دانشگاه یزد

مقدمه‌ای بر مهندسی نرم‌افزار

با هدف افزایش کیفیت در چرخه تولید نرم‌افزار

محمدعلی زارع چاهوکی

پیشگفتار

رشد و توسعه نرم‌افزارها در ابعاد گوناگون زندگی انسان‌ها سبب شده است تا داشتن دیدی جامع در فرآیند توسعه آن بسیار ضروری شود. با دقت در رشته‌های دیگر مهندسی، کاملا مشهود است که در آنها درسی مثلا با نام مهندسی مکانیک وجود ندارد. ولی چرا باید در رشته مهندسی نرم‌افزار درس‌ها و کتاب‌هایی با عنوان مهندسی نرم‌افزار وجود داشته باشد. پاسخ به این پرسش را باید در ذات و طبیعت این رشته از مهندسی جستجو کرد. شاید هم بتوان اطلاق کلمه مهندسی به این حوزه از دانش بشر را تلاشی برای تبدیل فرآیند توسعه نرم‌افزار به ساختاری مهندسی دانست.

تولید نرم‌افزار در حوزه تجربی بشری دانشی جدید است و ابزارهای ایجاد آن و ماهیت محصول خروجی آن مانند دیگر دست‌ساخته‌های بشری ملموس نبوده و بسیار گسترده می‌باشد. لذا ارائه قواعد مهندسی نیز برایش دشوار می‌باشد. در درس یا کتاب مهندسی نرم‌افزار مسلما نمی‌توان تمامی مهارت‌های ساخت نرم‌افزار، سنجش و نگهداشت آن را بیان کرد. چرا که ورود به هر بخش سبب پیدایش حجم انبوهی از مستندات ساخت‌یافته مانند کتاب و غیر ساخت‌یافته مانند انجمن‌های[1] تخصصی می‌شود که شاید بتوان روزگاری با نظم‌دهی آنان مرجعی برای مهندسی نرم‌افزار ایجاد کرد. ولی می‌توان این درس را برای دانشجویان، شروعی برای تفکر سیستماتیک در مراحل مختلف تولید نرم‌افزار در تقابل با تفکر برنامه‌نویسی درنظر گرفت.

در تولید نرم‌افزار هر چندسال یکبار شیوه‌های تولید نرم‌افزار متنوع می‌شود و امروزه تنوع معماری‌های نرم‌افزاری به گونه‌ای شده است که هر برنامه‌نویسی

---

[1] Forum

نمی‌تواند در بیش از یک حوزه از آن مهارت کافی را کسب کند. شاید متدولوژی‌های توسعه نرم‌افزار را بتوان بهترین مرجع برای اندیشیدن به سبک مهندسی در معماری‌های مختلف نرم‌افزاری دانست. ما در این کتاب برآن نیستیم تا معماری یا متدولوژی مشخصی در تولید نرم‌افزار را به عنوان مرجع مشخص کنیم. ولی از آنجایی که معماری شیء گرا بیش از دو دهه است که همچنان در تولید نرم‌افزار موثر بوده است، به عنوان بستری برای بیان مهمترین فعالیت‌ها در مهندسی نرم‌افزار انتخاب شده است.

همچنین از آنجایی که امروزه نگرش فرآیندی در سازمان‌ها بسیار گسترش یافته است، لذا پرداختن به مهندسی نرم‌افزار از این منظر نیز به نظر مولف بسیار اهمیت دارد. لذا مقدمه‌ای بر مفاهیم آن نیز در انتهای کتاب آمده است.

فهرست مطالب

بخش اول: مفاهیم و تاریخچه مهندسی نرم‌افزار

۱ مقدمه

مهندسی نرم‌افزار را می‌توان روش سیستمی در شناخت، آنالیز، طراحی، پیاده‌سازی، آزمون و نگهداری نرم‌افزار تعریف کرد. واژه مهندسی نرم‌افزار اولین بار در ۱۹۶۸ برای عنوان کنفرانسی که توسط ناتو[۲] پشتیبانی می‌شد، استفاده گردید. نتیجه آن کنفرانس درباره چگونگی توسعه نرم‌افزارها در گزارشی منتشر گردید.

هدف از مهندسی نرم‌افزار آن است تا بتوان تولید نرم‌افزار را در (۱) محدوده زمانی و هزینه قابل پیش‌بینی، (۲) با کیفیت مورد نیاز کاربران آن ایجاد کرد. چنانچه پروژه‌ای نرم‌افزای نتواند این دو الزام را برآورده سازد و یا به عبارتی فاصله غیرقابل قبولی با این دو معیار داشته باشد، عملا آن پروژه شکست خورده می‌باشد.

تمامی پروژه‌هایی که با شکست مواجه شده‌اند، دارای نشانه‌های مشترکی هستند. هرچند علت پدید آمدن نشانه‌های شکست را می‌توان با رویکردهای مبتنی بر مهندسی نرم‌افزار شناسایی کرد و با روش‌های مهندسی برطرف ساخت. مسلما بین نشانه شکست و دلیل آن تفاوت وجود دارد همچنانکه مابین نشانه تب در بیمار با علت تب تفاوت وجود دارد.

در ادامه ابتدا نشانه‌های عمومی از شکست در پروژه‌های نرم‌افزاری را بیان می‌کنیم و سپس مروری بر دلایل پدید آمدن آنها خواهیم داشت. مسلما هدف از مهندسی نرم‌افزار ارائه رویکردی سیستمی است تا از دلایل بروز شکست در پروژه‌های نرم‌افزاری، راهکاری برای پیشگیری از آنها ارائه دهد.

---

[2] NATO

۱,۱ نشانه‌هایی از شکست در پروژه‌های نرم‌افزاری

در این بخش نشانه‌هایی که بیانگر شکست در پروژه است آورده می‌شود. شاید نتوان برخی از این نشانه‌ها را در تمامی پروژه‌های شکست‌خورده مشاهده کرد ولی معمولا در پروژه‌هایی که با شکست مواجه شده‌اند این نشانه‌ها وجود دارد.

۱- عدم فهم درست از نیازمندی‌های کاربر نهایی[۳]

در پروژه‌های نرم‌افزاری هرچه از زمان پروژه می‌گذرد نشانه‌هایی پدیدار می‌شود که بیانگر آن است که فهم ما به عنوان تیم پروژه با نیازمندی‌های واقعی کاربران نهایی آن تفاوت دارد. این نشانه‌ها می‌تواند در زمان‌های مختلف از شروع پروژه پدیدار شود. هرچه این نشانه در اواخر برنامه زمانبندی پروژه پیدا شود، پروژه با چالش جدی‌تری مواجه خواهد شد. باید همواره به این نکته توجه داشت که هر نرم‌افزاری فارغ از اینکه توسط چه کسی سفارش داده می‌شود، باید منطبق بر نیازهای کاربر نهایی استفاده کننده از آن باشد.

۲- عدم توانایی تیم پروژه در پاسخگویی به تغییرات

مسلما تغییرات بخشی جداناپذیر هر موجودی است که می‌خواهد به صورت پویا با محیط در ارتباط باشد. از آنجایی که نرم‌افزار نیز در محیط‌های انسانی و در حال پیشرفت فعالیت می‌کند، لذا درخواست برای تغییرات آن نیز همواره وجود خواهد داشت. با وجود این ویژگی ذاتی در تعامل نرم‌افزار با کاربرانش، ولی نباید حجم درخواست‌ها سبب سردرگمی تیم پروژه شده و ایشان نتوانند طی زمان‌بندی قابل قبولی پاسخگوی تغییرات باشند.

۳- ماژول‌هایی که متناسب[۴] هم نیستند

[3] End-user

مسلما تمامی نرم‌افزارها از ماژول‌های متعددی ایجاد شده‌اند. این ماژول‌ها بسته به اینکه نرم‌افزار از چه نوع معماری تبعیت می‌کند متفاوت خواهند بود. هر ماژول بسته به بزرگی آن توسط یک برنامه‌نویس و یا یک گروه برنامه‌نویسی ایجاد می‌شود. مسلما ایجاد کننده‌های ماژول‌ها، آنها را آزمون می‌کنند و از صحت عملکرد آنها اطمینان حاصل می‌کنند. ولی در یکپارچه سازی ماژول‌ها، در قالب سیستمی نرم‌افزاری و یا بخشی از سیستم، عملکرد مورد انتظار از آنها مشاهده نمی‌شود.

۴- دشواری در نگهداری و یا توسعه نرم‌افزار

همانگونه که قبلا نیز بیان شد، نرم‌افزار در محیطی پویا فعالیت می‌کند و همواره نیاز به نگهداری[۵] یا توسعه[۶] آن می‌باشد. منظور از نگهداری نرم‌افزار، رفع اشکالات[۷] آن و منظور از توسعه، افزودن قابلیت‌هایی جدید به آن می‌باشد. چنانچه انجام این دو فعالیت، که جزئی جدایی ناپذیر در هر پروژه نرم‌افزاری هستند، با صرف هزینه و زمان زیاد امکان پذیرد باشد و به هر علتی سبب افت کارایی سیستم نرم‌افزاری گردد، نشانه‌ای از شکست پروژه نرم‌افزاری در آینده خواهد بود.

۵- شناسایی دیرهنگام معایب جدی نرم‌افزار

هرچه معایب عمده نرم‌افزار در ابتدای برنامه زمان‌بندی پروژه اتفاق بیفتد، امکان رفع آنها و ساختاردهی مجدد به معماری نرم‌افزار با هزینه کمتری انجام

[4] Fit
[5] Maintenance
[6] Extend
[7] Bug

خواهد پذیرفت. در این خصوص، رفع اشکالات اصلی نرم‌افزار در اواخر پروژه عملا با هزینه و زمان پیش‌بینی شده در آن امکان پذیر نمی‌باشد و سبب شکست در پروژه می‌گردد.

۶- کیفیت[8] ضعیف نرم‌افزار

مسلما کیفیت نهایی در نرم‌افزار زمانی حاصل می‌شود که خطایی[9] از جانب بهره‌برداران آن گزارش نشود. مسلما مقایسه کیفیت نرم‌افزارها با هم در قالب هزینه و زمان انجام آنها باید صورت پذیرد. ولی نکته‌ای که ناگذیر از پذیرش آن هستیم، توافق بر تعریف کیفیت بین تیم توسعه دهنده نرم‌افزار و کارفرما بر اساس هزینه و زمان پروژه می‌باشد. با این مقدمه کوتاه درباره کیفیت نرم‌افزار، چنانچه میزان گزارش خطاها در نرم‌افزار با گذشت زمان افزایش یابد و یا پس از اعلام اصلاح آنها، مجدد مشاهده شوند، کیفیت نرم‌افزار به شدت افت می‌کند.

۷- کارآیی[10] غیرقابل‌قبول نرم‌افزار

کارآیی در نرم‌افزار به ویژگی‌هایی اطلاق می‌شود که از نوع کارکردی نمی‌باشند. توضیحات تفصیلی در این حوزه در فصل ۷ آمده است. مثلا وب‌سایتی را درنظر بگیرید که امکان ورود کاربران متعددی با نام کاربری و کلمه عبور را فراهم می‌سازد. چنانچه با ورود کاربران متعدد به سایت، زمان باز شدن صفحات آن به گونه‌ای افزایش یابد که برای کاربران قابل تحمل نباشد، کارآیی نرم‌افزار در این مثال غیرقابل قبول می‌باشد.

۸- عدم امکان ردیابی[11] تغییرات در نرم‌افزار

[8] Quality
[9] Bug or Defect
[10] Performance

تیم پروژه از افراد متعددی تشکیل شده است که البته برنامه‌نویسان اکثریت را تشکیل می‌دهند. بین مدیریت بر پروژه‌ای که تمامی کدهای آن توسط یک برنامه‌نویس تهیه می‌شود با پروژه‌ای که از چند برنامه‌نویس استفاده می‌کند، تفاوت بسیاری وجود دارد. چنانچه در پروژه‌ای نتوان مشخص کرد که کدها و یا دیگر فرآورده‌های[۱۲] پروژه توسط چه کسی[۱۳]، کی[۱۴]، چرا[۱۵] و چگونه[۱۶] ایجاد شده و یا تغییر کرده است، مسلما مدیریت پروژه نمی‌تواند برنامه‌ریزی و پاسخگویی در کیفیت و کارآیی نرم‌افزار داشته باشد.

۹- فرآیند غیرقابل اعتماد در ساخت و انتشار[۱۷] نسخه‌های نرم‌افزار

هر نسخه نرم‌افزار براساس زمان‌بندی پروژه می‌بایست پاسخگوی نیازمندی‌های خاصی از کاربران باشد. با توجه به اینکه هر تغییر اندکی در کد برنامه سبب می‌شود تا کیفیت نرم‌افزار به شدت تحت تاثر قرار گیرد، لذا روند ساخت و انتشار بسیار اهمیت دارد. اگر کاربران احساس کنند که مدیریتی در این فرآیند وجود ندارد یعنی (۱) بخشی از نیازمندی‌های برنامه‌ریزی شده در نسخه منتشر شده وجود ندارد، (۲) خطاهایی که قبلا وجود نداشته است در نسخه جدید پدید آمده است و (۳) خطاهای گزارش شده قبلی برطرف نشده‌اند، موفقیت پروژه با ریسکی جدی مواجه خواهد شد.

[11] Tracking
[12] Artifact
[13] Who
[14] When
[15] Why
[16] How
[17] Release

در ادامه در بخش بعد، دلایلی که سبب پدیدآمدن نشانه‌های شکست در پروژه‌های نرم‌افزاری می‌شوند، مورد بررسی و تحلیل قرار می‌گیرند. مسلما پرداختن به دلایل می‌تواند تا حدودی مانع شکست پروژه جاری شده و یا برنامه‌ریزی برای موفقیت در پروژه‌های بعدی را سبب شود.

۱,۲ دلایل شکست در پروژه‌های نرم‌افزاری

پروژه‌های مختلف به دلایل گوناگونی با شکست مواجه می‌شوند. بنابراین هرچند نمی‌توان دلیل یا دلایل یکسانی برای شکست در همه این پروژه‌ها ذکر کرد، ولی ترکیبی از مجموعه دلایل زیر در شکست پروژه‌ها دخیل می‌باشد.

۱- مدیریت نیازمندی‌ها[۱۸] یکی از مهمترین فعالیت‌ها در پروژه‌های نرم‌افزاری می‌باشد. در مدیریت نیازمندی‌ها بایستی طرحی ذهنی یا به صورت مصوب با دیگر اعضای تیم پروژه، کارفرما و کاربران وجود داشته باشد تا چگونگی جمع‌آوری نیازمندی‌های نرم‌افزار، اعلام تغییرات و اشکالات و چگونگی مرتفع ساختن آنها سازمان‌دهی گردد. مسلما چنانچه هرکدام از این اجزا در پروژه‌ای به درستی مدیریت نشود، عاملی برای پدیدار شدن نشانه‌های شکست در آن پروژه می‌باشد. در فصل ۵ درخصوص مدیریت نیازمندی‌ها به تفصیل صحبت خواهد شد.

۲- ارتباطات مبهم و غیر دقیق میان اعضای پروژه نیز یکی از عوامل شکست در پروژه است. در پروژه‌های نرم‌افزاری کوچک، یک یا دو برنامه نویس معمولا با فردی مشخص به عنوان کارفرما در ارتباط هستند. البته در برخی از پروژه‌ها نیز فرد مشخصی به عنوان کارفرما وجود ندارد و اعضای تیم برنامه‌نویسی با توجه به ایده‌ای که در ذهن دارند درصدد پیاده‌سازی آن می‌باشند. در پروژه‌های بزرگی که

[1818] Requirments management

به غیر از اعضای تیم پروژه، سازمان کارفرما، سازمان نظارت، گروه‌های کاربری متعدد و همچنین سازمان‌های ذی‌نفع مختلفی وجود دارد، تبیین ارتباطات اعضای مختلف، بیان مسئولیت هرکدام و همچنین چگونگی ارتباط آنها با هم بسیار اهمیت دارد. چراکه چنانچه بواسطه ارتباطات غیر مصوب افراد با هم، بخشی از نیازمندی‌ها و اشکالات پروژه به درستی منتقل نگردد، پذیرش نرم‌افزار طبق برنامه‌ریزی انجام نخواهد پذیرفت. عدم پرداختن به مدیریت ارتباطات در پروژه هزینه‌های بسیاری دربر خواهد داشت.

۳- معماری نرم‌افزار باید مبتنی بر نیازمندی‌های آن باشد. اگر نیازمندی‌های نرم‌افزار را به دو بخش عمده کارکردی[۱۹] و غیرکارکردی[۲۰] تقسیم کنیم (بخش دوم کتاب)، خواسته‌های کاربران در خصوص چگونگی ثبت اطلاعات و بازیابی آنها در حوزه کارکردی قرار می‌گیرد. نیازمندی‌های غیرکارکردی مرتبط با پایداری و امنیت سیستم هستند. بنابراین با رشد تعداد کاربران و حجم داده‌ها برجسته می‌شوند. لذا در یک جمع‌بندی هرچند نیازمندی‌های کارکردی بر معماری نرم‌افزار تاثیرگذار هستند ولی بواسطه پنهان بودن نیازمندی‌های غیرکارکردی، شناخت صحیح و آزمون آنها بسیار مشکل می‌باشد. از این‌رو چنانچه این دو دسته از ویژگی‌ها در ابتدای فرآیند توسعه نرم‌افزار شناسایی و آزمون نشود، نرم‌افزار ایجاد شده از معماری مناسبی برخوردار نخواهد بود. در فصل‌های آینده راهکارهایی تفصیلی برای جمع‌آوری و آزمون نیازمندی‌های کارکردی و غیرکارکردی ارائه خواهد شد.

۴- ایجاد نرم‌افزار به گونه‌ایکه نگهداری و توسعه‌های آینده آن ساده باشد بسیار اهمیت دارد. لذا پیچیدگی در کد برنامه که سبب دشواری در تغییر آن شود نیز یکی از علل شکست پروژه‌های نرم‌افزاری می‌باشد. بهره‌گیری از الگوهای

[19] Functional
[20] Non-functional

طراحی[۲۱]، استفاده از استانداردهایی برای کدنویسی و همچنین بهره‌گیری از رویکرد برنامه‌نویسی جفتی[۲۲] می‌تواند راهکارهایی برای کاهش پیچیدگی و سادگی کدهای برنامه گردد.

۵- ناسازگاری[۲۳] در نیازمندی‌ها، طراحی و پیاده‌سازی نیز عامل دیگری در شکست پروژه‌های نرم‌افزاری است. ناسازگاری در نیازمندی‌ها به معنای وجود تناقض در نیازمندی‌های مختلف می‌باشد. ممکن است جمع‌آوری نیازمندی‌های نرم‌افراد توسط بیش از یک فرد یا تیم صورت پذیرد. در این صورت چنانچه در دسته‌بندی درخواست‌های کاربران و تبدیل آنها به ویژگی‌های نرم‌افزار روالی برنامه‌ریزی شده وجود نداشته باشد، ناسازگاری در این حوزه بعید نخواهد بود. همچنین چنانچه نتوان همواره بین آیتم‌های طراحی و نیازمندی‌های نرم‌افزاری ارتباطی برقرار کرد، ناسازگاری بین نیازمندی‌ها و طراحی نیز ممکن خواهد بود. چنین تفسیری را مابین طراحی و پیاده‌سازی نیز می‌توان متصور بود. چگونگی برقراری ارتباط بین نیازمندی‌ها، طراحی و پیاده‌سازی یکی از اجزای مدیریت نیازمندی‌ها است. در خصوص مدیریت نیازمندی‌ها در فصل ۵ به تفصیل صحبت خواهد شد.

ذکر این نکته را نیز در اینجا ضروری می‌دانم که شاید همواره اجزای مدیریت نیازمندی‌ها به صورت مکتوب و رسمی در پروژه انجام نشود. بسته به ماهیت پروژه و متدولوژی مصوب برای آن، مدیریت نیازمندی‌ها می‌تواند در چگونگی ارتباطات بین افراد تیم پروژه صورت پذیرد.

[21] Design pattern
[22] Pair pogramming
[23] Inconsistencies

۶- عدم آزمون کافی نیز از دیگر علل شکت در پروژه‌های نرم‌افزاری می‌باشد. همانگونه که از نیازمندی‌ها دو تقسیم‌بندی کارکردی و غیرکارکردی وجود دارد، آزمون آنها نیز در این دو بخش انجام می‌شود. مدیریت آزمون و استراتژی آزمون مناسب در هر پروژه نرم‌افزاری از عوامل کلیدی موفقیت پروژه می‌باشد.

۷- ارزیابی صحیح از وضعیت پروژه در موفقیت آن بسیار تاثیرگذار است. چنانچه ارزیابی از وضعیت پروژه بدون درنظر گرفتن فعالیت‌های انجام شده و ریسک‌های باقیمانده و به تبع آن فعالیت‌های باقیمانده صورت پذیرد و صرفا به صورت ذهنی انجام شود، خود عاملی در جهت شکست پروژه می‌باشد. چنانچه پروژه دارای ذی‌نفعان متعددی باشد، معیار ارزیابی وضعیت و پیشرفت پروژه باید به تصویب تمامی ذی‌نفعان در ابتدای پروژه برسد. در غیراینصورت ممکن است معیاری که پیمانکار برای پیشرفت پروژه در ذهن دارد و برنامه‌ریزی‌های خود را بر آن اساس انجام می‌دهد با معیار کافرما ویا کاربران سیستم نرم‌افزاری متفاوت باشد.

۸- چنانچه در ریسک‌هایی که در پروژه وجود دارد، برنامه‌ریزی برای مدیریت آنها وجود نداشته باشد، به عاملی دیگر در شکست پروژه‌های نرم‌افزاری تبدیل می‌شود. بهره‌گیری از متدولوژی مناسب با ماهیت پروژه، سبب استفاده از توصیه‌های متدولوژی در مواجهه با ریسک‌های پروژه می‌شود. بنابراین عاملی موثر در موفقیت پروژه می‌باشد. در خصوص متدولوژی‌های توسعه نرم‌افزار در فصل ۲ صحبت خواهد شد.

۹- همانگونه که قبلا نیز بیان شد، کنترل تغییرات یکی از اجزای مدیریت نیازمندی‌های پروژه می‌باشد. چنانچه تغییرات از هر دریچه‌ای به پروژه وارد شود و بدون نظارتی هدفمند توسط افراد تیم پروژه اعمال شود، به عاملی در جهت شکست پروژه تبدیل می‌شود. بنابراین از زاویه‌ای دیگر، چنانچه نحوه دریافت

تغییرات در پروژه توسط ذی‌نفعان مصوب شده باشد و چگونگی اعمال آن در نیازمندی‌ها، طراحی و پیاده‌سازی و به تبع آن آزمون بانظارت و قابل ردیابی باشد، به عاملی درجهت موفقیت پروژه تبدیل می‌شود.

۱۰- استفاده ناکافی از اتوماسیون و ابزارهای مناسب در جنبه‌های مختلف مدیریت پروژه، مدیریت نیازمندی‌ها، آزمون، مدیریت پیکربندی و تغییرات[۲۴] نیز یکی دیگر از عوامل شکت در پروژه‌های نرم‌افزاری می‌باشد. استفاده ناکافی از اتوماسیون مخصوصا در حوزه آزمون نرم‌افزار عاملی اساس در جهت شکست پروژه می‌باشد.

[24] Configuration and management

۲ متدولوژی‌های توسعه نرم‌افزار

دلایلی که برای شکست پروژه‌های نرم‌افزاری در فصل قبل مطرح گردید، عمومی هستند. بسته به ماهیت پروژه‌های نرم‌افزاری و با توجه به تجارب بدست آمده از پروژه‌های مشابه می‌توان راهکارهایی برای پیشگیری از شکست و افزایش موفقیت آنها ارائه داد.

در متدولوژی توسعه نرم‌افزار[۲۵] یا فرآیند توسعه نرم‌افزار[۲۶] رویکردی بادیسیپلین[۲۷] ارائه می‌شود که در آن، کارها و مسئولیت‌های افراد موثر در توسعه نرم‌افزار مشخص می‌شود. افراد موثر در توسعه نرم‌افزار می‌توانند هم در عضو تیم توسعه نرم‌افزار باشند و هم افرادی از سمت کارفرما و یا کاربران نرم‌افزار باشند.

هدف هر متدولوژی توسعه نرم‌افزار آن است تا برای دامنه‌ای مشخص از نرم‌افزارها، راهکاری ارائه دهد تا نرم‌افزاری باکیفیت مطابق با نیازهای کاربران نهایی[۲۸] در زمان و بودجه‌ای قابل پیش‌بینی تولید شود.

متدولوژی‌های مختلفی تاکنون ارائه شده‌اند که هرکدام برای دامنه‌ای خاص از نرم‌افزارها مناسب است. از آنجایی که هدف متدولوژی افزایش کیفیت در محدوده زمان و هزینه پروژه می‌باشد، لذا تمام فعالیت‌هایی که در متدولوژی‌های نرم‌افزاری تعریف می‌شوند می‌بایست در همین راستا باشد. از اینرو فعالیت‌های اضافه‌ای که در این راستا نباشند عملا نبایست در آنها وجود داشته باشند. از آنجایی که از ابتدا تا سال ۲۰۰۱ برخی از متدولوژی‌ها سبب کندشدن تولید نرم‌افزار و انجام فعالیت‌های اضافه می‌شدند، در آن سال بیانیه‌ای[۲۹] توسط گروهی

---

[25] Software development methodology
[26] Software development process
[27] Disciplined approach
[28] End user
[29] Manifesto

از مشاورین و پیمانکاران نرم‌افزار ارائه گردید که به "بیانیه توسعه چابک نرم‌فزار"[۳۰] مشهور گردید.

تاکنون تعاریف متعددی برای چابکی در متدلوژی و همچنین متدولوژی‌های چابک متعددی برای طیف‌های گسترده‌ای از نرم‌افزارها ارائه شده است. در مقدمه کتاب مدیریت و نظارت بر پروژه‌های نرم‌افزاری [۱]، به بیان مفاد بیانیه توسع چابک نرم‌افزار پرداخته‌ایم و چابکی متدولوژی را از منظرهای مختلف بررسی کرده‌ایم.

دراین کتاب هرچند متدولوژی مشخصی را مرور نمی‌کنیم ولی بخش عمده‌ای از آموزش که مبتنی بر تولید هدفمند فرآورده‌ها است، براساس متدولوژی RUP[۳۱] [۲] می‌باشد. این متدولوژی توسط شرکت رشنال[۳۲] ایجاد شده است که در سال ۲۰۰۳، IBM آن شرکت را خریداری کرد. هم‌اکنون در وب‌سایت IBM محصولاتی با ایده اولیه رشنال همچنان با همان برند ارائه می‌شود.

همانگونه که در [۱] بیان کرده‌ایم، موضوع اصلی در هر متدولوژی آن است تا برای رسیدن به نرم‌افزاری باکیفیت مورد نظر کاربران با بودجه و زمان مشخص، چه‌کسی[۳۳] باید چه[۳۴]کاری[۳۵] را چگونه[۳۶] و کی[۳۷] انجام دهد و ارتباط نقش[۳۸]هایی که کارها را انجام می‌دهند با هم و همچنین تقدم و تاخر کارها نسبت به هم

[30] Agile Software Development Manifesto
[31] Rational Unifie Process (RUP)
[32] Rational
[33] Who
[34] What
[35] Task
[36] How
[37] When
[38] Role

چگونه است. مسلما در ازای انجام هر کاری تعدادی فرآورده[۳۹] ایجاد می‌شود که این فرآورده‌ها مورد استفاده دیگر نقش‌ها برای تصمیم‌گیری و مشخص نمودن چگونگی انجام دیگر کارها قرار می‌گیرد. تمامی مفاهیم بیان‌شده در اینجا زمانی کاربردی است که این ارتباطات برای نوع خاصی از نرم‌افزار در محیطی[۴۰] مشخص بیان گردد. بنابراین مثلا متدولوژی‌ای که برای توسعه نرم‌افزارهای کاربردی در بستر موبایل در یک شرکت بزرگ تولیدکننده این نوع نرم‌افزارها استفاده می‌شود با متدولوژی که برای توسعه همین نرم‌افزارها توسط یک شخص و یا در شرکتی کوچک است، متفاوت می‌باشد.

لذا یکی از مهمترین فاکتورهایی که اجرای متدولوژی را تحت تاثیر قرار می‌دهد، محیطی است که متدولوژی می‌بایست در آن اجرایی شود. از این‌رو انتخاب متدولوژی مناسب با پروژه نرم‌افزاری موضوع مهمی است که قبل از شروع پروژه می‌باید مشخص شود. در [۳] مروری بر متدولوژی‌های چابک و تناسب آنها با انوع پروژه‌های نرم‌افزاری آورده می‌شود.

متاسفانه در بین مهندسینی که در صنعت نرم‌افزار کشور در حال فعالیت هستند، نگرشی منفی در استفاده از متدولوژی‌های توسعه نرم‌افزار شکل گرفته است. به علت عدم آموزش صحیح متدولوژی‌هایی پرکاربرد نظیر RUP[۴۱] به مدیران پروژه و رشد استفاده از این متدولوژی در پروژه‌هایی که مدیریتی اصولی بر آنها حاکم نبوده‌است، اینان بر این باورند که این متدولوژی در عمل به کار نمی‌آید و سبب افزوده شدن مستندات بی‌استفاده به پروژه و کند شدن روند جریان کار در پروژه می‌شود. در این بین برخی بر این باورند که بایستی از متدولوژی‌های چابک استفاده کرد و باز هم به دلیل عدم آشنایی صحیح با

[39] Artifact
[40] Environment
[41] Rational Unified Process (RUP)

متدولوژی‌های چابک، هر فعالیتی در پروژه را که دانش صحیحی پشت آن نیست را به چابکی در متدولوژی تعبیر می‌کنند.

۳ کنترل کیفیت و تضمین کیفیت نرم‌افزار

توجه: مطالب این فصل برگرفته شده از [۱] می‌باشد.

