فصل چهار

ترم که شروع شد یه سری مفاهیم و سازه ها رو توضیح دادیم.مهارتهای مورد نیاز برای تحلیلگر رو هم گفتیم. مدیریت پروژه که مجموعه مدیریت زمان و هزینه و ... را هم گفتیم.

انجام تحلیل: میشه در سه گام خلاصه اش کنیم.
۱- جمع‌آوری نیازها
۲- سازماندهی یا مدلسازی و نیازها(مدلسازی تحلیل)
۳- ارائه راه حل و پیشنهاد

مدل سازی تحلیل: یعنی اطلاعاتی که ما درباره سیستم جمع کردیم رو در قالب مدل نمایش بدهیم. اما برای چی این کار رو می‌کنیم؟ برای اینکه فهم بهترب از حل مسئله داشته باشیم.

مدل طراحی: راه حلی که ما برای سیستمی که شناختیم پیشنهاد می‌کنیم.

به راه حل جمع آوری نیاز ها و سازماندهی یا مدلسازی نیازمندی‌ها (مدلسازی تحلیل) Requirement Engineering (مهندسی نیازها) می‌گوییم.

اصطلاح مهندسی که قبلا اشاره شد:یعنی فرایندی که شامل چند گام هست و اونارو طی می‌کنیم تا قابلیت اطمینان محصول رو افزایش و هزینه محصول رو کاهش بدیم. پس شامل یک فرایند هست که نیاز به محاسبات داره، نیاز به ابزار و روش تحلیل داده داره که بهش مهندسی نیاز ها می‌گوییم.

System Requirement

مثلا بهمون گفتن سیستم بیمارستان رو بخش های مختلفش مثل رادیولوژی، داروخانه، پذیرش، بستری و... رو با هم متصل کنیم. و یک دیتا بیس یکپارچه داشته باشند و کاراشو بصورت paperless و بصورت اتوماسیونی انجام بدهند. این نیاز نیاز کلی هست و ما بهش stack holder request می‌گوییم.(این دو خط بالا)

اما لازمه ما بریم ببینیم داخل رادیولوژی چه کارهایی انجام می‌شود، چه نیازهایی دارند، در داروخانه چه نیازهایی دارن، چه کارهایی توی پذیرش انجام میشه. این شناختی که قراره بدست بیاریم میشه نیازمندی اونا : یعنی اون سازمان نیاز داره این قابلیت ها و این کار ها تبدیل بشه به نرم افزار. ما به اونا می‌گیم system requirement یا نیازمندیهای سیستم

تحلیل و طراحی سیستم ها – جلسه نهم

این نیازمندیهایی که داریم درموردش صحبت می‌کنیم یک Bench marks برای پذیرش سیستم یعنی ما قراره یک نرم افزار تحویل بدیم. باید یسری ویژگی ها این سیستم داشته باشه تا اونا بگن این همون چیزی هست که ما می‌خواهیم. این ویژگی ها همون System Requirement هست که در مرحله اونارو شناسایی می‌کنیم.

البته یک تصویر کلی از اونا در مرحله ی پروپوزال دادیم.

System Requirement : اون ویژگی هایی که ما باید در سیستم نهایی که قراره تحویل بدیم وجود داشته باشد.

- فعالیت مهندسی نیازها شامل
- جمع آوری نیازمندیها (داده ها)
- نمایش نیازمندیها (مدلسازی)
- ارائه پیشنهاد در انتهای کار (یعنی بگیم این سیستم چه مشکلاتی داره و احتمالا چه راه حلهایی داره، ما باید راه حل هارو در مرحله طراحی بررسی کنیم و راه حل نهایی خودمون رو طراحی کنیم) به زبان دیگه (۳) وارسی و تایید نیازمندی ها . یعنی معمولاً این مورد داخل دو گام اول انجام میشه

فرض کنید که رفتیم با یک کارمند بخش رادیولوژی یک بیمارستان صحبت کردیم و اون یه حرفایی زده و گفته سیستم ما باید فلان قابلیت رو داشته باشد. اگر قبول کنیم و همه اینارو ملاک قرار میدم و پیش میرم احتمالاً وقتی پروژه رو تحویل بگیره بگه این چیه! ما که اصلا این رو نمی‌خواستیم.

