

# مهندسی نیازمندی‌ها خانم دکتر سپیده آدابی

علیرضا سلطانی نشان

۲۳ فروردین ۱۴۰۳

## فهرست مطالب

۳	۱ مقدمه
۳	۱.۱ مهندسی نیازمندی
۳	۱.۱.۱ تعریف
۴	۲.۱ نکته تجرید
۴	۳.۱ متدولوژی
۴	۴.۱ دلیل متدولوژی‌های مختلف
۵	۵.۱ ماهیت مدل
۵	۶.۱ الگو
۵	۷.۱ استاندارد
۶	۸.۱ مهندسی نیازمندی
۶	۹.۱ دلیل استفاده از زبان UML
۶	۱۰.۱ بررسی شروع کار مهندسی نیازمندی
۶	۱۱.۱ بررسی UML to goal
۶	۱.۱.۱.۱ نمودار هدف
۶	۲.۱.۱.۱ نمودار ریسک
۶	۳.۱.۱.۱ نمودار Agent
۶	۱۲.۱ مهندسی نرم‌افزار و مهندسی نیازمندی
۷	۱۳.۱ مهندسی نیازمندی و مدیریت نیازمندی
۷	۲ فصل اول
۷	۱.۲ اصطلاحات
۷	۱.۱.۲ Environment یا World problem
۷	۲.۱.۲ Machine
۸	۳.۱.۲ Context
۸	۴.۱.۲ Statement یا جمله
۸	۵.۱.۲ Phenomena یا پدیده‌ها
۸	۶.۱.۲ System as is
۸	۷.۱.۲ System to be
۸	۸.۱.۲ Prescriptive عوامل

۹	۹.۱.۲ مفروضات یا Assumption
۹	۱۰.۱.۲ مثال
۱۰	۱۱.۱.۲ مفهوم Definition
۱۰	۱۲.۱.۲ مفهوم مانیتور کردن
۱۰	۱۳.۱.۲ مفهوم کنترل کردن
۱۰	۱۴.۱.۲ عوامل Descriptive
۱۰	۱۵.۱.۲ ویژگی دامنه یا Domain property
۱۰	۱۶.۱.۲ دامنه‌ها
۱۱	۱۷.۱.۲ اسکوپ‌ها
۱۱	۱۸.۱.۲ تفاوت‌های بین Prescriptive و Descriptive
۱۱	۲.۲ مولفه‌های مربوط به نیازمندی نرم‌افزار در نیازمندی سیستم
۱۲	۳.۲ توافق بر لغات
۱۴	۴.۲ دسته‌بندی نیازمندی‌ها
۱۴	۱.۴.۲ Functional requirement
۱۴	۲.۴.۲ Non-functional requirement
۱۵	۵.۲ کیفیت سرویس‌دهی یا QoS (محصول)
۱۵	۶.۲ Service Level Agreement
۱۵	۷.۲ تفاوت بین Limitation و Constraint
۱۵	۸.۲ مفهوم هنجارها یا Compliance (محصول)
۱۵	۹.۲ قیدهای معماری Architectural constraint (محصول)
۱۵	۱۰.۲ قیدهای توسعه Development constraint (مدیر پروژه)
۱۶	۱۱.۲ فرایند و مراحل مهندسی نیازمندی
۱۶	۱.۱۱.۲ پیشنهادات جایگزین، درک دامنه و جمع‌آوری داده‌ها
۱۶	۲.۱۱.۲ نیازمندی‌های توافق شده، ارزیابی و توافق
۱۷	۳.۱۱.۲ سند نیازمندی‌ها، اولویت‌بندی و مستندات
۱۷	۴.۱۱.۲ نیازمندی‌های ترکیبی، تایید و اعتبارسنجی
۱۷	۱۲.۲ نیازمندی‌ها در چرخه توسعه نرم‌افزار
۱۷	۱۳.۲ Request for Proposal یا RFP
۱۷	۱۴.۲ تعریف: به اجماع رسیدن مطالب از سند نیازمندی
۱۸	۱۵.۲ تأثیراتی که سند نیازمندی به فرآورده‌های نرم‌افزاری دارد
۱۸	۱.۱۵.۲ Prototype
۱۸	۲.۱۵.۲ Project estimations (Size, Cost, Schedules)
۱۸	۳.۱۵.۲ Acceptance test
۱۸	۴.۱۵.۲ Architectural design
۱۸	۵.۱۵.۲ Software quality assurance

