مهندسی نیازمندی‌ها خانم دکتر سپیده آدابی

علیرضا سلطانی نشان

۲۳ اردیبهشت ۱۴۰۳

فهرست مطالب

مجوز

۱ مقدمه

۱.۱ مهندسی نیازمندی

۱.۱.۱ تعریف

طبق تعریف کتاب پرسمن، نیازمندی‌ها تنها ثابت در حال تغییر می‌باشند. مهندسی نیازمندی مهم‌ترین فاز انجام هر کاری در مهندسی نرم‌افزار می‌باشد. زیرا مشتری دائماً در حال تغییر درخواست‌های خودش است به همین خاطر نیازمندی‌های برآورد شده ملزوم به بروز شدن هستند. هر تغییری که صورت می‌گیرد به دلیل ماهیت پیچیده نرم‌افزار بایستی پایدار [۱] باشد. پایداری به منظور بررسی تغییرات از جوانب مختلف مانند امنیت و آزمون عملکرد صحیح می‌باشد. نیازمندی‌ها کاملا پر دردسر هستند زیرا خیلی از دلایل شکست پروژه‌ها عدم بررسی نیازمندها بوده است. درست است که با آزمون و خطا تجربه به دست می‌آید ولی این تجربه‌ها در پروژه‌های مقیاس بزرگ می‌تواند خطر آفرین باشد چرا که خود تجربه‌ها نیز نیازمند بررسی و آزمون هستند که بتوانیم از آنها در پروژه‌های بعدی یا فعلی خود استفاده کنیم.

دو کلمه اصلی در مهندسی نیازمندی‌ها وجود دارد:

۱. کلمه چه چیزی [۲]: دقیقاً آن چیزی است که سیستم بایستی قادر به انجام آن باشد. مثلاً کاربر باید بتواند در نرم‌افزار لاگین کند.

۲. کلمه چطور [۳]: همانطور که از نامش پیداست چطور انجام شدن کار را تعریف می‌کند. برای مثال بالا می‌توان گفت سیستم لاگین باید کاملاً امن باشد. در این سیستم لاگین کاربران مختلف اعم از استاد، دانشجو و رئیس دانشگاه باید بتوانند زیر پنج ثانیه احراز هویت انجام دهند.

۳. کلمه چه کسی [۴]: عوامل محیطی (افراد، دستگاه‌ها، نرم‌افزارهای آماده) دخیل در برنامه

زمانی که می‌گوییم نرم‌افزار ثبت نام درس، دقیقا بالاترین سطح تجرید [۵] را در نیازمندی بیان کرده‌ایم.

نکات

- مفاهیم کیفی به اندازه مفاهیم اجرایی مهم هستند. درست است نرم‌افزار باید اجرا شود اما این اجرا شدن باید صحیح باشد. امنیت نرم‌افزار خود خواسته می‌تواند تخریب شود، یعنی نرم‌افزاری نوشته می‌شود که می‌تواند ورودی‌های اشتباه و نادرست را بپذیرد، پس در این صورت امنیت و کارایی درست را زیر سوال می‌برد.
- سوال چه چیزی به صورت عملیاتی است و سوال چگونه به صورت غیر عملیاتی
- همیشه باید بین مسائلی که در مهندسی نرم‌افزار پیش می‌آید یک سبک سنگینی [۶] صورت گیرد. معمولاً Benchmarks ها به ما این امکان را می‌دهند. یعنی نرم‌افزار می‌تواند به چند شکل مختلف توسعه پیدا کند اما با گرفتن Benchmark ها می‌توانیم بررسی کنیم که کدام یک از آنها در قسمت عملیاتی و عملکرد صحیح بهتر بوده‌اند. به عبارت دیگر، روش‌ها را نمی‌توان بدون بررسی و با میل شخصی انتخاب کرد، بلکه باید روش‌ها بررسی و سبک سنگین شوند.

---

[۱] Stable
[۲] What
[۳] How
[۴] Who
[۵] Abstract
[۶] Trade off

- فرایندها در مهندسی نیازمندی را process گویند
- توضیح و بازنویسی نیازمندی‌ها، کار پایه مهندس نیازمندی است.
- تمام مراحل در فرایند به یکدیگر وابسته می‌باشند، فرایند اساساً در مورد جزئیات صحبت نمی‌کند بلکه به ماهیت کلی و تجرید می‌پردازد. برای مثال فرایند جمع‌آوری داده و تحلیل و دیگر مراحل کاملاً به صورت مرحله‌ای و بازگشت پذیر می‌باشد. خروجی فرایند بعد از طی کردن تمام مراحل، نیازمندی را مشخص می‌کند.
- هیچ وقت فرایند با نیازمندی‌ها هم ارز نیست، بلکه نیازمندی خروجی فرایند می‌باشد. در حقیقت به خروجی فرایند، سند نیازمندی یا Requirement Document (RD) می‌گویند.
- در فرایند تکینک‌ها و استانداردها دیده می‌شود.

۲.۱ نکته تجرید

هر موقع در مورد تجرید صحبت شد، در واقعیت امر میزان سطح پرداختن به جزئیات را توضیح می‌دهد.

۳.۱ متدولوژی

متدولوزی [۷] یک جهان‌بینی کلی، در تولید نرم‌افزار است (دید از بالا برای انجام کارها و وظایف). تمام متدولوزی‌ها را برای تولید استفاده می‌کنند و تمام راهنمایی‌ها توضیحات دارند. در حقیقت تمام متدولوژی‌ها از خواستگاه تولید نرم‌افزار ایجاده شده‌اند و حتی می‌شوند. نکته مهم آن است که فرایندها درون متدولوژی‌ها هستند. متدولوژی یک نقشه است که آن را معمار نرم‌افزار با دیدگاه کاملاً جامع انتخاب می‌کند.

۴.۱ دلیل متدولوژی‌های مختلف

ماهیت و ذات پروژه‌ها متفاوت و پیچده است، پس در این جهت متدولوزی‌های مختلفی برای مهار آنها ارائه شده است که توع تولید را متفاوت می‌کند. متدولوژی بایستی کاملا منعطف باشد. مراحل و فرایندها در متدولوژی‌ها متغیر می‌باشد.

۵.۱ ماهیت مدل

انسان همیشه با خواندن مشکل دارد. خواندن دائماً با مشکلات محاوره‌ای همراه است. محاوره با ابهام همراه است. در پروژه مهندسی نرم‌افزار، وقتی افراد بخواهند با یکدیگر در مورد پروژه صحبت کنند، زبان میان آنها مدل‌های بصری و گرافیکی می‌باشد. افراد بعد از جمع‌آوری اطلاعات و تحلیل آنها، بایستی با آنها به مفهوم بصری برسند تا به کارشناسان دیگر آن را انتقال دهند. به بیانی دیگر، مدل زبان مشترک برای انجام فرایندها، بیان گرافیکی با حفظ سطح تجرید است.

انسان روی جمله‌های ترکیبی مشکل دارد:

$$(A \wedge B) \vee (C) \rightarrow x \quad (۱)$$

یا

$$A \wedge (B \vee C) \rightarrow x \quad (۲)$$

راهکار: استفاده از Decision table که بتوان منطقی به نتیجه رسید.

زبان مدلسازی: ریاضی و گرافیک (بصری)

[۷] Methodology

عملیات به دو دسته تقسیم می‌شوند

۱. عملیات ریاضی: $y = x$

۲. عملیات بصری: نمودارها و مختصات

نکات

- تجرید میزان پرداختن به جزئیات است
- سطح تجرید نسبت به هر کلاس و مدل‌های مختلف* متفاوت است
- خروجی هر فاز فرایند در متدولوژی مدل می‌شود. در حقیقت در متدولوژی مشخص می‌شود که مدل بخش مورد نظر به چه شکلی باشد.
- از آنجایی که زبان بین انسان و ماشین زبان برنامه نویسی (کامپایلر و گرامر) می‌باشد، زبان بین افراد برای نمایش بصیری نتیجه فرایندها مدل می‌باشد.
- عملیات ریاضی صرفاً محاسباتی نیستند، بلکه می‌توانند در قسمت آنالیز هم بررسی و انجام شوند

۶.۱ الگو

الگو، راهنمایی برای حل مسائل مشابه می‌باشد. مشابه بودن مسائل به دلیل پر تکرار بودن آنها در پروژه‌های مختلف است.

۷.۱ استاندارد

مجموعه‌ای از قواعد [۸] یا دستورات است. اجرای دستور ما را به خواسته می‌رساند. مانند تمام Rule هایی که روی فایروال شبکه اعمال می‌شوند. یا اینکه یکسری قواعد محیطی را بیان می‌کند.

۸.۱ مهندسی نیازمندی

مهندسی نیازمندی یعنی مدلی که همه روی آن توافق دارند. یکسری حساب و کتاب، استاندارد .مدل‌ها و غیره که خوش تعریف هستند بدون هیچ‌گونه ابهام، مطرح می‌شوند.

۹.۱ دلیل استفاده از زبان UML

در مهندسی نیازمندی زبان مشترک بین تیم توسعه و طراحی با مشتری (کسی که درخواست دارد) زبان UML است. زبان درخواست کننده محاوره‌ای است و می‌تواند از آن هر برداشتی داشت.

۱۰.۱ بررسی شروع کار مهندسی نیازمندی

۱۱.۱ بررسی UML to goal

قبل از انجام هر کاری بایستی اقدامات مهمی در شروع مهندسی صورت گیرد. تهیه نمودارهایی که با یکدیگر ارتباط مهمی دارند و لازمه ورود به بخش طراحی معماری نرم‌افزار است.

[۸] Rules

۱.۱۱.۱ نمودار هدف

اولین نموداری که در مهندسی باید کشیده شود نمودار هدف [۹] است. اهداف در نهایت به نیازمندی‌هایی می‌رسد که قرار است در سیستم محقق شود. بیان نیازمندی یعنی بیان اهداف.

۲.۱۱.۱ نمودار ریسک

رسیک‌ها اتفاقات محیطی هستند که باید اقداماتی نسبت به آن‌ها در سیستم پیاده شود. مانند برقرار امنیت یا مشکلات کند بودن سرویس‌دهی مربوط به لود بالانسینگ. آن مواردی که به عنوان ریسک در اهداف پیدا می‌شود هم نیازمند کشیدن نمودار ریسک است.

۳.۱۱.۱ نمودار Agent

برخی از اقدامات توسط نرم‌افزار انجام می‌شود و برخی دیگر توسط کاربر (عامل). برخی از اهداف ممکن است به یکسری قابلیت‌های محیطی مربوط شوند. یعنی نرم‌افزار هیچ قوه تحلیلی برای مشتری ندارد بلکه مشتری است که با دخالت خود می‌تواند به هدف مورد نظر برسد. عامل کسی است که تعیین میکند قرار است چه عملیاتی رخ دهد.

۱۲.۱ مهندسی نرم‌افزار و مهندسی نیازمندی

در مهندسی نرم‌افزار مجموعه‌ای از ترتیب‌های [۱۰] مخصوص به آن وجود دارد مانند:

۱. مدیر پروژه Project manager

۲. مالک پروژه Product owner

۳. بخش‌های زیرساختی مانند زیرساخت شبکه و پشتیبانی و سرویس

۴. بخش پیاده‌سازی Implementation

۵. بخش بررسی استانداردها و متدولوژی‌ها

۶. بخش مستندات Documentation

۷. بخش آزمون Test

مهندسی نیازمندی یکی از زیر بخش‌های مهم مهندسی نرم‌افزار است.

۱۳.۱ مهندسی نیازمندی و مدیریت نیازمندی

مهندسی کلمه‌ای است که داشتن یک فرایند مرحله به مرحله را الزام‌آور می‌کند. یعنی برای مهندسی یک پروژه نرم‌افزاری باید تمام جنبه‌های نرم‌افزاری به همراه ابزارها را بشناسیم که با صحیح و خطا و آزمایش موجب تولید یک محصول نهایی نشویم.
برای مثال فرایند مهندسی نیازمندی چهار مرحله‌ای زیر:

۱. جمع‌آوری نیازمندی‌ها

۲. تمیز کردن داده‌ها و معنادار کردن آنها

۳. بیان زبان برای مطرح کردن داده‌ها

[۹] Goal diagram
[۱۰] Discipline

۴. صحت‌سنجی و اعتبارسنجی کارها

مدیریت یعنی توزیع منابع. این منابع می‌تواند زمان، نیروی انسانی و ارزش‌های مالی مانند پول و غیره باشد. مدیریت نیازمندی شامل مجموعه‌ای از ترتیب‌ها و توضیحات است که بیشتر به مدیریت پروژه مربوط می‌شود. مدیر پروژه سهم بین هر بخش از توسعه را تقسیم می‌کند. وظیفه مدیر نیازمندی، تقسیم وظایف به زیر عوامل است، اینکه بتواند منابع اصلی را بین افراد و زیر بخش‌های خود (مفهوم چتری) تقسیم کند.

فعالیت اصلی زیر بخش مدیریت نیازمندی، مهندسی نیازمندی‌ها می‌باشد.

۲ فصل اول

۱.۲ اصطلاحات

۱.۱.۲ World problem یا Environment

دنیای مسئله جایی است که مشکلی در آن رخ داده است و کسی وجود دارد که این مشکل را در ابتدا بررسی و بعد از آن حل می‌کند. در حقیقت دنیا، محیط عملیاتی ما در مهندسی نیازمندی است. این دنیا می‌تواند سینما باشد یا دانشگاه. جنس این مسائل می‌تواند مشکل باشد که بایستی برطرف شود یا قابلیتی که می‌خواهیم در آینده اتفاق بیوفتد.

۲.۱.۲ Machine

ماشین راه‌حلی برای حل مسئله‌ای می‌باشد که پیش آمده است. ماشین می‌تواند به صورت آماده خریداری شود یا توسط تیم توسعه از صفر توسعه داده شود. ما باید در سند نیازمندی این نوع از نیازمندی را مشخص کنیم. ماشین در حقیقت نرم‌افزاری است که قرار است داشته باشیم [۱۱]. مدیر نیازمندی با توجه به هزینه می‌تواند برای مهندس نیازمندی تعیین کند که آیا داشتن نرم‌افزار آماده هزینه کمتری برایش دارد یا توسعه آن نرم‌افزار از صفر توسط تیم توسعه خود.

۳.۱.۲ Context

کلمه Context به معنای زمینه می‌باشد. تمام رفتارها و شکل‌های انجام کار را نشان می‌دهد. مشخص می‌کند که چه نیازمندی‌های علمی را باید بدانیم تا بتوانیم در نرم‌افزار آن را پیاده‌سازی کنیم. زمینه‌های مرتبطی برای توسعه که باید به علوم آنها واقف شویم. برای مثال هنگام توسعه یک نرم‌افزار تشخیص پیوند مولکولی و طراحی پروتئین نیازمند آن هستیم که در مورد شاخه‌های علمی بایولوژی، بایوتک و ژنتیک علومی را کسب کنیم. این علوم می‌تواند توسط تحقیقات و پژوهش‌های فردی بدست آید یا اینکه در راستای تحصیل در یک رشته می‌توانیم در رشته دیگر به تحصیلات آکادمیک بپردازیم و به نوعی مدرک کارشناسی آن حوزه را بدست آوریم که بتوانیم به صورت کامل روی موضوع عملیاتی خود واقف و مسلط شویم.