کنترل کیفیت[۴۲] (QC) به منظور شناسایی و رفع مشکلات در محصول جهت جلوگیری از بروز تولید محصولات بی‌کیفیت شکل گرفته است و تئوری‌های آمار نقش مهمی را در این زمینه ایفا می‌کنند. کنترل کیفیت شامل آزمایش کیفی نمونه‌ها و ارائه‌ی برداشت آماری از آنها می‌باشد. کنترل کیفیت در بخش‌های مختلف فرآیند تولید صورت می‌پذیرد و این کنترل ها توسط افراد آموزش دیده‌ای که درگیر در فرایند هستند انجام می شود.

تضمین کیفیت[۴۳] (QA) سیستمی برای بازبینی بیرونی و نیز یک سری رویه‌های ممیزی است که توسط افراد خارج از فرایند صورت می‌پذیرد. هرچند برای آنکه QA و بازبینی کارهای تعریف شده در فرآیند موثر باشد، مواردی که در بازبینی بررسی می‌شوند در فرآیند کاری گنجانده می‌شود. در بسیاری از اوقات QA فقط در مورد محصول نهایی و پایان فرایند بکار می‌رود، اما یک برنامه تضمین کیفیت موثر و کارا باید تمام مراحل اعم از برنامه ریزی و گام‌های کنترل کیفیت را مورد ارزیابی قرار دهد.

تضمین کیفیت بدین معناست که محصولات و خدمات را درست منطبق با خواست‌های مشتری تحویل دهیم. به عبارتی دیگر، تضمین کیفیت اجرای یکسری کنترلها بر اساس مستندات در مراحل حساس تجاری است به طوری که این اطمینان به وجود آید که دقیقاً نیازهای مشتری تأمین شده است. تضمین کیفیت را با کنترل کیفیت نباید اشتباه کرد. کنترل کیفیت به کیفیت خدمات و تولیدات مربوط می شود. به عبارت دیگر تضمین کیفیت احتمال بروز خطا را که

[42] Quality Control (QC)
[43] Quality Assurance (QA)

موجب ایجاد ضرر برای شرکت و مشتری می‌شود را از بین می‌برد. ضرری که ممکن است از منافع مشتری بکاهد یا حسن شهرت شرکت را از بین ببرد.

کنترل کیفیت در چرخه حیات توسعه نرم‌افزار به معنای آزمون نرم‌افزار در کلیه مراحل می‌باشد. بایستی توجه داشت که آزمون نرم‌افزار در دو سطح کارکردی[۴۴] و غیرکارکردی[۴۵] صورت می‌پذیرد. چگونگی انجام آزمون نرم‌افزار به صورت کاربردی در چرخه حیات توسعه نرم‌افزار و همچنین ایجاد نرم‌افزاری آزمون‌پذیر از اهمیت بالایی برخوردار است.

هرچند که گام‌های آزمون نرم‌افزار به عنوان بخش‌هایی مهم از روش‌های اجرایی تضمین کیفیت نرم‌افزار مطرح می‌باشد ولی تضمین کیفیت نرم‌افزار شامل اجزایی دیگر می‌باشد که تکمیل کننده پازل کیفیت در نرم‌افزار می‌باشد.

در فصل‌های مختلف از [۱]، اجزای مختلف تضمین کیفیت برای دامنه‌ای مشخص از پروژه‌های نرم‌افزاری آورده شده است.

[44] Functional
[45] Non-functional

۴ تولید نرم‌افزار آزمون‌پذیر

همانگونه که در فصل ۳ صحبت شد، آزمون نرم‌افزار در حوزه کنترل کیفیت نرم‌افزار قرار می‌گیرد. برای اینکه بتوان آزمون نرم‌افزار را انجام داد دو رویکرد مبتنی بر اسکریپت[۴۶] و اکتشافی[۴۷] مطرح می‌باشد. در انتهای فصل ۶ به آزمون مبتنی بر اسکریپت خواهیم پرداخت. برای آشنایی با آزمون اکتشافی و چگونگی انجام آن می‌توان به [۶] مراجعه نمود. آزمون اکتشافی به عنوان نوشدارویی در هر پروژه نرم‌افزاری قابل استفاده است. همچنین می‌تواند برخی از اشکالات پنهان که آزمون مبتنی بر اسکریپت قادر به کشف آنها نیست را پیدا کند. ولی در پروژه‌های نرم‌افزاری بزرگ برای رسیدن به کیفیت چاره‌ای جز آزمون مبتنی بر اسکریپت وجود ندارد. هرچند که توصیه می‌شود تیم‌هایی نیز به جستجوی برخی از مشکلات به صورت اکتشافی بپردازند.

برای رسیدن به نرم‌افزای آزمون‌پذیر باید از همان مراحل اولیه در شروع پروژه، اطلاعات مورد نیاز آزمون را نیز فراهم ساخت. بنابراین چنانچه در مصاحبه‌های اولیه با کاربران، درخواست‌های اولیه‌ای مطرح می‌شود، حتما باید جنبه آزمودنی آن را نیز درنظر گرفت. یعنی تحلیلگر باید در همین مرحله اسکریپت آزمونی را تهیه کند تا پس از پیاده‌سازی‌های اولیه سیستم بتوان با اجرای آن اسکریپت از حسن انجام درخواست کاربر مطمئن شد. با همین رویکر در هر مرحله از جمع‌آوری نیازمندی‌ها و یا در مراحل طراحی باید چگونگی آزمون آنها نیز مشخص شود. لذا در هر مرحله از تحلیل و طراحی باید اسکریپت مربوط به آزمون نیز تهیه شود. تهیه اسکریپت آزمون اصطلاحا طراحی آزمون نیز نامیده می‌شود.

[46] Script
[47] Exploratory

بنابراین نرم‌افزاری آزمون‌پذیر است که در هر زمان مستندات آزمون آن نیز تا آن مرحله طراحی شده باشد. لذا پس از تولید نسخه اولیه و یا در زمان کامپایل کدهای برنامه، اسکریپت‌های آزمون نیز می‌تواند به صورت خودکار با ابزارهای آزمون و یا به صورت دستی توسط آزمون‌گران اجرا شود. از این رو همواره گزارشی از چگونگی نتیجه آزمون در اختیار خواهد بود. لذا همواره تیم آزمون می‌تواند اجزای مختلف نرم‌افزار را در محیط‌های مختلف آزمون نماید.

بخش دوم: شناخت نیازمندی‌های نرم‌افزار

۵ مدیریت نیازمندی‌ها

مدیریت نیازمندی‌ها، فرایندی در مهندسی نرم‌افزار است که شامل (۱) مستندسازی نیازمندی‌ها، (۲) آنالیز آنها، (۳) ردیابی نیازمندی‌ها در مراحل مختلف تا پیاده‌سازی و آزمون، (۴) اولویت‌دهی به آنها در پیاده‌سازی و انتشار، (۵) چگونگی توافق ذی‌نفعان مختلف در نوع نیازمندی، (۶) کنترل، تایید و ردیابی تغییرات در نرم‌افزار و (۷) کنترل چرخه تغییرات تا انجام موفق آزمون آن می‌باشد.

فعالیت‌هایی که بواسطه مدیریت نیازمندی‌ها در پروژه صورت می‌پذیرد تنها به زمان خاصی از پروژه محدود نمی‌شود. بلکه در تمامی چرخه حیات[۴۸] پروژه جاری می‌باشد.

هر نیازمندی را می‌توان قابلیتی از سیستم‌نرم‌افزاری تولیدشده در پروژه تعریف کرد که می‌بایست در ارتباط با ذی‌نفعان انجام شود. آغاز مدیریت نیازمندی‌ها با شناخت اهداف و محدودیت‌های پروژه شروع می‌شود. می‌توان مستند مشخصات تجاری محصول[۴۹] را اولین سند مکتوب در این حوزه معرفی نمود.

در پروژه‌هایی که ذی‌نفعان متعددی در ارتباط با مدیریت نیازمندی‌ها در تصمیم‌گیری‌ها موثر می‌باشند، مستندی با عنوان طرح مدیریت نیازمندی‌ها توسط مدیر پروژه تهیه می‌شود. هدف از این طرح بیان رویکرد اتخاذ شده توسط تیم توسعه در مدیریت نیازمندی‌ها می‌باشد. این طرح که می‌باید به تصویب افراد

---

[48] Life cycle
[49] Business Case

موثر در پروژه برسد، اساس تصمیم‌گیری‌های آتی در ارتباط با مدیریت نیازمندی‌ها قرار می‌گیرد. در طرح مدیریت نیازمندی‌ها مواردی که زیر آورده می‌شود مطرح می‌گردد.

۱- افراد و مسئولیت‌ها: ذی‌نفعانی از پروژه که موثر در فرآیند مدیریت نیازمندی‌ها می‌باشند، معرفی می‌شوند. همچنین مسئولیت‌های هرکدام در ارتباط با مدیریت نیازمندی‌ها آورده می‌شود.

۲- ابزارها و زیرساخت[۵۰]: ابزارهایی که در این فرآیند استفاده می‌شود آورده می‌شود. ازمهمترین ابزارهای این حوزه، کنترل نسخه[۵۱] می‌باشد که برای ردیابی نسخه‌های مختلف فرآورده‌های پروژه مورد استفاده قرار می‌گیرد. رشنال کلیرکیس[۵۲] و SVN[۵۳] از ابزارهای کنترل نسخه در مدیریت نیازمندی‌ها می‌باشند. همچنین ابزارهای مدیریت کنترل تغییرات نیز در مدیریت نیازمندی‌ها دارای اهمیت می‌باشد. در ابزارهای مدیریت کنترل تغییرات، تمامی فرآیند از ثبت اشکال یا تغییر و در نهایت آزمون صحت انجام آن مدیریت می‌شود. ابزار رشنال کلیرکوئست[۵۴] را نیز در این حوزه می‌توان نام برد.

۳- شناسایی[۵۵] نیازمندی‌ها: تمامی فرآورده‌هایی از پروژه که می‌بایست قابلیت ردیابی آنها در پروژه وجود داشته باشد تعریف می‌شوند و هدف از ایجاد و پیگیری آنها در پروژه تبیین می‌شود. چگونگی نام‌گذاری و شماره‌گذاری نسخه‌های آنها نیز تعریف می‌شود.

---

[50] Infrastructure
[51] Version control
[52] Rational ClearCase
[53] Apache Subversion (SVN)
[54] Ratioanl ClearQuest
[55] Identification

۴- قابلیت ردیابی[۵۶]: در رشنال، ابزار رکوئیزیت‌پرو[۵۷] برای ثبت نیازمندی‌ها و ثبت ارتباط میان آنها در سطوح مختلف ارائه شده است. در مدیریت نیازمندی‌ها برای هر جزئی از نیازمندی‌ها باید قواعدی تعریف شده وجود داشته باشد که چگونگی ارتباط آن نیازمندی با نیازمندی‌های سطوح بالاتر و یا پایین‌تر قابل تعریف باشد. در این صورت چنانچه مثلا نیازمندی در سطح درخواست ذی‌نفعان، که در فازهای آغازین پروژه قابل شناسایی است، در ابزار ثبت شود می‌بایست در مراحل بعدی ارتباط آن با سطوح بعدی نیازمندی مانند موارد کاربری و یا نیازمندی‌هایی در حوزه غیرکارکردی مشخص گردیده باشد. با چنین قاعده‌ای همواره می‌توان گزارشی را در اختیار داشت که کدامین بخش از درخواست‌های ذی‌نفعان تاکنون با ویژگی‌هایی کارکردی و یا غیرکارکردی از سیستم نرم‌افزاری پوشش داده نشده است.

۵- ویژگی‌های نیازمندی‌ها: در طرح مدیریت نیازمندی‌ها، باید بتوان ویژگی‌هایی برای هر جزء از نیازمندی تعریف کرد و توضیحاتی کافی درباره آنها بیان نمود. در ادامه ویژگی‌هایی که می‌توان برای نیازمندی‌ها تعریف کرد و مقادیری که به آنها قابل تخصیص است آورده شده است.

الف) وضعیت: هر نیازمندی از ابتدای ورود به سیستم در طول مراحل مختلف می‌تواند وضعیت‌هایی مانند پیشنهادشده[۵۸]، تصویب‌شده[۵۹]، ردشده[۶۰]، ترکیب‌شده با دیگر نیازمندی‌ها[۶۱] داشته باشد. البته باید توجه داشت که مقادیر ممکن برای

[56] Traceability
[57] Ratioanl Requisite Pro
[58] Proposed
[59] Approved
[60] Rejected
[61] Incorporated

وضعیت نیازمندی‌ها، به پروژه و ماهیت آن وابسته است و پس از تعریف در طرح مدیریت نیازمندی‌ها باید در ابزاری مانند کلیرکوئست تعریف شود.

ب) اولویت: به هر نیازمندی با توجه به اثربخشی که بر منافع حاصل از سیستم نرم‌افزاری دارد، می‌توان اولویت‌های مختلفی نسبت داد تا بر همان اساس در برنامه‌ریزی‌های پروژه مشارکت داده شود. برای اولیت می‌توان مقادیری مانند حیاتی، مهم و مفید درنظر گرفت.

ج) نفرساعت: تخمین میزان تلاشی که تیم توسعه برای نیازمندی یا اشکال می‌بایست انجام دهند را بیان می‌کند.

د) ریسک: میزانی از ریسک که توسط نیازمندی به پروژه وارد می‌شود را بیان می‌کند. ممکن است اضافه شدن نیازمندی جدیدی در نرم‌افزار سبب شود تا اتفاقات غیرقابل پیش‌بینی در کارکرد دیگر اجزای نرم‌افزار حاصل شود. مقدار ریسک می‌تواند با پارامترهایی مانند بالا، متوسط و کم ارزش‌گذاری شود.

ه) پایداری[۶۲]: ممکن است نیازمندی جدیدی که در حال حاضر در پروژه مطرح گردیده است تغییر کند و یا برداشت تیم پروژه از آن هم‌اکنون اشتباه باشد. پایداری را می‌توان به نوعی ریسک در فهم صحیح نیازمندی تعبیر کرد. نیازمندی‌ها را از نظر پایداری مانند ریسک می‌توان با مقادیری مانند بالا، متوسط و کم مقداردهی کرد. مسلما نیازمندی‌هایی که براساس نظر تیم پروژه از پایداری کمی برخوردار باشند، در اولویت پایینی برای اجرا قرار می‌گیرند.

و) انتشار در نسخه هدف: برای هر نیازمندی می‌توان برنامه‌ریزی کرد که در کدام انتشار برای اولین بار پیاده‌سازی گردد.

[62] Stability

ز) تیم تخصیص داده شده: در پروژه‌های بزرگ نرم‌افزاری، نیازمندی‌ها و ویژگی‌های نرم‌افزاری به تیم‌های مجزایی تخصیص داده می‌شود. با این رویکرد می‌توان مسئولیت تیم‌های مختلف را بهتر بیان نمود.

ح) دلیل: به صورت متنی مشخص می‌شود که چه مرجعی سبب مطرح شدن این ویژگی در سیستم گردیده است.

۶- گزارش‌ها و معیارهای ارزیابی[۶۳]: فرمت، محتوی و هدف گزارش‌هایی که در خصوص مدیریت نیازمندی‌ها میان ذی‌نفعان پروژه ارسال می‌شود مشخص می‌شود. همچنین معیارهایی که برای سنجش مدیریت نیازمندی‌ها می‌تواند مناسب باشد، تعریف می‌شود.

۷- تعریف فرآیند کنترل تغییرات: این فرایند چگونگی تغییر وضعیت هر تغییر یا اشکال جدید را از زمان اضافه شدن، بازنگری شدن و در نهایت اعمال در سیستم آزمون نهایی را دربر می‌گیرد. این فرآیند می‌تواند علاوه بر تعریف به صورت متنی در ابزار مدیریت کنترل تغییرات مانند کلیرکوئست نیز اعمال گردد.

۸- هیات کنترل تغییرات[۶۴]: افرادی که از طرف تیم توسعه، بهره‌بردار و یا کارفرما مسئولیت تصویب تغییرات در نرم‌افزار را بر عهده دارند مشخص می‌شوند.

۱۱- آموزش و ابزارها: ابزارها و آموزش‌هایی که برای اجرای صحیح طرح مدیریت نیازمندی‌ها مورد نیاز می‌باشد آورده می‌شود.

[63] Measures
[64] Change control board (CCB)

۶ توصیف نیازمندی‌های کارکردی

همانگونه که در فصل‌های قبل تاکید شد، نیازمندی‌های نرم‌افزار در دو حوزه عمده کارکردی و غیرکارکردی تقسیم می‌شود. هر نیازمندی کاکردی را می‌توان به صورت تابعی[۶۵] $y = f(x)$، مدل کرد که در آن ورودی‌ها (x)، رفتار نرم‌افزار بواسطه آن ورودی‌ها (f) و همچنین خروجی‌ها (y) مشخص می‌باشند. بنابراین بسته به این که در چه مرحله از جمع‌آوری نیازمندی‌ها می‌باشیم، شیوه توصیف ورودی، رفتار و خروجی می‌تواند متفاوت باشد. در مراحل اولیه توصیف نیازمندی‌ها، از آنجایی که هدف ما شناخت دامنه کلی نیازمندی‌ها است، مسلما توصیف کارکردها با ابهام همراه است. همچنین در مراحل اولیه، برای توصیف کارکرد به رفتار و چگونگی تبدیل ورودی‌ها به خروجی‌ها پرداخته نمی‌شود. این درحالیست که بیان رفتار و چگونگی رسیدن به خروجی در تابع از مهمترین بخش‌های آن می‌باشد. لذا برای انکه در ابتدای شناخت نیازمندی‌ها، با تمرکز بر چگونگی رفتار، کارکردهای مختلف از قلم نیفتد، بنابراین چگونگی رفتار مورد بررسی قرار نمی‌گیرد. این روند در بخش اول و دوم این فصل کاملا ملموس است. ولی در بخش‌های سوم به بعد بیان چگونگی انجام بیشتر مورد توجه قرار می‌گیرد

۶,۱ دریافت درخواست ذی‌نفعان

در دریافت درخواست ذی‌نفعان، هر نوع درخواستی که ممکن است از سیستمی که در آینده توسعه داده می‌شود داشته باشند، آورده می‌شود. ذی‌نفعان ممکن است کارفرما، کاربران نهایی و هر فردی که در ارتباط با سیستم نرم‌افزاری است باشند. همچنین درخواست‌ها در این حوزه صرفا به کارکردها محدود نمی‌شود و مسائل غیر کارکردی مانند کارآیی را نیز دربر می‌گیرد. هرچند توصیف نیازمندی‌های کارکردی و غیر کارکردی در ادامه روند توسعه نرم‌افزار با جزئیات

[65] Function

بیشتری به تصویب ذی‌نفعان می‌رسد، ولی در این مرحله نیز می‌توان از دیدگاه کابران و بهره‌برداران سیستم آنها را ثبت کرد.

یکی از روش‌های اصلی در دریافت درخواست ذی‌نفعان انجام مصاحبه با آنها است. هرچند روش‌هایی مانند (۱) مشاهده و بررسی سیستم‌های گذشته ایشان، (۲) مشاهده و بررسی مدارک موجود آنها که در ارتباط با سیستم در جریان می‌باشد، (۳) حضور در محیطی که قرار است سیستم نرم‌افزاری در آن اجرایی گردد، (۴) جستجو در منابع اینترنتی و هر روش دیگری که به فهم بیشتر از نیازهای ذی‌نفعان کمک می‌کند، نیز در این حوزه موثر می‌باشد.

پس از انجام مصاحبه، یا روش‌های دیگر در دریافت درخواست ذی‌نفعان، مستندی با همین عنوان تکمیل می‌شود. در ادامه این بخش هرچند برآن نیستیم تا قالبی برای این مستند ارائه دهیم ولی سعی می‌کنیم به بخش‌های اصلی آن که در روند مصاحبه اهمیت دارد پرداخته شود.

۱- ثبت مشخصات ذی‌نفع

- نام: سازمان/موسسه:
- عنوان شغلی:
- مسئولیت اصلی شما در سازمان چیست؟
- چه مستندات یا خروجی‌هایی را شما تولید می‌کنید؟ و آنها را برای چه کسانی ارسال می‌کند؟
- موفقیت شما در کار چگونه و با چه معیارهایی سنجیده می‌شود؟
- چه مشکلاتی می‌توان مانع رسیدن شما به موفقیت شود؟
- چه اموری سبب می‌شود تا کار شما آسان‌تر و یا سخت‌تر شود؟

۲- بررسی مشکل

- بر هر مشکلی که قبلا بیان شد، آیا راه‌حل‌های مناسبی وجود دارد؟
- راه‌حل‌ها چه هستند؟ آیا باز هم راه حلی می‌توانید بیان کنید؟

برای هر مشکل پرسش‌های زیر را مطرح کنید:

- به نظر شما چرا این مشکل از اساس وجود دارد؟
- در وضعیت فعلی چگونه آن را حل می‌کنید؟
- انتظار دارید در آینده چگونه این مشکل مرتفع گردد؟

۳- درک محیط کاربر

- کاربرانی که در این محیط کار می‌کنند چه کسانی هستند؟
- چه سطح تحصیلی و یا آموزشی دارند؟
- چه پیش‌زمینه‌ای در ارتباط با کامپیوتر و ابزارهای مرتبط با آن دارند؟
- آیا کاربران برای اولین بار است که قرار است با چنین نوعی از نرم‌افزار کار کنند یا قبلا نیز با نرم‌افزارهای مشابه دیگری کار کرده‌اند؟
- درحال حاضر نرم‌افزارهایی که استفاده می‌کنند در چه بستری[۶۶] می‌باشد؟
- طرح ایشان برای بسترهای بعدی چه می‌باشد؟
- ازچه برنامه‌های کاربردی دیگری استفاده می‌کنید که در این پروژه نیز باید با آنها ارتباط برقرار کنیم؟
- انتظار شما از راحتی استفاده[۶۷] نرم‌افزاری که در آینده قرار است تولید شود چه می‌باشد؟
- انتظار شما از مدت زمان آموزش برای کاربرد نرم‌افزار جدید چیست؟

[66] Platform
[67] Usability

- چه مستنداتی را نیاز دارید تا آموزش نرم‌افزار به صورت نسخه فیزیکی[۶۸] و یا به صورت آنلاین در اختیار شما قرار گیرد؟

۴- تلاش برای درک بیشتر

- مشکلاتی که کاربر بیان کرده است را مجدد با بیان خودتان مطرح نمایید. بنابراین در گفتار بگویید "شما گفتید که ..." . سپس سوال کنید که آیا اینها مشکلاتی هستند که شما در ارتباط با سیستم جاری با آنها درگیر هستید؟
- آیا مشکلات دیگری هم هستند که تجربه کرده باشید؟

۵- اعتبارسنجی فرضیه‌هایی که در ارتباط با مصاحبه به دست آمده است

دراین مرحله، مشکلات دیگر یا نیازهای دیگری که با توجه به تجربه خودتان به ذهنتان می‌رسد که کاربر با آنها درگیر است و یا مورد نیازش می‌باشد را مطرح نمایید. سپس برای هر مشکل پیشنهادی، پرسش‌های زیر را از کاربر بپرسید:

- آیا این مشکلی که من مطرح کردم واقعا در سیستم وجود دارد؟
- به نظر شما علل وقوع چنین مشکلی چه می‌باشد؟
- در حال حاضر چگونه این مشکل را حل می‌کنید؟
- به نظرتان بهتر است در آینده چگونه این مشکل حل شود؟
- چنانچه بخواهید حل این مشکلات را نسبت به مشکلاتی که قبلا بیان کردید اولویت بندی کنید، این اولویت به نظر شما چگونه است؟

۶- بررسی راه‌حل پیشنهادی

[68] Hard copy

- قابلیت‌های بخشی از راه حل پیشنهادی خود را بیان کنید و سوال کنید که اگر اینگونه مشکل حل شود به نظر شما چگونه خواهد بود؟
- به نظر شما اولویت بخش‌های مختلف راه‌حل پیشنهادی چه می‌باشد؟

۷- بررسی امکان جاری‌سازی نرم‌افزار

- به نظر شما اجرایی سازی نرم‌افزاری که قرار است در این پروژه تولید شود، چه نیازمندی‌های اولیه‌ای دارد؟
- به نظر شما چه بخشی از کاربران تمایل به استفاده از این نرم‌افزار را دارند و یا می‌توانند از آن استفاده کنند؟
- به نظر شما یک راه حل موفق باید دارای چه ویژگی‌هایی باشد.

۸- بررسی نیازمندی‌های مرتبط با کارایی

- شما چه انتظاری از قابلیت اعتماد نرم‌افزار دارید؟
- شما چه انتظاری از کارایی[۶۹] نرم‌افزار دارید؟
- آیا پشتیبانی از این نرم‌افزار در آینده بر عهده شما خواهد بود و یا دیگران این وظیفه را بر عهده خواهند گرفت.
- آیا برای پشتیبانی از این نرم‌افزار، نیازمندی مشخصی دارید که سبب تسهیل کار شما در آینده شود.
- چه نیازمندی‌هایی در ارتباط با امنیت از این نرم‌افزار دارید؟
- در ارتباط با نصب و تنظیمات آن آیا پیشنهاد و یا نیازمندی مشخصی دارید؟

[69] Performance

- در ارتباط با محدودیت‌های مرتبط با مجوز[۷۰] آیا نیازمندی مشخصی دارید؟
- کارکردهای نرم‌افزار چگونه در سطح سازمان توزیع می‌شود؟
- برای شیوه نام‌گذاری نسخه‌های مختلف نرم‌افزار آیا پیشنهاد مشخصی دارید؟

نیازمندی‌های دیگر:

- آیا به نظر شما نرم‌افزار باید استانداردهای مشخصی را پشتیبانی کند؟
- آیا نیازمندی‌های دیگری می‌توانید در این حوزه برای نرم‌افزار بیان کنید؟

۹- جمع‌بندی مصاحبه

آیا پرسش دیگری هست که به نظرتان من مطرح نکرده‌ام؟

- اگر در حین بررسی این نیازمندی‌ها به پرسش‌های دیگری مواجه شوم آیا می‌توانم تلفنی با شما تماس بگیرم؟
- احتمالا در آینده‌ای نزدیک باید نیازمندی‌هایی که از کابران مختلف جمع‌آوری شده است را در جلسه‌ای مطرح کنیم و با حضور ذی‌نفعان مختلف بخواهیم آنها را بررسی کنیم. آیا تمایل دارید در چنین جلسه‌ای مشارکت داشته باشید؟

۱۰- بیان خلاصه نیازها توسط آنالیست

[70] Licensing

در این بخش تعدادی از نیازمندی‌های شاخص را که دستاورد این جلسه است را برای ذی‌نفع به صورت خلاصه بیان کنید.

۶,۲ ایجاد موارد کاربری

موارد کاربری[۷۱] در فعالیت‌های مرتبط با شناخت، آنالیز، طراحی، پیاده‌سازی و نهایتا آزمون در معماری شیء گرایی نقشی کلیدی برعهده دارد. این اهمیت همچنان در معماری‌های جدیدتر مانند سرویس‌گرا و فرآیندمحور همچنان نیز مورد توجه می‌باشد.

می‌توان تعریف زیر را برای مورد کاربری ارائه داد:

در هر مورد کاربری توصیف رفتار سیستم نرم‌افزاری تحت شرایط مختلف در پاسخ‌هایی که به درخواست‌های ذی‌نفعی مشخص داده می‌شود، ثبت می‌گردد. در هر مورد کابری، ذی‌نفع مشخص را کنشگر اصلی[۷۲] می‌نامند.

۶,۲,۱ کنشگر اصلی

در هر مورد کاربری، کنشگر اصلی، آغاز کننده ارتباطی فعال[۷۳] با سیستم نرم‌افزاری می‌باشد تا به هدفی مشخص دست یابد. بنابراین در هر مورد کاربری دنباله‌های متعددی از رفتارها یا سناریوها می‌تواند بازشناسی شود. مسلما در موارد کاربری، هر رفتار بواسطه شرایط خاصی که در ارتباط با درخواست کنشگر اصلی وجود داشته است، حاصل می‌شود. بنابراین با بیانی دیگر می‌توان هدف از هر مورد کاربری را گردآوری تمامی سناریوها و رفتارهای سیستم در ارتباط با درخواست کنشگر اصلی آن بیان نمود.

---

[71] Use Cases
[72] Primary actor
[7373] Interaction

۶,۲,۲ بدنه اصلی

موارد کاربری اساسا به صورت نوشتاری تهیه می‌شوند. هرچند بسته به نیاز می‌توان از فلوچارت، سودوکد[۷۴] و یا هر روش دیگری که کمک به توصیف بهتر روند کار تا آن مرحله می‌کند، استفاده کرد.

موارد کاربری از آنجایی که برای انتقال مطلب میان تهیه‌کننده آن با ذی‌نفعان دیگر و یا دیگر افراد تیم توسعه استفاده می‌شود، لذا داشتن متنی ساده و روان را می‌توان اولین ویژگی هر مورد کاربری موثر دانست.

استفاده از موارد کاربری را می‌توان به صورت زیر بیان کرد:

الف) در تیم‌های توسعه نرم‌افزار می‌تواند برای بحث درباره ویژگی‌های بخش‌های مختلف نرم‌افزار در راستای افزایش شناخت تیم موثر باشد.

ب) در تیم‌های توسعه نرم‌افزار پس از آنکه با بحث‌های متعدد درباره کارکردهای مختلف نرم‌افزار تصمیم‌گیری انجام گرفت، می‌تواند برای مستندسازی نیازمندی‌های نرم‌افزار استفاده شود.

ج) این امکان نیز وجود دارد که نیازمندی‌های شناسایی شده در قالب موارد کاربری در اختیار تیمی دیگر برای تکمیل طراحی قرار گیرد.

د) موارد کاربری می‌توانند به عنوان رسانه‌ای برای انتقال شناخت بین تیم توسعه نرم‌افزار و دیگر ذی‌نفعان صورت پذیرد. بحث‌های متعددی می‌تواند درباره رفتارهای سیتم در شرایط مختلف صورت پذیرد و در نهایت توافقات نهایی در این مستند انجام شود.