### ۳ فصل دوم، درک دامنه و جمع‌آوری نیازمندی‌ها

۱۹	۱.۳ دسته‌بندی جمع‌آوری داده
۱۹	۲.۳ تکنیک‌های جمع‌آوری اطلاعات فرآورده‌گرا
۱۹	۱.۲.۳ Background study
۲۰	۲.۲.۳ Data collection, questionnaires

۲۱	Repertory grids, Card sorts for concept acquisition	۳.۲.۳
۲۲	Scenarios, Storyboards for problem world exploration	۴.۲.۳
۲۳	Prototypes, Mock-ups for early feedback	۵.۲.۳
۲۳	Knowledge reuse: Domain-independent, Domain specific	۶.۲.۳
۲۴	Interviews	۷.۲.۳
۲۴	Observation and ethnographic studies	۸.۲.۳
۲۴	Group sessions	۹.۲.۳

## مجوز

به فایل license همراه این برگه توجه کنید. این برگه تحت مجوز GPLV۳ منتشر شده است که اجازه نشر و استفاده (کد و خروجی/pdf) را رایگان می‌دهد.

## 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<sup>۱</sup> باشد. پایداری به منظور بررسی تغییرات از جوانب مختلف مانند امنیت و آزمون عملکرد صحیح می‌باشد. نیازمندی‌ها کاملاً پر در درسر هستند زیرا خیلی از دلایل شکست پروژه‌ها عدم بررسی نیازمندی‌ها 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<sup>۲</sup>: دقیقاً آن چیزی است که سیستم بایستی قادر به انجام آن باشد. مثلاً کاربر باید بتواند در نرم‌افزار لاگین کند.
۲. کلمه چطور<sup>۳</sup>: 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<sup>۴</sup>: عوامل محیطی (افراد، دستگاه‌ها، نرم‌افزارهای آماده) دخیل در برنامه

زمانی که می‌گوییم نرم‌افزار ثبت نام درس، دقیقاً بالاترین سطح تجرید<sup>۵</sup> را در نیازمندی بیان کرده‌ایم.

---

Stable<sup>۱</sup>  
What<sup>۲</sup>  
How<sup>۳</sup>  
Who<sup>۴</sup>  
Abstract<sup>۵</sup>

## 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<sup>۶</sup> 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.
- 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 تکنیک‌ها 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

متدولوژی<sup>۷</sup> یک جهان‌بینی کلی، در تولید نرم‌افزار است (دید از بالا برای انجام کارها و وظایف). تمام متدولوژی‌ها را برای تولید استفاده می‌کنند و تمام راهنمایی‌ها توضیحات دارند. در حقیقت تمام متدولوژی‌ها از خواستگاه تولید نرم‌افزار ایجاد 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 پیچیده است، پس در این جهت متدولوژی‌های مختلفی برای مهار آنها ارائه شده است که نوع تولید را متفاوت می‌کند. متدولوژی بایستی کاملاً منعطف باشد. مراحل و فرایندها در متدولوژی‌ها متغیر می‌باشد.

---

<sup>۶</sup>Trade off  
<sup>۷</sup>Methodology

## 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)

### 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 تکرار بودن آنها در پروژه‌های مختلف است.

## ۷.۱ استاندارد

مجموعه‌ای از قواعد<sup>۸</sup> 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.

---

<sup>۸</sup>Rules

## 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.

#### ۱.۱۱.۱ نمودار هدف

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

#### ۲.۱۱.۱ نمودار ریسک

ریسک‌ها 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<sup>۱۰</sup> 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

<sup>۹</sup> Goal diagram

<sup>۱۰</sup> Discipline

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

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 Environment یا World problem

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<sup>۱۱</sup>. 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.

<sup>۱۱</sup> Software to be

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 علمی 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 نقد قرار گیرند.

#### 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 Prescriptive عوامل

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

۱. System requirement: یک System requirement مجموعه‌ای از Assumption ها و Software requirement ها است. 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.

