد. در حالی که نبودن نشانه، نشان دهنده این است که شرط مورد نیاز ارضاء نشده است و یا اینکه منبع در دسترس نیست. مکان‌ها و علامت‌گذاری در نظر گرفته شده برای آن، طبیعت توزیع شده سیستمها را به خوبی نشان می‌دهد. شکل ۲-۳ یک مثال ساده‌ای از شیوه علامت‌گذاری در شبکه پتری را نشان می‌دهد.



شکل ۲-۳: شیوه علامت‌گذاری در شبکه پتری

۲-۱۴-۳- تعریف رسمی شبکه پتری

یک شبکه پتری می‌تواند با یک پنج‌تائی مرتب به صورت (P, T, I, O, M_0) توصیف شود:

- P : مجموعه متناهی از مکان‌ها
- T : مجموعه متناهی از گذرها

[1] Tokens

- I : مجموعه‌ای از کمان‌ها (از مکان‌ها به گذرها یعنی کمان‌های ورودی) که اصولا با یک تابع ورودی به صورت (I: P→ T) نشان داده می‌شود.
- O : مجموعه‌ای از کمان‌ها (از گذرها به مکان‌ها یعنی کمان‌های خروجی) که اصولا با یک تابع خروجی به صورت (O: T→ P) نشان داده می‌شود.
- M_0 : تابع علامت‌گذاری اولیه

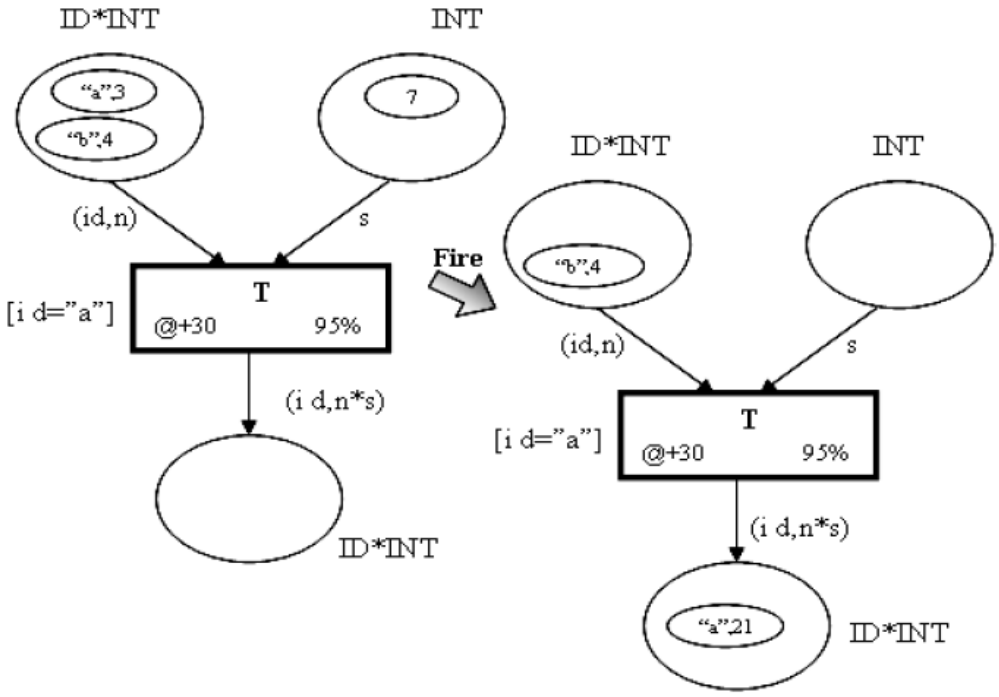
۲-۱۴-۴- شبکه‌های پتری رنگی[1]

شبکه پتری رنگی، به وسیله K.Jensen ارائه شد و در واقع یک نسخه توسعه یافته از شبکه پتری بود [۴۶]. در شبکه پتری رنگی مکان‌ها، گذرها و نشانه‌ها، مفاهیم رنگ‌ها، گاردها و عبارات برای اینکه بتوانند مقادیر داده‌ای محاسبه شده را به وسیله نشانه‌ها منتقل نمایند، معرفی شده‌اند. یک گذر در یک CPN شلیک[2] می‌شود اگر شرایط زیر فراهم شود:

۱. هر کدام از مکان‌های ورودی برای گذر، حداقل یک نشانه داشته باشد.
۲. برای نشانه‌ها در مکان‌های ورودی، عبارت لازم به کمان‌های ورودی الصاق شده باشند.
۳. گارد (عبارت بولی) برای انجام گذر، به مقدار مورد نظر مقداردهی شده باشد.

شکل ۲-۴ یک شبکه پتری رنگی ساده را نشان می‌دهد. همان طور که مشخص است شکلهای بیضی و مستطیل به ترتیب مکان‌ها و گذرها را نشان می‌دهند. نشانه‌ها با بیضی‌های کوچکتر در یک مکان نشان داده می‌شوند. مفهوم رنگ شبیه به تعریف نوع داده برای نشان دادن یک مجموعه از مقادیر داده قابل قبول است. مکان‌ها در یک شبکه می‌توانند نوع داشته باشند به طوری که رنگها می‌توانند به مکانها الصاق شوند. برای مثال فرض کنید که رنگ INT به مکانی الصاق شده است یعنی در این مکان نشانه‌ها از هر نوع داده‌ای نمی‌تواند قرار بگیرد و فقط نشانه از نوع INT (عدد صحیح) می‌تواند در این مکان قرار بگیرد. در شکل ۲-۴ عبارت (id, n) و s به کمان‌ها الصاق شده‌اند تا جریان داده از یک مکان به یک گذر را نشان دهند که (id, n) به صورت جفت از مقادیر (چندتائی) است که یکی از آنها از نوع ID و دیگری از نوع INT است. و این مقادیر به ترتیب به متغییرهای "id" و "n" مقداردهی شده‌اند. عبارت ["id="a""] یک گارد نامیده می‌شود. در این مثال وقتی که نشانه ("a",3") در مکان ورودی وجود دارد گذر می‌تواند شلیک شود. بعد از انجام گرفتن گذر، نشانه‌ای در مکان خروجی با مقدار ("a",12) ظاهر می‌شود که حاصل محاسبه عبارت (id, n*s) است.

[1] Coloured Petri Nets
[2] Fire



شکل ۲-۴: یک مثال از شبکه پتری رنگی

۲-۱۴-۵- شبیه‌سازی و تحلیل شبکه‌های پتری

ابزارهای مختلفی برای شبیه‌سازی و تحلیل شبکه‌های پتری وجود دارند که برای این تحقیق از CPN/Tools [۴۷] استفاده خواهد شد. CPN/Tools یک ابزاری برای تحلیل، شبیه‌سازی و ویرایش شبکه‌های پتری رنگی است. این ابزار دارای یک صفحه طراحی است که در آن کاربر می‌تواند شبکه پتری مورد نظر را طراحی کند. بعد از طراحی، کاربر می‌تواند بر حسب نیاز از ویژگیهای بازبینی[1] یا شبیه‌سازی[2] استفاده کند. در حالت بازبینی کاربر می‌تواند کلیه ورودی و خروجی‌های مربوط به گذرها، مکان‌ها و کمان‌ها را بررسی کرده و در صورت نیاز خروجی در قالب یک فایل از آنها بگیرد. به منظور گرفتن خروجی یا خروجی با فرمت خاص کاربر لازم است اسکریپ‌های مناسب را در بخشهای مورد نظر در ابزار وارد کند. در حالت شبیه‌سازی نیز، ابزار یک محیط گرافیکی قابل اجرا فراهم می‌کند که در آن کاربر می‌تواند مدل را به صورت کلی یا گام به گام اجرا کرده و وضعیت مدل و خروجی را مشاهده کند. البته برای شبیه‌سازی و اجرا مدل، باید مدل نهایی کاملا صحیح مدلسازی شده و نشانه‌های مورد نظر در مکان مناسب قرار داده شوند.

[1] Monitioring
[2] Simulation

۲-۱۴-۶- جنبه‌گرائی و شبکه‌پتری

همان طور که پیش‌تر ذکر شد جنبه‌گرائی در جهت پیمانه‌بندی هر چه بهتر سیستم‌های نرم‌افزاری حرکت می‌کند و این عمل را با استفاده از مفهوم جنبه انجام می‌دهد از طرفی امروزه برای اینکه بتوان میزان خطاها در سیستم تولید شده را کاهش داده و قابلیت اطمینان آن را افزایش داد از مدلسازی استفاده می‌کنند که شبکه پتری یک زبان مدلسازی برجسته در زمینه به شمار می‌رود.

با توجه به قابلیت مدلسازی و شبیه‌سازی شبکه پتری، بسیار کارا خواهد بود که بتوان مفاهیم جنبه‌گرائی را در قالب شبکه پتری نمایش داد. در واقع مدلسازی جنبه از طریق شبکه پتری میزان اطمینان به اینکه دغدغه‌ای به عنوان یک جنبه در نظر گرفته شده یا اینکه، اطمینان در مورد مولفه‌هایی که تحت تاثیر یک جنبه قرار می‌گیرند، را افزایش می‌دهد. به عبارت دیگر استفاده از شبکه‌پتری در مفهوم جنبه‌گرائی باعث می‌شود که یک روش سیستماتیک جایگزین روشی شود که در آن از چندین سناریو از پیش تعریف شده استفاده می‌شد.

از بعد دیگر قابلیت شبیه‌سازی شبکه پتری این امکان را می‌دهد که با مدلسازی جنبه‌ها در قالب شبکه جنبه‌ها تداخل‌های حاصل از اعمال جنبه‌های مختلف در یک نقطه اتصال را کشف کرد که این موضوع می‌تواند نقش موثری در سیستم‌های همروند یا سیستم خیلی بزرگ داشته باشد. مهمتر از همه اینها، شبکه پتری قابلیت بیان کردن مفاهیم جنبه‌گرا به صورت رسمی آن هم در قالب نماده‌هایی که قابلیت فهم بالائی دارند را امکان‌پذیر می‌سازد. بنابراین برقرای ارتباط میان این دو حوزه می‌تواند نقش موثری در توسعه سیستم‌های کارآمد و مطمئن داشته باشد.

۲-۱۵- خلاصه

در این فصل به معرفی ادبیات، موضوعات و عناوین مرتبط با تحقیق پرداخته شد. در این راستا، ابتدا نیازمندیها، دغدغه‌ها و دغدغه‌های مداخله‌ای معرفی شدند و ذکر شد که دغدغه‌های مداخله‌ای قابلیت محصور شدن درون یک مولفه را ندارند. سپس مفاهیم جداسازی دغدغه‌ها، خاصیت پیمانه‌بندی و دو تکنیک برجسته در زمینه جداسازی دغدغه‌ها به نام موارد کاربری و دیدگاه‌ها معرفی شدند. در ادامه، تعاریف مربوط به دو قابلیت مهم در سیستمهای نرم‌افزاری تحت عنوان قابلیت نگهداری و قابلیت ردیابی بیان شدند.

برای درک علت پیدایش جنبه‌گرائی دو مشکل اصلی روشهای توسعه سنتی سیستمهای نرم‌افزار تحت عنوان پراکندگی و درهم‌تنیدگی نیز تعریف شدند. سپس برنامه‌نویسی جنبه‌گرا به عنوان محرک توسعه‌نرم‌افزاری به همراه مفاهیم پایه، مزایا و معایب بررسی شد. همچنین توسعه نرم‌افزاری جنبه‌گرا معرفی و فازهای آن به صورت مختصر تشریح شدند. نهایتا شبکه پتری به عنوان یک ابزار اصلی در روش پیشنهادی برای شناسائی جنبه‌ها، در حد مورد نیاز مورد بررسی قرار گرفت.

فصل سوم

جایگاه مهندسی نیازمندیها

در این فصل هدف بررسی مهندسی نیازمندیها است. بدین منظور ابتدا بیان می‌شود که هدف از مهندسی نیازمندیها چیست و چرا باید مهندسی نیازمندیها نخستین مرحله از فرآیند توسعه یک سیستم نرم‌افزاری باشد. سپس فعالیت‌های مرتبط با مهندسی نیازمندیها معرفی می‌شوند و شرح داده می‌شود که هدف از هر یک از فعالیت‌ها چیست و هر فعالیت چه ورودی و خروجی دارد. همچنین تکنیک‌های مورد استفاده در هر یک از فعالیت‌ها مشخص شده و چگونگی تحقق هر تکنیک بیان می‌شود.

۳-۱- مقدمه

مهندسی نیازمندیها مربوط است به شناسائی، مدلسازی، مرتبط‌سازی[1] و مستندسازی نیازمندیها برای یک سیستم و همچنین زمینه‌ای[2] که در آن سیستم به کار گرفته خواهد شد. نیازمندیها چیزی را که باید انجام شود تشریح می‌کنند ولی چگونگی پیاده‌سازی آن را مشخص نمی‌کنند [٤٨].

تکنیکهای زیادی برای استفاده در طی فرآیند مهندسی نیازمندیها وجود دارد تا تضمین کند نیازمندیها کامل، سازگار و مناسب هستند. هدف از مهندسی نیازمندیها در مدل آبشاری کمک کردن به شناسائی چیزهایی است که باید ساخته شوند قبل از اینکه توسعه سیستم آغاز شود تا از دوباره کاریهای پرهزینه جلوگیری شود. این هدف مبتنی بر دو فرض مهم است:

- آخرین اشتباهات (خطاها) کشف شده هزینه زیادی را برای اصلاح می‌طلبند [٤٩].
- امکان‌پذیر است که قبل از شروع طراحی و پیاده‌سازی، یک مجموعه پایدار (با ثبات) از نیازمندیها تعیین شوند.

در جهت چگونگی مشخص کردن نیازمندیها در مراحل آغازین توسعه یک سیستم نرم‌افزاری، در ادامه فرآیند مهندسی نیازمندیها به صورت کامل بررسی شده و تکنیک‌های اصلی که برای آنها توسعه داده شده‌اند مطرح می‌شوند.

۳-۲- فرآیند مهندسی نیازمندیها

فرآیند مهندسی نیازمندیها از پنج فعالیت اصلی تشکیل شده است [٤٩] :

- استخراج
- مذاکره و تحلیل
- مستندسازی
- اعتبارسنجی
- مدیریت

فعالیت‌ها می‌توانند در مدلهای مختلف ارائه شوند. مشهورترین مدلها عبارتند از :

۱. مدل فعالیت Coarse-Grain [٤٩]

[1] Communicating
[2] Contexts

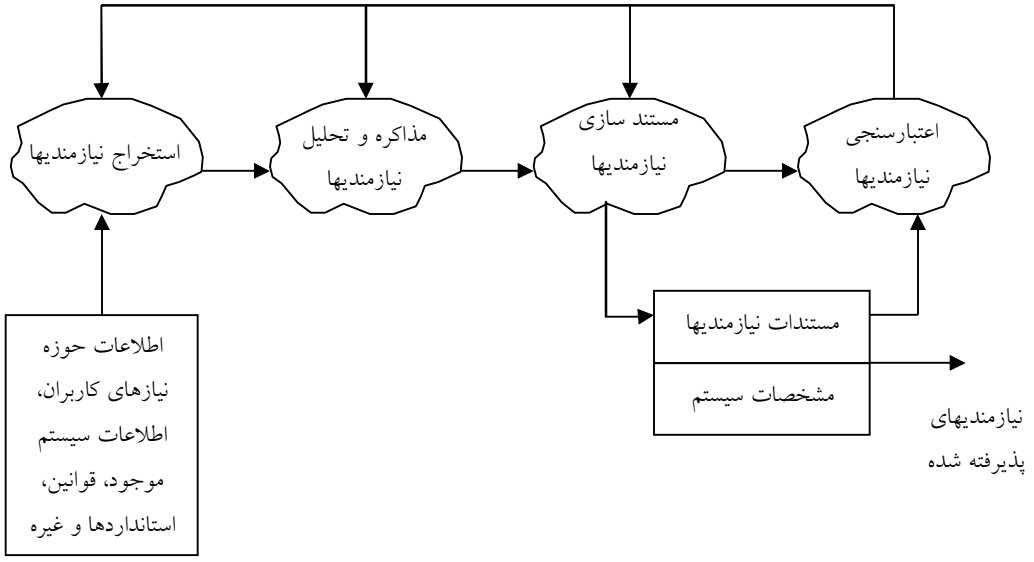
این مدل در شکل ۳-۱ نشان داده شده است. در شکل ۳-۱ نشانه‌های ابری شکل دلالت بر این موضوع دارند که مرز مشخصی بین فعالیت‌ها وجود ندارد.

۲. مدل آبشاری

این مدل، یک ترتیبی از فازهای مهندسی نیازمندیها را مشخص می‌کند که پشت سر هم انجام می‌شوند.

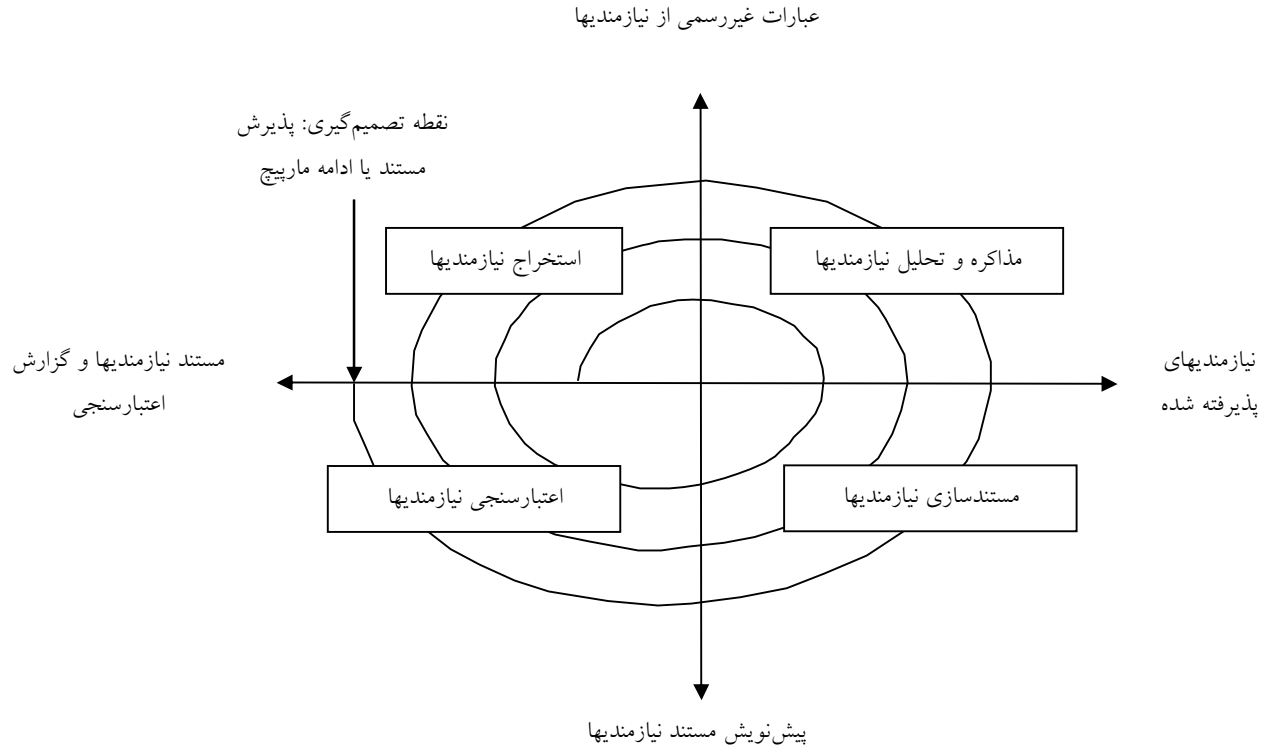
۳. مدل مارپیچی یا حلزونی[1] [۴۹]

این مدل در شکل ۳-۲ به تصویر کشیده شده است که نشان می‌دهد فعالیت‌های مختلف مهندسی نیازمندیها تا زمانی باید تکرار شوند که یک تصمیم‌گیری اتخاذ شود در مورد اینکه مستندات نیازمندی قابل قبول هستند.

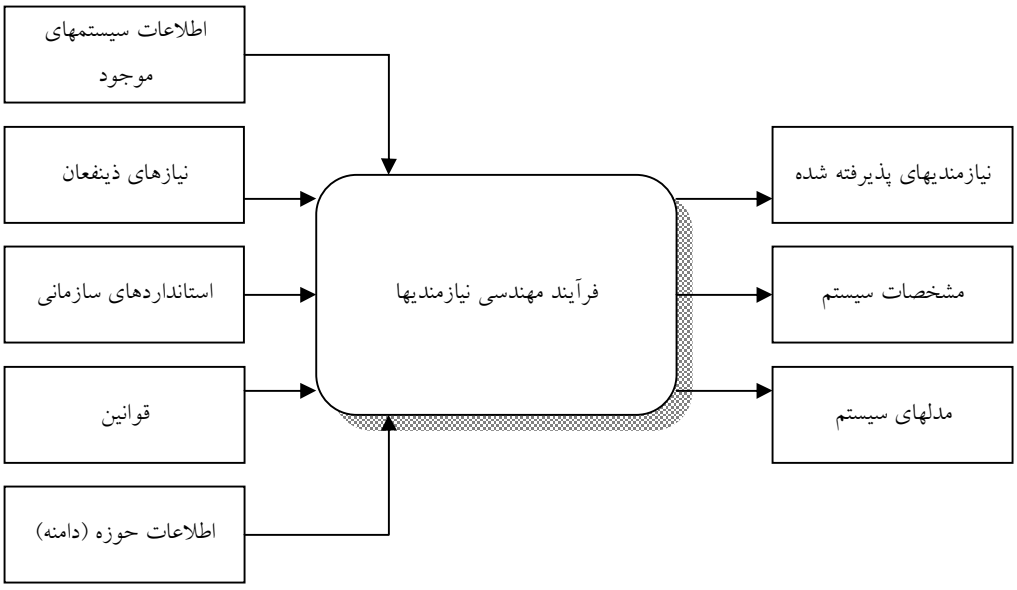


شکل ۳-۱: مدل فعالیت Coarse-Grain

[1] Spiral



شکل ۳-۲: مدل مارپیچی یا حلزونی



شکل ۳-۳: فرآیند مهندسی نیازمندیها- ورودی‌ها و خروجی‌ها

ورودی‌ها و خروجی‌های فرآیند مهندسی نیازمندیها در شکل ۳-۳ نشان داده شده است. اگر چه فرآیند مهندسی نیازمندیها در سازماندهی‌های مختلف متفاوت است ولی ورودی‌ها و خروجی‌ها در بیشتر حالات یکسان هستند. مهندسی نیازمندیها کمک می‌کند که خطاها زودتر کشف شوند که این امر منجر به کاهش هزینه‌های توسعه نرم‌افزار می‌شود زیرا برای اصلاح خطاهایی که در بخش‌های نهایی توسعه کشف می‌شوند باید هزینه زیادی صرف شود.

۳-۳- استخراج نیازمندیها

استخراج به شناسائی نیازمندیها و محدودیت‌های سیستم از طریق مشاوره یا رایزنی با ذینفعان (مانند: مشتریان، توسعه‌دهندگان و کاربران) ارجاع دارد. محدودیت‌های سیستم، کلیه تکنیک‌های استخراج را تحت تاثیر قرار داده و زمینه سیستم ارائه شده را تعریف می‌کند. درک کامل حوزه‌های مختلف (مانند: حوزه کاربرد، نیازهای حرفه و محدودیت‌های ذینفعان سیستم و خود مسأله) برای به تصویر کشیدن سیستمی که باید توسعه داده شود ضروری است. مهمترین تکنیک‌ها برای استخراج نیازمندیها در ادامه همین بخش تشریح شده‌اند.

۳-۳-۱- مصاحبه‌ها

مصاحبه یک روشی برای کشف واقعیت‌ها و نظراتی است که ذینفعان و کاربران سیستم در حال توسعه دارند. که به واسطه آن اشتباهات و درک‌های نادرست می‌توانند شناسائی و اصلاح شوند. دو نوع مختلف از مصاحبه وجود دارد:

- مصاحبه بسته[1]، در این نوع مصاحبه مهندس نیازمندیها یک سری سئوالات از پیش تعریف شده دارد و به دنبال جواب‌های آنها است.
- مصاحبه باز[2]، در این نوع مصاحبه هیچ سئوال از پیش تعریف شده وجود ندارد و در آن مهندس نیازمندیها و ذینفعان به صورت باز (پایان مصاحبه مشخص نیست) با یکدیگر پیرامون اینکه، چه انتظاراتی آنها (ذینفعان) از سیستم دارند، بحث می‌کنند.

در حقیقت، اغلب هیچ مرز مشخصی بین انواع مصاحبه‌ها وجود ندارد. مصاحبه با یک سری از سئوالات که بحث شده آغاز می‌شود و این منجر به سئوالات جدیدی می‌شود [٤٩]. مزیت مصاحبه آن است که به توسعه‌دهنده کمک می‌کند که مجموعه غنی از اطلاعات را بدست آورد و اشکال مصاحبه آن است که تحلیل

[1] Closed interview
[2] Open interview

این مقدار از داده‌های کیفی می‌تواند مشکل باشد و ذینفعان مختلف می‌توانند اطلاعات تداخلداری را مشخص کنند.

۳-۳-۲- موارد کاربری و سناریو

سناریوها مثالهایی از نشست‌های[1] تعاملی هستند که یک نوع منفرد از تعامل بین کاربر و سیستم شبیه‌سازی شده است. سناریوها باید موارد زیر را داشته باشند:

۱. توضیحی از وضعیت سیستم قبل از ورود و بعد از کامل شدن سناریو
۲. فعالیت‌هایی که می‌توانند همزمان رخ دهند.
۳. جریال عادی از وقایع و استثناءهای مربوط به رویدادها

اطلاعات مورد نیاز پیرامون موارد کاربری در بخش ۲-۶ ذکر شده است.

۳-۳-۳- طوفان ذهنی[2]

طوفان ذهنی قابلیتی برای توسعه راه‌حلهای مربوط به یک عنوان خاص است. معمولا، طوفان ذهنی یک گروه از فعالیت‌ها است اما طوفان ذهنی می‌تواند بر روی یک شخص نیز صورت پذیرد. طوفان ذهنی شامل دو فاز است. فاز تولید[3]، که در آن ایده‌ها جمع‌آوری می‌شوند و فاز ارزیابی[4]، که ایده‌های جمع‌آوری شده ارزیابی می‌شوند. در فاز تولید، ایده‌ها نباید ارزیابی و مورد نقد قرار داده شوند. ایده‌ها باید سریع ایجاد شده، و گسترده و خاص باشند. هر ایده می‌تواند منجر به ایده‌های جدید شود. همچنین طوفان ذهنی منجر به این می‌شود که هر شخص مسأله را به خوبی درک کرده و یک احساس مشترک از نتایج داشته باشد.

۳-۳-۴- مشاهده و تحلیل معاشرتی[5]

روشهای مشاهده‌ای، یک مامور[6] (محقق) را درگیر بررسی کاربران می‌کند آن هم هنگامیکه آنها در یک مطالعه میدانی[7] فعالیت (کار) می‌کنند. مامور رسیدگی، از فعالیت‌هایی که کاربران انجام می‌دهند یاداشت‌برداری می‌کند. روش مشاهده‌ای برای به دست آوردن داده‌های کیفی در مشخصات نیازمندیهای کاربران و همچنین مطالعه کارهای انجام شده و فرآیندها مفید است.

---

[1] Sessions
[2] Brainstorming
[3] Generation
[4] Evaluation
[5] Observation and social analysis
[6] Investigator
[7] Field study

مشاهده می‌تواند به دو صورت مستقیم[1] یا غیرمستقیم در نظر گرفته شود. در مشاهده مستقیم، محقق در طی فعالیت کاربران حضور دارد ولی در حالت غیرمستقیم، کار انجام شده به وسیله ابزارهای دیگر مانند دستگاه ضبط ویدئو بررسی می‌شود. مشاهده، به مشاهده کننده اجازه می‌دهد تا فعالیت‌هایی که کاربران واقعا در زمینه کاری خود انجام می‌دهد را بررسی کند و بر پیامدهای ناشی از تشریح ایده‌ال ذینفعان یا فرآیند کاری ساده شده آنها غلبه کند.

مشاهده مستقیم اجازه می‌دهد که محقق بر روی حوزه خاص مورد توجه خود متمرکز شود. مشاهده غیرمستقیم نیز فعالیت‌ها را ثبت می‌کند و اجازه می‌دهد که پیامدهایی که ممکن است در مشاهده مستقیم مورد توجه واقع نشده باشند شناسائی شوند.

۳-۳-۵- گروه‌های متمرکز[2]

گروه‌های متمرکز یک تکنیک غیررسمی است که در آن یک گروهی متشکل از ٤ تا ۹ کاربر از زمینه‌های مختلف با مهارتهای گوناگون در یک شکل آزاد پیرامون پیامدها و دغدغه‌های مربوط به ویژگیهای سیستم یا یک نمونه[3] از سیستم بحث می‌کنند. گروه‌های متمرکز کمک می‌کند تا نیازها و احساسهای کاربران- آنها چه فکر می‌کنند، چیزهایی که برای آنها مهم است و چیزهایی که از سیستم می‌خواهند- شناسائی شوند. آنها اغلب به خودی خود ایده‌ها و تعاملاتی[4] مطرح می‌کنند.

مدیر این گروه باید از یک طرح از پیش مشخص شده با عناوین مختلف پیروی کند که در هر بار یکی از عنوانها معرفی می‌شود تا دیدگاه افراد نسبت به آن عنوان مشخص شود. شرکت کنندگان در جلسه باید احساس آزادی و راحتی در جلسه داشته باشند. از آنجایی که اغلب ناهمخوانی‌های مهمی بین چیزی که افراد ذکر می‌کنند و چیزی که آنها انجام می‌دهند، وجود دارد مشاهده و تحلیل معاشرتی باید در این تکنیک مورد استفاده قرار بگیرد. گروه‌های متمرکز می‌تواند از بیان دیدگاه‌ها[5]، پشنهادهای طراحی[6] و مفهوم تولید[7] حمایت کند. بعلاوه، آنها به کاربران در تحلیل مشکلات خودشان و چیزهایی که باید تغییر یابند کمک می‌کنند و از توسعه یک مفهوم مشترک[8] از سیستم حمایت می‌کنند [۵۰، ۵۱].

[1] Direct
[2] Focus groups
[3] Prototype
[4] Interaction
[5] Articulation of visions
[6] Design proposals
[7] Product concept
[8] Shared meaning

۳-۳-۶- متدولوژی سیستم نرم[1]

متدولوژی سیستم نرم بر روی سیستم مطلوب و اینکه چگونه به آن دست پیدا کند، متمرکز است و از تحلیل وضعیت مسأله[2] از دیدگاه‌های مختلف حمایت می‌کند. متدولوژی سیستم نرم می‌تواند زمانی استفاده شود که یک وضعیت (مسأله) در زندگی روزمره وجود دارد و دست کم یک نفر مسأله را می‌بیند و می‌خواهد آن را بهبود ببخشد. متدولوژی سیستم نرم به جای اینکه به خود سیستم نگاه کند به شخص و زمینه سازمانی که سیستم مطلوب در داخل آن اجرا خواهد شد، نگاه می‌کند. مدل مرسوم متدولوژی سیستم نرم از هفت مرحله اصلی تشکیل شده است [۵۱، ۵۲]:

۱. تشریح وضعیت مسأله

چه کسی درگیر است و دیدگاه آنها از وضعیت مسأله چیست؟ زمینه سازمانی چیست؟

۲. تصویر دنیا[3]

وضعیت مسأله در یک روشی بیان می‌شود که کمک کند سیستم مناسب در مرحله ۳ انتخاب شود.

۳. تعریف پایه از سیستم مطلوب (مربوطه)

- تعریف خیالی[4] سیستمهایی که مناسب برای وضعیت مسأله هستند.
- راهنمایی برای تعریف پایه خوب فرموله شده، CATWOE [۵۱]

C[5]: مشتریانی که از سیستم سود می‌برند.

A[6]: عاملهایی که فعالیت‌های تعریف شده را اجرا می‌کنند.

T[7]: تبدیلات ورودی‌ها و خروجی‌ها

W[8]: جهان‌بینی (دید از دنیا)

O[9]: دارنده قدرت برای موفقیت با سیستم یا مجزا کردن سیستم

E[10]: محدودیتهای محیطی

[1] Soft System Methodology (SSM)
[2] Problem situation
[3] Picture of world
[4] Imaginary
[5] Customers
[6] Actors
[7] Transformations
[8] Weltanschauung
[9] Owner
[10] Environmental

۴. مدل مفهومی

یک مدل فعالیت انسانی که وابستگیهای بین فعالیت‌هایی که دقیقا منطبق با تعریف پایه هستند را نشان می‌دهد.

۵. مقایسه‌ای از دنیای واقعی یک مدل

فعالیت‌ها مقایسه می‌شوند با چیزهایی که در دنیای واقعی اتفاق می‌افتند. سئوالات خاص که برای هر فعالیت پرسیده می‌شوند عبارتند از :

- فعالیت در دنیای واقعی انجام می‌شود؟
- چگونه فعالیت انجام می‌شود؟
- چگونه کارائی می‌تواند اندازه‌گیری شود؟
- فعالیت به طور مناسب پردازش شده است؟

۶. شناسائی تغییرات مطلوب و امکان‌پذیر[1]

کدام فعالیت از نظر فرهنگی امکان‌پذیر[2] و از نظر سیستماتیک مناسب[3] است

۷. کنش (اقدام) برای بهبود دادن وضعیت مسأله

۳-۳-۷- استفاده مجدد نیازمندیها

روش استفاده مجدد نیازمندیها از مشخصات نیازمندیها یا بخشهایی که قبلا برای پروژهای پیشین به کار گرفته شده، استفاده مجدد می‌کند. استفاده مجدد نیازمندیها می‌تواند با استفاده از قالب‌ها، مقایسه و ساده شود تا زمانی که مسأله هر دو پروژه کاملا مشابه باشد [۵۳]. استفاده مجدد نیازمندیها می‌تواند تلاشهای عمومی مربوط به مرحله نیازمندیها را کاهش دهد [٥٤]. استفاده مجدد از نیازمندیها فقط در حالت‌های خاص امکان‌پذیر است:

- نیازمندی دربرگیرنده اطلاعات درباره حوزه کاربرد است

  نیازمندیها معمولا یک قابلیت عملکرد خاص از سیستم را مشخص نمی‌کنند اما محدود به سیستم یا عملکرد سیستم هستند که از حوزه کاربرد مشتق می‌شود.
- نیازمندیها با سبک ارائه مرتبط هستند

  نیازمندیها برای محصولات نرم‌افزاری توسعه داده شده برای یک شرکت خاص "احساس و ظاهر[4]" ویژه دارند.

[1] Identifying feasible and desirable changes
[2] Culturally feasible
[3] Systematically desirable
[4] Look and feel

- نیازمندیها سیاست‌های شرکت را بازتاب می‌کنند (سیاست‌های امنیتی).

۳-۳-۸- نمونه‌سازی یا نمونه اولیه[1]

یک نمونه اولیه از یک سیستم، یک نسخه اولیه از سیستمی است که در ابتدای فرآیند توسعه قابل دسترس است. نمونه‌های اولیه از سیستم‌های نرم‌افزاری اغلب برای کمک به استخراج و اعتبارسنجی نیازمندیهای سیستم مورد استفاده قرار می‌گیرند. دو نوع مختلف از نمونه‌سازی اولیه وجود دارد:

- نمونه دور انداختنی

به استخراج نیازمندیهایی کمک می‌کند که مشکلاتی در درک و فهم آنها وجود دارد.

- نمونه تکاملی

یک سیستم قابل کارکرد برای مشتری تحویل می‌دهد و می‌تواند به عنوان بخشی از سیستم نهایی در نظر گرفته شود.

این نمونه‌های اولیه می‌توانند به عنوان یک نمونه اولیه کاغذی[2] پیاده‌سازی شوند که در این حالت، یک مدل با اندازه واقعی از سیستم توسعه داده می‌شود و برای آزمایش سیستم مورد استفاده قرار می‌گیرد. یا به عنوان یک نمونه اولیه "Wizard of Oz" که در این صورت یک شخص، پاسخ سیستم را در جوابگوئی به بعضی ورودیهای کاربر شبیه‌سازی می‌کند. یا اینکه به عنوان یک نمونه اولیه مکانیزه[3]، که در این صورت یک زبان نسل چهارم یا بعضی محیط توسعه سریع[4] برای توسعه یک نمونه اولیه قابل اجرا استفاده می‌شود.

۳-۴- مذاکره و تحلیل نیازمندیها

تحلیل نیازمندیها، نیازمندیها را برای ضرورت (نیاز برای نیازمندی)، سازگاری (نیازمندیها نباید تناقض داشته باشند)، تمامیت (هیچ سرویس یا محدودیت نباید نادیده گرفته شود) و امکان‌پذیر بودن (تحقق بخشیدن نیازمندیها در محدوده بودجه و زمانبندی برای توسعه سیستم امکان‌پذیر باشد) بررسی می‌کند. تداخل‌ها در نیازمندیها از طریق اولویت‌بندی مذاکرات برطرف می‌شود. در مورد نیازمندیهایی که به نظر می‌رسند مشکل یا ابهاماتی وجود دارند بحث می‌شود و ذینفعان مربوطه دیدگاه‌های خود را پیرامون نیازمندیها ارائه می‌کنند. نیازمندیهای بحث برانگیز اولویت‌بندی می‌شوند تا نیازمندیهای مهم و انتقادی شناسائی شده و همچنین به فرآیند تصمیم‌گیری کمک شود. راه حلها برای مشکلات نیازمندیها شناسائی شده و یک مجموعه از نیازمندیهای

[1] Prototyping
[2] Paper prototype
[3] Automated prototype
[4] rapid

مورد توافق همه برای آن مشکلات در نظر گرفته می‌شود. عموما، این عمل با ایجاد تغییرات در بعضی از نیازمندیها همراه خواهد بود. تکنیک‌های اصلی که برای تحلیل نیازمندیها استفاده می‌شود نشستهای JAD[1]، اولویت‌بندی[2] و مدلسازی هستند.

۳-۴-۱- توسعه کاربردی مشترک (JAD)

JAD در سال ۱۹۷۷ به وسیله IBM توسعه داده شد و برای اولین بار در سال ۱۹۸۰ مورد استفاده قرار گرفت. JAD یک نشست گروهی تسهیل شده (یا کارگاه آموزشی) با یک روش تحلیل ساخته یافته است. هدف از JAD تعریف یک پروژه خاص بر سطوح مختلف از جزئیات است تا اینکه علاوه بر طراحی یک راه حل، پروژه را تا اتمام کامل مانیتور کند. شرکت کنندگان شامل مدیرهای اجرائی پروژه، کاربران، خبرگان سیستم و افراد تکنیکی خارجی هستند [۵۱]. نقشهای مختلفی وجود دارد که باید در هر نشست گروه وجود داشته باشند که عبارتند از :

- یک رهبر نشست
- یک نماینده کاربر
- یک متخصص
- یک تحلیل‌گر
- یک نماینده سیستم اطلاعاتی
- یک حامی[3] (سرپرست) اجرائی

رهبر نشست مسئول برگزاری دوره فرآیند است (نشست‌ها). رهبر باعث تسهیل بحث نشست و آماده‌سازی مستندسازی می‌شود [۵۱].

۳-۴-۲- اولویت‌دهی نیازمندیها

در یک پروژه با زمانبندی سفت و سخت، منابع محدود و انتظارات بالای مشتری، ضروری است که با ارزش‌ترین ویژگیها هر چه زودتر تحویل داده شوند. زمانیکه که مدت انجام پروژه در حال اتمام باشد و تمامی ویژگیها پیاده‌سازی نشده باشند بعضی از ویژگیها اجبارا حذف می‌شوند. تنظیم اولویت‌بندی در مراحل اولیه پروژه به این تصمیم کمک می‌کند که کدام ویژگیها نادیده گرفته شوند. اولویت‌دهی نیازمندیها باید توسط مشتری صورت بپذیرد. اگر گروه ثالثی این تصمیمات را برای مشتری انجام دهند ممکن است مشتری با این

[1] Join Application Development (JAD) sessions
[2] Prioritization
[3] Sponsor

اولیت‌بندی موافق نباشد. هر دوی مشتری و توسعه‌دهنده مجبور هستند ورودیهایی برای اولیت‌بندی نیازمندیها فراهم کنند. مشتری آن ویژگیهایی را با اولویت بالا مشخص می‌کند که بیشترین سود را به کاربر به ارمغان می‌آورد. توسعه‌دهنده مشکلات، هزینه و ریسک‌های تکنیکی را مد نظر قرار می‌دهد. با این دورنما، مشتری ممکن است اولویت بعضی از ویژگیها را تغییر دهد. همچنین توسعه دهنده ممکن است پیاده‌سازی ویژگی را پیشنهاد دهد که بیشترین تاثیر را روی معماری سیستم دارد اما این ویژگی قبلا کمترین اولویت را داشته باشد (از طرف مشتری مشخص شده است) [۵۵].

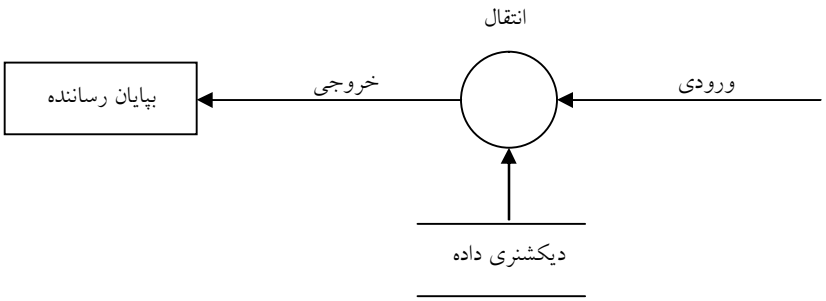
اولویت‌بندی یک تکنیک عمومی در مهندسی نیازمندیها است. تکنیک‌های گوناگون اولویت‌بندی (مانند مقایسه دو به دو[1] یا فرآیند سلسله مراتبی تحلیل[2]) سالیان طولانی است که در مهندسی نیازمندیها استفاده می‌شود.

۳-۴-۳- مدلسازی

مدلهای سیستم یک پل (اتصال) مهم بین فرآیند تحلیل و طراحی هستند. برخی از روشها از تکنیک‌های مدلسازی مختلف برای فرموله کردن یا تحلیل نیازمندیهای سیستم استفاده می‌کنند. رایج‌ترین تکنیک‌های مدلسازی عبارتند از [٤٩] : مدلهای جریان داده، مدلهای داده مفهومی، روشهای شیءگرا.

۳-۴-۳-۱- مدلسازی جریان داده :

مدل جریان داده مبتنی بر علامت‌گذاری است که در آن سیستم می‌تواند به صورت یک مجموعه از توابع تعاملی[3] مدلسازی شود. این روش، از نمودار جریان داده برای نمایش موجودیت‌های خارجی، فرآیندها، جریان داده و مخزن داده‌ها استفاده می‌کند (شکل ۳-۴ نمادهای نمودار جریان داده را نشان می‌دهد [۴۹]).

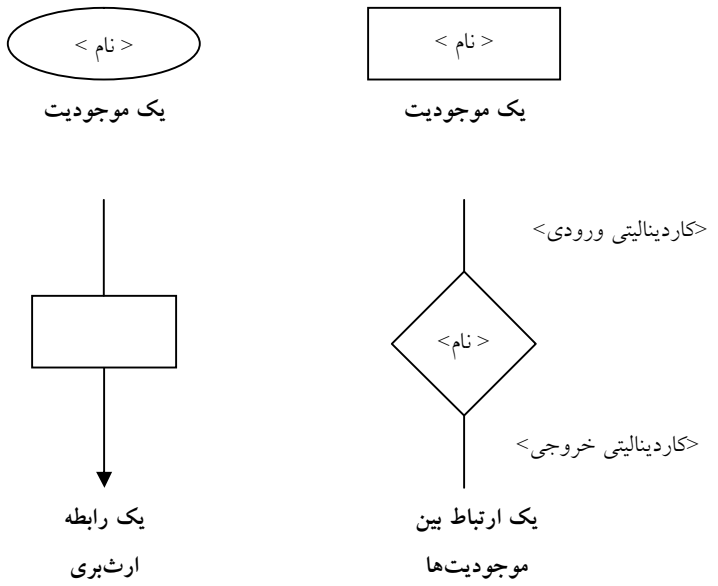


شکل ۳-۴: نمادهای نمودار جریان داده

[1] Pair-wise
[2] Analytic Hierarchy Process (AHP)
[3] Interacting functions

۳-۴-۳-۲- مدل داده مفهومی

یک مدل سیستم باید همچنین شکل منطقی داده‌ای را که به وسیله سیستم پردازش می‌شود تشریح کند. این عمل می‌تواند توسط مدل رابطه‌ای انجام شود که در آن داده به صورت یک مجموعه از جداول- با تعدادی ستون که کلیدهای مشترک را بدون توجه به سازماندهی فیزیکی پایگاه‌داده نمایش می‌دهد- مشخص شده است. مدل داده مفهومی شامل مدل رابطه-موجودیت است که در آن موجودیت در یک پایگاه داده، خواص متعلق به آن و ارتباطات بین آنها شناسائی می‌شوند (شکل ۳-۵ نمادها برای مدلهای داده مفهومی را نشان می‌دهد[٤٩]).