[74] Psedu code

ه) طراحان آزمون با بهره‌گیری از این مستند می‌توانند موارد آزمون[۷۵] را طراحی کنند.

و) چنانچه در پروژه‌ای موارد آزمون به صورت جامع تهیه نشده باشد، موارد کاربری می‌تواند نقطه مناسبی برای شروع آزمون اکتشافی قرار گیرد.

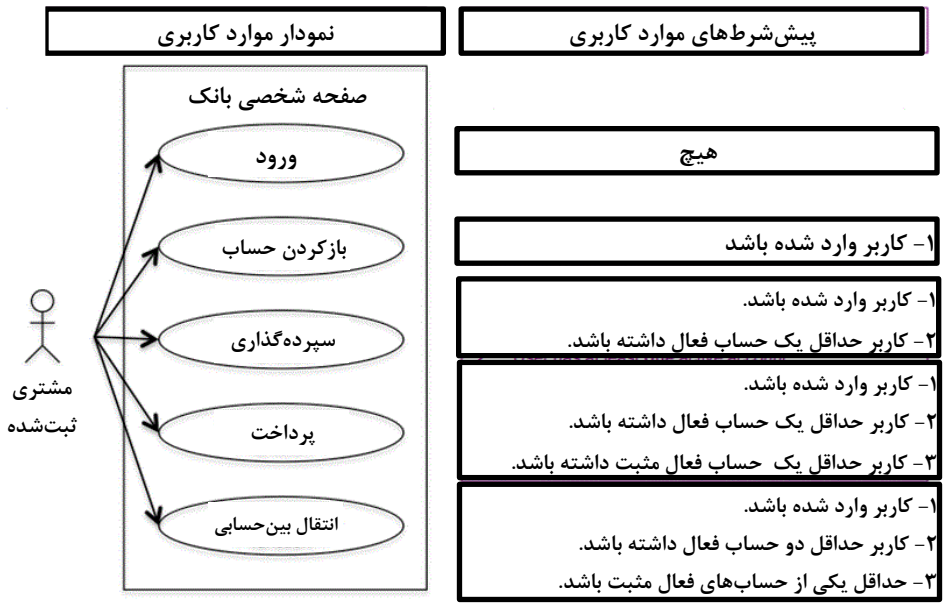
در هر مورد کاربری باید کنشگر اصلی مشخص شده باشد. معیار برای تشخیص کنشگر اصلی در هر مورد کاربری، نقشی در سازمان است که هدف از وجود آن مورد کاربری، پاسخ دادن به نیازمندی مشخصی از آن نقش می‌باشد. مثلا در سیستم آموزشی دانشگاه، هدف از مورد کاربری درخواست انتقال، ساده‌سازی و فرایند ثبت درخواست دانشجو برای انتقال از یک دانشگاه به دانشگاهی دیگر می‌باشد. لذا دانشجو، کنشگر اصلی در مورد کاربری درخواست انتقال می‌باشد. هرچند کنشگرهای دیگری مانند استاد راهنما، مدیر گروه، کارمند آموزش و یا مدیر آموزش نیز در رسیدن به هدف این مورد کاربری در موردهای کاربری دیگر می‌توانند کنشگر اصلی باشند، ولی دانشجو تعاملی انسانی با این مورد کاربری دارد.

۶,۲,۳ شرایط پیش‌نیاز و پایانی

برای هر مورد کاربری، در بخشی شرایط پیش‌نیاز[۷۶] نیز مشخص می‌شود. شرایط پیش‌نیاز هر مورد کاربری، شرایطی هستند که باید در ارتباط با داده‌ها و یا وضعیت کنشگر اصلی در سیستم تفاق افتاده باشد تا مورد کاربری بتواند آغاز گردد. مثلا در مورد کاربری درخواست انتقال، اجرای موفق مورد کاربری ورود به سیستم[۷۷] جزء پیش‌شرط‌های اولیه می‌باشد. چرا تا کاربری به عنوان دانشجو به

[75] Test cases
[76] Preconditions
[77] Login

سیستم نرم‌افزاری وارد نشده باشد نمی‌تواند از این مورد کاربری استفاده کند. همچنین سه ترم مهمانی دانشجو می‌تواند پیش‌نیاز دیگری برای شروع روال اصلی این مورد کاربری باشد. شکل زیر مثالی دیگر از موارد کاربری و شرایط پیش‌نیاز آنها را نشان می‌دهد.

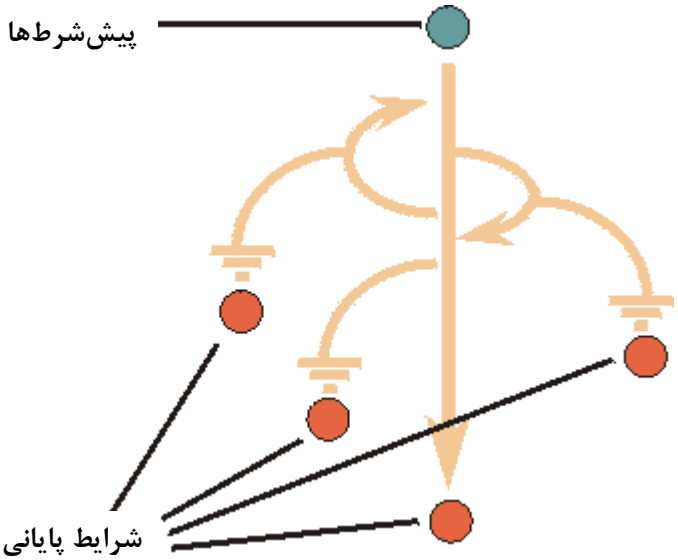


شکل ۱) مثالی از نمودار موارد کاربری در سیستم بانکداری الکترونیک به همراه پیش‌شرط‌هایی برای موارد کاربری

همچنین برای هر مورد کاربری، در بخشی دیگر، شرایط پایانی[۷۸] نیز آورده می‌شود. در هر مورد کاربری، روند از نقطه مشخصی شروع می‌شود و می‌تواند در نقاط مختلفی نیز پایان یابد. بنابراین هر نقطه پایانی می‌توان سبب شرایط پایانی مختص خود در مورد کاربری گردد. لذا اجتماع تمامی شرایط پایانی، شرایط

[78] Postconditions (gauranties)

پایانی مورد کاربری را مشخص می‌کند. شکل زیر نمایش خوبی از حالات متعدد شرایط پایانی در یک مورد کاربری فرضی را نشان می‌دهد.



شکل ۲) یک شروع و پایان‌های متعدد در مورد کاربری

مثلا برای مورد کابری پرداخت در سیستم بانکداری الکترونیک می‌توان شرایط پایانی مانند (۱) بین موجودی حساب و تراکنش‌های آن بایست تعادل وجود داشته باشد و (۲) کاربر به صفحه اولیه انتقال داده می‌شود در نظر گرفت.

۶,۲,۴ سناریوی اصلی

همانگونه که در شکل ۲ نیز مشاهده می‌شود، هر مورد کاربری شامل تنها یک سناریوی اصلی است. در سناریوی اصلی که قرار است معمولا اتفاق بیفتد، سیر تعامل‌های کنشگر اصلی با سیستم نرم‌افزاری بیان می‌شود. قاعدتا در سناریوی اصلی و دیگر سناریوی‌های مورد کاربری، تعامل میان کنشگر اصلی و سیستم نرم‌افزاری است. لذا در فرآیند سناریوی اصلی نمی‌توان تواقفی بواسطه تعامل با

کنشگر دیگری که به صورت برخط قادر به پاسخ‌دهی نیست ایجاد کرد. چنانچه این تعامل بواسطه الزاماتی بایست ایجاد گردد، عملا سناریوی اصلی یا دیگر سناریوها می‌بایست به پایان برسند تا ادامه کار در موارد کاربری دیگر ادامه یابد.

مثلا فرض کنید که در مورد کاربری افتتاح حساب، بعد از اینکه مشتری که کنشگر اصلی این مورد کاربری می‌باشد اطلاعات خواسته شده را وارد کرد نیاز به تایید از جانب کارمندان بانک داشته باشد. نمی‌توان این روند را در همین مورد کاربری قرار داد. به گونه‌ای که مشتری منتظر باشد تا در آینده‌ای نزدیک این تایید صورت پذیرد. بلکه، مورد کاربری بدون نیاز به این تایید پایان می‌پذیرد و تایید حساب‌های تازه افتتاح شده در مورد کاربری دیگری که کنشگر اصلی آن کارمند بانک است انجام می‌شود.

۶,۲,۵ سناریوهای فرعی

همانگونه که قبلا هم نیز بیان شد، روند معمول در سناریوی اصلی بیان می‌شود. ممکن است در یکی از گام‌های سناریوی اصلی، شرایطی که برای تحقق آن بیان شده است، در هنگام رسیدن کنشگر اصلی به آن گام محقق نشده است. مثلا در بخشی از سناریوی پرداخت در سیستم بانک‌داری الکترونیک گام‌های زیر آمده است:

...

۴- مبلغ پرداختی به ریال وارد می‌شود.

۵- چنانچه (۱) موجودی کاربر از مبلغ وارد شده به اضافه کارمزد بیشتر یا مساوی باشد و یا (۲) برای حساب حداقل موجودی تعیین شده باشد که کسر مبلغ پرداخت و کارمز سبب کمتر شدن موجودی از حداقل موجودی نگردد، از موجودی مبلغ انتقال و کارمزد آن کسر می‌شود

۶- مبلغ کسر شده و کارمزد آن به کاربر نشان داده می‌شود.

....

همانگونه که مشاهده می‌شود ممکن است برای مشتری که در تعامل با سیستم مبتنی بر این مورد کاربری است، شرایط خواسته شده در گام ۵ محقق نشود. لذا ادامه سناریوی اصلی برای مشتری امکان‌پذیر نمی‌باشد. بنابراین باید برای این حالت سناریویی در مورد کاربری پیش‌بینی شده باشد. هر سناریوی فرعی در مورد کاربری می‌تواند عنوانی داشته باشد. ولی شماره آن همان شماره گامی در سناریوی اصلی است که بواسطه محقق نشدن شرایط آن به این سناریوی فرعی منتقل شده‌ایم. در ادامه سناریوی فرعی به ازای گام پنجم آمده است.

۵-۱- کافی نبودن موجودی حساب

۱- پیغامی با کد Msg023 مبنی بر آنکه چه مبلغی کسری در حساب شما برای این انتقال وجود دارد نشان داده می‌شود.

۲- از مشتری با پیغامی با کد Msg024 پرسیده می‌شود که آیا مایل است مبلغ کمتری برای انتقال وارد کند.

۳- گزینه بله و خیر به مشتری نشان داده می‌شود.

۴- در صورتیکه پاسخ مشتری بله باشد، مراحل از گام ۴ سناریوی اصلی دنبال می‌شود.

۵- درصورتیکه پاسخ منفی باشد، از کاربر با پیغام Msg025 پرسیده می‌شود که آیا مایل است پرداخت از حساب دیگری انجام شود.

۶- درصورتیکه پاسخ مثبت باشد و کاربر حساب‌های دیگری در بانک داشته باشد، در فیلدی که به صورت لیست نیز قابل انتخاب است، حساب‌های کاربر در بانک نشان داده می‌شود.

۷- کاربر حساب مورد نظر را انتخاب می‌کند.

۸- مراحل از گام ۴ سناریوی اصلی دنبال می‌شود.

۹- درصورتیکه پاسخ منفی باشد، پیغامی با کد Msg026 به کاربر نشان داده می‌شود که باگزینه دستور پرداخت در آینده می‌توانید انتقال به حساب را با روش دیگری انجام دهید.

۱۰- سیستم پیامکی مبنی بر عدم موفقیت در پرداخت برای کاربر ارسال می‌کند.

۱۱- پایان

همچنین ممکن است در یکی از گام‌های سناریوی فرعی نیز شرایطی گذاشته شود که بواسطه محقق شدن آنها، گام‌های سناریوی فرعی ادامه می‌یابد. در این موارد نیز همانند سناریوی اصلی در صورت عدم تحقق شرط‌ها باید یک یا چند سناریوی فرعی دیگر نیز بیان شود. مثلا در گام ۶ از سناریوی فرعی ۵-۱، چنانچه کاربر حساب‌های دیگر در بانک نداشته باشد مراحل بعد نمی‌تواند ادامه یابد. بنابراین سناریوی فرعی دیگری به صورت زیر ایجاد می‌شود که وظیفه بیان دنباله کارها درصورت عدم برقراری شرط را بر عهده دارد.

۵-۱-۶- عدم وجود حساب دیگر

۱- پیغامی با کد Msg027 به کاربر نشان داده می‌شود که برای شما حساب دیگری در بانک تعریف نشده است.

۲- در پیغام دیگری با کد Msg028 به کاربر نشان داده می‌شود که چه حساب‌های متنوعی برای پس‌انداز در این بانک وجود دارد و عملا تبلیغات هدف‌مند برای جذب سرمایه‌های مشتریان انجام می‌شود.

۳- از کاربر پرسیده می‌شود که آیا تمایل دارد حساب پس‌اندازی در بانک افتتاح کند.

۴- در صورت پاسخ کاربر مثبت باشد، این مورد کاربری پایان می‌یابد و کاربر به مورد کاربری افتتاح حساب هدایت می‌شود.

۵- در صورتیکه پاسخ کاربر منفی باشد، این مورد کاربری پایان می‌یابد.

همانگونه که در گام ۵ سناریوی اصلی مورد کاربری پرداخت مشخص است، دو شرط وجود دارد که عدم تحقق هرکدام از آنها سبب توقف سناریوی اصلی می‌شود. در سناریوی فرعی ۵-۱، سناریویی فرعی برای دنبال کردن مورد کاربری در عدم تحقق بخش اول آورده شده‌است. در سناریوی فرعی دیگر که با شماره ۵-۲ مشخص می‌شود، می‌بایست روند مورد کاربری در صورت عدم تحقق بخش دوم بیان شود.

۶,۲,۶ بیان تشابه در موارد کاربری

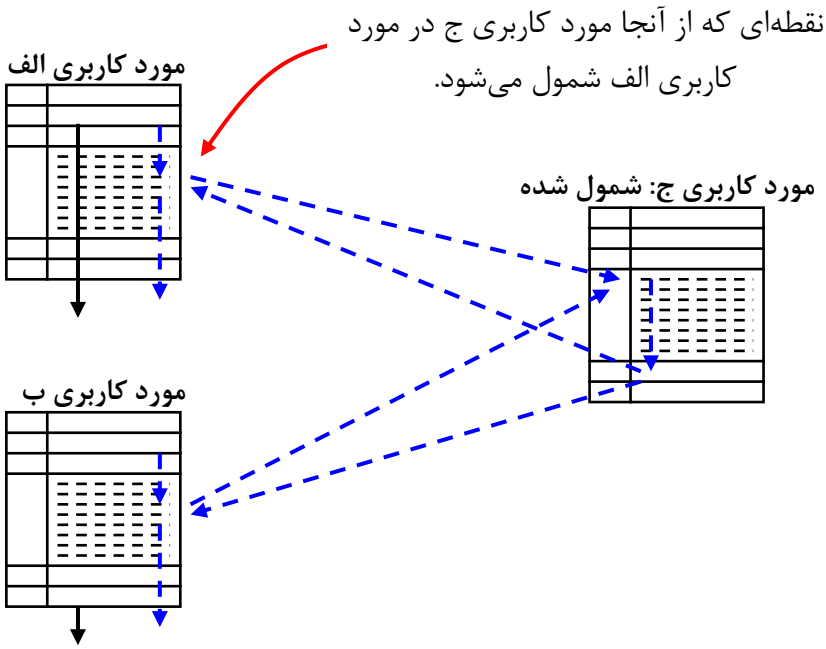
همانگونه که در طراحی جدول‌های پایگاه داده سعی می‌کنیم جدول‌ها را برای پیش‌گیری از افزونگی نرمال کنیم، در ایجاد موارد کاربری نیز می‌بایست با نرمال‌سازی موارد کاربری از بیان مجدد یک منطق در چندین مورد کاربری جلوگیری کنیم. در نرمال‌سازی موارد کاربری از سه رویکرد شمول[۷۹]، گسترش[۸۰] و تعمیم[۸۱] استفاده می‌شود.

۱- شمول

یکسری از مراحل می‌تواند در موارد کاربری متعددی یکسان باشد. در ابتدای نوشتن موارد کاربری نمی‌توان به سادگی بخش‌هایی از سناریوها که در چند مورد کاربری یکسان است را پیدا کرد. پس از اینکه سناریوهای چند مورد کاربری

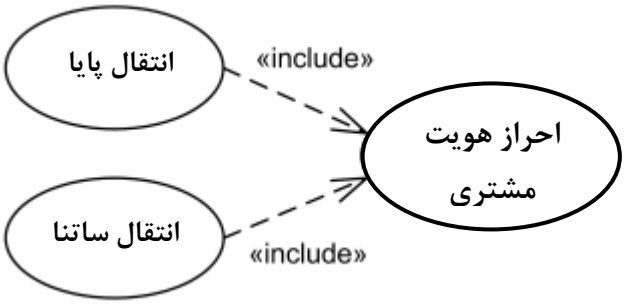
[79] Inclusion
[80] Extension
[81] Generalization/specialization

نوشته شد، مشخص می‌شود که بخش‌هایی از آنها تکراری است. در شمول به دنبال آن هستیم تا بخش‌های تکراری را در موارد کاربری مجزا گردآوری کنیم و در موارد کاربر دیگر به آنها ارجاع دهیم. در شکل زیر چگونگی شمول یک مورد کاربری در دو مورد کاربری دیگر نشان داده شده است.



شکل ۳) شمول یک مورد کاربری در دو مورد کاربری دیگر

در شکل زیر همچنین با نمادهایی مبتنی بر UML چگونگی شمول در موارد کاربری نشان داده شده است.

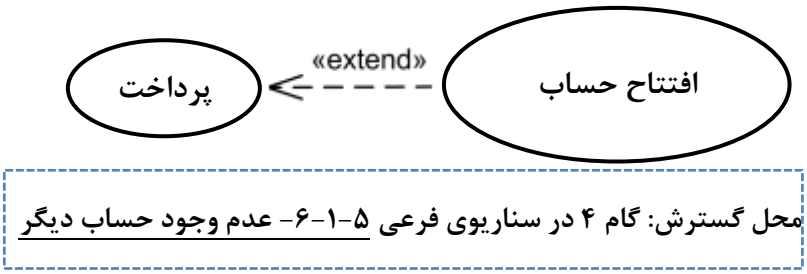


شکل ۴) مثالی از شمول مبتنی بر UML

نکته‌ای که باید در شمول مورد نظر باشد، این است که مثلا در شکل ۴، انتقال پایا بدون احراز مشتری در سناریوی اصلی آن امکان‌پذیر نمی‌باشد.

۲- گسترش

نموداری از موارد کاربری که در آن از گسترش استفاده شده است در شکل زیر آورده شده است.



شکل ۵) نمونه‌ای از گسترش در نمودار موارد کاربری

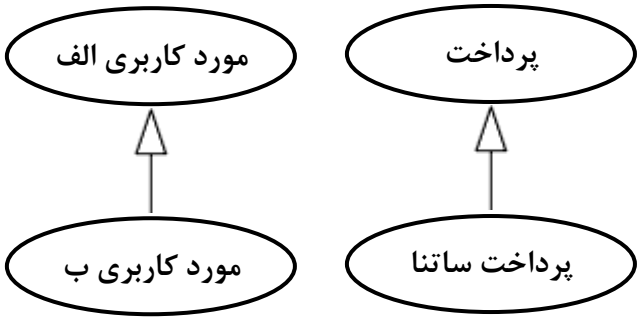
تفاوت گسترش با شمول آن است که روندهایی که در مورد کاربری شمول‌شده آمده است بخش اصلی از روند مورد کاربری است که به آن ارجاع داده

است. ولی در گسترش، روندهای آورده شده در آن در سناریوهای فرعی استفاده می‌شود.

در گسترش باید گامی که بواسطه انتخابی در آن به مورد کاربری گسترش داده شده مراجعه می‌شود مشخص گردد. مثلا در شکل ۵، محل گسترش[۸۲] آورده شده است.

۳- تعمیم

در بیان موارد کاربری زمانی که مثلا مورد کابری الف کاری تقریبا مشابه با مورد کاربری ب انجام می‌دهد ولی در بخش‌هایی به صورت خاص‌تر، کارهایی انجام می‌دهد که در مورد کاربری ب به آنها اشاره‌ای نشده است، در این صورت می‌توان ارتباط بین این دو مورد کاربری را با شکل زیر نشان داد.



شکل ۶) نمودار موارد کاربری که در آن از تعمیم استفاده شده است

بنابراین هرچند هر دو مورد کاربری پرداخت و پرداخت ساتنا به صورت جداگانه نوشته می‌شوند، ولی بخش‌هایی از روند پرداخت ساتنا که مشابه روند پرداخت است به صورت رنگی مشخص می‌گردد.

[82] Extension point

۶,۲,۷ اهمیت موارد کاربری

استفاده از موارد کاربری بسیار موثر می‌باشد. چرا که در آنها روایتی[۸۳] بیان می‌شود که چگونه سیستم نرم‌افزاری در ارتباط با کاربران و درخواست‌های آن رفتار می‌کند. کاربران سیستم با مطالعه موارد کاربری درک لازم از اینکه سیستم جدید چه ویژگی‌هایی در اختیار آنها قرار می‌دهد بدست می‌آورند. بنابراین با دنبال کردن روندهای بیان شده در قالب سناریوهای متعدد، قبل از پیاده‌سازی سیستم، می‌توانند وارد پروسه بهینه‌سازی روند، تایید و یا رد آن شوند.

از آنجایی که اسامی موارد کاربری بیانگر هدف از ایجاد آنها می‌باشد، لذا ارائه موارد کاربری شناسایی شده به صورت لیست، بیان‌کننده محدوده سیستم نرم‌افزاری و اهداف آن از توسعه و اجرا می‌باشد. از این‌رو موارد کاربری اولین رسانه‌ای[۸۴] هستند که ارتباط میان ذی‌نفعان پروژه ایجاد می‌کند.

لیست موارد کاربری توسط کاربران موثر در تولید سیستم، مدیران کارفرما، برنامه‌نویسان خبره و مدیران پروژه بررسی می‌شود. بر اساس نتیجه بررسی ایشان، می‌تواند برآوردی از هزینه و پیچیدگی سیستم نرم‌افزاری حاصل شود.

همچنین ذی‌نفعان می‌توانند با مشاهده لیست موارد کاربری، و برآورد بدست آمده از هزینه وپیچیدگی آنها درباره کارکردهایی که در نسخه‌های اول نرم‌افزار پیاده‌سازی شود، بحث و تبادل نظر کنند.

مدیران پروژه می‌توانند درباره چگونگی سازمان‌دهی اعضای تیم توسعه در قالب تیم‌های مختلف تصمیم‌گیری کنند. همچنین برآورد اولیه از زمان و منابع

[83] Story
[84] Media

مورد نیاز برای انجام پروژه با بهره‌گیری از لیست موارد کاربری نیز قابل دسترس می‌باشد.

در تهیه موارد کاربری، نویسندگان آن در جلساتی که اصطلاحا جوشش فکری[۸۵] نامیده می‌شود درباره مواردی که سبب اختلال در سناریوی اصلی هر مورد کاربری می‌شود، با هم به تبادل نظر می‌پردازند. این موارد لیست می‌شوند و درباره اینکه چگونه سیستم در این موارد بتواند پاسخگوی کنشگر اصلی باشد تصمیم‌گیری می‌شود. مسلما تا زمانیکه که سناریوهای موارد کاربری مستند نگردیده باشد، موارد استثنا در موارد کاربری با چنین جلساتی شناسایی نمی‌شود. چراکه تا سناریویی نوشته نشود نمی‌توان نظر قطعی ذی‌نفعان مختلف پروژه را درباره آن دریافت کرد. مواردی که با این رویکرد در قالب سناریوهای فرعی در موارد کاربری شناسایی می‌شود با رویکردی غیر از این نمی‌توانند با چنین عمقی شناسایی شوند.

لذا چنانچه موارد خطا در سناریوهای موارد کاربری شناسایی نشوند، بسیاری از آنها یا شناسایی نمی‌شوند و یا در زمانی آشکار می‌شوند که برنامه‌نویس در حال نوشتن کد آن مورد کاربری می‌باشد. چنین شناسایی دیرهنگامی اثرات بسیار بدی بر کیفیت سیستم نرم‌افزاری دارد. چراکه معمولا مشاوران خبره در آن زمان در اختیار نمی‌باشند. همچنین از آنجایی که به اواخر پروژه در حال نزدیک‌شدن هستیم، معمولا برنامه‌نویس‌ها به صورت ضمنی این اختیار را پیدا می‌کنند که هرچه را در آن زمان فکر می‌کنند صحیح است را انجام دهند. مسلما این روش نتیجه‌ای متفاوت در کیفیت با بحث‌هایی که در جلسات جوشش فکری می‌شود خواهد داشت.

۶,۲,۸ چگونگی استخراج و تدوین موارد کاربری

[85] Brainstorming

مسلما در مراحل اولیه، که موارد کاربری در حال شناسایی هستند، توصیف دقیق آنها سبب می‌شود تا نتوان در زمان محدود ابتدای پروژه به حوزه‌های مختلفی که ممکن است مرتبط با فعالیت نرم‌افزار تولیدی است پرداخت. بنابراین روشی که در ابتدای تدوین موارد کاربری توصیه می‌شود، استخراج طرحی کلی از هر مورد کاربری می‌باشد. سپس خلاصه‌ای از هر مورد کاربری بیان می‌شود. لذا می‌توان این امکان را در اختیار ذی‌نفعان پروژه قرار داد تا درباره صحت موارد شناسایی شده توسط ما و نیز اولویت آنها اظهار نظر کنند. سپس تیم توسعه می‌تواند در قالب تیم‌های متعدد و به صورت موازی موارد کاربری را تکمیل کنند. این رویکرد سبب افزایش بهره‌وری تیم توسعه می‌شود.

معمولا انسانها در مواجهه اولیه با هر موضوعی می‌خواهند از فاصله دوری آن را مشاهده و بررسی کنند. بنابراین بهترین رویکرد آن است تا در مراحل اولیه شناسایی هر موضوعی بدون آنکه به جزئیات پرداخته شود، با دقت[۸۶] کم به کلیت مساله نگریسته شود. سپس در آینده می‌توان در هر جزء به دقت مناسب رسید. نکته‌ای که در اینجا بیان آن ضروری است تفاوت بین دقت و درستی[۸۷] در بیان حقایق می‌باشد. مثلا فرض کنید بگوییم که عدد π برابر با ۴,۱۴۲۱۵۵۲۳ است. مسلما این مقدار دقت بالایی دارد و با چند رقم اعشار بیان شده است ولی صحیح نمی‌باشد. مسلما چنانچه بگوییم عدد π برابر با ۳ است، هرچند دقت زیادی در آن وجود ندارد ولی آن مقدار صحیح می‌باشد. لذا در مواجهه اولیه با موارد کابری چنانچه بخواهیم آنها را با دقت زیادی بیان کنیم این امکان وجود دارد که موارد بیان شده صحیح نباشد. بنابراین رویکرد کلی آن است تا در ابتدا با دقت کم موارد کاربری با گستردگی فراوان شناسایی شوند و سپس در ادامه پروژه با برنامه‌ریزی

[86] Precision
[87] Accuracy

به دقت آنها افزوده شود. این رویکرد سبب می‌شود تا نیازمندی‌های شناسایی شده در هر مرحله صحیح باشد.

گام‌هایی که در ادامه آورده می‌شود، رویکرد بیان شده در این بخش را تدوین می‌کند.

الف) کنشگران و اهداف آنها

در ابتدا لیستی ازکنشگران و اهدافی از آنها که سیستم نرم‌افزاری برای رسیدن به آنها طراحی می‌شود آورده می‌شود. سپس لیست ایجاد شده برای درستی و کامل‌بودن بازنگری می‌شود. بازنگری این لیست با ذی‌نفعان پروژه صورت می‌پذیرد.

سپس لیست ایجاد شده به کمک ذی‌نفعان اولویت‌بندی می‌شود و برنامه‌ریزی اولیه برای نسخه‌های نرم‌افزار تولیدی انجام می‌شود.

ب) بیان روند اصلی در موارد کاربری

برای هر هدف مشخص شده در مرحله قبل، می‌توان مورد کاربری متناسب با آن ایجاد کرد. برای هر مورد کاربری، سناریوی اصلی و شرایط پیش‌نیاز و پایانی آن تدوین می‌شود. موارد کاربری ایجاد شده توسط دیگر ذی‌نفعان بازبینی شوند.

ج) شرایط عدم موفقیت در سناریوی اصلی

پس از اینکه سناریوی اصلی تکمیل گردید، شرایطی که بواسطه آن سناریوی اصلی نمی‌تواند ادامه یابد در جلسه‌ای با دیگر اعضای تیم شناسایی می‌شوند. این جلسات که به صورت جوشش فکری برگزار می‌شود سبب می‌شود تا شرایط محتمل در شکست سناریوی اصلی مورد کاربری بررسی گردد.

برای هر مورد کاربری ابتدا شرایط شکست سناریوی اصلی لیست گردد. توصیه می‌شود در ابتدا به چگونگی پاسخ به شرایط شناسایی شده پرداخته نشود. چراکه پرداختن به چگونگی پاسخ سبب می‌شود تا روند همفکری اعضای تیم در شناسایی شرایط دیگر شکست مختل گردد.

د) بیان روند برای شرایط عدم موفقیت

در ادامه گام قبل، چگونگی پاسخ سیستم به شرایط عدم موفقیت سناریوی اصلی پرداخته می‌شود. در برخی موارد چگونگی پاسخ به عدم موفقیتی در سناریوی اصلی سبب ایجاد کنشگر جدیدی در سیستم نرم‌افزاری و یا پیدایش هدفی جدید در سیستم نرم‌افزاری می‌گردد.