برای اینکه مطمئن بشیم اون ویژگی رو درست استخراج کردیم لازمه اونا رو وارسی و تایید کنیم (Validate & Verify)

روشهایی که معمولاً نیازمندیها رو بهش میپردازیم اینه که انواع اون نیازمندیهایی که سراغشون میریم:

دسته های نیازمندی های وظیفه مندی و غیر وظیفه مندی

نیاز وظیفه مندی: اون دسته از نیازهایی که ما انتظار داریم مستقیماً داخل سیستم وجود داشته باشند، عملکرد سیستم عملاً براساس این نیازها مشخص میشه مثلاً در سیستم HIS فرض کنید یک بیمار میخواد پذیرش بشه، اینکه اطلاعاتش ثبت بشه و بعد دکمه پذیرش زده بشه میشه نیاز وظیفه‌مندی. اینکه در یک سیستم بانکی افتتاح حساب بصورت الکترونیکی اتفاق بیوفته. اینکه دانشجو باید لیست دروس رو ببینه و بین اونها تعدادی رو انتخاب و ثبت کنه. اینکه سیستم ما چک کند که این دانشجو مجاز به اخذ کدام واحد ها هست میشه نیاز وظیفه‌مندی.

نیاز غیر وظیفه مندی: نیازهایی که عمدتاً به کیفیت نرم افزار برمیگرده مشخصاً ممکنه در قرارداد یا System Requirement ای که میگیم ذکر نشده، مثل امنیت، اینکه امنیت سیستم باید بالا باشه. اینکه سیستم واسط کاربری یا UI باید User friendly باشه، اینکه performance سیستم خیلی خوب باشه یعنی اگه قراره یک بیماری رو ثبت کنیم و اطلاعاتش رو که وارد کردیم و دکمه ثبت رو که زدیم، نره دو دقیقه بعد ثبت کنه، باید کمتر از دوثانیه برای ما این ثبت اطلاعات اتفاق بیوفته.

یک دسته دیگه نیازمندی ها که داریم براساس ویژگی هایی که نرم افزار داره دسته بندی می‌کنیم.

چالش‌هایی که در نیازمندی باهاشون سروکار داریم:

- Imprecision : غیر دقیق بودن نیازمندیهایی که مشخص میشه .
- Agreement : اینکه آیا اصلا توافقی روی آنها وجود داره یا نه.
- Creep : مشاجره و عدم توافق (البته به معنی خزیدن و آدم مرموزه!) منظور اینه که ما با آدم های متفاوتی سروکار داریم، خود این یک چالش هست، یک کارمند ممکنه یکسری اطلاعات رو به ما بده، یک کارمند ممکنه نده، یک ترسی رو اون کارمند داره که اگه اطلاعات رو به ما بده ممکنه ما سیستمی رو که توسعه میدیم جای اون رو بگیره، یا اطلاعاتی که می‌خواد درباره نحوه عملکردش بده منجر بشه دیگران تسلط بیشتری روی کار اون پیده کنن و احتمالاً یک سری دور زدن قانون و تخلفاتی که انجام داده لو بره، این موضوعاتی هست که در تعیین نیازمندیها باهاش مواجهیم.
- Cognitive bias : (سو گیری های شناختی) همیشه فکر می‌کنیم چیزی که داریم می‌بینیم احتمالاً درسته، چیزی می‌شنویم درسته ذهن ما ذهنیه که عملکردش درستو میزانه ولی واقعیتش اینه که اینطوری نیست! ما آدم ها پر از سوگیری های شناختی و متفاوتی هستیم. اینا باث میشن شناخت ما نسبت به پدیده ها همواره نادقیق باشه و یه موردش که در حوزه کسبو کار مورد توجه قرار می‌گیره اثر هاله ای هست.