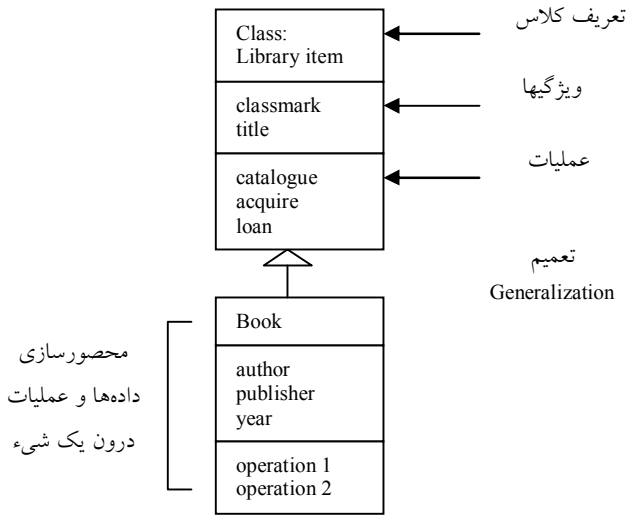


شکل ۳-۵: نمادهای مدل داده مفهومی

۳-۴-۳-۳- روشهای شیءگرا

پایه اصلی یک مدل شیءگرا ،مفهوم شیء است. یک شیء، یک مجموعه از ویژگیها مشترک و عملیات مرتبط با آن را تعریف می‌کند (شکل ۳-٦) [٤٩]. مفاهیم اصلی برای روش شیءگرا عبارتند از:

- کلاس
- عملیات‌های سرویس‌ها
- محصورسازی
- وراثت

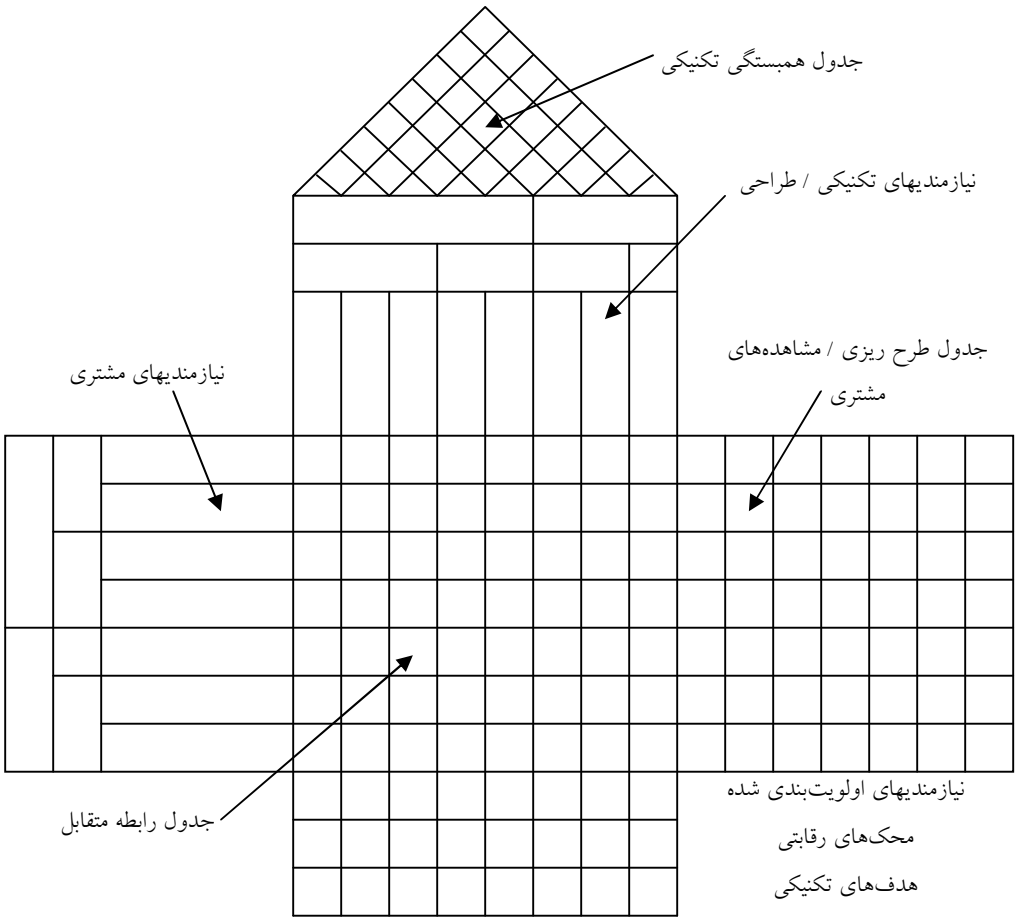


شکل ۳-۶: تصویری از مفاهیم شیءگرا

۳-۴-۴- استقرار تابع کیفیت[1] (QFD)

QFD به وسیله صنعت اتومبیل ژاپن توسعه داده شده است. هدف از آن طراحی خواسته‌های مشتری درون یک محصول بود قبل از اینکه آن محصول به وسیله تبدیل نیازمندیهای مشتری به مهندسی مشخصات ساخته شود. QFD شامل نشست‌های گروهی است که "House of Quality" در آن برای متمرکز کردن دیدگاه‌ها استفاده می‌شود. "House of Quality" (شکل ۵-۷ [۵۶]) دربرگیرنده یک ماتریس است که نشان دهنده ارتباط بین نیازمندیهای مشتری و ویژگیهای پیشنهاد شده است. سه گوش "roof" ماتریس "House of Quality" برای شناسائی این موضوع به کار می‌رود که نیازمندیهای تکنیکی که محصول را مشخص می‌کنند از نیازمندیهای دیگر حمایت یا ممانعت می‌کنند.

[1] Quality Function Deployment (QFD)



شکل ۳-۷ : House of Quality

۳-۵- مستند(سازی) نیازمندیها[1]

هدف از مستندسازی نیازمندیها، مرتبط کردن نیازمندیهای درک شده بین ذینفعان و توسعه‌دهندگان است. مستند نیازمندی، حوزه کاربرد[2] و سیستمی که توسعه می‌یابد را تشریح می‌کند. مستند نیازمندیها می‌تواند یک مبنا برای ارزیابی محصولات و فرآیندهای آتی (طراحی سیستم، موارد آزمایش، فعالیت وارسی و اعتبارسنجی[3]) و کنترل تغییرات در نظر گرفته شود. یک مستند نیازمندی خوب باید واضح / غیرمبهم، کامل، صحیح، قابل فهم، سازگار، فشرده و مختصر[4] و امکان‌پذیر باشد. مبتنی بر رابطه بین مشتری و تهیه کننده (سیستم یا نرم‌افزار)، مستند مشخصات می‌تواند بخشی از قرارداد در نظر گرفته شود [۵۱].

[1] Requirements documentation
[2] Application domain
[3] Verification and validation
[4] Concise

۳-۶- اعتبارسنجی نیازمندیها

هدف از اعتبارسنجی نیازمندیها تصدیق این موضوع است که نیازمندیها یک توضیح (شرح) قابل قبول از سیستم پیاده‌سازی شده را ارائه می‌کنند. ورودی فرآیند اعتبارسنجی مستند نیازمندیها، استانداردهای سازمانی و دانش سازمانی است. خروجی فرآیند اعتبارسنجی، لیستی است که دربرگیرنده مشکلات مرتبط با مستند نیازمندیها و کنش‌های مورد تفاهم است تا بتوان مشکلات گزارش شده را برطرف کرد. تکنیک‌های مورد استفاده برای اعتبارسنجی نیازمندیها، مرور نیازمندیها[1]، آزمایش نیازمندیها[2] و نمونه سازی است.

۳-۶-۱- مرور نیازمندیها

مرور نیازمندیها تضمین می‌کند که نیازمندیها :

- کامل هستند
- سازگار هستند
- غیرمبهم هستند
- می‌تواند آزمایش شود.

روشهای رسمی و غیررسمی برای مرور نیازمندیها وجود دارد. هدف از مرور غیررسمی نیازمندیها، پیدا کردن خطاها و بهبود بخشیدن به نیازمندیها است. مرور غیررسمی نیازمندیها به صورت تدریجی و در سرتاسر فرآیند مهندسی نیازمندیها با ذینفعان مربوط انجام می‌پذیرد. هدف از مرور رسمی نیازمندیها، تصدیق مستند نیازمندیها و صدور مجوز به پروژه است. بنابراین مرور رسمی زمانی می‌تواند انجام شود که مستند نیازمندیها کامل شده باشد [۵۷].

۳-۶-۲- آزمایش نیازمندیها

آزمایش نرم‌افزار یک بخش جدائی ناپذیر از مهندسی نرم‌افزار است. اگر نرم‌افزار مبتنی بر درک اشتباه نیازمندیها نوشته شده باشد نتایج مناسب و مورد انتظار حاصل نخواهد شد. به همین دلیل، نوشتن آزمایش برای نیازمندیها باید از همان مراحل اولیه -که کار بر روی نیازمندیها برای یک محصول شروع می‌شود- آغاز شود [۵۸]. آزمایش نوشته شده در طول تحلیل نیازمندیها می‌تواند برای اعتبارسنجی نیازمندیها در سیستم نهائی مورد استفاده قرار بگیرد. مزیت نوشتن آزمایش آن است که مشکلات مربوط به نیازمندیها اغلب قبل از طراحی و پیاده‌سازی کشف شوند [۵۳].

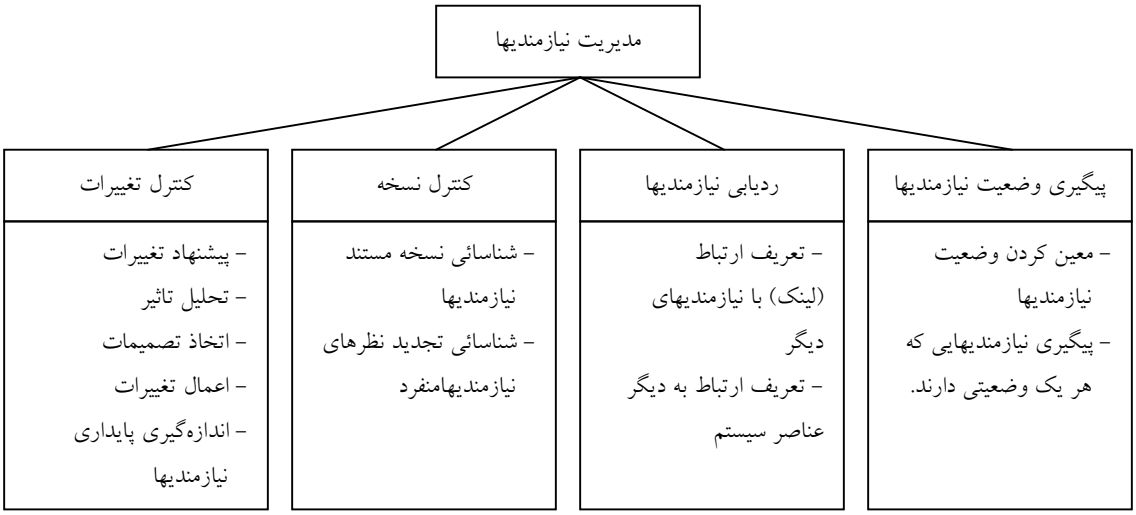
[1] Requirements review
[2] Requirements testing

۳-۶-۳- نمونه‌سازی

نمونه‌سازی دورانداختنی می‌تواند برای تعیین امکان پذیر بودن یک روش برای پیاده‌سازی سیستم بکار رود.

۳-۷- مدیریت نیازمندیها

مدیریت نیازمندیها شامل کلیه فعالیت‌های مرتبط با کنترل تغییرات، کنترل نسخه، ردیابی نیازمندیها[1] و پیگیری وضعیت نیازمندیها[2] است (شکل ۳-۸). هدف از مدیریت نیازمندیها دریافت[3]، ذخیره، پخش[4] و مدیریت اطلاعات است. ردیابی یک تکنیکی است که برای فراهم کردن یک ارتباط بین نیازمندیها، طراحی و پیاده‌سازی استفاده می‌شود تا اینکه بتوان تغییرات در یک سیستم را مدیریت کرد [۵۵].



شکل ۳-۸: فعالیت‌های مدیریت نیازمندیها

مدیریت نیازمندیها با شناسائی شروع می‌شود و سپس به هر نیازمندی یک شناسه‌ی منحصر به فرد داده می‌شود که می‌تواند به صورت <شماره‌ی نیازمندی> < نوع نیازمندی> باشد [۳۴].

نوع نیازمندی مقادیری دارد مانند:

F = نیازمندی تابعی، D = نیازمندی داده، B = نیازمندی رفتاری، I = نیازمندی ارتباطی، و P = نیازمندی خروجی.

[1] Requirements tracing
[2] Requirements status tracking
[3] Capture
[4] Disseminate

پس از مشخص شدن نیازمندیها، جداول ردیابی توسعه داده می‌شوند. این جداول که به صورت شماتیک در شکل ۳-۹ نشان داده شده‌اند هر یک، نیازمندیهای مشخص شده را با یک یا چند جنبه از سیستم یا محیط مرتبط می‌نماید. در میان بسیاری از جداول ردیابی موارد زیر نیز وجود دارند.

- جدول ردیابی جنبه‌ها. نشان می‌دهد که چگونه نیازمندیها با جنبه‌های مهم قابل مشاهده مشتری یا محصول مرتبط می‌شوند.
- جدوا ردیابی مبدا. مبدا هر نیازمندی را مشخص می‌کند.
- جدول ردیابی وابستگی. نشان می‌دهد چگونه نیازمندیها به یکدیگر مرتبط می‌شوند.
- جدول ردیابی زیر سیستم. نیازمندیها را بر اساس زیر سیستم‌های مرتبط طبقه‌بندی می‌کند.
- جدول ردیابی رابط. نشان می‌دهد که چگونه نیازمندیها به رابط‌های داخلی و خارجی سیستم مرتبط می‌شوند.

جدول ۳-۱: جدول ردیابی عمومی

نیازمندیها \ جنبه خاصی از سیستم یا محیط	A01	A02	A03	A04	A05			Aii
R01			√					
R02	√		√					
R03	√			√				√
R04		√			√			
R05	√	√		√				√
Rnn	√		√					

در بسیاری از موارد، این جدول ردیابی به عنوان بخشی از بانک اطلاعاتی نیازمندیها نگهداری می‌شوند به گونه‌ای که به سرعت قابل جستجو باشند تا این امکان فراهم شود که مشخص شود چگونه یک تغییر در یک نیازمندی بر جنبه‌های متفاوت سیستم در حال ساخت تاثیر می‌گذارد.

۳-۸- خلاصه و نتیجه‌گیری

در این فصل به بررسی مهندسی نیازمندیها پرداختیم. در این راستا ابتدا مهندسی نیازمندیها را تعریف کردیم و گفته شد که هدف از فاز مهندسی نیازمندیها در چرخه تولید نرم‌افزار چیست. سپس فعالیت‌های مهندسی نیازمندیها را ذکر کردیم و گفتیم که هدف از هر فعالیت چیست. ورودی و خروجی هر فعالیت چیست. چه کارهای باید در هر فعالیت صورت بپذیرد. همچنین تکنیک‌های مورد استفاده برای هر فعالیت را نام بردیم و گفتیم که تمرکز هر تکنیک بر روی چه موضوعاتی است. چگونه انجام می‌گیرد و برای چه مسائلی باید از آنها استفاده کرد.

با توجه به موضوعات ذکر شده پیرامون مهندسی نیازمندیها، این نتیجه حاصل می‌شود که یکی از مهمترین فاکتورها در موفقیت آمیز بودن توسعه یک سیستم نرم‌افزاری، درست و کامل اجرا شدن مهندسی نیازمندیها است. به منظور رسیدن به یک مستند نیازمندیها خوب، کامل، غیر مبهم و سازگار باید یک سری از فعالیت‌ها انجام شود که تکنیک‌های مورد استفاده در هر یک از این فعالیت‌ها می‌تواند بر حسب نوع و وسعت پروژه تغییر کند. مهندسی نیازمندیها که هدفش شناسائی نیازمندیهای کاربران و معتبرسازی آنها و فراهم کردن توافق بر روی نیازمندیها سیستم در حال توسعه است فقط یک فاز نیست که در آغاز فرآیند توسعه سیستم کامل شود و دیگر نیازی به آن نباشد. بلکه فرآورده‌های آن تا آخرین مراحل توسعه یک سیستم مورد استفاده قرار می‌گیرند و چه بسا در بخشی از فرآیند توسعه یک سیستم نیاز باشد که دوباره فاز مهندسی نیازمندیها انجام شود.

فصل چهارم

بررسی روشهای مهندسی نیازمندیهای جنبه‌گرا

در این فصل، هدف بررسی روشهایی است که برای مهندسی نیازمندیهای جنبه‌گرا ارائه شده‌اند. زیرا هر کدام از روشها برای خود مدل، ساختار، مزایا، معایب و کمبودهایی به همراه دارند که درک و فهم آنها می‌تواند نقش مهمی در این تحقیق ایفا کند. بدین منظور ابتدا روشهای مهندسی نیازمندیهای جنبه‌گرا همراه با چگونگی عملکرد و اهدافشان ذکر می‌شود سپس معیارهایی که در مهندسی نیازمندیها وجود دارد و یا معیارهایی که در بعضی از این روشها ذکر شده‌اند را مشخص کرده و مقایسه‌ای بین روشهایی ذکر شده پیرامون این معیارها ارائه می‌شود.

۴-۱- مقدمه

بعد از معرفی شدن مهندسی نیازمندیهای جنبه‌گرا در سال ۱۹۹۹ در مقاله آقای گراندی [۴]، کارهای زیادی در این حوزه انجام شده است که نقطه شروع آنها مقاله‌ای بود که در کنفرانس مهندسی نیازمندیهای سال ۲۰۰۲ ارائه شد [۳]. این مقاله یک مدل عمومی برای مهندسی نیازمندیهای جنبه‌گرا پیشنهاد داد و باعث پدیدار شدن یک دید اولیه برای مهندسی نیازمندیهای جنبه‌گرا شد. این مقاله همچنین دو ویژگی اولیه کلیدی نیز برای مهندسی نیازمندیهای جنبه‌گرا تعریف کرد. این دو ویژگی کلیدی عبارتند از :

۱. فراهم کردن پشتیبانی بهبود یافته برای جداسازی خصوصیات وظیفه‌مندی و غیروظیفه‌مندی مداخله‌ای در روند مهندسی نیازمندیها و لذا ارائه قابلیت بهتر برای شناسائی و مدیریت تداخل‌های ناشی از نمایش‌های درهم‌تنیده.
۲. مشخص کردن تاثیر و نگاشت جنبه‌های سطح نیازمندیها بر روی فرآورده‌های گامهای بعدی توسعه و لذا بنا نهادن مصالحه‌های[1] حیاتی قبل از حاصل شدن معماری.

با مشخص شدن مزایایی که مهندسی نیازمندیهای جنبه‌گرا می‌تواند فراهم کند کارهایی در زمینه ارائه روشهایی جدید و کامل برای آن انجام شد که تعدادی از آنها علاوه بر دو خصوصیت کلیدی ذکر شده، مزایایی زیادی را نیز معرفی کردند که بعضی از این روشها در همین فصل تشریح خواهند شد.

۴-۲- مدل عمومی مهندسی نیازمندیهای جنبه‌گرا

مدل عمومی که برای سروکار داشتن با دغدغه‌های مداخله‌ای در فاز مهندسی نیازمندیهای جنبه‌گرا معرفی می‌شود و در شکل ٤-١ نیز به تصویر کشیده شده است متشکل از شش فعالیت است [۳].

با توجه به مشخص بودن ترتیب اجرا فعالیت‌ها در شکل ٤-١ عملیات مربوط به هر فعالیت به شرح زیر است:

فعالیت شناسائی دغدغه‌ها[2] :

در این فعالیت، دغدغه‌های ذینفعان به وسیله تحلیل نیازمندیهای اولیه بدست می‌آیند.

فعالیت شناسائی دیدگاه‌ها، کشف نیازمندیها و مرتبط ساختن آنها به دغدغه‌ها :

در این فعالیت، نیازمندیهای ذینفعان کشف شده و به دغدغه‌های شناسائی شده در فعالیت "شناسائی دغدغه‌ها" مرتبط می‌شوند. در این فعالیت، شناسائی دیدگاه‌ها همان شناسائی نیازمندیها است و دلیل ذکر آن،

[1] Trade offs
[2] Identify concerns

مبتنی بودن مدل بر پایه دیدگاه‌ها است. البته ذکر آن به معنی وابستگی داشتن این مدل به دیدگاه‌ها نیست زیرا می‌توان این مدل را بر پایه دیگر روشها، مانند موارد کاربری نیز در نظر گرفت.

اینکه ابتدا این فعالیت باید انجام شود یا فعالیت "شناسائی دغدغه‌ها"، به پویای فعل و انفعالات (تعاملات) بین مهندسین نیازمندیها و ذینفعان وابسته است. همچنین در هر دو حالت باید عمل مرتبط کردن نیازمندیها به دغدغه‌ها صورت بپذیرد.

فعالیت مشخص کردن دغدغه‌ها[1]:

در این فعالیت، جزئیات دغدغه‌های شناسائی شده مشخص می‌شوند.

فعالیت شناسائی جنبه‌های نامزد:

در این فعالیت، جنبه‌های نامزد تعیین می‌شوند. اگر یک دغدغه چندین نیازمندی را میان‌بر کند (یعنی اگر یک دغدغه بیش از یک دیدگاه را محدود و یا تحت تاثیر قرار دهد) آن دغدغه به عنوان یک جنبه نامزد در نظر گرفته می‌شود.

فعالیت مشخص و اولویت‌دهی کردن جنبه‌های نامزد:

در این فعالیت اولین عمل، مشخص کردن جزئیات جنبه‌های نامزد است. این عمل باعث می‌شود که جنبه‌ها بهبودیافته[2] و معتبرتر[3] شوند و همچنین تعاملات و تداخل‌های بین آنها شناسائی شوند [۵۹]. دومین عمل، اولویت‌دهی جنبه‌های نامزد است که باعث حل تداخل‌های ما بین آنها می‌شود.

فعالیت مشخص کردن ابعاد جنبه[4]:

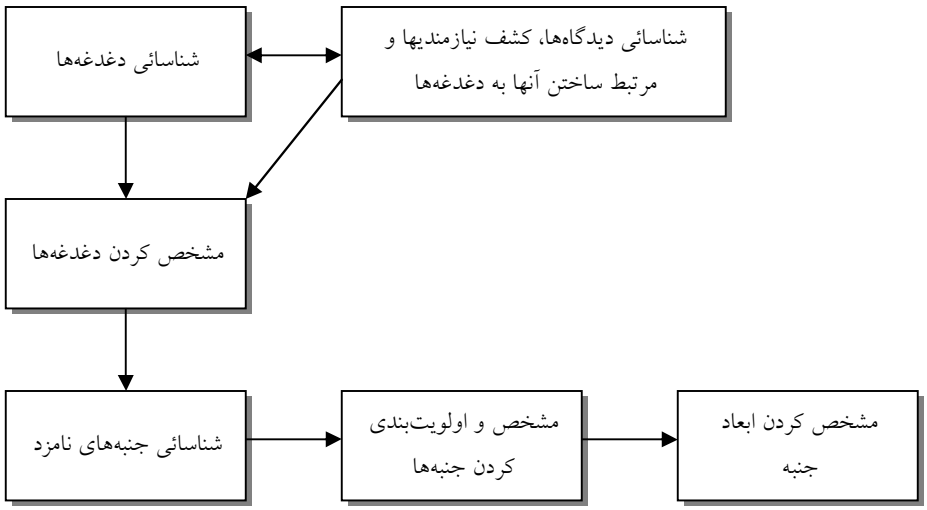
در این فعالیت، ابعاد یک جنبه مشخص می‌شود. زیرا جنبه‌ها در مراحل اولیه توسعه (مهندسی نیازمندیها) می‌توانند تاثیراتی در مراحل بعدی داشته باشند. این تاثیرات در دو بعد دسته بندی می‌شوند.

- نگاشت[5]: یک جنبه می‌تواند به یک ویژگی یا تابع (مانند متد ساده)، تصمیم (مانند یک تصمیم برای انتخاب معماری)، طراحی (و پیاده‌سازی) و جنبه (مانند زمان پاسخ) در سیستم نگاشت شود. این بعد، دلیل اصلی این موضوع است که چرا در مدل، جنبه‌ها در مرحله مهندسی نیازمندیها جنبه نامزد نامیده شده‌اند. زیرا با وجود ماهیت مداخله‌ای آنها در این مرحله ممکن است مستقیما به یک جنبه در مراحل بعدی نگاشت نشوند.

[1] Specify concerns
[2] Refine
[3] Concrete
[4] Specify aspect dimensions
[5] Mapping

- تاثیر[1] : یک جنبه ممکن است چندین نقطه در چرخه توسعه را تحت تاثیر قرار دهد. به عنوان مثال قابلیت دسترسی، معماری سیستم را تحت تاثیر قرار می‌دهد حال آنکه زمان پاسخ، هم معماری و هم جزئیات طراحی را تحت تاثیر خود قرار می‌دهد.

اما دلیل اینکه چرا باید اولویت‌دهی جنبه‌ها قبل از مشخص کردن بعدها انجام شود این است که تفکیک تداخل[2] می‌تواند در هنگام مشخص کردن ابعاد، مورد استفاده واقع شود.

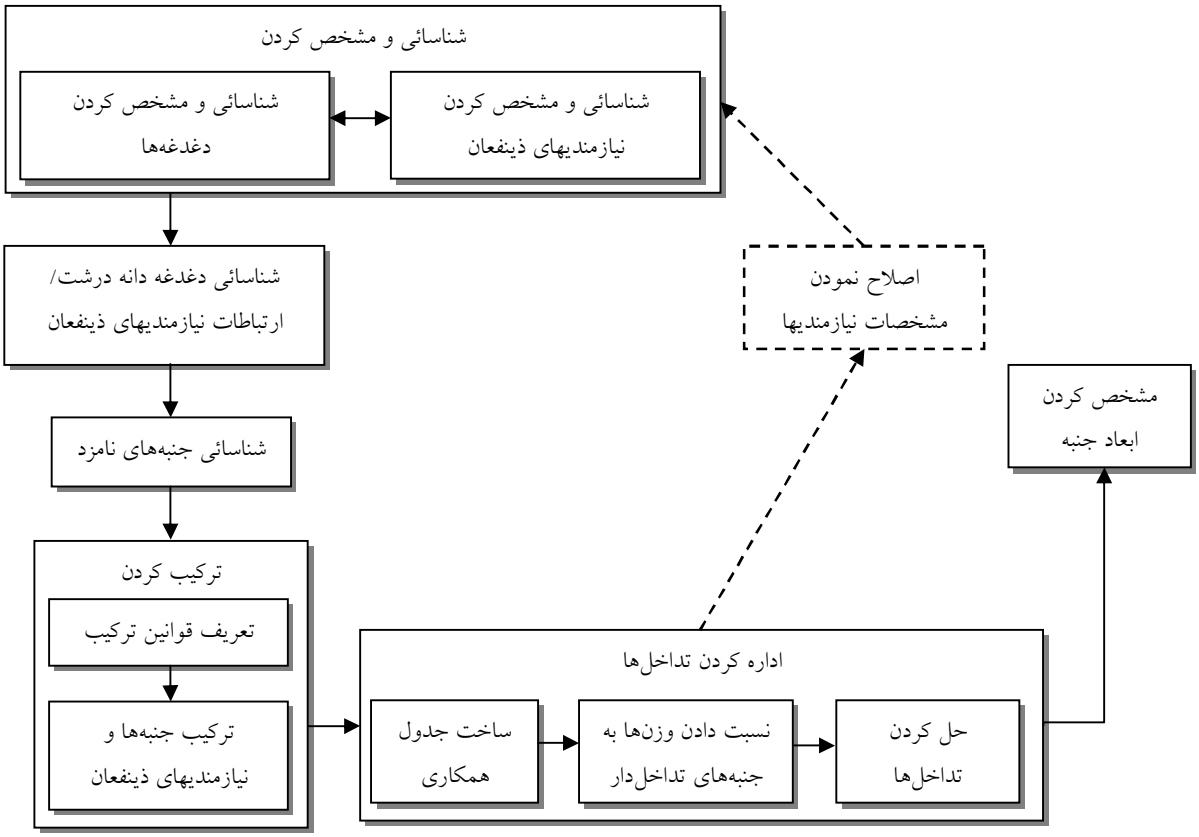


شکل ۴-۱: مدل عمومی برای مهندسی نیازمندیهای جنبه‌گرا

۴-۳- مدل بهبود یافته "مدل عمومی مهندسی نیازمندیهای جنبه گرا"

این مدل، یک مدل بهبود یافته از مدل تشریح شده در بخش ٤-۲ است که مبتنی بر یک ابزاری به نام ARCaDe[3] است. به همین دلیل از این پس با نام "مدل مهندسی نیازمندیهای جنبه‌گرا با ARCaDe" از این مدل یاد خواهیم کرد.

[1] Influence
[2] Conflict resolution
[3] Aspectual Requirements Composition and Decision Support Tool



شکل ۴-۲: مدل مهندسی نیازمندیهای جنبه‌گرا با ARCaDe

"مدل مهندسی نیازمندیهای جنبه‌گرا با ARCaDe" [۶۰] برای بهبود "مدل عمومی مهندسی نیازمندیهای جنبه‌گرا" معرفی شد زیرا "مدل عمومی مهندسی نیازمندیهای جنبه‌گرا" ارتباطات ترکیب[1] را در داخل تعاریف جنبه نگهداری می‌کرد و باعث می‌شد دانه درشتی[2] در PREView [۳۷] وجود داشته باشد. در نتیجه فقط امکان‌پذیر بود که مشاهد شود چگونه یک جنبه یک مجموعه از دیدگاه‌ها را تحت تاثیر قرار می‌دهد و امکان نداشت تاثیر یک نیازمندی جنبه‌ای[3] و محدودیت‌های که بر روی نیازمندیهای خاص درون دیدگاه‌های تحت تاثیر قرار گرفته به وسیله جنبه وجود داشت، تعیین شود. بعلاوه "مدل عمومی مهندسی نیازمندیهای جنبه‌گرا" پشتیبانی صریح برای مذاکره[4] نیازمندیها مبتنی بر مصالحه جنبه‌ای و اصلاح مشخصات را ندارد.

این مدل (شکل ۴-۲) با شناسائی و مشخص کردن دغدغه‌ها و نیازمندیهای ذینفعان آغاز می‌شود. شناسائی و مشخص کردن نیازمندیهای ذینفعان با استفاده از یکی از مکانیزمهای جداسازی دغدغه‌های موجود در سطح

[1] Composition
[2] Coarse-grained
[3] Aspectual requirements
[4] Negotiating

نیازمندیها از قبیل دیدگاه‌ها [۳۶]، موارد کاربری [۶۱]، هدفها[1] [۶۲] یا قابهای مشکل[2] [۶۳] انجام می‌شود. شناسائی و مشخص کردن دغدغه‌ها با تحلیل نیازمندیهای اولیه حاصل می‌شود. تصمیم در مورد اینکه ابتدا مشخص کردن دغدغه‌ها یا مشخص کردن نیازمندیهای ذینفعان انجام شود به پویائی فعل و انفعالات بین مهندسین نیازمندیها و ذینفعان وابسته است. اما در هر حالت (شروع با مشخص کردن دغدغه‌ها یا شروع با مشخص کردن نیازمندیهای ذینفعان) خیلی سودمند است که دغدغه‌ها از طریق یک جدول- به طوری که دغدغه‌ها بتواند نیازمندیها را محدود کنند- به نیازمندیها مرتبط شوند. این جدول می‌تواند به صورت جدول ۴-۱ در نظر گرفته شود.

جدول ٤-١: مرتبط کردن دغدغه‌ها به نیازمندیهای ذینفعان

نیازمندیهای ذینفعان / دغدغه‌ها	نیازمندیهای ذینفعان ۱	نیازمندیهای ذینفعان ۲	...	نیازمندیهای ذینفعان n
دغدغه ۱				*
دغدغه ۲		*		*
...				
دغدغه n	*	*		

با توجه به جدول ۴-۱ مشخص است که دغدغه‌ها، ماژولهایی که نیازمندیهای ذینفعان را محصور کرده‌اند میان‌بر می‌کنند و بنابراین می‌توانند به عنوان جنبه‌های نامزد در نظر گرفته شوند.

وقتی که که ارتباطات دانه درشت بین دغدغه‌ها و نیازمندیها محرز شدند و جنبه‌های نامزد شناسائی شدند مرحله بعدی تعریف جزئیات قوانین ترکیب است. قوانین ترکیب، ارتباطات بین نیازمندیهای جنبه‌ای و نیازمندیهای غیر جنبه‌ای را در دانه‌بندی ریز تعریف می‌کنند. این قوانین در دانه‌بندی از نیازمندیهای منفرد- و نه فقط ماژولهای محصورکننده آنها- عمل می‌کنند. در نتیجه امکان پذیر است که چگونگی تحت تاثیر قرار گرفتن و محدود شدن رفتارهای یک مجموعه از نیازمندیهای غیرجنبه‌ای در ماژولهای مختلف توسط یک نیازمندی جنبه‌ای مشخص شود. همزمان با همین عمل اگر مطلوب باشد مصالحه جنبه‌ای می‌تواند در دانه‌بندی ریز مشاهده شود. این عمل نیاز برای مذاکره غیرضروری بین ذینفعان، در حالاتی که می‌تواند مصالحه آشکاری بین دو (یا بیشتر) جنبه وجود داشته باشد را کم می‌کند اما در حقیقت نیازمندیهای مجزا و مختلف، تحت تاثیر آنها (جنبه‌ها) قرار می‌گیرند. این عمل همچنین، شناسائی نیازمندیهای جنبه‌ای تداخلدار و منفرد را با رجوع به اینکه

[1] Goals
[2] Problem frames

کدام مذاکره و مصالحه باید اجرا شده و تحقق یابد، آسان می‌کند. بعد از ترکیب کردن جنبه‌های نامزد و نیازمندیهای ذینفعان با استفاده از قوانین ترکیب، شناسائی و تفکیک تداخل‌های ما بین جنبه‌های نامزد آغاز می‌شود. این عملیات به صورت زیر انجام می‌پذیرند.

۱. ایجاد یک جدول همکاری[1] (جدول ۴-۲)، که در آن هر جنبه می‌تواند به صورت منفی (-) یا مثبت (+) با دیگر جنبه‌ها همکاری داشته باشد. در واقع در این همکاری تاثیر مثبت و منفی یک جنبه بر دیگر جنبه‌ها تعیین می‌شود (خانه خالی جدول نشان دهنده تاثیر خنثی است). به عنوان مثال امنیت تاثیر منفی بر زمان پاسخ و قابلیت دسترسی تاثیر مثبت بر زمان پاسخ دارد.

۲. تخصیص وزنها به جنبه‌هایی که به صورت منفی با دیگر جنبه‌ها در ارتباط با یک مجموعه از نیازمندیهای ذینفعان، همکاری دارند. یعنی تاثیر منفی بر آنها دارند. هر وزن یک عدد اشاری بین بازه [۱ .. ۰] است. این اعداد اولویت یک جنبه در ارتباط با یک مجموعه از نیازمندیهای ذینفعان را نشان می‌دهند. در واقع در اینجا از منطق فازی برای مشخص کردن اهمیت جنبه‌ها به صورت زیر استفاده می‌شود: [۱.. ۰/۸] : خیلی مهم

[۰/۸ .. ۰/۵] : مهم

[۰/۵ .. ۰/۳] : متوسط

[۰/۳ .. ۰/۱] : نه خیلی مهم

[۰/۱ .. ۰] : بی اهمیت

۳. حل تداخل‌ها با کمک ذینفعان، استفاده از روش اولویت دهی ذکر شده در بالا برای کمک کردن به ارتباطات

جدول ٤-۲: همکاریها بین جنبه‌های نامزد

جنبه‌ها / جنبه‌ها	جنبه ۱	جنبه ۲	...	جنبه n
جنبه ۱		+		
جنبه ۲				-
...				
جنبه n				

[1] Contribution

تفکیک تداخل ممکن است منجر به اصلاح در مشخصات نیازمندیها (نیازمندیهای ذینفعان، نیازمندیهای جنبه‌ای یا قوانین ترکیب) شود. اگر این موضوع اتفاق بیفتد بعد از آن دوباره نیازمندیها ترکیب شده و مجددا تداخل‌های اضافی که ممکن است پیش بیاید حل می‌شوند. این چرخه تا زمانی که تمام تداخل‌ها از طریق مذاکرات موثر حل شوند تکرار می‌شود. آخرین فعالیت در مدل، مشخص کردن ابعاد یک جنبه است. تخصیص بعدها به جنبه‌ها می‌تواند به صورت جدول ۴-۳ در نظر گرفته شود.

جدول ٤-۳: مشخصات ابعاد جنبه‌ها

جنبه نامزد	تاثیر	نگاشت
جنبه نامزد ۱		
جنبه نامزد ۲		
...		
جنبه نامزد n		

همان طور که در بخش ۴-۲ در تشریح "فعالیت مشخص کردن ابعاد جنبه" بیان شد هر جنبه دو بعد دارد که هر بعد می‌تواند انواع مختلفی داشته باشد این نوعها به صورت مختصر در جدول ۴-۴ نمایش داده شده است. در سلولهای زیرین ستون تاثیر و نگاشت در جدول ۴-۳ مقادیر جدول ۴-۴ قرار خواهند گرفت. در هر سلول زیر ستون نگاشت، فقط یک نوع از مقادیر نگاشت در جدول ۴-۴ می‌تواند قرار داده شود ولی در سلول زیر ستون تاثیر، چندین نوع از مقادیر تاثیر در جدول ۴-۴ می‌توانند به طور همزمان نوشته شوند. برای بررسی بیشتر یک مطالعه موردی که دربرگیرنده تمام مراحل و قوانین ترکیب است در [۶۰] ذکر شده است. در این بخش از ذکر ساختار استفاده شده در قوانین ترکیب خودداری می‌شود زیرا حالت کامل شده آن در بخش ۳-۶ تشریح خواهد شد.

جدول ٤-٤: ابعاد یک جنبه و انواع هر بعد

نگاشت	تابع، جنبه، تصمیم
تاثیر	مشخصات، معماری، طراحی، توسعه

۴-۴- مدل COSMOS [1]

cosmos یک طرح مدلسازی فضای دغدغه چند منظوره است [۵] که نیازمندیهای زیر را آدرس‌دهی می‌کند:

۱. حمایت از نمایش دغدغه‌های دلخواه
۲. حمایت از نمایش دغدغه‌های مرکب (از قبیل دغدغه‌های پیشامدی[2] مبتنی بر فعل و انفعالات دغدغه‌های پایه)
۳. حمایت از نمایش ارتباطات دلخواه ما بین دغدغه‌ها
۴. حمایت از نمایش مشارکت دغدغه‌ها با واحدهای نرم‌افزاری، محصولات کاری[3]، یا عناصر سیستم دلخواه
۵. مستقل بودن از زبان، یعنی اینکه:
   I. هیچ وابستگی به هر نوع زبان برنامه‌نویسی خاص یا دیگر رویه‌های توسعه نداشته باشد.
   II. رویه‌های توسعه مختلف را به مراحل مناسب مختلف چرخه حیات منطق کند.
   III. قادر به دریافت اطلاعاتی باشد که ضرورتا در رویه‌های توسعه خاص، بازتاب نمی‌شود.
۶. مستقل بودن از متدولوژی
۷. قابل بکارگیری در سرتاسر چرخه حیات نرم‌افزار
۸. حمایت از انواع مختلف وظایف مهندسی نرم‌افزار (یعنی مناسب بودن برای روشهای توسعه‌ای که در آن مورد استفاده قرار گرفته است).

چهارتای اولی، نیاز برای مدلسازی دغدغه‌ها و ارتباطاتشان را آدرس‌دهی می‌کنند. پنج تا از آخر نیز، نیاز برای استفاده از طرحها و مدلهای خاص را آدرس‌دهی می‌کنند. روش cosmos شامل سه نوع عنصر به نامهای دغدغه‌ها، ارتباطات و مسندها[4] است. که هر یک در ادامه تشریح خواهند شد. همچنین cosmos بین موارد زیر وجه تمایز قائل می‌شود.

- طرح هسته[5]
- گسترش به طرح هسته، که می‌تواند به نصب یا کاربردهای خاص طرح اعمال شود.

[1] Concern-Space Modeling Schema
[2] Emergent
[3] Work products
[4] Predicates
[5] The Core Schema

- مدلهای خاص[1]، که نمونه‌هایی از یک طرح هستند. یعنی اینکه یک طرح بعلاوه دغدغه تشریح کننده داده برای سیستم نرم‌افزاری خاص یا محیط توسعه

۴-۴-۱- دغدغه‌ها

دغدغه‌ها به دو دسته وسیع منطقی و فیزیکی طبقه‌بندی می‌شوند. دغدغه‌های منطقی، مفاهیمی را که در یک سیستم یا محصول در نظر گرفته می‌شوند را تداعی می‌کنند. به عنوان مثال پیامدها[2]، جنبه‌ها، ویژگیها، خصوصیات و .. نمونه‌هایی از دغدغه‌های منطقی هستند. دغدغه‌های فیزیکی، عناصر سیستم یا محصولات نرم‌افزاری را که با آنها سروکار داریم (با آنها کار می‌کنیم) یا دغدغه‌های منطقی به آنها اعمال می‌شوند را تداعی می‌کنند. دلیل تمایز قائل شدن بین دغدغه‌ها و دسته‌بندی کردن آنها در دو گروه این است که دغدغه‌های منطقی می‌توانند مستقل از دغدغه‌های فیزیکی در نظر گرفته شوند و دغدغه‌های فیزیکی نیز می‌توانند مستقل از دغدغه‌های منطقی مدلسازی شوند. اما چرا هر دو گروه از دغدغه‌ها مد نظر است؟ زیرا دغدغه‌های منطقی علایق توسعه دهنده در یک سیستم یا محصول را تحریک می‌کنند در حالی که دغدغه‌های فیزیکی این اجازه را می‌دهند تا سیستم یا محصول در جهت تحقق دغدغه‌های منطقی مدلسازی شود.

۴-۴-۱-۱- دغدغه‌های منطقی

دغدغه‌های منطقی موضوعات مورد علاقه در توسعه سیستم را نمایش می‌دهند. مثالهایی از این نوع دست دربرگیرنده عملکردی[3]، رفتار، کارائی، استحکام[4]، وضعیت، وابستگی، قابلیت پیکربندی، قابلیت استفاده، اندازه، هزینه و ... است. cosmos دامنه دغدغه‌های منطقی را محدود نمی‌کند و آن را برای توسعه‌دهندگان آزاد می‌گذارد تا توسعه‌دهندگان (یا دیگر ذینفعان) دغدغه‌های مورد علاقه خود را شناسائی کنند.

cosmos دغدغه‌های منطقی را از یکدیگر مجزا کرده و آنها را در پنج گروه دسته‌بندی می‌کند که عبارتند از : طبقه‌بندی‌ها[5]، کلاسها، نمونه‌ها[6]، خصوصیات و عنوانها[7].

طبقه‌بندی‌ها، سیستم‌هایی از کلاسهای مستقل را نمایش می‌دهند. آنها به ابعاد سطح بالا در فضای دغدغه مربوط هستند. زیرا فضاهای دغدغه چند بعدی هستند. یک نمونه خاص می‌تواند همزمان مطابق با چندین طبقه‌بندی

[1] Particular Models
[2] Issues
[3] functionality
[4] Robustness
[5] Classifications
[6] Instances
[7] Topics

دسته‌بندی شود. برای مثال یک واحد کد می‌تواند مطابق با یک زبان برنامه‌نویسی، کلاسهای نمایش داده شده، عملیات موجود، معیارهای پیچیدگی و غیره دسته‌بندی شود.

کلاسها، دغدغه‌هایی هستند که برای دسته‌بندی دغدغه‌های دیگر معرفی شده‌اند. کلاسها، متعلق به یک طبقه‌بندی هستند و می‌تواند کلاسهای دیگر (زیر کلاسها) و نمونه‌ها (منطقی و فیزیکی) را شامل شوند. مثالهایی از دغدغه‌های منطقی که از نوع کلاسها هستند عبارتند از: هدفها، مسئولیت‌ها، عملکردih، قابلیت پیکربندی (به عنوان یک دسته از انواع قابلیت پیکربندی) و مستندات طراحی.

نمونه‌ها، دغدغه‌های ویژه‌ای هستند که دغدغه‌های دیگر را دسته‌بندی یا مشخص نمی‌کنند. مثالهایی از این نوع عبارتند از: عملکردی خاص از قبیل اضافه کردن شیء، رفتارهای خاص از قبیل اعتبارسنجی ورودی، عناصر ویژه از قابلیت پیکربندی از قبیل پیکربندی حافظه میانجی[1]

خصوصیات، دغدغه‌هایی هستند که دیگر دغدغه‌ها (منطقی) مخصوصا کلاسها و نمونه‌ها را مشخص می‌کنند. مثالهایی از خصوصیات عبارتند از : عمومیت[2]، قابلیت پیکربندی (به عنوان یک خصوصیت سیستم)، کارائی، تاثیرپذیری[3]، اندازه، پیچیدگی، سهولیت تطبیق، قابلیت گسترش و ... .

عنوانها، اجتماعی از دغدغه‌های دلخواه هستند. عنوانها، دغدغه‌های مرتبط شده به موضوع[4] که می‌توانند متعلق به چندین طبقه‌بندی باشند را تسخیر می‌کنند (در بر می‌گیرند). برای مثال، عنوان قابلیت پیکربندی می‌تواند شامل کلاسها مرتبط به پیکربندی (از قبیل توابع پیکربندی و رفتارهای پیکربندی)، نمونه‌هایی از این کلاسها و خصوصیات مرتبط شده به پیکربندی باشد. بنابراین عنوانها روشی برای سازماندهی گروه‌هایی از دغدغه‌ها را فراهم می‌کنند که دیگر طبقه‌بندی‌ها را میان‌بر می‌کنند.

۴-۴-۱-۲- دغدغه‌های فیزیکی

دغدغه‌های فیزیکی عناصر دنیای واقعی یک سیستم به خصوص شامل نرم‌افزار، سخت‌افزار، سیستم‌ها و سرویس‌ها را نمایش می‌دهند. این دغدغه‌ها می‌توانند دغدغه‌های منطقی را نمایش، پیاده‌سازی، حمایت یا در غیر این صورت تحت تاثیر قرار دهند. cosmos سه دسته از دغدغه‌های فیزیکی را از یکدیگر تفکیک می‌کند این سه دسته عبارتند از: نمونه‌ها، مجموعه‌ها، صفت‌ها.

نمونه‌ها، عناصر خاص از سیستمهای نرم‌افزاری شامل محصولات کاری، عناصر سخت‌افزاری، سیستم‌ها و سرویس‌های ویژه هستند.

---

[1] Buffer
[2] Generality
[3] Responsiveness
[4] Theme-related

مجموعه‌ها، گروهی از دغدغه‌های فیزیکی هستند. مثالهایی از آنها عبارتند از: بسته‌هایی از فایلهای منبع و ایستگاه‌های کاری بر روی یک شبکه محلی. مجموعه‌ها می‌توانند شامل زیرمجموعه‌ها باشند. زیر مجموعه‌ها با توجه به نمونه‌های فیزیکی که در خود دارند می‌تواند همگن یا ناهمگن در نظر گرفته شوند.