به صورت معمول پاسخ به چگونگی عدم موفقیت در سناریوی اصلی سبب پیدایش سناریوهای فرعی جدید می‌شود. باید توجه داشت که شرایط عدم موفقیت در سناریوی اصلی می‌تواند با شرایطی دیگر در سناریوهای فرعی نیز وجود داشته باشد. لذا گام‌های ج و د می‌تواند در تعدادی تکرار تکمیل شود.

۶,۳ تهیه نمونه‌های اولیه کارکردی

نمونه‌های اولیه کارکردی[۸۸] را در هر زمان از چرخه حیات پروژه می‌توان تهیه کرد. ولی از آنجایی که هدف از تهیه آنها کاهش ریسک در عدم شناخت صحیح نیازمندی‌های مشتری می‌باشد، لذا عموما در مراحل اولیه پروژه تهیه می‌شود. هر نسخه‌ای از نمونه اولیه که برای ارائه به مشتری و یا برای بحث داخلی تیم‌توسعه تهیه می‌شود، کاربردش تنها در همان مرحله از پروژه است. لذا هرچند در آرشیو پروژه دخیره می‌شود ولی کدهای اصلی پروژه مستقل از آنها ایجاد می‌شود.

[88] Functional proto-type

قبلا بیان شد که تهیه موارد کاربری اثر قابل توجهی در استخراج نیازمندی‌های کاربران دارد. ولی از آنجایی که نمونه‌های اولیه مشابهت با سیستم نرم‌افزار واقعی دارند، لذا مشاهده آنها توسط کاربران می‌توان جنبه‌هایی از نیازمندی را نمایان سازد که با فرآورده دیگری مانند مورد کاربری قابل استخراج نمی‌باشد.

چگونگی تهیه نمونه اولیه از دو بعد قابل بررسی است.

الف) نمونه اولیه افقی[۸۹]

در این رویکرد دیدی جامع از کلیت سیستم نرم‌افزاری برای کاربران آن تهیه می‌شود. لذا در این بعد، بخش‌های مختلف سیستم که کاربر با آنها در تعامل است به صورت کلی نشان داده می‌شود. لذا هرچند تمامی کارکردهای سیستم نشان داده می‌شود، ولی کارکردهای سیستم به صورت جزئی‌تر آورده نمی‌شود.

لذا بهره‌گیری از این رویکرد در تهیه نمونه اول هر سیستم و یا زیرسیستم‌هایی از آن سبب می‌شود تا بتوانیم نیازمندی‌های زیر را از کاربر دریافت کنیم.

۱- از کاربر تاییدی مبنی بر محدوده سیستم نرم‌افزاری بدست آوریم.
۲- از کاربر تاییدی مبنی بر فرمت و ظاهر واسط کاربری[۹۰] بدست آوریم.
۳- از کاربر تاییدی مبنی بر قابلیت‌هایی که در نسخه بعدی نرم‌افزار گنجانده می‌شود، بدست آوریم.
۴- برآورد اولیه‌ای از زمان و قیمت نرم‌افزار بدست آوریم.

[89] Horizontal
[90] User interface

ب) نمونه اولیه عمودی[۹۱]

چانچه بخواهیم دانش پنهان در ذهن کاربر را برای کارکردی خاص از نرم‌افزار بدست آوریم، از این رویکرد استفاده می‌کنیم. بنابراین برخلاف رویکرد افقی، کارکردهایی از سیستم نرم‌افزاری به صورت پایین آورده می‌شود.

۶,۴ تهیه قواعد کسب‌وکار

مسلما سازمان‌ها برای رسیدن به ماموریت و اهدافشان، استراتژی‌هایی مشخص کرده‌اند که بسته به شرایط محیط کسب‌کار قابلیت تغییر دارند. مثلا یکی از اهداف بانک می‌تواند کسب‌سودآوری بیشتر باشد. در این خصوص باتوجه به ریسک بالای سرمایه‌گذاری مستقیم، استراتژی وام‌دهی حداکثری را در خط مشی خود قرار می‌دهد. قواعد کسب‌وکار[۹۲] چگونگی و راهنمای محقق شدن استراتژی را بیان می‌کنند. مثلا اینکه وام‌گیرنده قبلا بدهی معوقه‌ای در هیچ موسسه مالی دیگری نداشته باشد، یکی از قواعد کسب‌وکاری است که بخشی از چگونگی استراتژی وام‌دهی بانک را بیان می‌کند.

قواعد کسب‌وکار در هر سازمان و یا محیط کسب‌وکاری وجود دارد. هرچند که در بسیاری از موارد این قواعد به صورت مکتوب نیستند. اما می‌توان آنها را در چگونگی رفتار مدیریت و واحدهای مختلف سازمانی استخراج کرد. از این رو ممکن است سازمانی بخواهد قواعد کسب‌وکار خود را استخراج کند تا بتواند با کارشناسی بیشتر درباره چگونگی رفتار سازمانش، کنکاش بیشتری در موفقیت خود داشته باشد.

[91] Vertical
[92] Business rule

عموما قواعد کسب‌کار در مراحل اولیه پروژه‌های نرم‌افزاری کشف شده و به صورت مکتوب مستند می‌شود. چنانچه سازمانی بخواهد بواسطه اجرای پروژه نرم‌افزاری فرآیند خود را اصلاح کند و خدمت یا محصولی جدید را اضافه کند، قواعد کسب‌وکار جدیدی در طول پروژه تعریف می‌شود. باید توجه داشت که در مواردی ممکن است قواعد مستند شده در بخش‌های مختلف سازمان با هم سازگار نباشند. لذا شناسایی عدم سازگاری‌ها در بین قواعد کسب‌وکار و سازگار کردن آنها از فعالیت‌های دشوار در مهندسی نرم‌افزار می‌باشد.

شناسایی قواعد کسب‌وکار در پروژه‌های نرم‌افزاری از آنجایی دارای اهمیت هستند که عدم شناخت آنها سبب عدم پذیرش سیستم نرم‌افزاری و شکست پروژه می‌شود. مسلما روش شناسایی قواعد، چگونگی محقق ساختن آنها در سیستم نرم‌افزاری و طراحی آزمون آنها در مراحل مختلف پروژه جایگاه ویژه‌ای در مدیریت نیازمندی‌های نرم‌افزار دارد.

بسته به معماری نرم‌افزار، چگونگی پیاده‌سازی قواعد کسب‌وکار می‌تواند متفاوت باشد. مثلا در معماری سرویس‌گرا[۹۳](SOA) و یاد در سیستم‌های مدیریت فرآیند کسب‌وکار (BPMS[۹۴])، در معماری بخشی با نام موتور قواعد کسب‌وکار[۹۵] (BRE) وجود دارد که وظیفه آن تطبیق قواعد با کارکردهای مختلف سیستم در هر زمان می‌باشد. لذا بدون نیاز به دانش برنامه‌نویسی، توسعه دهندگان این سیستم‌ها قواعد را با استانداردی مشخص در پایگاه داده قواعد کسب‌وکار سیستم وارد می‌کند. لذا هر زمان چنانچه سازمان بخواهد با تغییر در استراتژی خود بخشی از قواعد را تغییر داده و یا قواعد جدیدی اضافه کند، نیازی به تغییرات در

[93] Service Oriented Architecture (SOA)
[94] Business Process Management System (BPMS)
[95] Business Rules Engine (BRE)

کد برنامه نمی‌باشد. بلکه قواعد در پایگاه داده مربوطه بهنگام شده و BRE از زمانی به بعد ،که مشخص گردیده است، قواعد جدید را اعمال می‌کند.

در معماری شیء گرایی[۹۶] (OO) رویکردی که در SOA یا BPMS برای قواعد کسب‌وکار است پیش‌بینی نشده است. لذا SOA و BPMS برای نرم‌افزارهایی که در آنها تغییر سریع فرآیند یکی از الزامات آن می‌باشد نسبب به OO برتری دارد. مروری بر فلسفه وجودی BPMS در فصل ۹ آورده شده است. مسلما OO نیز برای بسیاری از کاربردها دارای برتری می‌باشد که سبب گردیده است تا همچنان جایگاه خوبی در میان معماری‌های نرم‌افزاری داشته باشد. هرچند اخیرا چارچوب‌هایی مانند OSGI توسط IBM پیشنهاد شده است که در آن OO می‌تواند از مفروضات SOA استفاده کند.

در ادامه این بخش، دسته‌بندی از قواعد کسب‌وکار در سیستم‌های نرم‌افزاری آورده می‌شود. چگونگی شناسایی آنها و مثال‌هایی برای روشن‌تر شدن بحث نیز در هر موضوع آورده می‌شود. مستند قواعد کسب‌وکار، که ایجاد آن در مراحل اولیه پروژه و تکمیل آن تا آخرین مراحل چرخه حیات نرم‌افزار ادامه می‌یابد، در همین خصوص تهیه می‌شود.

در ادامه، تقسیم‌بندی اولیه‌ای از قواعد کسب‌وکار در هر سیستم نرم‌افزاری آورده شده است. تجمیع کلیه قواعد در مستندی با همین نام سبب می‌شود تا از افزونگی این قواعد در دیگر مستندات جلوگیری شود. مسلما در مستندات دیگر مانند موارد کاربری و موارد آزمون دادن آدرس به قواعد مرتبط در این مستند ضروری می‌باشد.

۶,۴,۱ پیغام‌های سیستم

[96] Object Oriened (OO)

پیغام‌هایی که سیستم نرم‌افزاری به کاربر نشان می‌دهد را می‌توان اولین قواعدی در نظر گرفت که مستندسازی آنها سبب می‌شود تا از افزونگی آنها در دیگر مستندات جلوگیری شود.

برای ارجاع‌دهی به قواعد کسب‌وکار در دیگر مستندات، در مستند قواعد کسب‌وکار برای هر نوع از قواعد می‌توان سیستمی برای کددهی تعریف کرد. لذا پیغام‌های سیستم را مثلا با فرمت کد Msgdd (d=0:9) مشخص می‌کنیم. پیغام‌های سیستم می‌توانند به صورت متمرکز در جدول زیر ذخیره شوند.

جدول ۱) پیغام‌های سیستم

ردیف	کد	پیغام
۱	Msg01	[آقا/خانم] [نام و نام خانوادگی مشتری]، اعتبار شما تا [تعدا روزهای باقی‌مانده اعتبار] روز دیگر پایان می‌یابد.
۲	...	

۶,۴,۲ الگوهای داده‌ای

چگونگی ورود اطلاعات توسط فرم‌های مختلف برنامه و یا دیگر سرویس‌هایی که در دسترس دیگران می‌باشد بسیار اهمیت دارد. چنانچه این اطلاعات براساس قواعد مشخصی دریافت نشود، سبب می‌شود پردازش‌های بعدی آنها با مشکل مواجه شود. برای قواعدی که در خصوص الگوهای ورود اطلاعات است می‌توان روش کددهی به صورت Ddd (d=0:9) تعریف کرد.

الگوهای داده‌ای، به صورت متمرکز در جدول زیر ذخیره می شوند. الگوهای آورده شده در جدول زیر بایستی به گونه‌ای آورده شوند که آزمون موردهای کاربری امکان پذیر گردد. بنابراین باید نهایت دقت در تعریف الگوهای داده‌ای صورت پذیرد.

جدول ۲) نمونه‌ای از قواعد کسب‌وکار برای الگوهای داده‌ای

ردیف	کد	الگو
۱	D01	۱- تنها حروف فارسی قابل قبول می باشد. بنابراین از حروف انگلیسی، اعداد و سایر حروف نبایستی استفاده شود. ۲- کلمات یک حرفی در بین کلمات نبایستی باشد. مثلا "محمدی ع اسلامی" قابل قبول نمی باشد.
۲	D02	۱- حداکثر ۳۰ حرف می باشد ۲- حداقل ۳ حرف باشد. ۳- D01 نمونه‌های مجاز: (توجه شود که نمونه‌هایی که در مرز قرار دارند از اهمیت ویژه‌ای برخوردارند) ▪ علی ▪ محمد ▪ علی محمد حسن امیر رضا (۳۰ حرف) نمونه‌هاهای غیر مجاز: (توجه شود که نمونه‌هایی که در مرز قرار دارند از اهمیت ویژه‌ای برخوردارند) ▪ ع ▪ یس ▪ [space]+عل ▪ علی محمد حسن امیر رضا ... (۳۱ حرف) ▪ ...
۳	D03	۱- حداکثر ۲۰ حرف می باشد ۲- D01
۴	D04	۱- حداقل ۶ حرف می باشد. ۲- حداکثر، ۱۲ حرف می باشد. ۳- D01 ۴- در نمایش، حروف به صورت * نشان داده شوند.
۵	...	

۶,۴,۳ محدودیت در اطلاعات جدول‌ها

بخشی از طراحی هر سیستم نرم‌افزاری به پایگاه داده آن مرتبط می‌باشد. از آنجایی که بخش عمده‌ای از پایگاه‌های داده در پروژه‌های نرم‌افزاری رابطه‌ای می‌باشد، افزودن بخشی از قواعد کسب‌وکار به صورت قواعد محدودیتی[۹۷] در جدول‌ها گریزناپذیر می‌باشد. این قواعد را می‌توان با کد T*dd* (*d*=0..9) مشخص نمود.

محدویت اطلاعات جدول‌ها، به صورت متمرکز در جدول زیر ذخیره می شوند. از همین محدودیت‌ها برای تعریف محدودیت در سطح پایگاه داده استفاده می‌شود.

جدول ۳ نمونه‌ای از قواعد کسب‌وکار برای محدودیت در اطلاعات جدول‌ها

ردیف	کد	جدول	شرح محدودیت
۱	T01	Student	تاریخ تولد نبایستی بزرگتر از تاریخ جاری سیستم منهای ۱۰ سال باشد.
...			

۶,۴,۴ قواعد قیدی

بخش عمده‌ای از قواعد کسب‌وکار در سیستم‌های نرم‌افزاری در این حوزه قرار می‌گیرند. این قواعد را می‌توان با کد R*dd* (*d*=0..9) مشخص نمود. دسته‌بندی این قواعد در ادامه آمده است.

الف) بخشی از این قواعد مربوط به چگونگی جریان کار در موارد کاربری است.

[97] Constraints

قاعده‌های زیر را در این حوزه درنظر بگیرید:

- زمانیکه سفارشی لغو می شود، چنانچه سفارش هنوز فرستاده نشده است، آن را لغو کن.
- هنگام بستن حساب، سود در صورتی تعلق می گیرد که یک ماه از افتتاح آن سپری شده باشد.

ب) بخشی دیگر از قواعد، شرایطی را مشخص می‌کنند که می‌بایست قبل و بعد از عملی برای درستی آن عمل کنترل شوند.

قاعده‌های زیر را در این حوزه درنظر بگیرید.

- سفارش را در صورتی برای مشتری بفرستید که مشتری دارای آدرس باشد.
- سفارش مشتریان حداکثر تا ۲۴ ساعت پس از پرداخت وجه بایستی فرستاده شود.
- برداشت از حساب مسدود، مجاز نیست.

ج) این قواعد برای موجودیت‌های شناسایی شده در سیستم تعریف می‌شود. موجودیت‌های سیستم مسلما در ادامه به جدول‌هایی در سطح پایگاه داده تبدیل می‌شوند. چنانچه تا این مرحله از پروژه جدول/جدول‌های مربوطه شناسایی شده باشند، به آوردن قاعده در سطح پایگاه داده اکتفا می‌کنیم تا از بروز افزونگی در قواعد پیشگیری شود.

قاعده‌های زیر را در این حوزه درنظر بگیرید.

- هر سفارش تنها به یک محصول اشاره دارد.
- هر سفارش بایستی دارای حداقل یک کالا باشد.

- هر حساب بانکی کوتاه مدت تنها به یک مشتری تعلق دارد.

۶,۴,۵ قواعد استنتاجی

در این قواعد چگونگی استنتاج یک حقیقت از سایر حقیقت‌ها بیان می‌شود. در ادامه مثال‌هایی از این قواعد آمده است. کدگذاری این قواعد را نیز می‌توان در ادامه قواعد قیدی درنظر گرفت.

قاعده‌های زیر را در این حوزه درنظر بگیرید.

- مشتری‌ای خوب است که در ماه بیش از صدهزار تومان خرید می‌کند.
- حساب غیرفعال به حسابی گفته می‌شود که به مدت شش ماه، برداشت از و یا واریزی به آن انجام نشده باشد.
- قیمت خالص محصول به صورت " قیمت محصول + (۱+درصدمالیات)" محاسب می شود.
- سود حساب سپرده کوتاه مدت در هر روز به شکل "سود روزانه=(حداقل موجودی حساب در روز*سود سالانه) تقسیم بر ۳۶۵" محاسبه می شود.

۶,۵ تهیه نمودار فعالیت

در UML[۹۸] برای آنکه بتوان گردش کارهای آورده شده در موارد کاربری را با سادگی بیشری نمایش داد از نمودار فعالیت[۹۹] استفاده می‌شود. مدل‌سازی یکی از روش‌هایی است که در مهندسی نرم‌افزار برای نمایش بهتر و کاهش شکاف

[98] Unified Modelling Language (UML)
[99] Activity diagram

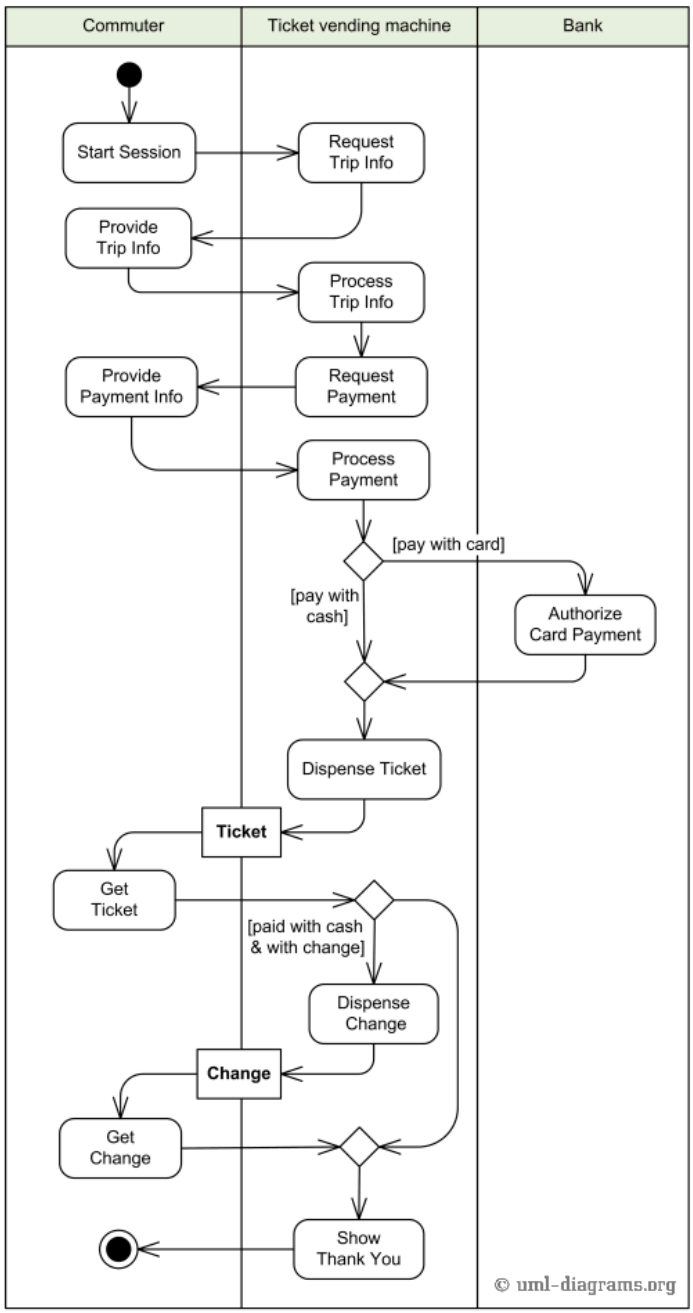
مفهومی[۱۰۰] میان ذی‌نفعان پروژه استفاده می‌شود. ابزارهای متعددی برای مدل‌سازی در مهندسی نرم‌افزار ارائه شده است که البته در هر پروژه باتوجه به توافقی که بین ذی‌نفعان پروژه می‌شود، مدل‌سازی در یکی از این ابزارها صورت می‌پذیرد.

یکی از موارد استفاده از نمودار فعالیت، دیداری‌سازی[۱۰۱] گردش فعالیت‌ها در مورد کاربری می‌باشد. چنانچه بخواهیم از نمودار فعالیت در مورد کاربری استفاده کنیم، برای سناریوی اصلی یک نمودار فعالیت و برای هر سناریوی فرعی نیز یک نمودار فعالیت آورده می‌شود. در نمودار فعالیت المان‌هایی پایه‌ای وجود دارد که با ترکیب آنها می‌توان نمایشی دیداری از گردش کار ایجاد کرد.

در شکل ۷ نمونه‌ای از نمودار فعالیت برای نمایش روند اصلی در مورد کاربری خرید بلیط آورده شده است.

در این مورد کاربری، روند اصلی با اجرای مورد کابری توسط مسافر (کنشگر اصلی) آغاز می‌شود. سپس سیستم اطلاعاتی درباره سفر از مسافر دریافت می‌کند. این اطلاعات می‌تواند شامل تعداد، نوع بلیط (سفر ماهانه، یک‌طرفه یا رفت‌وبرگشت) شماره مسیر، مقصد و غیره باشد. براساس اطلاعات دریافت شده، هزینه را محاسبه کرده و از کاربر می‌خواهد تا نوع پرداخت را نیز مشخص کند. نوع پرداخت می‌تواند شامل پول نقد، کارت اعتباری[۱۰۲] یا کارت بانکی[۱۰۳] باشد. اگر پرداخت با کارت انتخاب شده باشد، کنشگر دیگری مانند بانک نیز در این مورد کاربری مطرح می‌شود که در شناسایی کارت و کسر اعتبار از آن مشارکت خواهد داشت.

[100] Semantic gap
[101] Visualization
[102] Creadit card
[103] Debit card



شکل ۷) نمونه‌ای از نمودار فعالیت که برای روند اصلی مورد کاربری تهیه شده است.

۶,۶ ایجاد موارد و دنباله‌های آزمون

در مهندسی نرم‌افزار، مورد آزمون[۱۰۴] مجموعه‌ای از شرایطی است که بواسطه آنها آزمونگر باید بتواند تشخیص بدهد که آیا سیستم نرم‌افزاری یا یکی از ویژگی‌های آن همانگونه که از ابتدا برای آن برنامه‌ریزی شده است، عمل می‌کند.

بنابراین آزمون‌گر تنها بر اساس شرایط آورده شده در مورد آزمون عمل می‌کند. لذا چناچه وضعیت جاری سیستم تحت آزمون نتواند شرایط آورده شده در مورد آزمونی را پاسخ دهد، آن نسخه نرم‌افزار برای آن مورد آزمون رد[۱۰۵] می‌شود.

در شناخت نیازمندی‌های نرم‌افزار، پس از آنکه موارد کاربری توصیف شدند می‌بایست موارد آزمون آنها نیز تدوین گردند. معمولا به ازای هر مورد کاربری چندین مورد آزمون طراحی می‌شود. در طراحی و تدوین موارد آزمون باید حتما به این نکته تاکید داشت که اجرای آن برای آزمونگر بدون هیچ ابهامی همراه باشد. معمولا زمانی که نیاز باشد موارد آزمون به دفعات بسیار اجرا شوند، برای این منظور از ابزارهای آزمون استفاده می‌شود.

در ادامه سعی شده است با چهار مثالی که در جدول‌های مرتبط آورده می‌شود، انتقال مفاهیم به شیوه‌ای موثر صورت پذیرد. ابتدا در جدول ۴، برای مورد کاربری ساده‌ای مانند ورود به سیستم، مورد آزمونی طراحی شده است. طراحی مورد آزمون باید به گونه‌ای باشد تا تمامی حالات معمول و غیرمعمول در سناریوی اصلی و سناریوهای فرعی پوشش داده شود. مسلما گنجاندن تمامی جنبه‌ها در یک مورد آزمون برای موردهای کاربری پیچیده‌تر امکان پذیر

[104] Test case
[105] Rejected

نمی‌باشد. لذا بایست برای آزمون هر جنبه از مورد کاربری، مورد آزمون خاص آن را طراحی کرد. در طراحی مورد آزمون باید همواره به این نکته توجه داشت که اجرای آن بایست بدون هرگونه ابهامی توسط آزمون‌گر بدون دانش خاصی انجام پذیرد. هرچند فرض موجود بودن مستنداتی مانند موردهای کاربری و قواعد کسب‌وکار و ارجاع آن در مورد آزمون می‌تواند سبب کاهش افزونگی در مستندات و اثربخش‌بودن بیشتر تهیه موارد آزمون گردد.

در جدول ۵، مورد آزمون تهیه شده در جدول ۴ در تاریخ مشخصی توسط آزمونگری مشخص اجرا می‌شود و نتایج مشاهده اجرای آزمون ثبت می‌شود. لذا می‌توان چنین روندی را متصور شد که قبل از هر نصب، نسخه‌ای از موردهای آزمون مرتبط با آن به صورت کاغذی و یا الکترونیکی در اختیار آزمون‌گر قرار می‌گیرد و نتایج مشاهدات از اجرای آزمون در آنها ثبت می‌شود.

بسته به میزان جاری‌سازی اتوماسیون در پروژه، موردهای آزمونی که باید به تکرار زیاد در پروژه اجرا گردند، به ابزار آزمون داده می‌شود. این رویکرد سبب می‌گردد تا اجراهای آزمون در ادامه، در هر زمانیکه نیاز باشد، با سرعت بالا انجام شود.

نمونه دیگری از مورد آزمون طراحی شده برای مورد کاربری ثبت نام در جدول ۶ آورده شده است. توجه داشته باشید که چنانچه مورد کاربری و قواعد کسب‌وکار با دقت مناسب تهیه شده باشد، بیان جنبه‌های مختلف آزمون در مورد آزمون با کیفیت بهتری انجام می‌شود.

مسلما آزمون کارکردی در سیستم‌های نرم‌افزار فقط معطوف به آزمون جداگانه موردهای کاربری نمی‌شود. دربسیار موارد اجرای متوالی چند مورد کاربری سبب تکمیل شدن فرآیندی در سیستم می‌گردد. لذا آزمون این فرآیندها

نیز در قالب دنباله‌های آزمون[۱۰۶] اهمیت دارد. در جدول ۷، نمونه‌ای از دنباله آزمون آورده شده است.

نکته دیگری که برای طراحی موارد آزمون می‌توان مطرح کرد، استفاده موثر از قواعد کسب‌وکار است. به صورت توصیه‌ای کاربری می‌توان گفت که برای هر قاعده کسب‌وکار، بسته به پیچیدگی آن، می‌توان یک یا چند مورد آزمون طراحی کرد.

همچنین باید تاکید کنم که طراحی مورد آزمون روش خاصی ندارد و عملا بسیار وابسته به چگونگی نگرش طراح آزمون به سیستم نرم‌افزاری است. هرچند که طراح آزمون باید مراقب تمامی جنبه‌های سیستم برای شناسایی آزمون‌های آنها باشد ولی مسلما تاکید بهره‌برداران سیستم بر حیاتی بودن جنبه‌هایی مشخص از سیستم بر عمق آزمون آن موثر خواهد بود.