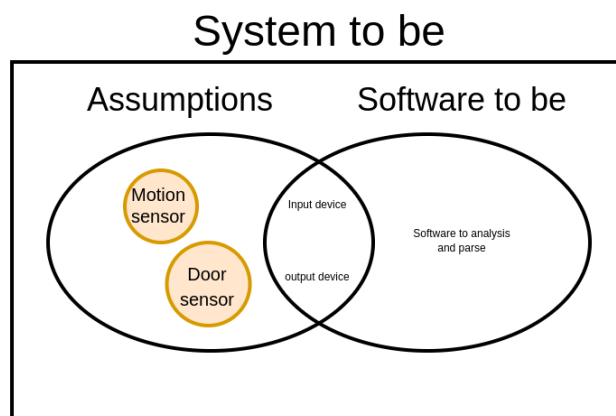
۱. People: مردم و کاربران
۲. Device: دستگاه‌ها مانند سنسورها، جمع‌آور داده و ارسال کننده به موتور تحلیل (نرم‌افزار)
۳. Exists softwares نرم‌افزارهای موجود: نرم‌افزارهایی که خودشان عملیات متعددی انجام می‌دهند و داده‌ها را برای تحلیل به نرم‌افزار اصلی سیستم ما ارسال می‌کنند.

عوامل محیطی گسترده هستند. برای مثال وقتی که کاربر در اپلیکیشن سبد خرید خود را می‌خواهد حساب کند، زدن روی دکمه "پرداخت آنلاین" کاملاً 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 Assumption ها سنسور تشخیص حرکت قطار و محرک بازوی درهای قطار هستند که دائماً در حال مانیتور و کنترل در و حرکت قطار هستند.
- Assumption ها 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 غیر

#### 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.

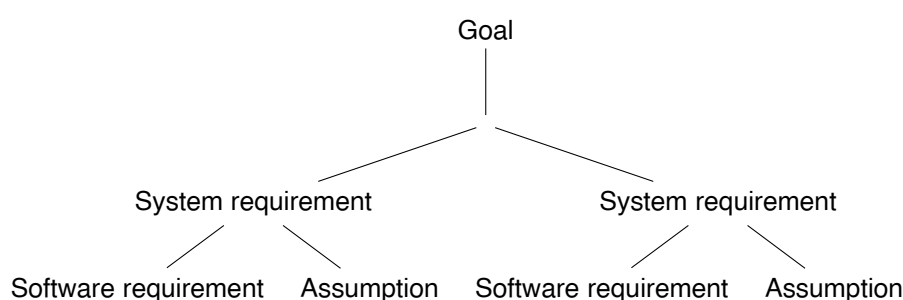
#### ۱۶.۱.۲ دامنه‌ها

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

مجموعه‌هایی از System requirement هستند که نرم‌افزار می‌تواند در آنها ورود داشته باشد. مثلاً فعالیت‌های مربوط به ثبت نام 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 دستگاه‌ها 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.
- بعد Why در نمودار معمولاً نشان‌دهنده اهداف است. مثلاً 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 Prescriptive و Descriptive

- 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 نرم افزار آنها را می تواند از طریق دستگاه های خروجی (Actuators) آنها را کنترل (اعمال) کند.

۳. مقادیر دستگاه های ورودی<sup>۱۲</sup>: تمام داده هایی که به عنوان ورودی در نرم افزار استفاده می شود.

۴. متغیرهای خروجی<sup>۱۳</sup>: مقادیری که نرم افزار آنها را در دستگاه های خروجی اعمال می کند.

## نکته

بیشتر سازمان ها به دو دسته زیر فعالیت های خودشان را انجام می دهند:

۱. سازمان هایی که هدفگرا هستند و تنها برای رسیدن به محصول آخرین تلاش و فعالیت خود را می کنند.

۲. سازمان هایی که تعداد Agent و کاربرانشان زیاد است و ارزش های زیادی برای آنها قائل می شوند به صورت گرا Agent یا عاملگرا هستند.