صفت‌ها، مقادیر ویژه‌ای هستند که نمونه‌های فیزیکی و مجموعه‌ها را مشخص می‌کنند. معمولا صفت‌ها خصوصیت‌های جالب در میان دغدغه‌های منطقی را بازتاب می‌کنند.

۴-۴-۲- ارتباطات

cosmos چهار دسته از ارتباطات را به نامهای قطعی[1]، تفسیری[2]، نگاشت و فیزیکی تعریف می‌کند. چنانچه که در ادامه تشریح خواهد شد. طرح هسته cosmos انواع خاصی از رابطه قطعی را تعریف می‌کند در حالی که دیگر نوعها از دسته دیگر باید مطابق با نصب[3] و کاربردهایشان تعریف شوند.

جدول ٤-٥: نمای کلی از طرح مدلسازی فضای دغدغه cosmos

هسته			گسترشها
دغدغه‌ها	**منطقی**	طبقه‌بندی‌ها، کلاسها، نمونه‌ها، خصوصیات، عنوانها	
	فیزیکی	مجموعه‌ها، نمونه‌ها، صفت‌ها	
ارتباطات	**قطعی**	کلاسی از، زیرکلاسی از، نمونه‌ای از، خصوصیتی از، موضوعی از، عضوی از، صفتی از	
	تفسیری	مهم برای	پذیرش، همکاری با، پیاده‌سازی منطقی، محرکها، منطقا بخشی از
	نگاشت	نگاشت به	تاثیر واقع شده به وسیله‌ی، تشریح‌ها، مدلها، پیاده‌سازی فیزیکی به وسیله‌ی، نمایشها
	فیزیکی	فیزیکی مرتبط به	اتصالات، متصل به، تاثیرات فیزیکی، بخش فیزیکی از
تفسیرها	**نوع خاصی تعریف نشده است**		

[1] Categorical
[2] Interpretive
[3] Installation

۴-۴-۲-۱- قطعی

ارتباطات قطعی، دغدغه‌ها را مبتنی بر دسته‌های خود مرتبط می‌سازند (به جدول ٤-٥ توجه شود). این ارتباطات، معنای انواع دغدغه‌ها را منعکس کرده و صحت مدلهای دغدغه را تعریف می‌کنند. چندین نوع از ارتباطات قطعی وجود دارد که به شرح زیر هستند:

طبقه‌بندی، کلاسها را به یک سیستم از طبقه‌بندی‌ها مرتبط می‌کند. *"به وسیله رفتار"* یک طبقه‌بندی با کلاس ریشه *"رفتار"* است (طبقه‌بندی‌ها اغلب با کلاسهای ریشه خود نامگذاری می‌شوند).

تعمیم، کلاسها را در یک رابطه ارث‌بری به یکدیگر مرتبط می‌کند. برای مثال، *"رفتار واقعه‌نگاری"* یک زیر کلاس از *"رفتار"* است.

نمونه‌سازی[1]، مرتبط می‌کند نمونه‌ها را به کلاسهایی که به آنها تعلق دارند. نمونه‌ها می‌توانند منطقی یا فیزیکی باشند. اما یک کلاس معین فقط می‌تواند منحصرا به یکی تعلق داشته باشد. برای مثال *"اضافه کردن شیء"* یک نمونه از *"عملکردی هسته"* است.

توصیف صفات، خصوصیات را به کلاسها یا نمونه‌های آنها مرتبط می‌کند. برای مثال *"کارائی"* یک خصوصیت از *"عملکردی"* است.

عضویت، نمونه‌های فیزیکی را به مجموعه‌های که به آنها تعلق دارند مرتبط می‌کند. برای مثال *"cache_core.java"* یک عضو از بسته *"com.ibm.ws.abr.gps"* است

تخصیص[2]، صفات را به نمونه‌های فیزیکی یا مجموعه‌ها مرتبط می‌کند. برای مثال *"اندازه"* یک صفت از *"cache_core.java"* است.

موضوع مورد بحث[3]، دغدغه‌ها از هر نوع را به عنوان مرتبط می‌کند. به عنوان مثال قابلیت پیکربندی زباله[4] به عنوان یک خصوصیت و الگوریتم جمع‌آوری زباله به عنوان یک کلاس، هر دو در عنوان جمع‌آوری زباله هستند.

۴-۴-۲-۲- تفسیری

ارتباطات تفسیری، معنی تفسیری وابستگی بین دغدغه‌های منطقی را منعکس می‌کنند. آنها اصولا بر تفسیر وابسته به متن[5]، از مفاهیم و معانی دغدغه وابسته هستند. طرح هسته cosmos انواع ارتباطات تفسیری خاصی را

[1] Instantiation
[2] Attribution
[3] Topicality
[4] Garbage-collection
[5] Context-dependent

از پیش تعریف نمی‌کند. این ارتباطات باید مطابق با نیازهای مدلسازی دغدغه اضافه شوند. به عنوان یک پیشنهاد می‌توان چهار نوع: همکاری، محرک[1]، پیاده‌سازی منطقی و پذیرش[2] را در نظر گرفت ولی هر حوزه می‌تواند تقسیم‌بندی خاص خود را داشته باشد.

همکاری: یک دغدغه منطقی با دغدغه دیگر همکاری دارد اگر روشی که یک دغدغه را برآورده می‌کند، روشی را که دغدغه دیگر به وسیله آن برآورده می‌شود را تحت تاثیر قرار دهد. به عنوان مثال *"بهینه‌سازی"* همکاری با *"کارائی"* دارد. همکاری ضرورتا مثبت نیست. برای نمونه، *"واقعه‌نگاری"* با *"کارائی"* در یک حالت منفی همکاری دارد. همکاریها برای اثر تحلیل و تغییر انتشار[3] مهم هستند.

محرک: یک دغدغه منطقی دیگری را تحریک می‌کند اگر یک دغدغه، یک نیروی جنبشی یا دلیل برای دیگری فراهم کند. برای مثال، *"کارائی"* می‌تواند *"قابلیت پیکربندی"*، و *"قالیت ترمیم"* می‌تواند *"واقع‌نگاری"* را تحریک کند.

پیاده‌سازی منطقی: یک دغدغه منطقی، منطقا دغدغه دیگر را پیاده‌سازی می‌کند اگر یک دغدغه در ارتباط با پیاده‌سازی دغدغه دیگر معرفی شده باشد. به عنوان مثال یک گراف وابستگی به منظور پیاده‌سازی کردن مدیریت وابستگی شیء معرفی می‌شود. این پیاده‌سازی مشابه با پیاده‌سازی فیزیکی نیست. یا اینکه، یک واحد کد خاص یک تابع خاصی را پیاده‌سازی می‌کند.

پذیرش: یک دغدغه دغدغه، دیگر را می‌پذیرد اگر یک دغدغه حساسیتی ایجاد کند تا دیگری را در نظر بگیرید. به عنوان مثال، معرفی *"واقعه‌نگاری"* ، *"میانجی ثبت وقایع"*[4] را پذیرش می‌کند. *"واقعه‌نگاری"* نیازی ندارد که *"میانجی ثبت وقایع"* را تحریک کند اما بدون *"واقعه‌نگاری"* بی معنی است که *"میانجی ثبت وقایع"* در نظر گرفته شود. پذیرش، این حقیقت را منعکس می‌کند که معرفی تعدادی دغدغه، فضای دغدغه را برای در نظر گرفتن دغدغه‌های دیگر باز می‌کند مخصوصا دغدغه‌هایی که به صورت معنایی، وابسته به دغدغه معرفی شده هستند.

۴-۴-۲-۳- فیزیکی

ارتباطات فیزیکی مربوط به دغدغه‌های فیزیکی هستند. برای مثال، ترکیبی از مولفه‌های درون یک برنامه‌کاربردی. مطابق با ارتباطات تفسیری، طرح هسته cosmos انواع خاصی را برای ارتباط فیزیکی تعریف نمی‌کند اما اجازه می‌دهد که آنها مطابق با نیاز تحلیل و مدلسازی اضافه شوند. رابطه ارتباطات ویژه می‌تواند به

[1] Motivation
[2] Admission
[3] Propagation
[4] Log buffering

نوع دغدغه‌های فیزیکی مرتبط، هدف از ارتباط و تکنولوژیهای خاص استفاده شده در عناصر فیزیکی وابسته باشد.

۴-۴-۲-۴- نگاشت

ارتباطات نگاشت، دغدغه‌های منطقی و فیزیکی را به یکدیگر مرتبط می‌کنند. *"پیاده‌سازیهای فیزیکی"* یک مثال از ارتباط نگاشت است که در آن یک دغدغه فیزیکی مانند *"واحد کد"* یک دغدغه منطقی مانند یک *"تابع"* را پیاده‌سازی می‌کند. منطبق با ارتباطات تفسیری و فیزیکی، طرح هسته cosmos انواع خاصی برای ارتباطات نگاشت تعریف نمی‌کند اما پیش‌بینی می‌کند که انواعی از ارتباطات نگاشت باید برای اهداف خاص تعریف شوند. ارتباطات نگاشت به طور خاص برای دغدغه‌های منطقی مهم است که برای سازماندهی، انتخاب یا ترکیب دغدغه‌های فیزیکی استفاده می‌شوند [۶۴].

۴-۴-۳- مسندها و سازگاری

مسندها بازترین حوزه از طرح cosmos هستند. سازگاری یک پیامد مهم در مدیریت فضای دغدغه‌هاست اما انواع شرایط سازگاری، شرایط خاص و قوانین اجرای در سرتاسر مدلها و طرح cosmos متغییر خواهند بود. بنابراین cosmos در طرح هسته خود انواعی از مسندها را تعریف نمی‌کند اما آن، انواع مختلف از مسندها را به وسیله گسترش[1] پذیرش می‌کند.

cosmos از چندین شرایط سازگاری مرتبط با صحت طرح پایه استفاده می‌کند. بعضی شرایط (نوع)، محدودیت روی ارتباطات را نمایش می‌دهند. برای مثال، صفت‌ها فقط بر مجموعه‌ها و نمونه‌های فیزیکی اعمال می‌شوند. دیگر مسندها زیاد اساسی نیستند و می‌توانند به صورت گسترشها تعریف شوند. برای مثال، آیا می‌تواند یک نمونه متعلق به بیش از یک کلاس باشد؟ آیا یک نمونه می‌تواند متعلق به هیچ کلاسی نباشد؟ در طرح هسته این موارد اجازه داده شده‌اند ولی در یک پیکربندی خاص از آن می‌توان از اینها جلوگیری کرد.
دیگر مسندها می‌تواند در ارتباط با بخش غیر هسته‌ای طرح باشند از قبیل، ارتباطات تفسیری. تعدادی از شرایط سازگاری که در بین این ارتباطات اعمال می‌شوند برای مثال عبارتند از:

- محرک دلالت بر همکاری دارد (بر عکس این صحیح نیست)
- محرک و پیاده‌سازی منطقی، منحصر به فرد هستند.

مسندها می‌توانند برای ارتباطات فیزیکی، نگاشت و حتی برای ترکیبی از انواع ارتباطات تعریف و اجرا شوند.

[1] Extension

به منظور درک بهتر cosmos یک سیستم فرضی در نظر گرفته می‌شود و دغدغه‌های منطقی آن و ارتباطات قطعی و تفسیری آن ذکر خواهد شد. سیستم فرضی یک GPS cache[1] است [۶۵]. برای شروع کار با استفاده از cosmos، ابتدا تحلیل به وسیله تمرکز روی دغدغه‌های منطقی در طراحی حافظه میانجی صورت می‌پذیرد. این دغدغه‌های منطقی در جدول ۴-۶ بیان شده‌اند. همچنین نمونه‌هایی از ارتباطات قطعی و تفسیری دغدغه‌های به ترتیب در جداول ۴-۷ و ۴-۸ ذکر شده‌اند.

جدول ٤-٦: دغدغه‌های منطقی برای GPS cache

عنوان‌ها	خصوصیات	نمونه‌ها	کلاسها	طبقه‌بندی‌ها
Logging Garbage Algorithms Buffering Invalidation	Generality Configurability Correctness Consistency Transparency Persistence Recoverability concurrency	Add-object function Delete-object function Add-dependency function Add-object behavior Invalidate-dependency behavior Log operations behavior Log statistics behavior Collect-garbage behavior Logging optionally Cache size Maximum object size	Core functionality Invalidation functionality Dependency-management Functionality Core behavior Logging behavior Garbage-collection behavior Logging configurability Storage configurability Caches state Cache-object state	Functionality Behavior Configurability States Parameters

جدول ٤-٧: مثالهایی از ارتباطات قطعی (از راست به چپ خوانده می‌شود)

حیطه	ارتباط	دامنه
Core functionality	طبقه‌بندی از	Functionality
Logging behavior	طبقه‌بندی از	Behavior
Logging behavior	زیر کلاسی از	Operation logging behavior
Operation logging behavior	نمونه‌ای از	Flush operation log buffer
Core functionality	نمونه‌ای از	Addobject function
Behavior	خصوصیتی از	Generality
Functionality	خصوصیتی از	Configurability
Logging topic	موضوعی از	Logging behavior
Logging topic	موضوعی از	Logging configurability

[1] General-purpose software cache

جدول ٤-٨: مثالهایی از ارتباطات تفسیری (از راست به چپ خوانده می‌شود)

دامنه	ارتباط	حیطه
Performance	محرک	Configurability
Performance	محرک (منفی)	Logging
Recoverability	محرک	Logging
Configurability	همکاری با	Generality
Logging behavior	همکاری با (منفی)	Performance
Optimization	همکاری با	Performance
Storage optimization	همکاری با	Optimization
Logging	پذیرش	Logging optionally
Object dependency graph	پیاده سازی منطقی	Object dependency management

مدلسازی فضای دغدغه کاربرد بالقوه‌ای در توسعه نرم‌افزار دارد. یک مدل cosmos، شکلی از مستندات برای اطلاعات پایه‌ای، پیرامون دغدغه‌ها و ارتباطات بین آنها فراهم می‌کند. اگر چه اطلاعات تفصیلی پیرامون دغدغه‌ها در محصولات کاری مختلف یافت خواهد شد، یک مدل cosmos می‌تواند یک دید کلی را فراهم کند که مبتنی بر ترکیب‌ها و ارتباط دغدغه‌ها از چندین محصولات کاری و مراحل چرخه حیات است. یک مدل cosmos می‌تواند از انواع پرس و جوها و تحلیل‌ها از دغدغه‌ها و ارتباط‌های آنها حمایت کند (برای نمونه‌های از سئوالات درباره سیستم فرضی GPS cache که می‌توان با توجه به مدل پاسخ داد به مرجع [۵] مراجعه شود). یک مدل cosmos که شامل دغدغه‌های منطقی و ارتباطات نگاشت است می‌تواند به عنوان یک شاخص فوق معنایی[1] ارائه شود که اجازه می‌دهد دغدغه‌ها در داخل محصولات کاری و وظایف توسعه ردیابی شود. این حمایت از ردیابی دغدغه‌ها در سرتاسر محصولات کاری و مراحل توسعه، امکان پذیر می‌سازد تا چگونگی به وجود آمدن، پخش شدن و احتمالا حذف شدن دغدغه‌ها در سرتاسر مراحل و فعل و انفعالات چرخه حیات مشاهده شود [۶۶].

ارتباطات نگاشت همچنین اجازه می‌دهند تا تاثیر تغییرات بر روی سطح فیزیکی، روی سطح منطقی ارزیابی شود. برای مثال، اگر توجه زیادی وجود نداشته باشد به استحکام[2] و از بین رفتن[3]، که محرکی برای دغدغه‌هایی مثل واقعه‌نگاری است دیگر این توانائی نمی‌تواند حاصل شود (وجود ندارد) که به طور امن یک واحد نرم‌افزاری که یک دغدغه را پیاده‌سازی می‌کند حذف کرد یا اینکه ممکن است واحدی که برای حذف در نظر

[1] Semantic hyper index
[2] Robustness
[3] Lose

گرفته شده است همکاری با دیگر اهداف داشته باشد و بنابراین نباید حذف شود (مانند واقعه‌نگاری، که همچنین از رسیدگی نیز حمایت می‌کند).

کاربرد دیگر cosmos در سازماندهی کد (یا دیگر واحدها) برای اهداف ترکیب برنامه مبتنی بر دغدغه[1] است. cosmos همراه با Hyper/j [۶۷] برای ترکیب مبتنی بر دغدغه مولفه‌های جاوا مورد استفاده قرار می‌گیرد [۶۴]. این موضوع از تکامل مولفه در پاسخ به تکامل دغدغه حمایت می‌کند و اجازه می‌دهد تا تعریف "محصولات هم خانواده[2]" به وسیله دغدغه‌ها سازمانده‌هی شوند.

۴-۵- روش Theme/Doc

روش Theme، روشی برای نمایش ارتباطات بین رفتارها در یک مستند نیازمندیها، شناسائی و مجزا کردن جنبه‌ها در نیازمندیها و مدلسازی آن جنبه‌ها با استفاده از یک زبان طراحی است [۶]. روش Theme/Doc از توسعه جنبه‌گرا در دو سطح (نیازمندی و طراحی) حمایت می‌کند. در سطح نیازمندی، Theme/Doc دیدهایی[3] را از متن مشخصات نیازمندیها فراهم کرده و ارتباط بین رفتارها در سیستم را نمایش می‌دهد. در سطح طراحی نیز، Theme/UML [۶۸، ۶۹] به توسعه‌دهنده اجازه می‌دهد که ویژگیها و جنبه‌های سیستم را مدلسازی کرده و مشخص کند که چگونه آنها باید با یکدیگر ترکیب شوند.

Theme/Doc به توسعه‌دهنده اجازه می‌دهد در جهت آشکار ساختن اینکه در سیستم کدام عملکرد مداخله‌ای است و کدام بخش از سیستم را میانبر می‌کند، دیدهای نیازمندیها را بهبود دهد. روش theme[4] به قابلیت ردیابی، از مرحله نیازمندیها تا طراحی کمک می‌کند زیرا، نیازمندیها مستقیما به دیدهای Theme/Doc نگاشت شده و آنها نیز مستقیما به Theme/UML نگاشت می‌شوند. این قابلیت ردیابی باعث می‌شود که میزان پوشش نیازمندیها در طراحی مشخص شود.

در روش theme، یک theme یک عنصر طراحی است. یعنی، مجموعه‌ای از ساختارها و رفتارهایی که یک ویژگی را نمایش می‌دهد. چندین theme می‌تواند با یکدیگر ترکیب یا یکپارچه شوند و یک سیستم را تشکیل دهند [۲۶]. themeها در سطح طراحی، در [۶۸] معرفی شده‌اند. مدل theme دو نوع theme دارد. این دو نوع، theme پایه و theme مداخله‌ای هستند. themeهای پایه می‌تواند بعضی رفتارها و ساختارها را با دیگر themeهای پایه به اشتراک بگذارند در حالی که مدلسازی هر یک از آنها، دید خاص خود را دارد. themeهای

[1] Concern-driven program composition
[2] Product families
[3] Views

[4] از نظر لغوی theme به معنی موضوع است

مداخله‌ای نیز رفتارهایی دارند که با themeهای پایه هم پوشانی دارند. در واقع themeهای مداخله‌ای همان جنبه‌ها هستند [۶۹].

روش theme به دو بخش Theme/Doc و Theme/UML تقسیم می‌شود. این دو به themeهای یکسانی ارجاع داشته و بر روی آنها عملیات انجام می‌دهند اما آنها را در فازهای مختلف چرخه حیات به تصویر می‌کشند. Theme/Doc دیدهایی را برای شناسائی و به تصویر کشیدن در فاز تحلیل فراهم می‌کند در حالی که Theme/UML اجازه می‌دهد که با استفاده از UML استاندارد در فاز طراحی، ساختارها و رفتارهای مناسب برای هر theme مدلسازی شود.

در ادامه، روش theme با استفاده از یک مثال ساده (سیستم مدیریت دوره آموزشی[1]) تشریح می‌شود تا مشخص شود که چگونه Theme/Doc و Theme/UML در جهت شناسائی، طراحی و بررسی طراحی جنبه‌ها در یک مجموعه از نیازمندیها به کار گرفته می‌شوند.

سیستم مدیریت دوره آموزشی: سیستم مدیریت دوره آموزشی یک سیستم خیلی کوچک با نه نیازمندی به شرح زیر است:

- R1. دانش آموز می‌تواند برای دوره‌های آموزشی ثبت نام کند.
- R2. دانش آموز می‌تواند ثبت نام دوره‌های آموزشی را لغو کند.
- R3. زمانی که یک دانش آموز ثبت نام کرد سپس باید وقایع در رکورد خودش ثبت شود.
- R4. زمانی که یک دانش آموز ثبت نام را لغو کرد وقایع آن نیز باید ثبت شود.
- R5. استاد می‌تواند ثبت نام دانش آموزان را لغو کند.
- R6. زمانی که استاد ثبت نام یک دانش آموز را لغو کرد باید وقایع ثبت شود.
- R7. زمانی که استاد ثبت نام یک دانش آموز را لغو کرد آن باید به عنوان یک حالت خاص علامت‌گذاری شود.
- R8. استاد می‌تواند برای دوره‌های آموزشی نمره بدهد.
- R9. زمانی که استاد یک نمره داد وقایع باید در رکورد ثبت شود.

روش theme برای عملکرد خود، سه مرحله را در نظر می‌گیرد این سه مرحله به ترتیب اجرا عبارتند از : شناسائی جنبه‌ها با استفاده از دیدهای کنش، طرح‌ریزی برای طراحی با استفاده از دید theme، و بررسی مجدد

[1] Course Management System (CMS)

themeها با استفاده از دید theme تقویت شده. این مراحل به ترتیب بیان شده برای سیستم مدیریت دوره آموزشی تشریح خواهند شد.

۴-۵-۱- شناسائی جنبه‌ها با استفاده از دیدهای کنش[1]

برای شناسائی رفتارهای مداخله‌ای در این روش از *دید کنش* مستند نیازمندیها استفاده می‌شود. برای ایجاد *دید کنش*، دو ورودی مورد نیاز است: اولی، یک لیستی از کنش‌های کلیدی است که به وسیله توسعه‌دهنده با جستجو در مستندات نیازمندیها بدست می‌آید و دیگری، انتخاب کردن فعل‌های مشهود و نیازمندیهایی است که در مستند اصلی نیازمندیها نوشته شده است.

Theme/Doc تحلیل لغوی را روی متن انجام می‌دهد و *دید کنش* را تولید می‌کند. این عملیات می‌تواند با ابزار Theme/Doc انجام شود. با استفاده از این ابزار توسعه‌دهنده می‌تواند ارتباط بین رفتارهای تشریح شده در مستندات نیازمندی را بررسی کرده و تعیین کند کدام رفتارها پایه و کدامیک مداخله‌ای هستند. هر *کنش* به طور بالقوه یک theme است که به صورت جداگانه در Theme/UML طراحی می‌شود.

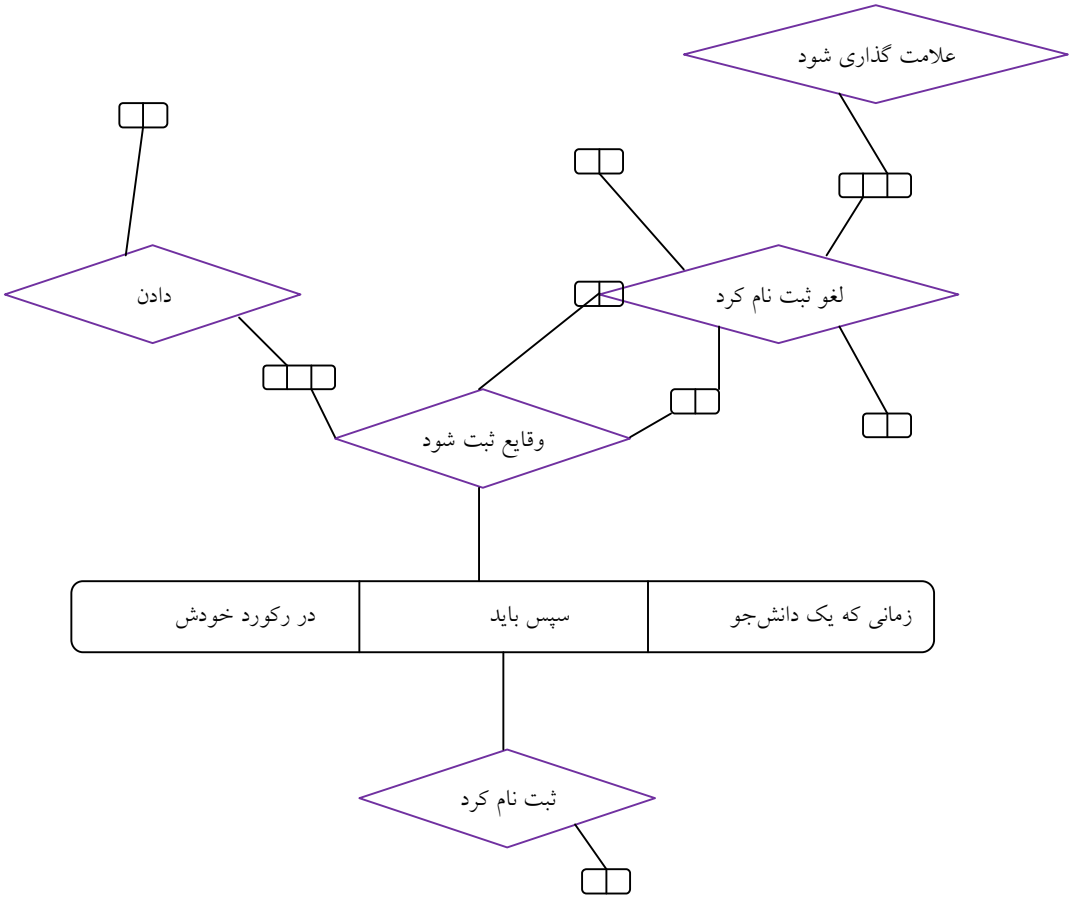
برای سیستم مدیریت دوره آموزشی پنج *کنش* شناسائی می‌شود که عبارتند از : ثبت نام کردن (Register)، لغو ثبت نام کردن (unregister)، ثبت وقایع کردن (logged)، علامت‌گذاری کردن (flagged) و دادن (give).

نکته: به دلیل اینکه یک کلمه در زبان انگلیسی ممکن است هم فعل و اسم باشد در هنگام استفاده از آنها در شکلها و متن سعی شده تغییرات لازم در معنای کلمات انگلیسی داده شده است تا روش، کارائی و خوانائی خود را حفظ کند.

شکل ۴-۳ *دیدهای کنش* ایجاد شده به وسیله ابزار Theme/Doc را برای سیستم مدیریت دوره آموزشی نشان می‌دهد. در شکل ۴-۳ پنج *کنش* به صورت لوزی مشاهده می‌شود. نیازمندیهای موجود در متن، به صورت جعبه‌هایی از کناره‌ها گرد شده (رکورد جمله[2]) در شکل نمایش داده شده‌اند که شامل کلماتی از جمله اصلی هستند. اگر جمله نیازمندی، شامل یک *کنش* کلیدی باشد آن *کنش* به رکورد جمله مورد نظر متصل خواهد شد. هدف از *دید کنش* برجسته کردن ارتباطات بین *کنش‌ها* است. به همین دلیل، خود متن درون هر نیازمندی در این مرحله مهم نیست و بنابراین نیازی نیست که مشاهده شوند. البته یک کاربر ممکن است یک نیازمندی را انتخاب کرده و آن را برای دریافت اطلاعات برجسته کند (یعنی محتویات نیازمندی را مشاهده کند). به عنوان مثال در شکل۴-۳ ، R3 انتخاب و برجسته شده است که به صورت زیر خوانده می‌شود:

″ زمانی که یک دانشجو ثبت نام کرد سپس باید وقایع ثبت شود در رکورد خودش″

[1] Action views
[2] Sentence record



شکل ۴-۳: دید کنش از نیازمندیهای سیستم مدیریت دوره آموزشی

همان طور که مشخص است هیچ کدام از *کنش‌ها* در نیازمندیها از بقیه مجزا نیستند و همه آنها به نحوی به بقیه متصل شده‌اند. این حقیقت که تمامی آنها به یکدیگر متصل هستند، نشان می‌دهد که رفتارهای مداخله‌ای و درهم تنیدگی در نیازمندیها وجود دارد. برای مثال، در شکل ۴-۳ کاملا مشخص است که *کنش "وقایع ثبت شود"* با چهار نیازمندی در ارتباط است و *کنش "ثبت نام کرد"* برای دو نیازمندی ذکر شده است. نیازمندیها اغلب به بیش از یک کنش ارجاع دارند. برای نمونه، R3 هم به *"ثبت نام کرد"* و هم به *"وقایع ثبت شود"* ارجاع دارد. جنبه‌ها در این روش به وسیله بازرسی این نوع نیازمندیهای مشترک شناسائی می‌شوند. در واقع هدف این است که گروهی از *کنش‌ها* و نیازمندیها، در *دید کنش* مجزا و ایزوله شوند تا به دو نوع از گروه‌های کنش/نیازمندی دست پیدا شود. اولی، گروه کاملی[1] است و نیازمندی ندارد که به کنشی در گروه‌های دیگر

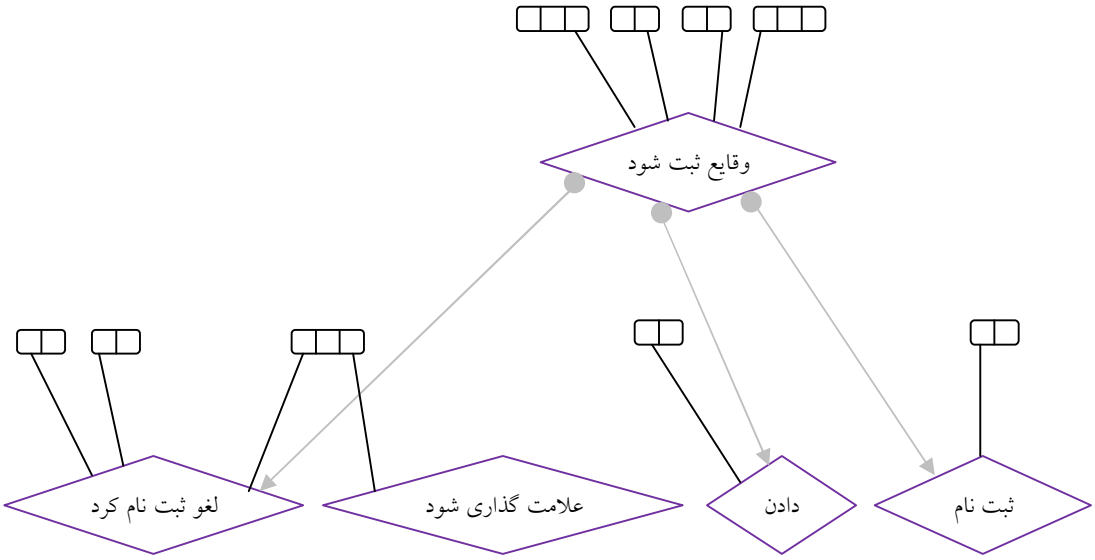
[1] Self-contained

ارجاع شده باشد. این نوع تحت عنوان *"پایه"* تعیین می‌شوند. دومین نوع، مداخله‌ای است که نیازمندیهایی دارد که ارجاع به کنش‌هایی در دیگر گروه‌ها دارد.

روش theme از عملکرد *"برش دادن"*[1] ابزار، برای دستیابی به مجزاسازی و ایزوله کردن استفاده می‌کند. نخست، هر نیازمندی مشترک بررسی می‌شود تا مشخص شود کدام کنش از آن، خیلی خوب به آن متصل است. اگر نیازمندی ابهام زیادی برای برقراری ارتباط با یک کنش یا کنش‌هایی دیگری داشته باشد. باید این ابهام حل شود. این عمل می‌تواند به وسیله نوشتن مجدد، تقسیم‌بندی کردن[2] و یا بهبود نیازمندی حاصل شود. برای نمونه، نیازمندی که *"وقایع ثبت شود"* را به *"ثبت نام کرد"* متصل می‌کند R3 است که رفتار *"وقایع ثبت شود"* را تشریح می‌کند. به عبارت دیگر، به رفتار *"ثبت نام کرد"* اضافه شده است. در اینجا قابل استباط است که *"وقایع ثبت شود"* رفتار اصلی این نیازمندی است و لذا، R3 باید با Theme *"وقایع ثبت شود"* پیوند بخورد. در نتیجه *"وقایع ثبت شود"* مداخله‌ای است و *"ثبت نام کرد"* پایه است. بعد از انجام مرحله نخست، در مرحله دوم پیکانی که R3 را به *"ثبت نام کرد"* متصل می‌کند باید حذف می‌شود. یعنی R3 فقط یک اتصال با *"وقایع ثبت شود"* پیدا می‌کند. سپس آن پیکان (یعنی، پیکان باقی مانده) با یک پیکان خاکستری رنگ با یک نقطه در شروع آن جایگذین می‌شود که *"وقایع ثبت شود"* را به *"ثبت نام کرد"* متصل می‌کند. این عمل نشان دهنده این است که *"وقایع ثبت شود"*، *"ثبت نام کرد"* را میان‌بر می‌کند.

بعد از این عمل، هر نیازمندی را که *"وقایع ثبت شود"* با دیگر کنش‌ها به اشتراک گذاشته است بررسی می‌شود تا اینکه مشخص شود آیا آنها متعلق به *"وقایع ثبت شود"* هستند یا اینکه، به کنش دیگری تعلق دارند. با توجه به اینکه مشخص شد *"وقایع ثبت شود"*، *"ثبت نام کرد"* را میان‌بر می‌کند احتمال این وجود دارد که کنش‌های دیگری را نیز که به آن متصل هستند میان‌بر کند. بدین منظور، حذف کردن ارتباط بین نیازمندیهای مشترک و کنش‌های پایه ادامه پیدا می‌کند. نهایتا بعد از انجام کلیه این عملیات به ساختار دلخواه که *"دید کنش برش شده"*[3] است خواهیم رسید که در شکل ۴-۴ به تصویر کشیده شده است.

[1] Clipping
[2] Splitting
[3] Clipped action view



شکل ۴-۴: دید کنش برش شده

در شکل ۴-۴ چهار کنش با نیازمندیهایی که فقط به خودشان ارجاع دارند، وجود دارد، بنابراین، اینها *"پایه"* هستند. یک کنش نیز وجود دارد که اشاره به دیگر کنش‌*ها* دارد بنابراین مداخله‌ای است. در *"دید کنش برش شده"* themeهای مداخله‌ای در بالای themeهایی قرار می‌گیرند که آنها را میان‌بر می‌کنند. همچنین پیکان‌های خاکستری رنگ، مداخله‌ای بودن را در *"دید برش شده"* نشان می‌دهند. این عمل به چگونگی پیکربندی ارتباطات پایه–جنبه در سطح طراحی کمک می‌کند.

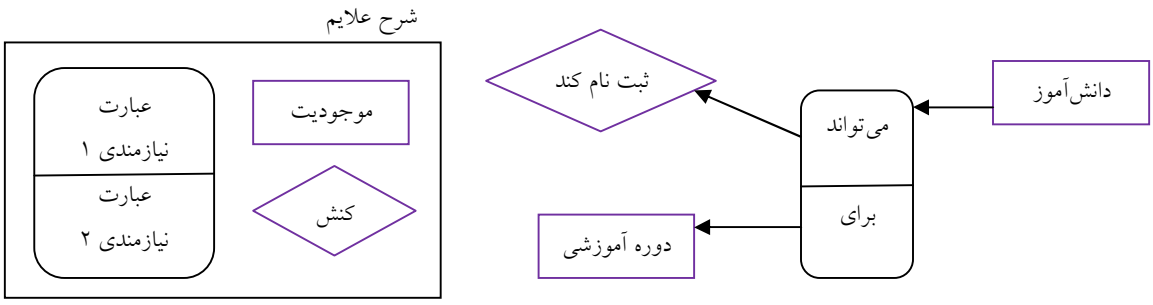
در شکل ۴-۴ همچنین مشاهده می‌شود که *"علامت‌گذاری شود"* به *"لغو ثبت نام کرد"* متصل است. اگر نیازمندی که میان آن دو مشترک است بررسی شود، دیده می‌شود که آن دو در رفتار *"استاد"* با یکدیگر اشتراک دارند. بنابراین می‌توان رفتار *"علامت‌گذاری شود"* را به عنوان میان‌بر کننده رفتار *"لغو ثبت نام کرد"* در نظر گرفت. بعد از این مراحل هر گروه، یک Theme می‌شوند که می‌توان آنها را با Theme/UML مدلسازی کرد.

۴-۵-۲- طرح‌ریزی برای طراحی با استفاده از دید Theme

در Theme/Doc دید theme برای طرح‌ریزی جهت طراحی و مدلسازی themهای شناسائی شده در مرحله قبلی به کار می‌رود. دیدهای theme از دیدهای *کنش* متفاوت هستند زیرا دیدهای theme فقط نیازمندیها و کنش‌*ها* را نشان نمی‌دهند بلکه عناصر کلیدی سیستم، که مورد نیاز است برای طراحی هر theme مد نظر قرار بگیرد را نیز نشان می‌دهد. برای ایجاد یک دید theme، توسعه‌دهنده باید مجموعه کلمات کلیدی اضافی فراهم

کند یعنی باید موجودیت‌های کلید را مشخص کند. مشابه دید کنش، دید theme از طریق تحلیل لغوی متن مستندات نیازمندی ایجاد می‌شود.

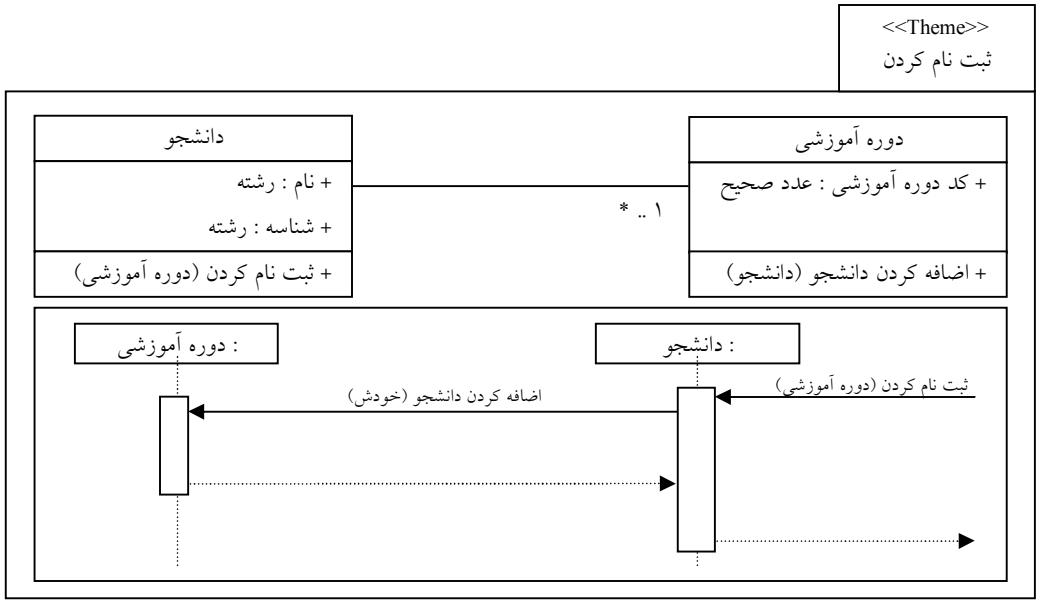
شکل ۴-۵ دید theme از theme *"ثبت نام کردن"* برش شده از رفتار مداخله‌ای را نشان می‌دهد. قابل مشاهده است که آن فقط یک نیازمندی دارد و به هیچ کنشی به غیر از رفتار *"ثبت نام کند"* ارجاع ندارد.



شکل ۴-۵: دید theme مربوط به Theme/Doc - ثبت نام کردن

هنگام خواندن جملات در Them/Doc، ابتدا عناصری خوانده می‌شوند که به رکورد جمله اشاره می‌کنند. سپس اولین عبارت در رکورد خوانده می‌شود، بعد عنصری که توسط آن عبارت مورد ارجاع قرار گرفته است خوانده می‌شود. اگر عبارت خالی باشد به معنی این است که فقط یک فضای خالی (یعنی، بدون هیچ عبارت متصل کننده) بین اولین و دومین عنصر وجود دارد. سپس عبارتهای بعدی در رکورد و عنصری که توسط آن عبارت مورد ارجاع قرار گرفته است خوانده می‌شوند. این عملیات جلو و عقب رفت بین عناصر و عبارتها تا جائی که به انتهای رکورد رسیده شود ادامه پیدا می‌کند.

با استفاده از دید theme می‌توان طرح‌ریزی کرد که کدام کلاسها و متدها در Theme/UML مربوط به *"ثبت نام کردن"* ظاهر شود. هنگام مدلسازی با استفاده از Theme/UML از شیوه‌های مناسب طراحی شیءگرا باید برای کمک کردن به تعیین اینکه کدام کلاسها و متدها تشریح شده‌اند استفاده کرد. این موضوع در شکل ۴-۶ برای *"ثبت نام کردن"* به تصویر کشیده شده است. در این حالت، هر کنش در دید theme یک متد است و هر موجودیت یک کلاس است. همچنین در شکل ۴-۶، تصمیمات طراحی اضافی، به منظور دادن قابلیت اجرا به رفتار *"ثبت نام کردن"* اضافه شده است.

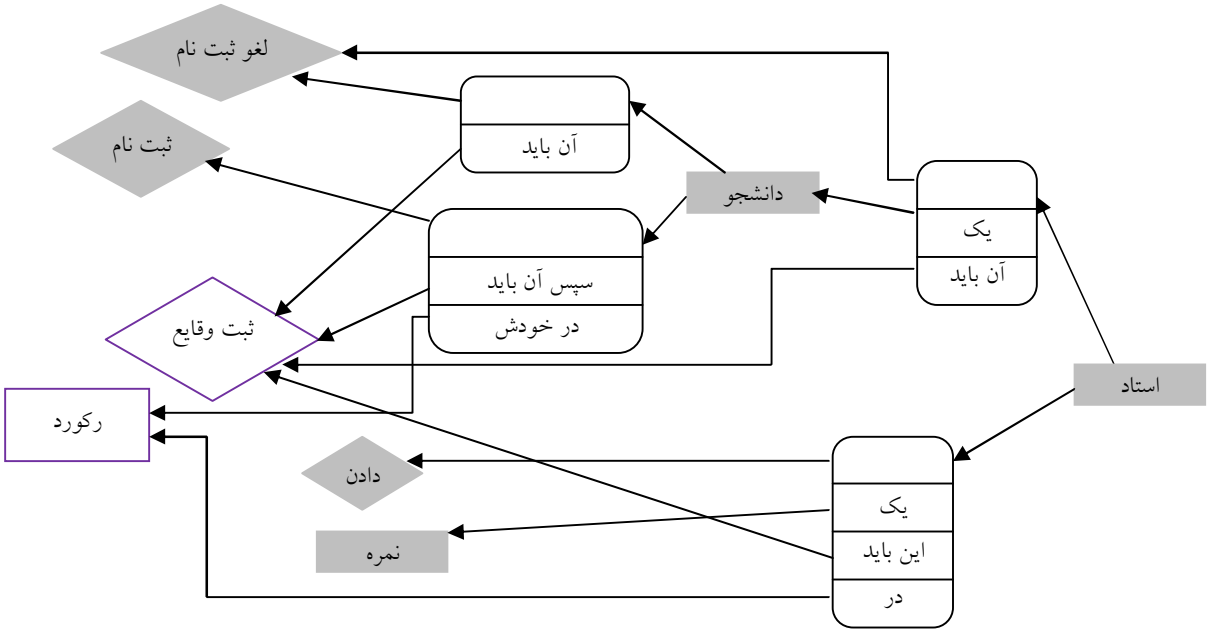


شکل ۴-۶: Theme/Doc - "ثبت نام کردن"

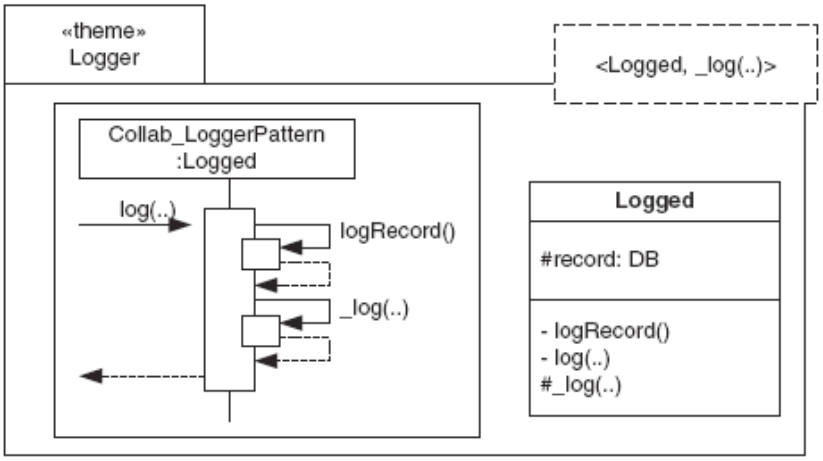
در هنگام مدلسازی دید theme مداخله‌ای، هدف مدلسازی کردن رفتار مداخله‌ای در یک انتزاع و روش قابل استفاده مجدد است و نیازی نیست که ارجاع صریح به هر کنش *پایه* یا موجودیت صورت پذیرد. دید theme به themeهای مداخله‌ای با شناسائی چنین عناصری به وسیله خاکستری رنگ کردن کنش*ها* و موجودیت‌های پیدا شده در دیگر themeها کمک می‌کند. دیگر کنش*ها* و موجودیت‌هایی که به رنگ سفید باقی مانده‌اند می‌توانند برای کمک به طراحی انتزاعی رفتار مداخله‌ای استفاده شوند. کنش*های* خاکستری رنگ همچنین برای تعیین الحاق یا انقیاد رفتار مداخله‌ای به *پایه* خاکستری رنگ استفاده می‌شوند. یک مثال از این حالت در شکل ٤-٧ نشان داده شده است. در شکل ٤-٧ مشخص است که فقط عناصر *"ثبت وقایع"* و *"رکورد"* منحصر به theme *"ثبت وقایع کردن"* هستند. بنابراین می‌توان رفتار انتزاعی برای *"ثبت وقایع کردن"* را بدون ارجاع مستقیم به *"ثبت نام"*، *"لغو ثبت نام"*، *"دانش آموزان"*، *"استاد"* و *"دادن نمره"* مدلسازی کرد.