[106] Test suit

جدول ۴) نمونه‌ای از مورد آزمون طراحی شده برای روند اصلی در مورد کاربری ورود به سیستم (Login)

 Test Case مورد آزمون: ورود به سیستم (Login)	
تاریخ اجرای آزمون: ...../...../.....	شماره ترتیب اجرای آزمون: .....
مورد های آزمون مرتبط: UserRegistration	
اولویت: زیاد: ■ متوسط: □ کم: □ (مطابق با دستورالعمل تعیین اولویت موارد آزمون. از دیدگاه کاربر ۳ و از دیدگاه طراح ۲)	
نوع و محدوده داده های مورد استفاده برای آزمون: نوع داده‌ها در مورد کاربری مربوطه آمده است و محدوده آنها جهت آزمون در مستند قواعد کسب و کار آمده است.	
مراحل انجام آزمون به ترتیب و تفکیک انجام فعالیت ها توسط آزمون گر: مطابق با گردش اصلی رخداد ورود به سیستم	
نتایج مورد انتظار از اجرای آزمون: (در راستای نقض هرکدام از قواعد کاربری در مورد کاربری مربوطه حداقل یک بند اضافه شود) توجه: ۱- چنانچه نام کاربری مطابق با D03 نباشد پیغام ۳ را نمایش می دهد. ۲- چنانچه نام کاربری قبلا توسط مورد کاربری "ثبت نام" وارد نشده باشد پیغام ۱ را نمایش می دهد. ۳- چنانچه کلمه عبور مطابق با D04 تعریف نشده باشد پیغام ۴ را نمایش می دهد. ۴- چنانچه کلمه عبور صحیح نباشد پیغام ۲ را نمایش می‌دهد. ۵- چنانچه نام کاربر و کلمه عبور صحیح باشند کاربر به صفحه اصلی هدایت می شود.	
نتایج بدست آمده از اجرای آزمون با در نظر گرفتن بند راهکار در آزمون قبلی (برای هرکدام از موارد بند قبل ۴ حالت مختلف وارد نمایید) ۱- ..... ۳-..... ۵-..... ۲- ۴-	
بیان دقیق علت اختلاف بین نتیجه مورد انتظار و نتیجه بدست آمده از اجرای آزمون:...	
راهکار برای آزمون دیگر:...	
آزمونگر: ..... **امضا:**	نتیجه اجرای آزمون: قبول □ غیر قابل قبول □

جدول ۵) نمونه‌ای از نتیجه اجرای آزمون از مورد آزمون طراحی شده در جدول ۴ برای مورد کاربری ورود به سیستم (Login)

 Test Case	مورد آزمون: ورود به سیستم (Login)
تاریخ اجرای آزمون: ۹۵/۸/۹	شماره ترتیب اجرای آزمون: ۱
مورد های آزمون مرتبط: UserRegistration	
اولویت: زیاد: ■ متوسط: □ کم: □ (مطابق با دستورالعمل تعیین اولویت موارد آزمون. از دیدگاه کاربر ۳ و از دیدگاه طراح ۲)	
نوع و محدوده داده های مورد استفاده برای آزمون: نوع داده‌ها در مورد کاربری مربوطه آمده است و محدوده آنها جهت آزمون در مستند قواعد کسب و کار آمده است.	
مراحل انجام آزمون به ترتیب و تفکیک انجام فعالیت ها توسط آزمون گر: مطابق با گردش اصلی رخداد ورود به سیستم	
نتایج مورد انتظار از اجرای آزمون: (در راستای نقض هرکدام از قواعد کاربری در مورد کاربری مربوطه حداقل یک بند اضافه شود) توجه: ۱- چنانچه نام کاربری مطابق با D03 نباشد پیغام ۳ را نمایش می دهد. ۲- چنانچه نام کاربری قبلا توسط مورد کاربری "ثبت نام" وارد نشده باشد پیغام ۱ را نمایش می دهد. ۳- چنانچه کلمه عبور مطابق با D04 تعریف نشده باشد پیغام ۴ را نمایش می دهد. ۴- چنانچه کلمه عبور صحیح نباشد پیغام ۲ را نمایش می‌دهد. ۵- چنانچه نام کاربر و کلمه عبور صحیح باشند کاربر به صفحه اصلی هدایت می شود.	
نتایج بدست آمده از اجرای آزمون با در نظر گرفتن بند راهکار در آزمون قبلی (برای هرکدام از موارد بند قبل ۴ حالت مختلف وارد نمایید) ۱- تایید ۳-تایید ۵-رد ۲- تایید ۴- تایید	
بیان دقیق علت اختلاف بین نتیجه مورد انتظار و نتیجه بدست آمده از اجرای آزمون:به نظر می رسد code page صفحه کلیدی که نام کاربری ثبت نام شده است با صفحه کلیدی کامپیوتری که آزمون روی آن انجام پذیرفته است متفاوت می باشد.	
راهکار برای آزمون دیگر:هنگام آزمون بعدی می بایست ثبت نام برای حروفی مانند ی، ک، ژ، پ با کامپیوتری که صفحه کلید آن متفاوت با صفحه کلید کامپیوتر آزمون است انجام پذیرد.	
آزمونگر: علی محمدی **امضا:**	نتیجه اجرای آزمون: قبول □ غیر قابل قبول ■

جدول ۶) نمونه‌ای دیگر از مورد آزمون طراحی شده برای مورد کاربری ثبت نام کاربری (User Registration)

 Test Case	مورد آزمون: ثبت نام کاربری (UserRegistration)
تاریخ اجرای آزمون: ...../...../.....	شماره ترتیب اجرای آزمون: .....
مورد های آزمون مرتبط: Login	
اولویت: زیاد: □ متوسط: □ کم: ■ (مطابق با دستورالعمل تعیین اولویت موارد آزمون. از دیدگاه کاربر ۱ و از دیدگاه طراح ۱)	
نوع و محدوده داده های مورد استفاده برای آزمون: نوع داده‌ها در مورد کاربری مربوطه آمده است و محدوده آنها جهت آزمون در مستند قواعد کسب و کار آمده است.	
مراحل انجام آزمون به ترتیب و تفکیک انجام فعالیت ها توسط آزمون گر: مطابق با گردش اصلی رخداد ورود به سیستم	
نتایج مورد انتظار از اجرای آزمون: ۱- چنانچه نام کاربری مطابق با UserNamePattern نباشد پیغام ۳ را نمایش می دهد. ۲- چنانچه نام کاربری قبلا برای شماره پرسنلی دیگری ثبت شده باشد پیغام ۵ را نمایش می دهد. ۳- چنانچه برای شماره پرسنلی وارد شده قبلا نام کاربری تعریف شده باشد پیغام ۶ را نمایش می دهد. ۴- چنانچه کلمه عبور مطابق با PasswordPattern تعریف نشده باشد پیغام ۴ را نمایش می دهد. ۵- چنانچه کلمه عبور مجدد مطابق با PasswordPattern تعریف نشده باشد پیغام ۴ را نمایش می دهد. ۶- چنانچه شماره پرسنلی EmployeeNumberPattern تعریف نشده باشد پیغام ۷ را نمایش می دهد. ۷- چنانچه کلمه عبور قبلا در سیستم وارد شده باشد پیغام ۸ را نمایش می دهد. ۸- چنانچه دو کلمه عبور وارد شده یکسان نباشند پیغام ۹ را نمایش می دهد. ۹- چنانچه مراحل چهارگانه تا قبل از زدن دکمه ثبت نام به درستی انجام پذیرد کاربر به صفحه اصلی هدایت می شود.	
نتایج بدست آمده از اجرای آزمون با در نظر گرفتن بند راهکار در آزمون قبلی (برای هرکدام از موارد بند قبل ۴ حالت مختلف وارد نمایید) ۱- ..... ۲-..... ۳-..... ۴-..... ۵-..... ۶-..... ۷-..... ۸-..... ۹-.....	
بیان دقیق علت اختلاف بین نتیجه مورد انتظار و نتیجه بدست آمده از اجرای آزمون:.....	
راهکار برای آزمون دیگر:...	
آزمونگر:..... **امضا:**	نتیجه اجرای آزمون: قبول □ غیر قابل قبول □

جدول ۷) نمونه‌ای ازدنباله آزمون طراحی شده برای سامانه تصادفات راهور

 دنباله آزمون دنباله ثبت تصادف (AccidentRegistrationSuite)	
تاریخ اجرای آزمون: ...../...../.....	شماره ترتیب اجرای آزمون: .....
اولویت: زیاد: ■ متوسط: □ کم: □ (مطابق با دستورالعمل تعیین اولویت موارد آزمون. از دیدگاه کاربر ۳ و از دیدگاه طراح ۳)	
مراحل انجام آزمون به ترتیب و تفکیک انجام فعالیت ها توسط آزمون گر: ۱- ثبت کارشناس (ExpertRegistration) ۲- تعریف انبار کروکی (InventoryRegistration) ۳- تعریف کروکی (PaperRegistration) ۴- انتقال کروکی به انبار (PaperReceipt) ۵- تخصیص کروکی به کارشناس (PaperInventryReceipt) ۶- ثبت تصادف (AccidentRegistration)	
نوع و محدوده داده های مورد استفاده برای آزمون: ۱- با مورد آزمون ثبت کارشناس، کارشناس با کد x ثبت می شود. ۲- با مورد آزمون تعریف انبار کروکی، انبار با کد i ثبت می شود. ۳- با مورد آزمون تعریف کروکی، کروکی از کد $k1$ تا $k2$ ثبت می شود. ۴- با مورد آزمون انتقال کروکی به انبار، کروکی ها با کد $k1 \geq k3$ تا $k2 \geq k4$ به انبار i منتقل می شود. ۵- با مورد آزمون تخصیص کروکی به کارشناس، کروکی ها با کد $k3 \geq k5$ تا $k4 \geq k6$ از انبار i به کارشناس x منتقل می شود. ۶- با مورد آزمون ثبت تصادف ، ثبت تصادف توسط کارشناس x با برگه کروکی $k5 \geq k \geq k6$ انجام می شود.	
نتایج مورد انتظار از اجرای آزمون: ۱- تمامی موارد آزمون به دوستی اجرا گردند و در نهایت ثبت تصادف به دستی انجام پذیرد.	
نتایج بدست آمده از اجرای آزمون با در نظر گرفتن بند راهکار در آزمون قبلی:	
بیان دقیق علت اختلاف بین نتیجه مورد انتظار و نتیجه بدست آمده از اجرای آزمون:...	
راهکار برای آزمون دیگر:	
آزمونگر: ..... **امضا:**	نتیجه اجرای آزمون: قبول □ غیر قابل قبول □

۷ توصیف نیازمندی‌های غیرکارکردی

در مدیریت نیازمندی‌ها، نیازمندی‌های غیرکارکردی[۱۰۷]، نیازمندی‌هایی از سیستم نرم‌افزاری هستند که معیارهایی برای ارزیابی کارکرد سیستم به‌جای ارزیابی رفتار مشخصی را بیان می‌کند. مثلا در تهیه موارد کاربری، موارد آزمون و قواعد کسب‌وکار، معیارهایی را بیان کردیم که رفتار مشخصی را دقیقا بیان می‌کرد و در ادامه می‌توانستیم آن رفتار را ارزیابی کنیم.

نیازمندی‌های غیرکارکردی نیز همانند کارکردی باید در ابتدای پروژه شناسایی شده تا در ادامه بتوان از آنها برای ارزیابی کلی سیستم نرم‌افزاری استفاده کرد. هرچند نیازمندی‌های مرتبط با امنیت نیز می‌تواند در حوزه نیازمندی‌های غیرکارکردی قرار گیرد، ولی در این فصل توجه ما بر نیازمندی‌هایی است که در حوزه کارایی سیستم قرار دارند.

ابتدا مروری اجمالی بر آزمون‌های مرتبط با کارایی خواهیم داشت تا در ادامه بهتر بتوانیم درباره شناسایی نیازمندی‌ها در این حوزه صحبت کنیم

۷.۱ آزمون‌های مرتبط با کارایی

الف) آزمون بار[۱۰۸]

از این آزمون جهت بررسی رفتار سیستم نرم‌افزاری تحت شرایط اوج بار از منظرهای مختلف مانند تعداد کاربران یا حجم داده استفاده می شود. آزمون بار بدین منظور مورد استفاده قرار می گیرد تا کنترل نماید که آیا نرم‌افزار می‌تواند اهداف کارایی را در زمان اوج بار سیستم نیز برآورده نماید. لذا بایست از ابتدا در

[107] Non-faunctional requirements
[108] Load test

شناخت نیازمندی‌های نرم‌افزار، اهداف کارایی نیز مشخص گرند. برای این آزمون نام دیگری مانند آزمون ظرفیت[۱۰۹] نیز اطلاق شده است.

براساس نتایج بدست آمده از آزمون بار می‌توان زمان پاسخ[۱۱۰]، نرخ توان عملیاتی[۱۱۱] و سطح بهره وری از منابع (مانند حافظه، پردازنده، پهنای باند شبکه) را اندازه گیری نمود. همچنین به کمک این آزمون می توان نقاط شکست در نرم افزار را مشخص نمود و به این نتیجه برسید که نقاط شکست در چه مرحله‌ای از اوج بار قرار دارند.

آزمون بار در ارتباطی نزدیک با برنامه‌ریزی بار می باشد. در برنامه‌ریزی بار مشخص می‌گردد که رشد آتی سیستم نرم‌افزاری، اعم از افزایش کاربران یا حجم داده‌ها به چه صورت می‌باشد. مثلا جهت حل کمبودهای ناشی از بارهای آینده سیستم نرم‌افزاری نیاز است بدانیم که چه تعداد منابع اضافه نظیر ظرفیت پردازش، حافظه، فضای دیسک و یا پهنای باند شبکه مورد نیاز می باشد. آزمون بار این امکان را می دهد تا استراتژی برای افزایش یا کاهش مقیاس منابع محیط فراهم گردد.

ب) آزمون پایداری[۱۱۲]

این آزمون نوعی از آزمون کارایی است که تمام توجه آن بر روی تعیین و یا تایید مشخصه‌های کارایی سیستم نرم‌افزاری در مدتی طولانی از بار کاری می

[109] Capacity test
[110] Response time
[111] throughput rate
[112] Endurance test

باشد. آزمون پایداری جهت تعیین میانگین زمان بین شکست[۱۱۳] (MTBF)، میانگین زمان تا شکست[۱۱۴] (MTTF) و پارامترهایی از این قبیل به کار می رود.

ج) آزمون تنش[۱۱۵]

هدف از این آزمون تعیین و یا تایید رفتار سیستم نرم‌افزاری هنگامی است که با سرعت در شرایط غیر عادی و یا اوج بار قرار می‌گیرد. تفاوت آزمون تنش با آزمون بار در لحظه‌ای بودن اوج بار در آزمون تنش می‌باشد. این در حالیست که در آزمون بار فرض بر آن است که شرایط بار مداوم است. ایراداتی از سیستم که در آزمون تنش می‌تواند شناسایی شود می‌تواند شامل مباحث هزمانی[۱۱۶]، شرایط رقابت[۱۱۷] و کمبود حافظه باشد. آزمون تنش این امکان را فراهم می سازد تا نقاط ضعف سیستم نرم افزاری مشخص گردد.

د) آزمون قابلیت استفاده[۱۱۸]

بسته به محیطی که سیستم نرم‌افزاری در آن استفاده می‌شود، شرایط استفاده از نرم‌افزار هم متفاوت می‌باشد. فرض کنید در پذیرش آزمایشگاهی دولتی، که مراجعه کنندگان متعددی دارد، مورد کاربری ثبت آزمایش به گونه‌ای نباشد که اپراتور استفاده‌کننده آن بتواند پاسخگوی مراجعات باشد. در این حالت هرچند طراحی نرم‌افزار از جنبه‌های مختلف برتر از سیستم قدیمی باشد، ولی نمی‌تواند جایگزین سیستم قدیمی شود. در آزمون قابلیت سیستم این نیازمندی

[113] Mean Time Between Failure (MTBF)
[114] Mean Time To Failure (MTTF)
[115] Stress test
[116] synchronization
[117] Race conditions
[118] Usability test

که آیا روش تعامل کاربر با نرم‌افزار با شرایط محیطی همخوانی دارد یا خیر بررسی می‌شود.

بسته به انواع آزمون‌های کارایی که انجام یا عدم انجام آنها وابسته به شرایط پروژه است، جنبه‌هایی از کارایی را می‌توان برای انجام آزمون‌های کارایی پیشنهاد داد. مسلما این جنبه‌ها پیشنهادهایی اولیه توسط مولف است که بسته به نظر ذی‌نفعان پروژه و شرایط پروژه تغییر می‌کند.

در جدول‌های ۸ و ۹ جنبه‌های کارآیی از چهار منظر تعداد کاربران، حجم داده، تعداد تراکنش یا پردازش در حال اجرا و درنهایت مدت زمان اجرای بدون وقفه سیستم آمده است. در این دو جدول همچنین برای هر جنبه از کارآیی شرایط پیشنهادی برای بررسی آنها به‌گونه‌ای آمده است که پاسخگوی آزمون‌های مرتبط با کارایی باشند.

جدول ۸) جنبه‌های تعداد کاربران و حجم داده برای انجام آزمون کارایی

شناسه	شرح	شرایط بررسی
Users	تعداد کاربر همزمان	جهت آزمون با توجه تعداد کاربر همزمان دو حالت عمده می بایست مورد بررسی قرار گیرد: ۱- تعداد کاربران وارد شده به سیستم نرم افزاری ۲- تعداد کاربران در حال استفاده از هر یک از نیازهای کارکردی جهت بررسی تعداد کاربر همزمان در مورد هر یک از دو مورد بایستی شرایط زیر برای آنها در نظر گرفته شود: ۱- تعداد معمول کاربران (Normal Users) ۲- حداکثر تعداد کاربران (Max Users) ۳- حداکثر تعداد کاربران در شرایط کاری سیستم با بالاترین مقیاس (Scale Up Max Users) ۴- تعداد معمول افزایش کاربران در ثانیه (Users Rate) ۵- تعداد افزایش کاربران در شرایط فشار در ثانیه(Stress Users Rate)
Datav	حجم داده مورد پردازش	جهت آزمون با توجه به حجم داده مورد پردازش دو حالت عمده می بایست مورد بررسی قرار گیرد ۱- حجم داده‌های سیستم نرم‌افزاری ۲- حجم داده مورد استفاده توسط هر یک از نیازهای کارکردی واحد سنجش جهت حجم داده تعداد رکود در جدول‌ها می‌باشد. جهت بررسی حجم داده مورد پردازش در مورد هر یک از دو مورد بایستی شرایطی مشابه زیر برای آنها در نظر گرفته شود: ۱- تعداد معمول حجم داده (Normal Datav) ۲- حداکثر حجم داده (Max Datav) ۳- حداکثر حجم داده در شرایط کاری سیستم با بالاترین مقیاس (Scale Up Max Datav) ۴- تعداد معمول افزایش حجم داده در ثانیه (Datav Rate) ۵- تعداد افزایش حجم داده در شرایط فشار در ثانیه(Stress Datav Rate)

جدول ۹) شرایط پیشنهادی برای انجام آزمون کارایی از منظر تعداد تراکنش‌ها و زمان اجرای بدون وقفه سیستم

شناسه	شرح	نکات مهم
Trans	تعداد تراکنش های در حال انجام	جهت آزمون با توجه به تعداد تراکنش مورد پردازش دو حالت عمده می بایست مورد بررسی قرار گیرد ۱- تعداد تراکنش های سیستم نرم افزاری ۲- تعداد تراکنش جهت اجرای ماموریت هر یک از نیازهای کارکردی جهت بررسی تعداد تراکنش مورد پردازش در مورد هر یک از دو مورد بایستی شرایط زیر برای آنها در نظر گرفته شود: ۱- تعداد معمول تراکنش (Normal Trans) ۲- حداکثر تراکنش (Max Trans) ۳- حداکثر تراکنش در شرایط کاری سیستم با بالاترین مقیاس (Scale Up Max Trans) ۴- تعداد معمول افزایش تراکنش در ثانیه (Trans Rate) ۵- تعداد افزایش تراکنش در شرایط فشار در ثانیه(Stress Trans Rate)
Conts	زمان اجرای بدون وقفه سیستم نرم افزاری	جهت آزمون با توجه به زمان اجرای بدون وقفه سیستم نرم افزاری سه حالت عمده می بایست مورد بررسی قرار گیرد ۱- زمان اجرای بدون وقفه کل سیستم نرم افزاری در سمت سرور ۲- زمان اجرای بدون وقفه کل سیستم در سمت کاربر ۳- زمان اجرای بدون وقفه هر یک از نیازهای کارکردی در سمت کاربر جهت بررسی زمان اجرای بدون وقفه در مورد هر یک از سه مورد بایستی شرایط زیر برای آنها در نظر گرفته شود: ۱- زمان اجرای معمول بدون وقفه (Normal Conts) ۲- حداکثر زمان اجرای بدون وقفه (Max Conts) ۳- حداکثر زمان اجرای بدون وقفه در شرایط کاری سیستم با بالاترین مقیاس (Scale Up Max Conts)

۷,۲ نیازمندی‌های کارکردی که آزمون کارایی برای آنها انجام می‌شود

همانگونه که قبلا هم تاکید شده است، ممکن است سامانه‌ای نرم‌افزاری به علت محیطی که در آن اجرا می‌شود نیازی به آزمون کارایی نداشته باشد. ولی اگر آزمون کارایی برای سیستمی نرم‌افزاری ضروری باشد مسلما تمامی بخش‌های

کارکردی آن نیازمند آزمون کارایی نمی‌باشند. بنابراین می‌بایست پس از بیان اهمیت آزمون‌های مرتبط با کارایی برای ذی‌نفعان پروژه، درباره اینکه کدام موارد کاربری نیاز به آزمون کارایی دارند تفاهم صورت پذیرد.

برای این منظور می‌توان جدولی همانند ۱۰ را در مراحل ابتدایی پروژه به تصویب ذی‌نفعان پروژه رساند.

جدول ۱۰) مشخص نمودن موارد کاربری که نیازمند آزمون کارآیی هستند.

شناسه	عنوان	شرح
F001	مورد کاربری کروکی خسارتی	افسران در مراکز پلیس کروکی های تهیه شده را وارد سیستم می کنند.
....		

۷,۳ شرایط آزمون کارایی برای موارد کاربری

درباره جنبه‌های آزمون کارایی در جدول‌های ۸ و ۹ صحبت شد. مشابه آنچه درباره عدم نیاز به آزمون کارایی برای تمامی موارد کاربری گفته شد، هر مورد کاربری که نیاز به آزمون کارایی داشته باشد نیز شاید تمامی جنبه‌های کارایی برای آن مصداق نداشته باشد.

لذا در هر پروژه می‌توان جدولی مشابه با جدول ۱۰ ایجاد کرد تا در آن برای هر مورد کاربری مشخص شود که چه جنبه‌هایی از کارایی برای آن مصداق دارد.

جدول ۱۱ مشخص‌کردن جنبه‌هایی از کارایی که برای موارد کاربری مصداق دارد. برای هر جنبه از کارآیی شرایط آن نیز مقداردهی می‌شود.

شناسه نیاز عملکردی	عنوان نیاز عملکردی	شرط آزمون	Normal	Max	Scale Up Max	(Per Second) Rate	Stress Rate (Per Second)	Scale Up Stress Rate (Per Second)
F001	مورد کاربری کروکی خسارتی	Users	20	100	1000	1	100	500
F001	مورد کاربری کروکی خسارتی	Datav						
F001	مورد کاربری کروکی خسارتی	Conts	1 Day	1 Month	6 Month	-	-	-
...								

۷,۴ انواع نیازهای غیرکارکردی

در جدول‌های ۸ و ۹ صحبت از جنبه‌های کارآیی و شرایط آنها برای انوع مختلف آزمون کارایی شد. لذا درباره جنبه‌های کلی کارایی مانند تعداد کاربران، حجم داده، تعداد تراکنش‌های همزمان و یا مدت‌زمان بدون وقفه سیستم و شرایط آن صحبت شد. سپس در جدول ۱۰ این توافق میان ذی‌نفعان پدید می‌آید که کدامین موردهای کاربری اساسا نیازمند آزمون کارایی هستند. سپس در جدول ۱۱ برای هر مورد کاربری که نیازمند آزمون کارایی می‌باشد مشخص می‌شود که چه جنبه‌هایی از کارایی برای آنها مصداق دارد.

غیر از جنبه‌های کارایی انواع مختلف نیازمندی‌های غیرکارکردی نیز مطرح هستند که تعدادی از آنها به عنوان نمونه در جدول ۱۲ آمده است. مسلما نمونه‌هایی از نیازمندی‌های غیرکارکردی که در جدول ۱۲ آمده است بر اساس پروژه‌ای مشخص آمده است که در پروژه‌های دیگر نیز شاید مصداق داشته باشد.

ولی نکته مسلم آن است که در هر پروژه باید افراد تیم توسعه به کمک دیگر ذی‌نفعان پروژه لیستی از نیازمندی‌های غیرکارکردی را تهیه نمایند.

در جدول ۱۲ همچنین مشخص می‌شود که هر نیازمندی غیرکاکردی بواسطه معماری سیستم نرم‌افزاری آیا به جنبه‌ها مختلف کاریی وابسته است. مثلا براساس سطر اول جدول ۱۲، آوره‌شدن تمام اجزای فرم در یک لحظه زمانی به تعداد کاربران استفاده کننده از سیستم وابسته است ولی به حجم داده‌های ذخیره‌شده وابسته نمی‌باشد. این که این نیازمندی غیرکارکردی به چه جنبه‌هایی از کارایی وابسته می‌باشد کاملا به معماری استفاده شده در نرم‌افزار بستگی دارد.

جدول ۱۲) انواع نیازهای غیرکارکردی

ردیف	شناسه	عنوان نیاز غیر کارکردی	Tools Base	Users dependent (Y/N)	Datav dependent (Y/N)	Trans dependent (Y/N)	Conts dependent (Y/N)	واحد سنجش	بهترین حالت	حد قابل قبول
۱	P001	آورده شدن تمام اجزاء فرم در یک لحظه زمانی	N	Y	Y	Y	Y	Y/N	Y	Y
۲	P002	زمان باز شدن LOV	N	Y	Y	Y	Y	msec	0	1
۳	P003	عدم refresh پس از ثبت	N		N			Y/N	Y	Y
۴	P004	عدم refresh پس از رفتن به رکورد بعدی	N		N			Y/N	Y	Y
۵	P005	آورده شدن تصاویر مربوط به صفحه در یک لحظه زمانی	N		Y			Y/N	Y	Y
۶	P006	زمان باز شدن فرم ورود اطلاعات	Y		Y			msec	0	5
۷	P007	زمان بستن فرم	N							
۸	P008	زمان ویرایش یک رکورد	Y							
۹	P009	زمان حذف یک رکورد	Y							
۱۰	P010	زمان جستجوی سریع برای یک رکورد در جدول	Y							
۱۱	P011	زمان ثبت رکورد جدید	Y							
۱۲	P012	زمان رفتن به رکورد بعدی	Y							
۱۳	P013	زمان فیلتر نمودن اطاعات در جدول	Y							
۱۴	P014	زمان مرتب سازی جدول								
۱۵	P015	زمان باز شدن فرم جهت گرفتن پارامتر								
۱۶	P016	زمان باز شدن گزارش با حداقل فیلتر								
۱۷	P017	زمان باز شدن گزارش با حداکثر فیلتر								
۱۸	P018	زمان باز شدن گزارش با حداقل فیلتر برای بیشترین بازه زمانی (یک سال)								
۱۹	P019	زمان باز شدن گزارش با حداکثر فیلتر برای کمترین بازه زمانی (۱ روز)								

۷,۵ ارتباط نیازهای غیرکارکردی با نیازهای کارکردی

نیازهای غیرکارکردی آورده شده در جدول ۱۲ ممکن است در تمامی نیازهای کارکردی مصداق نداشته باشد. در جدول ۱۳ مشخص می‌کنیم چه نیازهای غیرکارکردی در هر نیاز کارکردی مصداق دارد.

مسلما اتوماسیون آزمون در سرعت انجام آزمون بسیار اثربخش است. ولی به هر حال بردن فعالیت‌های آزمون در قالب اتوماسیون خود نیازمند زمان و هزینه می‌باشد. از آنجایی که زمان و منابع انجام آزمون قبل از هر نصب محدود می‌باشد، بهتر است آزمون‌های مختلفی که در جدول ۱۳ آمده است نسبت به هم اولویت‌بندی شوند تا بتوان باتوجه به محدودیت در منابع موجود بهترین اثربخشی را در آزمون تجربه کرد.

جدول ۱۳) اولویت دهی به نیازهای غیرکارکردی در ارتباط با نیازهای کارکردی

نیاز کارکردی		نیاز غیرکارکردی		اولویت
شناسه	عنوان	شناسه	عنوان	(۱-۱۰)
F001	مورد کاربری کروکی خسارتی	P006	زمان باز شدن فرم ورود اطلاعات	
F001	مورد کاربری کروکی خسارتی		زمان بستن فرم	
F001	مورد کاربری کروکی خسارتی		زمان ویرایش اطلاعات	
F001	مورد کاربری کروکی خسارتی		زمان حذف اطلاعات	
F001	مورد کاربری کروکی خسارتی		زمان جستجوی سریع برای یک رکورد در جدول	
F001	مورد کاربری کروکی خسارتی		زمان ثبت رکورد جدید	
F001	مورد کاربری کروکی خسارتی		زمان رفتن به رکورد بعدی	
F001	مورد کاربری کروکی خسارتی		زمان فیلتر نمودن اطاعات در جدول	
F001	مورد کاربری کروکی خسارتی		زمان مرتب سازی جدول	
R001	گزارش کروکی خسارتی		زمان باز شدن فرم جهت گرفتن پارامتر	
R001	گزارش کروکی خسارتی		زمان باز شدن گزارش با حداقل فیلتر	
R001	گزارش کروکی خسارتی		زمان باز شدن گزارش با حداکثر فیلتر	
R001	گزارش کروکی خسارتی		زمان باز شدن گزارش با حداقل فیلتر برای کمترین باز ه زمانی (۱ روز)	
R001	گزارش کروکی خسارتی		زمان باز شدن گزارش با حداقل فیلتر برای بیشترین باز ه زمانی (یک سال)	
R001	گزارش کروکی خسارتی		زمان باز شدن گزارش با حداکثر فیلتر برای کمترین باز ه زمانی (۱ روز)	
R001	گزارش کروکی خسارتی		زمان باز شدن گزارش با حداکثر فیلتر برای بیشترین باز ه زمانی (یک سال)	

بخش سوم: آنالیز و طراحی نرم‌افزار

۸ آنالیز و طراحی نرم‌افزار

همانگونه که قبلا نیز بیان شد، بحث اصلی در مهندسی نرم‌افزار این است که در هر پروژه نرم‌افزاری چه‌کسی، چه‌کاری را، کی و چگونه انجام دهد تا سیستم نرم‌افزاری براساس بودجه‌اش در زمان برنامه‌ریزی شده و باکیفیت موردنظر درخواست‌دهندگان آن تولید شود. تاکنون در بخش دوم کتاب، یعنی شناخت نیازمندی‌های نرم‌افزار، درباره کار (چه‌کاری) شناخت نیازمندی‌ها در پروژه صحبت کردیم و تاحدی نیز درباره چگونگی انجام آن به تفصیل صحبت کردیم. در ادامه کتاب و در این فصل درباره کار (چه‌کاری) تحقق[۱۱۹] نیازمندی‌ها و تاحدی چگونگی انجام آن صحبت می‌کنیم.

در شناخت نیازمندی‌ها مهندسین نباید خود را درگیر چگونگی پیاده‌سازی آن کنند. بلکه باید تمامی سعی خود را بر شناخت اهداف سیستم نرم‌افزری، ویژگی‌هایش، نیازهای کارکردی و غیرکارکردی‌اش و موارد آزمون آنها متمرکز سازند.

بسته به معماری انتخاب شده برای نرم‌افزار، مسلما چگونگی پیاده‌سازی نیازمندی‌ها متفاوت خواهد بود. هرچند جنبه‌های پایه معماری نرم‌افزار در چگونگی شناخت نیازمندی‌های نرم‌افزار اثر می‌گذارد ولی معماری نرم‌افزار بیشترین تاثیر را در چگونگی تحقق نیازمندی‌ها خواهد داشت.

