مبانی مهندسی نرم افزار

موسسه آموزش عالی اقبال لاهوری

مبانی مهندسی نرم‌افزار

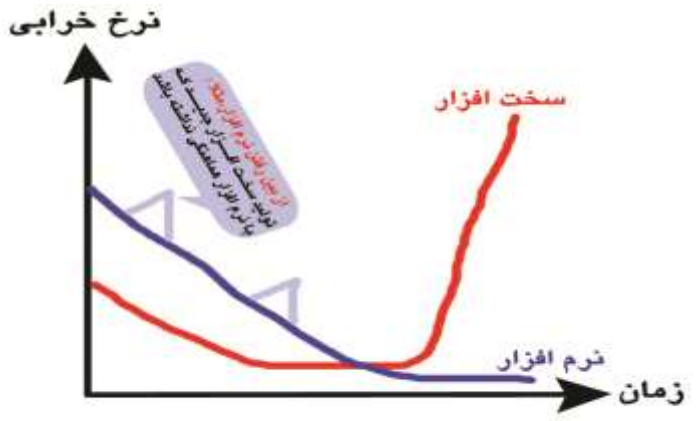
تعریف مهندسی نرم‌افزار: منظور از مهندسی نرم‌افزار فرآیندی است که در طی آن نرم‌افزاری با حداقل هزینه و حداکثر قابلیت اطمینان تولید گردد. بیان بهتر مهندسی نرم‌افزار مجموعه روش‌ها و ابزارهایی می‌باشد که باعث تولید نرم‌افزاری باکیفیت، با حداکثر قابلیت اطمینان و حداقل هزینه می‌شود (از دیدگاه پرسمن)

منظور از مهندسی نرم‌افزار استفاده از یکی از اصول زیر در حین تولید نرم‌افزار است:

- برای تولید نرم‌افزار به بیش از یک نفر نیاز باشد
- نرم‌افزار طوری تولید شود که امکان ایجاد نسخه‌های بالاتر از آن نیز باشد (از دیدگاه پرسمن)

دلایل پیچیدگی تولید نرم‌افزار نسبت به سخت‌افزار:

- نرم‌افزار به‌صورت تدریجی تولید می‌شود درحالی‌که سخت‌افزار یک محصول کارخانه‌ای بوده و دارای خط تولید است.
- سخت‌افزار عموماً از اتصال قطعات از پیش ساخته‌شده فراهم می‌شود درحالی‌که تولید نرم‌افزار از یک فرآیند تولید مهندسی که برای هر کاربر جدید منحصربه‌فرد است استفاده می‌شود.
- مفهوم فرسودگی در سخت‌افزار و نرم‌افزار متفاوت است. سخت‌افزار پس از تولید در طول چرخه حیات خود کم‌کم فرسوده شده تا حدی که دیگر قابل‌استفاده نباشد؛ اما در چرخه حیات نرم‌افزار اتمام عمر آن معنایی نداشته و فقط ممکن است پس از اعمال تغییرات زیاد کارایی اولیه خود را از دست بدهد.



نمودار مقایسه فرسودگی نرم افزار و سخت افزار

شکل ۱

دلایل پیچیدگی نرم‌افزار از دیدگاه بوچ (baoch):

۱. پیچیدگی در حوزه مسئله: غالباً از یک نرم‌افزار نیازهای گوناگون و گاه متضادی وجود دارد، در برخی موارد خود کاربران و مشتریان نیز دقیقاً نمی‌دانند که چه می‌خواهند. ضمناً علاوه بر نیازهای وظیفه مندی که در سیستم وجود دارد و برآوردن آن‌ها به‌اندازه‌ی کافی مشکل است نیازهای غیر وظیفه مندی مانند کارایی، سرعت، دقت، امنیت و ... نیز وجود دارد که برآوردن آن‌ها در بسیاری از موارد تولید را به چالش می‌کشد.

۲. پیچیدگی فرآیند تولید و مدیریت آن: پی بردن به آنکه چه فرآیندی برای تولید مناسب است، با چه الگویی فازهای مختلف تولید انجام شود؟ و سؤالاتی ازاین‌دست همواره مشکلی اصلی برای تیم تولیدکننده‌ی نرم‌افزار می‌باشد.

۳. انعطاف‌پذیری نرم‌افزار و نبود استانداردهای کافی: علی‌رغم اینکه از یک نرم‌افزار انتظار انعطاف‌پذیری بالایی می‌رود اما هیچ‌گونه استاندارد و روش جامعی برای تولید نرم‌افزار وجود ندارد که این دو عامل توأماً باعث پیچیدگی نرم‌افزار شده‌اند.

۴. مشکل توصیف رفتار سیستم‌های گسسته: نرم‌افزارها در کامپیوترهای گسسته کار می‌کنند درصورتی‌که اکثر سیستم‌هایی که ما برای آن‌ها نرم‌افزار تولید می‌کنیم سیستم‌های پیوسته هستند. تغییرات کوچک در ورودی سیستم‌های پیوسته تأثیر کمی در خروجی آن‌ها دارد درحالی‌که تغییرات کوچک در ورودی سیستم‌های گسسته ممکن است تغییرات زیادی در خروجی ایجاد کند که باعث پایین آمدن قابلیت‌های اطمینان می‌شود.

مدل‌های تولید نرم‌افزار:

این مدل‌ها یک سلسله‌ی مداومی از فعالیت‌ها هستند که منجر به تولید محصول می‌شوند یا در ارائه سرویس نقش دارند و نظم کلی روند تولید را مشخص می‌کنند.

انواع مدل‌های تولید:

۱. مدل آبشاری (water fall)
۲. مدل نمونه‌سازی (prototype)
۳. مدل تولید سریع نرم‌افزار (RAD)
۴. مدل تدریجی (افزایشی) (incremental)
۵. مدل چرخشی (حلزونی) (spiral model)
۶. مدل برنده-برنده چرخشی (win-win spiral model)
۷. مدل مؤلفه محور (component based)
۸. مدل روش‌های رسمی (formal method)
۹. مدل تولید تکنیک‌های نسل چهارم (4Gt)
۱۰. مدل توسعه هم‌روند (concurrent development model)

مدل آبشاری: در ابتدای تولید نرم‌افزار، تولید این محصول را مانند سایر محصولات به‌صورت خطی می‌دانستند، اولین مدل مبتنی بر این شیوه مدل آبشاری بود که در این مدل ترتیب انجام عملیات چنین است که خروجی هر مرحله ورودی مرحله بعد محسوب می‌شود.

شکل ۲

مزایای مدل آبشاری:

- بسیار روش ساده و قابل‌فهمی است
- به دلیل آنکه مراحل تولید نرم‌افزار را به‌طور واضح بیان می‌کند مدل بسیار خوبی برای آموزش می‌باشد.

معایب مدل آبشاری

- تولید نرم‌افزار به‌ندرت چنین طبیعت دنباله‌ای و جدا از هم دارد.
- هزینه بازگشت به عقب در این مدل بسیار بالاست.
- حجم زیادی از کار باید انجام شود تا خروجی برای مشتری قابل‌فهم باشد.
- ریسک این مدل بسیار بالاست.
- معمولاً مشتری آن‌قدر صبور نیست که اجازه دهد تمامی مراحل انجام شود سپس خروجی را ببیند.
- مرحله پیاده‌سازی باید به تأخیر بیفتد تا مراحل اولیه‌ی کار که بسیار هم‌زمان بر هستند انجام شود.