اثر هاله ای: ما آدم ها وقتی یک ویژگی مثبتی رو در یکی می‌بینیم اون رو توسعه‌اش میدیم به موارد دیگه. مثلاً اون افرادی که میان برای مصاحبه اگه توی همون جلسه اول کارفرما رو جذب کنن، کارفرما میگه این آدم حتماً کارش هم درسته، حتماً آدم مرتب و منظمی هم هست، حتماً کاربلد هم هست. یعنی ویژگی رفتاریش رو وصل کردیم به ویژگی های مهارتیش که ممکنه لزوماً درست نباشه ولی واقعیت اینه که ماها اسیر این اثر هاله ای هستیم.

Scability : یعنی ما بتونیم کسب و کارمون اگر دز این مقیاس داره به خوبی کار می‌کنه، اگر مقیاس کسب و کار بزرگتر شود، با افزودن یکسری زیر ساخت ها به درستی کار کنه.

مثلاً اسنپ توی یک شهری خیلی خوب کار می‌کنه، حالا قراره بیایم و تعمیم بدیم به شهر های دیگه. قرار نیست بریم خودرو بخریم، محصولمون و یا رن افزارمون رو عوض کنیم. کافیه زیر ساخت سیستم یا سیستم رو طوری تغییر بدیم که اپلیکیشن ما با همون ویژگی ها روی این سیستم طوری کار کنه که وقتی مثلا داشت به هزار تا کاربر پاسخ میداد حالا به ۲۰ هزار کاربر پاسخ دهد. در استارتاپ ها خیلی با این موضوع سروکار داریم.

Security : اینکه سیستم خودش رو محافظت کنه از نفوذهایی که بصورت غیر مجاز قراره وارد سیستم بشن و هزینه های مستقیم و غیر مستقیم ای که در سیستم هست باید در System Requirement دیده بشه.

تکنیک های جمع‌آوری نیازها

- Team-based
- Individual

Team-based : به صورت تیمی باید کار رو پیش ببریم. یه تعداد کاربر رو جکع می‌کنیم. مثلا در بیمارستان یکی کارمند بخش داروخانه هست. یکی از رادیولوژی میاد . یکی از پرستاری میاد و ... این افراد داخل یک تیم جمع می‌کنیم و یکسری سوال ازشون می‌پرسیم و مصاحبه های گروهی انجام می‌دیم و برای هر کدوم ازاینا یک نقش تعریف می‌کنیم. جلسه هم می‌تونه بصورت formal و informal اداره بشه و با استفاده از ابزار ارائه بشه. IBM یک ابزاری رو برای این کار توسعه داده که یک تعداد کامپیوتر توی شبکه قرار دارن و نرم افزاری روی این کامپیوتر ها نصب هست و هر فردی پشت یکی از این کاکمپیوتر ها میشینه و تحلیلگر ارشد یک سوال رو مطرح می‌کنهن و همه توی کامپیوترشون می‌بینن و می‌توانند بصورت شفاهی و تایپی جواب دهند.

مثلا اگه مدیر کنار ما نشسته باشه و من کارمند بخوام حرفی بزنم که به مذاق مدیر خوش نیاد می‌تونیم بنویسیم و هدف اینه که سیستم رو تا اونجایی که ممکن هست بهتر و بهتر بشناسیم و مشکلات اون رو استخراج کنیم.