البته واژه معماری نرم‌افزار مفهوم وسیعی دارد و دیدهای[۱۲۰] مختلفی از نرم‌افزار را شامل می‌شود. مثلا در فصل ۶ درباره شناخت موردهای کاربری

---

[119] Realization
[120] View

صحبت شد. نمودار موردهای کاربری و یا سناریوی مربوط به موردهای کاربری اصلی نیز از اجزای معماری نرم‌افزار می‌باشند. ولی بخش عمده معماری نرم‌افزار به چگونگی محقق‌ساختن نیازمندی‌ها تمرکز دارد.

در این کتاب رویکرد ما در بیان روش برای چگونگی تحقق‌ساختن نیازمندی‌ها، مبتنی بر شیءگرایی می‌باشد. لذا بخش‌های بعدی فصل ۹ کتاب مبتنی بر آن می‌باشد.

می‌بایست در ابتدا تاکید کنم که چگونگی آنالیز و طراحی در پروژه ارتباط بسیار نزدیکی به متدولوژی مورد استفاده دارد. مثلا در RUP برای آنالیز و طراحی ۱۹ فعالیت مشخص را بیان شده است و نقش‌هایی مانند معمار نرم‌افزار، طراح همروندی[۱۲۱]، طراح پایگاه داده، طراح، طراح واسط کاربری و بازبین فنی[۱۲۲] معرفی کرده است که هرکدام فعالیت‌هایی تخصصی مطابق با نقششان را انجام می‌دهند.

۸,۱ تفاوت آنالیز و طراحی در مهندسی نرم‌افزار

فرض کنیم معماری اصلی ما در پیاده‌سازی مبتنی بر شیءگرایی است. در این صورت در آنالیز به دنبال درک مساله با تبدیل آن به کلاس‌هایی هستیم که مساله را به شیءگرایی نگاشت می‌کند. ولی در طراحی، کلاس‌های آورده‌شده باید دقیقا مبتنی بر پیاده‌سازی باشد. لذا در طراحی باید دقیقا چارچوب مورد استفاده در پیاده‌سازی مشخص باشد تا طراحی کلاس‌ها نیز مبتنی بر آنها باشد.

[121] Capsule designer
[122] Technical reviewer

پس از طراحی باید برنامه‌نویسی به سادگی و با پیاده‌سازی متدها بتواند پیاده‌سازی نرم‌افزار را انجام دهد. این در صورتی است که خرجی آنالیز مناسب برای برنامه‌نویسی نمی‌باشد.

البته بستگی به شناخت تیم‌برنامه‌نویسی به معماری و آشنایی آنها با پیاده‌سازی‌های قبلی در حوزه نیازمندی‌های پروژه، می‌تواند آنالیز، طراحی و پیاده‌سازی در مرحله پیاده‌سازی به یکباره بدون نیاز به مدل‌سازی‌های رایج در آنالیز و طراحی انجام شود. مسلما تشخیص این مساله به شناخت ذی‌نفعان از پروژه و متدولوژی انتخابی بستگی دارد.

۸,۲ آنالیز موارد کاربری

اهداف آنالیز موارد کاربری را می‌توان به صورت زیر تشریح کرد:

شناسایی کلاس‌هایی که قابلیت اجرای سناریوهای اصلی و فرعی آورده شده در هر مورد کاربری را دارند. لذا برای هر مورد کاربری، کلاس‌های آن مشخص می‌شود.

۱) برای هر کلاس متدها[۱۲۳]، ویژگی‌ها[۱۲۴] و ارتباط آن با کلاس‌های دیگر شناسایی شود.
۲) با شناسایی کلاس‌های عمده و متدها و ویژگی‌های عمده آنها عملا معماری سیستم نرم‌افزاری پی‌ریزی می‌شود.

در آنالیز موارد کاربری گام‌های زیر برای هر مورد کاربری انجام می‌شود.

الف) ایجاد مدل برای تحقق نیازمندی

[123] Methodes
[124] Attributes

ابزارهای مدل‌سازی نقشی اساسی در انجام آنالیز و طراحی موارد کاربری دارند. بنابراین بسته به اینکه چه ابزاری برای مدل‌سازی در پروژه استفاده می‌شود، برای هر مورد کاربری پوشه‌ای برای تحقق نیازمندی‌های آن ایجاد می‌شود. در پوشه مربوطه برای هر مورد کاربری، نمودارها و المان‌های مختلف برای آنالیز و طراحی آن مورد کاربری آورده می‌شود.

ب) افزودن توصیف‌های تکمیلی به سناریوهای موارد کاربری

در بسیاری از موارد توصیف‌های آورده شده در موارد کاربری برای شناسایی کلاس‌ها در طراحی شیءگرایی کافی نیست. در این موارد باید توصیف‌های آورده شده تکمیل شود تا بتوان در ادامه کلاس‌های برنامه را از آنها استخراج کرد.

مثلا فرض کنید در سناریوی مربط به سامانه ماشین خودپرداز (ATM[۱۲۵]) جمله زیر آورده شده است:

در ATM، کارت مشتری بانک اعتبار سنجی می‌شود

این جمله از سناریو هرچند ممکن است برای آگاهی بهره‌برداران کافی باشد ولی اطلاعاتی درباره اینکه واقعا در ATM چه اتفاقاتی می‌افتد تا کارت اعتبارسنجی شود را شامل نمی‌شود.

لذا مورد کاربری باید مجدد بازنویسی شود تا اطلاعاتی کافی درباره چگونگی انجام کار را نیز شامل شود. چنانچه همان مثال قبل را بخواهیم برای این منظور مجدد بازنویسی کنیم، توصیف تکمیلی را به صورت زیر بیان می‌کنیم:

[125] Auto Teller Machine (ATM)

دستگاه ATM شماره کارت مشتری را به همراه کلمه عبور آن را به سرور ATM می‌فرستد تا اعتبارسنجی شود. سپس چنانچه شماره کارت با کلمه عبور آن همخوانی داشت، موفقیت و در غیر این صورت عدم موفقیت توسط سرور برگردانده می‌شود. با برگردانده شدن موفقیت، مشتری می‌تواند ادامه فعالیت را انجام دهد و در غیر این صورت ادامه فعالیت، نیازمند اعتبارسنجی مجدد خواهد بود.

با این بیان مجدد، اطلاعاتی که برای انجام کار نیاز است به صورت شفاف‌تری در اختیار تحلیگر قرار می‌گیرد. در این سناریوی تکمیلی مشخص شده است که چه‌کسی موظف به انجام عملیات اعتبارسنجی می‌باشد. لذا سرور ATM به عنوان کنشگری جدید در این مرحله شناسایی می‌شود. همچنین با این توصیف تکمیلی می‌توان کلاس مشتری با ویژگی‌های شماره کارت و کلمه عبور و نیز کلاسی برای واسطه سرور ATM شناسایی کرد.

بنابراین در این مرحله می‌بایست سناریوهای آورده شده در هر مورد کابری را مجدد بازنگری کنیم که آیا چگونگی رفتار سیستم به صورت کاملا شفاف بیان شده است. لذا نباید هیچگونه ابهامی در بیان چگونگی رفتار سیستم وجود داشته باشد. توجه داشته باشیم که در توصیف بدون ابهام رفتار، نیاز نیست تا مثلا کلاس‌هایی هم که می‌توانند مسئول اجرای رفتار باشند را شناسایی کنیم. تنها بیان شفاف کارهایی که انجام می‌شود کفایت می‌کند.

برای شفاف‌سازی سناریوها نیاز است تا با افراد خبره در آن دامنه‌ای که سیستم نرم‌افزاری پیاده‌سازی می‌شود، مشورت داشته باشیم. بنابراین در هر بخش از سناریوی موارد کاربری باید پاسخ این سوال مبنی بر "این که می‌گوییم سیستم این کار را انجام می‌دهد دقیقا چه معنایی می‌دهد؟" را دریافت داریم.

ج) یافتن کلاس‌های آنالیز از رفتار آورده‌شده در هر مورد کاربری

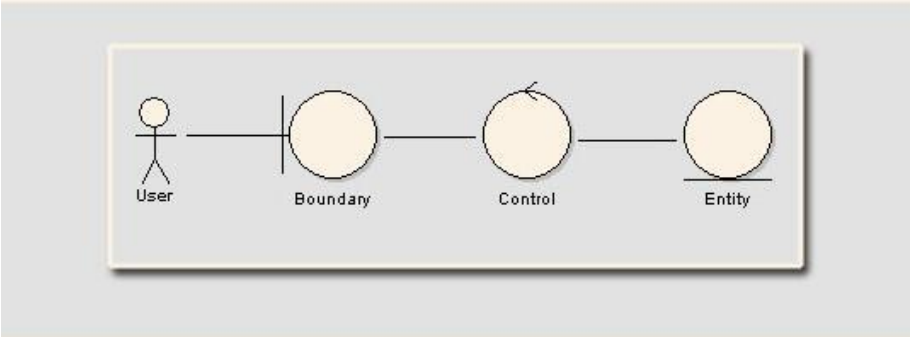
هدف از این مرحله آن است تا مجموعه‌ای اولیه از کلاس‌هایی که می‌توانند رفتار سیستم را پیاده‌سازی کنند را بدست آوریم. برای این که دسته‌بندی اولیه‌ای از کلاس‌های شناسایی شده داشته باشیم می‌توانیم سه کلیشه[۱۲۶] برای کلاس‌ها درنظر بگیریم. برای این منظور به صورت کلی سه کلیشه واسط کاربری[۱۲۷]، کنترلی و داده‌ای[۱۲۸] (مدل[۱۲۹]) را می‌توان برای کلاس‌ها درنظر گرفت.

کلاس‌هایی که در کلیشه کنترلی قرار می‌گیرند مسئول پیاده‌سازی منطق‌های کنترلی در سیستم هستند. کلاس‌های داده‌ای وظیفه ارتباط با داده‌ها و یا به عبارتی دیگر ارتباط با پایگاه داده را برعهده دارند. کلاس‌های واسط کاربری هم وظیفه نشان دادن اطلاعات و دریافت اطلاعات از کاربر را برعهده دارند. برای هر کلاس شناسایی شده در این مرحله باید نام و توصیف آنها در تعدادی جمله آورده شود.

در شکل ۸ در نمودار کلاسی فرضی، چگونگی ارتباط این کلاس‌ها با هم و نیز با کنشگر اصلی سیستم نشان داده شده است. همانگونه که در شکل نشان داده شده است، کاربر تنها با کلاس‌هایی از کلیشه واسط کاربری در ارتباط است. واسط‌های کاربری نیز تنها با کلاس‌هایی از کلیشه کنترل و کلاس‌های داده‌ای (مدل) نیز مجازند تنها با کلاس‌های کنترلی در ارتباط باشند. بنابراین در این میان تنها کلاس‌های کنترلی مجازند تا با هر دو کلیشه کلاس‌های واسط کاربری

[126] Stereotypes
[127] Boundary
[128] Entity
[129] Model

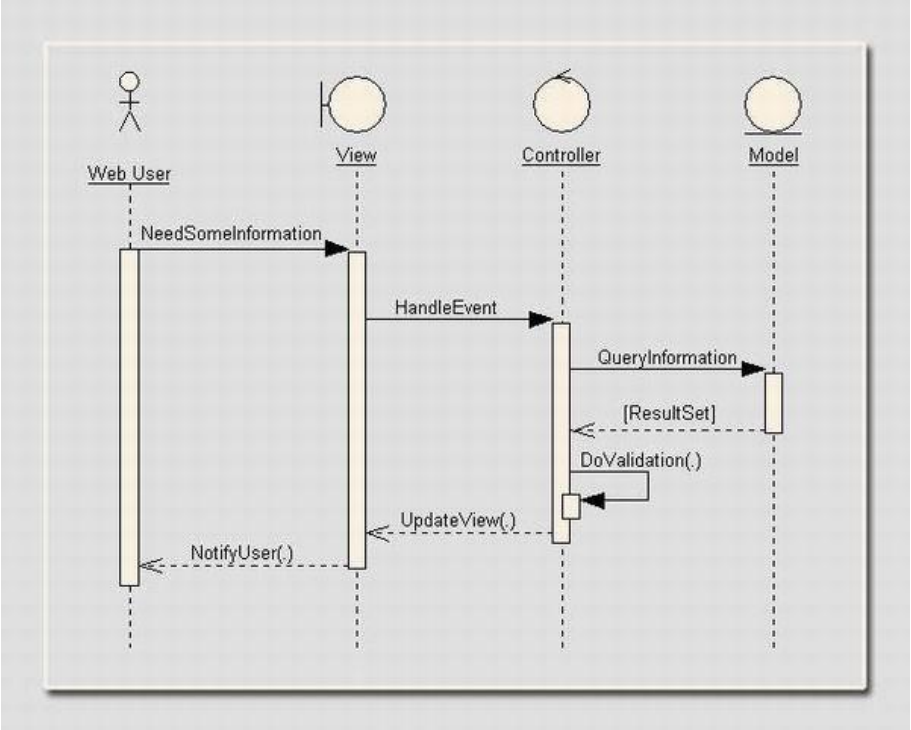
و داده‌ای در ارتباط باشند. این معماری که در چارچوب‌های مختلفی در طراحی مرسوم است با عنوان معماری سه‌لایه[۱۳۰] شناخته می‌شود.



شکل ۸) مثالی از نماد نمایش سه کلیشه متفاوت از کلاس‌ها و چگونگی ارتباط آنها با هم در معماری سه لایه

در شکل ۹، نمودار توالی[۱۳۱] از مثالی فرضی برای نمایش چگونگی ارتباط بین کلاس‌ها آورده شده است. در این نمودار مشخص شده است که برای آنکه درخواستی از کاربر پاسخ داده شود، چه نمونه‌هایی از چه کلاس‌هایی باید وجود داشته باشد و متدهای آنها به چه ترتیبی اجرا گردد.

[130] Three-layer
[131] Sequence diagram



شکل ۹) نموداری فرضی که توالی صدازدن متدهای کلاس‌ها از سه کلیشه مختلف را در معماری سه‌لایه نشان می‌دهد.

این گونه کلیشه‌بندی کلاس‌ها، هرچند در آنالیز بسیار اهمیت دارد ولی کلیشه‌بندی کلاس‌ها در طراحی مبتنی بر تکنولوژی پیاده‌سازی و چارچوب[۱۳۲] مورد استفاده در هر پروژه است. هرچند که این نوع کلیشه‌بندی در بسیاری از چارچوب‌های طراحی نیز رعایت می‌شود.

مثلا در شیءگرایی می‌توان برای ذخیره‌سازی داده‌ها و ارتباط با پایگاه‌داده رابطه‌ای از چارچوب هایبرنیت[۱۳۳] استفاده کرد. چنانچه این چارچوب برای

[132] Framework
[133] Hibernate

پیاده‌سازی درنظر گرفته شود مسلما چگونگی طراحی کلاس‌هایی که برای ذخیره‌سازی داده‌ها هستند نیز مبتنی بر آن خواهد بود.

همین مثال را برای طراحی کلاس‌های واسط کاربری در مرحله طراحی نیز می‌توان درنظر گرفت. مثلا برای طراحی واسط کاربر در جاوا مبتنی بر وب، چارچوب‌های مختلفی وجود دارد[۱۳۴]. چارچوب‌هایی مانند JSP[۱۳۵]، JSF[۱۳۶] و استراتس[۱۳۷] در این حوزه قرار می‌گیرند. با این رویکرد، طراحی کلاس‌های واسط کاربری باید مبتنی بر چارچوب انتخابی باشد.

مشابه دو مثال قبل، در جاوا می‌توان چارچوب اسپرینگ[۱۳۸] را برای کلاس‌های کنترلی درنظر گرفت. در این چارچوب بسته به اینکه کلاس ما چه متدهایی دارد و هر متد چه کاری را چگونه باید انجام دهد چگونگی طراحی مبتنی بر چارچوب متفاوت خواهد بود.

هدف من از بیان این سه دسته مثال، در چگونگی پیاده‌سازی سه کلیشه عمده واسط کاربری، کنترلی و داده‌ای، آن است تا هم تخصصی بودن فرآیند طراحی نرم‌افزار و هم تنوع انجام این کار را، بسته به رویکردهای پیاده‌سازی مانند جاوا و دات‌نت[۱۳۹]، برجسته نمایم. مسلما در این کتاب و یا هیچ کتاب دیگری نمی‌توان درباره چگونگی طراحی در چارچوب‌های مختلف و مزایا و معایب آنها صحبت کنیم. چراکه شیوه طراحی در هر چارچوب قواعد و دانش‌های تجربی نیاز دارد که برای آن کتاب‌های متعدد و منابع اطلاعاتی فراوانی در اینترنت وجود دارد.

---

[134] https://en.wikipedia.org/wiki/Java_view_technologies_and_frameworks
[135] Java Servere Page (JSP)
[136] JavaServer Faces
[137] Struts
[138] Spring
[139] Microsoft .Net

د) توصیف مسئولیت‌های هر کلاس[۱۴۰]

مسئولیت‌های هر کلاس از دو منظر قابل بررسی است:

۱) کارهایی[۱۴۱] که اشیاء از نوع آن کلاس می‌توانند انجام دهند.
۲) دانش و اطلاعاتی که هر کلاس نگه می‌دارد و می‌تواند در اختیار کلاس‌های دیگر نیز قرار دهد.

معمولا کلاس‌هایی که در سطح آنالیز هستند مسئولیت‌های مختلفی دارند. چراکه چنانچه بخواهیم برای هر مسئولیت یک کلاس ساده تعریف کنیم، در این صورت تعداد کلاس‌های آنالیز بسیار زیاد خواهد شد که خود سبب پیچیدگی در فهم تحلیل می‌شود.

در توصیف مسئولیت‌های هر کلاس باید از نام مناسب برای متد یا ویژگی استفاده کرد و چنانچه نام آنها نتواند بیان‌کننده مسئولیتشان باشد، باید با چند جمله آن مسئولیت توصیف شود.

ه) توصیف ویژگی‌ها، متدها و رابطه‌های بین کلاس‌ها

هدف از این مرحله آن است تا کلاس‌های دیگری که به کلاس‌های تعریف شده تا حالا وابسته هستند، شناسایی شوند. همچنین در این مرحله متدها و ویژگی‌های کلاس‌های که شناسایی شده‌اند دقیقتر تعریف می‌شوند.

در آنالیز نرم‌افزار مبتنی بر شیءگرایی، کلاس‌ها برای آنکه بتوانند مسئولیت‌های[۱۴۲] خود را انجام دهند به کلاس‌های دیگر وابسته هستند. برای این

[140] Responsibilities
[141] Actions
[142] Responsibilities

منظور ارتباطات[۱۴۳] بین کلاس‌ها باید مستند گردد. مستندسازی ارتباطات بین کلاس‌ها در آنالیز سبب می‌شود تا چگونگی اتصالات[۱۴۴] بین کلاس‌ها مشخص شود تا هرجا که امکان داشته باشد بتوان اتصالات بین کلاس‌ها را کاهش داد. کاهش اتصالات بین کلاس‌ها، چناچه صحیح صورت پذیرد، سبب می‌شود تا سیستم پایداری بیشتر داشته باشد. چرا که هرچه ارتباطات بین کلاس‌ها بیشتر باشد، کارکرد ناصحیح در هر یک می‌تواند به صورت بالقوه سبب بروز خطا در کلاس‌های دیگر شود.

با گام‌های زیر می‌توانیم ویژگی‌های هرکلاس و ارتباطات بین کلاس‌ها را تعریف کنیم.

۱) تعریف ویژگی‌ها

ویژگی‌ها برای این در کلاس‌ها تعریف می‌شوند تا اطلاعات کلاس را در خود ذخیره کنند.

همانگونه که می‌دانیم در شیءگرایی، تمامی سامانه نرم‌افزاری با کلاس‌هایی مدل می‌شود. کلاس‌ها ساختارهایی هستند که تا زمانی که یک نمونه یا نمونه‌هایی از آنها ایجاد نشود قابل استفاده و یا قابل ارجاع نیستند. مثلا فرض کنید که یک کلاس وظیفه گرفتن پشتیبان از پایگاه داده را برعهده داشته باشد. مسلما برای اینکه بتوان از وظیفه محول شده به این کلاس استفاده کرد، باید یک نمونه از این کلاس ایجاد شود و متدی که مربط به پشتیبان‌گیری از پایگاه داده است را در آن نمونه صدا زد. در شیءگرایی به نمونه ایجاد شده از کلاس شیء گفته می‌شود.

[143] Associations
[144] Coupling

ویژگی‌هایی که برای هر کلاس تعریف می‌شود یا اطلاعات کلاس را نگه می‌دارند و یا وظیفه نگهداری اطلاعات هر شیء را برعهده دارند. به صورت خاص چنانچه در تعریف ویژگی کلاس از واژه *static* استفاده شود، بیانگر آن خواهد بود که ویژگی در سطح کلاس است. یعنی چناچه مقدار آن تغییر کند، تغییرات وابسته به شیء مشخصی نمی‌باشد. از طرف دیگر چنانچه از واژه *static* استفاده نشود، بیانگر آن خواهد بود که ویژگی متعلق به شی می‌باشد و بخشی از اطلاعات شیء را نگهداری می‌کند.

ویژگی‌ها همچنین در هنگام تعریف در کلاس می‌توانند به سه روش دسترسی *public*، *private* و *protect* تعریف شوند. چنانچه ویژگی public باشد یعنی در خارج از کلاس هم می‌توان به آن دسترسی داشت. برعکس آن، چنانچه ویژگی private باشد می‌بایست حتما از طریق متدهای همان کلاس استفاده شود. در روش دسترسی protect، علاوه برمتدهای همان کلاس، متدهای کلاس‌هایی که ارتباط از نوع ارث‌بری[۱۴۵] با آن کلاس دارند نیز می‌تواند مقدار آن را خوانده و یا تغییر دهند. در شیءگرایی معمولا برای ویژگی‌ها حالت دسترسی *private* درنظر گرفته می‌شود و برای استفاده در دیگر کلاس‌ها از متدها *get* و *set* استفاده می‌شود.

همانگونه که قبلا نیز بیان گردید، ویژگی‌ها برای نگهداری اطلاعات کلاس و اشیاء ایجاد شده از آن کلاس استفاده می‌شود. چنانچه این اطلاعات (۱) رفتار پیچیده‌ای داشته باشد و نتوان با نوع‌های ساده مانند عدد و رشته آن را تعریف کرد، (۲) نیاز باشد تا بین چند شیء به اشتراک گذاشته شود و یا (۳) مقدار آن با

[145] Inheritance

ارجاع[۱۴۶] بین چند شیء به عنوان پارامتر انتقال داده شود، در این صورت اطلاعات خود نیز به صورت کلاس تعریف می‌شود.

در نام‌گذاری ویژگی‌های هر کلاس، باید دقت شود که نام آن بیانگر اطلاعاتی که در آن نگهداری می‌شود باشد. چنانچه نتوان ماهیت اطلاعات ذخیره شده را با نام ویژگی تبیین کرد، در کنار نام از توصیف ویژگی نیز استفاده می‌شود.

۲) برقراری ارتباط بین کلاس‌ها

کلاس‌های شناسایی شده در آنالیز می‌توانند با کلاس‌های دیگر نیز ارتباط داشته باشند. ارتباط بین کلاس‌ها بیانگر آن است که شیء‌های ایجاد شده از آن کلاس‌ها برای انجام سناریوهای آورده شده در مورد کاربری باید با هم ارتباط داشته باشند. هرچند در آنالیز تنها از ارتباط بین کلاس‌ها صحبت می‌شود، در طراحی بسته به چارچوب مورد استفاده جنبه‌های دقیق‌تری از ارتباط مطرح می‌شود.

در آنالیز بواسطه آنکه در ابتدای مراحل مربوط به شناسایی کلاس‌ها هستیم، تنها از ارتباطات بین کلاس‌ها به بیان نوع رابطه از نوع وابستگی[۱۴۷]، تعلق[۱۴۸] و وابستگی اشتراکی[۱۴۹] اکتفا می‌کنیم و مسائل بیشتر در وابستگی بین کلاس‌ها را به مرحله طراحی و چارچوب انتخاب شده واگذار می‌کنیم. در سه بخش بعدی درباره چگونگی بیان ارتباط بین کلاس‌ها با سه رابطه وابستگی، تعلق و وابستگی اشتراکی صحبت خواهیم کرد.

[146] By reference
[147] Association
[148] Aggregation
[149] Subscrib-association

بنابراین در این مرحله، برای هر مورد کاربری نمودار کلاس[۱۵۰] ایجاد می‌شود که ارتباطات بین هر کلاس با کلاس‌های دیگر را نشان می‌دهد. در نمودار کلاس برای ارتباطاتی که می‌تواند ابهام در چرایی ایجاد آنها وجود داشته باشد، باید توضیحات لازم را اضافه کرد.

۳) توصیف وابستگی اشتراکی در آنالیز

اشیاه در برخی از موارد نیاز دارند تا بدانند که کی اتفاقی[۱۵۱] در شیء دیگری (شیء هدف[۱۵۲]) روی داده است. در این موارد نیاز نیست تا حتما شیء هدف بداند که چه هنگام این اتفاق رخ داده است. برای اینکه بتوان چنین ارتباطی را بین دو شیء برقرار کرد، وابستگی اشتراکی[۱۵۳] را می‌توان در نمودار کلاس اضافه کرد تا به صورت کاملا موجز این ارتباط بیان شود.

در وابستگی اشتراکی باید شرطی را تعریف کرد تا پس از برقراری آن، کلاسی که نیاز به دانستن دارد (مشترک[۱۵۴]) مطلع گردد.

وابستگی اشتراکی در موارد زیر نیاز می‌باشد:

- چنانچه شیءای بواسطه تغییرات در شیءای دیگر تحت تاثیر قرار می‌گیرد.
- چنانچه شیء جدیدی بواسطه اتفاقی باید ایجاد شود. مثلا زمانی‌که خطایی رخ می‌دهد می‌بایست پنجره جدیدی ایجاد شود تا به کاربر اطلاع دهد.

[150] Class diagram
[151] Event
[152] Target object
[153] Subscribe-assiciation
[154] Subscriber

- چنانچه شیءای نیاز دارد تا درباره زمان مقداردهی، تغییرات و یا اتمام شیء دیگر مطلع شود.

معمولا اشیائی که اشتراک بر آنها ایجاد می‌شود، از نوع داده‌ای هستند. چرکه این اشیاء ماهیتی غیرفعال[۱۵۵] دارند و اشیاء دیگر را صدا نمی‌زنند. از طرفی اشیاء بسیاری هستند که نیاز دارند تا از تغییرات در اشیاء داده‌ای مطلع شوند. وابستگی اشتراکی سبب می‌شود تا اشیاء داده‌ای از چنین نیازی مطلع نشوند و در عین حال این نیاز برای دیگر اشیاء پاسخ داده شود.

در طراحی با چارچوب‌هایی مانند اسپرینگ در جاوا، چگونگی این فرآیند و نحوه ثبت اشیاء برای اشتراک کاملا مشخص می‌باشد. ولی در آنالیز چگونگی این عمل مهم نمی‌باشد بلکه توجه ما بر شناسایی چنین ارتباطی بین کلاس‌ها متمرکز می‌شود.

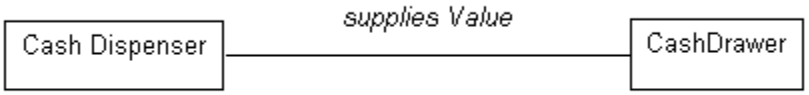
۸,۳ وابستگی بین کلاس‌ها

وابستگی بین کلاس‌ها را با مثال زیر شروع می‌کنم.

فرض کنیم در سیستم ماشین خودپرداز (ATM)، صندوق پول[۱۵۶]، پول مورد نیاز پرداخت‌کننده پول[۱۵۷] را فراهم می‌کند. بنابراین برای اینکه شیءای از نوع پرداخت‌کننده پول بتواند فرایند پرداخت پول را دنبال کند، بایست بتواند به شیء صندوق پول دسترسی داشته باشد. از نگاهی دیگر گاهی نیاز است تا صندوق پول بتواند پرداخت‌کننده پول را از وضعیت موجودی صندوق خود مطلع سازد. لذا

[155] Passive
[156] Cash drawer
[157] Cash dispenser

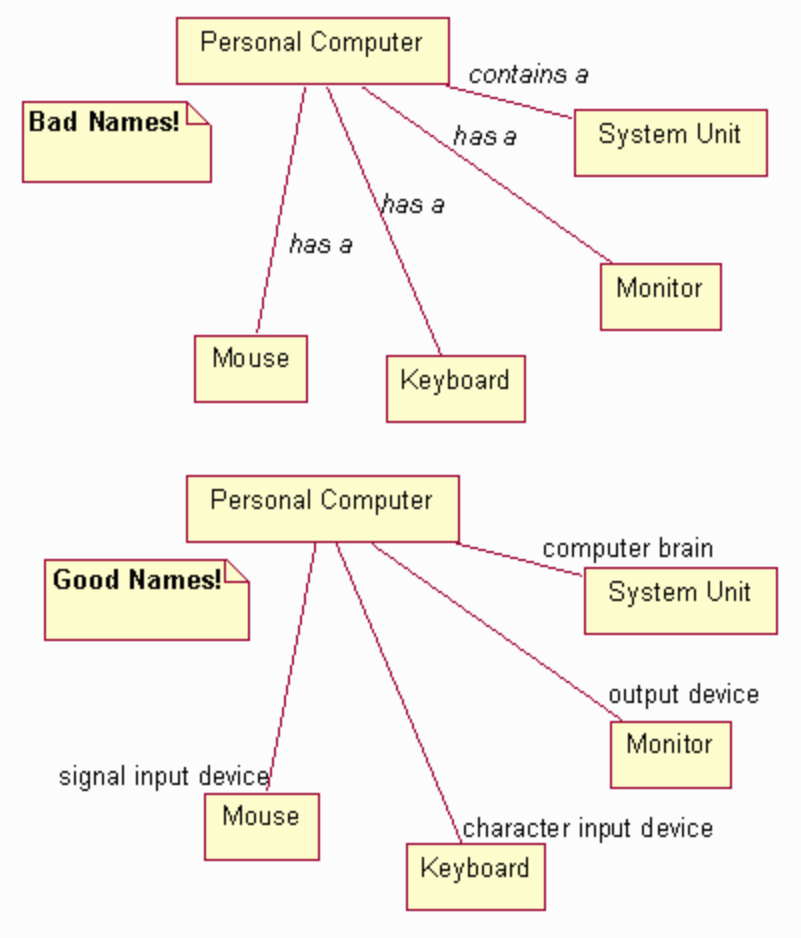
صندوق پول هم باید بتواند به پرداخت‌کننده پول دسترسی داشته باشد. وابستگی بین این دو کلاس که در شکل ۸ نشان داده شده است. از ارتباط نشان‌داده‌شده در این شکل مشخص می‌شود که هر دو کلاس نیاز دارند به متدهایی از کلاس دیگر دسترسی داشته باشند.