مدل نمونه‌سازی: در مدل آبشاری از کارفرما یا کاربر بازخوردی نداشتیم، در این مدل به دنبال افزایش بازخورد از مشتری هستیم، ابتدا یک نمونه آزمایشگاهی ساخته‌شده و به کاربر نمایش داده می‌شود. نظرات کاربر در هر مرحله دریافت شده و نمونه اصلاح می‌شود. بدین ترتیب درک بهتری از نیازهای

کاربر خواهیم داشت، این مدل به دنبال کاهش خطا در ریسک و درک‌های نادرست از صورت‌مسئله، ساخت صحیح نمونه‌ها با هزینه کمتر می‌باشد.

شکل ۳

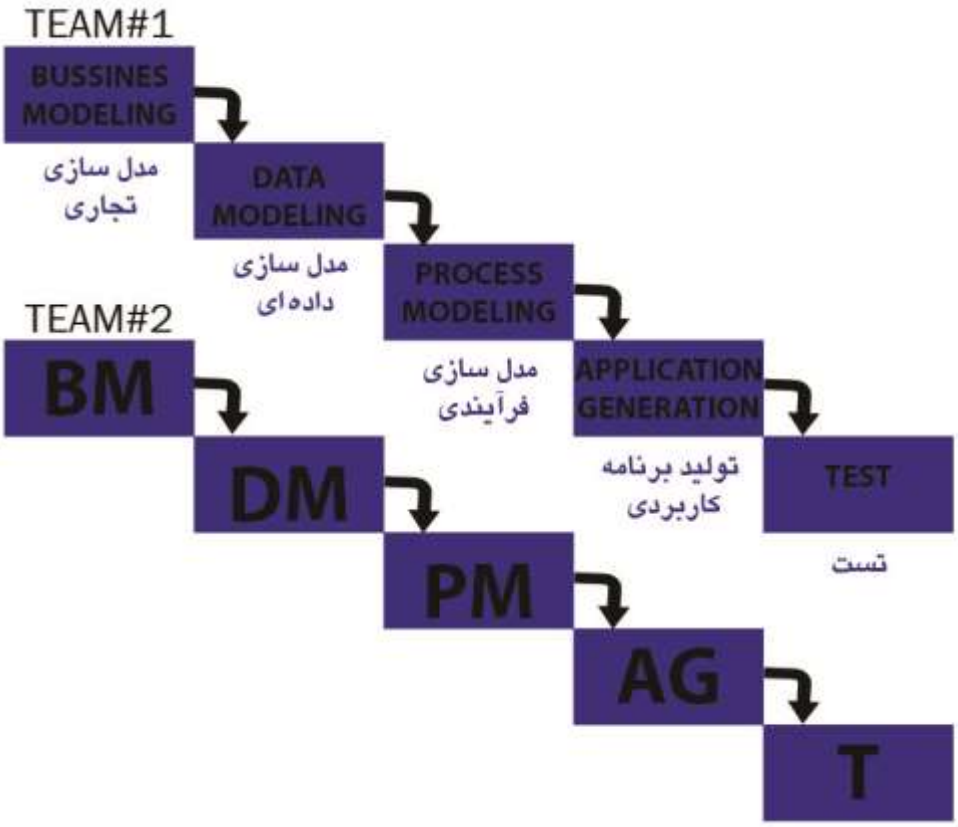
معایب مدل نمونه‌سازی:

- همه‌چیز به خواست مشتری پایه‌ریزی شده که گاه ممکن است متضاد و غیر مهندسی باشد
- صبر کم مشتری در دریافت محصول نهایی

مدل سریع (RAD): در این مدل برعکس مدل نمونه‌سازی ابتدا هسته‌ی اولیه ساخته‌شده و کم‌کم آن را گسترش می‌دهیم، در ابتدا نیازهای کلی و محدوده مسئله را شناسایی می‌کنیم، سپس هسته اولیه صورت‌مسئله تعیین می‌گردد. سپس سایر فرآیندها در قالب زیرتیمهایی به‌صورت موازی انجام می‌شود.

معایب مدل سریع (RAD):

- این مدل برای سازمان‌هایی مناسب است که هم تولیدکننده و هم مشتری به ادامه پروژه تعهد داشته باشند.
- این مدل تنها برای پروژه‌های بزرگ و مقیاس‌پذیر مناسب است
- نیروی انسانی زیادی می‌طلبد
- پروژه باید به‌گونه‌ای باشد که قابل‌تجزیه به مؤلفه‌های کوچک‌تر و قسمت‌هایی باقابلیت تولید موازی باشد.



شکل ۴

مدل تدریجی: در این مدل به‌مرور و تدریج و از طریق نوعی روش موازی و شبیه به روش خط لوله‌ای عمل می‌کند و به‌مرور نسخه‌ها و تحویل‌های تکمیلی ارائه می‌شود.

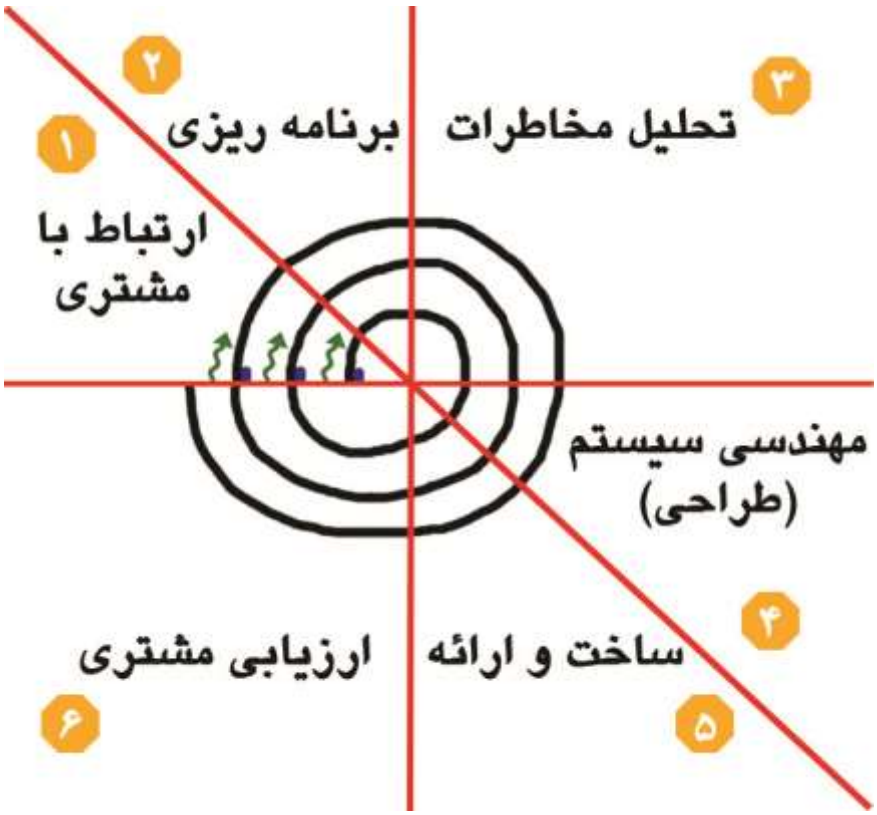
مزیت مدل تدریجی: حتماً نباید تیم بزرگی در اختیار باشد.