این روشی که IBM توسع داده و ما افراد رو کنار هم بچینیم رو می‌گیم JAD (Joint Application Development)

مزایا و معایب JAD :

نسبت به روش های مرسوم و سنتی که مصاحبه ی فردی انجام میشه و ما می‌رویم با تک تک افراد حرف می زنیم، هزینه بیشتری میبیره

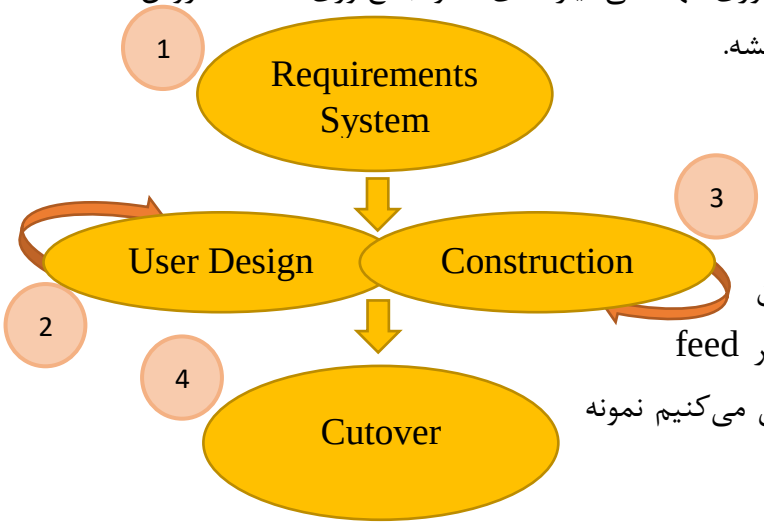
مدیریت کردن گروه های بزرگ کار سختیه

اجازه میده که کاربران بخش های مختلف که اتفاقاً توی پذیرش سیستم، توی ارائه نیازمندیهای سیستم مهم. موثر هستند، حضور داشته باشند و مشارکت کنند.

کاربران سیستم احساس می‌کنن خودشون مالک این سیستمی که قراره نوشته بشه هستند، بنابراین اهمیت پیدا می‌کنه که مشکلاتشون رو مطرح کنند.

معمولا نیازمندیها در این روش به صورت دقیق تری بیان میشه، چون اگر تضادی وجود داره همینجا مدیریت شود. مثلا کارمند۱ میگه این نیاز اینه و کارمند ۲ ممکنه بگه نه و این دو ممکنه با هم بحث کنند و ما نظاره گریم تا به یک اجماعی می‌رسن. بنابراین اون چالش Imprecision ایجا بهتر مدیریت میشه.

Rapid application Development : مثلاً تحت عنوان مدل Prototyping توضیح داده شده که یکی از روشهای تعیین کردن نیازمندی ها و جمع آوری مهندسی نیازمندی ها و جمع‌آوری اطلاعات روش Prototyping هست که حالا روش RAD شناخته میشه.



یکی از نمونه ها مشابه شکل رو به رو می‌باشد

برنامه ریزی می‌کنیم نیازمندیها رو جمع آوری کنیم.

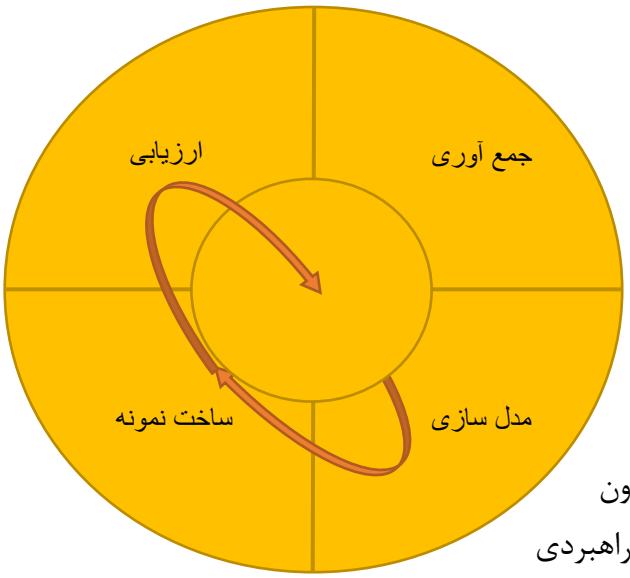
۱- یک طراحی اولیه از اون چیزی که ما تا الآن برداشت مردیم به ماربران ارائه می‌دهیم. و از کاربر feed back می‌گیریم . باهاش تعامل داریم و سعی می‌کنیم نمونه اولیه رو بسازیم.