مدلسازی انجام شده برای theme *"ثبت وقایع کردن"* در شکل ۴-۸ نشان داده شده است. در شکل ۴-۸ قابل مشاهده است که عنصر *"رکورد"* از theme *"ثبت وقایع کردن"* در طراحی به یک پایگاه داده تبدیل شده است و کنش *"ثبت وقایع"* خیلی ساده به متد logRecord تبدیل شده است. Theme/UML اجازه استدلال درباره عناصر را، از طریق یک *پایه* با استفاده از قالب‌هایی که محدود به عناصر واقعی *پایه* در مراحل بعدی شده‌اند، می‌دهد.

به منظور تعیین اینکه چگونه theme مداخله‌ای باید به ویژگیهای پایه وارد شود. باید به دید theme ″ثبت وقایع کردن″ نگاه شود (شکل ۴-۷). تمام کنش‌های خاکستری رنگ رفتارهایی هستند که به وسیله رفتار ″ثبت وقایع″ میان‌بر شده‌اند.

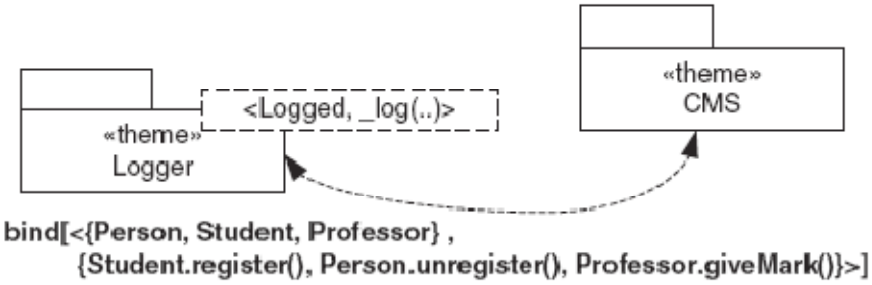


شکل ۴-۷: دید theme مربوط به Theme/Doc- ثبت وقایع کردن



شکل ۴-۸: Theme/UML - ثبت وقایع کردن

Theme/UML برای *"ثبت وقایع"* متد ()log_ را به عنوان یک متد کنترل کننده برای رفتار پایه در نظر می‌گیرد. برای حل این مشکل که متد ()log_ واقعا چه متدی است از ویژگی مقید کردن[1] Theme/UML برای قید کردن آن به یک متد واقعی‌تر[2] از ویژگیهای دیگر سیستم، استفاده می‌شود. بنابراین متد ()log_ به *کنش‌های* خاکستری رنگ در دید theme *"ثبت وقایع کردن"* قید می‌شود. این *کنش‌های* خاکستری رنگ *"ثبت نام"*، *"لغو ثبت نام"* و *"دادن (نمره)"* هستند. برای تعیین اینکه کلاسها به کدام متدها تعلق دارند باید به شکل ۴-۷ توجه شود. قابل مشاهده است که متد *"ثبت نام"* در ارتباط با کلاس *"دانشجو"* است (در شکل۴-۶ تشریح شده است) و متد *"دادن نمره"* متعلق به کلاس *"استاد"* است. هر دو کلاس *"استاد"* و *"دانشجو"* به متد *"لغو ثبت نام"* متصل هستند. بنابراین می‌توان از کلاس والد اینها که کلاس *"شخص"* است (این موضوع در شکل بیان نشده است) برای مقیدسازی استفاده کرد. شکل ۴-۹ نحوه قید کردن برای یکپارچگی theme *"ثبت وقایع کردن"* به theme *"سیستم مدیریت دوره آموزشی"* را نشان می‌دهد که یک فرآورده از ادغام تمام themeهای پایه است.



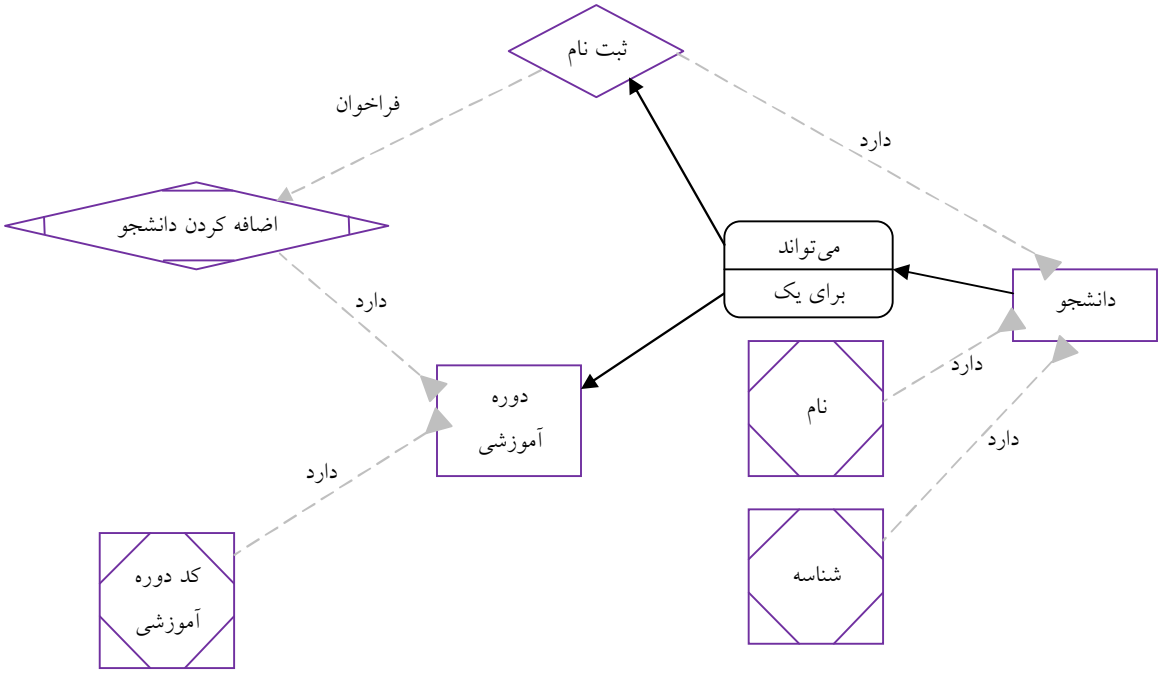
شکل ۴-۹: ترکیب ثبت وقایع با دید theme ها

۴-۵-۳- بررسی مجدد theme ها با استفاده از دید theme تقویت شده

بعد از اینکه themeها با استفاده از Theme/UML طراحی شدند باید دیدهای theme متعلق به Them/Doc برای بازبینی اینکه طراحی انجام شده همسو با نیازمندیها است، دوباره بررسی شوند. برای انجام این عمل، دید theme متعلق به Theme/Doc با ارائه تصمیمات طراحی تقویت می‌شود. برای نمونه، برای تقویت کردن دید theme *"ثبت نام کردن"* نیاز است که یک متد و سه ارتباط اضافه شوند. نتایج این عمل در شکل ۴-۱۰ به تصویر کشیده شده‌اند. در دید تقویت شده ارتباط *"دارد"*[3] با استفاده از یک پیکان معکوس شده در عنصر دربرگیرنده، اشاره به یک عنصر دربرگرفته شده دارد. ارتباط *"فراخوان"*[4] با استفاده از پیکان نقطه چین نشان داده

[1] Bind
[2] Concrete
[3] Has
[4] Call

شده است. در شکل ۴-۱۰ یک متد به دید اضافه شده، که به صورت کنش نشان داده شده است و همچنین کلاس نیز به صورت یک موجودیت نشان داده شده است. به منظور مشخص کردن اینها به صورت یک حالت تقویت شده (افزایش‌ها) گوشه‌های آنها علامت‌گذاری شده‌اند. این تقویت‌ها به وسیله کاربر مشخص شده و به داخل ابزار Theme/Doc که گراف تقویت شده را تولید می‌کند وارد می‌شوند.

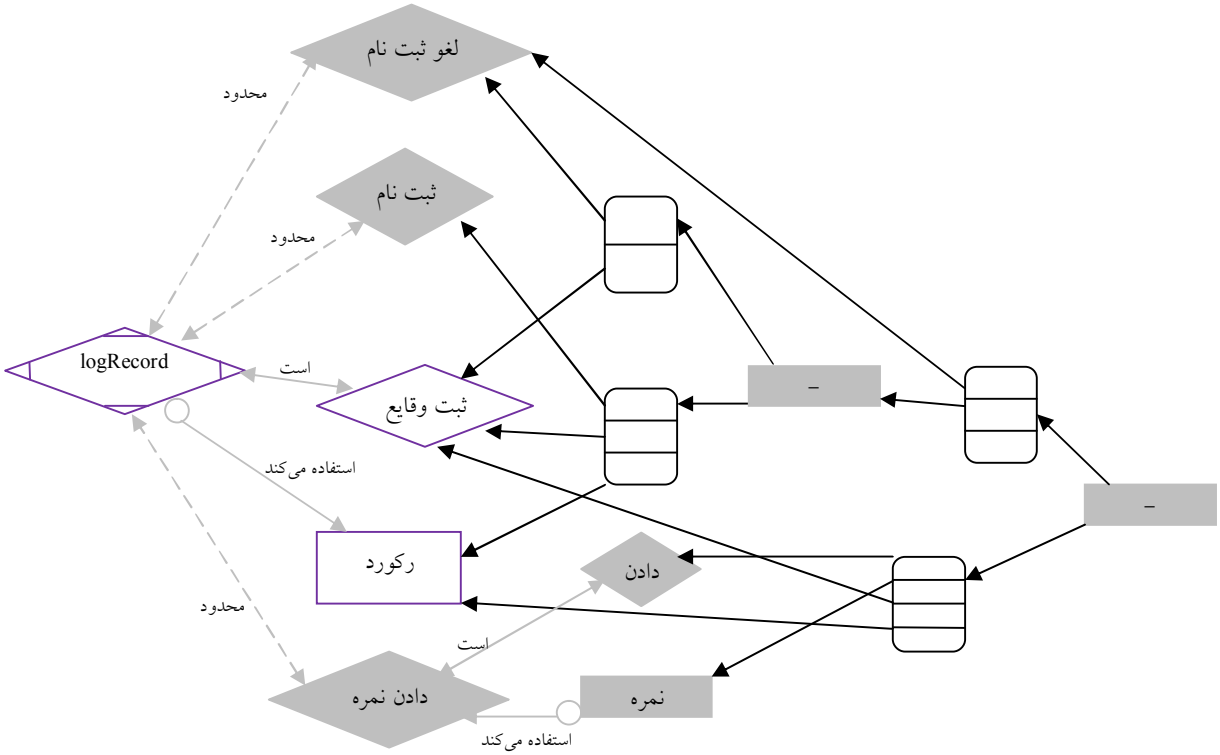


شکل ۴-۱۰: دید تقویت شده برای ثبت نام کردن

با نگاه کردن به شکل ۴-۱۰ قابل مشاهده است که طراحی Theme/UML همسو با نیازمندیهای نشان داده شده در Theme/Doc است. اگر عناصر طراحی دیگری (کلاسها و متدهای دیگر) در طراحی Theme/UML وجود داشته باشد آنها باید در اینجا مشاهده شوند و صحت آنها باید با یک نگاه به نیازمندیها قابل ارزیابی باشد. این دید (تقویت شده) تمایل ندارد که یک پایه رسمی[1] برای تحلیل فراهم کند و لذا، لزومی ندارد یک نگاشت ۱-۱ بین کنش‌ها/موجودیت‌های سطح نیازمندیها و ساختارها/عملیات‌های سطح طراحی وجود داشته باشد. به جای آن، هدف قرار دادن تصمیمات طراحی ایجاد شده در زمینه نیازمندی است. این زمینه سپس به توسعه‌دهنده یک دید برای بازبینی تصمیمات طراحی خودش می‌دهد.

[1] Formal basis

از آنجائی که Theme/UML برای theme مداخله‌ای، فقط اطلاعات مربوط به رفتار مداخله‌ای انتزاعی را در بر می‌گیرد بنابراین فقط ساختارهای مرتبط با رفتار به دید theme اضافه می‌شوند. این دید در شکل ۴-۱۱ نمایش داده شده است. در این شکل تمام عناصر تقویت شده برجسته شده‌اند[1]. مطابق قبل، تمام عناصری که منحصر به رفتار مداخله‌ای نیستند به صورت خاکستری رنگ نشان داده شده‌اند. این دید دو ارتباط دارد که دید تقویت شده *"ثبت نام کردن"* آن را ندارد. ارتباط "است یک[2]" که به وسیله یک پیکان دو طرفه نشان داده شده است و ارتباط "محدود[3]" که با یک پیکان دو طرفه خطچین نشان داده شده است. این ارتباطات در مفهوم قید کردن theme/UML وجود دارند. عناصری که در انقیاد (قید کردن) درگیر شده‌اند اما در دید نشان داده نشده‌اند نیز اضافه شده‌اند. برای نمونه، کنش *"دادن نمره"* به صورت خاکستری است زیرا آن در Theme/UML مربوط به theme *"ثبت وقایع کردن"* وجود ندارد. اما در اینجا در نظر گرفته شده است زیرا متد *"دادن نمره"* محدود به متد logRecord شده است.



شکل ۴-۱۱: دید تقویت شده برای ثبت وقایع کردن

[1] Enlarged
[2] Is-a
[3] bound

برای ارزیابی کامل بودن تصمیمات طراحی برای این دید، باید بررسی شود که آیا تمام عناصر خاکستری شده، به کنش نمایش دهنده رفتار مداخله‌ای محدود شده‌اند. همچنین باید بررسی شود که آیا تمام ساختارهای ذکر شده در نظر گرفته شده‌اند. برای مثال *"نمره"* در رفتار مداخله‌ای در نظر گرفته شده است زیرا آن به وسیله متد *"دادن نمره"* که محدود به logRecord است مورد استفاده قرار گرفته است.

۴-۶- روش جداسازی چند بعدی دغدغه‌ها در مهندسی نیازمندیهای جنبه‌گرا

این روش یک عملکرد یکنواخت[1] از دغدغه‌ها در سطح نیازمندیها، بدون در نظر گرفتن ماهیت وظیفه‌مندی، غیروظیفه‌مندی و مداخله‌ای بودن آنها ارائه می‌کند. روش، مبتنی بر مشاهده‌ای[2] است که دغدغه‌ها در یک سیستم دارند در حقیقت، یک زیر مجموعه -تحقق منسجم[3]- از دغدغه‌های انتزاعی[4] در یک فضای فوق دغدغه[5] است. هر شخص می‌تواند نیازمندیها را مطابق با این دغدغه‌ها انتزاعی تعیین کند تا دغدغه‌های منسجم‌تر و مختص به سیستم بیشتری را استخراج کند. در این روش یک مفهوم از ترکیب اشتراکی[6] معرفی خواهد شد که اجازه می‌دهد مجموعه‌ای از دغدغه‌های مناسب برای جداسازی چند بعدی (همین روش)، به عنوان پایه‌ای برای مشاهده مصالحه‌های میان دیگر دغدغه‌ها، انتخاب شوند. این عمل باعث فراهم شدن یک تحلیل دقیق ازمصالحه‌های سطح نیازمندی، بعلاوه درون‌بینی‌های[7] مهم در انواع تصمیمات معماری قابل دسترس برای بر طرف کردن دغدغه‌های وظیفه‌مندی و غیروظیفه‌مندی خاص، می‌شود [۷].

در واقع این روش پیشنهاد می‌کند که باید نیازمندیها در یک روش یکنواخت بدون در نظر گرفتن ماهیت وظیفه‌مندی و غیروظیفه‌مندی تجزیه شوند. این عمل این امکان را ایجاد می‌کند که هر مجموعه خاص از نیازمندیها روی یک محدوده‌ای از نیازمندیهای دیگر ارائه شوند (انتخاب شوند). لذا این باعث جداسازی چند بعدی می‌شود. بنابراین خصوصیت‌های کلیدی این روش را می‌توان به صورت زیر بیان کرد:

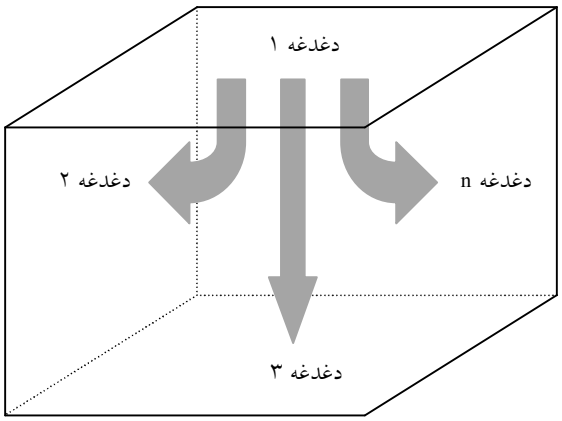
- تعریف یک فضای فوق دغدغه، باعث می‌شود که دغدغه‌های منسجم شده مختص به سیستم، بتوانند از ویژگیهای خاص حوزه مسأله مشتق شوند.

[1] Uniform treatment
[2] Observation
[3] Concrete
[4] Abstract concerns
[5] Meta concern space
[6] Compositional intersection
[7] Insights

- مفهوم یک ترکیب اشتراکی، این امکان را فراهم می‌کند تا شناسائی مجموعه‌ای مناسب از دغدغه‌ها در روش جداسازی چند بعدی به عنوان پایه‌ای برای مشاهده مصالحه‌های موجود بین دغدغه‌ها صورت پذیرد.
- استفاده از تحلیل مصالحه زود هنگام، باعث فراهم شدن درون‌بینی از تصمیمات مناسب معماری برای هر دغدغه می‌شود و لذا همسو کردن معماری مشتق شده با نیازمندیهای مورد نظر را تسهیل می‌دهد.

۴-۶-۱- جداسازی چند بعدی دغدغه‌ها

در این جداسازی چند بعدی، دغدغه‌ها به هر مجموعه منسجم از نیازمندیها دلالت دارند. در نظر گرفتن کلیه دغدغه‌ها در یک روش یکنواخت باعث می‌شود که دغدغه‌ها به دیدگاه‌ها، موارد کاربری و یا جنبه‌ها دسته‌بندی نشوند. به هر حال، این نوع دغدغه‌ها هنوز هم مجموعه منسجمی از نیازمندیهای وظیفه‌مندی و غیروظیفه‌مندی را محصور می‌کنند. در این جداسازی، فضای دغدغه در سطح نیازمندی به صورت یک فوق مکعب[1] در نظر گرفته می‌شود که شکل ٤-۱۲ نمایش ساده شده از فوق مکعب را به صورت یک مکعب نشان می‌دهد.



شکل ۴-۱۲: نمایش فوق مکعبی دغدغه‌ها در سیستم

هر سطح از فوق مکعب یک دغدغه خاص را نشان می‌دهد. با در نظر گرفتن کلیه دغدغه‌های هم ارز یکدیگر، می‌توان یک مجموعه از دغدغه‌ها را به عنوان پایه در نظر گرفت تا تاثیر دغدغه‌های دیگر یا مجموعه از دغدغه‌ها را روی پایه انتخاب[2] کرد. پیکان‌های موجود در شکل ۴-۱۲ انتخاب‌ها را نشان می‌دهند. این انعطاف‌پذیری دید چند بعدی، کنترل نیازمندیهای غیروظیفه‌مندی و وظیفه‌مندی مداخله‌ای را در یک روش

[1] Hypercube
[2] Project

کارآمد ممکن می‌سازد. (در بخش‌های بعدی تشریح خواهد شد که معماری انتخاب شده در مکانی در حوزه محدود شده به وسیله دیوارهای فوق مکعب قرار دارد)

۴-۶-۱-۱- مثال اجرائی

در جهت تشریح مفاهیم و تکنیک‌ها، یک مطالعه موردی به نام "سیستم راهنمای توریست حساس به زمینه و محل[1]" در متن این روش استفاده می‌شود. خصوصیات کلیدی این سیستم به شرح زیر است:

سیستم یک راهنمای دستی الکترونیکی فراهم می‌کند که تسهیلات زیر را برای بیننده خود ارائه می‌دهد:

۱. دریافت اطلاعات درباره شهر، شامل اطلاعاتی درباره موقعیت فعلی خود
۲. فراهم کردن راهنمای مسیریابی برای کمک به حرکت توریست‌ها بین محل‌های دیدنی
۳. وارد کردن مجموعه‌ای از علایق و ترجیحات، برای پیشنهاد دادن تورهای(گردشهای) مناسب از شهر
۴. دسترسی به سرویس‌های خارجی، مانند رزرو کردن هتل یا بلیط هواپیما

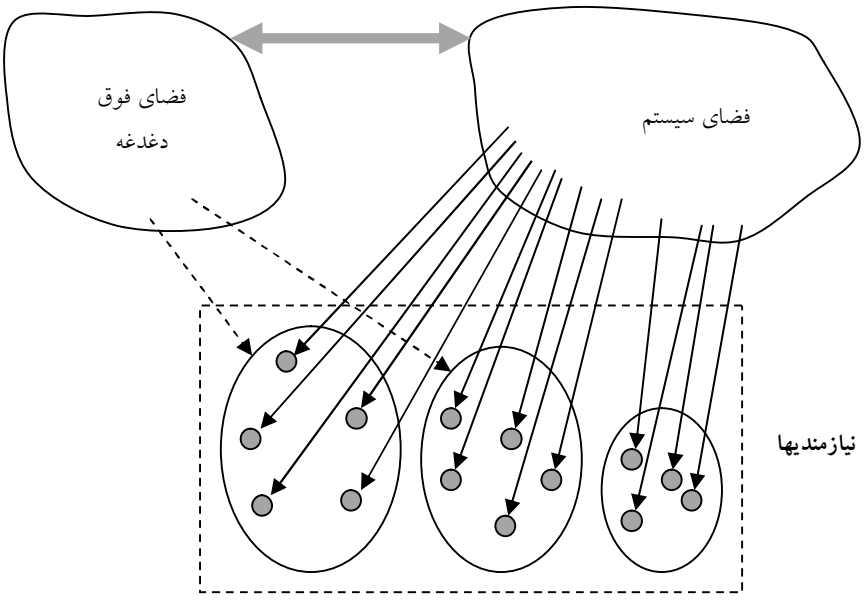
انواع دغدغه‌های سطح نیازمندی در این سیستم و دسته‌بندیهای آنها، مبتنی بر فضای فوق دغدغه در بخش بعدی تشریح خواهد شد.

۴-۶-۲- فضای سیستم و فضای فوق دغدغه

شناسائی دغدغه در این روش مبتنی بر مشاهده‌ای است که یک دغدغه مشخص –وظیفه‌مندی و غیروظیفه‌مندی– در طول توسعه نرم‌افزار یا همان لحظه از خود نشان می‌دهد. یک فهرستی از دغدغه‌های غیروظیفه‌مندی در [۷۰] مشخص شده‌اند که در روش چند بعدی (همین روش در حال تشریح) مفهوم یک چنین فهرستی، برای دغدغه‌های وظیفه‌مندی عمومی توسعه داده می‌شود. مثالهایی از چنین دغدغه‌ها که مکررا ظاهر می‌شوند شامل ثبت‌نام، مرتب‌سازی، صدور صورت حساب، رزرو بلیط، تغییرپذیری (قابلیت حرکت)، قابلیت دسترس و امنیت است (توجه شود که مثال‌ها دربرگیرنده هر دو نوع دغدغه‌های وظیفه‌مندی و غیروظیفه‌مندی است). بنابراین فضای نیازمندیها به دو فضای مجزا تقسیم می‌شوند (شکل ۴-۱۳):

- فضای سیستم، متشکل از انواع مختلفی از سیستم‌ها است که توسعه‌دهندگان خواهان تحقق آنها هستند.
- فضای فوق دغدغه، متشکل از مجموعه‌ی انتزاعی از دغدغه‌ها عمومی (وظیفه‌مندی و غیروظیفه‌مندی) اشاره شده در بالاست که مکررا خودشان را در انواع سیستم‌ها آشکار می‌سازند.

[1] Location and context sensitive tourist guide system



شکل ۴-۱۳: فضای سیستم و فضای فوق دغدغه

هر سیستم در فضای سیستم تعدادی ویژگیهای مطلوب[1] (مورد انتظار) دارد. این ویژگیها منجر به تشکیل نیازمندیهایی (نقاط خاکستری رنگ در شکل ۴-۱۳) برای توسعه می‌شوند که از طریق مصاحبه‌ها، مطالعات تشریحی[2]، تحلیل شیوه‌های کسب وکار سیستم، حاصل می‌شوند (به وسیله پیکان‌های تیره در شکل ۴-۱۳ نشان داده شده است).

به محض اینکه نیازمندیها از ویژگیهای مطلوب سیستم مشتق شدند آنها در دغدغه‌هایی از، فضای فوق دغدغه دسته‌بندی می‌شوند (به وسیله پیکان‌های خط چین در شکل ۴-۱۳ نشان داده شده‌اند). این دسته‌بندی منجر به تعاریف خاص منسجم حوزه و سیستم[3]، از دغدغه‌های انتزاعی می‌شود (بیضی‌های در شکل ۴-۱۳) و می‌تواند به صورت تدریجی و تکاملی به وسیله کنترل مجموعه کوچکی از دغدغه‌ها در هر مرتبه بدست آید. نکته قابل توجه این است که ضرورتی ندارد کلیه دغدغه‌ها در فضای فوق دغدغه در این دسته‌بندی استفاده شوند. اساسا، فقط یک زیر مجموعه‌ای از دغدغه‌های مربوطه به حوزه مسأله مورد نیاز است. دسته‌بندی نیازمندیها به داخل دغدغه‌های منسجم منجر به ایجاد یک ارتباط بین فضای فوق دغدغه و فضای سیستم می‌شود (به وسیله پیکان خاکستری رنگ در شکل ۴-۱۳ مشخص شده است). این باعث ایجاد یک انقیاد مفهومی بین نمایش انتزاعی دغدغه‌ها در یک فضای فوق دغدغه، با نمایش‌های منسجم خودشان در فضای سیستم می‌شود.

[1] Desirable
[2] Ethnographic studies
[3] System- and domain-specific

از مشخصات کلی مطالعه موردی راهنمای توریست می‌توان چندین دغدغه از فضای فوق دغدغه را شناسائی کرد. برای مثال، توریست (بیننده دستگاه) در حالی که در حال حرکت بین محلهای مختلف است اطلاعات را مشاهده می‌کند بنابراین *بازیابی اطلاعات* و *قابلیت* حرکت دو دغدغه مورد نیاز سیستم هستند. سیستم همچنین با سرویس‌های رزرواسیون خارجی ارتباط برقرار می‌کند بنابراین، *سازگاری* نیز یک دغدغه است. دیگر دغدغه‌ها، خیلی روشن در مشخصات کامل سیستم، نوشته‌های مربوط به مصاحبه و گفتگوی با ذینفعان، بعلاوه دانش حوزه‌ای فرد خبره در طراحی سیستم‌های محاوره‌ای قابل حمل، مشخص هستند. نهایتا، دغدغه‌های منسجم (از فهرست دغدغه‌های انتزاعی در فضای فوق دغدغه) شناسائی شده برای مطالعه موردی راهنمای توریست قابل حمل به شرح زیر هستند:

- *تعیین اعتبار*: بررسی اطلاعات شخصی با استفاده از یک شناسه معتبر، قبل از نمایش اطلاعات برای بیننده
- *قابلیت دسترسی*: تضمین اینکه سیستم همیشه فعال برای پاسخگوی است.
- *سازگاری*: تضمین تعامل با سرویسهای خارجی از قبیل رزرواسیون بلیط یا هتل
- *اتصال*: فراهم کردن ارتباط شبکه‌ای با سرویسهای خارجی قابل دسترس
- زمینه: تشخیص تغییر محل یک توریست در هنگام حرکت در شهر
- *قابلیت سفارشی کردن*: اجازه دادن به بیننده برای تنظیم تورها با توجه به علایق و ترجیحات
- *دریافت اطلاعات*: برای دریافت اطلاعات از سیستم
- *به روزرسانی اطلاعات*: اضافه، حذف، به روزرسانی اطلاعات پیرامون انواع مکانهای مورد علاقه توریست
- *قابلیت* حرکت: تضمین اینکه بیننده می‌تواند هنگام حرکت به اطلاعات دسترسی داشته باشد.
- *راهبری*: فراهم کردن راهنمایی‌هایی برای بیننده به منظور ادامه تورها
- *قابلیت حمل*: فراهم کردن دستگاه کم وزن برای دسترسی به سیستم
- *ثبت نام*: گرفتن اطلاعات شخصی از بیننده قبل از اجاره دادن دستگاه‌های الکترونیکی برای دسترسی به سیستم
- هزینه: محاسبه کردن هزینه (برای مثال توسعه، تجهیزات، استقرار، نگهداری عملیات)

برای منسجم کردن روش، دغدغه‌های انتزاعی در فضای فوق دغدغه بعلاوه تحقق منسجم آنها، به صورت یکجا با استفاده از قالب خوش تعریفی مبتنی بر XML نمایش داده می‌شوند. یک چنین نمایش نیمه ساخت‌یافته از دغدغه‌ها این امکان را فراهم می‌کند که بتوان قوانین ترکیب را تعریف کرد که مشخص می‌کند

چگونه یک دغدغه، نیازمندیها در دغدغه دیگر را محدود یا تحت تاثر قرار می‌دهد. همچنین باعث می‌شود بتوان از طرق تحلیل دغدغه‌ها وترکیبات آنها نقاط مصالحه بالقوه را ایجاد کرد. یک چنین روش مبتنی بر XML که به طور موثر تحلیل نیازمندیها را به صورت جزئی مکانیزه می‌کند در [۳۶] تشریح شده است. در حقیقت قوانین ترکیب در این روش مبتنی بر کنش‌ها و عملگرهای تعریف شده در [۳۶] هستند.

به منظور تشریح این روش از دو دغدغه *بازیابی اطلاعات* و *قابلیت حمل* مطالعه موردی استفاده می‌شود. شکل ۴-۱۴ و ۴-۱۵ تعریف انتزاعی این دو دغدغه را در فضای فوق دغدغه نشان می‌دهند. تعریف انتزاعی در درون تگ‌های <MetaConcern> </MetaConcern> قرار دارد که دربرگیرنده نام دغدغه در فضای فوق دغدغه نیز است. تعریف، همچنین شامل یک توضیح مختصر از دغدغه، تعدادی مثالهای عمومی که از تجربیات گذشته و دانش حوزه بدست آمده است و همچنین دغدغه‌هایی که ممکن است به آنها مرتبط بوده باشند (نام این دغدغه‌ها در درون تگ‌های <Relationship> </Relationship> قرار گرفته‌اند). نکته قالب توجه در مورد ارتباطات این است که اطلاعات ارتباطات به عنوان یک راهنما در نظر گرفته شده‌اند و نمی‌تواند برای تمام سیستم‌ها در نظر گرفته شوند زیرا در چندین حالت دیگر ممکن است ارتباطات اضافی با دیگر دغدغه‌ها در سیستم وجود داشته باشد.

```
<?xml version="1.0" ?>
- <MetaConcern name="InformationRetrieval">
        <Description>The operation of accessing information from a
                computer system </Description>
        <Examples>Database retrieval, Multimedia retrieval</Examples>
        <Relationships> Availability, Mobility, InformationUpdate
                </Relationships>
  </MetaConcern>
```

شکل ۴-۱۴: فوق دغدغه *دریافت اطلاعات* در XML

```
<?xml version="1.0" ?>
- <MetaConcern name="Mobility">
        <Description>The quality of moving freely </Description>
        <Examples>Wireless networks, Mobile phones, Context aware
                systems </Examples>
        <Relationships> Availability, Portability, Context </Relationships>
</MetaConcern>
```

شکل ۴-۱۵: فوق دغدغه *قابلیت حرکت* در XML

```
<?xml version="1.0" ?>
- <Concern name="InformationRetrieval">
        - <Requirement id="1">
                It should be possible to retrieve information from the system.
                <Requirement id="1.1">It should be possible to access
                        information about the attractions. </Requirement>
                <Requirement id="1.2">It should be possible to access
                        information about the current location. </Requirement>
        </Requirement>
        <Requirement id="1.3">It should be possible to obtain a list of
                available preset tours. </Requirement>
</Concern>
```

شکل ۴-۱۶: دغدغه *بازیابی اطلاعات* در XML

```
<?xml version="1.0" ?>
- <Concern name="Mobility">
        - <Requirement id="1">
                The system will be accessed on the move.
                <Requirement id="1.1">The system will be accessed from within
                        a limited area. </Requirement>
        </Requirement>
</Concern>
```

شکل ۴-۱۷: دغدغه *قابلیت حرکت* در XML

شکل ٤-١٦ و ٤-١٧ مشخصات XML دغدغه‌های منسجم شده *بازیابی اطلاعات* و *قابلیت* حرکت را نشان می‌دهند. یک تگ دغدغه، شروع یک دغدغه و یک تگ نیازمندی، شروع یک نیازمندی را مشخص می‌کند. اصلاحیه‌هایی همچون زیر-نیازمندی[1] از طریق تگ‌های تودرتو نمایش داده می‌شوند. هر نیازمندی یک شناسه دارد که در داخل حوزه خود (دغدغه) منحصر به فرد است. همچنین نام دغدغه‌ها در داخل مطالعه موردی منحصر به فرد هستند. با این حال از فضای نام[2] XML می‌تواند برای تحقق حوزه اسمی استفاده کرد.

۴-۶-۳- ترکیب و تحلیل مصالحه

به خاطر دسته‌بندی کردن نیازمندیهای مختلف در درون دغدغه‌های منسجم در فضای فوق دغدغه، باید قوانین ترکیبی برای هر دغدغه تعریف شود. این عمل به وسیله انتخاب مجموعه خاصی از دغدغه‌ها- در یک روش منظم- به عنوان پایه‌ای برای مشاهده مصالحه‌ها میان دیگر دغدغه‌ها انجام می‌شود.

۴-۶-۳-۱- مشخصات ترکیب

[1] Sub-requirements
[2] Namespaces

ارتباطات بالقوه‌ای که یک دغدغه می‌تواند با دیگر دغدغه‌ها داشته باشد در تعریف انتزاعی دغدغه در فضای فوق دغدغه مستند شده‌اند. این ارتباطات نمی‌تواند برای تمام حالات سیستم در نظر گرفته شود و ممکن است در حالاتی ارتباطات اضافی با دیگر دغدغه ایجاد شود. با این وجود، این ارتباطات نقطه شروع خوبی برای مهندس نیازمندی است تا مشخص کند که چگونه یک دغدغه، نیازمندیهای دغدغه‌های دیگر را که با آن در ارتباط است را محدود یا تحت تاثیر قرار می‌دهد. به عبارت دیگر، مشخصات ترکیب در قالب قوانین ترکیب بیان می‌شوند که شرح می‌دهند چگونه یک دغدغه، دغدغه‌های دیگر را در جداسازی چند بعدی میان‌بر می‌کند.

قوانین ترکیب، ارتباطات بین نیازمندیهای دغدغه‌ها را در دانه‌بندی ریز تعریف می‌کنند. مشابه با تعاریفی که برای دغدغه وجود داشت یک زبان توصیف ترکیب مبتنی بر XML نیز برای قوانین ترکیب تعریف می‌شود. تعاریف قوانین ترکیب می‌تواند توسط طرح‌های XML اداره شوند با این وجود برای سادگی بیشتر، به جای تعریف طرح XML ساختار قوانین ترکیب با توجه به چند مثال تشریح می‌شود. همان طور که در شکل ۴-۱۸ و ۴-۱۹ نشان داده شده است یک مجموعه منسجم قوانین ترکیب در یک تگ ترکیب احاطه شده‌اند.

شکل ۴-۱۸ کلیه ترکیبات برای *بازیابی اطلاعات* را محصور کرده است در حالی که شکل ۴-۱۹ عمل محصورسازی را برای *قابلیت* حرکت انجام داده است. مفهوم تگ نیازمندی در اینجا متفاوت از مفهوم آن در تگ‌های تعریف دغدغه است. نیازمندی متعلق به هر دغدغه به صورت ویژگی در آن ذکر شده است این عمل برای رسیدن به هدف لازم است. اگر یک نیازمندی دغدغه، زیر-نیازمندیهایی داشته باشد این نیازمندیها باید به صورت صریح، در نظر گرفته شده یا باید مستثنی شوند. این عمل به وسیله مشخص کردن مقدار "include" و"exclude" صورت می‌پذیرد. یک مقدار "all" برای ویژگی id در تگ نیازمندی بر این موضوع دلالت دارد که کلیه نیازمندیهای درون دغدغه مشخص شده، محدود شده‌اند. تگ محدودیت یک عملگر و کنش تعریف می‌کند. این عملگرها و کنشها مشخص می‌کند چگونه نیازمندی از یک دغدغه، نیازمندیهایی که در دغدغه‌های دیگر وجود دارند را محدود می‌کند.

```
<?xml version="1.0" ?>
- <Composition>
        - <Requirement concern="InformationRetrieval" id="all">
                - <Constraint action="provide" operator="during">
                        <Requirement concern="Customisability" id="all" />
                        <Requirement concern="Navigation" id="1" />
                        <Requirement concern="Mobility" id="1" children="include" />
                        <Requirement concern="InformationUpdate" id="all" />
                </Constraint>
                <Outcome action="fulfilled" />
        </Requirement>
</Composition>
```

شکل ۴-۱۸: قانون ترکیب برای *بازیابی اطلاعات*

```
<?xml version="1.0" ?>
- <Composition>
        - <Requirement concern="Mobility" id="all">
                - <Constraint action="affect" operator="on">
                        <Requirement concern="Availability" id="all" />
                        <Requirement concern="Connectivity" id="all" />
                        <Requirement concern="Context" id="all" />
                        <Requirement concern="Navigation" id="1" />
                        <Requirement concern="InformationRetrieval" id="all" />
                </Constraint>
                <Outcome action="fulfilled" />
        </Requirement>
</Composition>
```

شکل ۴-۱۹: قانون ترکیب برای *قابلیت حرکت*

اگرچه کنش‌ها و عملگرها غیررسمی هستند اما آنها معنی و مفهوم واضحی برای تضمین صحیح بودن ترکیب شدن دغدغه دارند. این برای معماران و طراحان یک توانائی سیستماتیک برای تفسیر مشخصات نیازمندیها فراهم می‌کند. تگ Outcome نتایج محدود کردن نیازمندیهای دغدغه را تعریف می‌کند. مقدار کنش تشریح می‌کند که آیا نیازمندی دغدغه یا مجموعه‌ای از نیازمندیهای دغدغه دیگر باید ارضا[1] شوند یا فقط محدودیت مشخص شده باید برآورده[2] شود.

غیررسمی بودن کنشها و عملگرها تضمین می‌کند که هنوز مشخصات ترکیب، قابل خواندن توسط ذینفعان هستند که یک ملاحظه مهم در طی مهندسی نیازمندیها است. برای مثال اگر به نخستین قانون ترکیب در شکل ۴-۱۸ توجه شود و تمرکز بر روی مقدار ضخیم شده[3] باشد، آنگاه داریم: "**بازیابی اطلاعات** باید **فراهم بشوند** در طی **کلیه نیازمندیهای قابلیت سفارشی کردن**، نیازمندی **۱** از **راهبری** و نیازمندی **۱** از **قابلیت حرکت** شامل کلیه فرزندانش، با نتیجه‌ای که مشخص می‌کند محدودیت برآورده شدن (fulfilled) است". بخش دوم از قانون ترکیب در یک روش مشابه با قبل تفسیر می‌شود. قوانین ترکیب در روش چند بعدی از قوانین ترکیب مشخص شده در [۳۶] تبعیت می‌کند. تنها تفاوت کلیدی این است که یک دغدغه، دغدغه‌ها دیگر را محدود می‌کند نه فقط دیدگاه‌ها را.

جدول‌های ۴-۹، ۴-۱۰ و ۴-۱۱ مفهوم کنش‌ها و عملگرها را برای Constraint و Outcome، شامل آنهائی که در قوانین ترکیب نشان داده شده در شکل‌های ۴-۱۸ و ۴-۱۹ مورد استفاده قرار گرفته‌اند را توضیح می‌دهند. نقطه جالب توجه این است که تمام عملگرها مختص دغدغه[4] نیستند. به عنوان مثال، XOR یک عملگر عمومی است همچنین اینکه کنشها برای Outcome عمومی هستند نه اینکه به یک دغدغه خاص

[1] Satisfied
[2] Fulfilled
[3] Bold
[4] Concern-specific

مشخص شده باشند. با این وجود ممکن نیست که گفته شود کنشهای Outcome همیشه عمومی هستند. بنابراین مطالعه موردی زیادی باید اجرا شوند قبل از اینکه بتوان چنین نتیجه‌گیری کرد. بیان این موضوع اهمیت دارد که ذکر شود اگرچه عملگرهای یکسان ممکن هستند به نیازمندیهای دغدغه متفاوت اعمال شوند، کلیه ترکیبات کنش-عملگر در مشخصات constraint برای یک دغدغه خاص معتبر نیستند. این موضوع قبلا در [۳۶] بررسی شده است.

جدول ۴-۹: شرح کنش‌های محدودیت

کنش محدودیت	
نوع	*توضیحات*
Enforce	برای اعمال شرایط اضافی روی مجموعه‌ای از نیازمندیهای دغدغه استفاده می‌شود.
Ensure	برای تاکید اینکه یک شرطی که باید برای یک مجموعه از نیازمندیهای دغدغه وجود داشته باشد واقعا وجود دارد، استفاده می‌شود.
Provide	برای مشخص کردن ویژگیهای اضافی ثبت شده برای یک مجموعه از نیازمندیهای دغدغه استفاده می‌شود.
Applied	برای شرح قوانینی که به یک مجموعه از نیازمندیهای دغدغه اعمال می‌شود و می‌توان خروجی خود را حذف کند، استفاده می‌شود.
Exclude	برای مستثنی کردن تعدادی دغدغه یا نیازمندی، اگر مقدار all مشخص شده باشد، استفاده می شود.
Affect	برای مشخص کردن اینکه مجموعه‌ای از نیازمندیهای دغدغه وضعیت دغدغه دیگر را حذف خواهند کرد، استفاده می‌شود.

جدول ۴-۱۰: شرح عملگرهای محدودیت

عملگر محدودیت	
نوع	*توضیحات*
During	تشریح بازه زمانی بین اینکه یک مجموعه از نیازمندیها ارضا شده‌اند.
Between	بازه زمانی قرار داشته بین ارضا دو نیازمندی را شرح می‌دهد. بازه، زمانی شروع می‌شود که نخستین نیازمندی ارضا شده و هنگامی به پایان می‌رسد که دومین نیازمندی شروع به ارضا شدن کرده باشد.
On	نقطه زمانی بعد از یک مجموعه از نیازمندیها که ارضا شده‌اند را شرح می‌دهد.
For	شرح می‌دهد که ویژگیهای اضافی نیازمندیهای دغدغه را کامل خواهند کرد.
With	شرح می‌دهد که یک شرط برای دو مجموعه از نیازمندیها با ارجاع به هر یک نگهداری خواهد شد.
In	شرح می‌دهد که یک شرط برای یک مجموعه از نیازمندیهای که ارضا شده‌اند نگهداری خواهد شد.
AND, OR, XOR	اجتماع، تفکیک و or انحصاری (زمانی که یکی از دو نیازمندی ارضا شده نه هر دو).

جدول ۴-۱۱: شرح کنش‌های خروجی

کنش خروجی	
نوع	*توضیحات*
Satisfied	برای تاکید اینکه یک مجموعه از نیازمندیهای دیدگاه ارضا خواهند شد بعد از اینکه محدودیت یک نیازمندی دغدغه اعمال شد، استفاده می‌شود.
Fulfilled	برای تاکید اینکه محدودیت یک نیازمندی دغدغه با موفقیت اعمال شده، استفاده می‌شود.

۴-۶-۳-۲- ترکیب اشتراکی

به منظور شناسائی نقاط مصالحه، مشاهده تعاملات یک دغدغه با دیگر دغدغه‌ها، همراه با ارجاع به تعدادی پایه، مورد نیاز است. این کاملا هم جهت با روشهای دو بعدی برای جداسازی دغدغه‌ها در مهندسی نیازمندیها است. نمونه‌هایی از این روشها، PREView [۳۷]، The NFR Framwork [۷۰] و مهندسی نیازمندیهای جنبه‌گرا [۳، ۶۰] است. در کلیه این روشها، نیازمندیهای وظیفه‌مندی به عنوان پایه‌ی از پیش تنظیم شده[1] برای مشاهده تعاملات بین دغدغه‌ها در جهت شناسائی نقاط بالقوه مصالحه مورد استفاده قرار می‌گیرند. در جداسازی چند بعدی یک چنین پایه از پیش تنظیم شده وجود ندارد و تمام دغدغه‌ها همتا بوده و یک نوع خاص از دغدغه‌ها بر دغدغه‌های دیگر ارجحیت یا برتری ندارد. از قوانین ترکیب ارائه شده در بخش ۴-۶-۳-۱ قابل مشاهده است که در جداسازی چند بعدی، نیازمندیهای وظیفه‌مندی (بازیابی اطلاعات) می‌توانند نیازمندیهای درون دغدغه‌های غیروظیفه‌مندی را محدود یا آنها را تحت تاثیر قرار دهند. بنابراین تحلیل مصالحه نیز باید ماهیتا چند بعدی باشد. روشهای کورکورانه[2] سعی دارند این عمل را با انتخاب کلیه ترکیبات ممکن از دغدغه‌ها، به عنوان پایه‌ای برای مطالعه تعاملات و سپس تلفیق نتایج تحلیل، انجام می‌دهند. با این وجود این نوع روشها یک سربار قابل توجهی دارند. زیرا که، در صورت وجود یک ابزار پشتیبانی باز هم تعداد ترکیب‌های دغدغه‌ها که به عنوان پایه در نظر گرفته می‌شوند بسیار زیاد است. بنابراین، بدین منظور مفهوم ترکیب اشتراکی (به وسیله $\cap$ مشخص می‌شود) معرفی می‌شود تا ترکیبات بالقوه از دغدغه‌ها را، که به عنوان پایه‌ای استفاده می‌شود برای ترکیب‌هایی که ارزش واقعی ارائه شدن به صورت تحلیل مصالحه در سطح نیازمندیها را دارند، محدود کند.