شکل ۵

مدل چرخشی (حلزونی): این مدل یکی از مدل‌های مطرح تولید نرم‌افزار است که روشی تکاملی بوده و از مدل‌های موفق و بهترین ایده می‌باشد.

معایب مدل چرخشی:

- متقاعد کردن مشتری در قابل‌کنترل بودن و کارایی روش تکاملی مشکل است
- مهارت ارزیابی خطر فراوانی می‌طلبد
- به دلیل آنکه زیاد مورداستفاده قرار نگرفته است بازدهی آن مشخص نیست



شکل ۶

مدل برنده برنده چرخشی: در این روش به دنبال متقاعد کردن مشتری برای دریافت مرحله‌به‌مرحله کار هستیم. برنده شدن از دیدگاه مشتری زمانی است که مشتری محصولی را دریافت کند که اکثر نیازهای او را برطرف کند؛ اما برنده شدن از دیدگاه تولیدکننده زمانی است که کاری را انجام دهد که درزمانی نزدیک به واقعیت، با هزینه مناسب، یعنی با هزینه و زمان تخمینی انجام شود. در این مدل تیم تولیدکننده باید سهام‌داران را بشناسد و عوامل برنده شدن آن‌ها را در هر مرحله در نظر بگیرد.

معایب مدل برنده برنده چرخشی:

- متقاعد کردن مشتری در قابل‌کنترل بودن و کارایی روش تکاملی مشکل است
- مهارت ارزیابی خطر فراوانی می‌طلبد
- همه‌چیز به خواست مشتری پایه‌ریزی شده که گاه ممکن است متضاد و غیر مهندسی باشد.

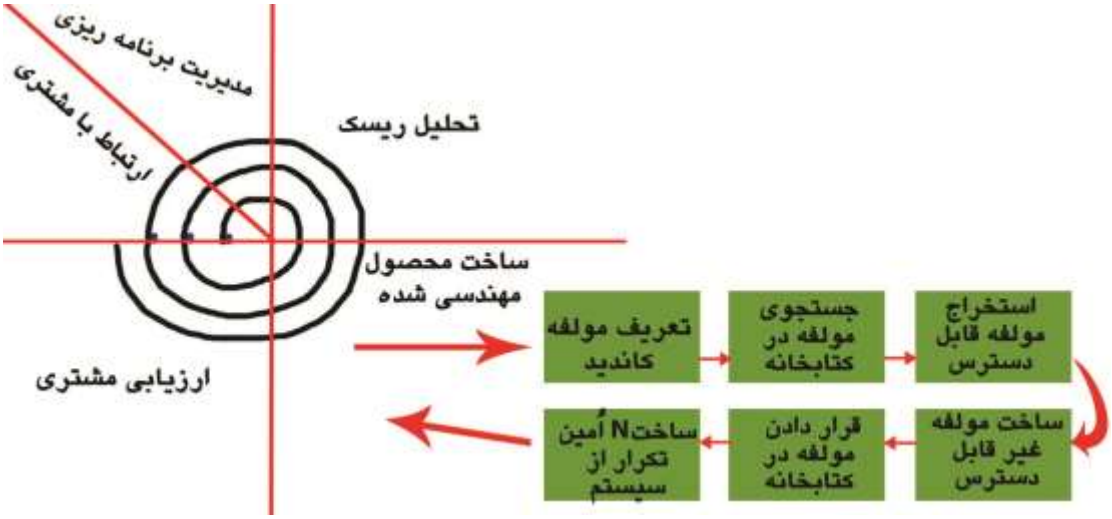
شکل ۷

مدل مؤلفه محور: component based: در این مدل به‌جای اینکه به دنبال تسریع در تولید باشیم مبنای کار را شناسایی مؤلفه‌هایی قرار می‌دهیم که این مؤلفه‌ها یک واحد ارائه‌کننده‌ی سرویس در یک واحد کاربردی هستند. این مدل تکمیل‌کننده‌ی مدل چرخشی می‌باشد.

نکته: این مدل بهترین مدل ارائه‌شده تاکنون است.

مزایای مدل مؤلفه محور:

- شیءگرا بودن
- سرعت ساخت بالا
- کلیه‌ی ویژگی‌های مدل چرخشی را نیز دارد



شکل ۸

مدل روش‌های رسمی (formal method): ایده‌ی این روش‌ها در توصیف و وارسی نرم‌افزار در قالب زبان‌های رسمی مانند beta و z است، درواقع این مدل‌ها مشکل اصلی تولید را در عدم توصیف دقیق نرم‌افزار در سایر فرآیندهای تولید می‌دانند و به دنبال توصیف صورت‌مسئله با زبان‌های توصیفی دقیق می‌باشند. با توجه به عدم ارائه یک فرآیند تولید، معمولاً به‌صورت ترکیبی، در کنار سایر فرآیندهای تولید نرم‌افزار استفاده می‌شوند.

مزیت مدل روش‌های رسمی:

- توصیف دقیق صورت‌مسئله

معایب مدل روش‌های رسمی:

- مدل کاملی نیست (ترکیبی استفاده می‌شود)
- گسترش مدل‌های رسمی در حال حاضر بسیار وقت‌گیر و پرهزینه است
- ازآنجایی‌که تعداد محدودی از نرم‌افزار سازان دارای زمینه‌ی لازم برای اجرای روش‌های رسمی هستند، آموزش گسترده‌ای موردنیاز است.
- استفاده از این مدل به‌عنوان راهکار ارتباطی با مشتریان که دید فنی ندارند دشوار است.

مدل تکنیک‌های تولید نسل چهارم (4GT): این روش‌ها مبتنی بر استفاده از امکانات فنی می‌باشد. ایده اصلی این روش‌ها اتوماتیک نمودن پیاده‌سازی است. برای این منظور در فاز پیاده‌سازی از ابزارهای تولید کد استفاده می‌شود.

معایب مدل تکنیک‌های تولید نسل چهارم:

چرخه کامل نیست، فقط سرعت تولید را بالا می‌برد.

ویژگی مدل تکنیک‌های تولید نسل چهارم:

- استفاده از 4GT روشی عملی برای اکثر زمینه‌های کاربردی به همراه ابزارهای مهندسی نرم‌افزار و مولدهای کد ارائه می‌دهد.
- زمان لازم برای تولید نرم‌افزار تا حد زیادی برای کاربردهای کوچک و متوسط کاهش می‌یابد و حتی میزان تحلیل و طراحی لازم برای کاربردهای کوچک نیز کاهش می‌یابد.
- استفاده از 4GT برای امور توسعه نرم‌افزارهای بزرگ به همان اندازه تحلیل و طراحی و آزمایش نیاز دارد و فقط از طریق حذف یا کاهش کد نویسی می‌توان در وقت صرفه‌جویی کرد.