سطح System requirement بالا می باشد، چرا که مشتری تنها درخواست می کند که می خواهد چنین قابلیت هایی 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

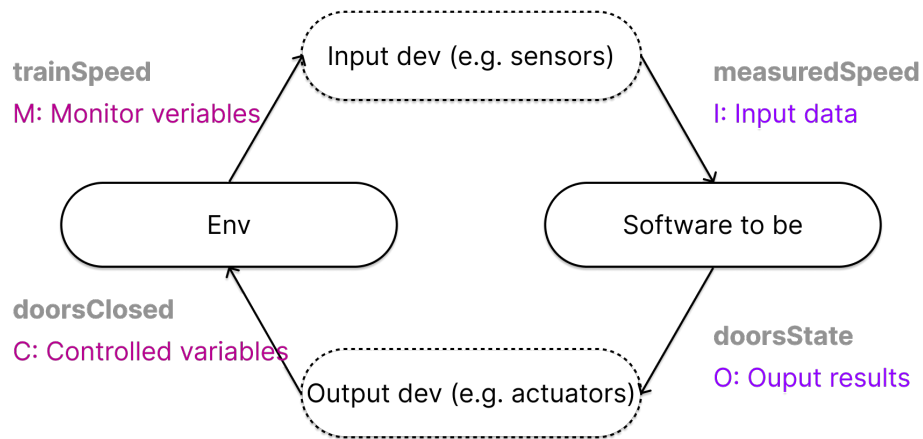
۳. DOM: منظور دامنه یا Domain

$$SOFTREQ + ASM + DOM \rightarrow SYSTEMREQ \quad (3)$$

اگر نیازمندی نرم افزار، مفروضات و دامنه ها همگی مقید و راضی باشند نیازمندی سیستم نیز بدست می آید. با استفاده از پارامترهای بالا می توان به سیستم نهایی رسید.

---

Input<sup>۱۲</sup>  
Output<sup>۱۳</sup>



شکل ۲: ارتباط نیازمندی سیستم در نرم افزار به همراه استدلالها

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

## استدلال سناریو

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

$$ASM1 : measuredSpeed \neq 0 \text{ if } trainSpeed \neq 0 \quad (۵)$$

$$ASM2 : doorsState = "closed" \text{ if } doorsClosed \quad (۶)$$

$$DOM : trainMoving \text{ if } trainSpeed \neq 0 \quad (۷)$$

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

## 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 آلام 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 Limitation و Constraint

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 تکنولوژی 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

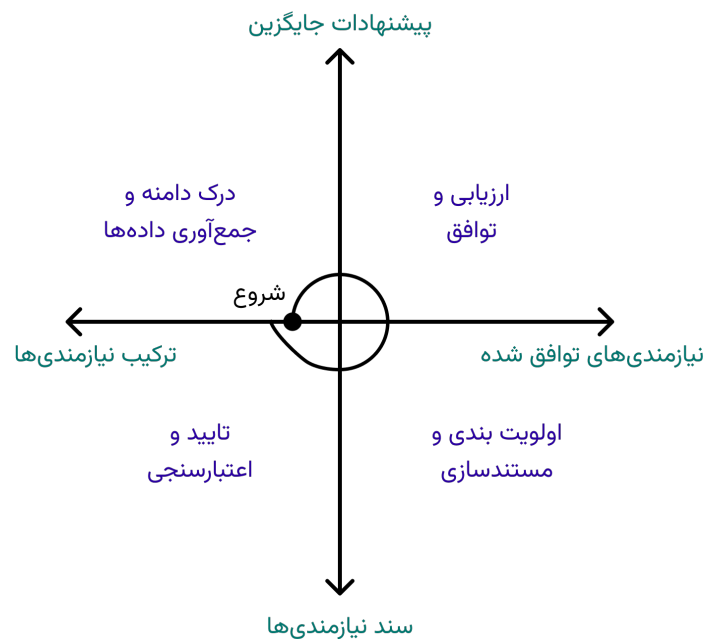
۲. قید توزیع: ورودی و خروجی از دو درب مختلف در دانشگاه، به دلیل آنکه داده‌های محیطی ورودی و خروجی در دو محل متفاوت است برای رسیدن به توافق در این توزیع باید این داده‌ها را در یک جا با هم سینک کنیم تا اطلاعات ورودی و خروجی مناسب یکدیگر پدید آید.

## 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 صرفاً 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 نیازمندی‌های 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 نیازمندی‌های مشتری به اجماع رسیده‌ایم. یک سبدي از Scope ها که خیلی آشفته بود به یک سبدي تبديل 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

سبدي 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 کنند.

### 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 استارت‌آپی 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 می‌خواهیم در آینده داشته باشیم (System-to-be) به شکل اهداف<sup>۱۴</sup>، قید و بندها<sup>۱۵</sup>، مفاهیم ارجاع داده شده، تسک‌ها و تکالیف مشخص شده، نیازمندی‌ها، فرضیات<sup>۱۶</sup> و ویژگی دامنه‌های مربوطه تعریف شده است.