۴.۱.۲ Statement یا جمله

Statement یک جملست که ترکیبی از پدیده‌ها می‌باشد. برای مثال گفته می‌شود، وقتی ترمز خودرو فشرده شد، درها قفل شود و کاربر بتواند وضعیت دنده خود را تغییر دهد. بعضی از این پدیده‌ها در دنیای مسئله یا محیط اتفاق می‌افتد. فعل‌های محیطی را به هم متصل می‌کند و به فعل‌های نرم‌افزاری دخالتی ندارد.

نکته: کیفیت جمله‌ها لزومی ندارد که درست باشند و می‌توانند مورد نقدر قرار گیرند.

[۱۱] Software to be

۵.۱.۲ Phenomena یا پدیده‌ها

تمام اتفاقاتی که در مسئله (یا جمله) رخ می‌دهد را پدیده یا Phenomena گویند. برخی پدیده‌ها دقیقاً داخل نرم‌افزار رخ می‌دهد، مانند خطای TLS یا خطای پیدا نشدن صفحه. برخی پدیده‌ها بین ارتباطات رخ می‌دهد مانند نرمال‌سازی دیتابیس. پدیده خرید کردن یک پدیده محیطی است. وقتی برای کاربر اعلانی ارسال می‌شود در واقع این اعلانات پدیده بین محیط و نرم‌افزار است.

۶.۱.۲ System as is

سیستمی که در حال حاضر وجود دارد سیستم جاری یا System as is گویند. سیستم جاری بیشتر به محیط مربوط است. به عبارتی دیگر، المان‌ها و ارتباطاتی است که الان وجود دارد مانند افراد و دستگاه‌ها.

۷.۱.۲ System to be

System to be دقیقاً سیستمی است که در آینده خواهیم داشت. تمام فرایند مهندسی که منجر به تولید سیستمی جدید می‌شود. چیزی که باید رخ دهد. مجموعه‌ای از المان‌های محیطی و Software to be.

۸.۱.۲ عوامل Prescriptive

عواملی که تجویزی هستند که نیاز سیستم را مشخص می‌کنند که چه کاری باید انجام شود:

۱. System requirement: یک System requirement مجموعه‌ای از Assumptionها و Software requirementهاست. تمام تک کارهای کوچکی که به محیط اختصاص می‌دهیم.

۲. Software requirement: تمام نیازمندی‌های نرم‌افزاری که می‌تواند به دو دسته Functional و Non-functional تقسیم شود. تمام تسک‌های کوچکی که به نرم‌افزار اختصاص می‌دهیم.

۳. Assumption: تمام عوامل محیطی که در پایین توضیح داده شده است.

مثال‌هایی از انواع System requirement:

- تمام درهای قطار بایستی در هنگام حرکت بسته باشند.
- مشتریان هیچ وقت نمی‌توانند بیشتر از سه کتاب را در یک زمان قرض بگیرند.
- تمام محدودیت‌های دعوت یک شرکت کننده به یک میتینگ آنلاین بایستی به زودی برطرف شود.

۹.۱.۲ مفروضات یا Assumption

تمام عواملی که محیطی هستند و مستقیماً با نرم‌افزار ارتباطی ندارند. در واقعیت امر همان محیط و یا World problem هستند. ابزارهایی واسط بین انسان و انجام کار.

۱. People: مردم و کاربران

۲. Device: دستگاه‌ها مانند سنسورها، جمع‌آور داده و ارسال کننده به موتور تحلیل (نرم‌افزار)

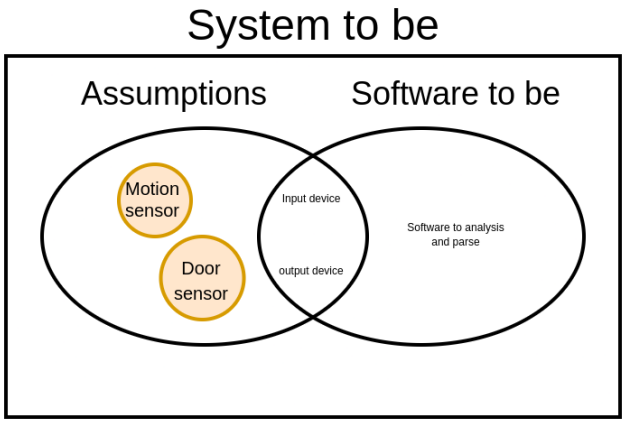
۳. Exists softwares نرم‌افزارهای موجود: نرم‌افزارهایی که خودشان عملیات متعددی انجام می‌دهند و داده‌ها را برای تحلیل به نرم‌افزار اصلی سیستم ما ارسال می‌کنند.

عوامل محیطی گسترده هستند. برای مثال وقتی که کاربر در اپلیکیشن سبد خرید خود را می‌خواهد حساب کند، زدن روی دکمه "پرداخت آنلاین" کاملا یک عامل محیطی است یعنی Assumption. زیرا با دخالت کاربر می‌توان سبد خرید را پرداخت کرد، در غیر این صورت نرم‌افزار خودش نمی‌تواند تصمیم بگیرد که پرداخت نهایی را کی باید انجام دهد (دیدگاه یک سیستم ساده).

۱۰.۱.۲ مثال

سناریو: درهای قطار موقع حرکت قفل شود. در این سناریو Statement، پدیده‌ها (Phenomena) و نیازمندی سیستم و پدیده‌های محیطی را مشخص کنید.

- جمله: درهای قطار موقع حرکت قفل شود.
- پدیده‌ها در این جمله دو نمونه هستند. حرکت کردن قطار و بسته شدن درها
- عوامل محیطی یا Assumptionها سنسور تشخیص حرکت قطار و محرک بازوی درهای قطار هستند که دائماً در حال مانیتور و کنترل در و حرکت قطار هستند.
- Assumptionها یعنی سنسورهای قطار و نرم‌افزاری که قوه تحلیل دارد یا Software requirement می‌شود نرم‌افزاری که قرار است در آینده داشته باشیم یا Software to be.
- کل این مجموعه را System to be گویند.



شکل ۱: مهندسی نیازمندی بیشتر به Assumption و قسمت اشتراکی شامل می‌شود.

۱۱.۱.۲ مفهوم Definition

یک معنای دقیق از چیزایی است که می‌نویسم به عبارت دیگر تمام اصطلاحاتی که در سیستم می‌تواند وجود داشته باشد را بیان می‌کند.

۱۲.۱.۲ مفهوم مانیتور کردن

مانیتور کردن یعنی بررسی داده‌های ورود و انجام تحلیل روی آنها.

۱۳.۱.۲ مفهوم کنترل کردن

کنترل کردن یعنی فرایند بعد از تحلیل، یعنی اعمال کردن نتایج بدست آمده.

۱۴.۱.۲ عوامل Descriptive

عوامل توصیفی، قوانین طبیعی و قید و شرط‌های فیزیک که غیرقابل مذاکره و انکار می‌باشند.

۱۵.۱.۲ ویژگی دامنه یا Domain property

یک عبارت توصیفی است که یک حقیقت از فیزیک را بیان می‌کند. این عبارت قابل مذاکره نیست که برای مثال بگوییم بعداً می‌توان آن را تغییر داد. به هیچ وجه نمی‌توان آن را کم یا زیاد کرد.
برای مثال:

۱. برای مثال دانشجو نمی‌تواند دو درس مختلف در زمان یکسان اخذ کند. یعنی از نظر فیزیک نمی‌توان همزمان در دو کلاس در زمان یکسان حاضر شد. و این پیام را نیازمندی نرم‌افزار در حقیقت برنامه نویس مشخص می‌کند.

۲. هنگامی که درهای قطار بسته باشند، یعنی دیگر باز نیستند.

۳. اگر شتاب قطار مثبت باشد، بدان معانست که سرعت قطار =! صفر می‌باشد.

۱۶.۱.۲ دامنه‌ها

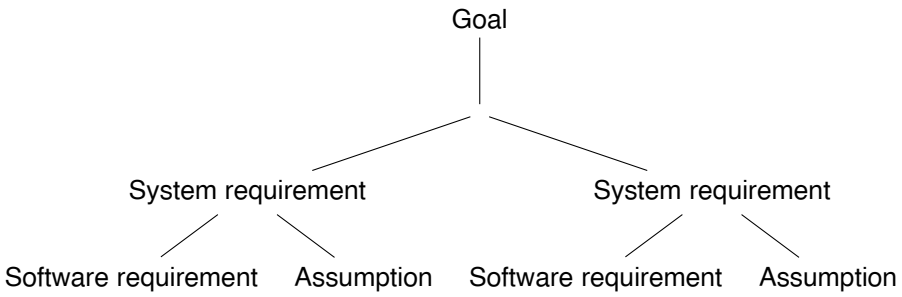
دامنه‌های در دل سازمان‌ها هستند، مانند دامنه پژوهشی، دامنه‌های مالی و ارتباط بین آدم‌ها در دامنه وجود دارد.

۱۷.۱.۲ اسکوپ‌ها

مجموعه‌هایی از System requirement هستند که نرم‌افزار می‌تواند در آنها ورود داشته باشد. مثلا فعالیت‌های مربوط به ثبت‌نام دانشجو، که اصطلاحاً به آنها System scope می‌گویند. به عبارتی دیگر، مجموعه‌ای از قابلیت‌ها که در Domain property تعریف می‌شود.
برای درک سازمان، دامنه و Scope می‌توانیم بگوییم که سازمان در واقع یک بستنی فروشی است که از سطوح بالا به پایین می‌توان به آن نگاه کرد. هر سطح پایینی را می‌توان به دامنه‌ها مشابه دانست مانند ظروفی که در آن‌ها بستنی است. و داخل هر دامنه اسکوپ‌هایی تعریف می‌شود.

نکات

- مهندس نیازمندی باید در کنترل و مدیریت اسکوپ‌ها حساسیت داشته باشد که نرم‌افزار از دست خارج نشود و باعث پیچیده‌تر شدنش نگردد.
- دامنه‌ها درست است که ثابت و غیرقابل مذاکره هستند، اما از یک دامنه به دامنه دیگر می‌تواند ویژگی‌ها تغییر کنند در حالی که ساختار این دامنه حفظ شود. برای مثال زمانی که دامنه مورد نظر یک کتابخانه فیزیکی است، همزمان دو نفر نمی‌توانند یک کتاب مشترک را تقاضا کنند. اما در کتابخانه دیجیتال که به صورت اپلیکیشن می‌باشد، درست است که ساختار دامنه همانند موجودیت‌ها و شکل کتابخانه فیزیکی است اما نحوه استفاده آن کاملاً تغییر کرده و چندین کاربر می‌توانند همزمان یک کتاب را به صورت دیجیتال مطالعه کنند.
- در مهندسی نیازمندی تنها یک نمودار استفاده نمی‌شود. برای مثال زمانی که یک نمودار Sequence برای نمایش ارتباطات دستگاه ها کشیده می‌شود نیازمند آن است که نمودار هدف نیز داشته باشد. بعد از آن بایستی تمام ریسک‌های مربوط به آن نیز به صورت نمودار اعلام شود. چرا که باعث تولید یک سند مهندسی نیازمندی کامل می‌شود که در زمان‌های مختلف می‌توان به آن مراجعه کرد و متوجه تمام موضوعات بدون فراموشی تنها یک بخش شد.
- بُعد Why در نمودار معمولاً نشان‌دهنده اهداف است. مثلا پیاده‌سازی این قابلیت هدف‌اش رضایت مشتری است.
- همیشه از اهداف شروع می‌کنیم و به نیازمندی‌های سیستمی می‌رسیم و نیازمندی سیستمی را در نیازمندی‌های نرم‌افزاری و محیطی بررسی می‌کنیم.



۱۸.۱.۲ تفاوت‌های بین Descriptive و Prescriptive

- جملات تجویزی را می‌توان برای آنها مذاکره کرد، آنها را کم و زیاد کرد یا حتی برای آنها جایگزینی معرفی نمود.
- جملات توصیفی اصلاً قابل تغییر نیستند.

۲.۲ مولفه‌های مربوط به نیازمندی نرم‌افزار در نیازمندی سیستم

۱. مانیتورینگ: تمام مقادیر محیطی که نرم‌افزار توسط دستگاه‌های ورودی مانند سنسورها، داده‌های آن را دریافت می‌کند.

۲. کنترل: مقادیر محیطی که نرم‌افزار آنهارا می‌تواند از طریق دستگاه‌های خروجی (Actuators) آن‌ها را کنترل (اعمال) کند.

۳. مقادیر دستگاه‌های ورودی [۱۲]: تمام داده‌هایی که به عنوان ورودی در نرم‌افزار استفاده می‌شود.

۴. متغیرهای خروجی [۱۳]: مقادیری که نرم‌افزار آنها را در دستگاهّای خروجی اعمال می‌کند.

نکته

بیشتر سازمان‌ها به دو دسته زیر فعالیت‌های خودشان را انجام می‌دهند:

۱. سازمان‌هایی که هدفگرا هستند و تنها برای رسیدن به محصول آخرین تلاش و فعالیت خود را می‌کنند.

۲. سازمان‌هایی که تعداد Agent و کاربرانشان زیاد است و ارزش‌های زیادی برای آنها قائل می‌شوند به صورت گراAgent یا عاملگرا هستند.

سطح System requirement بالا می‌باشد، چرا که مشتری تنها درخواست می‌کند که می‌خواهد چنین قابلیت‌هایی وجود داشته باشد، به ماهیت و نیازمندی و حتی پیچیدگی آنها کاری ندارد.

۳.۲ توافق بر لغات

۱. SOFTREQ: منظور Software requirement

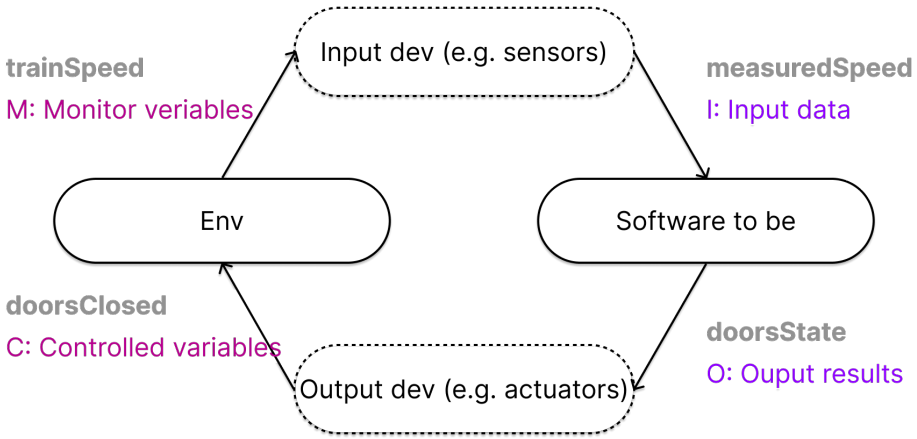
۲. ASM: منظور مفروضات یا Assumption

۳. DOM: منظور دامنه یا Domain

$$SOFTREQ + ASM + DOM \rightarrow SYSTEMREQ \quad (۳)$$

[۱۲] Input
[۱۳] Output

اگر نیازمندی نرم‌افزار، مفروضات و دامنه‌ها همگی مقید و راضی باشند نیازمندی سیستم نیز بدست می‌آید. با استفاده از پارامترهای بالا می‌توان به سیستم نهایی رسید.