مدل توسعه هم‌روند (concurrent development model): ((((HOME WORK))))

۱-مدل توسعه هم‌روند را به‌طور کامل توضیح دهید.

۲-بلوغ پروسه تولید را تعریف کرده، سطوح آن را نام‌برده و KPAهای مربوط به هر سطح را بر اساس نظر SEI بنویسید.

تولید نرم‌افزار:

خیلی از سیستم‌ها در فرآیند ساخت با شکست مواجه می‌شوند، زیرا تحلیلگر و سازنده به دنبال ساخت یک سیستم فوق‌العاده هستند بدون اینکه درک درستی از سیستم و سازمان داشته باشند.

چرخه حیات توسعه سیستم

چرخه حیات توسعه سیستم فرآیندی است که در طی آن مشخص می‌کنیم که:

۱. یک سیستم اطلاعاتی چگونه می‌تواند نیازهای تجاری را با ساخت سیستم اطلاعاتی برطرف کند؟
۲. این سیستم چگونه طراحی می‌شود؟
۳. چگونه ساخته می‌شود؟
۴. چگونه در اختیار کاربر قرار می‌گیرد؟

نقش کلیدی در SDLC را، تحلیلگر سیستم بازی می‌کند وظیفه او تحلیل تجارت، شناسایی نقاط بهبود فرآیند تجارت و طراحی سیستم اطلاعاتی می‌باشد.

SDLC از چهار فاز اصلی تشکیل‌شده است که هر فاز شامل یک سری مراحل است که هر مرحله توسط یک سری از تکنیک‌های خاص انجام می‌شود و ماحصل این مراحل Deliverable می‌باشد. این‌ها، مصنوعات ساخته‌شده توسط افرادی هستند که در SDLC مشغول به کارند و در حقیقت خروجی‌های هر فاز می‌باشند. هر فاز از deliverables فازهای قبلی استفاده می‌کند. باید توجه شود تکامل در SDLC به‌صورت تدریجی انجام می‌شود، یعنی در هر فاز نتایج به‌دست‌آمده در فازهای قبلی می‌توانند تغییر کرده و بهبود یابند.

برای ساخت یک سیستم اطلاعاتی توسط SDLC می‌توان به‌صورت زیر عمل کرد:

۱. به ترتیب فازهای SDLC را طی کنیم.
۲. در هر فاز deliverable ها را تولید کنیم.
۳. از deliverable های به‌دست‌آمده برای پیاده‌سازی استفاده کنیم.
۴. سیستم اطلاعاتی بسازیم.
۵. سیستم را پالایش کنیم.

چهار فاز اصلی در SDLC:

- طرح‌ریزی
- تحلیل
- طراحی
- مستندسازی

در مرحله طرح‌ریزی مشخص می‌کنیم که چرا یک سیستم اطلاعاتی باید ساخته شود و همچنین اعضای یک تیم پروژه را مشخص می‌کنیم. این فاز در دو مرحله تشکیل‌شده است:

الف: شروع پروژه

در این فاز ما مشخص می‌کنیم که چه منافع تجاری برای سازمان ارزشمند هستند؟ می‌توان با پاسخ به سؤال چگونه می‌توان هزینه‌های سازمان را کاهش داد و منافع بیشتری برای سازمان فراهم کرد، این منافع را شناسایی کرد. نتیجه این سؤال را در گزارشی به نام درخواست سیستم‌ها ارائه دهیم. در این گزارش به‌طور خلاصه بیان می‌کنیم که نیازمندی‌های تجاری سازمان کدام‌اند و ساخت سیستم اطلاعاتی چه منافعی را برای سازمان به ارمغان می‌آورد. سپس به تحلیل امکان‌سنجی می‌پردازیم که در آن مشخص می‌کنیم آیا ساخت سیستم ازلحاظ تکنیکی اقتصادی سازمانی انجام‌پذیر است یا خیر؟

سپس درخواست سیستم‌ها و نتیجه امکان‌سنجی را به یک انجمن بنام کمیته تائید برای گرفتن مجوز انجام کار ارسال می‌کنیم.

ب: مدیریت پروژه

در صورت تائید، پروژه وارد مرحله مدیریت پروژه می‌شود که در این مرحله طرح پروژه و کارمندان آن مشخص می‌شوند. نتایج این مرحله اغلب گزارشی به نام طرح پروژه جمع‌آوری می‌شوند. در این گزارش مشخص می‌کنیم که چگونه تیم پروژه عملیات توسعه سیستم را انجام می‌دهد.

تحلیل:

در فاز تحلیل مشخص می‌کنیم که:

الف: چه کسانی از سیستم استفاده می‌کنند؟

ب: سیستم چه‌کاری انجام می‌دهد؟

ج: سیستم چه زمانی و در کجا قابل‌استفاده خواهد بود؟

در طی این فاز، تیم پروژه به بررسی سیستم‌های موجود پرداخته، نقاط بهبود سیستم را پیدا و سیستم جدید را شناسایی می‌کند. این فاز از سه مرحله تشکیل شده است:

- تهیه یک استراتژی تحلیل برای مشخص کردن فعالیت‌های تیم پروژه
- جمع‌آوری نیازمندی‌ها و تهیه تحلیل که در آن مشخص می‌شود فعالیت‌های تجاری در سیستم آینده به چه صورت خواهد بود.
- تهیه گزارش پیشنهاد سیستم که در آن کلیه مفاهیم سیستم و مدل‌ها مشخص می‌شوند و این گزارش را در اختیار تصمیم‌گیرندگان سیستم قرار داده تا تصمیم نهایی را برای ساخت سیستم بگیرند.

طراحی:

در فاز طراحی چگونگی عملکرد سیستم را توسط تعیین کردن مشخصات سخت‌افزار، نرم‌افزار، زیرساخت‌های شبکه واسط کاربری فرم‌ها و گزارش‌ها مشخص می‌کنیم. این فاز از چهار مرحله تشکیل شده است:

الف: تهیه استراتژی طراحی که در آن مشخص می‌شود که سازمان توسط برنامه نویسان خود به ساخت سیستم بپردازد و یا از سازمان‌های خارجی برای ساخت سیستم استفاده کند.

ب: تهیه معماری طراحی که در آن زیرساخت‌های نرم‌افزاری و شبکه‌ای مشخص می‌شود.

ج: ساخت خصوصیت فایل‌ها و پایگاه داده که در طی آن مشخص می‌کنیم چه داده‌هایی و در چه زمانی ذخیره می‌شوند.

د: تهیه برنامه طراحی که در طی آن مشخص می‌کنیم چه برنامه‌هایی باید نوشته شوند و هرکدام در این برنامه‌ها چه‌کاری انجام می‌دهند.

درنهایت تمامی موارد گفته‌شده در غالب گزارشی به نام توصیف سیستم‌ها به تیم برنامه‌نویسان و پیاده‌سازان ارائه می‌شود.

پیاده‌سازی:

در این فاز، سیستم ساخته و نصب می‌شود. این فاز از سه مرحله تشکیل می‌شود:

- ساخت و تست سیستم
- نصب سیستم
- تهیه طرح پشتیبانی

متدولوژی توسعه سیستم‌ها:

متدولوژی چیست؟

متدولوژی (روش‌شناسی) مجموعه‌ای منظم از شیوه‌ها، ایده‌ها، استانداردها و راهکارهایی است که برای پیاده‌سازی SDLC بکار می‌روند. در آن‌ها ترکیب‌ها و سری‌های مختلفی از مراحل SDLC مشخص‌شده است.