<sup>۱۴</sup> Objectives

<sup>۱۵</sup> Constraints

<sup>۱۶</sup> Assumption

## ۱۵.۲ تاثیراتی که سند نیازمندی به فرآورده‌های نرم‌افزاری دارد

### ۱.۱۵.۲ Prototype

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 سبب 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.

### ۲.۱۵.۲ Project estimations (Size, Cost, Shedules)

یکی از نیازمندی‌های غیرعملیاتی مربوط به توسعه است که روی سبب 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.

### ۳.۱۵.۲ Acceptance test

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 نیازمندی‌ها طراحی و آماده می‌شود.

### ۴.۱۵.۲ Architectural design

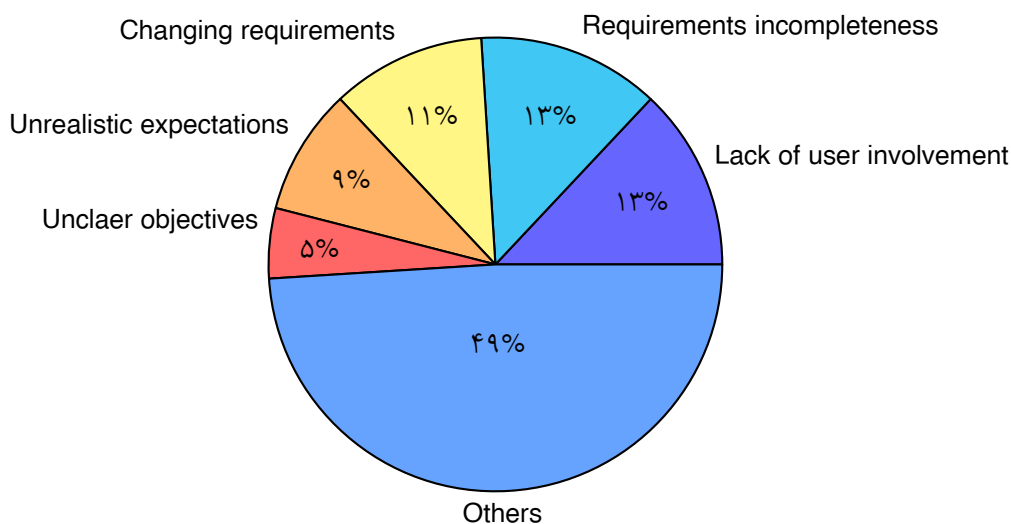
طراحی معماری نرم‌افزار به طور مشخص با نیازمندی‌های نرم‌افزار در ارتباط است که ممکن است تاثیر به سزایی بر روی نیازمندی‌های Non-functional داشته باشد. بر این اساس، سند نیازمندی‌ها ورودی اساسی برای طراحی معماری نرم‌افزار از دیدگاه‌های زیر می‌باشد:

- 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

### ۵.۱۵.۲ Software quality assurance

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 Assumption 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<sup>۱۷</sup>

<sup>۱۷</sup> Meeting minutes

- دامنه‌ها: کتاب‌ها، نظرسنجی‌ها، مقالات، مقررات و استانداردها، گزارش‌های سیستم‌های مشابه در دامنه مشابه
- سیستم کنونی یا System as is: جریان‌های 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 نامرتب برای مثال بخش بایگانی اسنادی را نگهداری می‌کند که ممکن است کاملاً با یکدیگر نامرتب 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

## پرسشنامه

لیستی از سوالاتی که توسط ذینفعان مشخص شده را آماده می‌کنیم که هر کدام یک جواب مناسب را می‌تواند در برگرد. 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<sup>۱۸</sup> 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.

<sup>۱۸</sup> 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 راه حل است.

## نکته

کاملاً بستگی به نیاز سیستم دارد که مشخص کنیم یک اسم کلاس باشد یا نه. زیرا یک اسم هم 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 کوثری‌هایی 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 بینیم 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.