در نظر بگیرید $C_1, C_2, C_3, \ldots, C_n$ دغدغه‌های منسجم در نیازمندیهای سیستم هستند و $Sc_1, Sc_2, Sc_3, \ldots, Sc_n$ مجموعه‌ای از دغدغه‌هایی است که هر کدام مداخله‌ای هستند. حال فرض کنید می‌خواهیم مصالحه بین C_1

[1] Preset base
[2] Brute force

و C_2 شناسائی شود. بدین منظور، باید ترکیب اشتراکی Sc_1 وSc_2 انجام شود (یک ترکیب اشتراکی یک اشتراک ساده در تئوری مجموعه نیست). در نظر بگیرید که C_a یک عضوی از هر دوی Sc_1 و Sc_2 است. C_aدر ترکیب اشتراکی پدیدار خواهد شد اگر و فقط اگر C_1 و C_2 یک مجموعه یکسان یا مجموعه‌ای دارای هم پوشانی، از نیازمندیهای درون C_a را محدود یا تحت تاثیر قرار دهند. یعنی اینکه، اگر C_1 و C_2 مجموعه منفصل از هم را تحت تاثیر قرار دهند در این صورت C_a در ترکیب اشتراکی وجود نخواهد داشت.

اگر نتیجه ترکیب اشتراکی مجموعه تهی نباشد در این صورت باید تحلیل مصالحه و مشخص کردن اولویت‌ها انجام شود. این فرآیند برای C_1 و C_3 و الی تا C_1 و C_n تکرار می‌شود. سپس همین فرآیند برای C_2 انجام می‌شود. کاملا واضح است که دیگر نیازی برای انجام این عمل برای C_1 و C_2 وجود ندارد. زیرا آن هنگام مشخص کردن نقاط مصالحه برای C_1 انجام پذیرفته است. فرآیند برای تمام دغدغه‌ها تا رسیدن به C_n تکرار می‌شود. این نشان می‌دهد که ماکزیمم تعداد ترکیب اشتراکی که مجبور به انجام آن هستیم برابر است با : $C_n^2 = \frac{n!}{(n-2)!*2!}$ که n تعداد دغدغه‌ها در سیستم است. (به علت ساختار روش و درک بهتر در این بخش از نامهای لاتین استفاده خواهد شد)

به عنوان مثال، یک مشخصات نیازمندی با n دغدغه را در نظر بگیرید. مجموعه دغدغه‌هایی را که هر دغدغه آنها را میان‌بر می‌کند عبارتند از : $Sc_1=\{C_2, C_5, C_7, C_n\}$, $Sc_2=\{C_5, C_6, C_n\}$, $Sc_n=\{C_1, C_2, C_9\}$. در این جا نیازمندیهای که مشخص نشده‌اند بیان می‌کنند که آنها هیچ دغدغه‌ای را میان‌بر نمی‌کنند (ترکیب اشتراکی بین دو دغدغه C_i و C_j به صورت $Sc_i \cap Sc_j$ نمایش داده می‌شود). فهرستی از ترکیب اشتراکی ممکن برای مشخصات مذکور به این صورت خواهد بود: $Sc_1 \cap Sc_2= \{C_5, C_n\}$, $Sc_1 \cap Sc_n= \{C_2\}$, $Sc_2 \cap Sc_n= \phi$ این موضوع نشان می‌دهد که تعداد ترکیبات اشتراکی احتمالا کوچکتر از تعداد ترکیبات است. زیرا بعضی از دغدغه‌ها نمی‌توانند دغدغه‌های زیادی را تحت تاثیر قرار دهند (با آنها ارتباط داشته باشند). برای سیستم راهنمای توریست اگر $S_{InformationRetrieval}=\{$Csustomizability, Mobility, InformationUpdate, Navigation$\}$ مجموعه دغدغه‌هایی را مشخص کند که InformationRetrieval آنها را میان‌بر می‌کند و همچنین $S_{Mobility}= \{$Availability, Conncetivity, Context, Navigation, Informationretrieval$\}$ مجموعه مشابهی را برای Mobility مشخص کند. ترکیب اشتراکی میان این دو دغدغه به صورت $S_{InformationRetrieval} \cap S_{Mobility}= \{$Navigation$\}$ خواهد بود. زیرا قوانین ترکیب هر دو دغدغه، نیازمندی ۱ از Navigation را تحت تاثیر قرار می‌دهد. بنابراین ترکیب اشتراکی نقاط trad-off را برای تحلیل مصالحه مشخص می‌کند.

۴-۶-۳-۳- تحلیل مصالحه

مصالحه‌ها مبتنی بر نوع همکاری که یک دغدغه می‌تواند با دغدغه دیگر، با توجه به پایه شناسائی شده از طریق ترکیب اشتراکی داشته باشد، تحلیل می شوند. این همکاریها می‌توانند مثبت، منفی یا بی‌اثر باشند.
نخست، یک جدول همکاری ایجاد می‌شود (جدول ٤-١٢) که در آن هر دغدغه می‌تواند به صورت مثبت (+) یا منفی (-) با دیگر دغدغه‌ها ارتباط داشته باشد (یعنی همکاری داشته باشد). در این جدول سلول خالی حالت بی‌اثر را تداعی می کند. هر سلول علاوه بر مشخص کردن نوع همکاری، ترکیب اشتراکی استفاده شده برای پیدا کردن همکاری را نیز نشان می‌دهد. اگر هیچ همکاری در یکی از بعدهای رابطه وجود نداشته باشد در آن صورت، سلول فقط ترکیب اشتراکی را نشان می‌دهد. همچنین سلول خالی بیان کننده عدم وجود رابطه است. بنابراین، ارتباطات نشان داده شده در جدول ٤-١٢ دو طرفه هستند. به این معنی که، رابطه باید، در دو بعد، با توجه به پایه یکسان تحلیل شود. برای مثال، C_1 با C_2 به صورت مثبت با توجه به C_5, C_n همکاری دارد در حالی که C_2 با C_1 به صورت منفی با توجه به C_5, C_n همکاری دارد. با توجه به این موضوع، کاملا مشخص است که بعدهای همکاری از اهمیت بالائی برخوردار هستند. تصمیم‌گیری پیرامون نوع همکاری برای هر حالت خاص معمولا بسیار دشوار است مخصولا اگر، پایه یک مجموعه با چندین دغدغه باشد.

جدول ٤-١٢: همکاری بین دغدغه‌ها

	C_1	C_2	…	C_n
C_1		+ $\{C_5, C_n\}$		+ $\{C_2\}$
C_2	- $\{C_5, C_n\}$			
…				
C_4	+ $\{C_2\}$			

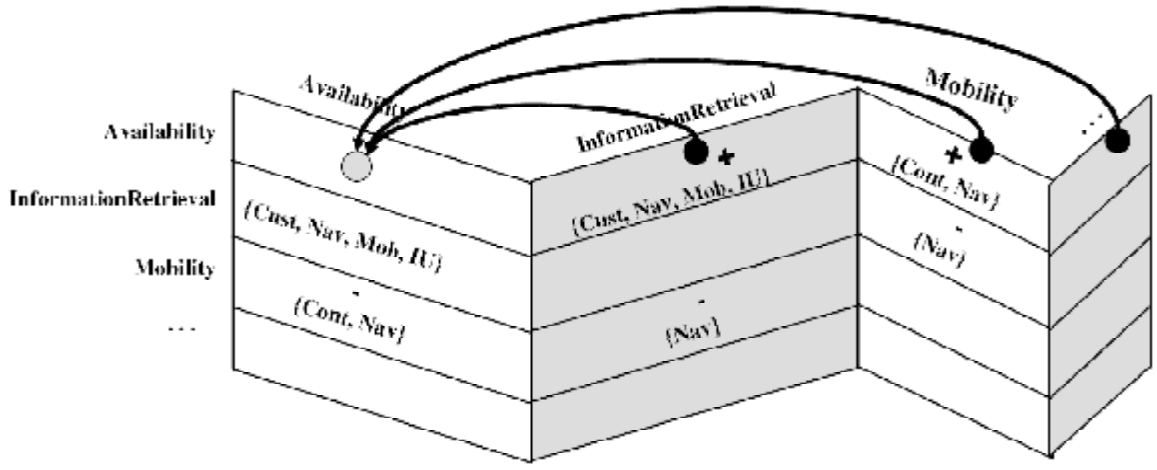
جدول ۴-۱۳ همکاری بین دغدغه‌های سیستم راهنمای توریست را نشان می‌دهد. در سیستم راهنمای توریست برای تحلیل مصالحه بین InformationRetrieval و Mobility ، باید همکاری بین آنها را با توجه به پایه Navigation بررسی کرد. در جدول ۴-۱۳ Mobility به صورت منفی با InformationRetrieval با توجه به پایه Navigation همکاری دارد. این بدین معنی است که هر چه قدر قابلیت حرکت توریست بیشتر شود مشکلات بیشتری در بازیابی اطلاعات از سیستم وجود خواهد داشت. همکاری در بعد مخالف نیز منفی است زیرا اگر اطلاعات پیچیده‌ی بیشتری مورد نیاز باشد قابلیت حرکت کمتر در سیستم باید وجود داشته باشد.

جدول ٤-١٣: بخشی از جدول همکاری برای سیستم راهنما

(**Cont**:Context;**Cust**:Customisability;**IU**:Information Update;**Mob**:Mobility; **Nav**:Navigability)

	Availability	**Information Retrieval**	**Mobility**	**…**
Availability		+ {Cust, Nav, Mob, IU}	+ {Cont, Nav}	
Information Retrieval	{Cust, Nav, Mob, IU}		- {Nav}	
Mobility	- {Cont, Nav}	- {Nav}		
…				

بعد از شناسائی نقاط مصالحه باید تاثیر یک مجموعه از دغدغه‌ها بر روی یک دغدغه مشخص، بررسی شود. در این مرحله به عمل مشخص کردن چگونگی تاثیر، project گفته می‌شود. فرض کنید P_1 یک مجموعه از دغدغه‌ها است که به صورت مثبت یا منفی با دغدغه C_1 همکاری دارد. در این صورت باید عمل project برای تمام دغدغه تا رسیدن به C_n تکرار شود. این فرآیند می‌تواند به وسیله برجسته کردن هر ستون متوالی بهتر تشریح شود تا تاثیر یکجا[1] برای موقعیت‌هایی که چندین دغدغه مستقیما یک دغدغه خاص را تحت تاثیر قرار می‌دهند، بدست آید (شکل ۴-۲۰). در نظر بگیرید که دغدغه‌ها در ستون‌ها و سطرها تکرار شده‌اند. تاثیر یکجا می‌تواند در هر دو بعد مشاهده شود. به عنوان مثال، از Availability تا InformationRetrieval و از InformationRetrieval تا Availability . این برجسته سازی Projection یکجا فراهم می‌کند. یعنی تاثیرهای ترکیب شده یک مجموعه از دغدغه‌ها روی دغدغه خاص.



شکل ۴-۲۰: جدول همکاری برجسته شده همراه با ستونهایش

[1] Cumulative affect

برای مثال، تاثیر یکجا از InformationRetrieval و Mobility روی Availability مثبت است. بنابراین مصالحه‌ای مورد نیاز نیست. در طرف دیگر، تاثیر یکجا Availability و InformationRetrieval روی Mobility منفی است. بنابراین یک مصالحه باید در جهت کنترل کردن این وضعت تداخل انجام شود. در [۶۰] یک راه پیشنهادی برای این امر وجود دارد. ایده اصلی، خصوصیت وزن‌دهی به آن دغدغه‌هایی است که همکاری منفی با دیگر دغدغه‌ها در ارتباط با یک پایه خاص دارند. هر وزن یک عدد اشاری در بازه [۱ .. ۰] است که اولویت یک دغدغه در ارتباط با دغدغه‌ای که project کرده را نشان می‌دهد. این مقادیر بر حسب اهمیتی که هر دغدغه در ارتباط با دیگری دارد تعیین می‌شود. مقیاس استفاده شده مبتنی بر منطق فازی است که معانی زیر را دارد:

- خیلی مهم، یک مقدار در بازه [۱.. ۰/۸]
- مهم، یک مقدار در بازه [۰/۸ .. ۰/۵]
- متوسط، یک مقدار در بازه [۰/۵ .. ۰/۳]
- نه خیلی مهم، [۰/۳ .. ۰/۱]
- بی اهمیت، یک مقدار در بازه [۰/۱ .. ۰]

استفاده از مقادیر فازی (خیلی مهم، مهم، نه خیلی مهم، و غیره) عمل تخصیص اولویت‌دهی برای حل تداخل‌ها را آسان می‌کند. مشکل اصلی زمانی اتفاق می‌افتد که دو دغدغه که در یک تداخل قرار گرفته‌اند اولویت یکسان داشته باشند. در این حالت تصمیم گیری باید توسط ذینفعان صورت پذیرد تا تعیین شود کدام یک باید ارزش کمتری داشته باشد. حل کردن تداخل‌ها ممکن است منجر به بازنگری مشخصات نیازمندی شود. اگر این حالت اتفاق بیفتد projection بهبود پیدا کرده و هر تداخل که ایجاد شده باید مجددا حل شود. این چرخه ادامه پیدا می کند تا کلیه تداخل‌ها از طریق مذاکرات موثر حل شوند.

۴-۶-۴- انتخاب‌های معماری

بعد از تحلیل نقاط مصالحه در دغدغه‌های سطح نیازمندی، باید قابلیت فهم آنها را زمانی که انتخاب‌های معماری انجام می‌شود بهبود داد. قبل از اینکه به بحث چگونگی تبدیل شدن مصالحه‌های شناسائی شده به انتخاب معماری در بعدها مختلف پرداخته شود خیلی مهم است که بیان شود که هر دغدغه در جداسازی چند بعدی منجر به یک تعداد از انتخاب‌های معماری می‌شود که می‌تواند نیازمندیها را در سطوح مختلف رضایت ذینفعان[1] نشان دهد (شکل ۴-۲۱). جعبه‌های خاکستری در شکل ۴-۲۱ انتخاب‌های معماری مناسب و ایده‌ال را

[1] Satisfaction

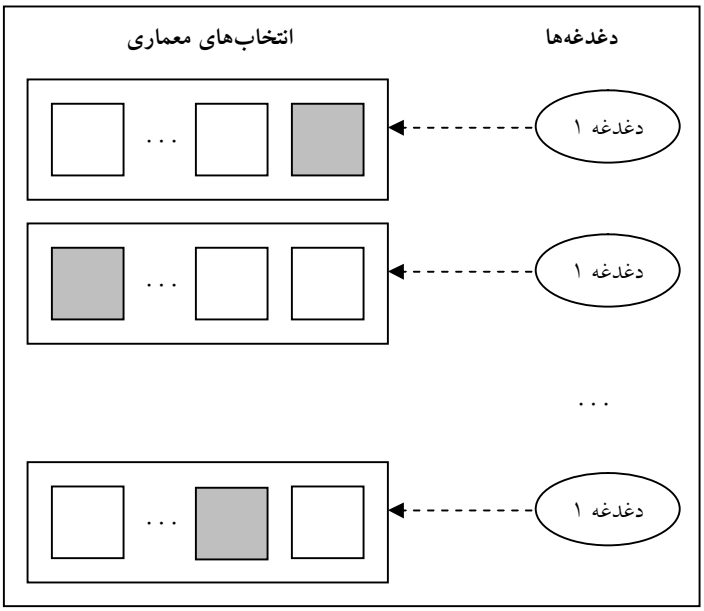
برای هر دغدغه نشان می‌دهند. توجه شود که احتما اینکه این تصمیمات معماری یکسان باشند بسیار ضعیف است و حتی ممکن است تداخل‌هایی نیز وجود داشته باشد (در اغلب موارد این حالت رخ می‌دهد).

تحلیل مصالحه انجام شده در سطح نیازمندی این نوع تداخل‌های بالقوه را اطلاع داده و در این مرحله می‌توان آنها را در هنگام انتخاب معماری به وضوع مشاهده و درک کرد. این موضوع اهمیت تحلیل مصالحه زود هنگام (اولیه)، که در سطح نیازمندی انجام شد و بعضی از این تداخل‌ها را از طریق مذاکره و اولویت‌دهی دغدغه‌ها حل کرد را مشخص می‌کند. برای تشریح این مرحله دوباره سیستم راهنمای توریست را مورد تحلیل قرار می‌دهیم. دغدغه قابلیت دسترسی می‌تواند به وسیله تعدادی انتخاب معماری برآورده شود.

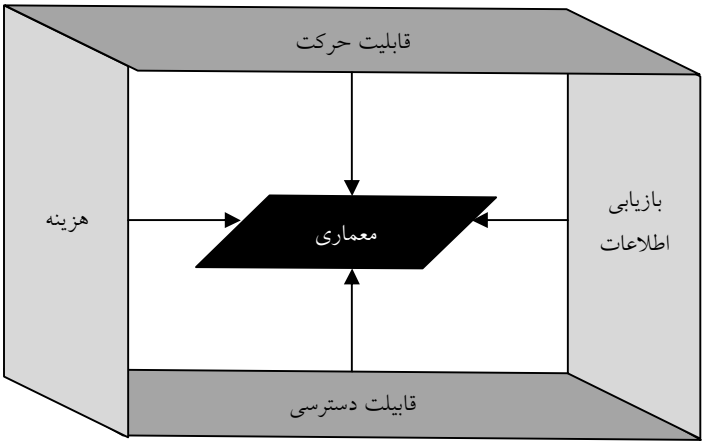
به عنوان مثال از طریق چندین نسخه داشتن سرویس دهنده، اتصال شبکه یکپارچه قوی، یا ترکیبی از این دو حالت. در هر لحظه قابل مشاهده است که دغدغه قابلیت حرکت نیاز به یک معماری مبتنی بر شبکه بی‌سیم دارد به صورتی که توریست بتوان حرکت کند و همزمان به اطلاعات نیز دستری داشته باشد. این موضوع با نیاز معماری برای قابیلت دسترسی مغایرت دارد (تحلیل نیازمندی قبلا پیرامون همکاری منفی قابلیت دسترسی و قابلیت حرکت اطلاعاتی داده بود) زیرا که شبکه‌های بی‌سیم قابلیت اطمینان کمتری نسبت به شبکه‌های سیمی سنتی دارند. نگهداری قابلیت دسترسی در شبکه‌های بی‌سیم نیازمندی یک تعداد اندکی ایستگاه‌های پایه بی‌سیم قدرتمند در یک حوزه محدود است. با این وجود این می‌تواند منجر به قابلیت حرکت شود که هدفش اجازه دادن به کاربر است که بتواند آزادانه حرکت کند.

برای دیگر دغدغه‌های سیستم نیز یک چنین حالتی را می‌توان در نظر گرفت. برای نمونه بازیابی اطلاعات را در نظر بگیرید. انتخاب‌های معماری مختلف، وابسته به انواع رسانه‌ها قابل دسترس برای بیننده دستگاه است. این می‌تواند بازیابی مبتنی بر متن ساده، مکانیزم بازیابی چندرسانه شامل عکس، نقشه تصویری، انیمیش، ویدئو و غیره باشد. در بخش ۴-۶-۳-۳ مشاهده شد که قابلیت حرکت و بازیابی اطلاعات، همکاری منفی با یکدیگر دارند همچنین ذکر شد که این همکاری منفی با توجه به راهبری است.

هزینه یک دغدغه‌ای است که احتمالا با دیگر انتخاب‌های معماری تداخل داشته و اغلب نقش مهمی در تعیین معماری نهایی سیستم دارد. در سیستم راهنمای توریست، انتخاب انواع و تعداد مختلف سرویس‌دهنده‌های چند نسخه‌ای، ایستگاه‌های پایه بی‌سیم، میزان محتوای چندرسانه‌ای توسعه داده شده و غیره به شدت توسط دغدغه هزینه محدود و تحت تاثیر قرار می‌گیرند. از دید هزینه، بهترین انتخاب کم هزینه‌ترین است اما احتمالا این انتخاب مناسب‌ترین انتخاب برای پشتیبانی از دیگر دغدغه‌ها نیست. نهایتا اینکه کلیه عوامل باعث می‌شود که انتخاب‌های معماری به ابعاد مختلف کشیده شود (شکل ۴-۲۲).



شکل ۴-۲۱: انتخاب‌های معماری برای برآوره کردن هر دغدغه



شکل ۴-۲۲: بیرون کشیدن معماری از دغدغه‌های مختلف

۴-۷- مقایسه روشها

تا به اینجا با معرفی چندین روش برجسته در زمینه مهندسی نیازمندیهای جنبه‌گرا توانستیم دید کلی از اهداف اصلی و نحوه عملکرد آن بدست آوریم. اما آنچه که مشخص است اینکه هر کدام از روشها از ساختار و الگوهای خاص برای دستیابی به اهداف استفاده می‌کند و این امر باعث می‌شود که روشهای مختلف در سطوح متفاوت، از اهداف مهندسی نیازمندیهای جنبه‌گرا حمایت کنند. حتی چه بسا به دلیل استفاده از یک ساختار مناسب و جدید موضوعاتی را پوشش دهند که روشهای دیگر در ماهیت خود آن را ندارند.

با توجه به این موضوع، نیاز است که اهداف و موضوعاتی (معیارها) که یک روش مهندسی نیازمندیهای جنبه‌گرا باید از آن حمایت کند یا دارای آن باشد را مشخص کرده و سپس میزان موفقیت و پشتیبانی هر روش را از معیارهای مشخص شده مورد بررسی قرار بگیرد. البته مشکلات خاصی در این مورد وجود دارد زیرا با توجه به اینکه حوزه مهندسی نیازمندیهای جنبه‌گرا جدید است و یک سری اهداف اولیه برای آن مشخص است اهداف ثانویه آن به مرور در قالب روشهای جدید مطرح می‌شوند و لذا انسجام و اطمینان در معیارهای جدید وجود ندارد. همچنین یک سری اهداف نیز هستند که از ماهیت توسعه نرم‌افزاری نشات می‌گیرند از قبیل قابلیت استفاده، ردیابی و ... .

۴-۷-۱- معیارهای مقایسه

معیارهای مقایسه برای روشهای مهندسی نیازمندیهای جنبه‌گرا به صورت زیر هستند:

۱. **شناسائی دغدغه‌ها**: در این معیار هدف آن است که مشخص شود که آیا روش می‌تواند نیازمندیهای وظیفه‌مندی و غیروظیفه‌مندی را با استفاده از تکنیک‌های خاص استخراج کند.
۲. **جداسازی دغدغه‌ها**[1]: هدف بررسی این فاکتور است که آیا دغدغه‌های مداخله‌ای و هسته‌ای از یکدیگر مجزا شده‌اند یا نه؟
۳. **نمایش دغدغه‌ها**[2]: هدف نمایش دغدغه‌ها در قالب موارد کاربری، گرافهای هدف، لیست ویژگیها[3]، نمایش‌های گرافیکی، قالب‌ها[4] یا مشخصات XML یا سایر قالب‌ها است.
٤. **ترکیب دغدغه‌ها**[5]: هدف ترکیب دغدغه‌های مداخله‌ای با دغدغه‌های هسته است.

[1] Separate concerns
[2] Represent concerns
[3] Feature lists
[4] Templates
[5] Compose concerns

۵. **تفکیک تداخل‌ها** : هدف تفکیک دغدغه‌هایی است که همکاری منفی با یکدیگر دارند همچنین باید تداخل‌ها حاصل از آنها حل شود زیرا همکاریهای منفی بین دغدغه‌ها یک تاثیر بد روی کل ترکیب دغدغه‌ها و سیستم دارد و همچنین منجر به معماری ضعیف می‌شود.

٦. **پشتیبانی ابزار**: این معیار بر روی این هدف متمرکز است که آیا روش مهندسی نیازمندیهای جنبه‌گرا توسط یک ابزار پشتیبانی می‌شود یا نه؟ زیرا خود حمایت ابزار امکانات زیادی را به همراه می‌آورد.

۷. **تاثیر و نگاشت جنبه‌ها**: هدف مشخص کردن این است که یک جنبه به چه چیز نگاشت شده و یا چه تاثیری بر سایر فازهای توسعه دارد.

۸. **وابستگی به یک زبان تعریف یا پیاده‌سازی**: هدف بررسی این موضوع است که آیا روش مبتنی بر یک زبان خاص است یا نه؟ یعنی اینکه بدون آن زبان، امکان استفاده از روش وجود دارد یا نه؟

۹. **دغدغه نقش رده اول**: این معیار مشخص کننده این است که آیا روش، دغدغه را رده اول خود در نظر گرفته است یا نه؟

۱۰. **ارتباط دو جهته**: این معیار بیان کننده این موضوع است که آیا روش یک قابلیت حرکت دو طرفه یعنی از ورودی به خروجی و از خروجی به ورودی را فراهم می‌کند یا نه؟

۱۱. **فراهم کردن مستندات کافی**: بیان کننده این فاکتور است که آیا روش، مستندات لازم و مناسب را تولید می‌کند یا نه؟

۱۲. **بیان قوانین ترکیب**: این معیار مشخص کننده این است که آیا روش ساختار یا زبانی را برای تعریف قوانین ترکیب فراهم می‌کند یا نه؟

۱۳. **قابلیت ردیابی**: این فاکتور مشخص کننده این است که آیا روش مکانیزمی برای ردیابی جنبه‌های شناخته شده در فرآیند توسعه فراهم می‌کند یا نه؟

١٤. **قابلیت رسمی‌سازی**: این فاکتور بیان کننده این است که آیا روش قابلیت رسمی شدن را دارد. به عبارت بهتر آیا روش رسمی است یا نه؟ (این فاکتور جزء اهداف مهندسی نیازمندیهای جنبه‌گرا نیست ولی برای مقایسه منسجم‌تر در این بخش بیان شده است).

۴-۷-۲- مقایسه اجمالی

اعمال معیارهای مقایسه بر روی چندین روش و مقایسه آنها مستلزم آن است که روشها بر روی یک پروژه واحد اعمال شده و خروجی آنها مبتنی بر معیارها بررسی شود و مشخص شود که هر کدام تا چه حدی عمل پشتیبانی را انجام می‌دهند. ولی با توجه به اینکه مهندسی نیازمندیهای جنبه‌گرا در دروان جوانی خود است

(کارهای تحقیقاتی آن انجام می‌شود) و اعمال آن بر روی یک پروژه واقعی با سختیهای فراوانی همراه است (زیرا هیچ انسجامی بر خروجی آن وجود ندارد) بنابراین در این تحقیق، معیارها بر اساس نتایج تحقیقاتی مقایسه می‌شود که نویسندگان آن را بر روی یک مطالعه موردی ادعا کرده‌اند.

این مقایسه فقط در قالبی خواهد بود که مشخص شود که آیا روش از معیار مورد نظر حمایت می‌کند یا نه. کمیت و کیفیت آنها بدلیل اعمال نشدن آن در یک پروژه واحد زیاد قابل تصمیم‌گیری نیست. همچنین یک بررسی تحلیلی نیز بر روی دو روش خواهیم داشت که می‌تواند کمبودهای موجود را مشخص کند. مقایسه روشها در جدول ٤-١٤ نمایش داده شده است. روشهایی که با توجه به معیارها مقایسه می‌شوند عبارتند از:

- مدل عمومی مهندسی نیازمندیهای جنبه‌گرا (AORE Model)
- مدل بهبود یافته مدل عمومی مهندسی نیازمندیهای جنبه‌گرا (AORE with ARCaDe)
- روش COSMOS
- روش Theme/Doc
- روش جداسازی چند بعدی در مهندسی نیازمندیهای جنبه‌گرا (MDSoC)

جدول ٤-١٤: مقایسه روشهای برجسته مهندسی نیازمندیهای جنبه‌گرا

AORE Model	AORE with ARCaDe	COSMOS	Theme/Doc	MDSoC	
×	×	×	×	×	شناسائی دغدغه‌ها
×	×	×	×	×	جداسازی دغدغه‌ها
×	×	×	×	×	نمایش دغدغه‌ها
×	×	×	×	×	ترکیب دغدغه‌ها
×	×	–	–	×	تفکیک تداخل‌ها
–	×	–	×	×	پشتیبانی ابزار
×	×	–	–	×	تاثیر و نگاشت جنبه‌ها
–	×	–	×	×	وابستگی به یک زبان تعریف
–	–	×	–	×	دغدغه نقش رده اول
–	–	×	–	–	ارتباط دو جهته
–	×	×	–	×	فراهم کردن مستندات کافی
×	×	–	–	×	بیان قوانین ترکیب
–	×	×	–	×	قابلیت ردیابی
–	–	–	–	–	قابلیت رسمی‌سازی

(علامت × به معنای دارد و علامت – به معنای ندارد)

در روش AORE Model، شروع روش مبتنی بر شناسائی دغدغه‌ها و نیازمندیها است و در آن به نوعی تاکیدی بر اینکه کدام اجرا شود وجود ندارد. اما آنچه که در مبحث جنبه‌گرائی مهم است اینکه هر روش جنبه‌گرا باید دغدغه را رده اول خود قرار دهد و سایر فعالیت را مبتنی بر آن انجام دهد. بنابراین لازم است که در این روش ابتدا دغدغه‌ها شناسائی شده سپس نیازمندیها کشف شده و به دغدغه‌ها انتساب شوند.

در روش AORE with ARCaDe دغدغه یک نیازمندی غیروظیفه‌مندی در نظر گرفته شده است ولی یک دغدغه می‌تواند یک نیازمندی وظیفه‌مندی نیز باشد. در نتیجه با توجه به این مدل یک نیازمندی غیروظیفه‌مندی می‌تواند یک جنبه نامزد در نظر گرفته شود که این موضوع صحیح نیست. زیرا یک نیازمندی وظیفه‌مندی نیز می‌تواند به عنوان یک جنبه در نظر گرفته شود. مانند رزرواسیون در سیستم مدیریت هتل [۸]. نکته دیگر مشخص کردن نگاشت و تاثیر جنبه‌ها است. یعنی اینکه آیا این نگاشت و تاثیری که برای جنبه‌ها مشخص می‌شود پایدار هستند یا نه؟ این روش یک فرآیند دقیق و کامل برای ترکیب دغدغه‌ها با یکدیگر پیشنهاد نمی‌کند ولی با کنشها و عملگرها مشخص می‌کند چگونه نیازمندیهای جنبه‌ای رفتارهای یک مجموعه از نیازمندیهای غیرجنبه‌ای را تحت تاثیر قرار می‌دهند.

۴-۸- خلاصه و نتیجه‌گیری

در این فصل به بررسی پنج روش برجسته در زمینه مهندسی نیازمندیهای جنبه‌گرا پرداخته شد. هر روش به صورت کامل تشریح شد و در صورت لزوم مطالعه موردی برای آن مطرح شد تا روش به صورت کامل تشریح شود. در هنگام بررسی هر روش سعی شد که مزایایی که از آن ناشی می‌شوند مشخص شوند. معایب روشها نیز تا حدودی مطرح شدند البته دلیل ذکر معایب آنها به صورت جزئی، نو بودن اهداف و عدم بکارگیری آنها در صنعت است. در بررسی روشها، هر کدام علاوه بر دارا بودن دو هدف اصلی مهندسی نیازمندیهای جنبه‌گرا، اهداف دیگری را نیز معرفی کردند که هم خاصیت کمی و هم خاصیت کیفیتی داشتند. نهایتا یک مقایسه اجمالی با توجه به معیارهای مشخص شده بر روی آنها انجام شد. البته این مقایسه بیشتر به جای اینکه به بررسی کیفیت پشتیبانی معیارها بپردازد به این موضوع پرداخت که آیا روش مورد نظر از معیار حمایت می‌کند یا نه (میزان حمایت مد نظر نبود). با توجه به مقایسه اجمالی انجام شده در جدول ٤-١٤ مشخص است که فاکتورهای ارتباط دو جهته، دغدغه نقش رده اول و قابلیت رسمی سازی از جمله فاکتورهایی هستند که توسط کاملترین روشها زیاد مورد حمایت نیستند و به عنوان نقاط ضعف و کمبودی برای آنها به شمار می‌روند. البته سایر فاکتورها نیز در برخی روشها پشتیبانی نمی‌شود که تحقق آنها می‌تواند باعث منسجم‌تر شدن و موفقیت آن روشها شود.

فصل پنجم

روش پیشنهادی برای شناسائی جنبه‌ها

در این فصل، هدف معرفی روش پیشنهادی برای شناسائی جنبه‌ها است. بدین منظور ابتدا یک دید کلی از روش پیشنهادی که مبتنی بر شبکه پتری است ارائه می‌شود. سپس عناصر پایه (مفاهیم اصلی) روش پیشنهادی تعریف و کاربرد هر یک تشریح می‌شود. نحوه اجرا روش که متشکل از هشت مرحله است گام به گام تشریح شده و نهایتا نیز مزایا و معایب روش پیشنهادی ذکر می‌شود.

۵-۱- دید کلی

با توجه به مطالب ذکر شده در فصول یک و چهار، شناسائی جنبه‌ها در مراحل اولیه‌ی (مهندسی نیازمندیهای جنبه‌گرا) توسعه از اهمیت بالائی برخوردار است و از طرفی مشخص کردن یک روش به صورت رسمی می‌تواند نقش بسیار مهمی در موفقیت و درستی نتایج حاصل از بکارگیری آن داشته باشد. بدین منظور، در این فصل یک روش مبتنی بر شبکه پتری برای شناسائی جنبه‌ها ارائه می‌شود که با پیروی از گامهای مشخص شده در آن می‌توان به جنبه‌ها دست پیدا کرد. این روش پیشنهادی که گام اول خود را با شناسائی دغدغه‌ها شروع می‌کند متشکل از هشت گام است که اجرا کلیه این گامها منجر به شناسائی جنبه‌ها می‌شود. گامها عبارتند از:

- گام اول، شناسائی دغدغه‌ها
- گام دوم، شناسائی و مشخص کردن نیازمندیها
- گام سوم، ایجاد (ساخت) شبکه نیازمندیها[۱]
- گام چهارم، مشخص کردن ترتیب اجراها[۲]
- گام پنجم، ایجاد شبکه دغدغه‌ها[۳]
- گام ششم، شناسائی وابستگی‌ها، محدودیت‌ها و رابطه‌ها
- گام هفتم، شناسائی دغدغه‌های نامزد[۴] جنبه شدن (اجرا مدل)
- گام هشتم، مشخص کردن موجودیت‌های منطقی[۵]

در جهت تحقق روش، سه تعریف پایه مطرح شده است که عبارتند از شبکه دغدغه، شبکه نیازمندی و ترتیب اجرا. این سه مفهوم به همراه وابستگی‌ها، محدودیت‌ها و رابطه‌های حاصل از ماهیت مسئله منجر به تشکیل یک مدل مبتنی بر شبکه پتری برای سیستم نهائی می‌شوند. سیستم نهائی که با استفاده از شبکه پتری مدل شده، اجرا می‌شود و خروجی حاصل می‌دهد که در آن دغدغه‌هایی مشخص شده‌اند که احتمال جنبه شدن را دارند. سپس با شناسائی موجودیت‌های منطقی نیازمندیهای متعلق به دغدغه‌های مشخص شده در خروجی مدل نهائی، جنبه‌ها حاصل می‌شود.

این روش پیشنهادی بیشتر تمرکزش بر روی سیستماتیک کردن (رسمی‌سازی) شناسائی جنبه‌ها است نه اینکه سعی در شناسائی جنبه‌هایی کند که به نوعی کشف آنها در سیستم مشکل است. برای اجرا روش پیشنهادی بر

[1] Requirements Net
[2] Executions Order
[3] Concerns Net
[4] Candidate
[5] Logical entities

روی یک سیستم می‌توان از ابزار استفاده کرد. ابزار مورد استفاده برای روش پیشنهادی باید قابلیت مدلسازی[1]، شبیه‌سازی[2] و مانیتور کردن[3] شبکه پتری را داشته باشد (ابزاری مورد استفاده در این تحقیق CPN/Tools است). زیرا تا زمانی که مدل نهائی سیستم مورد نظر اجرا نشود و گذرهای موجود در شبکه پتری سیستم مانیتور نشده و خروجی حاصل نشود نمی‌توان نظری در مورد دغدغه‌ها داد که آیا آنها جنبه هستند یا نه. ولی آنچه که مشهود و مشخص است اینکه، جنبه‌های شناسائی در این روش حتما یک دغدغه مداخله‌ای هستند. اما نکته قابل تأمل این است که چرا شبکه پتری در روش پیشنهادی برای شناسائی جنبه‌ها استفاده شده است. دلایل استفاده به شرح زیر هستند:

- شبکه پتری یک زبان مدلسازی قابل اجرا است. این قابلیت شبکه پتری باعث می‌شود که درک و فهم روش شناسائی جنبه‌ها آسانتر شده و بتوان جنبه‌های سیستم را از طریق اجرا مدلسازی انجام شده برای سیستم مورد نظر بدست آورد.
- شبکه پتری یک زبان مدلسازی رسمی است که باعث می‌شود روش پیشنهادی نیز از قابلیت رسمی بودن برخوردار شود (البته می‌توان از مفاهیم پایه رسمی نیز برای این روش استفاده کرد ولی کارائی و قابلیت درک لازم را نخواهند داشت).
- بین شبکه پتری و UML نگاشتی وجود دارد که استفاده از شبکه پتری باعث می‌شود در زبان UML نیز چنین ساختارهایی حاصل شود (روش پیشنهادی در UML نیز قابل بکارگیری باشد).
- شبکه پتری برای مدلسازی و اجرا خود ابزارهای بسیار سودمندی دارد که این امر می‌تواند نیاز روش پیشنهادی به یک ابزار را برای پیاده‌سازی ساختار و منطق خود تامین کند.

۵-۲- تعاریف پایه روش

برای شناسائی جنبه‌ها با استفاده از شبکه پتری نیاز است تعاریف پایه‌ای مبتنی بر شبکه پتری وجود داشته باشند تا بتوان با استفاده از آنها مدل نهائی سیستم مورد انتظار را رسم کرد. این تعاریف، در واقع نگاشتی از تعاریف غیررسمی به رسمی، دغدغه، نیازمندی و ترتیب اجرا شدن نیازمندیها برای تحقق دغدغه‌ها هستند. نحوه نگاشت دادن تعاریف غیررسمی به رسمی تاثیر وافری در عملکرد صحیح و قابلیت گسترش روش دارد. به منظور تعریف این مفاهیم با استفاده از شبکه پتری، یک تعریف رسمی ساده از شبکه پتری در بخش بعدی ارائه می‌شود (حالت ساده شده تعریف ارائه شده در بخش ۲-۱۴-۳ است) که سایر تعاریف نیز به نوعی مبتنی بر

[1] Modeling
[2] Simulation
[3] Monitoring

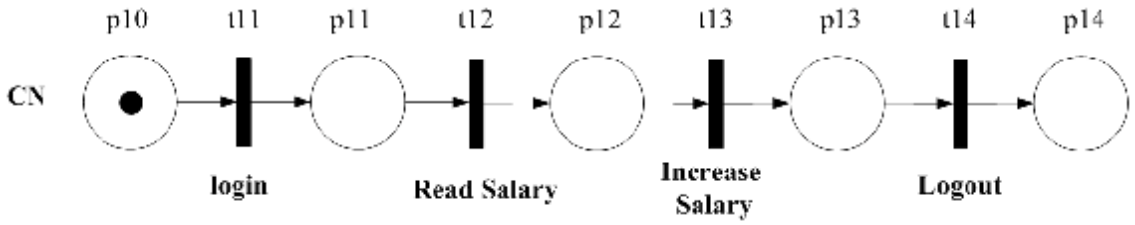
این تعریف شبکه پتری حاصل خواهند شد. این تعریف مطرح شده فقط برای تشریح نحوه عملکرد روش استفاده می‌شود و در عمل (شبیه‌ساز) باید از شبکه پتری رنگی استفاده شود.

۵-۲-۱- شبکه پتری

شبکه پتری یک تاپل سه تائی $PN= (P, T, F)$ است که:

- P یک مجموعه متناهی از مکان‌ها است.
- T یک مجموعه متناهی از گذرها است و اشتراک آن با P تهی است ($P \cap T= \phi$).
- $F \subseteq (P \times T) \cup (T \times P)$ یک مجموعه از کمان‌ها است.

در [۷۱]، تفسیر عمومی از مکان‌ها و گذرها ارائه شده است. در این مقاله یک گذر (رویداد)، یک تعداد مشخص از ورودی‌ها و خروجی‌ها از مکان‌ها دارد که پیش شرط‌ها و پس شرط‌های آن را مشخص می‌کنند. یک نشانه در یک مکان نشان دهنده این است که شرایط مربوط به آن مکان ارزش درست را دارند. با توجه به مفاهیم ذکر شده می‌توان هر سیستم را با استفاده از شبکه پتری مدلسازی کرد. به عنوان مثال یک سری از عملیات ترتیبی در شکل ۵-۱ به وسیله شبکه پتری نمایش داده شده است.



شکل ۵-۱: یک شبکه پتری برای سیستم مدیریت پرسنل

۵-۲-۲- شبکه نیازمندی[1]

نیازمندی عبارت است از مواردی که سیستم باید دارای آن باشد. یک نیازمندی در دانه‌بندی درشت می‌تواند متشکل از چندین نیازمندی با دانه‌بندی ریز باشد ولی در پائین‌ترین سطح، متشکل از موجودیت‌های منطقی است که این موجودیت‌های منطقی کوچکترین جزء را تشکیل می‌دهند. بنابراین یک تعریف رسمی مبتنی بر شبکه پتری برای نیازمندی به شرح زیر است.

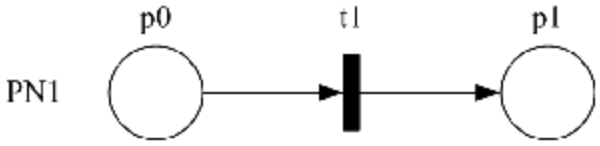
[1] Requirement Net (RN)

تعریف رسمی شبکه نیازمندی : شبکه نیازمندی یک تاپل دوتائی (RN= (PN, LE[1] است که:

- PN یک شبکه پتری با |P|=2, |T|=1, |F|=2 است.
- LE= (O_1, O_2, …, O_n) یک مجموعه از موجودیت‌های منطقی است. موجودیت‌های منطقی می‌توانند کلاس، پایگاه داده و ... در نظر گرفته شوند.

به عنوان یک مثال از شبکه نیازمندی، شکل ۵-۲ را در نظر بگیرید. شکل ۵-۲ یک شبکه نیازمندی به نام RN_1 را نشان می‌دهد. شبکه نیازمندی RN_1 از یک شبکه پتری به نام PN_1 و یک مجموعه از موجودیت‌های منطقی به نام LE_1 تشکیل شده است. LE_1 نیز از سه موجودیت منطقی به نام‌های O_1, O_2, O_3 تشکیل شده است. از آنجائی که مفهوم شیء[2] به نوعی مشخص کننده یک موجودیت مستقل است (فایل، پایگاه‌داده، کلاس) در مثالهای مطرح شده برای تشریح تعاریف، یک موجودیت منطقی، یک شیء در نظر گفته می‌شود.

هر شبکه نیازمندی که تعریف می‌شود مستقل بوده و می‌تواند مشخصاتش تغییر پیدا کند بدون آنکه تاثیری بر دیگر شبکه‌های نیازمندی وارد کند. در واقع همین قابلیت مستقل تعریف شدن شبکه نیازمندی باعث می‌شود که روش پیشنهادی برای شناسائی جنبه‌ها دارای قابلیت توسعه افزایشی[3] شود.



RN_1= (PN_1, LE_1)
LE_1= (O_1, O_2, O_3)

شکل ۵-۲: یک مثال برای شبکه نیازمندی

۵-۲-۳- ترتیب اجرا[4]

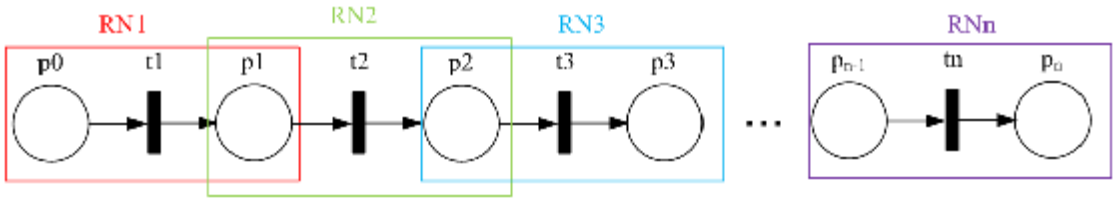
یک دغدغه متشکل از چندین نیازمندی است که ترتیبی از تحقق نیازمندیها منجر به تحقق دغدغه می‌شود. در حالت‌هایی ممکن است که تحقق یک دغدغه ناشی از تحقق نیازمندیها با ترتیب‌های مختلف باشد یعنی اینکه یک دغدغه چندین ترتیب اجرا شدن برای نیازمندیهای خود داشته باشد تا اینکه بتواند کاملا تحقق پیدا کند.

[1] Logical Entity (LE)
[2] Object
[3] Incremental
[4] Execution Order (EO)

بنابراین یک نیازمندی ممکن است در یک ترتیب اجرا و گاها در چندین ترتیب اجرا شرکت داشته باشد تا اینکه در نهایت آن دغدغه تحقق پیدا کند. با نگاشت این مطلب به تعریف رسمی نیازمندی (شبکه نیازمندی)، یک ترتیب اجرا به صورت زیر تعریف می‌شود.

تعریف ترتیب اجرا : یک ترتیب اجرا دربرگیرنده یک تعداد شبکه نیازمندیهای متعلق به یک شبکه دغدغه است که به صورت منظم پشت سر هم اجرا می شوند.

نحوه نمایش یک ترتیب اجرا به صورت $EO= (RN_1, RN_2, \ldots, RN_n)$ است که در شکل ۵-۳ به تصویر کشیده شده است



شکل ۵-۳: نحوه ترکیب شبکه نیازمندیها در یک ترتیب اجرا

ساختار مجموعه EO که متشکل از شبکه نیازمندیها است دارای ترتیب است یعنی در این مجموعه، شبکه نیازمندی RN_1 اول از همه‌ی شبکه‌های نیازمندی باید اجرا شود و شبکه نیازمندی RN_n نیز باید آخرین شبکه نیازمندی باشد که اجرا می‌شود. در واقع بعد از اجرا شدن RN_n است که یک دغدغه (یا بخشی از یک دغدغه) تحقق پیدا می‌کند. به عنوان یک مثال، دغدغه‌ای را در نظر بگیرید که چهار شبکه نیازمندی به نامهای RN_1, RN_2, RN_3, RN_4 دارد و تحقق این دغدغه از اجرای دو ترتیب اجرا ناشی می‌شود که این ترتیب اجراها به صورت زیر هستند:

$$EO_1 = (RN_1, RN_2, RN_4) \qquad EO_2= (RN_1, RN_3, RN_4)$$

همانطور که مشخص است دو تا از شبکه نیازمندیهای RN_1, RN_4 در هر دو ترتیب اجرا در نظر گرفته شده‌اند و مابقی شبکه نیازمندیها، هر کدام در یک ترتیب اجرا ذکر شده‌اند. این نوع اشتراکات در ترتیب‌های اجرا ناشی از این است که ممکن است یک دغدغه چندین حالت را در ماهیت خود داشته باشد و بعضی از این حالتها، به یک یا چند نیازمندیهای یکسان نیاز داشته باشند. اشتراک در شبکه نیازمندیهای متعلق به ترتیب اجرا بیشتر در شبکه نیازمندیهای اول و شبکه نیازمندیهای آخر اتفاق می‌افتد.