شکل ۲: ارتباط نیازمندی سیستم در نرم‌افزار به همراه استدلال‌ها

- SOFTREQ: Input ‘ Ouput
- ASM1: Monitor ‘ Input
- ASM2: Ouput ‘ Control
- SYSREQ: Monitor ‘ Control

استدلال سناریو

$$SOFTREQ : measuredSpeed \neq 0 \rightarrow doorsState = "closed" \quad (۴)$$

$$ASM1 : measuredSpeed \neq 0 iff trainSpeed \neq 0 \quad (۵)$$

$$ASM2 : doorsState = "closed" iff doorsClosed \quad (۶)$$

$$DOM : trainMoving iff trainSpeed \neq 0 \quad (۷)$$

$$SYSREQ : trainMoving \rightarrow doorsClosed \quad (۸)$$

۴.۲ دسته‌بندی نیازمندی‌ها

۱.۴.۲ Functional requirement

تعیین می‌کند که چه سرویسی قرار است در Software to be ارائه شود. برای مثال:

- نرم‌افزار کنترل قطار باید بتواند سرعت تمام بخش‌های سیستم قطار را کنترل کند.
- سیستم آنلاین فهرست کتب باید براساس موضوع کتاب نام تمام کتابخانه را نمایش دهد.
- کاربران در سیستم پارکینگ آنلاین باید بتوانند رزرو لحظه‌ای و رزرو روزانه را به انتخاب خودشان استفاده کنند.
- دانشجویان زمانی که وارد کلاس آنلاین می‌شوند باید قابلیت به اشتراک گذاری صفحه نمایش خود را داشته باشند.

همچنین می‌توانند براساس شرایط محیطی باشند که تحت آن چه عملیاتی باید انجام شود:

- درهای قطار تنها در زمانی می‌توانند باز شوند که قطار به طور کامل ایستاده باشد.

دسته‌بندی توابع

۱. Information: اطلاع رسانی، اعلانات هر چیزی که قابلیت ارسال و دریافت را داشته باشد.

۲. Satisfaction: تعیین State یک کار است که در جریان معنا دارد.

۳. Stim-response: محرک پاسخ، وقتی دکمه در UI زده شد آلارم را صدا کند.

۲.۴.۲ Non-functional requirement

تعیین می‌کنند که چگونه یک سرویس می‌تواند ارائه شود. برای این دسته باید مجموعه‌ای از اقدامات که بار اجرایی دارند را استفاده کرد:

- معیارها و نیازمندی‌های کیفی:
  - معیارهای ایمنی
  - معیارهای امنیتی
  - سرعت و دقت
  - عملکرد زمانی و حافظه‌ای
  - قابلیت استفاده
- بقیه موارد
  - هنجارها
  - معماری

– نیازمندی‌های توسعه

برای مثال:

- دانشجویان هنگام به اشتراک گذاری صفحه خود کیفیت صوت را به خوبی قبل از اشتراک گذاری داشته باشند.
- قطار هنگام حرکت امکان باز کردن در را نداشته باشد.
- دستورات شتاب قطار هر ۳ ثانیه یکبار می‌تواند ارسال شود.

۵.۲ کیفیت سرویس‌دهی یا QoS (محصول)

پارامتری را نشان می‌دهد که می‌خواهیم آن را از نظر کیفی تامین کنیم. برای مثال برقراری اهداف امنیتی.

۶.۲ Service Level Agreement

یک توافق بین معمار نرم‌افزار و کارفرما برای تعیین سطح سرویس از نظر کیفی می‌باشد. در قراردادهای SLA مقدار قابل قبولی از QoSهایی که دنبالش هستیم را بیان می‌کنیم.

۷.۲ تفاوت بین Constraint و Limitation

Constraint به معنای قید و شرط است، مقید شدن به چیزی. برای مثال نرم‌افزاری توسعه داده شود که قابلیت نصب روی دستگاه‌های موبایل را داشته باشد.

Limitation به معنای محدودیت است که بار منفی دارد. در این حالت نرم‌افزار باید با آن کنار بیاید.

۸.۲ مفهوم هنجارها یا Compliance (محصول)

منظور از Compliance قواعد و هنجارهایی است که الزاما ثابت نیستند. نرم‌افزار باید تابع این هنجارها باشد. قواعدی که در نرم‌افزار قید می‌شود برای مثال فاصله بین دو ماشین در سال ۲۰۲۰ با تصمیم‌گیری شهرداری برای ماشین‌های خودران ۴ متر توافق شد. اما بعد از پیشرفت تکنولوزی و علوم مربوطه این فاصله به یک متر کاهش یافت.

۹.۲ قیدهای معماری Architectural constraint (محصول)

بعضی از قیدهای معماری مربوط به نصب و راه‌اندازی هستند و برخی دیگر مربوط به توزیع می‌باشند.

۱. نصب

(آ) نرم‌افزار باید روی پلتفرم موبایل یا عینک گوگل قابل نصب باشد

(ب) مشخصات لازم برای نصب موفقیت‌آمیز نرم‌افزار و بازی

(ج) این نیاز می‌تواند پایین‌تر از سطح سکو نیز باشد، مثلاً نصب تنها در یک سیستم عامل مخصوص

(د) قابلیت نصب تنها در سخت‌افزارهای X۸۶

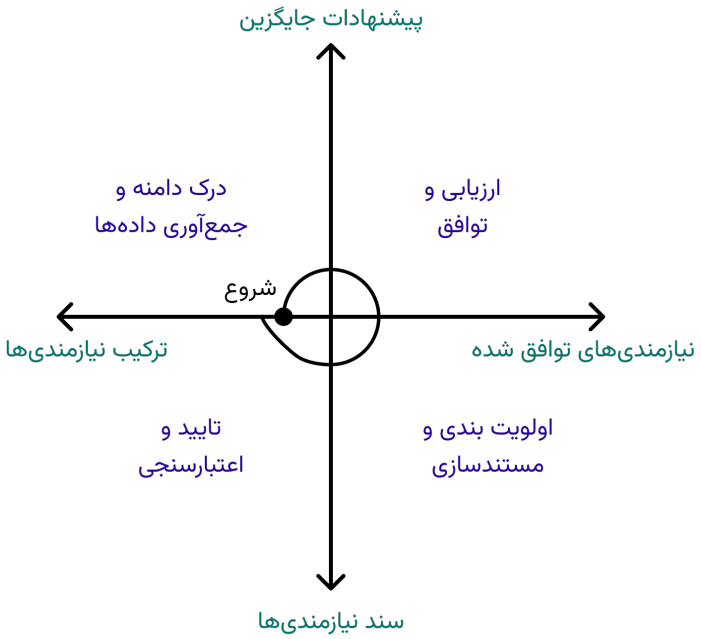
۲. قید توزیع: ورودی و خروجی از دو درب مختلف در دانشگاه، به دلیل آنکه داده‌های محیطی ورودی و خروجی در دو محل متفاوت است برای رسیدن به توافق در این توزیع باید این داده‌ها را در یک جا با هم سینک کنیم تا اطلاعات ورودی و خروجی مناسب یکدگیر پدید آید.

۱۰.۲ قیدهای توسعه Development constraint (مدیر پروژه)

یکی از مهم‌ترین عوامل نگرانی مدیر پروژه است، کاری به ماهیت محصول ندارد بلکه برای او مهم‌ترین عوامل انتخاب مناسب متدولوژی و تصمیم درست می‌باشد. تعیین هزینه زمانی و مالی نیز از دیگر نگرانی‌های مدیر پروژه می‌باشد تا در نهایت طراح معماری بتواند با دید کامل و بدون تحت فشار قرار گرفتن، معماری مناسب را طراحی کند.

۱۱.۲ فرایند و مراحل مهندسی نیازمندی

برای ساخت سبد دامنه خود نیازمند انجام فرایند مهندسی نیازمندی هستیم. این فرایند چهار قدم اصلی را بیان می‌کند. مهم‌ترین ویژگی این فرایند مراحل آن هستند که می‌توانند به صورت تکرار پذیر انجام شوند. حرکت در بین این فرایند به صورت ساعتگرد می‌باشد.



شکل ۳: مراحل مهندسی نیازمندی‌ها

۱.۱۱.۲ پیشنهادات جایگزین، درک دامنه و جمع‌آوری داده‌ها

این بخش با دامنه‌ها و استخراج نیازمندی‌ها ارتباط دارد. یعنی مهم‌ترین وظیفه در این ناحیه جمع‌آوری داده‌ها می‌باشد. سعی می‌کنیم تمام سناریوها را بررسی کنیم و به لیستی از داده‌های در رابطه با دامنه خواسته‌های مشتری برسیم. به یاد داشته باشیم که داده‌های جمع‌آوری شده صرفا همه آنها مفید نمی‌باشد پس نتیجه می‌گیریم که این لیست قابل تغییر و حذف می‌باشد که به داده‌های اصلی برسیم. برای مثال وقتی در حال جمع‌آوری داده برای توسعه سیستم مالی هستیم با داده‌های بخش بایگانی هم رو به رو خواهیم شد که هیچ ارتباط مستقیمی با سناریوهای مالی ندارد پس می‌توانیم از جمع‌آوری داده در بخش بایگانی صرف نظر کنیم.

۲.۱۱.۲ نیازمندهای توافق شده، ارزیابی و توافق

همانطور که از نامش پیداست در این ناحیه به تجزیه و تحلیل و ارزیابی داده‌ها می‌پردازیم. به گونه‌ای که سعی می‌کنیم داده‌هایی که نامربوط به Scope می‌باشد را شناسایی کنیم و آنها را حذف کنیم. هر خواسته‌ای در Scope مشتری می‌تواند ریسک‌هایی باشد که به عنوان قابلیت در نرم‌افزار می‌خواهد پیاده شود.

- قضیه برنامه LMS را در نظر داشته باشید. کلاس آنلاین به حضور دانشجویان نیاز دارد و قابلیت‌هایی در خصوص عضویت آنها در این سامانه وجود دارد اما ریسکی که در این میان به وجود می‌آید آن است که ممکن است اینترنت قطع شود و دسترسی دانشجویان به این سامانه با مشکل مواجه شود.

سوالی که در این میان مطرح می‌شود آن است که آیا تمام نیازمندی‌هایی که به سیستم وارد می‌شود الزاماً هم‌راستا می‌باشد؟ پاسخ به این سوال خیر می‌باشد چرا که ممکن است نیاز دو Assumption با یکدیگر تداخل داشته باشد.

- قضیه کارنامه را به یاد داشته باشید. درخواست مشتری اول (استاد) آن است که فقط او بتواند در هنگام ثبت نمره کارنامه را دسترسی داشته باشد. در راستای آن مشتری دوم (دانشجو) هم دقیقاً همین نیاز را دارد. این دو نیاز هم‌راستا نمی‌باشد چرا که اگر یکی را تنها برای یک نوع مشتری برآورده کنیم ممکن است با مشتری دیگر تداخل یا Conflict ایجاد شود.

۳.۱۱.۲ سند نیازمندی‌ها، اولویت‌بندی و مستندات

وقتی به این مرحله رسیده‌ایم یعنی با دو مرحله قبلی در نیازمندهای مشتری به اجماع رسیده‌ایم. یک سبدی از Scopeها که خیلی آشفته بود به یک سبدی تبدیل می‌شود که همه افراد روی آن توافق دارند. این توافق‌ها در سند نیازمندی نوشته می‌شوند. این سند یک قالب استاندارد دارد و در این قالب مشخص می‌شود که با چه ابزاری باید کار کنیم، چگونه بنویسیم و نماد بصریمان به چه شکلی باشد. بعد از این توافق‌ها این سند به طراح معماری نرم‌افزار تحویل داده می‌شود. این سند با نمودارهای بصری‌اش زبان مشترک بین طراح و مهندس نیازمندی است تا مطالب صریح و سریع به طراح معماری منتقل شود.

۴.۱۱.۲ نیازمندی‌های ترکیبی، تایید و اعتبارسنجی

سبدی که تا الان آماده شده است می‌تواند دستخوش تغییرات باشد تا به حدی که به ۸۰ درصد نیازمندی‌های ثابت و ۲۰ درصد نیازمندی‌هایی که باید تغییر کنند یا بروز شوند. این تغییر ۲۰ درصدی می‌تواند بخش‌های صحیح را هم تحت تاثیر خودش قرار دهد (اشاره به قضیه Side effect). پس در هر بار ایجاد تغییر در نیازمندی‌ها بایستی در ابتدا اعتبارسنجی شوند و تایید ایجاد تغییرات را دریافت کند.

نکته

مراحل نیازمندی‌ها می‌تواند چندین دور حلقوی داشته باشد تا همه موارد دخیل در آن به نسخه پایدار خود برسند.

۱۲.۲ نیازمندی‌ها در چرخه توسعه نرم‌افزار

سوال: آیا هر سیستمی نیازمند مهندسی نیازمندی می‌باشد؟

خیر، سند نیازمندی برای سازمان‌ها با سیستم بزرگ (سیستم‌های Legacy) کاملاً مورد احتیاج می‌باشد. به طور کل سازمان‌هایی که جریان کاری (Workflow) اصلی را اداره می‌کنند نیازمند سند نیازمندی هستند. پروژه‌های استارتاپی که به مردم خدمت می‌کنند در اصل جنس خدمت با دیگر سازمان‌ها یکی است اما نحوه انجام آن متفاوت می‌باشد. این سیستم‌ها هم سند نیازمندی برایشان اهمیت دارد.

به خاطر داشته باشید که سند نیازمندی قابلیت استفاده مجدد را به پروژه‌های مشابه می‌دهد. به طور کلی گفتنی است که سند نیازمندی یک منبعی برای پروژه‌های مشابه می‌باشد نه یک الگو.

به طور کلی، در سند نیازمندی، خواسته‌های مشتری تحلیل و جمع‌آوری می‌شود و بعد قرارداد در پروژه پیاده‌سازی می‌شوند.

۱۳.۲ Request for Proposal یا RFP

سازمان‌ها بر اساس RFP کار می‌کنند. مهندس نیازمندی و متخصصین با هم روی این سند بر اساس خواسته‌های مشتری توافق می‌کنند که کار خودشان را شروع کنند. معمولاً واحدهای IT مسئول این اسناد هستند.

۱۴.۲ تعریف: به اجماع رسیدن مطالب از سند نیازمندی

سند نیازمندی یا Requirement Document محصول اصلی فرایند مهندسی نیازمندی است. در آن سیستمی که می‌خوایم در آینده داشته باشیم (System-to-be) به شکل اهداف [۱۴]، قید و بندها [۱۵]، مفاهیم ارجاع داده شده، تسک‌ها و تکالیف مشخص شده، نیازمندی‌ها، فرضیات [۱۶] و ویژگی دامنه‌های مربوطه تعریف شده است.