شکل ۱۰) نمودار کلاس که در آن وابستگی بین دو کلاس آورده شده است.

۸,۳,۱ نام‌گذاری در وابستگی بین کلاس‌ها

در شکل ۱۰ برای وابستگی بین دو کلاس، نام نیز درنظر گرفته شده است. در انتخاب نام وابستگی، باید دقت‌نظر داشت. چراکه نام نامناسب می‌تواند سبب شود تا آنالیز، اثربخشی مناسب را برای انتقال مفاهیم میان اعضای تیم توسعه نداشته باشد. در شکل ۱۱، دو نمودار کلاس با وابستگی بین کلاس‌ها آورده شده است. در بخش اول شکل ۱۱ در نام‌گذاری دقت نشده است و در بخش دوم آن سعی‌شده است نام‌های کاربردی برای وابستگی تعریف شود.



شکل ۱۱) مقایسه نام‌گذاری خوب و بد در وابستگی بین کلاس‌ها.

۸,۳,۲ نقش هر کلاس در وابستگی

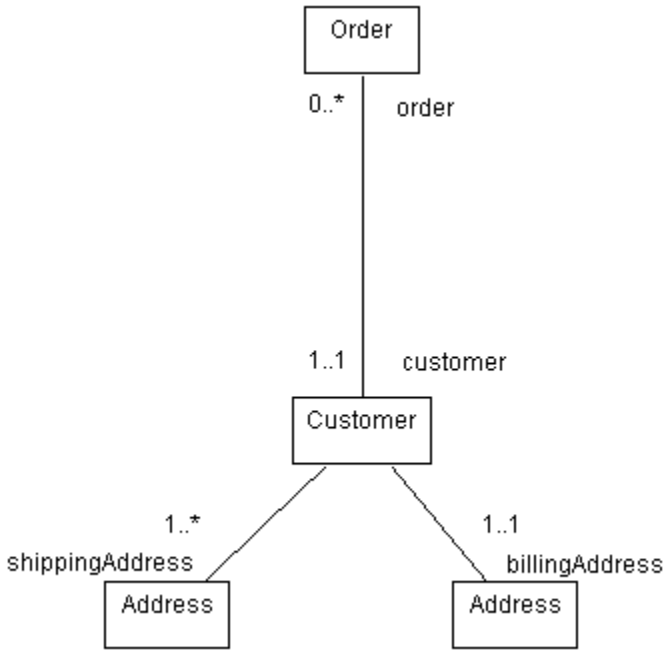
در هر وابستگی، هر کدام از کلاس‌ها نقشی[۱۵۸] دارند که بیانگر چگونگی تاثیرگذار آن کلاس در وابستگی می‌باشد. مثلا چنانچه دو کلاس استاد و درس داشته باشیم، می‌توان نام مدرس را برای نقش کلاس استاد در وابستگی درنظر

[158] Role

گرفت. در نام‌گذاری نقش‌ها نباید از کلماتی مانند "دارد" و یا "شامل می‌شود" استفاده کرد. چراکه این کلمات اطلاعات خاصی درباره ارتباطات بین کلاس‌ها منتقل نمی‌کنند.

همچنین توجه داشته باشیم که یا برای وابستگی نام‌گذاری انجام می‌شود و یا برای کلاس‌های در وابستگی نقش تعریف می‌شود. لذا هردوی آنها باهم انجام نمی‌شود. معمولا مشخص‌کردن نقش نسبت به نام‌گذاری رابطه اولویت دارد و این اولویت در مرحله طراحی اجباری می‌شود.

مثالی از نمودار کلاس که در شکل ۱۲ آمده است را درنظر بگیرید. همانگونه که در شکل آمده است، مشتری می‌تواند دو نوع آدرس متفاوت داشته باشد. یکی از آدرس‌ها برای ارسال صورتحساب است و آدرس دیگر برای ارسال سفارش می‌باشد. بنابراین بین کلاس مشتری و آدرس دو وابستگی وجود دارد که نقش کلاس آدرس در هر وابستگی متفاوت می‌باشد.



شکل ۱۲) نشان‌دادن نقش کلاس‌ها و نیز چندگانگی در وابستگی.

۸,۳,۳ چندگانگی

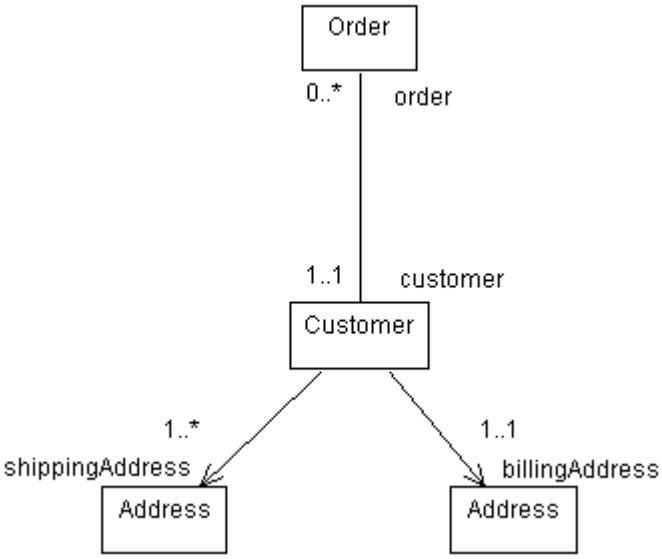
چندگانگی[۱۵۹] برای نقش هر کلاس در وابستگی مشخص می‌شود. در چندگانگی مشخص می‌شود که چند شیء از کلاس با یک شیء از کلاس دیگر می‌تواند وابسته باشد. مثلا در شکل ۹ نشان داده شده است که هر سفارش (هر شیء از نوع سفارش) دقیقا با یک مشتری (شیء از نوع مشتری) وابسته است. این وضعیت در نمودار کلاس به صورت 1..1 نشان داده شده است. همچنین هر مشتری می‌تواند سفارشی نداشته باشد و یا چندین سفارش برایش وجود داشته باشد. این حالت نیز در شکل با *..1 نشان داده شده است. چندگانگی بین

[159] Multiplicity

مشتری و آدرس در وابستگی که نقش آدرس در آن آدرس ارسالی است، *..1 می‌باشد. یعنی هر مشتری حداقل باید یک آدرس برای ارسال سفارش داشته باشد. چندگانگی دیگر آورده شده در شکل به صورت 1..1 می‌باشد. یعنی هر مشتری باید دقیقا یک آدرس برای ارسال صورتحساب داشته باشد.

۸,۳,۴ جهت در وابستگی

هر وابستگی در هر دو سمت خود می‌تواند دارای جهت باشد. در شکل ۱۱ وابستگی بین سفارش و مشتری دارای جهت دو سویه است. البته در نمودار کلاس برای نشان‌دادن دوسویه بودن وابستگی، در هیچ سمتی از آن جهت مشخص نمی‌شود. براساس جهت مشخص شده در این وابستگی، هم سفارش باید اطلاعات مشتری خود را بداند و هم مشتری باید بداند که چه سفارش‌هایی تاکنون داشته است. ولی ارتباط بین مشتری و آدرس یک‌طرفه است.یعنی نیازی نیست که آدرس بداند تا متعلق به چه مشتری است.



شکل ۱۳) نشان‌دادن جهت در وابستگی.

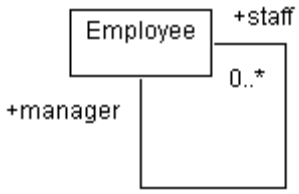
۸,۳,۵ وابستگی به خود

در پاره‌ای از موارد، کلاسی ممکن است به خود وابستگی داشته باشد. البته این الزما به این معنی نمی‌باشد که نمونه‌ای از کلاس دقیقا به خود وابسته باشد. بلکه اغلب به این معنا می‌باشد که نمونه‌ای از کلاس به نمونه‌های دیگری از همین کلاس وابسته است. در وابستگی به خود[۱۶۰] مشخص‌کردن نقش‌ها بسیار مهم است. چراکه با این روش می‌توان منظور از وابستگی را بهتر بیان کرد.

در شکل ۱۴ هر کارمند یک مدیر دارد و هر مدیر صفر یا بیشتر کارمند دارد. جهت وابستگی در این نمودار دوطرفه است. یعنی هم کارمند می‌تواند بداند که

[160] Self-association

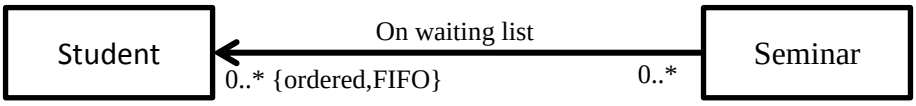
چه کسی مدیرش است و هم مدیر می‌تواند به اطلاعات کارمندانش دسترسی داشته باشد.



شکل ۱۴) وابستگی به خود در کلاس کارمند.

۸,۳,۶ مرتب بودن چندگانگی

زمانیکه چندگانگی در نقشی بیشتر از یک باشد در اینصورت مرتب‌بودن[۱۶۱] نمونه‌ها می‌تواند اهمیت داشته باشد. در شکل ۱۵ مشخص شده است که دانشجویان که در انتظار سمیناری هستند صفر و بیشتر می‌توانند باشند و چگونگی ترتیب آنها هم براساس تقدم در زمان ثبت‌نام می‌باشد.

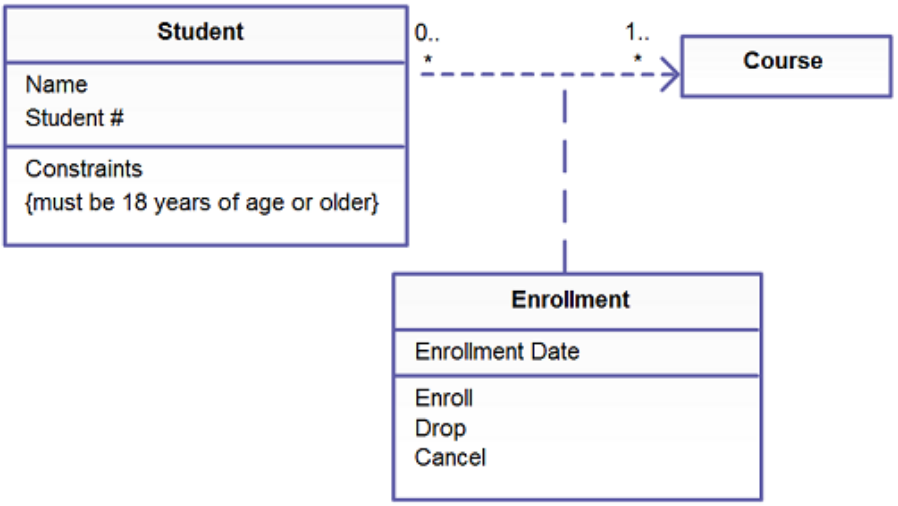


شکل ۱۵) مشخص کردن مرتب بودن در چندگانگی

۸,۳,۷ کلاس‌های وابستگی

کلاس وابستگی[۱۶۲]، گونه‌ای از وابستگی است که مشخصه‌های کلاس را نیز دارد. این نوع از وابستگی در شکل ۱۶ آورده شده است.

[161] Ordering
[162] Association class

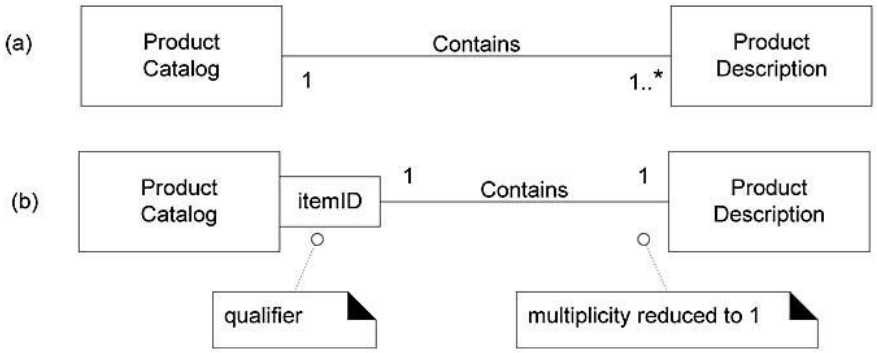


شکل ۱۶) تخصیص کلاس به وابستگی در نمودار کلاس

در این شکل هر دانشجو در یک درس یا بیشتر ثبت‌نام می‌کند ولی از ثبت‌نام ویژگی‌های دیگری مانند تاریخ ثبت‌نام نیز مطرح می‌باشند که در کلاس وابستگی متناظر آمده است.

۸,۳,۸ وابستگی مشروط

همانگونه که در شکل ۱۷ نشان داده شده است، هر گاتالوگ شامل تعدادی محصول می‌شود. اما زمانی که از محصول شناسه آن را بدانیم دیگر می‌دانیم که منظورمان کدام محصول مشخص می‌باشد. در مقایسه بخش الف و ب در شکل ۱۷، با قرارگرفتن شناسه محصول از کلاس محصول در کنار کلاس کاتالوگ، چندگانگی *..1 به 1 تبدیل می‌شود.

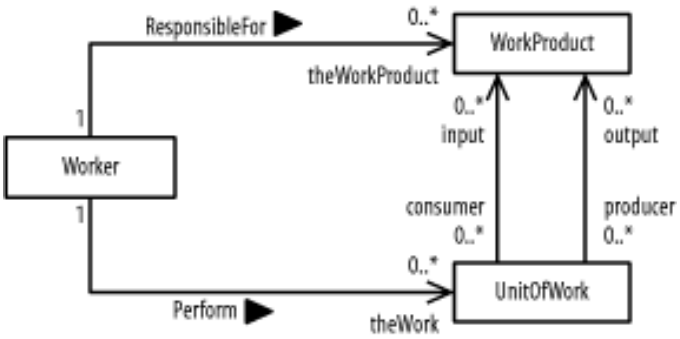


شکل ۱۷) مقایسه وابستگی و وابستگی مشروط در نمودار کلاس

۸,۳,۹ وابستگی‌های چندطرفه

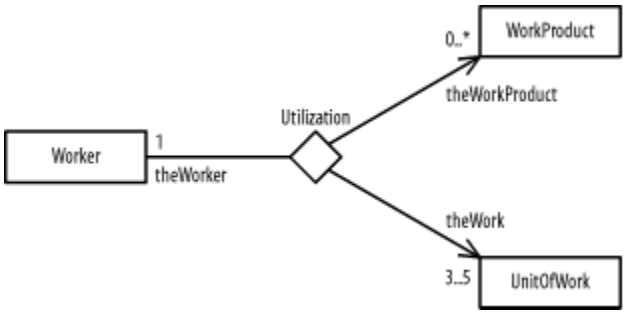
در شکل ۱۸، نمودار کلاس برای سیستم کنترل پروژه آورده شده است. فرض این سیستم بر آن است که بواسطه فعالیتی[۱۶۳]، فرآورده‌ای[۱۶۴] ایجاد می‌شود که در آنها کارگر[۱۶۵] نیز نقش دارد. ارتباط بین فرآورده و فعالیت به این صورت است که فرآورده، خروجی فعالیت می‌باشد و همچنین فرآورده می‌تواند ورودی فعالیت نیز باشد. براساس چندگانگی‌هایی که در شکل مشخص شده است، هر کارگر مسئول انجام *..0 فرآورده می‌باشد. همچنین هر کارگر *..0 فعالیت را انجام می‌دهد. از سویی دیگر هر فعالیت ایجاد کنند *..0 فرآورده و همچنین هر فعالیت مصرف کننده *..0 فرآورده نیز می‌باشد. یا به عبارتی دیگر از نمودار کلاس مشخص است که هر فرآوده ورودی *..0 فعالیت و همچنین خروجی از *..0 فعالیت می‌باشد.

[163] Unit of work
[164] Work product
[165] Worker



شکل ۱۸) نمودار کلاسی که در آن سه کلاس با هم وابستگی دارند.

در وابستگی چندطرفه[۱۶۶]، بیش از دو کلاس در وابستگی مشارکت دارند. در این نوع وابستگی می‌توان از کلاس وابستگی نیز برای مشخص کردن ویژگی‌های آن استفاده کرد. وابستگی سه‌طرفه از نمودار کلاس شکل ۱۸ در شکل ۱۹ آمده است.



شکل ۱۹ نمونه‌ای از وابستگی سه‌طرفه در نمودار کلاس

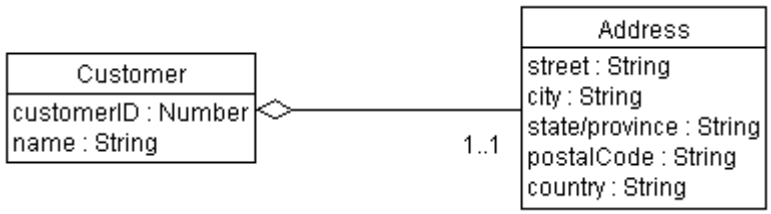
هرچند در وابستگی سه‌طرفه مفهوم انتزاعی راحت‌تر انتقال داده می‌شود، ولی تبدیل آن به وابستگی‌های دوطرفه همانگونه که در شکل ۱۸ آورده شده است به طراحی نزدیک‌تر می‌باشد.

[166] N-ary association

۸,۴ تعلق بین کلاس‌ها

از تعلق[۱۶۷] در بین کلاس‌ها زمانی استفاده می‌شود که در ارتباط بین آنها نوعی وابستگی وجود داشته باشد. مثلا کتابخانه شامل کتاب‌ها می‌شود، دپارتمانی درسازمان از کارمندانش شکل می‌گیرد و یا کامپیوتر از تعدادی قطعه شکل گرفته است.

در مثال شکل ۲۰، از رابطه تعلق بین دو کلاس مشتری و آدرس استفاده شده است. چراکه این دو کلاس عملا دربرگیرنده مفهومی بزرگتر هستند. البته به این دلیل کلاس آدرس به صورت مجزا دیده شده است که می‌خواهیم اطلاعات کاملی از آدرس را در آن نگهداری کنیم.

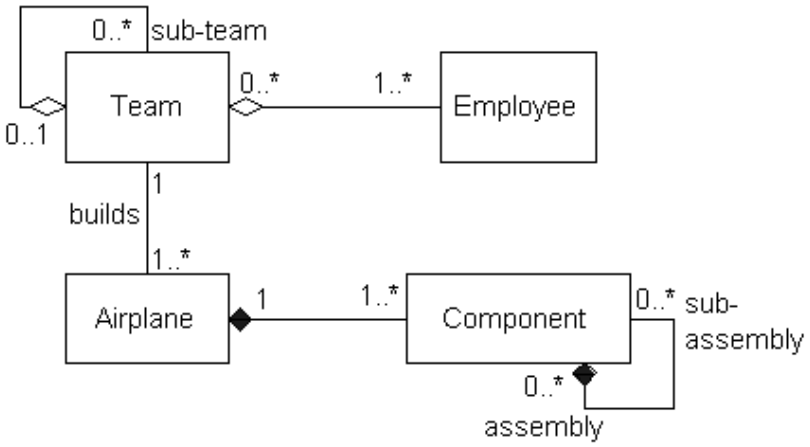


شکل ۲۰) مثالی از تعلق بین دو کلاس

۸,۴,۱ ترکیب بین کلاس‌ها

ترکیب[۱۶۸]، نوعی از تعلق است که ویژگی مالکیت در آن قوی‌تر می‌باشد. مثال‌هایی از ترکیب در شکل ۲۱ آورده شده است. تفاوت تعلق و ترکیب در نمایش آن است که لوزی آورده شده در تعلق توخالی و در ترکیب توپر می‌باشد.

[167] Aggregation
[168] Composition



شکل ۲۱) تفاوت تعلق و ترکیب در نمودار کلاس

همانگونه که در شکل ۲۱ نشان داده شده است، ارتباط بین کارمند و تیم از نوع تعلق است. هر تیم از تعدادی کارمند تشکیل شده است و عملا کارمندان به تیم تعلق دارند. همانگونه که هر تیمی از *1.. کارمند تشکیل شده است، هر کارمند می‌تواند در *0.. تیم عضو باشد. ارتباط بین تیم و هواپیما از نوع تعلق نیست. چراکه نه هواپیما در مفهوم به تیم تعلق دارد و نه تیم به هواپیما تعلق دارد. از سویی دیگر ارتباط بین هواپیما و قطعات آن از نوع ترکیب است. در ارتباطاتی که همانند این مثال از نوع ترکیب هستند هر قطعه دقیقا به یک هواپیما تعلق دارد و نمی‌تواند به هواپیمای دیگر نیز تعلق داشته باشد.

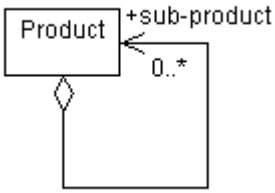
۸,۴,۲ استفاده از ترکیب برای مدل‌سازی ویژگی‌های کلاس

همانگونه که در مثال شکل ۱۸ نشان داده شده است، از رابطه‌های تعلق و ترکیب می‌توان مثلا برای مدل‌سازی ویژگی‌های آدرس از کلاس کارمند استفاده کرد. تصیم‌گیری درباره اینکه آیا از رابطه تعلق و ترکیب استفاده کنیم و یا مستقیما ویژگی‌ها را در کلاس قرار دهیم می‌تواند به موارد زیر وابسته باشد.

- آیا برای ویژگی‌ها نیاز است تا شناسه مستقلی داشته باشند. مثلا آیا می‌خواهیم هر آدرسی شناسه مستقلی داشته باشد یا خیر. درصورت مثبت بودن پاسخ از تعلق استفاده می‌کنیم.
- آیا تعدادی نمونه مثلا در کلاس کارمند هستند که آدرس یکسانی داشته باشند. درصورت مثبت بودن پاسخ از تعلق استفاده می‌کنیم.
- آیا ویژگی‌ها ساختار پیچیده‌ای دارند. در صورت مثبت بودن پاسخ از تعلق استفاده می‌شود.
- در غیر این‌صورت همان ویژگی‌ها را در کلاس اصلی استفاده می‌کنیم.

۸,۴,۳ تعلق به خود

همانگونگه که در وابستگی به خود بیان گردید، در تعلق به خود[۱۶۹] نیز قرار نیست که یک نمونه از کلاس به خود تعلق داشته باشد. بلکه، این بدان معنا است که یک نمونه از کلاس به نمونه‌ای دیگر از همان کلاس متعلق است. در شکل ۲۲ مثالی در این مورد آورده شده است.



شکل ۲۲) نمودار کلاسی که در آن تعلق به خود آورده شده است.

۸,۵ وابستگی اشتراکی بین کلاس‌ها

[169] Self-aggregation

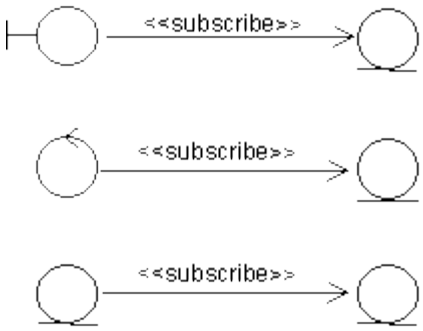
در برخی موارد، شیءای به رخداد اتفاقی در شیء دیگری وابسته است. پاره‌ای از اوقات ممکن است که این اتفاق در کلاسی کنترلی روی دهد. در این موارد کلاس کنترلی می‌تواند شیء را از رخداد مطلع سازد. ولی ممکن است که رخداد در شیءای از نوع داده‌ای (مدل) رخ دهد. در معماری‌های شیءگرایی که مبتنی بر سه لایه داده، کنترل و واسط کاربری هستند، کلاس‌های داده‌ای توسط کلاس‌های کنترلی فراخوانی می‌شوند و خود آنها مجاز به فراخوانی متدهایی از اشیاء دیگر کلاس‌ها نیستند. لذا اشیاء کلاس‌های داده‌ای نمی‌توانند با فرخوانی متدهای دیگر اشیاء، آنها را از اتفاقاتی مطلع سازند.

در مثال سیستم بانک به صورت ساده فرض کنید کاربر از طریق صفحه‌ای اطلاعات حساب خود را دارد و براساس آن برنامه‌ریزی برای پرداخت‌های بعدی را انجام می‌دهد. برای کاربر در این سیستم بسیار مهم است که چنانچه موجودی حسابش تغییر می‌کند مطلع گردد تا برنامه‌ریزی‌های آتی خود را براساس آن تغییر دهد. در این سیستم، حساب به صورت کلاسی از نوع داده‌ای مدل می‌شود که نمی‌تواند متد کلاس‌های واسط کاربری را صدا بزند تا تغییرات موجودی را از آن طریق به اطلاع کاربر برساند. از طرفی شاید گفته شود که واسط کابر می‌تواند به صورت متناوب کلاس حساب را صدا بزند تا از تغییرات آن مطلع شود. در آنالیز نمی‌توانیم تصمیم مشخصی درباره چگونگی دقیق پیاده‌سازی این نیازمندی اتخاذ کنیم. ولی می‌توانیم این نیازمندی را از طریق وابستگی اشتراکی[۱۷۰] بین دو کلاس واسط کاربر و داده‌ای بیان نماییم.

در وابستگی اشتراکی کلاسی از هر کلیشه، که می‌توانیم آن را کلاس مشترک بنامیم، را به کلاسی از نوع داده وابسته می‌کنیم. به ازای ایجاد این وابستگی،

[170] Subscribe-association

کلاس مشترک هر زمان که اتفاق مشخصی در کلاس داده‌ای اتفاق بیفتد از آن مطلع می‌شود.



شکل ۲۳) سه حالت مختلف که در آن کلاس‌ها می‌توانند از اتفاق مشخصی در کلاس داده‌ای مطلع شوند.

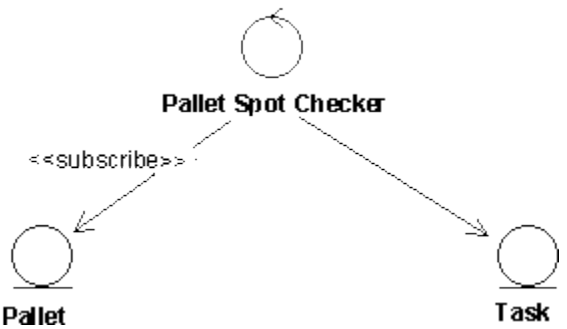
همانگونه که در جهت وابستگی در شکل ۲۳ نشان داده شده است، تنها کلاس مشترک از این وابستگی مطلع است.

در وابستگی اشتراکی، شیء‌ای که مشترک است نیاز است فقط از وقوع اتفاق مطلع گردد. بنابراین نیازی نیست تا اطاعات دیگری از اتفاق برایش ارسال گردد. بنابراین چنانچه شیء مشترک نیاز به اطلاعات تکمیلی داشته باشد، این نیازمندی را از طریق فراخوانی متدهای شیء داده‌ای به صورت عادی مرتفع می‌سازد.

برای روش‌تر شدن بحث در وابستگی اشتراکی مثال دیگری در ادامه آورده می‌شود. فرض کنید در سیستم انبار کلاسی داده‌ای برای پالت‌ها[۱۷۱] تعریف شده است. همچنین براساس قاعده کسب‌وکاری می‌بایست پالت‌ها پس از صد بار استفاده از نظر کیفی کنترل شوند. برای این منظور شمارنده‌ای برای هر شیء در کلاس پالت تعریف می‌شود که پس از هربار استفاده یکی به آن افزوده می‌شود.

[171] Pallet

سپس ارتباطی از نوع وابستگی اشتراکی بین کلاس کنترلی چک‌کننده پالت و کلاس داده‌ای پالت برقرار می‌شود. بواسطه این ارتباط، پس از اینکه در هر شیءای از نوع پالت که شمارنده تکرار آن به صد رسید، شیء کنترل پالت مطلع می‌شود و سپس شیءی از نوع کار را ایجاد می‌کند که در آن اطلاعات برای آزمایش کنترل کیفی پالت مورد نظر درج می‌شود و تا زمانیکه آزمایش کنترل کیفی پالت به اتمام نرسیده باشد، پالت اجازه فعالیت نخواهد داشت. در نمودار کلاس شکل ۲۴، آنالیز این نیازمندی در سطح نام کلاس‌ها و وابستگی آنها با هم آورده شده است.



شکل ۲۴) نمودار کلاسی که در آن کلاس کنترلی مشترک در وابستگی اشتراکی پس از اطلاع از اتفاق کار جدیدی را سازمان‌دهی می‌کند.

نوشتن شرایط در وابستگی اشتراکی باید به صورت انتزاعی باشد و به ویژگی‌های خاصی از کلاس داده‌ای وابسته نباشد.