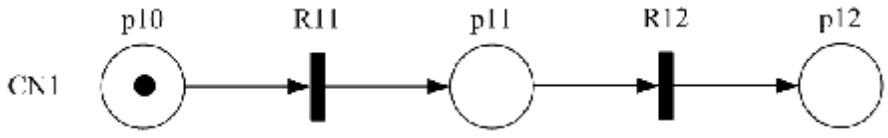
۵-۲-۴- شبکه دغدغه

همان طور که در بخش ۲-۲ تعریف شد دغدغه عبارت است از یک یا چند نیازمندی وابسته به ذینفعان که قابلیت پیاده‌سازی با یک ساختار داده را دارد. بنابراین یک تعریف رسمی مبتنی بر شبکه پتری برای دغدغه به شرح زیر است.

تعریف رسمی شبکه دغدغه : شبکه دغدغه یک تاپل دوتائی (SoR^1, SoE^2) =CN است که:

- SoR= (RN_1, RN_2, …, RN_n), (n>0) یک مجموعه محدود از شبکه‌های نیازمندی است.
- SoE= (EO_1, EO_2, …, EO_n), (n>0) یک مجموعه از ترتیب‌های اجرا است.

در شکل ۵-۴ یک شبکه دغدغه به تصویر کشیده شده است. شبکه دغدغه CN_1 از دو مجموعه به نامهای SoR و SoE تشکیل شده است. مجموعه SoR که نشان دهنده شبکه نیازمندیهای متعلق به شبکه دغدغه CN_1 است دارای دو عضو به نام شبکه نیازمندی RN_1, RN_2 است. مجموعه SoE نیز که نشان دهنده ترتیب‌های اجرا شبکه دغدغه CN_1 است از یک ترتیب اجرا به نام EO_1 تشکیل شده است. ترتیب اجرا EO_1 دربرگیرنده دو شبکه نیازمندی است که ترتیب نمایش آنها را نشان می‌دهد. در این ترتیب اجرا ابتدا شبکه نیازمندی RN_1 و بعد شبکه نیازمندی RN_2 باید اجرا شود.



CN_1= (SoR, SoE)

SoR= (RN_{11}, RN_{12}) SoE= (EO_1) EO_1= (RN_{11}, RN_{12})

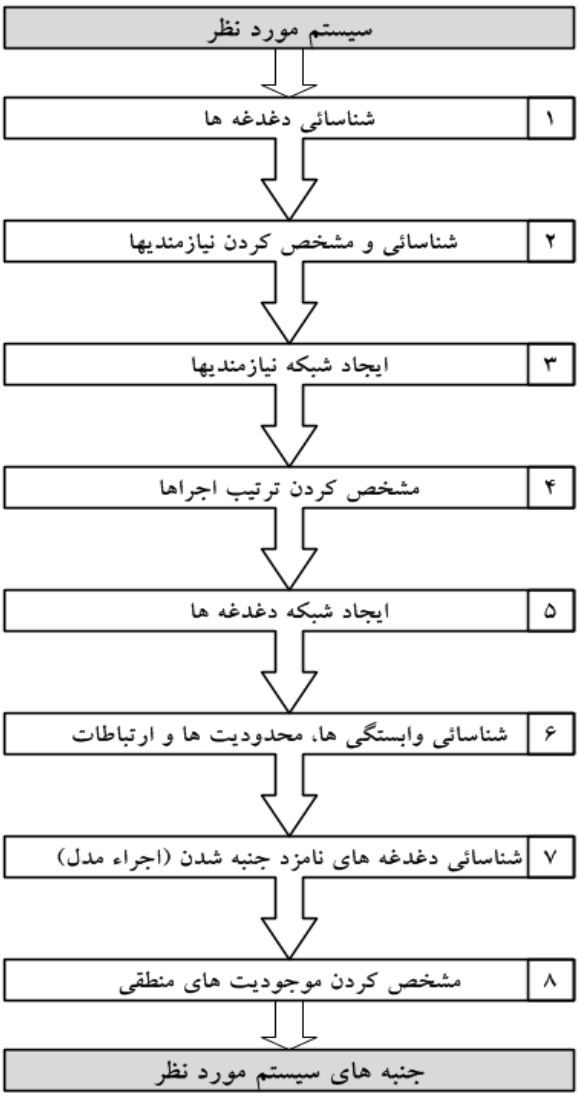
شکل ۵-۴: یک مثال برای شبکه دغدغه

در هر شبکه دغدغه، به ازای هر ترتیب اجرا باید یک نشانه در مکان اول قرار بگیرد. مکان شامل نشانه در شکل ۵-٤، p_{10} است که به دلیل وجود یک ترتیب اجرا یک نشانه در آن قرار گرفته است. شبکه دغدغه با توجه به اینکه از اجتماع چندین شبکه نیازمندی تشکیل شده است مستقل بوده (به دلیل مستقل بودن شبکه نیازمندی) و به راحتی می‌تواند به یک مدل از سیستم اضافه و یا از آن حذف شود در نتیجه قابلیت نگهداری سیستم و تغییر پذیری در مدل سیستمی که مبتنی بر شبکه دغدغه است به آسانی صورت می‌پذیرد.

[1] Set of Requirement nets (SoR)
[2] Set of Execution order (SoE)

۵-۳- تشریح روش شناسائی

روش پیشنهادی برای شناسائی جنبه‌ها مبتنی بر شبکه پتری، متشکل از هشت مرحله است که در هر مرحله یک سری فعالیت‌ها انجام می‌شود که به نوعی در جهت تکمیل مدل نهائی از سیستمی است که خواهان کشف جنبه‌های آن هستیم. این مراحل باید به ترتیب شماره انجام شوند ولی در هر مرحله می‌توان در صورت نیاز به مراحل قبلی یا مرحله اول برگشت و نتایج آنها را تکمیل کرد. روش پیشنهادی به دلیل ماهیت تعاریف پایه یک روش تدریجی و تکاملی است زیرا بعد از تشکیل مدل می‌توان آن را گسترش داده تا دامنه شناسائی جنبه‌ها افزایش پیدا کند. شکل ۵-۵ یک نمودار بلوکی از مراحل این روش را نشان می‌دهد که با معرفی سیستم مورد نظر شروع و با مشخص شدن جنبه‌های سیستم به پایان می‌رسد.



شکل ۵-۵: نمودار بلوکی از مراحل اجرا روش شناسائی دغدغه‌ها

۵-۳-۱- مرحله اول. شناسائی دغدغه‌ها

در این مرحله، دغدغه‌های سیستم شناسایی می‌شوند. این عمل می‌تواند توسط هر روش موجود برای شناسائی دغدغه انجام شود. دلیل شناسائی دغدغه‌ها به عنوان اولین مرحله به جای شناسائی نیازمندیها این است که اولا برای اینکه یک روش برای جنبه‌گرائی ارائه شود باید دغدغه نقش رده اول[1] را در سیستم داشته باشد. ثانیا به دلیل اینکه در روش پیشنهادی دانه‌بندی دغدغه‌ها درشت‌تر از دانه‌بندی نیازمندیها در نظر گرفته شده است و روش عملکرد بالا به پائین[2] دارد بنابراین باید اول دانه درشت‌ها و سپس دانه ریزها شناسائی شوند. در واقع یک دغدغه (دانه درشت) مجموعه از نیازمندیها (دانه ریز) را در بر می‌گیرد. در این مرحله همچنین باید، توضیحاتی پیرامون هر دغدغه شناسائی شده ذکر شود زیرا برای شناسائی ترتیب اجرا دغدغه‌ها این توضیحات مورد نیاز است (توضیحات مربوط به دغدغه‌ها به صورت مستند نوشتاری در نظر گرفته می‌شود).

۵-۳-۲- مرحله دوم. شناسائی و مشخص کردن نیازمندیها

نیازمندیهای متعلق به هر دغدغه در این مرحله شناسائی می‌شوند. شناسائی نیازمندیها می‌تواند توسط هر روش موجود برای مهندسی نیازمندیهای سنتی انجام شود. کیفیت[3] و کمیت[4] نیازمندیهای شناسائی شده به فعل و انفعالات یا تعاملات بین ذینفعان و مهندس نیازمندیها وابسته است [۶۰]. با این وجود شناسائی کلیه نیازمندیها برای یک سیستم به صورت کلی کاری مشکل و تقریبا غیر ممکن است و بسیاری از نیازمندیها به مرور شناسائی می‌شوند که این موضوع یکی از مشکلات مربوط به مهندسی نیازمندیها است.

با این وجود، عدم شناسائی نیازمندیهای سیستم به صورت یکجا هیچ مشکلی را برای شناسائی جنبه‌ها ایجاد نمی‌کند زیرا نیازمندیها به مرور (در مراحل بعدی) می‌توانند شناسائی شده و به سیستم اعمال شوند بدون اینکه باعث شوند که مدل از اول طراحی شود یا اینکه مشکلات ناسازگاری ایجاد کنند زیرا نیازمندیها طبق تعریف به یک شبکه پتری مستقل تبدیل خواهند شد که می‌تواند به راحتی به مدل اضافه یا از آن حذف شوند. حتی بعد از شناسائی جنبه‌ها باز هم می‌توان با اضافه کردن شبکه نیازمندیهای جدید به سیستم بدون از بین بردن ساختار مدل قبلی (مدل نهائی سیستم)، از همان مدل استفاده کرد و جنبه‌های زیادی را شناسائی کرد یا اینکه جنبه‌هایی از سیستم را نادیده گرفت.

نیازمندیهای شناسائی شده علاوه بر این که دارای نام و مشخصات هستند (مشخصات می‌تواند در قالب مستندات تهیه شود) دارای یک شناسه نیز می‌باشند. این شناسه با حرف بزرگ R شروع شده و بعد از آن یک

[1] First-class
[2] Top down
[3] Quality
[4] Quantity

عدد طبیعی ظاهر می‌شود. این عدد طبیعی منحصر به فرد است. به عنوان مثال R_1, R_{11}, R_{1001} می‌توانند شناسه‌ای برای نیازمندیها در نظر گرفته شوند. نیازمندیهای مشخص شده در این فاز در مراحل بعدی شناسائی جنبه، با شناسه خود مورد ارجاع قرار می‌گیرند.

برای سادگی و همچنین برای پروژه‌های بزرگ می‌توان قالب شماره‌گذاری نیازمندیها را تغییر داد (قالب شماره‌گذاری اختیاری است). به عنوان یک قالب نمونه در نظر بگیرید بعد از حرف R، اولین عدد طبیعی که ظاهر شود شماره دغدغه را مشخص کند و بعد از آن حرف "-" و نهایتا یک عدد طبیعی دیگر ظاهر شود که نشان دهنده شماره نیازمندی متعلق به دغدغه است. مثلا، شناسنه R_{21-9} نشان می‌دهد که نیازمندی R_{21-9} نهمین نیازمندی متعلق به دغدغه بیست و یک است. اما قالبی که در این فصل به کار برده می‌شود به این صورت است که اولین عدد (بین ۹-۱) مشخص کننده شماره دغدغه و دومین عدد مشخص کننده شماره نیازمندی است. در این قالب برای سادگی و درک بهتر دیگر از جداکننده "-" استفاده نمی‌شود (این نوع قالب برای سیستم‌هایی با حداکثر نه دغدغه صحیح عمل می‌کند).

۵-۳-۳- مرحله سوم. ایجاد شبکه نیازمندیها

نیازمندیهای شناسائی در مرحله قبل (دوم) نظیر به نظیر به شبکه نیازمندی تبدیل می‌شوند. این تبدیل با توجه به تعریف ذکر شده در قسمت ۵-۲-۲ به صورت زیر می‌باشد:

- شماره شناسه شبکه نیازمندی باید با شماره نیازمندی مشخص شده در مرحله قبل یکسان باشد. به عنوان مثال، اگر یک نیازمندی دارای شناسه R_{11} باشد در این صورت شناسه شبکه نیازمندی متناظر با آن به صورت RN_{11} خواهد بود.
- شناسه شبکه پتری متعلق به شبکه نیازمندی نیز مطابق شماره‌گذاری مشخص شده برای شبکه نیازمندی خواهد بود. به عنوان مثال، در نیازمندی R_{11} شبکه پتری متعلق به شبکه نیازمندی RN_{11} به صورت PN_{11} نامگذاری خواهد شد.

نکته قابل توجه این است که طبق تعریف باید موجودیت‌های منطقی متعلق به هر نیازمندی شناسائی شوند. ولی عمل شناسائی موجودیت‌های منطقی در این مرحله انجام نمی‌شود و به مرحله هشت انتقال داده می‌شود. علت به تعویق اندازی این است که برای شناسائی جنبه‌ها نیازی نیست که کلیه موجودیت‌های منطقی متعلق به کلیه نیازمندیهای شناسائی شده، استخراج شوند. بلکه باید موجودیت‌های منطقی نیازمندیهایی استخراج شوند که احتمال دارد دارای مشکل درهم تنیدگی یا پراکندگی باشند. بنابراین با توجه به اینکه در مرحله هفتم این نوع دغدغه‌ها و نیازمندیها مشخص می‌شوند عمل استخراج نیز به مرحله هشت انتقال داده شده است تا فقط

موجودیت‌های منطقی آن نیازمندیها شناسائی شوند (در این مرحله موجودیت‌های منطقی هر نیازمندی تهی (ϕ) در نظر گرفته می‌شود). شکل ۵-۲ یک شبکه نیازمندی نمونه را نشان می‌دهد

عمل عدم شناسائی موجودیت‌های منطقی کلیه نیازمندیها در این مرحله باعث تسهیل فرآیند شناسائی جنبه‌ها می‌شود زیرا دیگر نیازی نیست که مهندس نیازمندیها موجودیت‌های منطقی کلیه نیازمندیهای شناسائی شده را شناسائی کند، که می‌تواند بسیار زمانبر و باعث به وجود آمدن مستنداتی شود که حجیم بوده و منجر به پیچیده شدن فرآیند شناسائی جنبه‌ها می‌شود.

۵-۳-۴- مرحله چهارم. مشخص کردن ترتیب اجراها

برای اینکه بتوان شبکه دغدغه‌ها را ایجاد کرد باید ترتیب اجراها را برای هر دغدغه مشخص کرد. نحوه بدست آوردن ترتیب اجرا برای یک دغدغه مبتنی بر توضیحات مشخص شدن در مرحله اول برای دغدغه و نیازمندیهای مشخص در مرحله دوم است. ترتیب اجراهای بدست آمده برای هر دغدغه با همان قالب مشخص شده برای شناسه نیازمندیها، مشخص می‌شوند. به عنوان مثال $EO_{11}=(RN_{18}, RN_{19})$ نشان می‌دهد که یک ترتیب اجرا (اولین ترتیب اجرای مشخص شده) برای دغدغه شماره یک وجود دارد که در آن ابتدا باید نیازمندی هشت (RN_{18}) و بعد نیازمندی شماره نه (RN_{19}) اجرا شود.

نکته قابل توجه در مورد تعداد ترتیب‌های اجرا برای یک دغدغه این است که هرچه قدر دانه‌بندی[1] دغدغه درشت[2] باشد (در نظر گرفته شود) در این صورت تعداد ترتیب‌های اجرا نیز زیاد خواهد بود زیرا دانه‌بندی درشت به معنی داشتن نیازمندیهای بیشتر و همچنین تحقق بخشیدن به اهداف زیادی است که برای آن در نظر گرفته شده است. البته در مواردی که دانه‌بندی دغدغه درشت باشد می‌توان آن را به چندین دغدغه با دانه‌بندی ریز[3] تبدیل کرد در واقع به نوعی زیر دغدغه[4] تعریف کرد. اصولا باید در این روش پیشنهادی از دغدغه‌هایی با دانه‌بندی درشت و ریز دوری کرد زیرا دانه‌بندی درشت باعث پیچیده شدن شبکه دغدغه شده و دانه‌بندی ریز هم باعث می‌شود که نتوان شبکه نیازمندی برای آن تعریف کرد چون ممکن است خود آن دغدغه در سطحی باشد که بعد از شکستن آن به موجودیت‌های منطقی رسید. بنابراین باید دانه بندی دغدغه متوسط در نظر گرفته شود تا مدلسازی پیچیده نشود (نوع نگرش به درشت یا ریز بودن دانه‌بندی یک دغدغه به دیدگاه مهندس نیازمندی بستگی دارد).

[1] Granularity
[2] Coarse
[3] Fine
[4] Sub-concern

به عنوان یک مثال عملی، دغدغه‌ای به نام "ثبت نام[1]" را در نظر بگیرید. این دغدغه نیاز به سه نیازمندی به نامهای "ورود به سیستم"، "وارد کردن اطلاعات" و "ثبت اطلاعات" دارد. اگر دغدغه به صورت C_1 مشخص شود نیازمندیهای آن نیز دارای شناسه‌های به صورت R_{11}, R_{12}, R_{13} (به ترتیب نامشان) خواهد بود. در نتیجه شبکه نیازمندی متناظر نیز به ترتیب $RN_{11}, RN_{12}, RN_{13}$ نامگذاری خواهند شد. و ترتیب اجرا نیز برای دغدغه C_1 به صورت $EO_{11}=(RN_{11}, RN_{12}, RN_{13})$ تعیین خواهد شد.

۵-۳-۵- مرحله پنجم. ایجاد شبکه دغدغه‌ها

در این مرحله باید به ازای هر دغدغه شناسائی شده در مرحله اول شبکه دغدغه متناظر ساخته شود. نحوه ساخت شبکه دغدغه مبتنی بر تعریفی است که برای آن در بخش ۵-۲-۴ ارائه شد بنابراین داریم:

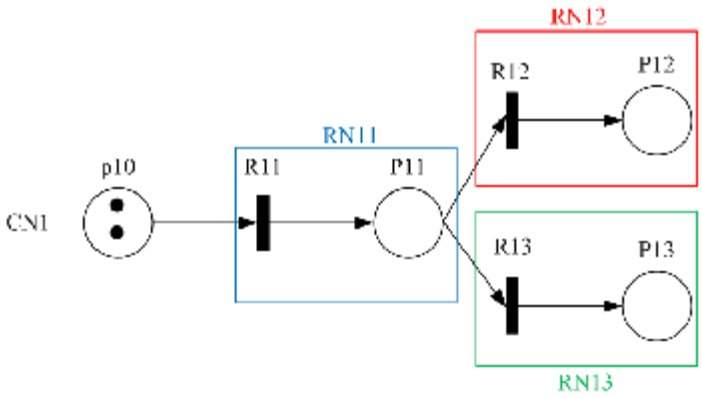
- مجموعه SoR که مشخص کننده مجموعه‌ی شبکه نیازمندیهای دغدغه است، اعضایش در مرحله دوم شناسائی شده‌اند.
- مجموعه SoE که مشخص کننده مجموعه‌ی ترتیب اجراها برای دغدغه است، اعضایش در مرحله چهارم مشخص شده‌اند.
- نامگذاری مکان‌ها مطابق قالب به کار رفته برای نیازمندیها است.
- گذرهای متعلق به شبکه نیازمندیها که با حرف t شروع می‌شوند در این مرحله به R تبدیل می‌شوند ولی اعداد طبیعی بعد حرف تغییر نمی‌کند.

برای مدلسازی (ایجاد) شبکه دغدغه با توجه به مجموعه SoR و SoE متعلق به هر دغدغه باید به صورت زیر عمل شود.

۱. ابتدا یک مکان ایجاد می شود و به تعداد ترتیب‌های اجرا موجود در مجموعه SoE، نشانه در آن قرار داده می‌شود.
۲. مکان اول کلیه شبکه نیازمندیهای متعلق به هر دغدغه حذف می‌شود.
۳. بر حسب هر ترتیب اجرا برای هر دغدغه، شبکه نیازمندیها برحسب ترتیب به یکدیگر متصل شده و نهایتا مکان اول ایجاد شده برای هر شبکه دغدغه به گذر شبکه نیازمندی اول هر ترتیب اجرا وصل می‌شود.
۴. اگر در دو ترتیب اجرا برای یک شبکه دغدغه شبکه نیازمندی اول مشترک باشد در این صورت از مکان بعدی گذر مربوط به شبکه نیازمندی مشترک دو کمان خارج می‌شود که به گذرهای شبکه

[1] Registration

نیازمندیها موجود در ترتیب اجراها بعد از نیازمندی مشترک وصل می‌شود. به عنوان مثال دغدغه‌ای که شامل سه شبکه نیازمندی و دو ترتیب اجرا است و شبکه نیازمندی اول در هر دو ترتیب اجرا با هم مشترک هستند، شبکه دغدغه آن به صورت شکل ۵-۶ است. در شبکه دغدغه نشان داده شده در شکل ۵-۶ به علت وجود دو ترتیب اجرا دو نشانه در مکان p_{10} قرار داده شده است تا اینکه از کلیه گذرها نشانه‌ای عبور کرده باشد.



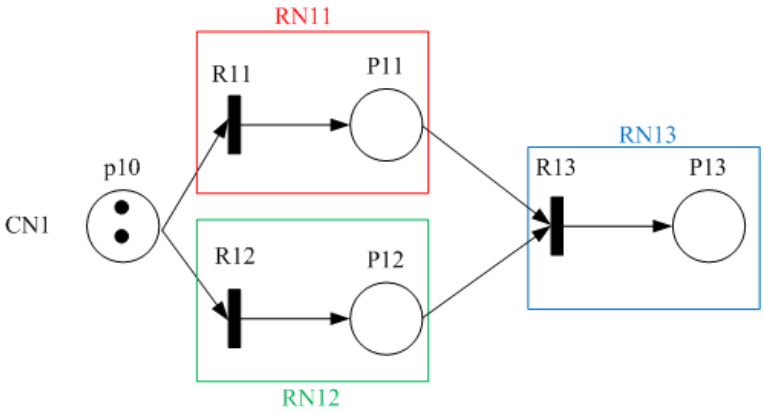
$CN_1 = (SoR, SoE)$

$SoR = (RN_{11}, RN_{12}, RN_{13}), SoE = (EO_{11}, EO_{12})$

$EO_{11} = (RN_{11}, RN_{12}), EO_{12} = (RN_{11}, RN_{13})$

شکل ۵-۶: یک شبکه دغدغه با یک شبکه نیازمندی مشترک در ابتدای دو ترتیب اجرا

۵. اگر در دو ترتیب اجرا برای یک شبکه دغدغه، شبکه نیازمندی آخر مشترک باشد در این صورت از مکانهای بعد گذر مربوط به شبکه نیازمندیهای قبل آخر هر دو ترتیب اجرا یک کمان به گذر مربوط به آخرین شبکه نیازمندی وصل می‌شود. یعنی گذر مربوط به شبکه نیازمندی مشترک دارای دو کمان ورودی خواهد بود. به عنوان مثال شبکه دغدغه، برای دغدغه‌ای با سه نیازمندی و دو ترتیب اجرا که آخرین شبکه نیازمندی در ترتیب اجراهای آن مشترک هستند به صورت شکل ۵-۷ خواهد بود.



$CN_1 = (SoR, SoE)$

$SoR = (RN_{11}, RN_{12}, RN_{13})$, $SoE = (EO_{11}, EO_{12})$

$EO_{11} = (RN_{11}, RN_{13})$, $EO_{12} = (RN_{12}, RN_{13})$

شکل ۵-۷: یک شبکه دغدغه با یک شبکه نیازمندی مشترک در آخر دو ترتیب اجرا

۶. در سایر موقعیت‌ها که شبکه نیازمندی مشترکی وجود دارد مشابه بند ۵ و ۴ عمل می‌شود.

نکته قابل توجه در طراحی یک شبکه دغدغه با تعداد نشانه‌هایی که رابطه مستقیم با تعداد ترتیب‌های اجرا دارد این است مدل نهائی سیستم مورد نظر باید اجرا شده و کلیه نشانه‌ها به مکان آخر انتقال داده شوند در غیر این صورت طراحی شبکه دغدغه نادرست است.

در مورد نشانه باید به این نکته اشاره کرد که بر حسب نوع طراحی می‌توان نوع نشانه مختلفی را تعریف کرد ولی در هر صورت باید در نوع تعریف شده برای نشانه، متغییری وجود داشته باشد که نام دغدغه در آن ثبت شود. در مورد شبکه‌های دغدغه‌ای که فقط یک ترتیب اجرا وجود دارد فقط تعریف یک متغییر از نوع رشته برای نشانه کافی است ولی در مورد دغدغه‌هایی که بیش از یک ترتیب اجرا دارند باید متغییری دیگری نیز به نشانه الصاق شود که در آن شماره ترتیب اجرا باید وارد شود تا در موقعی که یک ترتیب اجرا می‌خواهد تحقق پیدا کند فقط ار نشانه متعلق به خود استفاده کنند. همچنین در شبکه‌های دغدغه‌ای که بیش از یک ترتیب اجرا وجود دارد باید گاردهای لازم در مورد تشخیص نشانه در گذرهای مناسب وارد شود (در شبکه‌های دغدغه با یک ترتیب اجرا نیازی به گارد نیست).

۵-۳-۶- شناسائی وابستگی‌ها، محدودیت‌ها و ارتباطات

در این مرحله باید وابستگی‌ها[1]، محدودیت‌ها[1] و ارتباطات[2] میان شبکه نیازمندیها شناسائی شوند. به عنوان مثال، محدودیت در ترتیب اجرای شبکه نیازمندیها در یک ترتیب اجرا، نوعی ارتباط است که در مجموعه شبکه

[1] Dependencies

نیازمندیهای یک دغدغه وجود دارد. اما ممکن است حالتهایی وجود داشته باشد که محدودیت، رابطه یا وابستگی بین شبکه نیازمندیهای یک شبکه دغدغه با شبکه نیازمندیهای شبکه دغدغه دیگر وجود داشته باشد که این موارد باید در این مرحله شناسائی شوند. البته وابستگیهایی نیز بین شبکه دغدغه‌ها می‌تواند وجود داشته باشد (در روش پیشنهادی برای پیاده‌سازی وابستگیها بین شبکه دغدغه‌ها راه حلی ارائه نشده است).

وابستگی‌ها، نتیجه‌ی منطق حرفه[3] سیستمی است که خواهان شناسائی جنبه‌های آن هستیم [۷۲]. ارتباطات نوعی رابطه منطقی هستند که می‌تواند فرآیندی[4] یا داده‌ای[5] باشد. یعنی اینکه دو شبکه نیازمندی ممکن است به خاطر ماهیت فرآیندی خودشان با هم ارتباطات داشته باشند یا اینکه به خاطر داده‌ای که میان آنها مشترک است. البته ارتباط شبکه نیازمندیها همچنین می‌تواند به صورت تفسیری[6] نیز در نظر گرفته شود. ارتباطات تفسیری، معنی تفسیری وابستگی بین دغدغه‌های منطقی را منعکس می‌کنند [۵]. این نوع ارتباطات اصولا بر تفسیر وابسته به متن[7]، از مفاهیم و معانی دغدغه وابسته هستند (این نوع ارتباطات که برای دغدغه تعریف شده است در این روش کمی تغییر داده می‌شود و به جای دغدغه‌های منطقی، شبکه نیازمندی در نظر گرفته می‌شود).

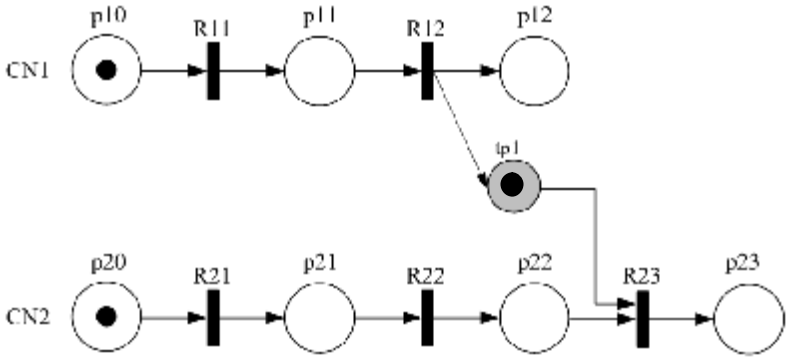
در روش پیشنهادی برای شناسائی جنبه‌ها، هیچ گونه دسته‌بندی برای وابستگی‌ها، محدودیت‌ها و ارتباطات در نظر گرفته نمی‌شود زیرا این سه مفهوم هم پوشانی با یکدیگر دارند. مهندس نیازمندیهای جنبه‌گرا می‌تواند برحسب صلاح دید خود نوعی وابستگی تعریف کند (از نکته نظر خود) و آن را در این روش به کار ببرد. زیرا هدف از این مرحله این است که هر گونه ارتباطی که بین شبکه نیازمندیها وجود دارند شناسائی شوند تا بتوان جنبه‌ها را به درستی شناسائی کرد (ارتباطات بین شبکه نیازمندیها دو به دو در نظر گرفته می‌شود). بعد از شناسائی ارتباطات باید آن را به مدل که متشکل از تعدادی شبکه دغدغه است اعمال کرد. این عمل به شرح زیر صورت می‌پذیرد:

- به ازای هر یک رابطه بین دو شبکه نیازمندی یک مکان ایجاد می‌شود. این مکان به عنوان یک مکان موقت[8] برای برقرار کردن ارتباط مورد استفاده قرار می‌گیرد (به صورت tp نامگذاری می شود).
- کمانی مکان موقت را به گذرهای شبکه نیازمندیهایی که به یکدیگر وابسته هستند یا با هم ارتباط دارند، متصل می‌کند.

---

[1] Restrictions
[2] Relationships
[3] Business logic
[4] Co-process
[5] Co-data
[6] Interpretive
[7] Context-dependent
[8] Temp place

به عنوان مثال، دو شبکه دغدغه CN_1, CN_2 را در نظر بگیرد که به ترتیب دو و سه شبکه نیازمندی دارند. در این دو شبکه دغدغه، شبکه نیازمندی RN_{23} محدود به اجرای شبکه نیازمندی RN_{12} است در واقع اول باید RN_{12} و بعد RN_{23} تحقق پیدا کند. نحوه رسم ارتباط آنها در شبکه پتری در شکل ۵-۸ نشان داده شده است. همان طور که در شکل مشخص است از گذر مربوط به شبکه نیازمندی RN_{12} کمانی به مکان موقت tp_1 وارد شده و از مکان موقت tp_1 کمانی به سمت گذر مربوط به شبکه نیازمندی RN_{23} خارج شده است که نشان دهنده ارتباط بین آنها است.



شکل ۵-۸: نحوه مدلسازی وابستگی در دو شبکه دغدغه

۵-۳-۷- مشخص کردن دغدغه‌های نامزد جنبه شدن

برای اینکه بتوان جنبه‌ها را شناسائی کرد باید در این مرحله شبکه دغدغه‌هایی که نیازمندی آنها احتمالا دارای مشکل درهم تیندگی هستند را شناسائی کرد تا در مرحله بعد موجودیت‌های منطقی آنها شناسائی شده و مشخص شود که آیا دغدغه متناظر با آن نیازمندی می‌تواند جنبه در نظر گرفته شود یا نه؟

بدین منظور باید کلیه گذرهایی که دارای دو یا بیشتر کمان ورودی هستند را بررسی کرد. اگر گذری دو یا بیشتر کمان ورودی داشته باشد و این کمانهای ورودی از انواع مختلف باشند (یعنی اینکه کمان ورودی از یک شبکه دغدغه دیگر به گذر متصل شده یا به بیان بهتر نشانه حمل شده توسط کمانهای ورودی، متفاوت باشند) در این صورت آن نیازمندیها احتمالا دارای مشکل درهم تنیدگی است و بنابراین دغدغه دربرگیرنده آنها می‌تواند به عنوان نامزدی برای جنبه شدن در نظر گرفته شود. بعد از مشخص شده کلیه گذرهای دارای شرایط ذکر شده، به ازای هر دو کمان ورودی با نشانه‌های متفاوت برای یک گذر (یک ورودی مربوط به خود شبکه نیازمندی و دیگری از شبکه نیازمندی دیگر) یک تاپل دوتائی[1] به صورت (<token1>, <token>) برای آن گذر در نظر گرفته می‌شود. بنابراین اگر یک گذر، سه ورودی با نشانه‌های مختلف داشته باشد در این صورت دوتا

[1] 2-tuple

تاپل دوتائی به صورت (<token1>, <token2>), (<token1>, <token3>) خواهیم داشت. که token1 مربوط به خود گذر و token2, token3 متعلق به گذرهای شبکه نیازمندیهای دیگر هستند. نکته قابل توجه این است که نشانه‌ها دارای قالبی هستند که در آن شناسه شبکه دغدغه و شبکه نیازمندی ذکر شده است.

عمل مشخص کردن خروجی تاپل دوتائی به صورت خودکار توسط قابلیت مانیتوریگ هر ابزار پشتیبانی کننده از شبیه‌سازی و اجرای شبکه پتری قابل دست‌یابی است به شرط آنکه نوشته لازم در گذرها وارد شده باشد. در این نوشته می‌تواند قالب تاپل دوتائی را در صورت نیاز تغییر دارد تا خروجی در قالب مورد انتظار ظاهر شود.

۵-۳-۸- مشخص کردن موجودیت‌های منطقی

بعد از شناسائی کلیه تاپل‌های دوتائی، موجودیت‌های منطقی مرتبط با هر یک از شبکه‌های نیازمندی موجود در تاپل دوتائی باید مشخص شوند. اگر یک موجودیت منطقی یافت شود که در مجموعه موجودیت‌های منطقی هر دو شبکه نیازمندی مشخص شده در یک تاپل دوتائی وجود دارد در این صورت موجودیت منطقی به عنوان عنصری در نظر گرفته می‌شود که دارای مشکل درهم تنیدگی است بنابراین دغدغه‌ای که این موجودیت منطقی در شبکه نیازمندی آن است به عنوان دغدغه مداخله‌ای در نظر گرفته می‌شود و دغدغه مداخله‌ای نیز باید به عنوان جنبه در نظر گرفته شود.

خروجی حاصل از مانیتور کردن مدل نهایی سیستم و موجودیت‌های منطقی شناسائی شده برای شبکه‌های نیازمندی مشخص شده در خروجی، باعث شناسائی جنبه‌ها می‌شود. با انجام مرحله هشت یک مجموعه‌ای حاصل می‌شود که شامل جنبه‌ها و موجودیت‌هایی منطقی است که جنبه به آنها باید اعمال شود. از آنجائی که ممکن است در مجموعه نهایی جنبه و موجودیت منطقی، اشتراکاتی وجود داشته باشد باید آنها را حذف کرد تا بتوان به طور واضح مشخص کرد که هر جنبه به چه موجودیت‌های منطقی باید اعمال شود. خروجی مربوط به جنبه و موجودیت‌های منطقی تحت تاثیر آنها را می‌توان به صورت جدول ۵-۱ نشان داد.

جدول ۵-۱: جدول ارتباط جنبه‌ها با موجودیت‌های منطقی

	موجودیت منطقی ۱	موجودیت منطقی ۲	...	موجودیت منطقی n
جنبه ۱				
جنبه ۲				
...				
جنبه ۴				

۵-۴- خلاصه فصل

در این فصل به تشریح روش شناسائی جنبه‌ها مبتنی بر شبکه پتری پرداخته شد. ذکر شد که روش پیشنهادی از هشت مرحله‌ی، شناسائی دغدغه‌ها، شناسائی و مشخص کردن نیازمندیها، ایجاد شبکه نیازمندیها، مشخص کردن ترتیب اجراها، ایجاد شبکه دغدغه‌ها، شناسائی وابستگی‌ها، شناسائی دغدغه‌های نامزد جنبه شدن و نهایتا مشخص کردن موجودیت‌های منطقی تشکیل شده است که ورودی اولین مرحله مستندات سیستم و خروجی مرحله هشتم جنبه‌های سیستم به همراه موجودیت‌های منطقی تحت تاثیر جنبه‌ها هستند.

در جهت تشریح مراحل، مفاهیم پایه روش که عبارتند از شبکه نیازمندی، شبکه دغدغه و ترتیب اجرا معرفی شدند و مثالهایی در مورد هر کدام ذکر شد و بیان گردید که هدف از هر کدام از این تعاریف چیست و چه کاربردی دارند. سپس مراحل روش پیشنهادی به صورت کامل تشریح شد و مثالهایی برای هر کدام عنوان شد تا بتوان درک کاملی از هر مرحله بدست آورد. طی تشریح مراحل بیان شد که روش، یک فرآیند تدریجی و تکاملی است یعنی اینکه لازم نیست مدل نهایی سیستم به صورت کلی مدلسازی شود و بعد اجرا شده و جنبه‌ها کشف شوند بلکه می‌توان به صورت تدریجی مدل را ایجاد کرده و جنبه‌ها را به همین شکل شناسائی کرد که دلیل این تدریجی و تکاملی بودن، مستقل تعریف شدن شبکه نیازمندی و شبکه دغدغه عنوان شد.

فصل ششم

مطالعه موردی

در این فصل یک مطالعه موردی به نام سیستم مدیریت هتل مطرح می‌شود و جنبه‌های آن با استفاده از روش پیشنهادی در فصل قبل شناسائی می‌شوند. هدف از مطرح کردن مطالعه موردی این است که علاوه بر تشریح مراحل روش پیشنهادی بر روی یک سیستم واقعی، نشان داده شود که روش پیشنهادی قابلیت شناسائی جنبه‌ها را دارد.

۶-۱- مقدمه

روش پیشنهادی برای شناسائی جنبه‌ها با استفاده از شبکه پتری، مکانیزمی را فراهم می‌کند که بتوان از مستندات نیازمندیها که به قالب شبکه پتری تبدیل می‌شوند جنبه‌ها را شناسائی کرد. همچنین تعاریف پایه این روش، ساختار رسمی برای توسعه‌دهندگان فراهم می‌کند که بتوانند سایر فعالیت‌های مربوط به توسعه نرم‌افزاری جنبه‌گرا (به خصوص مهندسی نیازمندیها و معماری جنبه‌گرا) را مبتنی بر آنها بیان کنند. هر روش کاربردی در اولین گام معرفی خود، به بررسی رفتارها، ساختارها و الگوریتم خود می‌پردازد و بعد از استقرار خواسته‌های خود مبتنی بر مفاهیم اولیه، الگوریتم خود را بر روی یک مطالعه موردی اعمال می‌کند تا اینکه اولا، روش به صورت عملی تشریح شده و استفاده کنندگان از آن با نحوه عملکرد آن آشنا شوند. ثانیا، نشان داده شود که ادعاهایی که در مورد عملکرد خود دارد درست است. به عبارت بهتر، خروجی مورد انتظار بکارگیران خود را تولید می‌کند.

با توجه به این که روش پیشنهادی از مفاهیم پایه رسمی شروع شده و با اجرای گامهای مناسب، به خروجی مورد نظر خود می‌رسد. باید علاوه بر تشریح رسمی آن، یک مطالعه موردی نیز با آن بررسی شود تا اینکه مشخص شود مفاهیم تعریف شده رسمی، صحیح بوده و همچنین جنبه‌ها به درستی شناسائی می‌شوند. بدین منظور یک مطالعه موردی به نام سیستم مدیریت هتل[1] مطرح می‌شود که روش پیشنهادی بر روی آن اعمال شده و جنبه‌های آن مشخص می‌شوند (کشف می‌شوند).

سیستم مدیریت هتل یک مطالعه موردی است که در مرجع [۸] تعریف شده و توسعه نرم‌افزاری جنبه‌گرا با موارد کاربری با استفاده از آن تشریح شده است. از دلایل انتخاب این مطالعه موردی، اولا استفاده آن در کارهای تحقیقاتی دیگر است که در آن هر محقق روش خاص خود رو بر روی آن اعمال کرده تا نتیجه مورد انتظار خود را بدست آورده و مقایسه‌ای بین نتیجه کار خود و دیگر کارها انجام دهد. ثانیا، دغدغه‌های این مطالعه موردی قابلیت فهم آسانتری دارند. بنابراین اعمال روش پیشنهادی بر روی این مطالعه موردی و گرفتن خروجی از آن و بررسی نتایج حاصله می‌تواند نقش مهمی در درک و کاربرد روش داشته باشد.

نکته مهم در مورد نحوه اعمال روش پیشنهادی بر روی مطالعه موردی این است که در فصل قبل، روش پیشنهادی با استفاده از یک تعریف ساده از شبکه پتری تشریح شد تا اینکه بتوان ماهیت روش را بدون پیچیدگی‌های مربوط به نحوه پیاده‌سازی معرفی کرد. ولی برای اینکه بتوان روش پیشنهادی را بر روی یک سیستم با یک ابزار مناسب پیاده‌سازی کرد باید از شبکه پتری رنگی استفاده شود. البته استفاده از شبکه پتری رنگی دلیل بر نقض تعاریف (شبکه نیازمندی و شبکه دغدغه) نیست و این تعاریف همچنان معتبر هستند.

[1] Hotel management system

در اعمال روش پیشنهادی بر روی سیستم مدیریت هتل از ابزار CPN/Tools استفاده می‌شود [۴۷] که از امکانات مربوط به مدلسازی، شبیه‌سازی و مانیتورینگ شبکه پتری به خوبی حمایت می‌کند و یکی از برجسته‌ترین ابزارها در این زمینه است.

۶-۲- معیارها و شاخص‌ها

همان طور که در فصل اول بیان شد هدف از این کار تحقیقاتی فراهم کردن یک روشی است که بتوان مبتنی بر آن جنبه‌های یک سیستم را شناسائی کرد. مانند هر روش دیگر که انتظارات خاصی از به کارگیری آن وجود دارد از روش پیشنهادی برای شناسائی جنبه‌ها نیز انتظار می‌رود عملیات‌های زیر را پشتیبانی کند:

- عمل شناسائی جنبه‌ها انجام بپذیرد. در واقع در پایان اجرا روش، دغدغه‌های سیستم مشخص شده باشند. این عمل می‌تواند با استفاده از روشهای مهندسی نیازمندیهای جاری صورت بپذیرد ولی آنچه که مهم است و در این روش به آن دقت شده است مشخص شدن دغدغه‌ها است.
- جداسازی دغدغه‌ها تحقق یابد. یعنی اینکه دغدغه‌های سیستم به دو دسته دغدغه‌های هسته و دغدغه‌های مداخله‌ای تبدیل شوند.
- دغدغه‌های سیستم نمایش داده شوند. در واقع بعد از اینکه دغدغه‌ها شناسائی شدند بتوان آنها را در قالب‌های خاص نمایش داد. در این قالب‌ها می‌توان مشخصات جزئی و کلان از یک دغدغه‌ها را ذکر کرد.
- دغدغه، رده اول باشد. یعنی اینکه سیستم محوریت کار خود را بر روی مفهوم دغدغه بنا کند (این فاکتور از جمله انتظاراتی است که باید در روشهای جنبه‌گرائی تمرکز بیشتری بر روی آن صورت بپذیرد).
- جنبه‌ها هم نیازهای وظیفه‌مندی و غیروظیفه‌مندی در نظر گرفته شوند. در واقع جنبه‌های شناسائی شده فقط متعلق به یک دسته‌بندی از نیازهای سیستم نباشند زیرا جنبه می‌تواند نیازمندی وظیفه‌مندی و یا غیروظیفه‌مندی در نظر گرفته شود.
- برای مفاهیم دغدغه و نیازمندی، تعاریف رسمی ارائه شود تا بتوان فرآیند شناسائی جنبه‌ها را سیستماتیک‌تر کرد.

با توجه به انتظاراتی که بیان شد مسلما نیاز به یک یا چند شاخص اصلی است که بتوان مبتنی بر آنها روش پیشنهادی را مورد ارزیابی قرار داد. مهمترین شاخها عبارتند از:

- جنبه‌های اصلی در سیستم شناسائی شود.
- تمام دغدغه‌های سیستم بتوانند بر مبنای تعاریف پایه، تعریف شوند.
- فقط جنبه‌های مربوط به یک دسته‌بندی از نیازها (وظیفه‌مندی و غیروظیفه‌مندی) شناسائی نشوند.
- روش به صورت رسمی قابل پیگیری باشد.

همان طور که قبلا نیز ذکر شد روش پیشنهادی با توجه به شاخهایی ارزیابی خواهد شد که شاخص‌ها در این قسمت بیان شدند. ارزیابی نیز با استفاده از یک مطالعه موردی صورت خواهد گرفت. مطالعه موردی یک سیستم مدیریت هتل است که هم دارای نیازهای وظیفه‌مندی و غیروظیفه‌مندی است و دغدغه‌های آن قابلیت فهم راحت‌تری دارند. بنابراین در ادامه سیستم مدیریت هتل تشریح خواهد شد و مراحل اعمال روش پیشنهادی نیز بعد از آن به صورت کامل بر روی آن اعمال شده و نتایج نمایش داده خواهند شد.

۶-۳- سیستم مدیریت هتل

هدف از سیستم مدیریت هتل این است که کارهای مربوط به هتلداری مدیریت شود. در سیستم مدیریت هتل، مشتری باید بتواند از قبل اتاق مورد نظر خود را رزرو کند. عمل رزرو اتاق[1] بدین شکل انجام می‌پذیرد که مشتری لیست اتاقهای هتل را همراه با مشخصاتشان (قیمت، وسایل و ..) مشاهده کرده و اتاقی که با شرایط خود سازگار است را رزرو می‌کند. شخصی که عمل رزرواسیون را انجام داده اگر در روزی که اتاق برای آن روز رزرو شده، به هتل مراجعه کند در این صورت متصدی هتل باید عملیات ثبت نام[2] را برای مشتری انجام دهد. متصدی هتل بدین منظور، ابتدا باید بررسی کند که شخص مورد نظر چنین اتاقی را رزرو کرده یا نه، سپس اتاق را از حالت رزرو خارج کرده و به نام شخص رزرو کننده ثبت می‌کند (در واقع به سیستم اعلام می‌شود که اتاق پر است). نهایتا، متصدی هتل یک صورت‌حساب اولیه برای مشتری ایجاد می‌کند (باز می‌کند).