## گریدبندی کردن کارت‌ها

گرید بندی کردن کارت‌ها  
کلاس با محیط مسئله و طراح همراه است  
کتگوری کردن از طراح اطلاعات اجمالی از ویژگی‌ها محیطی است.  
فضای مسئله: محیطی است یعنی همینی که هست. وضعیت موجود را نمایش می‌دهد. فضای راه حل رو طراح اگه نیاز باشه مطرح می‌کنه.  
شماره دانشجویی چون بایستی از اطلاعاتی محیطی طراحی.  
در حقیقت خروجی این پوت‌های ماست  
افزودن value که مهندس نیازمندی آن را در نظر می‌گیرد.  
attribute رنج دارد. تاپیش چیه. اندازش چیه.  
سادست ارزانه ولی خیلی از این جمله‌ها می‌تواند دقت پایین داشته باشد، نامرتب باشد و یا ممکن است به اسکوپ ما مرتبط نباشه.  
کسایی می‌تونن هندل کنن که اسکوپ رو خیلی خوب درک کرده باشن.

## ۴.۲.۳ Scenarios, Storyboards for problem world exploration

سناریو، داستان است که می‌تواند شرایط کنونی و شرایط آینده را بگوید. یعنی system-to-be system-as-is سناریو جاهایی استفاده می‌شود که پیچیدگی سیستم زیاد است. مهمان بودن دانشجو کاملاً داستان پیچیده‌ای دارد. اهمیت سناریوها به شدت بالاست. چه کاری کی انجامش می‌ده چرایی کار توضیح چرایی یا if what چه می‌شود اگر. دیدن همان شرایط خاص Exception سناریو را در sequence دیاگرامی مشخص می‌کند که در آن what وجود دارد، if what وجود دارد ولی why ندارد. متوازی الاضلاع چرایی سیستم را مشخص می‌کند. در خیلی سناریوها انواع سناریوها را موقعی که در sequence نشان می‌دهیم در حقیقت بعد why را ندارد.

منفی: کارایی که سیستم نباید بکنه، روزهایی که در تقویم هر کدام افراد سازمان نمی‌تونن شرکت کنند باید کار محرمانه باشه. کارایی که نباید انجام بشه. مثبت: رفتاری که انتظار داریم. لیست دروس قابل ارائه شده. چه کارایی انجام می‌ده. نرمال: مجموعه منفی و مثبت‌ها

را بیان می‌کنیم بعد if what ندارد. اول نرمال سناریو رو می‌نویسم بعد برای what ها از غیرنرمال‌ها انجام میدم. در مورد استیت صحبت میکند. غیرنرمال: اگه همه چی خوب و خوش نبود باید سناریو غیرنرمال رو بگیرم. تمام چیزایی که اونطوری که ما انتظار داریم پیش نمیره. یه آینده نگری برای سیستم می‌باشد. در مورد استیت صحبت میکند. مزایا و معیاب سناریو: بعد why در سناریو وجود نداره انگار حذف شده. قصه گفتن سخته. پیچیدگیش بالاست. حسنی که دارد اینه که سادست.

### ۵.۲.۳ Prototypes, Mock-ups for early feedback

یه بخش‌هایی باید پروتوتایپ باشه. توش مشخص می‌کنیم که چه قابلیت‌هایی داریم. می‌تونه ui مستقیم ترین فرایورده ایک ه برای استخراج نیازمندی‌های استفاده می‌شود. عملیاتی در این قسمت بیشتر دیده میشن. اطلاعات تکمیلی: سرچ همزمان با چند چیز یک کار فاکشناله ولی نان فاکشنالیش توی یوزر اینترفیسه. فوکسش برای بحث روی بدست آوردن نیازمندی‌هایی گنگ بوده. کار تقریبا سریع تا نشون بدیم چالش‌های سیستمی چی بوده. سرچ شود که سخت افزار می‌تواند اسم prototype را بگیرد یا خیر. استاد میگوید. testbed.