بخش چهارم: فرآیندهای کسب‌وکار

۹ مدیریت فرآیندهای کسب‌وکار

معمولا در سازمان‌ها دو نوع فعالیت وجود دارد. نوع اول فعالیت‌ها از نوع پروژه است. ماهیت این فعالیت‌ها به گونه‌ای است که شروع و پایان مشخصی دارند و معمولا تکرار پذیر نیستند. نوع دوم فعالیت‌ها از نوع فرآیند[۱۷۲] است. فعالیت‌هایی که از نوع فرآیند هستند تکرار پذیر هستند وبه دفعات اجرا می‌شوند. مثلا در سازمانی مانند دانشگاه، فعالیتی مانند تاسیس دانشکده‌ای جدید می‌تواند وجود داشته باشد. چنانچه این فعالیت برای اولین‌بار بخواهد انجام شود، مسلما از نوع پروژه خواهد بود. چنانچه این فعالیت برای دومین بار نیز بخواهد برای ایجاد دانشکده‌ای دیگر انجام شود باز هم از نوع پروژه خواهد بود چراکه نحوه انجام کارها[۱۷۳] در این فعالیت هنوز به دفعات تکرار نشده است تا فرآیند آن مشخص شده باشد. فعالیت تاسیس دانشکده‌ای جدید در هر دانشگاه معمولا تبدیل به فرآیند نمی‌شود. چراکه معمولا به تعداد کم تکرار می‌شود. ولی همین فعالیت در سازمانی دیگر مانند وزارت علوم می‌تواند به تکرار برای دانشگاه‌های مختلف در سراسر کشور تکرار شود. بنابراین در وزارت علوم فرآیند تاسیس دانشکده می‌توانیم داشته باشیم. در دانشگاه فعالیتی دیگر مانند جذب هیات علمی مسلما از نوع فرآیند است. چراکه تمامی کارهای آن بواسطه دفعات تکرار بسیار، قابل برنامه‌ریزی می‌باشد.

شناسایی فرآیندهای هر سازمان، فعالیتی اساسی می‌باشد که آغاز هر فعالیت بهبودی در سازمان می‌باشد. پروژه‌های مختلفی در سازمان مانند تضمین کیفیت مبتنی بر ISO[۱۷۴]، تعالی سازمانی با راهکار EFQM[۱۷۵]، معماری سازمانی، ERP[۱۷۶]

---

[172] Process
[173] Task or Work
[174] International Organization for Standardization

که همگی در راستای بهبود عمکرد سازمان‌ها می‌باشند مبتنی بر شناسایی و بهبود فرآیندها می‌باشند.

۹,۱ تعریف فرآیند

آقای مایکل همر[۱۷۷] از مولفین حوزه تئوری‌های مدیریتی در فرآیندهای کسب‌وکار تعریف زیر را از فرآیند ارائه می‌دهد.

فرآیند کسب‌وکار عبارتست از تعدادی وظایف که بایکدیگر ارزش مورد نظر مشتری را فرآهم می‌کنند.

البته واژه مشتری به مفهوم عام در هر سازمان می‌باشد. مثلا در دانشگاه مشتری‌ها می‌توانند دانشجویان، کارمندان، استادان، ذینفعان دیگر مانند وزارت علوم و یا دانشگاه‌های دیگر باشد. در این مثال، فرآیندهایی مانند انتخاب واحد و یا ثبت‌نام، ارزش افزده برای دانشجویان ایجام می‌کنند. همچنین فرآیندهایی مانند ترفیع یا جذب، ارزش افزوده را برای استادان دانشگاه ایجاد می‌کند. مسلما در دانشگاه یا هر سازمان دیگر، نتیجه نهایی ارزش افزوده فرآیندها برای ذیفنعان مختلف سبب ارتقای جایگاه سازمان می‌شود.

در پروژه‌های فرآیند کسب‌وکار[۱۷۸] (BP) که در سطح سازمان تعریف می‌شود، شناسایی فرآیندها و مدل‌سازی آنها اولین و مهمترین فعالیت می‌باشد.

---

[175] European Foundation for Quality Management (EFQM)
[176] Enterprise Resource Planning (ERP)
[177] Michael Martin Hammer
[178] Business Process (BP)

نکته اساسی که در مدیریت سازمان‌ها مبتنی بر فرآیندهای کسب‌وکار وجود دارد آن است که تمامی سازمان‌ها عملا خدمات خود را بواسطه ترکیبی از فرآیندهای مختلف ارائه می‌دهند. بنابراین اصطلاحا اینگونه می‌توان بیان کرد که فرآیندهای کسب‌وکار در تمامی سازمان‌ها وجود دارد. تفاوتی که در سازمان‌ها وجود دارد آن است که برخی از آنها به این موضوع و اهمیت آن توجه بیشتری داشته‌اند و زودتر شروع به شناسایی فرآیندها، مدل‌سازی و اجرای آنها در سازمان خود نموده‌اند.

در روشی مانند شش‌سیگما[۱۷۹] که بر بهبود عمکرد سازمان‌ها مبتنی بر بهبود فرآیندهای آنها تمرکز دارد، فرآیند از منظر دیگری نیز مورد توجه قرار می‌گیرد. در شش‌سیگما برای هر فرآیند ورودی و خروجی‌های آن نیز مورد توجه می‌باشد. برهمین اساس برای هر فرآیند معیارهایی برای ورودی فرآیند و خروجی آن تعریف می‌شود. معیار ورودی[۱۸۰]، مجموعه‌ای از نیازمندی‌هایی است که قبل از آغاز فرآیند باید فراهم شده باشد. هرچند شاید فرآیندها را دقیقا نتوانیم مشابه موارد کاربری که در فصل ششم درباره آنها صحبت کردیم درنظر گیریم، ولی در موارد کاربری نیز شرایط پیش‌نیاز را برای هر مورد کاربری تعریف می‌کردیم. منطق معیار ورودی و یا شرایط پیش‌نیاز از آنجا نشات می‌گیرد که برای آنکه فرآیند یا مورد کاربری بتواند وظیفه محوله را انجام دهند، داده‌ها و اطلاعات باید شرایط خاصی داشته باشند. وگرنه کارها و سناریوهای تعریف‌شده در آنها اثربخش نخواهد بود. معیار خروجی در فرآیند را نیز می‌توان مشابه شرایط پایانی در مورد کاربری تعریف کرد. معیارهای خروجی در فرآیند، مجموعه‌ای از نیازمندی‌ها و یا معیارهایی است که می‌بایست قبل از اتمام کار در فرآیند محقق شده باشند. از

[179] Six sigma
[180] Entry criteria

این منظر هر فرآیند را می‌تواند به فرآیندهای کوچکتر تقسیم کرد که هرکدام بخشی از کار را انجام می‌دهند و از ترتیب انجام آنها فرآیند اصلی محقق می‌شود.

۹,۲ اهمیت مدیریت فرآیندهای کسب‌وکار (BPM)

اهمیت‌دادن به فرآیندها به عنوان پایه انجام کارها در سازمان، شناسایی، مدل‌سازی، اجرا و پایش مداوم آنها را می‌توان مدیریت فرآیندهای کسب‌وکار[۱۸۱] (BPM) نامید. عملا BPM فلسفه‌ای مدیریتی است که مزایای عمده‌ای را در سازمان‌ها سبب می‌شود.

الف) کاهش زمان فرآیند

مشکل عمده‌ای که در سازمان‌ها وجود دارد، طولانی بودن زمان خدمت‌رسانی یا تولید در آنها می‌باشد. معمولا سازمان‌ها وارث فرآیندهایی هستند که شکل‌دهی آنها به سال‌ها قبل برمی‌گردد. مسلما بوروکراسی در فرآیندهای فعلی به سطح اتوماسیون، قوانین، فرهنگ سازمانی و توانایی افراد دخیل در آن، در زمان شکل‌گیری فرآیند مرتبط می‌باشد. در طول سال‌ها این عوامل می‌تواند دستخوش تغییرات عمده‌ای شده باشد. لذا شناسایی مجدد فرآیندها و مدل‌سازی مجدد آنها بر اساس شرایط موجود می‌تواند سبب کاهش زمان خدمت‌رسانی در فرآیند گردد.

ب) کاهش هزینه و قیمت تمام‌شده محصولات

مسلما نگرش مجدد به فرآیندها و بازنگری مجدد آنها نه تنها می‌تواند سبب کاهش زمان فرآیند گردد بلکه کاهش هزینه را نیز دربر خواهد داشت.

ج) افزایش انعطاف پذیری

[181] Business Process Management (BPM)

در BPM بواسطه مدل‌سازی فرآیندها و درنظرگیری مسائل محیطی مختلف در حوزه انجام کار فعالیت، می‌توان به افزایش انعطاف پذیری فرآیند در شرایط مختلف رسید.

د) افزایش رضایت مشتریان

کاهش زمان، هزینه و افزایش انعطاف‌پذیری در اجرای فرآیند مسلما به افزایش رضایت مشتریان نیز منجر خواهد شد. همانگونه که در تعریف اولیه از مشتری در ابتدای این فصل بیان شد، مشتریان می‌توانند طیف وسیعی از ذینفعان خارج و داخل سازمانی را شامل گردد.

ه) بهبود بهره‌وری در سازمان

از آنجایی که بواسطه هزینه کمتر، تولید بیشتری در حوزه خدمات و کالا در سازمان اتفاق می‌افتد، می‌تواند بهبود بهره‌وری را نیز سبب شود.

و) افزایش رضایت کارکنان

از آنجایی که فرآیندهایی که اجرای آنها فرآهم آوری خدمات برای کارکنان است نیز در BPM مورد توجه قرار می‌گیرد، لذا افزایش رضایت کارکنان و در نتیجه خدمت‌دهی بهتر به مشتریان خارج از سازمان، نیز از مزایای اجرای مدیریت فرآیندهای کسب‌وکار می‌باشد.

ز) افزایش بهره‌وری کارکنان

از آنجایی که BPM همواره مبتنی بر سیستمی نرم‌افزاری است، لذا فعالیت‌هایی که کارکنان در فرآیندهای مختلف انجام می‌دهند کاملا رصد و

کنترل می‌شود. بنابراین کارکنان به میزان کاری که انجام می‌دهند قابل شناسایی هستند و این امر سبب افزایش روحیه بهره‌وری در کارکنان می‌شود.

ح) افزایش رقابت‌پذیری

سازمان‌ها برای رقابت با هم و جذب مشتریان بیشتر نیازمند بهبود مداوم در فرآیندهای کاری هستند. اجرای BPM سبب می‌شود تا فرآیند توجه اصلی تمامی دیدگاه‌ها در سازمان باشد. لذا از آنجایی که تغییر اجرای فرآیند می‌تواند سریع اتفاق بیفتد، لذا قابلیت انعطاف برای افزایش رقابت را خواهد داشت.

۹,۳ انواع ساختار سازمانی

مراجع مختلفی تقسیم‌بندی‌های متفاوتی از ساختار سازمانی ارائه کرده‌اند. در بین تقسیم‌بندی‌های موجود دو ساختار عمده مبتنی بر وظیفه و مبتنی بر فرآیند را در این بخش مرور می‌کنیم.

سازمان‌هایی که ساختاری وظیفه‌ای[۱۸۲] دارند، هر بخش از سازمان وظیفه انجام کار مشخصی را دارد. مثلا چنانچه دانشگاه را به عنوانی سازمانی نمونه درنظر بگیریم دارای بخش‌های مختلفی مانند، آموزشی، پژوهشی، منابع انسانی، امور دانشجویی، مالی و غیره می‌باشد. مسلما بخش‌های سازمانی، ساختاری درختی دارند که مجدد هر بخش خود می‌تواند به بخش‌های کوچکتر تقسیم شود. در سازمان‌هایی که ساختاری وظیفه‌ای دارند انجام خدمات و یا تولید نیازمند همکاری بخش‌های مختلف می‌باشد. از آنجایی که در این سازمان‌ها هر بخش یا زیربخش، مدیریت خاص خود را دارد و ارتباط بین بخش‌های مختلف حتما باید از طریق مکاتبات اداری میان سلسله‌مراتب مدیریت‌های مختلف انجام شود، لذا

[182] Functional organization

انجام کاری که نیازمند همکاری میان بخش‌های مختلف باشد معمولا زمان‌گیر می‌باشد. چراکه بوروکراسی حاکم براین ساختار سازمان، ذاتا سبب کندی انجام فرآیندها می‌شود. در این سازمان‌ها معمولا برقراری ارتباط بین سلسله مراتب مختلف حجم عمده کار در سازمان را به خود اختصاص می‌دهد که برای مشتری ارزش افزوده‌ای ندارد.

از معایب سازمان‌های وظیفه‌ای می‌توان به موارد زیر اشاره کرد.

۱) مانع نوآوری و خلاقیت در کارکنان هستند.
۲) مشتری مداری در این سازمان‌ها در اولویت نیست. بلکه هر واحد سازمانی سعی می‌کند کار خود در به بهترین وجه انجام دهد.
۳) رقابتی بین واحدهای مختلف وجود ندارد. چراکه عملا هر واحد سازمانی کار خود را انجام می‌دهد و هیچ مقایسه‌ای بین واحدهای مختلف از منظر خدمت‌رسانی بیشتر به مشتریان صورت نمی‌پذیرد.
۴) میان کارکنان علاقه‌ای به نتیجه نهایی کارشان وجود ندارد. چراکه هر کس کاری در حوزه مشخص انجام می‌دهد و نهایت انجام کار برایش حساسیت برانگیز نمی‌باشد.
۵) هزینه‌های غیر مستقیم در کار بواسطه پررنگ بودن ارتباطات بین واحدهای سازمانی افزایش می‌یابد. لذا بهره‌وری در کار افزایش می‌یابد.

لذا نیاز به حرکت به سازمان فرآیند محور بواسطه معایب سازمان‌های وظیفه‌محور وجود دارد.

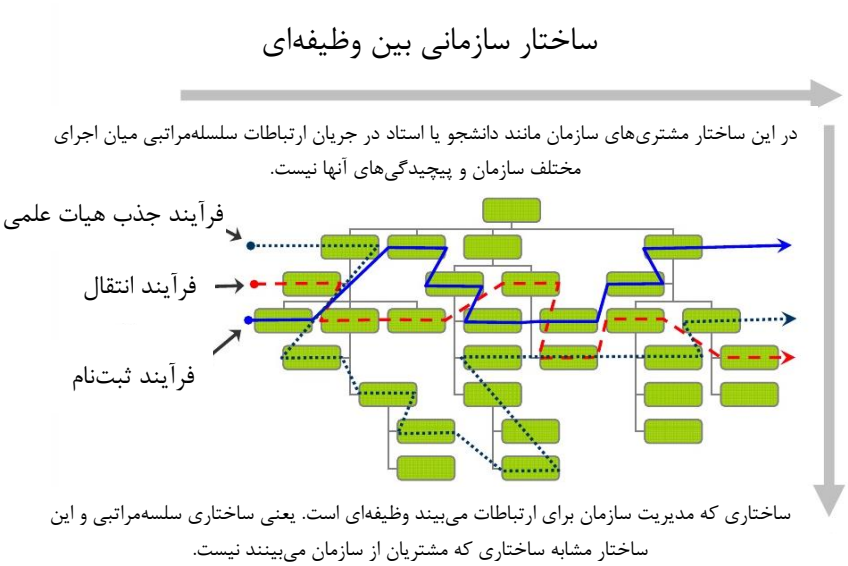
البته در ساختارهای سازمانی نمی‌توان ادعا کرد که سازمانی می‌تواند کاملا فرآیندمحور باشد. چراکه فرآیندمحوری کامل در سازمان‌ها نیز معایب خاص خود

را دارد. مسلما باید در سازمان بخش‌هایی به صورت تخصصی کارهایی را انجام دهند و مدیران آن بخش‌ها یا واحدها مسئولیت کیفیت و دقت انجام آن کار تخصصی را به عهده داشته باشند.

فرآیندهایی که در سازمان وجود دارند معمولا بین وظیفه‌ای[۱۸۳] می‌باشند. یعنی برای اجرا و تکمیل فرآیند باید افراد مختلف از بخش‌های مختلف سازمان با هم همکاری داشته باشند. مثلا برای اینکه فرآیند فارغ‌اتحصیلی در دانشگاه تکمیل شود، نیازمند انجام کارهای تخصصی در بخش‌های مختلف دانشگاه مانند مالی، آموزش، پژوهش، امور خوابگاه‌ها و غیره می‌باشد. اینگونه تعریف فرآیندها به صورت بین واحدی سبب می‌شود تا انجام کارها بسیار تسهیل گردد. بنابراین در پروژه‌های BPM که در سطح سازمان‌ها تعریف می‌شود، هدف تغییر ساختار سازمانی نمی‌باشد. بلکه هدف آن است تا با ساختار وظیفه‌ای موجود بهترین گردش کار برای انجام فرآیندها تعریف شود.

شکل ۲۵ به خوبی مزیت و چگونگی ساختار بین وظیفه‌ای را در سازمان‌ها نشان می‌دهد.

[183] Cross-functional



شکل ۲۵) ساختار سازمانی بین وظیفه‌ای حاصل تلفیق ساختار وظیفه‌ای و ساختار فرآیندی می‌باشد.

۹,۴ استانداردهای مدل‌سازی فرآیند

برای مدل‌سازی فرآیند، استانداردهای مختلفی وجود دارد. از جمله مهمترین آنها می‌توان به BPMN[۱۸۴]، نمودار جریان‌کار[۱۸۵] و UML[۱۸۶] اشاره کرد.

BPMN را می‌توان بهترین استاندارد در مدل‌سازی فرآیند نامید. چراکه نمادهای بسیارگوناگونی برای بیان اجزای مختلف فرآیند در آن پیش‌بینی شده است.

[184] Business Process Modelling Notation (BPMN)
[185] Flow Chart
[186] Unified Modelling Language (UML)

فاز اول پروژه‌های BPM، مدل‌سازی می‌باشد. بنابراین BPMN از ابتدا در پروژه‌های BPM مورد توجه می‌باشد.

۹,۵ سیستم مدیریت فرآیند کسب‌وکار (BPMS)

سیستم مدیریت فرآیند کسب‌وکار (BPMS[۱۸۷]) فصل مشترک مباحث مدیریتی (BPM) و فناوری اطلاعات (IT) می‌باشد. کار BPMS آن است که زیرساختی نرم‌افزاری فراهم آورد که از طریق آن بتوان فرآیندهای سازمان را مکانیزه تعریف و مدیریت کرد.

البته از طریق روش‌های معمول برنامه‌نویسی مانند معماری شیء گرایی نیز می‌توان فرآیندهای سازمان را مکانیزه کرد. ولی پروژه‌هایی که با این رویکرد تعریف می‌شوند بسیار زمان‌بر هستند. همچنین فرآیندهای سازمان ذاتا مستعد تغیرات هستند و نمی‌توان انتظار داشت در مدتی که تیم توسعه درحال برنامه‌نویسی و توسعه نرم‌افزار است بدون تغییر بماند. این واقعیت سبب می‌شود تا استفاده از روش‌های معمول برنامه‌نویسی بواسطه زمان‌بر بودنشان نتوانند پاسخگوی سیستم‌های مدیریت فرآیند کسب‌وکار در سازمان‌ها باشد.

هدف از BPMS آن است تا در زمان بسیار کمتری نسبت به شیوه‌های رایج توسعه نرم‌افزار، نرم‌افزار BPM در سازمان اجرایی گردد.

باید به این نکته توجه داشت که بدون BPMS نیز فرآیندها می‌توانند در سازمان به صورت دستی اجرا شوند. ولی استفاده از BPMS به سرعت اجرای فرآیند، افزایش کیفیت آن در اجرا و مدیریت بهتر منابع در اجرای فرآیند کمک می‌کند.

[187] Business Process Management System (BPMS)

نرم‌افزارهایی مانند پراسس‌میکر[۱۸۸] و BizAgi در این حوزه قرار می‌گیرند. در این نرم‌افزارها فرآیند با BPMN تعریف می‌شود و با بهره‌گیری از ابزار به BPMS تبدیل می‌شود.

۹,۶ ریسک‌های پروژه و شناسایی آنها در پروژه‌های BPM

شناسایی ریسک‌ها[۱۸۹] یا خطراتی که هر پروژه نرم‌افزاری با آن می‌تواند درگیر درگیر شود یکی از مهمترین وظایف مدیر پروژه می‌باشد. در ادامه این بخش ابتدا به تعریف مفهوم ریسک به صورت عمومی می‌پردازیم. سپس استراتژی‌های مختلفی که در برخورد با ریسک می‌تواند وجود داشته باشد را مرور می‌کنیم. در نهایت مهم‌ترین ریسک‌هایی که در پروژه‌های BPM وجود دارند را بررسی و تحلیل می‌شوند.

۹,۶,۱ انواع ریسک

در نگاه اول ریسک‌ها را می‌توان به دو دسته مستقیم[۱۹۰] و غیرمستقیم[۱۹۱] تقسیم‌بندی کرد. ریسک‌های مستقیم، ریسک‌هایی هستند که ذی‌نفعان پروژه و مدیر پروژه می‌توانند کنترل بالایی بر روی آنها داشته باشند. عمده ریسک‌هایی که در پروژه‌های نرم‌افزاری مطرح هستند از این دسته می‌باشند. ریسک‌های غیر مستقیم ریسک‌هایی هستند که تیم پروژه یا کنترلی بر آنها ندارد و یا کنترل

[188] ProcessMaker

[189] Risk

[190] Direct

[191] Indirect

بسیار کمی بر آنها دارد. مثلا ریسک‌هایی که در ارتباط با مسائلی مانند جنگ یا تحریم ممکن است در پروژه مطرح شود را می‌توان در این خانواده قرار داد.

ریسک‌ها از دو جنبه احتمال وقوع[۱۹۲] و میزان بدی[۱۹۳] در پروژه اهمیت دارند. بنابراین برای هر ریسک باید بررسی کنیم که با چه احتمالی امکان دارد در پروژه پدیدار شود. همچنین باید آنالیز شود که ریسک در صورت وقوع چه میزان اثرات منفی در پروژه برجای خواهد گذاشت.

۹,۶,۲ استراتژی‌های برخورد با ریسک

اصل اساسی در مدیریت ریسک آن است که نباید به صورت غیرفعال[۱۹۴] منتظر باشیم تا ریسک تبدیل به مساله در پروژه گردد.

سه استراتژی کلی در مدیریت ریسک وجود دارد که در ادامه به بیان آنها می‌پردازیم.

الف) اجتناب از ریسک[۱۹۵]

در این استراتژی، سازمان‌دهی پروژه را به گونه‌ای تغییر می‌دهیم تا دیگر در معرض ریسک قرار نگیرد.

---

[192] Likelihood

[193] Severity

[194] Passive

[195] Risk avoidance

مثلا یکی از ریسک‌های عمده‌ای که وجود دارد، از دست دادن برنامه‌نویس‌ها در پروژه می‌باشد. شغل برنامه‌نویسی به واسطه ماهیت و تنوع بازاری که دارد، معمولا از بازار کار خوبی برخوردار است. بنابراین همواره این امکان وجود دارد که یکی از برنامه‌نویس‌ها تیم را ترک کند. هرچند شرکت‌ها می‌توانند راهکارهای مختلفی برای این ریسک درنظر بگیرند، ولی معمولا یکی از راهکارهایی که در متدولوژی‌های نرم‌افزاری مطرح می‌شود، برنامه‌نویسی جفتی[۱۹۶] می‌باشد. در برنامه‌نویسی جفتی معمولا برنامه‌نویس‌ها با مشارکت هم، کدهای نرم‌افزار را تولید می‌کنند. این شیوه هرچند سبب افزایش هزینه‌های پروژه می‌شود ولی مزایای متعددی دارد. یکی از مزایای برنامه‌نویسی جفتی، اجتناب از ریسک ترک برنامه‌نویسان است. با خروج یکی از برنامنویسان پروژه، افراد دیگری می‌توانند وظیفه او را بر عهده بگیرند و مدیر پروژه فرصت خواهد داشت تا فردی جایگزین پیدا کند.

ب) انتقال ریسک[۱۹۷]

در این استراتژی، سازمان پروژه به گونه‌ای بازنگری می‌شود تا مخاطرات ریسک در صورت وقوع به یکی دیگر از ذینفعان پروژه منتقل شود.

مثلا پروژه‌های نرم‌افزاری همواره نیازمند خرید سخت‌افزار نیز می‌باشند. یکی از چالش‌های این فعالیت در کشور ما، تغییرات نرخ ارزهای خارجی و احتمالا شرایط تحریمی می‌باشد. می‌توان فعالیت‌هایی از این قبیل را در قرارداد برعهده کارفرما قرار داد.

---

[196] Pair programming

[197] Risk transfer

برای شرکت‌های نوپا معمولا هزینه اجاره محل کار برای تیم پروژه، بخش قابل‌توجهی از هزینه‌های پروژه است. هرچند این بخش از هزینه در مبلغ قرارداد نیز پیش‌بینی می‌شود ولی همواره این امکان وجود دارد که به دلایل مختلف، مدت زمان اجرای قرارداد طولانی‌تر از زمان‌پیش‌بینی شده گردد. در این صورت اجاره محل کار تیم پروژه می‌تواند سبب چالش عمده‌ای برای پیمانکار شود. بنابراین مدیر پروژه می‌تواند با پیش‌بینی این ریسک، وظیفه تامین محل کار تیم پروژه را برعهده کارفرما بگذارد. این رویکرد علاوه‌براینکه مزایای عمده‌ای در کیفیت پروژه دارد، می‌تواند سبب انتقال این ریسک از پیمانکار شود.

ج) پذیرش ریسک[۱۹۸]

در این استراتژی فرض می‌کنیم که همواره ریسک در پروژه وجود دارد و عملا بواسطه ماهیت پروژه نرم‌افزاری، راه خروجی برای فرار از آن وجود ندارد. بنابراین مدیر پروژه تمام تلاش خود را بر مراقبت[۱۹۹] از ریسک و دریافت علائم[۲۰۰] آن معطوف می‌کند. در این شیوه، همواره مدیر پروژه در زمان‌بندی و تخصیص منابع پروژه مراقبت از ریسک را مورد توجه قرار می‌دهد.

مثلا عدم شناخت صحیح نیازمندی‌های کلیه ذی‌نفعان پروژه، همواره از ریسک‌هایی است که چاره‌ای جز پذیرش آن توسط مدیر پروژه نیست. لذا مدیر پروژه با انتخاب متدولوژی متناسب با ساختار پروژه و نیز با بهره‌گیری از مشاوران متخصص در حوزه مورد نظر، سعی می‌کند برنامه‌ریزی فعالیت‌ها را در پروژه به گونه‌ای انجام دهد که این ریسک کمترین اثر منفی را در پروژه داشه باشد.

---

[198] Risk acceptance

[199] Monitoring

[200] Symptoms

۹,۶,۳ شایع‌ترین ریسک‌ها در پروژه‌های BPM

پروژه‌های BMP نیز به مانند سایر پروژه‌های سازمانی، که در آن نیازمند عزم مدیر سازمان برای بهبود و نیز انگیزه کارکنان برای رسیدن به جایگاهی بهتر در تولید و خدمات می‌باشد، درگیر ریسک‌هایی عمده می‌باشد. می‌توان ریسک‌های عمده در پروژه‌های BPM را به شرح زیر عنوان کرد.

الف) عدم مشارکت فعال کارکنان سازمان در پروژه های BPM

اجرا شدن فرآیندی جدید در هر سازمان نیازمند مشارکت فعال تمامی افراد سازمان می‌شود. هرچند عزم مدیریت برای پروژه‌های BPM شرطی اساسی و پایه‌ای می‌باشد، ولی اجرای فرآیند نیازمند همکاری جدی افراد سازمان می‌باشد.

چنانچه افراد سازمان از ابتدا در جریان پروژه و هدف از اجرای آن و اثربخشی که در شرایط سازمان خواهد داشت قرار گیرند و در ادامه در شناسایی و مدل‌سازی فرآیندها مشارکت داشته باشند، در زمان اجرایی شدن فرآیند نیز حامی پروژه خواهند بود.

ب) نگاه مقطعی به پروژه‌های BPM

شاید بتوان فعالیت‌های BPM در سازمان را از زمان عقد قرارداد با پیمانکار تا زمان اتوماسیون فرآیندهای اصلی سازمان را به صورت پروژه درنظر گرفت. ولی نمی‌توان رسیدن به اهداف BPM را در قالب پروژه‌ای مشخص تضمین کرد. هرچند پروژه می‌تواند یکسری آموزش‌ها و بسترهای تکنیکی را به همراه داشته باشد ولی رسیدن به مزایای BPM جز با مستمرسازی فعالیت‌های آن امکان‌پذیر نمی‌باشد. بنابراین باید منابع لازم برای استمرار جریان BPM در سازمان نهادینه گردد.

ج) پرداختن به همه فرایندها به صورت ناقص

در هر سازمان، فرآیندهای گوناگونی وجود دارد که اجرای مناسب هرکدام سبب ارزش افزود برای مشتریان سازمان و به تبع آن سودآوری برای صاحبان سازمان را در پی خواهد داشت.

می‌بایست در ابتدای فعالیت‌های BPM، فرآیندهای سازمان شناسایی و بر اساس اثربخشی اولویت‌بندی شوند. در ادامه باید برای شروع BPM، یکی از فرآیندهای سازمان به صورت پایلوت اجرایی شود.

اگر به اشتباه از همان ابتدا بخواهیم همزمان بر تمامی فرآیندها متمرکز شویم، به سبب بروز ریسک‌های متعدد و سختی کار موفق نخواهیم شد هیچ‌کدام را موثر اجرایی کنیم. لذا پس از مدتی با کاهش انگیزه‌ها در سازمان، بواسطه نرسیدن به نتایجی ملموس، چیزی جز تعدادی نمودار، خروجی کار نخواهد بود.

مراجع

[۱] محمدعلی زارع چاهوکی، مدیریت و نظارت بر پروژه‌های نرم‌افزاری، انتشارات دانشگاه یزد، در مرحله انتشار.

[2] P. Kruchten, An Introduction to Rational Unified Process, Third Edition, Addison Wesley.

[3] Ratioal Unified Process, Version 2003.06.00.65, Copyright 1987 - 2003, Rational Software Corporation.

[4] A. Cockburn, Writing Effective Use Cases, Addison Wesley, 2001.

[۴] محمدعلی زارع چاهوکی، مروری بر متدولوژی‌های چابک در مهندسی نرم‌افزار، در درست تهیه.

[5] http://bpmtraining.net seen at 11 Dec. 2016

[6] J. Bach, Exploratory testing explained, 2003.