بعد از اینکه مشتری در هتل اقامت کرد و خواست هتل را ترک کند متصدی هتل باید با مشتری تسویه حساب کند. فرآیند تسویه حساب[3] بدین صورت است که متصدی کلیه هزینه‌هایی که (اعم از کرایه اتاق، سرویس‌های ویژه، ناهار و شام) شخص مالک اتاق باید پرداخت کند را محاسبه کرده و صورت‌حساب نهائی برای مشتری تهیه می‌کند. بعد از اینکه مشتری صورت‌حساب را پرداخت کرد متصدی هتل، اتاق اختصاص داده شده به مشتری را از حالت پر خارج می‌کند.

[1] Reserve room
[2] Check In
[3] Check Out

علاوه بر این ویژگیهای که سیستم برای مشتری و متصدی هتل فراهم می‌کند مدیر اصلی هتل خواهان آن است که برای کلیه عملیات انجام شده در این سیستم یک واقعه‌نگاری[1] انجام شود. واقعه‌نگاری فرآیندی است که در آن کلیه کارهایی که یک سیستم انجام می‌دهد (چه انجام داده شده توسط کاربر یا توسط خود سیستم) در یک فایل ثبت می‌شود تا در صورت نیاز بتوان حالتهای قبل و بعد یک عمل، یا میزان تعداد عملهای انجام شده در سیستم را مشاهده و مورد تحلیل قرار داد. بنابراین در سیستم مدیریت هتل باید از عملیات ثبت نام، رزرواسیون، تهیه صورت‌حساب و غیره واقعه‌نگاری شود. اطلاعاتی که باید توسط واقعه‌نگار نگهداری شود می‌تواند درون یک فایل معمولی یا درون یک پایگاه داده باشد که بسته به نوع پیکربندی سیستم قابل تغییر است.

۶-۴- شناسائی جنبه‌ها

برای اینکه بتوان جنبه‌های سیستم مدیریت هتل را شناسائی کرد باید از گامهای مشخص شده در بخش ۵-۳ برای شناسائی جنبه‌ها تبعیت کرد. همان طور که در نمودار بلوکی در شکل ۵-۵ نشان داده شده است ورودی گام اول، مستندات سیستم است که باید وجود داشته باشد تا گام اول شروع و به دنبال آن گامهای بعدی ادامه پیدا کنند. مستندات مربوط به سیستم مدیریت هتل در بخش ۶-۲ ارائه شده است بنابراین می‌تواند با توجه به آن شناسائی جنبه‌ها را آغاز کرد.

۶-۴-۱- مرحله اول

در این گام باید دغدغه‌ها را از مستندات سیستم مدیریت هتل شناسائی کرد. شناسائی دغدغه‌ها با توجه به تحلیل لغوی مستندات نیازمندیها صورت می‌پذیرد. بعد از شناسائی دغدغه‌ها باید مشخصات هر دغدغه (اطلاعاتی راجب نحوه تحقق) را نیز بدست آورد. بنابراین با توجه به مستندات، دغدغه‌های زیر برای سیستم مدیریت هتل شناسائی می‌شوند:

۱. رزرو اتاق C_1

برای رزرو یک اتاق، مشتری باید اتاقها را بررسی کرده و اگر اتاق خالی و مناسب وجود داشته باشد عمل رزرواسیون را انجام دهد.

۲. ثبت نام کردن C_2

برای ثبت نام باید یک اتاق به مشتری تخصیص داده شود و رزرواسیون آن حذف شود و نهایتا یک صورت‌حساب اولیه برای آن ایجاد شود (باز شود).

[1] Logging

۳. تسویه حساب کردن C_3

برای تسویه حساب، متصدی هتل باید کلیه پرداختیهایی که مشتری باید انجام دهد را محاسبه کرده و بعد یک صورت‌حساب برای آن صادر می‌کند. بعد از اینکه مشتری صورت‌حساب را کاملا پرداخت کرد اتاق آن خالی شود (مشتری مورد نظر باید از لیست مشتریان حذف شود).

۴. واقعه‌نگاری C_4

برای واقعه‌نگاری باید عملیات انجام شده در سیستم مانیتور شود و در صورت انجام یک عمل خاص (مانند پرداخت صورت‌حساب) آن تشخیص داده شده و خروجی در یک پایگاه‌داده/فایل ذخیره شود.

۶-۴-۲- مرحله دوم

بعد از شناسائی دغدغه‌ها، باید نیازمندیهای هر یک از دغدغه‌ها شناسائی شوند. به عبارت ساده‌تر، باید نیازمندیهای سیستم شناسائی شده و آنها را به دغدغه‌ها نسبت داد. شناسائی می‌تواند توسط هر روش سنتی استفاده شده برای مهندسی نیازمندیهای انجام شود (مانند دیدگاه‌ها). برای نشان دادن دغدغه‌ها و نیازمندیهای هر یک، از ساختار ساده شده مربوط به دیدگاه‌ها استفاده می‌شود [۳۶، ۷۳].

از آنجائی که دغدغه‌های این سیستم کمتر از ده در نظر گرفته شده‌اند قالبی که برای شماره‌گذاری شناسه نیازمندیها استفاده می‌شود تا بتوان ارتباط آنها با دغدغه‌ها را به سادگی مشخص کرد به صورت زیر است:

عدد طبیعی دوم عدد طبیعی اول R

عدد طبیعی اول (بین ۱ تا ۹)، شماره دغدغه را نشان دهد که مشخص می‌کند نیازمندی متعلق به کدام دغدغه است و عدد طبیعی دوم (بین ۱ تا ۹)، شماره نیازمندی را نشان می‌دهد که مشخص می‌کند این نیازمندی، چندمین نیازمندی دغدغه مشخص شده در عدد طبیعی اول است. بنابراین با توجه به این قالب و ساختار ساده شده دیدگاه‌ها، نیازمندیهای سیستم مدیریت هتل در شکل ۶-۱ به تصویر کشیده شده‌اند.

(الف)

دغدغه : رزرو اتاق
نیازمندیها :
۱. بررسی قابل دسترس بودن اتاق (R_{11})
۲. انجام عمل رزرواسیون (R_{12})

(ب)

دغدغه : ثبت نام کردن
نیازمندیها :
۱. تخصیص اتاق (R_{21})
۲. حذف یا استفاده کردن رزرواسیون (R_{22})
۳. ایجاد صورت‌حساب اولیه (R_{23})

(پ)

دغدغه : تسویه حساب کردن
نیازمندیها :
۱. محاسبه صورت‌حساب (R_{31})
۲. پرداخت صورت‌حساب (R_{32})
۳. خالی کردن اتاق (R_{33})

(ت)

دغدغه : واقعه‌نگاری
نیازمندیها :
۱. واقعه‌نگاری (R_{41})
۲. ذخیره کردن در فایل یا پایگاه‌داده (R_{42})

شکل ۶-۱: دغدغه‌ها و نیازمندیهای سیستم مدیریت هتل

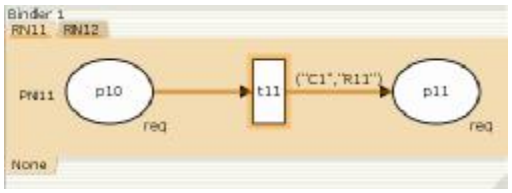
۶-۴-۳- مرحله سوم

در این گام باید برای هر نیازمندی شناسائی شده در مرحله قبل، شبکه نیازمندی متناظر ساخته شود. با توجه به اینکه شبکه نیازمندیها با استفاده از ابزار CPN/Tools ایجاد می‌شود و ابزار نیز مبتنی بر شبکه پتری رنگی است باید برای هر مکان یک نوع تخصیص داده شود. این نوع تحت عنوان req نامگذاری می‌شود و به صورت زیر در CPN/Tools تعریف می‌شود:

colset req=product STRING * STRING;

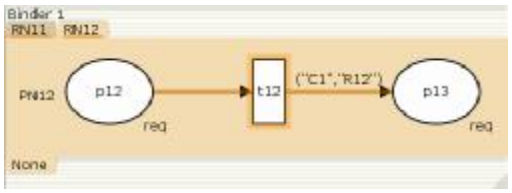
نوع req متشکل از دو متغییر رشته‌ای است که اولی دربرگیرنده شناسه دغدغه و دیگری در برگیرنده شناسه نیازمندی است.

هر کمانی که از یک گذر در شبکه پتری رنگی خارج می‌شود باید دارای یک inscription باشد تا نشانه از طریق آن انتقال داده شود. در مورد کمان خروجی از هر گذر مربوط به شبکه نیازمندی نیز باید inscription وجود داشته باشد. از آنجائی که هدف، شناسائی دغدغه‌ها است هر گذر متعلق به شبکه نیازمندی باید نشانه‌ای تولید کند که شامل دو متغییر (از نوع رشته) باشد که متغییر اول شامل شناسه دغدغه دربرگیرنده نیازمندی بوده و متغییر دوم شناسه خود نیازمندی باشد. بنابراین inscription کمان خروجی به صورت (شناسه نیازمندی، شناسه دغدغه) خواهد بود. شکلهای ۶-۲ تا ۶-۵ به ترتیب شبکه نیازمندیها مربوط به سیستم مدیریت هتل را نمایش می‌دهند.



(a)

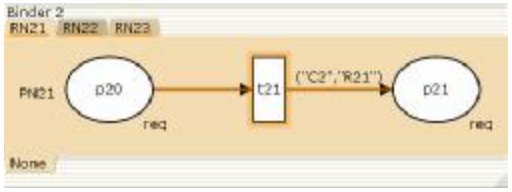
$RN_{11}= (PN_{11}, LE_0)$
$LE_0= (\phi)$



(b)

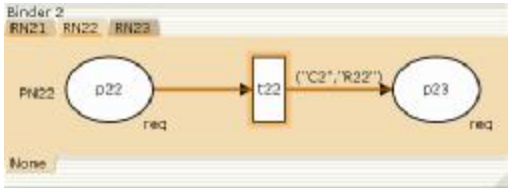
$RN_{12}= (PN_{12}, LE_1)$
$LE_1= (\phi)$

شکل ۶-۲: شبکه نیازمندیهای R_{11}, R_{12} سیستم مدیریت هتل



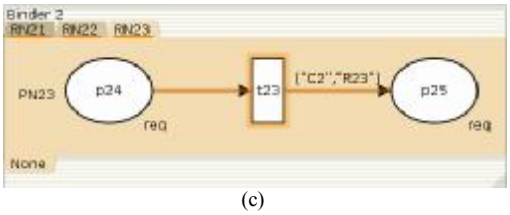
(a)

$RN_{21}= (PN_{21}, LE_2)$
$LE_2= (\phi)$



(b)

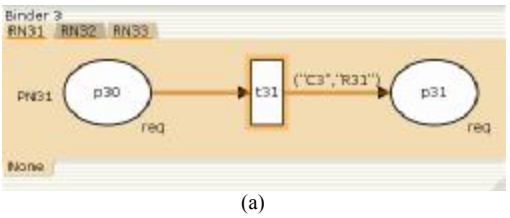
$RN_{22}= (PN_{22}, LE_3)$
$LE_3= (\phi)$



(c)

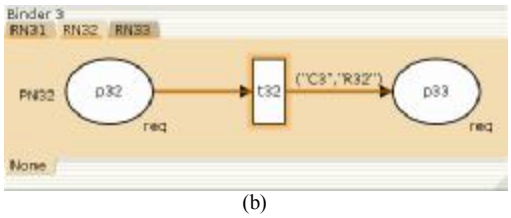
$RN_{23}= (PN_{23}, LE_4)$
$LE_4= (\phi)$

شکل ۶-۳: شبکه نیازمندیهای R_{21}, R_{22}, R_{23} سیستم مدیریت هتل



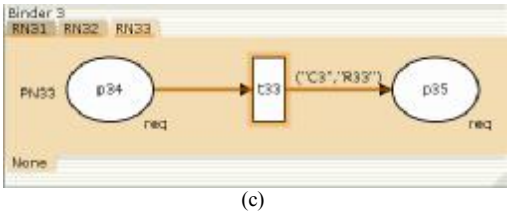
(a)

$RN_{31}= (PN_{31}, LE_5)$
$LE_5= (\phi)$



(b)

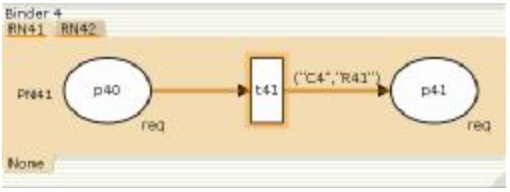
$RN_{32}= (PN_{32}, LE_6)$
$LE_6= (\phi)$



(c)

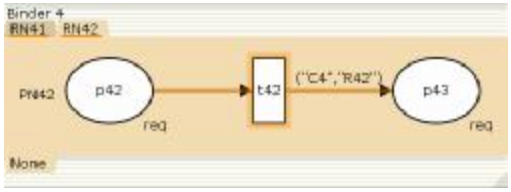
$RN_{33}= (PN_{33}, LE_7)$
$LE_7= (\phi)$

شکل ۶-۴: شبکه نیازمندیهای R_{31}, R_{32}, R_{33} سیستم مدیریت هتل



(a)

$RN_{41}= (PN_{41}, LE_8)$
$LE_8= (\phi)$



(b)

$RN_{42}= (PN_{42}, LE_9)$
$LE_9= (\phi)$

شکل ۶-۵: شبکه نیازمندیهای R_{41}, R_{42} سیستم مدیریت هتل

همان طور که از شکلهای ٦-٢ تا ٦-٥ مشخص است هر گذر مربوط به شبکه نیازمندی، نشانه‌ای را تولید خواهد کرد که متغییر اول آن شناسه دغدغه مالک و دومین متغییر شناسه خود نیازمندی است. در گامهای بعدی شرح داده خواهد شد که چگونه به جای ذکر تک تک شناسه دغدغه‌ها، از نشانه وارد شده به گذر استفاده شود تا متغییر اول به صورت اتوماتیک مقدار دهی شود.

۶-۴-۴- مرحله چهارم

حال که شبکه نیازمندیها تشکیل شدند باید ترتیب‌های اجرا برای هر دغدغه شناسائی شوند. شناسائی ترتیب اجرا برای یک دغدغه از طریق بررسی مشخصات هر دغدغه و در نظر گرفتن شبکه نیازمندیهای آنها حاصل

می‌شود. یک شیوه‌ای که می‌توان از آن استفاده کرد این است که مشخصات هر دغدغه خوانده شده و بر حسب ترتیب ظاهر شدن افعال مربوط به شبکه نیازمندیها، شبکه نیازمندیها در کنار یکدیگر قرار داده شوند تا اینکه یک ترتیب اجرا حاصل شود. البته ممکن است نیاز باشد که این عمل چندین بار تکرار شود تا کلیه حالتهایی که باید به تحقق دغدغه منجر شود، کشف شوند. یا اینکه می‌توان به جای چندین بار تکرار، در هر بار رسیدن به پایان یک ترتیب اجرا، عمل عقبگرد را انجام داد. عمل عقبگرد تا جائی باید ادامه پیدا کند که مشخص شود می‌توان یک شبکه نیازمندی را جایگزین کرد که منجر به یک ترتیب اجرایی می‌شود که جزء ترتیب‌های اجرا کشف شده نیست. به عنوان مثال اگر یک مرحله عمل عقبگرد در یک ترتیب اجرا کشف شده صورت بپذیرد و مشاهده شد که در این حالت می‌تواند یک شبکه نیازمندی دیگری جایگزین کرد که متفاوت از حالت کشف شده است باید عمل عقبگرد را لغو کرد و رو به جلو حرکت کرد. با توجه به این شیوه، ترتیب‌های اجرا برای سیستم مدیریت هتل به شرح زیر است:

- در دغدغه C_1 ، اول باید اتاقها بررسی شده و بعد عمل رزرواسیون انجام بپذیرد بنابراین ترتیب اجرا برای دغدغهC_1 به صورت زیر است:

$$EO_{11}= (RN_{11}, RN_{12})$$

- در دغدغه C_2 ، ابتدا تخصیص اتاق صورت پذیرفته بعد رزرواسیون مشتری حذف شده و نهایتا یک صورت‌حساب اولیه برای آن ایجاد می‌شود. بنابراین ترتیب اجرا برای دغدغه C_2 به صورت زیر خواهد بود:

$$EO_{21}= (RN_{21}, RN_{22}, RN_{23})$$

- در دغدغه C_3 ، ابتدا باید متصدی صورت‌حساب مشتری را محاسبه کرده و بعد از اینکه مشتری آن را پرداخت کرد اتاقی که به نام مشتری ثبت شده بود را از حالت پر بودن خارج کند. با توجه به این مشخصات، ترتیب اجرا به صورت زیر خواهد بود:

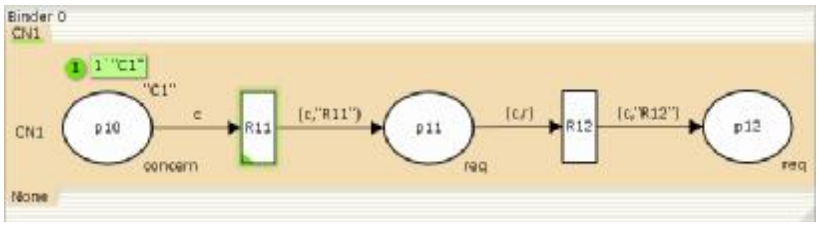
$$EO_{31}= (RN_{31}, RN_{32}, RN_{33})$$

- در دغدغه C_4 ، ابتدا باید عملیات انجام شده در سیستم تشخیص داده شده و نهایتا اطلاعات مربوط به آن عملیات در فایل ذخیره شود. بنابراین ترتیب اجرا به صورت زیر خواهد بود:

$$EO_{41}= (RN_{41}, RN_{42})$$

۶-۴-۵- مرحله پنجم

با توجه به اینکه دو مجموعه اصلی برای ایجاد یک شبکه دغدغه، در این مرحله قابل تشکیل هستند شبکه دغدغه‌های سیستم مدیریت هتل را می‌تواند ایجاد کرد. شبکه دغدغه‌های سیستم مدیریت هتل به ترتیب در شکلهای ۶-۶ تا ۶-۹ به تصویر کشیده شده است. همان طور که در شکل ۶-۶ نشان داده شده است شبکه دغدغه CN_1 از ادغام دو شبکه نیازمندی نشان داده شده در شکل ۶-۲ ایجاد شده است. با توجه به اینکه در گام قبلی برای دغدغه C_1 یک ترتیب اجرا کشف شده بود بنابراین یک نشانه در مکان اولیه (p_{10}) قرار داده شده است.



$CN_1= (SoR_1, SoE_1)$
$SoR1=(RN_{11}, RN_{12})$, $SoE_1= (EO_{11})$
$EO_{11}= (RN_{11}, RN_{12})$

شکل ۶-۶: شبکه دغدغه C_1 برای سیستم مدیریت هتل

نوع مکان اول از شبکه دغدغه CN_1 ، concern است که یک متغیر از نوع رشته است و به صورت زیر در ابزار CPN/Tools تعریف می‌شود.

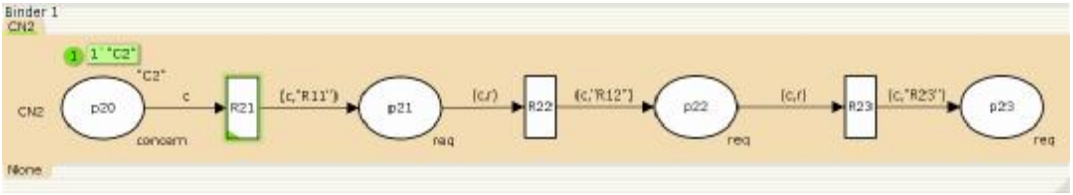
```
var concern:STRING;
```

در نشانه باید شناسه دغدغه مرتبط با شبکه دغدغه قرار داده شود تا در هنگام مانیتورینگ بتوان عمل تشخیص را انجام داد. همچنین در روی اولین کمان خروجی از مکان p_{10} یک متغییر از نوع رشته به نام c تعریف شده تا بتوان عمل انتقال نشانه را انجام داد.

```
var c:STRING;
```

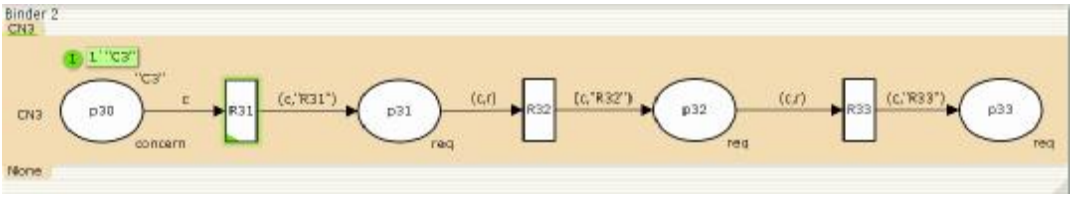
در مورد inscription بر روی کمانهای خروجی از گذرهای متعلق به شبکه نیازمندیها نیز این تغییر صورت پذیرفته است که به جای ذکر شناسه دغدغه به صورت صریح در خروجی از نشانه وارد شده توسط inscription کمان ورودی به گذر استفاده می‌شود (همان متغییر c) زیرا نشانه ورودی خود حامل شناسه دغدغه است. در مورد کمانهایی که نشانه را از یک مکان شبکه نیازمندی به یک گذر متعلق به شبکه نیازمندی دیگر در این شبکه دغدغه انتقال می‌دهد علاوه بر نشانه که با متغییر c انتقال داده می‌شود از متغییر r نیز برای انتقال نام شبکه نیازمندی استفاده می‌شود.

```
var r:STRING;
```



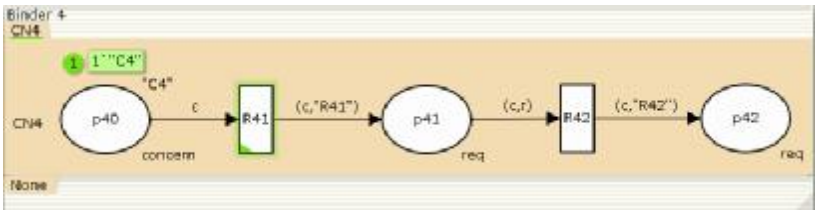
$CN_2= (SoR_2, SoE_2)$
$SoR_2=(RN_{21}, RN_{22}, RN_{23})$, $SoE_2= (EO_{21})$
$EO_{21}= (RN_{21}, RN_{22}, RN_{23})$

شکل ۶-۷: شبکه دغدغه C_2 برای سیستم مدیریت هتل



$CN_3= (SoR_3, SoE_3)$
$SoR_3=(RN_{31}, RN_{32}, RN_{33})$, $SoE_3= (EO_{31})$
$EO_{31}= (RN_{31}, RN_{32}, RN_{33})$

شکل ۶-۸: شبکه دغدغه C_3 برای سیستم مدیریت هتل



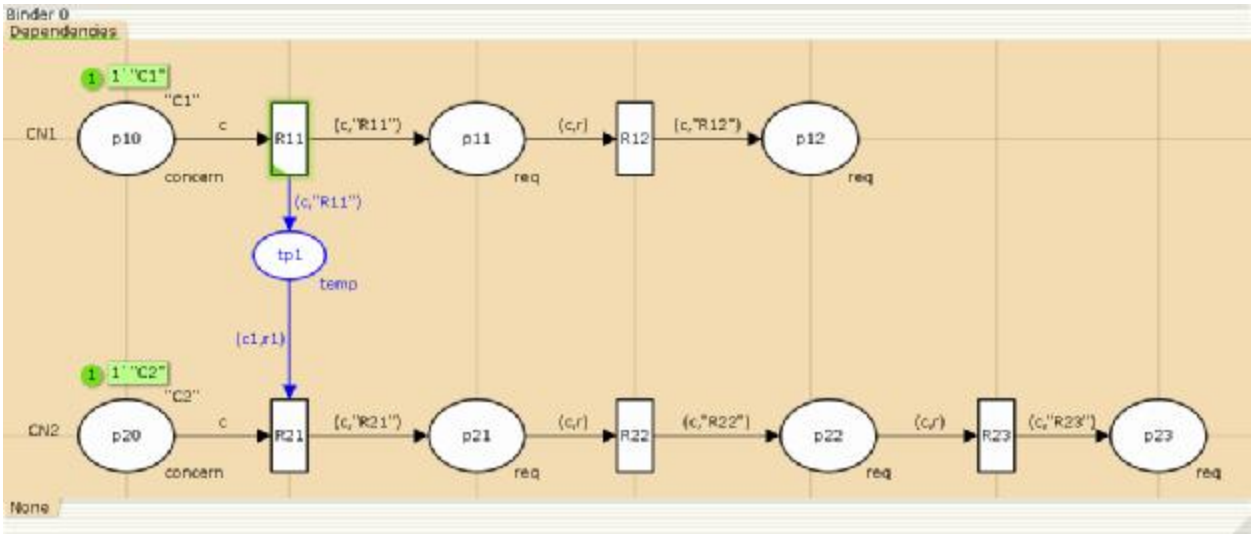
$CN_4= (SoR_4, SoE_4)$
$SoR_4=(RN_{41}, RN_{42})$, $SoE_4= (EO_{41})$
$EO_{41}= (RN_{41}, RN_{42})$

شکل ۶-۹: شبکه دغدغه C_4 برای سیستم مدیریت هتل

۶-۴-۶- مرحله ششم

در این مرحله با توجه به مستندات و مشخصات دغدغه‌ها و نیازمندیها، وابستگی‌ها، محدودیت‌ها و ارتباطات برای سیستم مدیریت هتل به صورت زیر تعیین می‌شود.

۱. در سیستم مدیریت هتل ابتدا باید عمل بررسی وجود اتاق انجام شود تا اینکه بعد عمل تخصیص اتاق صورت بپذیرد. با توجه به اینکه عمل بررسی اتاق متناظر با شبکه نیازمندی RN_{11} است و تخصیص اتاق با شبکه نیازمندی RN_{21} نمایش داده می‌شود ابتدا باید شبکه نیازمندی RN_{11} انجام شده و بعد شبکه نیازمندی RN_{21} انجام شود. یعنی اینکه شبکه نیازمندی RN_{21} محدود به اجرا شدن شبکه نیازمندی RN_{11} است بنابراین باید بین این دو شبکه نیازمندی در مدل نهائی ارتباطی لحاظ شود. این ارتباط در شکل ۶-۱۰ نمایش داده شده است



شکل ۶-۱۰: نمایش وابستگی بین شبکه‌های نیازمندی RN_{11}, RN_{21}

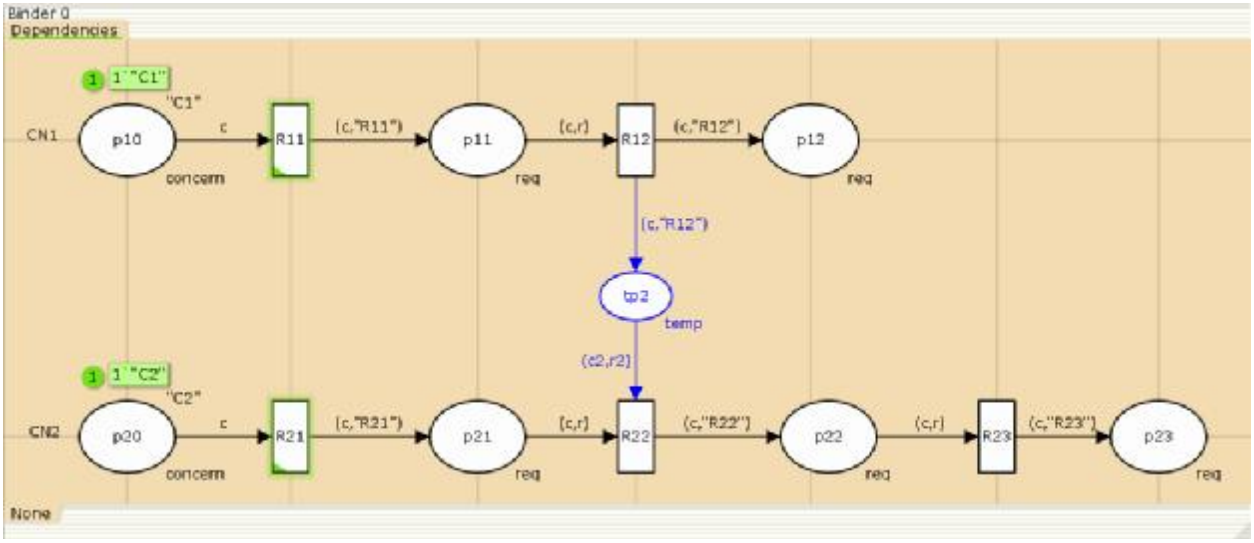
در شکل ٦-۱۰ مکان موقت (tp_1) با نوع temp مشخص شده است که مانند نوع req است و از دو متغییر که یکی در برگیرنده شناسه دغدغه و دیگری شناسه نیازمندی است، تشکیل شده است و به صورت زیر تعریف می‌شود.

colset temp=product STRING*STRING;

inscription کمان خروجی از گذر R_{11} مشابه inscription کمان خروجی دیگر آن است. اما inscription کمان خروجی از مکان tp_1 کمی متفاوت‌تر از کمان ورودی است (متغییر c به متغییر c_1 و شناسه نیازمندی به متغییر r_1 تبدیل شده است). دلیل اول که همان علت تغییر نام c به c_1 است ناشی از این است که ابزار در هنگام مانیتورینگ یک گذر باید نشانه‌ها را از متغییرهایی با نامهای مختلف دریافت کند تا اینکه بتواند عمل تشخیص را انجام دهد. یعنی اینکه کمانهایی که به یک گذر وارد می‌شوند باید نشانه‌ها را با متغییرهایی با نام مختلف ارائه کنند. علت تبدیل شدن شناسه نیازمندی به r_1 فقط یک جایگذاری به منظور تسهیل طراحی و کمک به نوشته نوشته شده برای مانیتورینگ است.

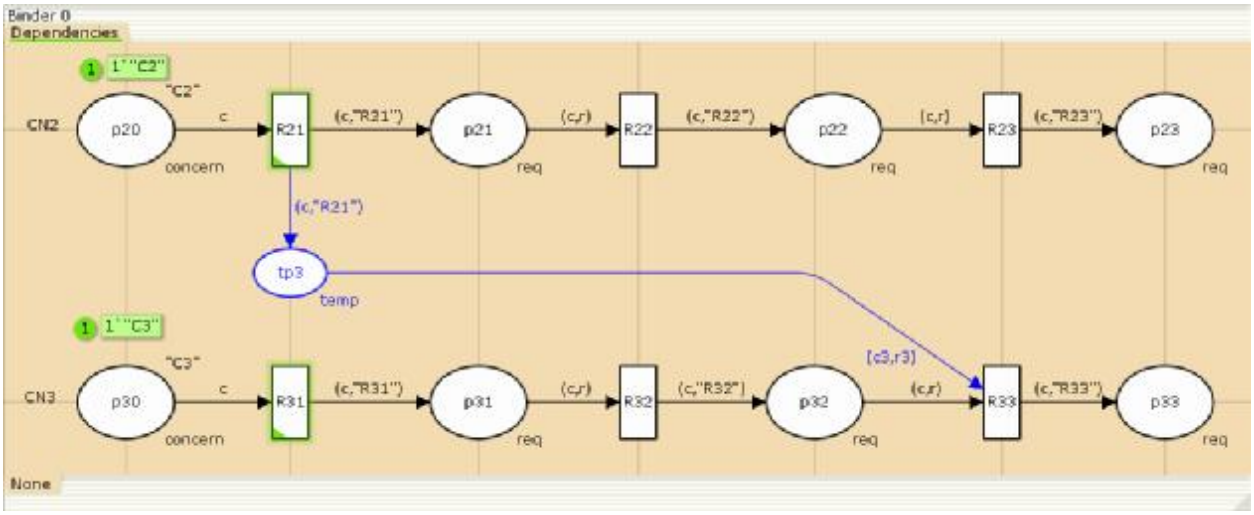
نکته‌ای که لازم است رعایت شود نامگذاری متغییرهای r, c هر مکان موقت است. هر نشانه که می‌خواهد از طریق کمان خروجی از مکان موقت خارج شود باید از طریق متغییرهای r, c خارج شود که دارای اندیکس مربوط به آن مکان موقت است. به عنوان مثال متغییرهای r, c یک مکان موقت به نام tp_{10} به صورت r_{10}, c_{10} تعریف شده و نشانه وابستگی از طریق اینها به سایر گذرها انتقال داده می‌شود. همان طور که قبلا هم ذکر شد این موضوع علاوه بر اینکه برای ابزار مدلسازی و شبیه‌سازی لازم و ضروری است مدلسازی را قابل فهم‌تر نیز می‌کند.

۲. در سیستم مدیریت هتل، باید عمل رزرواسیون انجام شده باشد تا اینکه متصدی هتل بتواند رزرواسیون را حذف کند. با توجه به اینکه شبکه نیازمندی RN_{12} متناظر با رزرواسیون و شبکه نیازمندی RN_{22} متناظر با حذف رزرواسیون است. شبکه نیازمندیRN_{22} محدود به اجرا شدن شبکه نیازمندی RN_{12} است. بنابراین بین این دو شبکه نیازمندی باید وابستگی در مدل نهائی لحاظ شود. این وابستگی در شکل ۶-۱۱ به تصویر کشیده شده است.



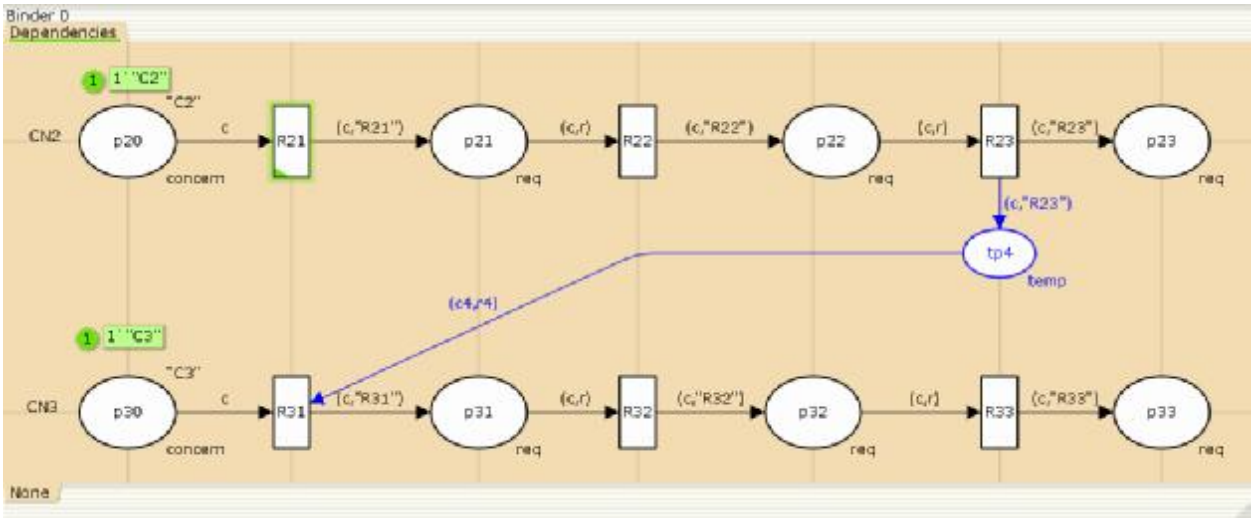
شکل ۶-۱۱: نمایش وابستگی بین شبکه‌های نیازمندی RN_{12}, RN_{22}

۳. برای اینکه بتوان در سیستم مدیریت هتل، اتاقی را از حالت پر بودن خارج کرد باید آن اتاق قبلا به مشتری تخصیص داده شده باشد. طبق این دید و شبکه‌های نیازمندی متناظر با هر دو عمل، ابتدا باید شبکه نیازمندی RN_{21} اجرا شود تا اینکه بعد شبکه نیازمندی RN_{33} بتواند قابل اجرا شود. بنابراین نیاز به یک وابستگی در مدل نهائی میان این دو شبکه نیازمندیها است. این وابستگی در شکل ۶-۱۲ به تصویر کشیده شده است.



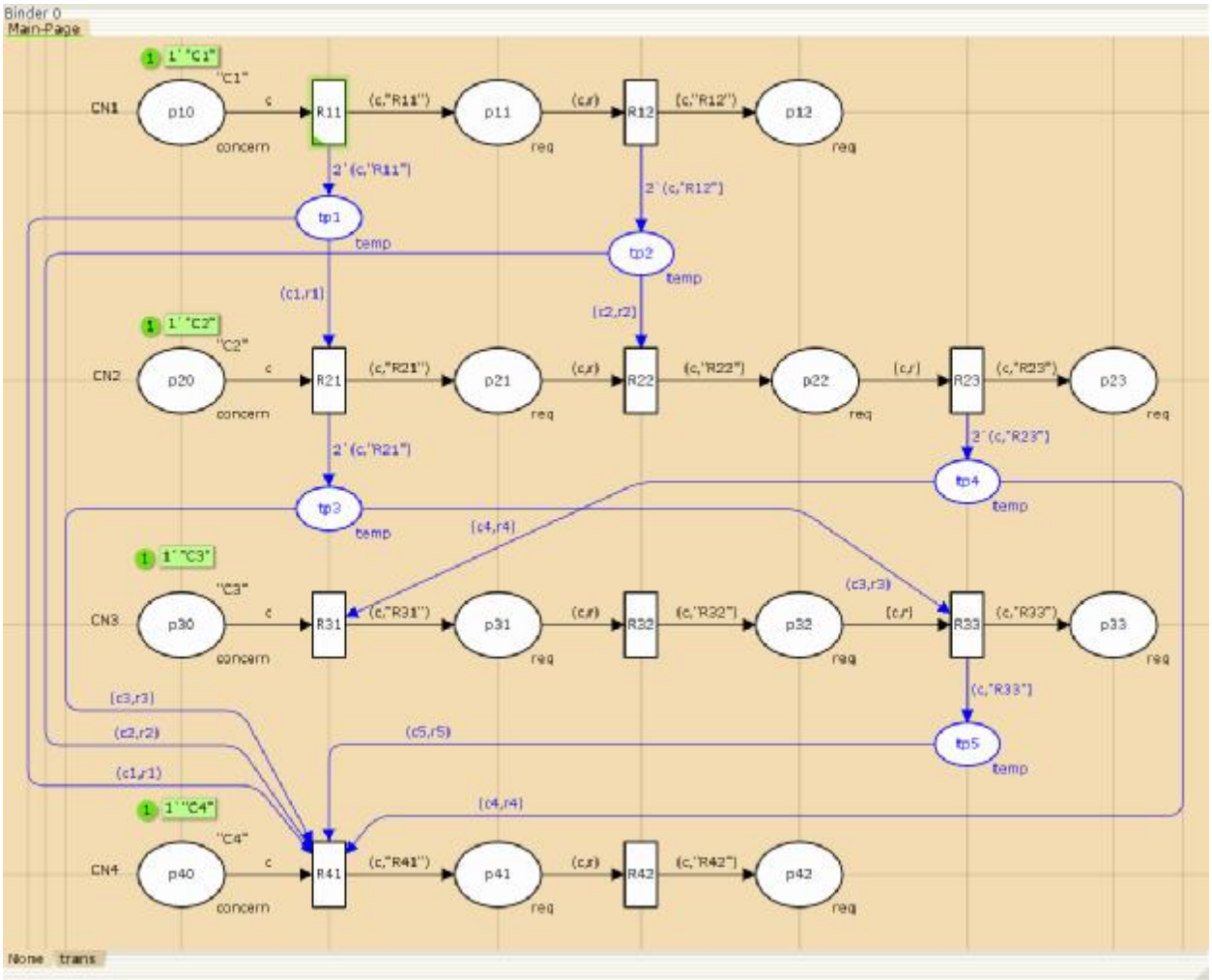
شکل ۶-۱۲: نمایش وابستگی بین شبکه‌های نیازمندی RN_{21}, RN_{33}

٤. محاسبه صورت حساب نهائی برای مشتری مستلزم صادر شدن یک صورت حساب اولیه است که در هنگام ثبت نام برای مالک اتاق ایجاد می‌شود. با در نظر گرفتن شبکه‌های نیازمندی این دو عمل که به ترتیب با RN_{23} و RN_{31} نمایش داده می‌شوند ابتدا باید شبکه نیازمندی RN_{23} و بعد شبکه نیازمندی RN_{31} تحقق پیدا کند. بنابراین باید بین این دو شبکه نیازمندی وابستگی در مدل نهائی در نظر گرفته شود. این وابستگی در شکل ۶-۱۳ نمایش داده شده است.



شکل ۶-۱۳: نمایش وابستگی بین شبکه‌های نیازمندی RN_{23}, RN_{31}

۵. در فرآیند واقعه‌نگاری اولین گام تشخیص این موضوع است که آیا عملیاتی انجام شده یا نه؟ و بعد از تشخیص رخ دادن یک عملیات یا مجموعه‌ای از عملیات، اطلاعات آن عملیات در فایل یا پایگاه داده ذخیره می‌شود. عمل اول که همان تشخیص است (با نام واقعه‌نگاری به آن ارجاع می‌شود) با شبکه نیازمندی RN_{41} نمایش داده می‌شود و سایر عملیات‌ها که باید واقعه‌نگار منتظر آنها باشد عبارتند از: بررسی قابل دسترس بودن اتاق، رزرواسیون، تخصیص اتاق، ایجاد صورت حساب اولیه و خالی کردن اتاق. شبکه‌های نیازمندی متناظر با این عملیات‌ها (نیازمندیها) با RN_{11}, RN_{12}, RN_{21}, RN_{23}, RN_{33} نمایش داده می‌شوند. بنابراین باید بین این شبکه نیازمندیها و شبکه نیازمندی RN_{41} در مدل نهائی وابستگیهایی وجود داشته باشد. این وابستگی‌ها در شکل ۶-۱۴ به تصویر کشیده شده‌اند.



شکل ۶-۱۴: نمایش وابستگی‌ها در سیستم مدیریت هتل

شکل ۶-۱۴ علاوه بر نمایش وابستگیهای مربوط به این شبکه نیازمندیها، سایر وابستگیها نمایش داده شده در شکلهای ۶-۱۰ تا ۶-۱۳ را نیز نشان می‌دهد زیرا وابستگیهای شبکه نیازمندیهای $RN_{11}, RN_{12}, RN_{21}, RN_{23}$ با RN_{41} باید از طریق مکانهای موقت قبلی ایجاد شده انجام شود و وابستگی RN_{33} با RN_{41} نیز می‌تواند از طریق ایجاد یک مکان موقت به نام tp_5 صورت بپذیرد.

در شکل ۶-۱۴ inscription های مربوط به کمان ورودی به مکانهای tp1, tp2, tp3, tp4 تغییر پیدا کرده است زیرا این مکانها برای اعمال دو وابستگی به کار می‌روند و بنابراین باید توسط کمان ورودی دو نشانه وارد مکان شود که این موضوع با اعلان "2`" تعریف می‌شود (اگر یک مکان موقت سه وابستگی را تداعی کند بنابراین باید اعلان "3`" برای آن ذکر شود).

۶-۴-۷- مرحله هفتم

در این مرحله باید گذرهایی که دارای دو یا بیشتر کمان ورودی هستند را مشخص کرد و نوشته مربوطه به شناسائی نشانه‌های نیازمندیها و دغدغه‌ها را در آنها وارد کرد تا اینکه بتوان خروجی لازم را گرفت. بدین منظور ابتدا یک گروه در ابزار ایجاد شده (به نام trans) و گذرهایی که دارای دو یا بیشتر کمان ورودی هستند جزء این گروه قرار داد می‌شود. سپس از بخش tools box منوی monitoring را انتخاب کرده و از گزینه‌های موجود "write in file" را به گروه گذرها اعمال کرده تا اینکه در بخش monitoring ابزار، اعلانی همنام گروه ایجاد شود و بتوان نوشته[1] لازم را به آن وارد کرد. این نوشته در شکل ۶-۱۵ به تصویر کشیده شده است.

بعد از اینکه نوشته، نوشته شد و مدل نهائی سیستم توسط شبیه‌ساز ابزار CPN/Tools اجرا شد. خروجی به صورت شکل ۶-۱۶ حاصل می‌شود.

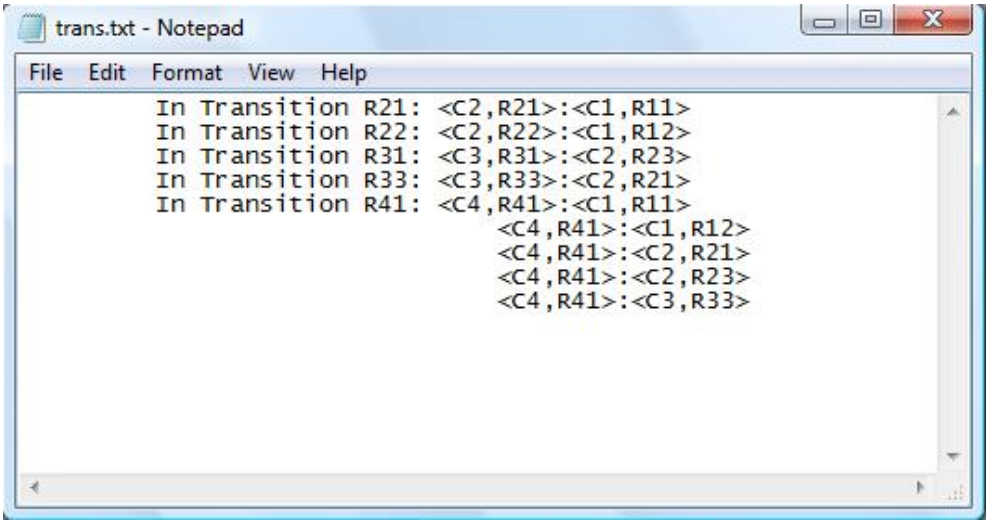
در شکل ٦-١٦، در هر ردیف شناسه گذر و در جلوی آن دو جفت تاپل دوتائی که با ":" از یکدیگر جدا شده‌اند، نشان داده شده است. این دو تاپل دوتائی نمایش دهنده این است که دو دغدغه ذکر شده در این دو تاپل دوتائی می‌توانند به عنوان یک دغدغه مداخله‌ای در نظر گرفته شوند (در واقع نامزدی برای جنبه شدن هستند). همچنین در ردیف مربوط به گذر R_{41} پنج تا وابستگی نشان داده شده است که نشان دهنده میزان درهم‌تنیدگی آن دغدغه است.