۱۵.۲ تاثیراتی که سند نیازمندی به فرآورده‌های نرم‌افزاری دارد

Prototype ۱.۱۵.۲

بعد از جمع‌آوری داده‌ها به عنوان ورودی به سیستم آینده (System to be)، یک نمونه آزمایشی یا Prototype که اصطلاحاً به آن Mock-up هم گفته می‌شود، را طراحی و آماده می‌کنیم تا بتوانیم نیازهایی که از مشتری در نسخه اول سند نیازمندی دریافت کرده‌ایم را به طور کاملاً اولیه پیاده‌سازی کنیم تا بازخورد مشتری را در رابطه با آن دریافت کنیم. دلیل دو طرفه بودن این بخش با سند نیازمندی آن است که بررسی کنیم آیا نیازهایی که به ما منتقل شده است صریح و مناسب با درخواست‌های مشتری بوده است؟ ممکن است نیاز شود برخی از موارد حذف یا حتی موارد جدید را اضافه کنیم تا سبد Scope ما تکمیل شود. نکته مهم آن است که Prototype می‌تواند در سطح Functional باشد و هم در سطح Non-functional. البته باید در نظر داشت که همیشه Prototype لزومی ندارد که به صورت کامل آماده شود، بلکه ممکن است در خصوص برخی از نیازمندی‌ها که مبهم است یک Prototype درست کنیم.

Project estimations (Size, Cost, Shedules) ۲.۱۵.۲

یکی از نیازمندی‌های غیرعملیاتی مربوط به توسعه است که روی سبد Scopeها تاثیر گذار می‌باشد. در این قسمت رابطه سند نیازمندی با آن دو طرفه می‌باشد تا مشخص کنیم برای نیازمندی‌های خود چقدر زمان، چه مقدار هزینه و چه تعداد نیروی انسانی به طور مثال تعیین کنیم. در این قسمت سند نیازمندی ممکن است چند بار دستخوش تغییرات قرار گیرد و اصطلاحاً نسخه‌بندی شود. ممکن است در نسخه اولیه نیاز ما با زمان مطابقت داشته باشد اما به علت بزرگ شدن پروژه و بروز شدن خواسته‌های مشتری، دیگر این زمان با نیازمندی‌های جدید سازگاری ندارد و بایستی بروز شود.

Acceptance test ۳.۱۵.۲

این مورد رابطه یک طرفه با سند نیازمندی‌ها دارد، چرا که نیازمندی‌ها در این مرحله به درستی تنظیم شده‌اند و بعد از آن توسط معمار نرم‌افزار پیاده‌سازی صورت گرفته است. پس نیازمند مجموعه‌ای از سناریوها هستیم تا بررسی کنیم که نیازمندی‌ها با خواسته‌های مشتری مطابقت داشته است یا خیر. سناریوهای تست از سند نیازمندها طراحی و آماده می‌شود.

Architectural design ۴.۱۵.۲

طراحی معماری نرم‌افزار به طور مشخص با نیازمندی‌های نرم‌افزار در ارتباط است که ممکن است تاثیر به سزایی بر روی نیازمندی‌های Non-functional داشته باشد. بر این اساس، سند نیازمندی‌ها ورودی اساسی برای طراحی معماری نرم‌افزار از دیدگاه‌های زیر می‌باشد:

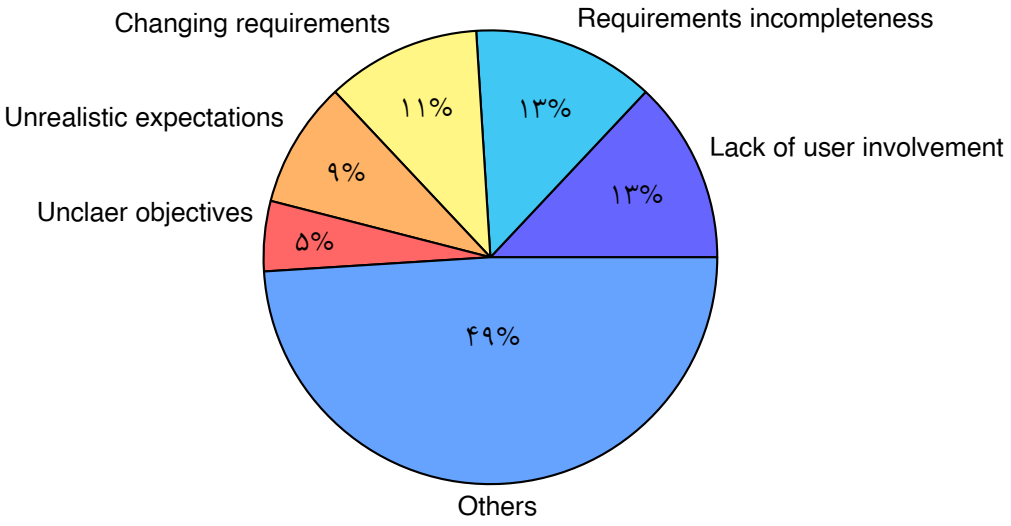
- شناسایی معماری کامپوننت‌ها و اتصالات
- مشخصات آنها که در سند نیازمندی مطرح می‌شود.
- مجموعه‌ای از سبک‌های معماری نرم‌افزار
- ارزیابی گزینه‌های معماری نرم‌افزار در برابر نیازمندی‌های Non-functional

[۱۴] Objectives
[۱۵] Constraints
[۱۶] Assumption

Software quality assurance ۵.۱۵.۲

رابطه یک طرفه با سند نیازمندی‌ها دارد به گونه‌ای که سند نیازمندی‌ها مراجع نهایی را برای فعالیت‌های اطمینان کیفیت ارائه می‌دهد.

- نیازمندی‌ها اساسی را برای تست داده‌ها و پذیرش آنها ارائه می‌دهد.
- این اساس‌ها به عنوان یک سری چک لیست برای بررسی نرم‌افزار استفاده می‌شوند.



شکل ۴: منبع اصلی شکست در پروژه‌ها مهندسی نیازمندی ضعیف (حدوداً ۵۰٪)

نکته

گاهی ممکن است در برخی از جملات این کتاب (این جزوه) از کلمه Expectation استفاده شود که در اصل منظور همان Assumptionها می‌باشد.

۳ فصل دوم، درک دامنه و جمع‌آوری نیازمندی‌ها

این فصل معادل فاز (فرایند) اول مهندسی نیازمندی یعنی استخراج داده‌ها می‌باشد. تمام مشکلاتی که در Scope می‌باشد در حقیقت System as is را مشخص می‌کند.

۱.۳ دسته‌بندی جمع‌آوری داده

جمع‌آوری داده‌ها را می‌توانیم به دو دسته زیر تقسیم کنیم، (درک دامنه و جمع‌آوری داده‌ها ترکیبی از تکنیک‌های متفاوت می‌باشد):

۱. تکنیک‌های فرآورده‌گرا یا Artifact driven: هر آن چیزی که در پروژه تولید یا استفاده می‌شود.

۲. (آ) می‌توانیم از قواعد آموزشی نیازمندی‌هایی را خارج کنیم.

(ب) Prototypeها

(ج) مستندات موجود در سازمان‌ها

۳. تکنیک‌های ذینفع‌گرا یا Stakeholder driven: هر آن چیزی که در ارتباط با آدم‌ها در سازمان باشد.

۴. (آ) جلسات

۲.۳ تکنیک‌های جمع‌آوری اطلاعات فرآورده‌گرا

۱.۲.۳ Background study

- سازمان: نمودارهای سازمانی، بیزینس پلن‌ها، گزارش‌های مالی، صورتجلسه [۱۷]
- دامنه‌ها: کتاب‌ها، نظرسنجی‌ها، مقالات، مقررات و استانداردها، گزارش‌های سیستم‌های مشابه در دامنه مشابه
- سیستم کنونی یا System as is: جریانات کاری مستند شده، فرایندها، قوانین بیزینسی، مستندات مبادله شده، گزارش‌های مربوط به شکایات، مستندات مربوط به تغییر خواسته‌های مشتری و غیره.

یکی از نیازمندی‌های مهم برای ذینفعان می‌باشد تا آن‌ها را نسبت به جلسه بعدی‌شان آماده کند.
مهم‌ترین مشکلات:

۱. حجم مستندات به شدت زیاد است

۲. جزئیات نامرتبط برای مثال بخش بایگانی اسنادی را نگهداری می‌کند که ممکن است کاملاً با یکدیگر نامرتبط باشد.

۳. اسناد ممکن است منسوخ شده یا Outdated باشند.

راه حل مشکلات این بخش:
استفاده از تکنیک هرس کردن مستندات می‌باشد. بررسی بخش‌هایی که معتبر است و حذف بخش‌هایی که منسوخ شده و غیرمعتبر می‌باشد. این تکنیک مانند خواندن فصل‌های مشخص شده از یک درس می‌باشد تا اینکه کل فصل‌های مطرح شده را بخواند.

۲.۲.۳ Data collection, questionnaires

جمع‌آوری داده‌هایی که مستندسازی نشده‌اند. مانند حقایق و ارقام. حقایق و ارقام به صورت صریح در مستندات موجود نیستند. این داده‌ها می‌تواند به صورت Meta data با شد مانند فرم ثبت‌نام، جمله ندارد بلکه براساس داده‌ها می‌توان به یک جمله رسید. بر اساس داده‌هایی که جمع‌آوری کرده‌ایم می‌توانیم جملات Functional بنویسم. نوشتن جمله و تفسیر توسط مهندس نیازمندی‌ها بر اساس داده‌های جمع‌آوری شده انجام می‌پذیرد.
این داده‌ها مانند موارد زیر می‌باشد:

- داده‌های مربوط به دیجیتال مارکتینگ، آمار استفاده، ارقام اجرایی و عملکردی، هزینه‌ها
- استفاده از تکنیک‌های نمونه‌گیری آماری

مشکلات

- ممکن است تفسیر مهندس نیازمندی لزوماً درست نباشد.
- داده کاوی مطمئن و درست ممکن است بسیار زمانبر باشد.

در روش قبل که اسنادی که می‌خواندیم اسناد عملیاتی بودند اما در این روش اسنادی که مطالعه می‌شود کاملاً غیرعملیاتی هستند (مانند معیارها و کیفیت ارائه سرویس).

روش‌های احتمالی

- requirement elicitation
- Text mining

[۱۷] Meeting minutes

پرسشنامه

لیستی از سوالاتی که توسط ذینفعان مشخص شده را آماده می‌کنیم که هر کدام یک جواب مناسب را می‌تواند در برگیرد.
نمونه‌ها میتواند:

- انتخاب یک گزینه از چند گزینه. مانند استفاده از Radio button
- سوالاتی که وزن‌دار هستند:
- – کیفی: عالی، خوب، بد
  – کمی: اعلام مقدار به صورت درصدی

ویژگی‌های یک پرسشنامه خوب

۱. تنوع زیاد کاربران و عدم تمرکز موقعیت مکانی و فرهنگ مختلف که در تمام کاربران متغیر می‌باشد. پس برای پوشش تنوع و گوناگونی [۱۸] کاربران از پرسشنامه استفاده می‌کنیم.

۲. سریع، ارزان و قابل دسترس از راه دور نیازمندی بسیاری از کاربران را جمع‌آوری می‌کنیم.

۳. پرسشنامه‌ای خوب است که روایی و کارایی داشته باشد.

تفاوت پایایی و روایی در پرسشنامه‌ها

یک پرسشنامه خوب باید دو ویژگی پایایی و روایی را به همراه داشته باشد.

- پایایی قابلیت اطمینان پرسشنامه به همراه دقت در اندازه‌گیری می‌باشد. یعنی اگر همان پرسشنامه در همان شرایط بخواهد به صورت مجدد صورت گیرد، امتیاز یا مقدار حاصل از پرسشنامه هیچ تغییری نخواهد کرد.
- روایی به معنای آن است که میزان مطابقت نتایج بدست آمده از پرسشنامه با دنیای واقعی به چه اندازه‌ای می‌باشد.

۳.۲.۳ Repertory grids, Card sorts for concept acquisition

جمله مجموعه‌ای از اسم‌ها را با فعل به یکدیگر متصل می‌کند تا یک جمله کامل را تشکیل دهد. برای مثال جمله «دانشجو باید بتواند درس انتخاب کند.» اسم‌ها به ترتیب، «دانشجو» و «درس» هستند و فعل این جمله که این دو اسم را به یکدیگر متصل می‌کند «انتخاب کردن» می‌باشد.

اسم‌ها تبدیل به کارت می‌شوند و تمام کارت‌ها معادل به کلاس هستند. تمام کلاس‌ها در فضای مسئله بررسی می‌شوند و فضای راه‌حل در حقیقت خروجی ارتباط آنها (جمله) است. یکی از مثال‌های فضای راه‌حل اتصال به دیتابیس می‌باشد.

فضای مسئله

دقیقاً وضعیت موجود را نمایش می‌دهد. تمام چیزهایی که می‌بینیم در حقیقت فضای مسئله می‌باشد.

فضای راه‌حل

فضای راه‌حل نتیجه ارتباط جملات و کلاس‌ها هستند که طراح مشخص می‌کند.

[۱۸] Diversity

مثال

برای مثال می‌توان به دانشجو و شماره دانشجویی اشاره کرد. نام و نام خانوادگی، تاریخ تولد، سال ورودی دانشگاه، رشته ورودی، گرایش رشته و غیره تمام مسائلی هستند که موجودیت دانشجو را تعریف می‌کنند پس فضای مسئله می‌باشند.

طراح سیستم دانشگاهی با توجه به این فضای مسئله ورودی‌ها را بررسی می‌کند و یک خروجی برای مشخص کردن یکتا بودن دانشجو تولید می‌کند و آن هم شماره دانشجویی می‌باشد که یکی از مهم‌ترین فرآورده‌های فضای راه‌حل است.

نکات

- کاملاً بستگی به نیاز سیستم دارد که مشخص کنیم یک اسم کلاس باشد یا نه. زیرا یک اسم می‌تواند کلاس باشد یا می‌تواند به عنوان ویژگی کلاس دیگری یا Attribute باشد. برای مثال کتابخانه می‌توان اشاره کرد که اگر بخواهیم «کتاب‌ها» و «نویسندگان» را کلاس جداگانه در نظر بگیریم می‌توانیم کوئری‌هایی در این بابت داشته باشیم که یک کتاب را چه نویسندگانی تالیف کرده‌اند و یا یک نویسنده چه کتاب‌هایی دارد. یا می‌توانیم نیاز سیستم را در این ببینیم که یکی از Attributeهای کتاب نویسنده باشد به جای آن که یک کلاس جداگانه داشته باشد.
- در حالت کلی می‌توان گفت که قانون سفت و سختی برای تشکیل کلاس از روی کارت‌ها وجود ندارد و کاملاً نیاز سیستم مشخص می‌کند که کلاس باشند یا Attribute.
- کلاس با محیط مسئله و طراح همراه می‌باشد
- اطلاعات یک محصول از ویژگی‌های کلاس است و دسته‌بندی کردن و کتگوری از راه‌حل مسئله
- صفات یا Attributeها حاوی اعتبارسنجی هستند. برای مثال دارای محدوده هستند، نوع دارند و می‌توانند بیان کننده اندازه و پذیرنده مقدار ورودی باشند.