به‌طورکلی متدولوژی‌ها به سه دسته تقسیم می‌شوند:

۱. متدولوژی‌های ساخت‌یافته
۲. متدولوژی‌های RAD
۳. متدولوژی AGILE

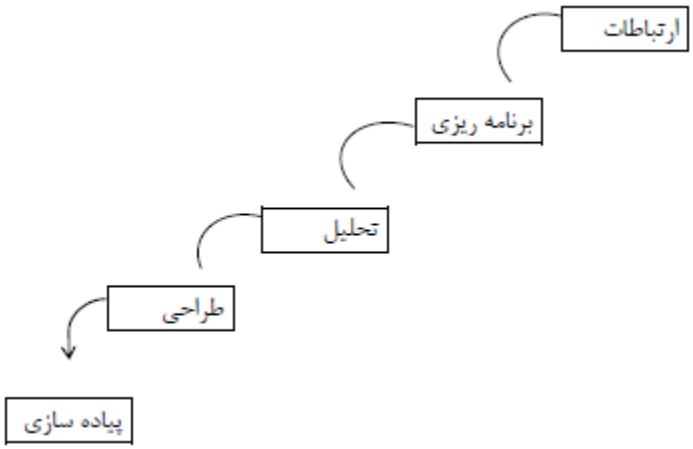
متدولوژی‌های ساخت‌یافته:

یکی از متدولوژی‌های قدیمی است که امروزه در سیستم‌های بزرگ عملاً غیرقابل‌استفاده است. در این متدولوژی مراحل مختلف SDLC به ترتیب اجرا می‌شوند. دو نمونه از این متدولوژی آبشاری و دیگری متدولوژی موازی است.

نکته: همواره نرم‌افزار، طراحی و توسعه می‌یابد درحالی‌که سخت‌افزار ساخته می‌شود و نرم‌افزار منطقی و انتزاعی است.

سخت‌افزار کهنه و فرسوده می‌شود درحالی‌که نرم‌افزار این‌گونه نیست و سخت‌افزار از دور خارج می‌گردد.

سخت‌افزار از طریق مونتاژ و قطعات ساخته می‌شود درحالی‌که نرم‌افزار باید به‌صورت سفارشی‌شده ساخته و اماکن مونتاژ در آن پایین است.



شکل ۹

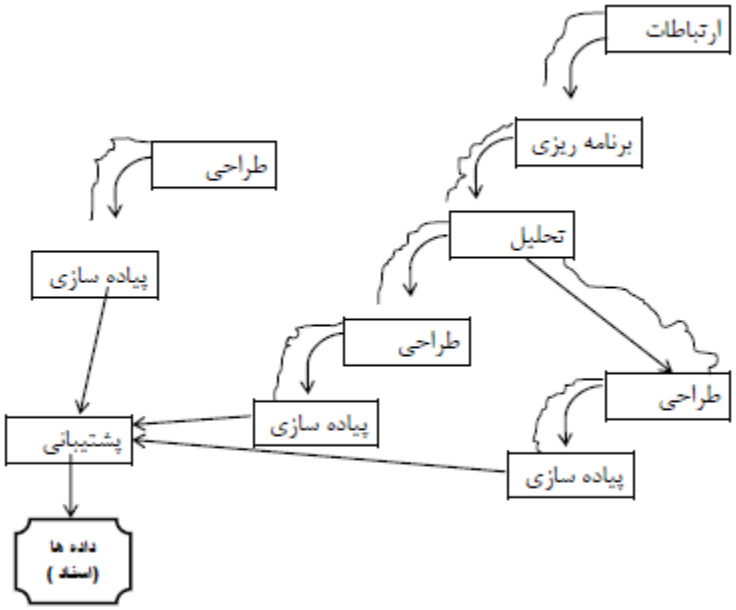
الف: متدولوژی آبشاری

مدل آبشاری و به‌طورکلی، مدل‌های خطی همگی دارای این ویژگی هستند:

- شروع یک مرحله ملزم به پایان مرحله‌ی قبلی است.
- امکان برگشت به مراحل قبلی وجود ندارد. به همین لحاظ امکان کار تیمی به حداقل می‌رسد و هزینه رفع مشکلات بسیار بالاست. ضمناً کاربران و یا مشتری باید تمامی خواسته‌ها یا نیازمندی‌های خود را در همان ابتدا معلوم نماید.

ب: متدولوژی موازی:

برای رفع فاصله زمانی زیاد مابین فاز تحلیل تا تحویل سیستم این متدولوژی پیشنهاد شد. در این متدولوژی بعد از پایان فاز تحلیل مرحله طراحی به چند قسمت تقسیم‌شده که در هر قسمت توسط یکسری افراد به‌طور موازی و همزمان انجام می‌شود. بعد از پایان پیاده‌سازی هر قسمت یک‌فاز یکپارچه‌سازی قسمت‌های ساخته‌شده‌ی سیستم در نظر گرفته‌شده است.



شکل ۱۰

متدولوژی‌های RAD (Rapid Application Development)

در متدولوژی RAD سعی بر این است که تا حد ممکن نسخه‌های اولیه سیستم سریع‌تر به دست کاربران برسد تا کاربر بتواند با استفاده از آن‌ها به فهم بهتری از سیستم دست یابد و خواسته‌های خود را سریع‌تر به توسعه‌دهندگان سیستم اعلام دارد.

سه نمونه مختلف از این متدولوژی به شرح زیرند:

الف: توسعه فازی (Phase Development)

ب: پروتوتایپ (Prototyping)

ج: پروتوتایپ دورانداختنی (Throw Away Prototyping)

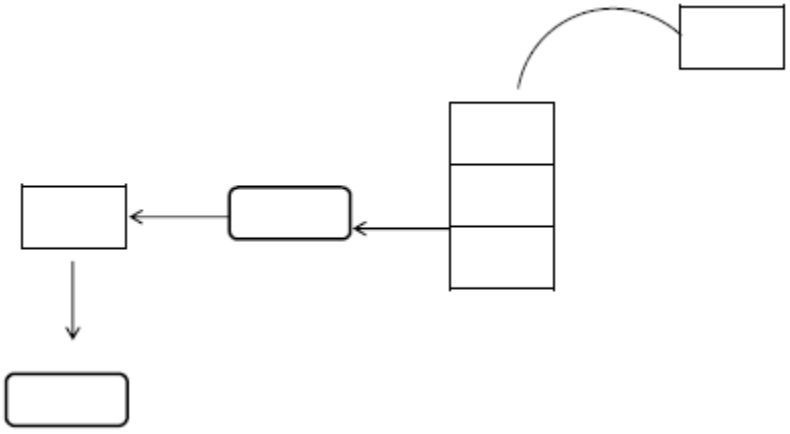
توسعه فازی:

در این متدولوژی کل سیستم را به نسخه‌های مختلفی تقسیم می‌کنیم. معمولاً نیازمندی‌های اصلی و مهم در نسخه اولیه گنجانده‌شده و در نسخه‌های بعدی ویژگی‌های دیگر و تغییرات موردنظر کاربر اعمال می‌شود. به‌محض پایان هر نسخه ساخت نسخه بعدی آغاز می‌شود.