### ۶.۲.۳ Knowledge reuse: Domain-independent, Domain specific

بازیابی دانش در چه حوزه‌ای؟ دانش در مورد سازمان، دامنه و اسکوپ‌ها رو می‌خوایم. بانکداری‌ها معمولا یه سری رول‌های مشابه ندارند. (پوشا) نمایش دانش، المان‌های دانشی که تشکیل میدن رو شناسایی کنیم. المان‌های دانشی از جنس زیر هستند: کانسپت اهداف تسک‌ها افراد نیازمندی‌ها دامنه‌ها دانش باید نمایش داده بشه. روش نمایش گرافیکی است. به صورت تکست هم هست اما رسمی نخواهد بود. در خیاطی نمایش دانش را الگو می‌گویند. نمایش گرافیکی برای تسکمون. سه مرحله: مثال خیاطی: retrieve دانش مناسب را از بقیه سیستم دریافت کنیم. ممکن الگوی کاملی وجود نداشته باشه پس ماژپلار براساس الگو پیش می‌رویم. سیستم ثبت نام باشگاه: آیا عضو گیری داریم. تنها برای باشگاه نیست بلکه یه کاری است که تکرار آن زیاد است. پس باید المان‌های دانشی وجود داشته باشه که بگه بخش‌هایی داره دانشی وچه دامین‌هایی داره چه کانسپت‌هایی داره. راه حل مشترک برای خانواده مشترکی از مسائل قدم اول پیده کردن الگو است. قدم دوم: transpose ماهیتی جنس‌ها فرق دارد،، الگو راهنما است، یسری چیزا به درد می‌خوره. یه سری چیزا باید تبدیل بشه. مثال فروش راکت تنیس نسبت به فروش پنیر که ماهیتی فرق دارد. قدم آخر Validate اعتبار سنجی و تطبیق الگو با نیازهامون. الگوها بایستی با یکدیگر تطابق ببینند چون ممکن است یه تغییری داشته باشه تا بتونن کنار همین دیگه قرار بگیرن.

transpose به سه طریق انجام میشه. چرا که سخت‌تره:

نمونه سازی: Instantiation: مستقل از دامنه خاص سازی: Specialization: دقیق در مورد دامنه می‌گه. : specific Domain عمل واجب برای بیان فرمول با کلمات سیستم: reformulation  
نمونه سازی در سازمان تطبیق دادنش به سیستم سخته چرا که یک مفهوم کلی را بیان میکند که باید BP شناخته بشه مدل‌هاش یادگرفته بشه. چون د رمورد domain اصلاً صحبت نمیکنه.  
اینکه خاص دامنه باشه جزئیات بیشتری خواهد داشت.

—

دامنه را باید با مثال بیان بشه:

سیستم هدف چیست از چه دامنه‌ای است. دامنه مدیریت منبع مدیریت کتاب خانه  
المان‌های دانشی الگوی انتخاب شده بالا:

کانسپت: کتاب میشه منبع، اسلاید مربوط به آن خوانده شود.

user resource <- borrow

دامنه اگه تغییر بکنه دیگه شرایط و خاص بودن نمیتونه همان نمونه قبلی باشد (الگو تغییر میکند).  
روش مستقل از دامنه:

در نظر گرفته سلسله موارد نیازمندی‌ها بازیابی متا مدل‌ها

نان فانکشنال‌ها مستقل از دامنه هستند، مانند امنیت و الگوریتم جست و جو

متامدل‌ها در مورد فانکشن‌ها هستند که در سطح سازمان عمل میکنه.

تاکسونومی یا نان فانکشنال: درخت بررسی شود.

نیازمندی‌های فانکشنالی که در سطح سازمان انجام میشه متامدل است.

چون جزئیات رو نمیکه خیلی کمتر استفاده می‌شود.

### ۷.۲.۳ Interviews

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

پرسشنامه هیچ وقت کامل نمیشه. برای کامل شدنش نیاز به مصاحبه داره. بدهیات در مصاحبه پرسیده نمی‌شه.

حالات ساخت یافته و غیر ساخت یافته دارد:

ساخت یافته براساس سوال است که پرسیده میشود.

بدون ساختار گفت و گو آزاد و بدون پیش در آمدی است.

### ۸.۲.۳ Observation and ethnographic studies

تنها system-as-is مشاهده میشه. هم فانکشن‌ها دیده میشه هم نان فانکشن‌ها استخراج میشه. حالا انجام کار افراد را بصری در صورت آنها می‌توان دید. چهره خسته کارمندان.

می‌تواند به صورت غیر فعال باشه: ناظر باشد می‌تواند به صورت اکتیو به صورت درگیر شدن با فرآیندها که همراه با یادداشت برداری است.

### ۹.۲.۳ Group sessions

ساخت یافته: هر کسی زمان دارد، همه چیز به صورت مشخص می‌باشد. کاملاً نقش‌ها مشخص است.

ساختار نیافته برای brainstorm: خودش انگار داره صورت جلسه می‌نویسه.

سازنور و مسخره کردن وجود ندارد.