ویژگی‌ها و معایب

- ساده و ارزان
- خیلی از این جمله‌ها می‌تواند دقت پایینی داشته باشند و نامرتبط باشند. حتی ممکن است به اسکوپ ما مربوط نباشند.
- افرادی می‌توانند این بخش را مدیریت کنند که اسکوپ را خیلی خوب درک کرده باشند.

۴.۲.۳ Scenarios, Storyboards for problem world exploration

سناریو به معنای شرح داستانی است که می‌تواند وضعیت و شرایط کنونی و آینده را تعریف کند. یعنی تعریف System-as-is تا System-to-be. فقط شروط، جمله‌ها و سطح پیچیدگی در سیستم افزایش پیدا می‌کند نیازمند تعریف سناریوها هستیم تا بتوانیم سیستمی که می‌خواهیم طراحی کنیم را بهتر درک کنیم. برای مثال سناریو انتخاب واحد دانشجو یا اخذ دانشجوی مهمان حاوی شرایط بسیار گسترده‌ای است که بایستی برای هر کدام از آنها سناریویی در نظر گرفته شود به همین دلیل اهمیت سناریو بسیار بالا می‌باشد.

- چه کاری یا What
- چه کسی یا Who
- چرایی یا Why
- چه می‌شود اگر این اتفاق در نظر گرفته نشود. یعنی دیدن تمام Exceptionها که What if را مشخص می‌کند.

سناریوها را با استفاده از نمودارهای Sequence نمایش می‌دهیم که در آن تمام ابعاد بالا وجود دارد به غیر از بُعد Why.

انواع سناریوها در کنار یکدیگر

سناریو منفی

سناریو منفی تمام کارهایی است که سیستم نباید انجام دهد.

سناریو مثبت

تمام کارها و رفتارهایی که انتظار داریم سیستم انجام دهد.

سناریو نرمال

مجموعه سناریوهای مثبت و منفی در کنار یکدیگر است. برای مثال دانشجو باید بتواند انتخاب واحد کند (سناریو مثبت). بدون پرداخت شهریه دانشجو نمی‌تواند انتخاب واحد انجام دهد (سناریو منفی).

سناریو غیرنرمال یا Abnormal

سناریوای است که در آن Exceptionها مشخص می‌شود. برای مثال، دانشجویی که شهریه پایه را پرداخت کرده باشد می‌تواند انتخاب واحد را انجام دهد. اگر واحدی را در ترم گذشته مشروط شده باشد که در گروه درسی اجباری باشد در این صورت بایستی این ترم آن درس را مجدداً اخذ کند در غیر این صورت انتخاب واحد او در این ترم ناقص خواهد بود.
در حقیقت سناریو غیرنرمال آینده‌نگری روی سیستم System-to-be خواهد بود.

نکته

در سناریو نویسی اول سناریو نرمال نوشته می‌شود و سپس برای آن تمام Exceptionها را براساس What ifها در نظر می‌گیرند تا سناریو منفی را تشکیل دهند.

مزایا و معایب سناریوها

- در سناریوها بُعد Why وجود ندارد.
- قصه گفتن سخت است و سطح پیچیدگی بالایی را دارد.
- ساده بودن از بزرگترین حسن آن است.

۵.۲.۳ Prototypes, Mock-ups for early feedback

همانطور که در صفحات قبلی هم گفته شد، برای متوجه شدن RFP بخشی از درخواست‌ها را به صورت اسکیس یا User interface خیلی کلی طراحی می‌کنیم که کمترین حالت تعامل را دارد تا مشخص شود آیا تا به اینجا کار درخواست‌های مشتری را متوجه شده‌ایم و بعد از آن مهندسی‌های نیازمندی با درست بود است یا خیر. بعد از جواب گرفتن [۱۹] از این قسمت می‌توانیم بخش UI را کامل و سپس شروع به پیاده‌سازی کل سیستم کنیم.

- تهیه Prototypeها مستقیم‌ترین فرآورده‌ای است که برای استخراج نیازمندی‌ها استفاده می‌کنیم. بیشتر در آن نیازمندی‌های عملیاتی یا Functional requirement دیده می‌شود.
- نکته مهم آن است که قابلیت‌هایی ارائه می‌دهیم را به صورت Functional و طراحی که بابت Prototype انجام داده‌ایم را Non-Functional می‌دانیم.

[۱۹] Feedback

- بیشتر تمرکز این روش برای بحث در مورد نیازمندی‌هایی که گنگ بوده می‌باشد.
- در این روش کار تقریباً سریع می‌باشد تا بتوانیم چالش‌های سیستم را نشان بدهیم و تدابیری برای آن بیاندیشیم.

۶.۲.۳ Knowledge reuse: Domain-independent, Domain specific

در مورد بازیابی دانشی صحبت می‌کند. هدفش افزایش سرعت جمع‌آوری اطلاعات به وسیله بازیابی دانش از تجربه‌ها نسبت به سیستم‌های مرتبط است. دانشی مشابه در مورد سازمان‌ها، دامنه‌ها، جهان مسئله مانند نیازمندی‌ها، فرضیه‌ها، ویژگی دامنه‌ها و غیره.

استفاده از فرآیندهایی که قبلاً در سیستم‌های مشابه حضور داشتند

برای مثال بانکداری‌ها معمولاً مجموعه‌ای از تسک‌ها و نقش‌های مشابه‌ای را دارند که به صورت پوشا یا Overlap می‌باشند. یعنی اگر بانک مرکزی هر تعریفی داشته باشد، دیگر بانک‌های کشور نیز از همان تعاریف در سیستم‌های خود بدون تغییر و سازگاری استفاده می‌کنند. چرا که همه چیز در سیستم بانکداری مشابه می‌باشد.

در اینجا هدف بر نمایش دانش می‌باشد که المان‌های دانشی که در آینده تشکیل می‌شوند را شناسایی کنیم. المان‌های دانشی از جنس زیر هستند:

- مفاهیم یا Concepts
- اهداف یا Goals
- وظایف یا Tasks
- افراد یا Agents
- نیازمندی‌ها یا Requirement
- دامنه‌ها یا Domain

روش نمایش دانش به صورت گرافیکی است. البته به صورت متنی یا Text هم می‌تواند باشد اما رسمی نخواهد بود چرا که متن را هر کسی می‌تواند با برداشت خودش بنویسد. اما تصاویر همه چیز را به همه کس یک شکل نشان می‌دهند. برای مثال در خیاطی نمایش دانش را الگو یا Pattern می‌گویند.

مراحل بازیابی دانش

شامل سه مرحله زیر می‌باشد:

۱. Retrieve: دانش مرتبط (مناسب) را از سایر سیستم‌ها دریافت کنیم. در این مرحله ممکن است الگوی کاملی برای سیستم ما وجود نداشته باش پس تکه تکه از هر نرم‌افزار الگوهای آن را استفاده می‌کنیم.

   (آ) سیستم ثبت‌نام باشگاه: قابلیت عضوگیری. تنها مختص به باشگاه نیست بلکه کاری است تکرار پذیر. پس باید المانی‌های دانشی وجود داشته باشد که مشخص کند چه بخش‌هایی دارد و مفاهیم و اهداف و غیره چیست؟

   (ب) راه‌حلی مشترک برای خانواده‌ی مشترکی از مسائل است.

۲. Transpose: گاهی همه چیز دقیقاً از همان الگوی قبلی برای سیستم جاری ما قابل استفاده نیستند و بایستی نسبت به نیاز سیستم تغییر کنند.

(آ) ماهیت جنس‌ها متفاوت است. درست است که الگو به شکل یک راهنما برای System-to-be خواهد بود، اما باید یکسری موارد تبدیل شوند. برای مثال وقتی می‌خواهیم فروشگاه آنلاین بزنیم، می‌دانیم که فروش وسایل ورزشی یکسری شرایط دارد و فروش محصولات لبنیاتی هم شرایط مخصوص به خودش را دارد. استفاده مشابه الگوی فروش تنیس با فروش پنیر کاملاً اشتباه می‌باشد.

۳. Validate: الگوهایی که یافت می‌شود بایستی با نیازهای سیستم سازگار شوند و با سیستم آینده یکپارچگی پیدا کنند (تطبیق‌پذیری). الگوها بایستی با یکدیگر سازگار شوند چون ممکن است تغییری داشته باشند تا بتوانند در کنار هم قرار بگیرند.

روش‌های انتقال و سازگاری یا Transpose

انتقال به سه روش انجام می‌شود چرا که اهمیت بسیار بالایی دارد و مقداری سطح پیچیدگی متفاوتی نسبت به بقیه مراحل دارد:

۱. Instantiation یا نمونه‌سازی: مستقل از دامنه می‌باشد.

۲. Specialization یا خاص‌سازی: دقیقاً در مورد دامنه می‌باشد.

۳. Reformulation: عمل واجب برای بیان فرمول با کلمات سیستمی

در نمونه‌سازی سازمان، تطبیق دادن آن به سیستم کمی دشوار است چرا که یک مفهوم کلی را در بالاترین سطح بیان می‌کند که بایستی دانشی در مورد Business planهای آن داشته باشیم و مدل‌های سازمانی آن یادگرفته شود. زیرا هیچ وقت در مورد دامنه صحبت نمی‌کند. برای اینکه خاص دامنه باشد جزئیات بیشتر خواهد بود.

نکته

اگر دامنه تغییر کند شرایط و خاص بودن دامنه نمی‌تواند مانند سابق باشد، فلذا الگو نیز تغییر می‌کند.

بازیابی اطلاعات دارای دو نوع است

۱. مستقل از دامنه یا Domain independent

۲. خاص دامنه یا Domain specific

در ابتدا از بالاترین سطح سازمان توضیح می‌دهیم تا بتوانیم به جزئی‌ترین بخش‌های دامنه داخل سازمان برسیم. در بالاترین سطح می‌پردازیم که یک خدمت چگونه و به چه شکلی ارائه داده شود؟ در پایین‌ترین سطح سازمان که دامنه‌ها و اسکوپ‌های فعالیتی آنها می‌باشد می‌پردازیم چه چیزی را قرار است ارائه دهیم؟

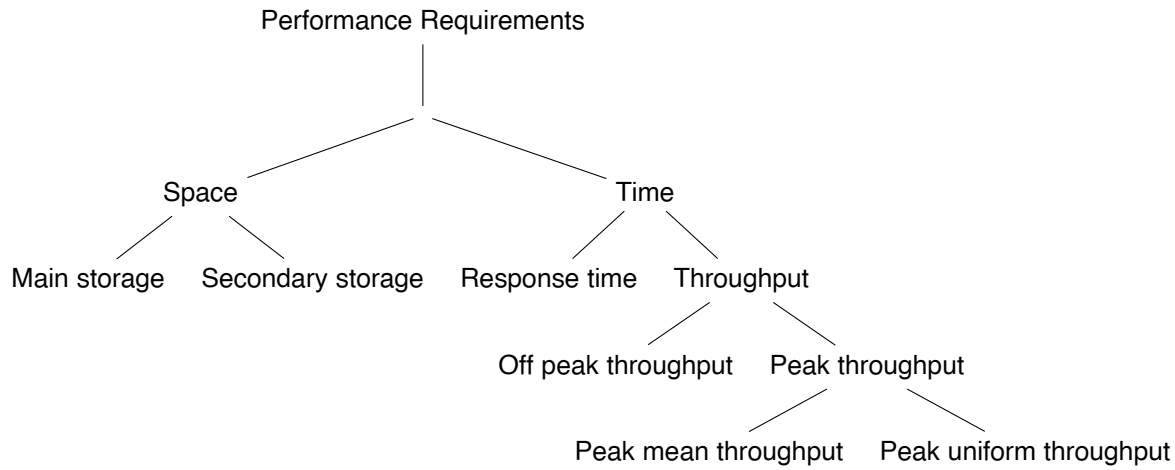
مستقل از دامنه یا Domain independent

۱. توجه به تاکسونومی نیازمندی‌ها

۲. بازیابی از متا_مدل‌ها

توجه به تاکسونومی

این روش سلسله موارد نیازمندی‌ها را در نظر گرفته است که به صورت نیازمندی‌های غیرعملیاتی یا Non-functional باشند. عملیاتی مانند الگوریتم‌های جست و جو و پارامترهای امنیتی.
نکته مهم آن است که خاص‌تر شدن تاکسونومی باعث جست و جوی متمرکزتر خواهد شد.



پارامترهایی که در درخت تاکسونومی مشاهده می‌کنید در حقیقت تماماً به صورت نیازمندی‌های غیرعملیاتی هستند که در راس سازمان قرار دارند و به جزئیات دامنه‌ها کاری ندارد.

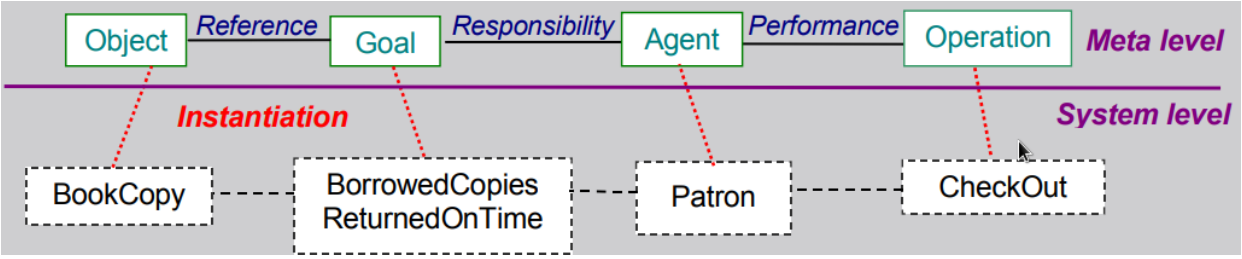
به عنوان مثال میزان زمان پاسخ‌دهی یک قید برای شرکت کنندگان در یک میتینگ آنلاین می‌باشد. پارامترهایی مانند زمانبندی کردن و ارسال اعلانات می‌تواند از نیازمندی‌های غیرعملیاتی باشد که در این مورد استفاده می‌شوند.

و پارامتر PeakMeanThroughput مشخص می‌کند که چه تداد شرکت کنندگانی می‌توانند در یک جلسه آنلاین حضور داشته باشند.

متامدل‌ها

متامدل‌ها در مورد نیازمندی‌های عملیاتی یا Functional می‌باشد که در سطح سازمان عمل می‌کند. بیشتر در مورد جزئیات صحبت می‌کند به همین خاطر خیلی کم مورد استفاده قرار می‌گیرد زیرا حاوی سطح پیچیدگی بالا می‌باشد.