پروتوتایپ:

پروتوتایپ به معنی ساخت یک نسخه آزمایشی است. در ابتدای مدل فازهای تحلیل، طراحی و ساخت باهم انجام می‌شوند و این فازها آن‌قدر تکرار می‌شوند تا سیستم کامل شود. در ابتدای کار یک تجزیه‌وتحلیل سریع و مقدماتی انجام‌شده و پروتوتایپ اولیه سیستم ساخته می‌شود.

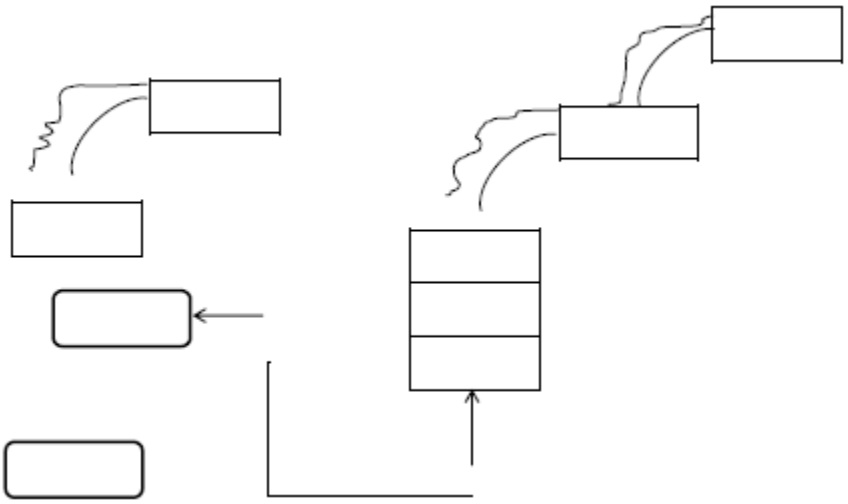
سپس این پروتوتایپ در اختیار کاربر قرارگرفته و مجدد فازهای گفته‌شده تکرار می‌شوند. پروتوتایپ در هر مرحله کامل‌تر می‌شود تا زمانی که کل سیستم بر اساس آن پروتوتایپ ساخته شود.



شکل ۱۱

پروتوتایپ دورانداختنی (Throw Away Prototyping)

این متدولوژی شبیه متدولوژی قبلی می‌باشد با این تفاوت که در این متدولوژی پروتوتایپ ساخته‌شده سیستم موردنظر نیست، بلکه فقط شبیه‌سازی‌شده قسمت‌های مختلفی از سیستم است که به این منظور ساخته‌شده‌اند که کاربر با آن‌ها کارکرده و بدین‌وسیله تحلیلگر بتواند نیازمندی‌های کاربران را بهتر درک کند. درنهایت این پروتوتایپ به دور انداخته‌شده و سیستم از ابتدا طراحی می‌شود.



شکل ۱۲

انتخاب یک متدولوژی مناسب

تمامی متدولوژی گفته‌شده مزایا و معایبی دارند و هیچ متدولوژی به‌عنوان بهترین، وجود ندارد؛ بنابراین بر اساس یک سری از پارامترها و نوع پروژه باید متدولوژی مناسبی انتخاب شوند که این پارامترها به شرح زیرند:

- میزان وضوح نیازمندی کاربران
- آشنایی با تکنولوژی
- پیچیدگی سیستم
- قابلیت اطمینان سیستم
- لزوم تحویل سیستم در یک بازه زمانی کوتاه‌مدت
- با برنامه پیش رفتن

مهارت‌های تحلیلگر:

مهارت‌هایی که یک تحلیلگر باید داشته باشد به شش دسته تقسیم می‌شوند:

مهارت‌های تکنیکی: تحلیلگر برای فهم تکنیک‌های استفاده‌شده در یک سازمان و تکنولوژی‌ای که برای ساخت سیستم استفاده خواهد شد و اینکه به چه صورت با استفاده از این تکنیک‌ها و تکنولوژی‌ها، راه‌حل مناسبی برای حل مشکلات سازمان بیابد احتیاج به مهارت‌های تکنیکی دارد.

مهارت‌های تجاری: تحلیلگر برای اینکه بداند چگونه یک سیستم اطلاعاتی می‌تواند نیازهای تجاری یک سازمان را برآورده کد به مهارت‌های تجاری نیازمند است.

مهارت‌های تحلیلی: تحلیلگر باید توسط مهارت‌های تحلیلی خود بتواند هم در سطح سازمانی و هم در سطح پروژه، مشکلات را حل کند.

مهارت‌های شخصی: تحلیلگر باید علاوه بر مهارت‌های علمی، مهارت‌های شخصی هم داشته باشد. مثلاً برای بهتر ارتباط برقرار کردن با مسئولین سازمان و برنامه نویسان

مهارت‌های مدیریتی: تحلیلگر باید از تکنیک‌های مختلف مدیریتی آگاه باشد تا بتواند تیم پروژه را در مسیر مناسب هدایت کند.

مهارت‌های اخلاقی: تحلیلگر باید با حسن اخلاق، شجاعت و راست‌گویی در کل پروژه باعث افزایش اطمینان مابین افراد درگیر در پروژه شود.

۱. USE Case Driven

بدین معنی است که Use Case ها به‌عنوان اصلی‌ترین ابزار مدل‌سازی برای تعریف رفتار سیستم استفاده شوند. یک Use Case نحوه تعامل کاربر با سیستم را برای انجام یک کار خاص توصیف می‌کند و در یک‌زمان فقط بر روی یک فعالیت متمرکز است.

۲. Architecture Centric

بدین معنی است که معماری نرم‌افزار و سیستم اطلاعاتی به‌صورت لایه به لایه باشد. یک معماری لایه به لایه حداقل از سه لایه تشکیل می‌شود:

الف: دید عملکردی (Functional View) که توصیف‌کننده رفتارهای خارجی سیستم از دید کاربر است.

ب: دید ایستا (Static View) که توصیف‌کننده ساختار سیستم و کلاس‌های آن و روابط مابین کلاس‌ها می‌باشد.

ج دید پویا (Dynamic View) که توصیف‌کننده رفتارهای داخلی سیستم است، مانند ارسال پیام‌ها مابین اشیاء

مرحله اول فرآیند SDLC، شناخت پروژه به نحوی است که ارزشمند است. پس‌ازآن باید گزارش درخواست سیستم‌ها که در آن اطلاعات اولیه در مورد سیستم اطلاعاتی ذکرشده است را آماده نمود. سپس تحلیلگر اقدام به تحلیل امکان‌سنجی می‌کند تا امکان ساخت را ازلحاظ اقتصادی، تکنیکی و سازمانی بررسی کند.

پروژه چیست؟

به مجموعه‌ای از فعالیت‌ها که یک نقطه شروع و یک نقطه پایانی دارند و منجر به ایجاد سیستمی می‌شوند که ارزش‌هایی را برای سازمان به ارمغان می‌آورد پروژه می‌گویند.