با توجه به خروجی، قابل مشاهده است که بعضی از دغدغه‌ها در گذرهای مختلف به عنوان نامزد در نظر گرفته شده‌اند. این عمل باعث می‌شود که در مراحل بعدی بتوان موجودیت‌های منطقی تحت تاثیر جنبه را شناسائی کرد. البته ذکر چندین بار نام یک دغدغه به عنوان یک جنبه نامزد هیچ تاثیری مثبت و منفی بر ماهیت آن ندارد. زیرا از تمامی آنها فاکتورگیری خواهد شد.

[1] Script

```
Monitors
  trans
    Type: Write in file
    Nodes ordered by pages
    Init
    Predicate
    Observer
fun obs (bindelem) =
let
  fun obsBindElem (Main'R21 (1, {c,c1,r1})) =
if c<>c1 then
"R21: <"^c^",R21>:"^"<"^c1^","^r1^">\t"
else
""
    | obsBindElem (Main'R22 (1, {c,c2,r,r2})) =
if c<>c2 then
"R22: <"^c^",R22>:"^"<"^c2^","^r2^">\t"
else
""
    | obsBindElem (Main'R31 (1, {c,c4,r4})) =
if c<>c4 then
"R31: <"^c^",R31>:"^"<"^c4^","^r4^">\t"
else
""
    | obsBindElem (Main'R33 (1, {c,c3,r,r3})) =
if c<>c3 then
"R33: <"^c^",R33>:"^"<"^c3^","^r3^">\t"
else
""
    | obsBindElem (Main'R41 (1,
                    {c,c1,c2,c3,c4,c5,r1,r2,r3,r4,r5})) =
"R41: <"^c^",R41>:"^"<"^c1^","^r1^">\t"^"<"^c^",R41>:"^"<"^c2^","^r2^">\t"^
"<"^c^",R41>:"^"<"^c3^","^r3^">\t"^"<"^c^",R41>:"^"<"^c4^","^r4^">\t"^
"<"^c^",R41>:"^"<"^c5^","^r5^">"
    | obsBindElem _ = ""
in
  "\n\t In Transition "^(obsBindElem bindelem)
end
    Stop
```

شکل ۶-۱۵: نوشته مربوط به مانیتورینگ گذرها در سیستم مدیریت هتل



شکل ۶-۱۶: خروجی حاصل از مانیتورینگ گذرها در سیستم مدیریت هتل

دلیل اینکه دغدغه‌های شناسائی شده در این مرحله دغدغه نامزد نامیده می‌شوند این است که باید موجودیت‌های منطقی متعلق به هر دو شبکه نیازمندی موجود در دو تاپل دوتائی متعلق به یک گذر استخراج شود و در صورت وجود حداقل یک موجودیت منطقی می‌توان آن را به عنوان یک جنبه در نظر گرفت. به عنوان مثال در دو تاپل دوتائی $<C_2,R_{21}>:<C_1,R_{11}>$ باید بین موجودیت‌های منطقی R_{21}, R_{11} یک اشتراک وجود داشته باشد تا اینکه C_1, C_2 بتوانند به عنوان جنبه در نظر گرفته شوند.

۶-۴-۸- مرحله هشتم

در این مرحله باید موجودیت‌های منطقی مربوط به شبکه نیازمندیهای استخراج شوند که شناسه نیازمندیهای آنها در خروجی مانیتورینگ (شکل ۶-۱۶) وجود دارد. موجودیت‌های منطقی برای خروجی حاصل از مانیتورینگ مدل نهائی سیستم مدیریت هتل به صورت جدول ۶-۱ است.

جدول ٦-١: موجودیت‌های منطقی نیازمندیهای سیستم مدیریت هتل

نیازمندی	شبکه نیازمندی	موجودیت‌های منطقی
R_{11}	RN_{11}	اتاق
R_{12}	RN_{12}	رزرواسیون
R_{21}	RN_{21}	اتاق
R_{22}	RN_{22}	صورت حساب
R_{23}	RN_{23}	صورت حساب
R_{31}	RN_{31}	اتاق
R_{33}	RN_{33}	اتاق
R_{41}	RN_{41}	اتاق، رزرواسیون، صورت حساب

حال دو تاپل دوتائی هر گذر بررسی می‌شود تا اینکه مشخص شود آیا موجودیت منطقی بین نیازمندیها مشخص شده در تاپل‌ها وجود دارد تا اینکه دغدغه‌های آنها به عنوان یک جنبه معرفی شوند.

گذر R_{21} : تاپلهای خروجی برای این گذر به صورت $<C_2,R_{21}>:<C_1,R_{11}>$ است. موجودیت منطقی شناسائی شده برای R_{11} اتاق است و برای R_{21} نیز اتاق است بنابراین یک موجودیت منطقی به نام اتاق یافت شد که در هر دو نیازمندی وجود دارد. این موجودیت منطقی دارای مشکل درهم تنیدگی است در این صورت دغدغه‌های C_1, C_2 که این مشکل را در نیازمندیهای خود دارند به عنوان جنبه در نظر گرفته می‌شوند و به صورت $C_1=A_1$, $C_2=A_2$ نامگذاری می‌شوند.

گذر R_{22} : تاپلهای خروجی برای این گذر به صورت $<C_2,R_{22}>:<C_1,R_{12}>$ است. موجودیت منطقی شناسائی شده برای R_{12} رزرواسیون و برای R_{22} رزرواسیون است. پس یک موجودیت منطقی مشترک کشف شد و می‌توان دغدغه‌های مربوطه که C_1, C_2 هستند را به عنوان جنبه در نظر گرفت. نامگذاری جنبه در این مورد نیاز نیست زیرا این دو دغدغه قبلا به عنوان جنبه نامگذاری شده‌اند.

گذر R_{31} : تاپلهای خروجی برای این گذر به صورت $<C_3,R_{31}>:<C_2,R_{23}>$ است. موجودیت منطقی شناسائی شده برای R_{23} صورت حساب و برای R_{31} صورت حساب است. پس یک موجودیت منطقی مشترک وجود دارد و می‌توان C_2, C_3 را به عنوان جنبه در نظر گرفت. با توجه به اینکه C_2 قبلا کشف شده است دیگر نیازی به نامگذاری نیست ولی باید C_3 نامگذاری شود که آن به صورت A_3 نامگذاری می‌شود.

گذر R_{33} : تاپلهای خروجی برای این گذر به صورت $<C_3,R_{33}>:<C_2,R_{21}>$ است. موجودیت منطقی شناسائی شده برای R_{21} اتاق است و برای R_{33} نیز اتاق می‌باشد. پس با توجه به اشتراک در موجودیت‌های منطقی، می‌توان C_2, C_3 را به عنوان جنبه در نظر گرفت. به دلیل شناسائی شدن این دغدغه‌ها قبلا به عنوان جنبه دیگر نیازی به نامگذاری مجدد نیست.

گذر R_{41} : این گذر، پنج تا دو تاپلی دارد که هر کدام باید به صورت جداگانه بررسی شوند بنابراین برای تک تک آنها داریم:

- *دو تاپل* $<C_4,R_{41}>:<C_1,R_{11}>$

موجودیت منطقی شناسائی شده برای R_{41} اتاق، رزرواسیون و صورت حساب است و برای R_{11} نیز موجودیت منطقی اتاق شناسائی شده است. پس دو نیازمندی در موجودیت منطقی اتاق با یکدیگر اشتراک دارند و دغدغه‌های آنها می‌تواند به عنوان جنبه در نظر گرفته شود. بدلیل شناسائی C_1 به عنوان جنبه، فقط C_4 به صورت A_4 نامگذاری می‌شود.

- *دو تاپل* $<C_4,R_{41}>:<C_1,R_{12}>$

موجودیت منطقی نیازمندی R_{12} رزرواسیون و موجودیت‌های منطقی نیازمندی R_{41} اتاق، رزرواسیون و صورت حساب است. پس به دلیل اشتراک در موجودیت منطقی رزرواسیون دو دغدغه C_1, C_4 به عنوان جنبه در نظر گرفته می‌شوند ولی بدلیل اینکه قبلا به عنوان جنبه شناسائی شده‌اند نامگذاری نمی‌شوند.

- *دو تاپل* $<C_4,R_{41}>:<C_2,R_{21}>$

موجودیت منطقی شناسائی شده برای R_{21} اتاق و برای R_{41} اتاق، رزرواسیون و صورت حساب است. بنابراین به دلیل وجود اشتراک، دو دغدغه C_2, C_4 به عنوان جنبه در نظر گرفته می‌شوند ولی نامگذاری نمی‌شوند.

- دو تاپل $<C_4,R_{41}>:<C_2,R_{23}>$

موجودیت منطقی نیازمندی R_{23} رزرواسیون و موجودیت‌های منطقی R_{41} اتاق، رزرواسیون و صورت حساب است. بنابراین به دلیل وجود اشتراک، دو دغدغه C_2, C_4 به عنوان جنبه در نظر گرفته می‌شوند ولی نامگذاری نمی‌شوند.

- دو تاپل $<C_4,R_{41}>:<C_3,R_{33}>$

موجودیت منطقی نیازمندی R_{33} اتاق و موجودیت‌های منطقی R_{41} اتاق، رزرواسیون و صورت حساب است. بنابراین به دلیل وجود اشتراک، دو دغدغه C_3, C_4 به عنوان جنبه در نظر گرفته می‌شوند ولی نامگذاری نمی‌شوند.

حال بعد مشخص شدن جنبه‌ها و عامل جنبه شدن آنها که یک موجودیت منطقی است، آنها در یک جدول لیست می‌شود (جدول ۶-۲).

جدول ٦-٢: جنبه‌ها و موجودیت منطقی منجر به جنبه شدن دغدغه‌ها در سیستم مدیریت هتل

جنبه	موجودیت منطقی مشترک (دارای مشکل درهم‌تنیدگی)
A_1	اتاق
A_2	اتاق
A_1	رزرواسیون
A_2	رزرواسیون
A_2	صورت حساب
A_3	صورت حساب
A_2	اتاق
A_3	اتاق
A_1	اتاق
A_4	اتاق
A_1	رزرواسیون
A_4	رزرواسیون
A_2	اتاق
A_4	اتاق
A_2	رزرواسیون
A_4	رزرواسیون
A_3	اتاق
A_4	اتاق

حال می‌توان از ستون جنبه‌ها فاکتورگیری کرده و ستون مربوط به موجودیت‌های منطقی مورد اجتماع قرار داد تا اینکه مشخص شود هر جنبه چه موجودیت‌های منطقی را تحت تاثیر قرار می‌دهد. نتیجه حاصله در جدول ۶-۳ نشان داده شده است.

جدول ۶-۳: ارتباط جنبه‌ها با موجودیت‌های منطقی در سیستم مدیریت هتل

	اتاق	رزرواسیون	صورت حساب
A1	√	√	
A2	√	√	√
A3	√		√
A4	√	√	√

۶-۵- خلاصه و نتیجه‌گیری

این فصل یک مطالعه موردی به نام سیستم مدیریت هتل را معرفی کرد تا اینکه بتوان روش پیشنهادی برای شناسائی جنبه‌ها را بر روی آن اعمال کرد. مطالعه موردی علاوه بر اینکه باعث آشنائی با نحوه اعمال روش پیشنهادی بر روی یک پروژه واقعی شد، نشان داد که روش پیشنهادی خروجی مورد انتظار را تولید می‌کند. بدین منظور ابتدا مستند سیستم ارائه شد و بعد مرحله هشت گانه‌ای که برای استخراج جنبه‌ها معرفی شده بود با کمک ابزار CPN/Tools، به سیستم مدیریت هتل اعمال شد. خروجی حاصل، جنبه‌های سیستم و موجودیت‌های منطقی بودند که توسط جنبه‌ها تحت تاثیر قرار می‌گرفتند. بنابراین با توجه به اینکه سیستم مدیریت هتل چهار دغدغه اصلی داشت خروجی حاصل از روش پیشنهادی نشان داد که هر چهار دغدغه باید به عنوان جنبه در نظر گرفته شوند زیرا این دغدغه‌ها دارای موجودیت‌های منطقی هستند که مشکل درهم‌تنیدگی را ایجاد می‌کنند.

نتایج بدست آمده از اعمال روش پیشنهادی برای سیستم مدیریت هتل نشان داد که روش پیشنهادی قابلیت تحقق شاخص‌های مشخص شده را دارد زیرا:

- جنبه‌های اصلی در سیستم که چهار تا بودند به خوبی شناسائی شدند.
- تمام دغدغه‌های سیستم با استفاده از تعریف مطرح شده برای دغدغه تعریف شدند. یعنی تعریف توانست انواع دغدغه‌های سیستم مدیریت هتل را پوشش دهد.
- جنبه‌ها فقط نیازهای وظیفه‌مندی یا غیروظیفه‌مندی در نظر گرفته نشدند زیرا در این سیستم جنبه واقعه‌نگاری یک نیازمندی غیروظیفه‌مندی و جنبه رزرو اتاق یک نیازمندی وظیفه‌مندی است.

- با توجه به اینکه کلیه مفاهیم و موجودیت‌ها تعاریف رسمی دارند و از طرفی کلیه قابلیت‌های ارائه شده توسط شبکه‌پتری منطق ریاضی دارد که از آنها پشتیبانی می‌کند بنابراین روش رسمی است (البته کمی ابهام در مورد رسمی بودن ارتباطات مشاهده می‌شود که آن نیز در صورت دسته‌بندی کردن قابل رسمی سازی است).

فصل هفتم
نتیجه‌گیری

در این فصل ابتدا یک مرور بر کل مطالب ذکر شده در این تحقیق خواهیم داشت سپس مزایا و معایب روش پیشنهادی برای شناسائی جنبه‌ها ذکر می‌شود و نهایتا نیز فرصت‌های تحقیقاتی حاصل آمده از این تحقیق ذکر خواهد شد.

۷-۱- مروری بر تحقیق

هدف از این تحقیق ارائه یک روش رسمی مبتنی بر شبکه پتری برای شناسائی جنبه‌ها در فاز مهندسی نیازمندیهای جنبه‌گرا بود. در جهت رسیدن به این هدف، فعالیت‌های مختلفی انجام شده است که این فعالیت‌ها در قالب شش فصل در این مستند ارائه شدند.

در فصل اول، به معرفی موضوع تحقیق پرداخته شد. بدین منظور ابتدا مقدمه‌ای ارائه شد. سپس صورت مسئله تعریف گشت که در آن مشخص شد در این تحقیق جواب چه سئوالی پاسخ داده می‌شود. در ادامه نیز کارهای تحقیقاتی انجام شده در این حوزه به صورت اجمالی عنوان شدند و ذکر شد که نتایج تحقیقاتی مورد انتظار از این کار تحقیقاتی چیست.

در فصل دوم ادبیات تحقیق بیان شد. در این فصل، مفاهیم یا روشهایی که در این تحقیق با آنها سروکار خواهیم داشت یا آشنائی با آنها در درک و فهم موضوع مفید است، ذکر شدند. مفاهیم پایه موضوعاتی همچون دغدغه، دغدغه‌های مداخله‌ای، جداسازی دغدغه‌ها، خاصیت پیمانه‌ای و چند مفهوم دیگر بودند. همچنین تعریف منسجمی نیز برای دغدغه مداخله‌ای ارائه شد که تکمیل کننده دیگر تعریف‌های ارائه شده برای آن بود.

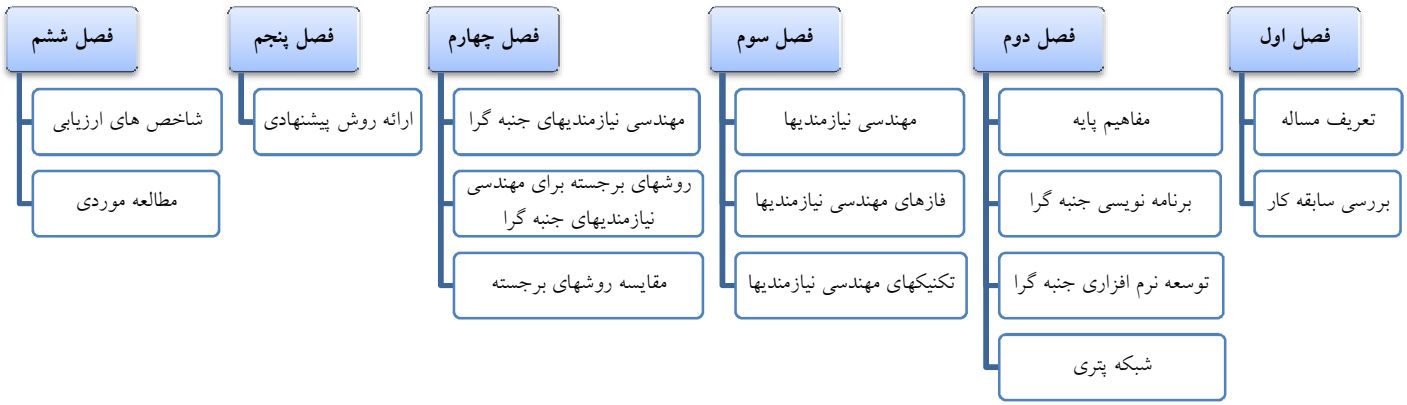
علاوه بر اینها، برنامه‌نویسی جنبه‌گرا به عنوان آغاز کننده مفهوم جنبه‌گرائی در حوزه مهندسی نرم‌افزار، معرفی و مزایا و معایب آن ذکر شدند. سپس به منظور آشنائی با هدف و ساختار کلی جنبه‌گرائی، توسعه نرم‌افزاری جنبه‌گرا که شامل فازهای مهندسی نیازمندیهای جنبه‌گرا، معماری سیستم جنبه‌گرا، طراحی و مدلسازی جنبه‌گرا، برنامه‌نویسی جنبه‌گرا و آزمایش جنبه‌گرا است، تشریح شد نهایتا نیز، شبکه پتری معرفی شد که یک تکنیک اصلی است که از آن در روش پیشنهادی بر شناسائی جنبه‌ها استفاده شده است.

در فصل سوم مهندسی نیازمندیها مورد تحلیل و بررسی قرار گرفت. بدین منظور ذکر شد که مهندسی نیازمندیها از پنج فعالیت اصلی تحت عنوان استخراج، مذاکره و تحلیل، مستندسازی، اعتبارسنجی و مدیریت تشکیل شده است. سپس هر فعالیت به صورت جداگانه تشریح شده و تکنیک‌های به کارگرفته برای هر یک از فعالیت‌ها معرفی و توضیحاتی پیرامون نحوه عملکرد هر تکنیک ذکر شد.

در فصل چهارم جهت شناخت مهندسی نیازمندیهای جنبه‌گرا، چندین روش پیشنهادی برجسته برای آن به طور کامل تشریح و مورد بررسی قرار گرفتند. این روشها علاوه بر اینکه در جهت رسیدن به دو هدف اصلی مهندسی نیازمندیهای جنبه‌گرا حرکت می‌کنند باعث پدید آمدن اهداف دیگری نیز برای مهندسی نیازمندیهای جنبه‌گرا می‌شوند که در حین تشریح روشها، آن اهداف نیز ذکر شدند. نهایتا نیز یک سری معیارهایی برای مقایسه این روشها مطرح شد و روشها بر اساس این معیارها با یکدیگر مقایسه شدند.

در فصل پنجم، روش پیشنهادی برای شناسائی جنبه‌ها مبتنی بر شبکه پتری ارائه شد. ابتدا مفاهیم پایه‌ی مورد استفاده در این روش پیشنهادی معرفی و تعریف شدند و مثالهایی برای هر یک از این مفاهیم ارائه شد تا بتوان مفاهیم را خوب درک کرد. سپس روش پیشنهادی تشریح شد و ذکر گردید این روش از هشت مرحله تشکیل شده است که انجام صحیح و مرتب این مراحل منجر به شناسائی جنبه‌ها می‌شود.

در فصل ششم به منظور بررسی اعتبار روش پیشنهادی و همچنین تشریح روش بر روی یک سیستم واقعی (بررسی قابلیت عملکرد آن)، یک مطالعه موردی تحت عنوان سیستم مدیریت هتل معرفی شد و همچنین شاخص‌هایی نیز برای ارزیابی روش پیشنهادی مشخص شدند. مراحل هشت گانه روش پیشنهادی برای شناسائی جنبه‌ها، یک به یک به آن (سیستم مدیریت هتل) اعمال شدند و نتایج حاصل از اعمال روش پیشنهادی بر روی مطالعه موردی نشان داده شده و میزان دستیابی به شاخص‌ها نیز بررسی شدند. در اعمال روش بر روی سیستم مدیریت هتل از یک ابزار مدلسازی شبکه پتری تحت عنوان CPN/Tools استفاده شد و ذکر شد که کلیه ابزارهایی که قابلیت مدلسازی، شبیه‌سازی و مانیتورینگ شبکه پتری را داشته باشند می‌توانند به عنوان یک ابزار پشتیبانی کننده این روش مطرح شوند (البته ممکن است بعضی از این ابزارها در بخش نوشته‌نویسی دارای کمبودهایی باشند). یک نمای کلی از مسیر و فعالیت‌های انجام شده در روند این تحقیق در شکل ۷-۱ به تصویر کشیده است.



شکل ۷-۱: مسیر طی شده در این تحقیق

۷-۲- مزایا و معایب

هر روش دارای مزایا و معایبی است که آن را مناسب یا نامناسب برای بکارگیری در یک مسئله خاص می‌گرداند. روش پیشنهادی برای شناسائی دغدغه‌ها با استفاده از شبکه پتری نیز از این امر مستثنا نیست و دارای مزایا و معایبی به شرح زیر است.

۷-۲-۱- مزایا

با توجه به اینکه روشهای بکارگیری در زمینه مهندسی نیازمندیهای جنبه‌گرا به صورت غیررسمی هستند با ارائه این روش، مهندس نیازمندیهای جنبه‌گرا می‌تواند یک روش رسمی برای شناسائی جنبه‌ها در پیش بگیرد و از مزایای حاصل از بکارگیری روش رسمی استفاده کند. در روش پیشنهادی تعاریف پایه‌ای مطرح شده‌اند که این تعاریف مبنایی را برای تحت پوشش قرار دادن کل فرآیند جنبه‌گرائی با استفاده از شبکه پتری فراهم می‌کنند.

روش پیشنهادی به خاطر ماهیت تعاریف پایه‌ای خود یک روش تدریجی و تکاملی فراهم می‌کند که در آن مهندس نیازمندیها می‌تواند با شناخت جزئی از سیستم مورد نظر، شروع به شناسائی جنبه‌ها کند و به مرور که شناخت آن از سیستم افزایش یافت و دغدغه‌ها و نیازمندیهای دیگری را مشخص کرد آنها را به مدل اضافه کرده و جنبه‌های جدیدی را شناسائی کند.

روش پیشنهادی به دلیل مبتنی بودن بر شبکه پتری توسط ابزارهای شبیه‌سازی مختلفی پشتیبانی می‌شود و بنابراین می‌تواند بخشی از فعالیت شناسائی را به صورت خودکار انجام دهد همچنین دیگر لزومی برای طراحی ابزار خاص برای این روش نیست.

از آنجائی که روشهای شناسائی غیررسمی جنبه‌ها از یک روش adhoc برای شناسائی جنبه‌ها استفاده می‌کنند اطمینانی وجود ندارد که دغدغه‌ها شناسائی شده به عنوان یک جنبه، یک دغدغه مداخله‌ای باشند ولی با توجه به اینکه روش پیشنهادی رسمی است جنبه‌های شناسائی شده در سیستم موردنظر مطمئنا یک دغدغه مداخله‌ای هستند. از طرفی هم با توجه به اینکه جنبه‌ها در این روش با استفاده از مدلسازی شناسائی می‌شوند فعالیت مربوط به مدلسازی از سیستم نیز می‌تواند تسهیل شود زیرا این عمل در فاز مهندسی نیازمندیهای جنبه‌گرا تا حدی صورت پذیرفته است.

۷-۲-۱- معایب

روش پیشنهادی، مانند هر روش رسمی دیگر دارای معایبی به شرح زیر است:

- استفاده از روشهای رسمی باعث اتلاف وقت شده و گران هستند.

- چون اندکی از توسعه دهندگان نرم‌افزاری زمینه لازم برای بکارگیری روشهای رسمی را دارند آموزش زیادی لازم است.
- تولید شبکه پتری لازم برای مدلسازی و سناشائی به سادگی انجام نمی‌شود.

همچنین، روش پیشنهادی از ترکیب جنبه‌های شناسائی شده و حل تداخل‌های حاصل از آنها پشتیبانی نمی‌کند. در این روش هیچ گونه دسته‌بندی برای وابستگی‌ها، محدودیت‌ها و ارتباطات وجود ندارد و این امر ممکن است باعث شود که بکارگیری آن برای اشخاصی که زیاد با مهندسی نیازمندیها آشنا نیستند مشکل باشد. محدودیت دیگر آن است که روش فقط جنبه‌ها را شناسائی می‌کند و نمی‌تواند به صورت کامل نقاط اتصال را برای جنبه‌ها مشخص کند.

۷-۳- مقایسه روش پیشنهادی با روشهای موجود

مقایسه یک روش با روشهای دیگر مستلزم آن است که هر کدام از روشها به صورت جداگانه بر روی یک مطالعه موردی اعمال شوند بعد نتایج هر یک از این روشها بر اساس شاخص‌هایی که مشخص می‌شوند با یکدیگر مقایسه شوند و برتری داشتن هر روش نسبت به دیگری یا بهتر بودن شاخصی نسبت به شاخص دیگر در روشهای متفاوت مشخص شود.

در مورد مقایسه روشها در حوزه مهندسی نیازمندیهای جنبه‌گرا دو مشکل اساسی وجود دارد. اولا اینکه حوزه مهندسی نیازمندیهای جنبه‌گرا یک حوزه نوپایی است و شاخص‌های اساسی برای آن مشخص نشده است. دلیل آن این است که روز به روز خصوصیات تاثیر گذار بر شاخص‌ها افزایش پیدا می‌کند. ثانیا هنوز مهندسی نیازمندیهای جنبه‌گرا بر روی یک مطالعه موردی واقعی که در صنعت پیاده‌سازی شده باشد اعمال نشده است و ادعاهای مربوط به حمایت هر روش فقط در مطالعه موردی خود آن روش مطرح شده است.

مطابق با این روند رو به رشد در زمینه مهندسی نیازمندیهای جنبه‌گرا ما نیز اقدام به ارائه روشی کردیم که فاکتور اصلی در آن شناسائی جنبه‌ها با استفاده از یک زبان رسمی است که تا کنون در هیچ روش دیگری به آن پرداخته نشده است. لذا مقایسه صحیحی نمی‌شود بین روش پیشنهادی و روشهای دیگر انجام داد. از طرفی هم روش پیشنهادی دارای بعضی کمبودها است که روشهای دیگر آن را حمایت می‌کنند. با این حال با توجه به اینکه بخشی از کمبودهای این روش جزء اهداف کارهای تحقیقاتی آتی در نظر گرفته شده است و تا حدودی کارهای عمده آنها انجام شده است امید می‌رود که آن کمبودها نیز در کارهای آتی برطرف شود. اما با این حال در اینجا سعی می‌شود روش پیشنهادی را با روشهای برجسته‌ای که در فصل چهارم مطرح شد مقایسه کرد در

این مقایسه باز هم از شاخص‌های مطرح شده در جدول ۴-۱۴ استفاده خواهیم کرد. این مقایسه در جدول ۷-۱ نشان داده شده است.

جدول ۷-۱: مقایسه روشهای برجسته مهندسی نیازمندیهای جنبه‌گرا با روش پیشنهادی

*Proposed Method	AORE Model	AORE with ARCaDe	COSMOS	Theme/Doc	MDSoC	
×	×	×	×	×	×	شناسائی دغدغه‌ها
×	×	×	×	×	×	جداسازی دغدغه‌ها
×	×	×	×	×	×	نمایش دغدغه‌ها
-	×	×	×	×	×	ترکیب دغدغه‌ها
-	×	×	–	–	×	تفکیک تداخل‌ها
×	–	×	–	×	×	پشتیبانی ابزار
-	×	×	–	–	×	تاثیر و نگاشت جنبه‌ها
-	–	×	–	×	×	وابستگی به یک زبان تعریف
×	–	–	×	–	×	دغدغه نقش رده اول
-	–	–	×	–	–	ارتباط دو جهته
-	–	×	×	–	×	فراهم کردن مستندات کافی
-	×	×	–	–	×	بیان قوانین ترکیب
×	–	×	×	–	×	قابلیت ردیابی
×	–	–	–	–	–	قابلیت رسمی‌سازی

(علامت × به معنای دارد و علامت – به معنای ندارد)

روش پیشنهادی هیچ مکانیزمی برای شناسائی و جداسازی دغدغه‌ها ارائه نمی‌کند و بیان این دو معیار به منظور این است که در روش پیشنهادی باید این فعالیت‌ها صورت بپذیرد (توسط سایر مکانیزمها) و بعد گامهای دیگر تحقق پیدا کند. روشهای دیگر نیز مکانیزم خاصی را پیشنهاد نمی‌دهند بلکه فقط ذکر می‌کنند که این فعالیت‌ها باید در روش تحقق پیدا کند.

همچنین همان طور که قبلا هم ذکر شد به دلیل نو بودن این حوزه و عدم بکارگیری آن در صنعت نمی‌توان مقایسه کیفی روی معیارها انجام داد و تمام این نتایج حاصل از مطالب ذکر شده در مراجع خود آن روشها هستند.

۷-۵- میزان تحقق اهداف اولیه

همان طور که در آغاز این تحقیق در بخش ۱-٤ مشخص شد هدف دستیابی به یک سری اهدافی بود که مهمترین آنها رسمی‌سازی شناسائی جنبه‌ها بود. با توجه به روش پیشنهادی ارائه شده در فصل پنج و مطالعه

موردی انجام شده در فصل شش مشهود است که توانستیم به هدف مشخص شده در فصل یک دست پیدا کنیم. البته کمی در مبحث مشخص کردن ارتباطات رسمی‌سازی را تضعیف کردیم تا بتوان تحلیل درستی را از سیستم به دست آورد که در صورت محدود سازی آنها، می‌توان ارتباطات را نیز رسمی کرد. همچنین این تحقیق علاوه بر دست‌یافته‌های اصلی یک سری دست‌یافته‌های فرعی را به ارمغان می‌آورد که عبارتند از:

- رسمی سازی گام یا فعالیت اولیه از فاز مهندسی نیازمندیهای جنبه‌گرا
- به نوعی کاهش فعالیت‌های مربوط به شناسائی جنبه‌ها از طریق عدم نیاز به تجزیه کلیه نیازمندیها
- دستیابی به دید جامع از مهمترین روشهای پیشنهادی برای مهندسی نیازمندیهای جنبه‌گرا
- دستیابی به یک مقایسه در مورد روشهای پیشنهادی برای شناسائی جنبه‌ها
- تدریجی و تکاملی بودن روش به دلیل ارائه تعاریف مناسب برای عناصر پایه‌ای روش پیشنهادی

۷-۴- فرصت‌های آتی

مهندسی نیازمندیهای جنبه‌گرا به عنوان فاز اولیه توسعه نرم‌افزاری جنبه‌گرا یکی از پرچالش‌ترین و پراهمیت‌ترین حوزه تحقیقاتی در زمینه جنبه‌گرائی است که دلیل آن متکی بودن سایر فعالیت‌های فازهای توسعه نرم‌افزاری جنبه‌گرا به خروجی محصولات این فاز است. دراین فاز اهداف مختلفی وجود دارد و روز به روز نیز در حال به روز شدن هستند بنابراین فرصت‌های مطالعاتی فراوانی می‌تواند در این بخش مطرح باشد. در اینجا فرصت‌های تحقیقاتی آتی را که هم در ارتباط با این تحقیق انجام شده است یا می‌تواند از این تحقیق نشات بگیرد را معرفی می‌کنیم که عبارتند از:

- مشخص کردن قوانین ترکیب به صورت رسمی مبتنی بر شبکه پتری
- به کارگیری قابلیت تبدیل شبکه پتری به UML به منظور استفاده از روش پیشنهادی برای شناسائی جنبه‌ها در نمادگذاری UML
- استفاده از تعاریف پایه شبکه نیازمندی و شبکه دغدغه برای استخراج جنبه‌ها[1] از سیستم‌های سنتی
- استفاده از قابلیت مانیتورینگ در مدلسازی سیستم و انتقال نشانه مشخص، جهت کشف دو مشکل حاصل از دغدغه‌های مداخله‌ای به نام پراکندگی و درهم تنیدگی
- استفاده از شبکه پتری برای حل تداخل‌های حاصل از اعمال جنبه‌ها

[1] Aspect mining

مراجع

[۱] وحدت عبدل‌زاد، توسعه نرم‌افزاری جنبه‌گرا، سمینار کارشناسی ارشد، دانشگاه آزاد اسلامی واحد علوم و تحقیقات تهران، ۱۳۸۷.

[2] E. Baniassad, P. Clements, J. Araujo, A. Moreira, A. Rashid, B. Teki-nerdogan, “Discovering Early Aspects” , 2006, IEEE Software, 23(1), pp. 61-69.

[3] Awais Rashid, Peter Sawyer, Ana Moreira, Joao Araujo, “Early Aspects: A Model for Aspect-Oriented Requirements Engineering”, IEEE Joint Int'l Conf. on Requirements Engineering, 2002, IEEE CS, pp. 199-202.

[4] Awais Rashid, Ruzanna Chitchyan, "Aspect-Oriented Requirements Engineering: A Roadmap", ACM 978-1-60558-032, 2008.

[5] Stanley M. Sutton and Isabelle Rouvellou, “Modeling of Software Concerns in Cosmos”, International Conference on Aspect-Oriented Software Development, 2002, ACM, pp. 127-133.

[6] Elisa Baniassad, Siobhan Clarke, "Theme: An Approach for Aspect-Oriented Analysis and Design", Int'l Conf. Software Engg.(ICSE), 2004, IEEE CS, pp. 158-167.

[7] Ana Moreira, Awais Rashid, Joao Araujo, "Multi-Dimensional Separation of Concerns in Requirements Engineering", Int'l Conf. on Requirements Engg. (RE), 2005, IEEE CS, pp. 285-296.

[8] Ivar Jacobson, Pan-Wei Ng, Aspect-Oriented Software Development with Use cases, Addison-Wesley, ISBN: 0-321-26888-1, 2004.

[9] J. Araujo, J. Whittle, D.-K. Kim, “Modeling and Composing Scenario-Based Requirements with Aspects”, Int'l Conf. Requirements Engg. (RE), 2004, IEEE CS, pp. 58-67.

[10] R. Chitchyan, A. Rashid, P. Rayson, R. W. Waters, “Semantics-based Composition for Aspect-Oriented Requirements Engineering”, Int'l Conf. Aspect-Oriented Software Development (AOSD), 2007, ACM, pp 36-48.

[11] Siobhán Clarke, Elisa Baniassad, Aspect-Oriented Analysis and Design: The Theme Approach, Addison Wesley Professional, ISBN: 0-321-24674-8, 2005.

[12] C. Haley, R. Laney, B. Nuseibeh, “Deriving Security Requirements from Crosscutting Threat Descriptions”, 2004, Int'l Conf. Aspect-Oriented Software Development (AOSD), ACM.

[13] S. Katz, A. Rashid, “From Aspectual Requirements to Proof Obliga-tions for Aspect-Oriented Systems”, Int'l Conf. Requirements Engg., 2004, IEEE CS, pp. 48-57.

[14] L. K. Kit, K. C. Man, E. Baniassad, “Isolating and Relating Concerns in Requirements using Latent Semantic Analysis”, 2006, ACM Conf. on Object-Oriented Programming, Languages, Systems and Applica-tions (OOPSLA), ACM.

[15] A. Moreira, J. Araujo, “Handling Unanticipated Requirements Change with Aspects”, Int'l Conf. Software Engg. and Knowledge Engg. (SEKE), 2004, pp. 411-415.

[16] A. Moreira, J. Araujo, J. Whittle, “Modeling Volatile Concerns as Aspects”, Conference on Advanced Information Systems Engineer-ing (CAiSE), 2006.

[17] J. Whittle and J. Araujo, “Scenario Modelling with Aspects”, 2004, IEE Proceedings - Software Special Issue, 151(4), pp. 157-172.
[18] Y. Yu, J. Leite, and J. Mylopoulos, “From Goals to Aspects: Discov-ering Aspects from Requirements Goal Models”, 2004, Int'l Conf. on Re-quirements Engg. (RE), IEEE CS, pp. 38-47.
[19] Young, Ralph R. Effective Requirements Practices. Boston: Addison-Wesley, 2001.
[20] Boehm, B.W. and Papaccio, P.N., 1988, Understanding and controlling software costs, IEEE Trans of Software Engineering, 14(10), pp. 1462-1477.
[21] Bridges, W., 1995, Managing Transitions, Making the most of change, Nicholas Brealey Publishing, UK.
[22] Brinkkemper, S., 1996, Method engineering: engineering of information systems development methods and tools, Inf. Software Technol., 38(4), pp. 275-280.
[23] Davis, A.M., 1993, Software Requirements: Analysis and Specification, Prentice Hall, second Edition.
[24] http://www.research.ibm.com/hyperspace/ConcernSpaces.htm, 2004.
[25] Klein, M. and Kazman, R. 1999. Attribute-based architectural styles. Tech. Rep. CMU/SEI-99-TR-022, Software Engineering Institute, Carnegie Mellon University. Oct.
[26] P. Tarr, H. Ossher, W. H. Harrison, and S. S. Jr. N degrees of separation: Multi-dimensional separation of concerns. In Proceedings of the International Conference on Software Engineering, pages 107–119. IEEE Computer Society Press, 1999.
[27] Merriam-Webster. Merriam-Webster collegiate dictionary online. http://www.merriam-webster.com.
[28] IEEE. 2000. IEEE recommended practice for architectural description of software-intensive systems. IEEE Std.14712000.
[29] Nuseibeh, B., Kramer, J., and Finkelstein, A. 1993. Expressing the relationships between multiple views in requirements specification. In 15th Int'l Conf. Software Engineering (ICSE), (Baltimore, Maryland). IEEE, 187196.
[30] Sutton Jr.,S. M. 1999. Multiple dimensions of concern in software testing. In Workshop on Multi-Dimensional Separation of Concerns (OOPSLA), (Denver). http://www.cs.ubc.ca/~murphy/multid-workshop-oopsla99/positionpapers/ws13-sutton.pdf.
[31] http://en.wikipedia.org/wiki/Cross-cutting_concern, 2009.
[32] Dijkstra, Edsger W. (1982), “On the role of scientific thought”, in Dijkstra, Edsger W., Selected writings on Computing: A Personal Perspective, New York, NY, USA: Springer-Verlag New York, Inc., pp. 60–66, ISBN 0-387-90652-5.
[33] Reade, Chris (1989). Elements of Functional Programming. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 600 pages. ISBN 0201129159.
[34] Roger S.Pressman, Software Engineering: a practitioner’s approach, Fifth edition, McGraw-Hill, page 541, 2001.
[35] Grady Booch, Ivar Jacobson, and James Rumbaugh, Object Oriented Analysis and Design with Applications 3rd Edition, Addison.Wesley, 2007.
[36] Anthony Finkelstein and Ian Sommerville, “The Viewpoints FAQ”, 1996, BCS/IEE Software Engineering Journal, 11(1).
[37] I. Sommerville, P. Sawyer, Requirements Engineering- A Good Practice Guide: John Wiley and Sons, 1997.

[38] Mehdi Khosrow-Pour, Encyclopedia of Information Science and Technology; Prolonging the Aging of Software System, Second edition, 2009, pp. 3152.
[39] http://en.wikipedia.org/wiki/Maintainability/, 2009.
[40] Renaud Pawlak, Lionel Seinturier, and Jean-Philippe Retaillé, Foundations of AOP for J2EE Development, Springer-Verlag New York, ISBN: 1-59059-507-6, 2005.
[41] G. Kiczales, et al., "Aspect-Oriented Programming", European Conf. on Object-Oriented Programming (ECOOP), 1997, Springer, LNCS 1241, pp. 220-242.
[42] Doug Rosenberg, Matt Stephens, Use Case Driven Object Modeling with UML, Springer-Verlag, ISBN: 1-59059-774-5, 2007.
[43] http://trese.cs.utwente.nl/taosad/aosd.htm, 2009.
[44] R.T. Alexander, J.M. Bieman, and A.A. Andrews,"Towards the Systematic Testing of Aspect- Oriented Programs", 2004, Technical Report CS-4-105, ColoradoState University.
[45] Reza Meimandi Parizi, Abdul Azim Ghani, "A Survey on Aspect-Oriented Testing sApproaches", 2007, IEEE DOI 10.1109/ICCSA.
[46] Fukuzawa, K. AND Saeki, M. 2002. Evaluating Software Architectures by Coloured Petri Nets. ACM SEKE '2002, July 15-19.
[47] CPN Tools. 2008. webpage, URL http://www.daimi.au.dk/CPNTools/, 2009.
[48] Alan M. Davis, Software Requirements Revision Objects: Functions & States, Prentice Hall PTR, 1994.
[49] Gerald Kotonya and Ian Sommerville, Requirements Engineering, John Wiley & Sons, 1997.
[50] Jakob Nielsen, The Use and Misuse of Focus Groups, http://www.useit.com/papers/focusgroups.html, 1997.
[51] Linda A. Macaulay, Requirements Engineering, Springer Verlag, 1996.
[52] Peter Checkland and Jim Scholes, Soft Systems Methodology in Action, John Wiley & Sons., 1999.
[53] Christopher McPhee, "Requirements engineering for projects with critical time-to-market", 2001, Master's thesis, University of Calgary.
[54] O. L´opez and M.A. Laguna, "Requirements reuse for software development", August 2001, In Fifth IEEE International Symposium on Requirements Engineering, Toronto, Canada.
[55] Karl E. Wiegers. Software Requirements. Microsoft Press, 1999.
[56] Jennifer Tapke, Allyson Muller, Greg Johnson, Josh Sieck, House of Quality. Steps in Understanding the House of Quality, IE 361.
[57] Colin Potts. Requirements Reviews: Understanding Customer Requirements. http://www.cc.gatech.edu/ computing/classes/cs3302 98 summer/7-21-reqts/sld001.htm, 1998.
[58] Suzanne Robertson, "Requirements testing: Creating an effective feedback loop", July 2000, In FEAST 2000, London.
[59] H. Klaeren, E. Pulvermueller, A. Rashid, and A. Speck, "Aspect Composition Applying the Design by Contract Principle", 2nd International Symposium on Generative and Component-based Software Engineering (GCSE), 2000, Springer-Verlag, LNCS 2177, pp. 57-69.

[60] Awais Rashid, Ana Moreira, Joao Araujo, “Modularisation and Composition of Aspectual Requirements”, 2nd Int'l Conf.Aspect-Oriented Soft-ware Development, 2003, ACM, pp.11-20.
[61] Ivar Jacobson, Object-Oriented Software Engineering- a Use Case Driven Approach: Addison-Wesley, 1992.
[62] A. Lamsweerde, “Goal-Oriented Requirements Engineering: A Guided Tour”, 5th Int'l Symp. on RE, 2001, IEEE CS Press, pp. 249-261.
[63] M. Jackson, Problem Frames: Analyzing and Structuring Software Development Problems: Addison Wesley, 2000.
[64] Sutton Jr., S. M. and Rouvellou, I. Advanced Separation of Concerns for Component Evolution. Workshop on Engineering Complex Object Oriented Systems for Evolution. Conf. on Object-Oriented Programming. Systems, Languages, and Appfications, Tampa, Florida, Oct. 2001.
[65] lyengar, A. Design and Performance of a General Purpose Software Cache. In]hoe. of the IBth IEEE Int. Performance, Computing, and Communications Conf. (IPCCC'99), Phoenix/Scottsdale, Arizona, Feb. 1999.
[66] Memmert, J. Personal commtmication. September, 2001.
[67] IBM. Hyper/J, http://www.research.ibm.com/hyperspace/HyperJ/, 2006.
[68] S. Clarke. “Extending standard uml with model composition semantics”, July 2002, Science of Computer Programming, 44(1): pp. 71–100.
[69] S. Clarke and R. J. Walker. Composition patterns: An approach to designing reusable aspects. In International Conference on Software Engineering, pages 5–14, 2001.
[70] L. Chung, et al., Non-Functional Requirements in Software Engineering: Kluwer, 2000.
[71] T. Murata, Petri nets: Properties, analysis and applications, 1989, Proceedings of the IEEE, 77(4):541–580.
[72] Binder, R. V, “Testing Object-Oriented Systems: Models, Patterns and Tools”, Addison-Wesley, 2000.
[73] I. Sommerville, Software Engineering, Seventh edition, Addison-Wesley, 2005.

An Approach for Aspect-Oriented Requirements Engineering

Abstract

Aspect-Oriented Programming (AOP) improves modularity of concerns, particular crosscutting concerns in implementation level. However modularity of concerns in implementation level does not emerge important performance, because this operation (modularity) must be done in early phase of software system development. This phase of system development is known Aspect-Oriented Requirements Engineering (AORE). The goal of AORE is separation of concerns as well as possible and identification of aspects from documentation of system. Therefore, we must identify aspects in the first stage of developing. If we do not identify aspects in the early phases of software development (requirements engineering), we can only use the conception of aspect-orientation in the code level.

Until now, various and informal methods are proposed for identification of aspects in AORE. Each method presented their means, structures and tools for supporting aspects identification. Moreover these methods have solved the problem of aspects identification and also have achieved the goals such as aspects composition, conflict resolution and trade-off points for aspect-oriented architecture. In context of aspects formalization, some methods are proposed in order to utilize the concept of aspects in Petri Nets for security problems and aspects conflicts in the same join point. However, not any method presents a formal method for identification of aspects.

In order to formalization, a method based on Petri Nets for identification of aspects is proposed in this research. The proposed method starts with two definitions is called requirement nets and concern nets, finally aspects are identified by executing the final model of system. This method only identifies aspects but its definitions and structures may support other characteristics of AORE that this requires more researches.

Keywords

Identification of aspects, Aspect-Oriented Requirements Engineering (AORE), Petri Nets, concern nets, requirement nets, crosscutting concerns, Aspect-Oriented Programming (AOP), separation of concerns.

Vahdat Abdelzad

v.abdelzad@gmail.com

Islamic Azad University
Science and Research Branch

Master of Science Thesis

Computer Engineering – Software

Subject:

An Approach for Aspect-Oriented Requirements Engineering

Supervisor:

Dr. F. Shams

Advisor:

Dr. A. Seyyedi

Author:

Vahdat Abdelzad

Date:

December 2009