- مفاهیم و روابطی که موارد RD براساس آنها جمع‌آوری شده است.
- دانشی که از سازمان و سیستم مورد هدف بدست می‌آوریم به عنوان نمونه‌ای از المان‌های متا_مدل‌ها نسبت به سازمان و سیستم آینده‌ای می‌باشد که مدل شده است.
- این به ما کمک می‌کند که بدانیم دقیقاً چه می‌خواهیم و چه سوالی نسبت به مسئله داریم.
- استخراج اطلاعات به وسیله پیمایش متا_مدل‌ها انجام می‌شود.
- اگر برای هر جز بخواهیم متا_مدل در نظر بگیریم بایستی برای آنها متا کلاس‌ها را بنویسیم:
  - Actor ← actors
  - Task ← tasks
  - Resource ← resources
  - Dependency ← dependencies
- یا برای مثال برای موارد زیر:
  - Goal ← goal
  - Object ← objects
  - Agent ← agents
  - Operation ← operations



شکل ۵: بازیابی دانش مستقل از دامنه، با استفاده از پارامترهای Objects, Goal, Agents, Operation

خاص دامنه یا Domain specific

در حقیقت همانطور که از نامش پیداست تماماً و عمیقاً در مورد دامنه صحبت می‌کند تا بالاترین سطح سازمان.
برای مثال وقتی سیستم مورد نظر نمایشگاه ماشین باشد دامنه‌های آن می‌تواند موارد زیر باشد:

- مدیریت نمایشگاه ماشین
- مدیریت تحویل و دریافت ماشین‌ها از کمپانی
- مدیریت بخش تصادفات
- مدیریت بخش بیمه ماشین

مثال

سیستم مورد هدف: مدیریت کتابخانه و دامنه: مدیریت منابع
در این مثال منابع در حقیقت کتاب‌ها هستند.

۱. Concepts یا مفاهیم:

(آ) منابع (Resource)

(ب) Resource unit

(ج) Repository

(د) Resource usage

(ه) Resource availability

(و) Resource reservation

۲. Tasks:

(آ) Tracking the history of resource usage (Like a ledger)

(ب) Handling resource requests

۳. Actors:

(آ) Resource users

(ب) Resource manager

۴. Objectives:

(آ) Ensuring wide accessibility of resource units to users

(ب) Ensuring appropriate resource localization in the respoitory for easy retrieval

۵. Requirements یا نیازمندی‌های این سیستم

(آ) بررسی حد استفاده منابع به طور مناسب توسط کاربران

(ب) تعریف رنج و محدودیت: برای مثال کاربران بیشتر از ۵ کتاب را نمی‌توانند از کتابخانه قرض بگیرند.

۶. Domain property یا ویژگی دامنه:

(آ) به دلیل آن که کتابخانه فیزیکی است ممکن است از یک کتاب فقط یک نسخه موجود باشد، پس وقتی کاربری کتابی را قرض می‌گیرد کاربر دیگر نمی‌تواند تقاضای مشابه داشته باشد (یک fact می‌باشد).

(ب) وقتی کاربری خوش قول نباشد و کتابی به کتابخانه تحویل ندهد، امتیاز منفی برای او ثبت می‌شود و از اون جریمه دریافت خواهد شد.

بخش‌هایی که خاص دامنه می‌باشد

- Book ← Resource
- Book copy ← Resource unit
- Library shelves ← Repository
- Patron ← User
- Library staff ← Manager

۳.۳ تکنیک‌های جمع‌آوری اطلاعات ذینفع‌گرا

بیشتر با انسان‌ها ارتباط دارد تا فرآورده‌هایی که در فرایند مهندسی ممکن است تولید شوند یا مورد استفاده قرار گیرند.

۱.۳.۳ Interviews

تمام سوالاتی که برای آن‌ها جوابی نداریم را در بخش Interview می‌پرسیم. مثلا بعد از ثبت‌نام کاربر برای او اعلاناتی ارسال شود؟ یا مثلاً می‌گویم که در سناریو تغییر کلاس کاربر اعلانات ارسال شود.

در حقیقت برای تمام سوالاتی جوابی نداریم آن‌ها را در بخش Interview مطرح می‌کنیم. پرسش‌نامه به تنهایی کامل نیست، بلکه برای کامل شدنش نیاز به مصاحبه دارد. معمولاً بدیهیات در مصاحبه پرسیده نمی‌شود.

۱. ساخت‌یافته: براساس سوال است که پرسیده می‌شود.

۲. بدون ساختار: گفت و گو آزاد و بدون هیچ پیش‌درآمدی می‌باشد.

۲.۳.۳ Observation and ethnographic studies

در مورد مشاهدات صحبت می‌کند که بیشتر سیستم حال حاضر یا System-as-is را مورد بررسی قرار می‌دهد. هم Functional requirements استخراج می‌شوند و هم Non-functional requirements.

این بخش می‌تواند به صورت غیرفعال مانند حالت ناظر باشد یا می‌تواند به صورت فعال به شکل درگیر شدن با فرآیندها که با یادداشت برداری همراه است باشد.

۳.۳.۳ Group sessions

- ساختار یافته: به هر عضو تیم زمانی داده می‌شود. همه موارد به صورت مشخص از پیش تعین شده است. کاملاً نقش هر فردی در تیم مشخص می‌باشد.
- بدون ساختار: مانند لحظات طوفان فکری که می‌تواند به شکل نگارش صورت جلسه به عنوان نتیجه همفکری‌ها باشد.
- بدون سانسور شدن و تمسخر می‌باشد.

۴ فصل سوم

در فصل یک و دو در مورد قدم اول مهندسی نیازمندی یعنی جمع‌آوری اطلاعات صحبت شد. در این فصل در مورد بررسی و ارزیابی داده‌های جمع‌آوری شده مرحله قبل صحبت خواهیم کرد. نتیجه‌ای که این مرحله دارد آن است که همه اعضای تیم به یک اجماع و توافق بر سر تمام مواردی که انتخاب شده است برسند.

۱.۴ چهار کار اصلی ارزیابی داده‌های جمع‌آوری شده

۱. Inconsistency management: مدیریت ناسازگاری، نویسنده کتاب در شرایط خاصی که دو چیز با هم سازگاری ندارند را می‌گوید ناسازگار است و گاهی در برخی قسمت‌های کتاب از کلمه تضاد یا Conflict استفاده کرده است. تضاد زمانی رخ می‌دهد که جملات با هم تضاد داشته باشد.

(آ) اگر بین جملات تناقص باشد به آن می‌گویند تضاد یا Conflict که در نیازمندی‌های نرم‌افزاری، نیازمندی سیستم و Assumptionها رخ می‌دهد.

(ب) اگر تناقص بین المان‌ها باشد می‌گویند المان‌ها ناسازگاری دارند.

۲. Risk analysis: بررسی ریسک‌ها

(آ) در حقیقت تمام اتفاقاتی را می‌گوید که ممکن است در برابر آنها کارهایی انجام بدهیم یا قابلیتی را طراحی کنیم که معمولاً محیطی، نرم‌افزاری و دامنه‌ای هستند.

(ب) برای مثال: فراموشی گذرواژه یک بررسی ریسک بوده است، که پیامک شدن گذرواژه یا OTP به صورت عامل محیطی یا Assumption بوده، طراحی لاگین و فرم فراموشی گذرواژه از نوع نیازمندی نرم‌افزاری که در برابر ریسک تمهیداتی در نظر گرفته شده است.

(ج) نکته مهم در ریسک‌ها آن است که هیچ وقت در زمان جمع‌آوری داده‌ها ریسک را بررسی نمی‌کنیم چون ممکن است ناخودآگاه برخی موارد را ریسک در نظر بگیریم و در جمع‌آوری آنها حساس شویم.

(د) می‌تواند در خصوص مجموعه اقداماتی باشد که در سیستم تکرار پذیراند مانند تحلیل‌گر ریسک در سیستم‌های مخشص مانند سیستم‌های مالی

(ه) ریسک Not یک جمله می‌باشد.

(و) ریسک برای یک جمله می‌باشد، اما تضادها برای دو یا چند جمله می‌باشد (به تمرین p2.pdf مراجعه شود).

۳. انتخاب بین گزینه‌ها: بعد از ریسک‌ها گزینه‌هایی که به نظرم مناسب بوده است که فیلتر کردیم را بایستی بین آنها یکی را انتخاب کنیم که در سیستم نهایی خود استفاده کنیم.

۴. اولویت‌بندی کردن کارها: همه کارها در یک سطح اهمیت نخواهند بود. پس نیازمند اولویت‌بندی کارهای مشخص شده در مرحله قبل هستیم. یکی از بارزترین مثال‌ها نسخه‌بندی کردن کارها می‌باشد.

۲.۴ ناسازگاری‌ها

ناسازگاری بین المان‌های دانشی اتفاق می‌افتد که مرتبه تکرار بسیار زیادی در مهندسی نیازمندی دارد. معمولا دو بُعد ناسازگاری وجود دارد:

- Inter-viewpoint: مربوط به NFRها نیست و معمولاً ذینفعان تمرکز و نگرانی‌های خودشان را دارند. برا مثال کارشناس دامنه در برابر بخش بازاریابی.
- Intra-viewpoint: خواسته‌های مختلف کاربران که به صورت عملیاتی هستند. حلشان با استفاده از الگوریتم‌ها امکان‌پذیر می‌باشد.

ناسازگاری‌ها به ۳ دسته تقسیم می‌شوند تا قبل از طراحی توسط طراح سیستم همه با این مفاهیم به اجماع برسند:

۱.۲.۴ تصادم معنایی یا Terminology clash

استفاده از چندین نام برای یک مفهوم مشترک را می‌گوید.

- کسی در دانشگاه درس می‌دهد نام‌های مختلفی دارد: استاد، دکتر، مدرس
- کسی که کتاب را از کتابخانه قرض می‌گیرد: کاربر، قرض‌گیرنده، متشری یا Patron

این تضاد معنایی به گونه‌ای است که هر معنا یک کلاس خاص خواهد بود که هیچ ربطی ندارند تا به یکدگیر متصل شوند.

۲.۲.۴ تصادم در تعیین و طراحی یا Designation clash

استفاده از یک نام برای چند مفهوم مختلف را می‌گوید. برای مثال: کسانی که در دانشگاه کار می‌کنند را کارمند می‌گویند. این کارمندان شامل، آبدارچی، رییس دانشگاه، مدرسان و اعضای هیات علمی می‌باشد. دقیقاً در این رابطه منظور از کارمندان کدام است. قواعد به طور کلی متفاوت هستند و تعاریف مختلف اسامی مخصوص به خودشان را دارند.

یا مثالی دیگر در رابطه با کارمندان دانشگاه این است که دولت می‌خواد حقوق کارمندان دانشگاه را افزایش دهد. الان چه قشری از دانشگاه قرار است حقوقشان افزایش پیدا کند؟ اساتید؟ اعضای هیات علمی؟ معاونین و رییس دانشگاه؟ دقیقاً کدام بخش قرار است اثر بخشی این مسئله صورت گیرد؟

۳.۲.۴ تصادم ساختاری یا Structure clash

کلاسی به نام درس داریم که یک صفت به عنوان زمان دارد. در یک قسمت می‌گوییم که کلاس آزمایشگاهی دو ساعت می‌باشد و در یک قسمت می‌گوییم که کلاس آزمایشگاهی بین ساعت ۱۰ تا ۱۲ ظهر می‌باشد. در دو زمان هستند اما نوع و ساختار متفاوتی دارند. از نظر منقط دارند در مورد زمان صحبت می‌کنند ولی ساختارشان متفاوت است که باعث شکست در سیستم خواهد شد.

تمام مشکلات ۳ مورد ناسازگاری را می‌تواند در فهرست واژگان یا Glossary سند نیازمندی‌ها RD مطرح کرد تا همه بتوانند با تمام قواعد و معنای سیستم به صورت اصولی آشنا شوند. در حقیقت مطرح کردن این واژگان وظیفه مهندس نیازمندی است و طراح سیستم بایستی تمام این موارد را مطالعه کند و در کامل کردن مطالب نقش داشته باشد. می‌تواند نوع کلاس‌های خود را تعیین کند. تایپی مشخص را برای سیستم تعریف کند و غیره.

نکات

- نکته: منظور از Handle کردن یعنی راست و ریس کردن ناسازگاری‌هایی که بعد از جمع‌آوری اطلاعات رخ داده است.
- سازمان‌های با تعریف Ontology یا هستی شناسی، تفاوت بین المان‌های دانشی را مطرح می‌کنند.
- هستی شناسی ارتباط بین معناها با معناهای دیگر، که در نهایت موجب ایجاد نود و معنای جدید می‌شود که بسیار وابسته به دامنه است.

۳.۴ تضادها

تضاد ها به دو دسته تقسیم می‌شوند:

۱.۳.۴ تضاد قوی یا Strong conflict

در هیچ شرایطی نمی‌توانیم هر دو جمله را با هم در سبد نیازمندی خود نگهداریم. از نظر منطقی امکان پذیر نمی‌باشد. برای مثال دو جمله زیر بیان می‌شود:

- دانشجو بتواند کارنامه خود را ببیند.

- استاد در هنگام ثبت نمره بتواند کارنامه دانشجو را ببیند.

در دو جمله بالا اگر هر دو خواسته را بخواهیم برقرار کنیم حتماً به تضاد بر می‌خوریم. در این شرایط طراح انتظار دارد که مهندس نیازمندی تکلیف کار او را روشن کند که دقیقاً باید چه سیستمی طراحی کند و چه دسترسی‌هایی را بین هر دو کاربر برقرار سازد. مثال بیشتر در تمرین دوم در فایل p2.pdf.

در مثال کتابخانه سنتی، دو کاربر هیچ وقت نمی‌توانند یک کتاب با ISBN و جلد یکسان را از کتابخانه قرض بگیرند.

یا برای مثالی شفاف‌تر، از نظر تضادها می‌توانیم به شرایط NFRها اشاره کنیم. هیچ وقت نمی‌توان بهترین امنیت را با بالاترین سرعت داشت، زیرا از نظر منطق الگوریتم‌های امنیتی شرایط را پیچیده‌تر می‌کنند و خودآگاه باعث کاهش سرعت ورودی و خروجی داده‌ها در سیستم خواهند شد.

۲.۳.۴ تضاد ضعیف یا Weak conflict

تضادی است که تا یک مرزی همه چیز خوب پیش می‌رود و هم سیستم راضی است هم کاربر، اما بعد از آن شرایطی به وجود می‌آید که سیستم را متاثر می‌کند. مانند Deadlineها، تا زمانی که رخ نداده است هیچ مشکلی پیش نمی‌آید ولی به محض اینکه از زمانش می‌گذرد در سیستم تضاد ایجاد می‌کند.

برای مثال کتابخانه سنتی می‌توان گفت وقتی مهلت تحویل کتاب توسط خواننده کتاب، ۳ هفته باشد، تا قبل از سه هفته اگر تحویلی انجام شود سیستم هیچ تضادی ندارد اما به محض اینکه وارد هفته چهارم می‌شود و خواننده، کتاب را به کتابخانه تحویل نداده باشد در سیستم کتابخانه تضاد ایجاد می‌کند.

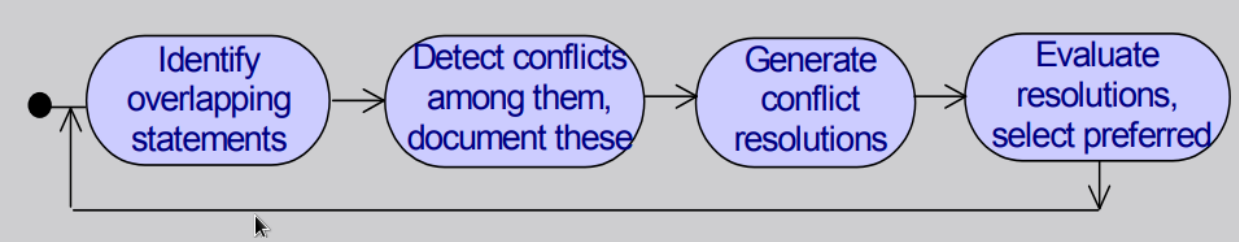
راه‌حل این دو تضاد

- مهم‌ترین رویکرد مدیریت کردن است.
- برای بهتر کردن فرآیندها در خصوص تضادها، استفاده از تکنیک‌های الکوریتمیک ضروری می‌باشد.