یک پروژه زمانی شروع می‌شود که شخص یا اشخاصی به‌عنوان اسپانسر می‌گویند متوجه ارزش‌های تجاری یک سیستم اطلاعاتی هستند و می‌توانند سودهایی را عاید سازمان نمایند.

اهداف پروژه توسط گزارشی به نام درخواست سیستم‌ها مشخص می‌شوند که این گزارش به گروهی به نام کمیته تائید ارسال می‌شود. این کمیته به بررسی گزارش پرداخته و در صورت تائید آنان وارد مرحله تحلیل امکان‌سنجی می‌شویم.

تحلیل امکان‌سنجی نقش مهمی را در اجرا و یا عدم اجرای یک پروژه و پیشرفت آن دارد. توسط تحلیل امکان‌سنجی مزایا و معایب پروژه را از جنبه‌های مختلف بررسی می‌کنیم و جزئیات کافی و دقیقی را در مورد منافع پروژه برای اسپانسر پروژه شرح می‌دهیم. در اکثر مواقع سفارش‌دهندگان سیستم همراه با تحلیلگر در تحلیل امکان‌سنجی همکاری می‌کنند. درنهایت نتیجه امکان‌سنجی به کمیته تائید ارسال می‌شود.

شناسایی ارزش‌ها و منافع تجاری (Ientifying Business Value)

ایده ارزش‌ها و منافع تجاری که توسط یک سیستم اطلاعاتی عاید سازمان می‌شود در ابتدا معمولاً توسط اسپانسر پروژه مطرح می‌شود. این ارزش‌ها و منافع می‌توانند منافع قابل‌لمس (Tangible) و یا غیرقابل‌لمس باشند.

ارزش‌های قابل‌لمس منافعی هستند که دارای مقدار و کمیت‌پذیری باشند. (Quantified) و معمولاً به‌سادگی قابل‌سنجش هستند. مثلاً کاهش سالانه ۲٪ هزینه‌های عملکردی سازمان منافعی که غیرقابل مقداردهی هستند و به‌سختی سنجیده می‌شوند ارزش‌های غیرقابل‌لمس نامیده می‌شوند: مثلاً سرویس‌دهی به مشتری.

درخواست سیستم‌ها (System Request):

درخواست سیستم‌ها گزارشی است که در آن دلایل و توجیه ساخت یک سیستم و منافعی که با ساخت آن سیستم، عاید سازمان می‌شوند را مشخص می‌کند. معمولاً اسپانسر پروژه خود این گزارش را تهیه می‌کند.

گزارش درخواست سیستم‌ها از پنج عنصر اصلی تشکیل شده است:

۱. Project Sponsor

تعیین اسپانسرهای پروژه

۲. Business Needs

دلایل تجاری که برای آغاز ساخت سیستم لازم‌اند؛ مانند افزایش فروش و یا بهبود دست‌یابی به اطلاعات

۳. BusinessRequirements

توانایی‌های تجاری که با ساخت سیستم به سازمان اضافه می‌شوند؛ مانند دسترسی به اطلاعات از طریق وب و یا امکان جستجو در بین محصولات توسط کاربران

۴. Business Value

منافعی که سیستم اطلاعاتی برای سازمان مهیا می‌کند مانند ۳٪ افزایش فروش و یا ۱۵۰۰ تومان کاهش هزینه در هرماه

۵. Restricts

نکات خاص و محدودیت‌هایی که برای ساخت سیستم با آن مواجه هستیم؛ مانند "سیستم باید برای تعطیلات نوروزی آماده باشد" و یا "نیاز به امنیت بالا"

خدمات نرم‌افزار:

جامعیت: یعنی نرم‌افزار در مقابل‌حملات خرابکاران مقاوم باشد و امکان نفوذپذیری یا آسیب‌پذیری آن در حداقل باشد. به عبارتی نرم‌افزار امن باشد. قابل‌ذکر است صفات اشاره‌شده با عنوان شروطی جهت استاندارد کردن نرم‌افزار از جانب ISO مطرح هستند.

صفات یا ویژگی‌های نرم‌افزار:

۱. قابلیت اطمینان: داده‌های کاربر را بیخود از بین نمی‌برد. قفل نمی‌کند و کمتر با شکست مواجه می‌شود. (Reliability)
۲. کارایی: حافظه را الکی پر نکند. سخت‌افزار را به‌طور بهینه مورداستفاده قرار دهد. (Efficiency)
۳. قابل‌حمل باشد: یعنی نرم‌افزار بر روی سکوهای مختلف سخت‌افزاری و نرم‌افزاری قابل‌اجرا باشند. هم بر روی سیستم‌عامل XP و هم بر روی Linux (Portability)

۴. قابلیت پشتیبانی: پرهزینه‌ترین فاز توسعه نرم‌افزاری از مستندات کافی و منابعی برخوردار باشد، راهنمای استفاده کاربران، راهنمای عیب‌یابی، رفع عیب (Maintainability)
۵. کاربرپسند: باید دارای ظاهر و محیطی باشد که کاربر به‌راحتی کار خود را انجام دهد. بر اساس مستندات نرم‌افزار، آن را مورد آزمایش قرار می‌دهیم؛ یعنی ورودی‌های لازم را به برنامه می‌دهیم و بررسی می‌کنیم آیا خروجی‌های لازم تولید می‌گردند یا خیر. (User Friendly)

روش‌های تست نرم‌افزار:

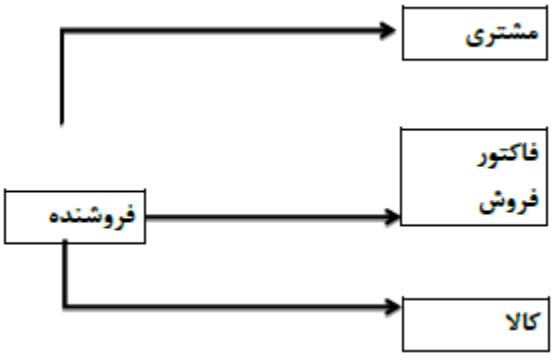
۱. تست جعبه سیاه: در این نوع تست بدون در نظر گرفتن جزئیات درونی نرم‌افزار، ورودی‌ها و خروجی‌ها
۲. تست جعبه سفید: باید جزئیات درونی برنامه را به‌صورت خط به خط، ردگیری کرد در قالب موارد زیر:
   a. جزئیات درونی حلقه‌ها را آزمایش می‌کنیم.
   b. تمامی مسیرهای درون برنامه را حداقل یک‌بار دنبال می‌کنیم.
   c. دستور IP را در حالات مختلف (درست بودن یا نادرست بودن) شرط می‌کنیم.
   d. داده‌های معتبر یا تألیف شده را به سیستم داده و کارکرد سیستم بر روی آن‌ها را آزمایش می‌کنیم.
۳. تست آلفا: در محل شرکت سازنده نرم‌افزار و توسط مهندسان توسعه‌دهنده در حضور مشتری انجام می‌گیرد.
۴. تست بتا: توسط کاربر در محل کار و یا استفاده او بدون نظارت سازندگان انجام می‌شود اما نتیجه درنهایت به سازندگان ارجاع خواهد شد.
۵. تست فشار: در این نوع تست، نرم‌افزار با داده‌های حجیم ازنظر تعداد و همچنین بزرگ بودن داده‌ها مورد آزمایش قرار می‌دهند.
۶. تست امنیت: در این نوع تست از طریق شیوه‌های مختلف نرم‌افزار را مورد هجوم قرار می‌دهند. (ویروس‌ها، کرم‌ها و نرم‌افزارهای جاسوسی) و نحوه دفاع نرم‌افزار در مقابل چنین تهدیداتی را بررسی می‌نماییم.
۷. تست خوب: یک تست خوب باید بتواند خطاهای بیشتری را در یک محدوده زمانی مشخص پیدا نماید یا به عبارتی حداکثر خطاهای ممکن را پیدا نماید. روش تست باید به‌گونه‌ای باشد که خطاها را پیش از تحویل نرم‌افزار به متقاضی تحویل دهد. روش تست باید توانایی پیش‌بینی خطاهای ممکن و متفاوت را داشته باشد.
۸. تست یکپارچه: چنانچه سیستم از چندین مؤلفه (Component) مختلف تشکیل شده باشد، تمامی این مؤلفه‌ها باهم سازگار بوده و به‌صورت یکپارچه کار می‌کنند. به عبارتی