۲- اون نمونه اولیه رو به کاربر نشون بدیم.

این چرخه بین مرحله ۲و۳ آنقدر ادامه پیدامی‌کنه تا روی اون بخشی که داشتیم کار توسعه رو انجام می‌دادیم، کاربر ها بگن آره این همون چیزی هست که ما می‌خواهیم.

یعنی روی نیازمندیها توافق حاصل شده باشه، حالا میریم به سمت cutover . یعنی ای قسمت محصول رو پیاده سازی می‌کنیم، ساخته شد، به مرحله بعدی پروژه می‌رویم. اینارو توی cutهای مختلف می‌بینیم.

مدل دیگه ای برای RAD :



قبلاً توضیح داده شده

مزایا و معایب Tram-based (RAD)

- توسعه سیستم رو با سرعت قابل قبولی جلو می‌بره
- خیلی تاکید زیادی رو نیازهای استراتژیک نداره و خیلی متمرکز بر نظارت کاربران هست، خیلی مهمه که کاربرا بگن این همون چیزیه که می‌خوان و آنقدر اصلاح کنند و اصلاح کنند تا مورد توافق طرفین شود. اون کاربرانی که دارن نظر میدن ممکنه خیلی به موضوعات راهبردی و استراتژیک سازمان توجهی نداشته باشن.
- به بحث کیفیت و سازگاری بیشتر توجه میشه و بیشتر مهمه که سیستم کار کنه.

روش های دیگر Team-based Agile :

ما در این روشها سعی می‌کنیم توسعه محصول رو به صورت گام به گام داشته باشیم. به صورت مجموعه ای از Prototype ها ایجاد کنیم . اینا رو دائماً به کاربران سیستم نشون بدیم و از آنها feed back بگیریم و اصلاح انجام بدیم. روش های Agile ایده‌اشون از همون روش های RAD گرفته میشه.

Scrum : این روش یکی از معروفترین ها می‌باشد. در این روش معمولاً فرایندی تعریف می‌کنند. هر فرابند در ابتدای جلسه، افرادی که توی تیم قراره کار کنند، یک warm up ای دارن. یک جلسه ی کوتاه ربع ساعتی درباره ی اینکه امروز قراره چیکار کنیم و task ها رو مدیر از قبلا تقسیم بندی کرده و اونا رو به افراد میده و در قالب یکسری بلاک ها بصورت task هایی که در دوره های زمانی مشخصی تعریف شده، اونارو به برنامه نویس ها،طراح ها ، تحلیلگران میدن و این taskها رو انجام میدن و اونارو ما به کاربرها نشون میدیم.. یعنی کسانی که ناظر نهایی ما هستند و feed back می‌گیریم. و اینا رو میاریم توی تیم و اصلاحاتی رو انجام می‌دیم و آنقدر این چرخه ادامه پیدا می‌کند تا ورژن نهایی نرم افزار تکمیل بشود.

مزیتش اینه که خیلی انعطاف پذیر هست و برای تغییرات خیلی خوب کار می‌کنه . یعنی مدیریت تغییرات رو خیلی خوب می‌تونیم انجام بدیم چون عملاً داریم نمونه هایی از محصول رو در جریان تحلیل سیستم، طراحی سیستم، تولید می‌کنیم. این درست هست که ممکنه همه قسمت ها لزوماً کار نکند، اما نمونه ای هست که توسط ماربرهای نهایی قابل دیدن می‌باشد.

اشکالش هام اینه که نیاز هست که افراد تیم مهارتهای کافی داشته باشن، برای اینکه بتونن هم بصورت تیمی کار کنند. و هم این ابزارهای مختلفی که توی Scrum مورد استفاده قرار می‌گیره رو به کار بگیرن.