نکات

- رفع کردن تمام تضادها به صورت برد_برد امکان پذیر نیست.
- منظور از مدیریت کردن در مورد تضادها به معنای آن است که جملات را در کنار یکدیگر راضی نگه داریم.

۴.۴ مدیریت تضادها Managing conflicts



شکل ۶: چهار قدم چرخشی مدیریت تضادها

مدیریت تضادهای ضعیف و قوی در ۴ قدم انجام می‌شود:

۱. Identify overlapping statements: شناسایی عباراتی که با هم مشترک هستند و در مورد یک مفهوم مشترک صحبت می‌کنند. شباهت‌های می‌توانند فاعل، فعل و مفعول باشند. عملیاتی که در کنار یکدیگر دچار تضاد نمی‌شوند.

(آ) پدیده‌های باز و بسته شدن درهای خطار مفاهیم رایج در سبد نیازمندی‌های آن است.

(ب) پدیده‌های بدست آوردن کتاب، قرض گرفتن و بازگرداندن کتاب نیز از مفاهیم رایج مرتبط به Book copy می‌باشد.

۲. Detect conflicts among them, document these: از میان جملات جمع‌آوری شده باید بررسی کنیم که ببینیم چه نظراتی با هم همپوشانی دارند. اگر از میان همپوشانی‌ها تضادی پیدا شد بایستی تضادها را داکیومنت کنیم. راه‌های تشخیص تضاد:

(آ) Informally یا غیررسمی: به صورت غیررسمی اعلام می‌کنیم که همپوشانی عبارات با هم رضایت بخش هستند و تحت چه شرایطی راضی کننده نیستند؟ به گونه‌ای که به صورت چشمی منطقی نیستند.

(ب) استفاده از روش‌های اکتشافی یا Heuristics (استفاده از درخت): براساس یک جدول مشخص می‌کند که جملات چگونه می‌توانند با یکدیگر تضاد داشته باشند.

(ج) استفاده از روش رسمی یا Formally: تکنیک‌های اثبات قضیه. در حالت رسمی نرم‌افزارهای بحرانی را نمی‌توان UML کرد چرا که نیاز به اثبات دارند. نمایش و اعتبارسنجی هم با استفاده از زبان‌های رسمی امکان پذیر می‌باشد.

(د) استفاده از الگوهای تضاد که نسخه‌ای سبک‌تر از تکنیک‌های رسمی هستند. نتیجه به صورت گرافیکال می‌باشد.

۳. Generate conflict resolutions: رزولوشن یک مفهوم است که کتاب مرجع برای مدیریت تضاد از آن استفاده می‌کند. هر راه‌حلی که به ذهن مهندس نیازمندی رسید باید کامل آن را بیان کند.

۴. Evaluate resolutions, select preferred: باید یکی از راه‌حل‌هایی که در مرحله پیشین ارزیابی کردیم را بررسی کنیم و بهترین آنها را انتخاب کنیم.

نکات

- همانطور که می‌دانیم Statements سبد ما می‌باشد و راه‌حل‌هایی که بدست می‌آوریم باید از جنس سبد باشد. اولین راه‌حل Drop کردن می‌باشد. همچنین از دیگر راه‌حل‌های تغییر جمله و سازگار کردن آن است، حتی ما می‌توانیم برای حل تضاد جمله به آن جمله‌ای مناسب را اضافه کنیم.
- مهندس نیازمندی باید در Intra-viewpointها بازه را تعیین کند.
- برای رفع تضاد ممکن است جمله‌ای را حذف، اضافه یا حتی تغییر دهیم. در این حین ممکن است بازم ایجاد تضاد صورت گیرد به همین خاطر ۴ قدم مدیریت تضادها به صورت چرخشی می‌باشد.

- ما باید عواملی که با هم تضاد دارند را بشناسیم که بتوانیم آن‌ها را مدیریت کنیم.
- باید بدانیم که کدام موارد باعث ایجاد تضاد می‌شوند و بعد از تغییر سبد می‌توانند دردسرزا شوند.
- معمولاً Conflictخیزها معمولاً در Overlapهای زیادی شرکت می‌کنند.
- جملاتی را باید استفاده کنیم که در Overlapهای زیادی شرکت داشتن و شرکتشان خوب و بدون تضاد بوده.

۵.۴ تکنیک‌های داکیومنت کردن

Statement	S1	S2	S3	S4	Total
S1	0	1000	1	1	1002
S2	1000	0	0	0	1000
S3	1	0	0	1	2
S4	1	0	1	0	2
Total	*1002*	*1000*	*2*	*2*	*2006*

S_{ij} =
1: conflict
0: no overlap
1000: no conflict

شکل ۷: تشخیص تضادها و همپوشانی‌ها

نکته

- باقی مانده نشان‌دهنده تضادهای بین دو جمله می‌باشد.
- خارج قسمت نشان‌دهنده همپوشانی مناسب و بدون تضاد است.

$$Conflicts(S1) = remainderOf(1002/1000) \rightarrow 2 \quad (۹)$$

$$nonConflictingOverlaps(S1) = quotientOf(1002/1000) \rightarrow 1 \quad (۱۰)$$

$$Conflicts(Total) = remainderOf(2006/1000) \rightarrow 6 \quad (۱۱)$$

$$nonConflictOverlaps(Total) = quotientOf(2006/1000) \rightarrow 2 \quad (۱۲)$$

۶.۴ تکنیک‌های رفع تضاد

برای رفع تضاد ۴ روش مطرح شده است که هر کدام از آن‌ها می‌توانند منجر به تولید نیازمندهای جدید شوند تا تضاد موجود در جمله را رفع کنند:

۱.۶.۴ خاص‌سازی منبع یا هدف تضاد

تضاد در سطح جمله رخ می‌دهد، یعنی یک قانون به کل وارد می‌شود که یکسری جزئیات دارد. نقض قانون بالایی به جز وارد می‌شود. هر کدام از جزئیات قوانین خودشان را دارند و چون جز هم قانون خودش را دارد باعث ایجاد تضاد می‌شود. قانون جدید (جمله جدید) در مورد کل سیستم نبوده و بلکه در مورد یک جز خاص می‌باشد. به جای اعمال قانون به کل سیستم بایستی به یک نود و قشر مشخص این قانون جدید اعمال شود. معمولاً در فاعل و مفعول رخ می‌دهد.
برای مثال:

- به کاربران (Users) اجازه داده شود که بتوانند از وضعیت کتابی که به امانت گرفته شده است مطلع شوند.
- دانشجویان نباید از وضعیت کتاب امانت گرفته شده مطلع باشند.

خاص‌سازی باید روی منبع یا رابطه کل به جز اعمال شود. در مثال بالا مشخص نیست که کاربران دقیقاً چه قشری هستند و آیا شامل قشر دانشجویان می‌شود؟ پس بایستی قانونی تعریف کنیم که مشخص شود چه گروهی قادر به مطلع شدن وضعیت باشند و چه گروهی نمی‌توانند. به همین دلیل مجوزهایی برای Staff users صادر می‌کنیم و مجوز دیگری به نام Students. در این دو گروه به روشنی می‌توان قابلیت‌هایشان را شخصی‌سازی نمود. گروه خاص ما مفعول بوده است. چه کسانی بتوانند و چه کسانی نتوانند؟

۲.۶.۴ ضعیف‌تر کردن جملاتی که تضاد دارند

در این روش معمولاً جمله سخت‌تر را ضعیف (Weak) می‌کنیم. دقیقاً جزئی که قانون را می‌بندد.
برای مثال:

- پدر می‌گوید ساعت ۱۰ شب خانه باش اما مادر شما می‌گوید که هر چقدر بودی مشکلی نداره، در این جمله تضاد قوی را مشاهده خواهیم کرد.
- قرض گیرنده کتاب، باید کپی کتاب را سر مهلت سه هفته‌ای تحویل دهد مگر اینکه یک مجوز (Permision) برای استفاده بیشتر کپی کتاب برای دانشجو صاد شود.
- مثال دقیق‌تر: دانشجوها می‌توانند تا سه هفته کپی کتاب را از کتابخانه قرض بگیرند اما در صورتی که عضو انجمن علمی دانشگاه باشند می‌تواند ۵ هفته کتاب را داشته باشند.

۳.۶.۴ ری‌استور کردن

در این روش تا زمانی که به تضاد بر نخورده‌ایم پیش می‌رویم و بعد از برخورد به تضاد سیستم را به حالت قبل از تضاد خواهیم برد.
برای مثال دانشجو می‌خواهد کتاب را بیشتر از ۳ هفته قرض بگیرد، اما کتابخانه تنها ۳ هفته امکان قرض گرفتن را برای دانشجو فراهم کرده است. برای حل این تضاد کتابخانه از ری‌استور کردن استفاده می‌کند و می‌گوید برای قرض گرفتن بیشتر از ۳ هفته، سر موعد مهلت قرض گرفتن کتاب را تمدید کن.

۴.۶.۴ پرهیز از شرایط مرزی

آخرین راه‌حل که سخت‌تر از بقیه می‌باشد این روش است که در مورد تضادهای ضعیف یا (Weak conflict) صادق خواهد بود. در این روش تلاش بر این است که شرایط مرزی را برای تضادها به گونه‌ای کنترل کنیم که هیچ وقت رخ ندهند تا هدف سیستم را از بین برود.
برای مثال، فرض کنید کتابخانه از یک کتاب مخصوص، تنها سه کپی دارد. اگر هر کدام از این سه کپی را سه دانشجو مختلف قرض بگیرد، دانشجوی چهارم نمی‌تواند این کتاب را درخواست کند. یعنی ریشه‌یابی یکسری کتاب که مرجع آن مشخص است که دیگر هیچ کپی از آن در کتابخانه موجود نیست به این صورت رسماً رسالت کتابخانه زیر سوال رفته است. سوالی که می‌تواند مطرح شود این است که آیا این نگرانی برای همه کتاب‌ها وجود دارد؟ باید این مسئله بررسی گردد که برای چه کتاب‌هایی نیاز داریم شرط جدیدی را وضع کنیم. برای

کامل کردن مثال، فرض کنید از آن کتابی که در ابتدای فرضمان سه کپی داشتیم تنها دو کپی قابل قرض دادن به دانشجو باشد و کپی آخر کتاب تنها زمانی قابل استفاده است که خواننده کتاب درون کتابخانه باشد و نخواهد آن را به بیرون از کتابخانه ببرد.

برای مثال بالا ممکن است به دنبال الگوریتم‌های دسته‌بندی برویم که بتوانیم رضایت را برای همه طرفین برقرار کنیم تا همه بتوانند از تمام کپی‌ها به صورت مناسب استفاده کنند.

در شرایط مثال بالا سیستم کتابخانه بسیار اساسی‌تر خواهد شد و باید در صفات کلاس مربوطه و نیازمندی‌ها خود اعلام کنیم که چند کپی از کتاب‌ها قابل قرض و چند کپی قابل استفاده در محل کتابخانه می‌باشد.

۷.۴ تمرین اول

در یک سیستم مانند اسنپ، مسافر می‌خواهد نزدیک‌ترین ماشین به او تخصیص داده شود، مدیر سیستم می‌خواهد در راستای طرح تشویقی خود رانندگانی با امتیاز بالاتر را به مشتری تخصیص دهد. آیا تضادی می‌بینید؟ اگر بله از چه نوعی است و راه‌حل آن چیست؟

بله تضاد دارند، دو جمله وجود دارد:

- کاربر به دنبال نزدیک‌ترین راننده اسنپ می‌باشد
- مدیر می‌خواهد راننده‌ای انتخاب شود که بالاترین امتیاز را داشته باشد.
- تضاد در جایی رخ می‌دهد که ممکن است راننده‌ای با امتیاز بالا در شعاع دورتری قرار داشته باشد.
- در این سناریو مشکلی برای راننده، مدیر و کاربر پیش نمی‌آید. پس تضاد ضعیف است. ما می‌توانیم با رویکرد Restore کردن این تضاد را به گونه‌ای پوشش دهیم که همه راضی باشند.
- لزوماً امتیاز راننده می‌تواند ۵ ستاره اولین شعاع نزدیک به کاربر نباشد براساس درخواست کاربر مشخص می‌شود که کدام راننده با امتیاز بالا بایستی فیلتر شود و از بین آنها کدام راننده می‌خواهد درخواست کاربر را بپذیرد. این بدان معناست که درخواست کاربر برای رانندگانی که در همان شعاع هستند که امتیاز آن‌ها کم باشد، ارسال نمی‌شود.

۸.۴ مدیریت ریسک

معمولاً در فازهای اولیه یک پروژه نرم‌افزاری، مهندسان نیازمندی و ذینفعان انتظارات عجیبی را دارند:

- محیط و نرم‌افزار همانگونه که انتظار دارند رفتار کند.
- برنامه توسعه نرم‌افزاری پروژه هماگونه که برنامه‌ریزی شده است رو به جلو باشد.

اما در حقیقت جا به جایی از System-as-is به Sysmte-to-be ممکن است دچار ریسک‌های مختلفی شود.

۱.۸.۴ شدت ریسک یا Severity

درجه از دست دادن رضایت نسبت به یک هدف را شدت ریسک یا Severity می‌گویند.

ریسک‌ها نقض (Not) یک نیاز می‌باشند. وقتی یک نیاز به درستی انجام نشود یا به هر دلیلی دیر انجام شود و نقض یک جمله باشد می‌توان گفت که به ریسک تبدیل شده است. نکته حائز اهمیت آن است که ریسک روی یک جمله می‌باشد و روی دو جمله مانند تضادها تاثیر ندارد.

ریسک‌ها به دو دسته تقسیم می‌شوند:

۱. مرتبط با محصول یا Product-related

۲. مرتبط با فرایند یا Process-related

۲.۸.۴ مرتبط با محصول یا Product-related

بیشترین ارتباط را به مهندس نیازمندی دارد. الزاماتی که در طراحی یک سیستم بایستی در نظر گرفته شود تا بتوانیم سیستم را در برابر آن‌ها تجهیز کنیم؛ لذا افرادی در حوزه مدیریت ریسک کار می‌کنند که متخصص آن دامنه و سیستم هستند. یعنی کاملاً در مورد دامنه تجربه و اطلاعات مناسب را دارا هستند. این افراد معمولاً جایگاه‌های ثابتی در دامنه خود داشتند. مانند سیستم‌های حسابداری بانکی، سیستم‌های CRM و غیره. تمام حالات سیستم را دیده‌اند و در استراتژی‌های مختلف در برابر ریسک‌های مرتبط را تجربه کرده‌اند. به عبارتی ساده‌تر یعنی به طور کل این افراد شناخت بسیار کاملی نسبت به آن دامنه دارند.
این ریسک‌ها می‌توانند مانند مثال‌های زیر باشند:

- ریسک در برابر ارسال و دریافت اطلاعات داخل برنامه‌ای؛ پیامی که ارسال می‌شود و قرار است به یک نفر برسد ریسک موارد زیر را دارد:
  - پیام برای آن شخص مشخص ارسال نشود و به تمام کاربران داخل شبکه بدون اجازه ارسال شود. (Broadcastly send)
  - پیام با تاخیر در شبکه ارسال شود و به دست دریافت کننده پیام برسد.
  - شبکه شنود شود و محتوای پیام را بتوان به صورت غیرقانونی در شبکه مشاهده نمود.
  - پیام قابلیت ویرایش پس از ارسال را داشته باشد.
- سیستم حاوی احراز هویت می‌باشد و ریسک آن:
  - اگر کاربر گذرواژه خود را فراموش کرده باشد؟ پس بایستی استراتژی مناسب در برابر این ریسک را در نظر بگیریم و برای این سیستم احراز هویت گزینه فراموشی گذرواژه را طراحی کنیم.

۳.۸.۴ مرتبط با فرایند یا Process-related

تمام اتفاقاتی که در ارتباط مستقیم با محصول نمی‌باشد را شامل می‌شود. برای مثال ممکن است ارزش پولمان کم‌تر شود یا یکی از اعضا/پرسنل‌مان استعفا دهد.اینگونه ریسک‌ها مرتبط با مدیر پروژه می‌باشد.

۹.۴ چرخه مدیریت ریسک

- فرایند پیدا کردن ریسک در پروژه‌های نرم‌افزاری یک فرایند تکرارپذیر می‌باشد که شامل سه مرحله زیر می‌باشد:

۱. Risk identification: شناسایی ریسک: دقیقاً ریسکی در سیستم به صورت مشخص رخ می‌دهد؟ یا اتفاق افتاده است؟

۲. Risk assessment: ارزیابی ریسک: آیا عواقب احتمالی بدی دارد؟ می‌توان از آن جلوگیری کرد؟ آیا می‌توان تاثیرات رخدادش را مدیریت و کنترل کرد؟

۳. Risk control: کنترل ریسک: مدیریت و کنترل ریسک به عنوان نیازمندی جدید

- در این میان نکته بسیار مهم آن است که در چرخه تکرار بررسی ریسک ممکن است هر عملیات و اقداماتی منجر به ایجاد ریسک جدیدی شود.
- مدیریت ضعیف ریسک‌ها عامل اصلی شکست در پروژه‌های نرم‌افزاری می‌باشد.

۱. اشتباه فکر کردن به جریانات پروژه که انگار قرار نیست هیچ فرایندی مشکل داشته باشد.

۲. عدم شناسایی و دستکم گرفتن ریسک‌ها که باعث ناقص و ناکافی در نظر گرفتن نیازمندی‌ها در پروژه شود.

۱۰.۴ شناسایی ریسک

در این قسمت به چهار تکنیک شناسایی ریسک در پروژه‌های نرم‌افزاری می‌پردازیم:

۱.۱۰.۴ چک لیست‌های ریسک

بررسی چک لیست‌های ریسک هم می‌تواند در مورد ریسک‌های محصول باشد و هم در مورد فرایندها. برای مثال حوزه مالی اولین حوزه‌ای نیست که قبلاً وجود نداشته باشد و دقیقاً این اولین سیستمی باشد که از قابلیت‌های مالی استفاده می‌کند. در حقیقت این حوزه از قبل چندین بار مورد استفاده قرار گرفته شده است و توسط متخصصان مختلفی مورد آزمون و تلاش بسیاری بوده که توانسته به بیشتر چالش‌ها و ریسک‌های آن پاسخ دهند. به این ترتیب می‌توانیم تمام ریسک‌های آن را از قبل تهیه کنیم و بتوانیم در سیستم خود آن‌ها را بررسی کنیم که اگر برخی قابلیت‌ها منجر به تولید ریسک شد بتوانیم راهکاری برای آن طراحی و پیاده‌سازی کنیم. در حقیقت یک لیست راهنما از پیش تعیین شده می‌باشد.

برای سناریو زیر می‌توان تمام ریسک‌ها را از قبل پیشبینی کرد و لیستی از بایدها را برای آن به شکل زیر بررسی می‌کنیم:
وقتی ارسال کننده پیام بخواهد پیامی را برای دریافت کننده‌ای ارسال کند ریسک‌های احتمالی موارد زیر خواهد بود:

- درگاه پیام تغییر کند: استفاده از رویکردی امن.
- پیام در شبکه شنود شود: رمزنگاری و استفاده از شبکه‌های توزیع شده.
- تاخیر در ارسال پیام: بررسی زیرساخت‌های شبکه‌ای و حتی الگوریتم‌های ارسال و دریافت پیام.

نکته: محصولات همگی مشخص هستند و فرایندهای آن‌ها کاملاً عمومیت دارد.

۲.۱۰.۴ بازبینی مولفه‌ها

مولفه‌ها مخصوص محصول می‌باشد؛ المان‌ها در حقیقت همان مولفه‌ها هستند. آدم‌ها، نرم‌افزارهای موجود و نرم‌افزارهایی که قرار است توسعه داده شود، تماماً المان محسوب می‌شوند. برای مثال در سیستم قطار المان سرعت‌سنج را مورد بررسی قرار می‌دهیم تا ریسک‌هایش را متوجه شویم:

- آیا می‌تواند وظیفه‌اش را به درستی انجام ندهد؟ بله ممکن است. وظیفه آن بررسی حرکت قطار و سرعت آن است. عوامل مختلفی وجود دارد که می‌تواند سبب درست کار نکردن و یا توقف کار کردن آن شود.
- اگر سرعت فیزیکی قطار با سرعت اندازه‌گیری شده برابر نباشد یعنی این دستگاه مشکلی دارد.
- یکی از ریسک‌ها آن است که در حین حرکت یکی از مسافرها اقدام به خراب کرد دستگاه کند.

یا مثال اپلیکیشن‌های وب و موبایل:

- اگر کاربر به اشتباه دستش روی گزینه پاک کردن بخورد جا به جا مورد انتخاب شده حذف شود یک ریسک در نظر گرفته می‌شود.
- برای این ریسک طراحی دیالوگ را می‌توان در نظر گرفت که از کاربر تایید مجدد برای انجام کار خودش گرفته شود.
- همچنین بعد از طراحی و پیاده‌سازی دیالوگ می‌توانیم در مورد پیاده‌سازی قابلیت لیست موارد پاک شده بپردازیم؛ یعنی سیستم را به گونه‌ای بنویسیم که قابلیت حذف آن به صورت Soft delete باشد.

همه ریسک‌ها را نمی‌توان مدیریت کرد بلکه باید برخی از آنها پذیرفته شود. به همین خاطر هر سیستمی بنا به مقدار آستانه تحمل خودش ریسک‌ها را می‌پذیرد. ریسک‌ها را بررسی می‌کند اگر از مقدار آستانه کوچک‌تر بود مدیریتش را به کاربران می‌سپارد. بارزترین مثال سیستم انتخاب واحد آموزشیار که به روشنی امکانات لود بالانس کردن درخواست‌های کاربران زیاد را با پایین‌ترین کیفیت می‌تواند مدیریت کند. به خاطر اینکه به کل سیستم صدمه‌ای وارد نمی‌کند، طراح سیستم از مدیریت این ریسک صرف‌نظر می‌کند.

۳.۱۰.۴ تعریف عواقب یا Consequence

تمام عواقب Consequence یک ریسک بایستی برآورد و بررسی شود. اگر نتوانیم یک ریسک را مدیریت کنیم پس ممکن است رخ دهد. بعد از رخ دادن آن باید عواقب و تاثیرات (Side effectهای) آن را در نظر بگیریم که چقدر می‌تواند مضر باشد و به سیستم صدمه وارد کند. نوشتن تمام عواقب یک ریسک می‌تواند در کاهش نارضایتی‌ها تاثیرگذار باشد.

آیا سیستم‌ها و سازمان‌ها از یک آستانه تحمل یکسان و مشخصی استفاده می‌کنند؟

خیر؛ هر سازمانی تحت شرایط و پروتکل‌های خاص خودش کار می‌کند و هر کسی نمی‌تواند طبق میل و اراده خودش عمل کند. یک سازمان بررسی می‌کند که این مقدار آستانه چقدر ارزش دارد.

پرسیدن چهار سوال زیر برای بررسی مولفه‌های ریسک الزامی می‌باشد؛ بایستی مولفه‌های خیلی بزرگ را کوچک کنیم تا بتوانیم به سادگی به پاسخ سوالات زیر برسیم.

نکته: اینکه یک سناریو تبدیل به ریسک شود و به وقوع بپیوندد بایستی عواقب بعد از آن را کنترل کنیم. مهندس نیازمندی لازم است که آثار ریسک را کمتر کند یا حداقل بتواند آن‌ها را مدیریت کند. قدم‌های مدیریت ریسک متفاوت می‌باشد.

۱. Can it fall? – آیا امکان رخ دادن همچین رسیکی وجود دارد؟

۲. How? – بیشتر برای پیامدها و بعد از درد رخ دادن ریسک است.

۳. Why? – چرا همچین ریسکی به وجود آمده است؟ دقیقاً به ریسک اشاره دارد.

۴. What are possible consequences? – پیامد و عواقبی که ممکن است به همراه داشته باشد چه مواردی هستند؟ آیا از آن‌ها می‌توان چشم‌پوشی کرد؟ یا بایستی برای رفع آن‌ها هزینه‌ای داشته باشیم و راهکاری مناسب را ارائه دهیم؟

۴.۱۰.۴ درخت ریسک

به جزئیات این تکنیک در فصل نهم بیشتر پرداخته می‌شود.

تمام نودهای درخت ریسک‌ها هستند. از ریشه شروع می‌کنیم و به ریسک‌های کوچک‌تر شکسته می‌شود. برای مثال:

- اگر خانه آتش بگیرد:
  ۱. انفجار گاز
  ۲. اتصالی سیم برق داخل ساختمان
  ۳. یعنی یک اتفاق بزرگ و بد (آتش گرفتن یک خانه) می‌تواند چند عامل کوچک در رخ دادن آن تاثیر داشته باشند.

چه زمانی نیامند کشیدن درخت ریسک هستیم؟

زمانی که ریسک به اندازه‌ای بزرگ و پیچیده باشد که نتوانیم آن را بفهمیم و درک کنیم، نیازمند آن هستیم که با استفاده از درخت ریسک، یک ریسک را به عوامل مهم و تاثیرگذارش بشکنیم تا ببینیم می‌توانیم آن را کنترل کنیم یا با سطح آستانه تحمل سیستم ما حل خواهد شد.

نکات

- در این بین بعضی از معیارهایی که در درخت مشخص می‌کنیم ممکن است به صورت آماری باشند؛ یعنی طی ۵۰ سال مثلاً دوبار خانه آتش گرفته است.
- در تمام صنایع ریسک وجود دارد.
- ریسک‌ها را یا با مستطیل نمایش می‌دهیم یا با بیضی.
- اگر ریسک نیاز به شکستن داشته باشد آن را با نماد مستطیل نمایش می‌دهیم.
- اگر به کوچک‌ترین حالت ریسک رسیده باشیم یعنی آن را بتوانیم کامل به ساده‌ترین روش درک کنیم و نیاز به شکست نداشته باشد از نماد بیضی استفاده می‌کنیم.
- نمادهای دیگری مانند AND و OR در این درخت استفاده می‌شوند.

- برگ‌ها و نودهای پایانی درخت همیشه با نماد بیضی نمایش داده می‌شوند.
- برای آنکه پیدا کردن ریشه اتفاقاتی که رخ می‌دهد، ساده‌تر باشد از درخت ریسک استفاده می‌کنیم تا بتوانیم اتفاق بالایی را شناسایی کنیم و سپس بعد از آن با الگوریتم Cutset درخت را ساده‌تر می‌کنیم.
- میزان بزرگی توپ، بزرگی خطر را نشان نمی‌دهد.
- پارامتر دیگر در بررسی بزرگی توپ ریسک احتمال وقوع هر عاقبت می‌باشد.
- اول عواقب پیدا می‌شود و سپس احتمال هر عاقبت سنجیده می‌شود.
- احتمال وقوع ریسک اگر کم باشد نمی‌توانیم بگوییم که میزانش نیز کمتر بوده است.
- احتمال وقوع ممکن است کم باشد اما اگر رخ بدهد ممکن است سیستم را از کار خارج کند.

۵.۱۰.۴ فلسفه درد

درد صفر شدنی نیست اما قابل کم شدن است. چون اگر عواقب هم مورد بررسی قرار گیرد می‌تواند درد داشته باشد؛ اگر درد نداشته باشد باید شک کنیم که آیا ریسک روی سیستم ما رخ داده است؟ یا روی سیستم دیگری بوده؟ جعبه کمک‌های اولیه بهبود و کاهش اثر اتفاقاتی است که مدتی است که رخ داده.

۶.۱۰.۴ نکات گره‌های AND و OR

دقیقاً همانند مدار منطقی عملگرهای زیر به این شکل کار می‌کند:

- نود AND: تمام زیر نودها بایستی اتفاق بیوفتند تا نود والد رخ دهد.
- نود OR: تنها نیاز است یک نود رخ دهد تا اتفاق والد رخ دهد.

شرط‌های Cutset کردن یک درخت ریسک نوشته شود.
درخت‌های اولیه و Cutset در این قسمت نوشته شود.

۷.۱۰.۴ استفاده از تکنیک‌های جمع‌آوری داده

با افراد و آدم‌ها در ارتباط می‌باشد. کسانی که ذینفع هستند در قسمت استخراج اطلاعات می‌توانند نقش پر رنگی را ایفا کنند. اگر از متخصصین این حوزه سوال بپرسیم می‌توانیم رخدادها و پیامدهای مختلفی که می‌تواند یک ریسک داشته باشد را یاد بگیریم. مهم‌ترین راه‌ها: مانند استفاده از روش Interview و روش Group session.

نکته: همان‌گونه که راه‌حل رفع تضاد رزولوشن بود، برای حل و کنترل ریسک از اقداماتی مخصوص استفاده می‌کنیم تا بتوانیم به نسبت متعادلی ریسک را کنترل کنیم و حتی آن را رفع کنیم (استفاده از تکنیک‌های Countermeasures که در بالاتر گفته شد).

در این روش ابتدا راه‌حل را ارزیابی می‌کنیم و سپس بهترین راه‌حل را نسبت به بقیه انتخاب و در سیستم در حال طراحی (System-to-be) استفاده می‌کنیم.

۱۱.۴ ارزیابی ریسک یا Risk assessment

هدف اصلی از ارزیابی ریسک، ارزیابی احتمال خطر + شدت، احتمال عواقب برای کنترل خطرات بالا با اولویت بالا می‌باشد.
متغیرهایی که می‌توانیم برای ارزیابی کیفی مورد استفاده قرار دهیم:

- برای احتمالات: (بسیار محتمل، محتمل، ممکن، بعید و...)

- برای شدت: (فاجعه‌بار [۲۰]، شدید [۲۱]، بالا، متوسط و....)

[۲۰] Catastrophic
[۲۱] Severe