مجموعه قوانین و مقررات در آن‌ها برقرار باشند و چنانچه مؤلفه جدیدی نیز به سیستم متصل شود، مابقی مؤلفه‌ها با معماری آن مؤلفه جدید هماهنگ است.

طراحی:

در این فاز بر مبنای خروجی‌های حاصل از مرحله تحلیل سیستم کار طراحی سیستم آغاز می‌شود. لذا در اینجا باید همان مدل‌ها و محصولات کاری در مرحله تحلیل را تکمیل نمود. به‌عنوان‌مثال در محیط Rational Rose در ابتدا سیستم در مرحله Use Case View از نگاه کاربران تحلیل و مدل‌سازی می‌شود. مثلاً مدل کلاس فقط دربرگیرنده کلاس‌های کسب‌وکار (Business) و یا موجودیتی باشد؛ اما در مراحل بعدی مانند مرحله Logical باید همان مدل کلاس طوری تکمیل گردد که آماده برنامه‌سازی گردد و کلاس‌های کنترلی مرزی و مربوط آن‌ها اضافه شود.

کلاس‌های کسب‌وکار:



شکل ۱۳

به‌طورکلی در مرحله طراحی اعمال زیر باید انجام گردد:

۱. طراحی و تمرین جزئیات واسط کاربری
۲. تعریف جزئیات بانک اطلاعات سیستم
۳. تعیین سخت‌افزارهای موردنیاز سیستم به همراه مشخصات کامل آن‌ها
۴. تعریف زیرساخت‌های موردنیاز سیستم
۵. تعریف شبه دستورالعمل‌های روال‌ها و توابع سیستم
۶. طراحی ماژولار سیستم در غالب مؤلفه‌هایی باقابلیت استفاده مجدد

مؤلفه: (Component)

یک بخش ماژولار از سیستم‌ها می‌باشد که در انتهای مرحله طراحی آماده می‌شود و عمل کد نویسی متنی برای آن‌ها نیز انجام می‌شود.

برای طراحی ماژولار مؤلفه‌ها، دو شرط زیر باید برقرار باشد:

الف- Copling: یعنی نباید هیچ‌یک از کلاس‌های یک مؤلفه به‌طور مستقیم به کلاس‌های مؤلفه دیگری وصل باشند و یا درخواست داده‌هایی از آن‌ها را بدهند. لذا برای رفع این مشکل از رابط Interface استفاده می‌شود.

رابط: کلاسی است که فقط حاوی عملیات می‌باشند و فاقد هرگونه صفت است. این کلاس‌ها رابط درخواست‌های کلاس‌های دیگر را برای دسترسی به اطلاعات دریافت کرده و اطلاعات درخواستی آن‌ها را فراهم می‌نماید تا از ارتباط مستقیم بین کلاس‌ها جلوگیری به عمل آورد.

ب: Coehesion: مجموعه‌های کامل درونی یک مؤلفه باید همگی باهم به‌صورت یک هدف واحد وجود داشته باشند.

اصول حاکم بر توسعه Agile:

۱- توسعه نرم‌افزار به مدل افزایشی: (Incremental) یعنی طی چندین بار تکرار نرم‌افزار تکمیل می‌شود. به این نسخه‌ها Sprint می‌گویند و مرتباً در تکرارهای بعدی نرم‌افزار کامل می‌گردد. درواقع بعد از هر بار تکمیل نرم‌افزار به متقاضی بازخورد کار را ارائه داده و در ادامه مورداستفاده قرار می‌دهند.

۲- تیم بندی تحلیل گران و طراحان: به‌گونه‌ای باشد که کارها در حد مقدور کوچک‌تر و تیم‌ها نیز با نفرات کمتری شکل گیرند تا درنهایت بالاترین حالت ممکن از کارکرد تیمی حاصل شود.

۳- کاستن از حجم عملیات مستندسازی و پرداختن به جنبه‌های عملیاتی سیستم: قابل‌ذکر است که مستندات سازی یکی از عملیات مهم و پرحجم در فرآیند توسعه سیستم‌ها است که طی آن باید کلیه اسناد سیستم از قبیل طرح‌ها، برنامه‌ها، مدل‌ها، فرم‌ها و گزارشات کد نویسی شده راهنمای استفاده کاربران، راهنمای عیب‌یابی و رفع عیب و ... تهیه و بایگانی شوند. لذا معمولاً جزو عملیات پرحجم و زمان‌گیر می‌باشند. لذا در اینجا توسعه بر این است که از حجم مستندات کاسته گردد و به‌جای آن به کارهای اجرایی سیستم پرداخته شود.

۴- انعطاف‌پذیری و افزایش کارایی در پاسخ به تغییرات: یکی از موارد چالش‌برانگیز در توسعه سیستم‌های نرم‌افزاری، تغییر در خواسته‌ها و نیازمندی‌های متقاضی است. در مدل Agile

به‌منظور مدیریت بهینه‌تر، این ساختار به‌گونه‌ای تنظیم می‌شود که در هنگام مواجهه با تغییرات احتمالی بتوان با کمترین هزینه و حجم کار، تغییرات را عملی کرد.

نمونه‌سازی اولیه:

در راستای انجام عمل امکان‌سنجی و همچنین تحلیل ریسک در توسعه نرم‌افزار عمل نمونه‌سازی انجام می‌شود. در طی ایجاد نمونه‌های اولیه فقط به موارد کلی سیستم می‌پردازند و مواردی همچون تکمیل رابط کاربر حذف اخطارها به حداقل رسانی حافظه معرفی به حداکثر رسانی سرعت اجرایی و ... در این مرحله مورد رسیدگی قرار نمی‌گیرند و در پی ساخت یک نمونه کلی از نرم‌افزار است تا هر چه زودتر آماده شود. امروزه برای ساخت چنین نمونه‌هایی از یک سری ابزارهای Case و مهندسین نرم‌افزار استفاده می‌گردد.

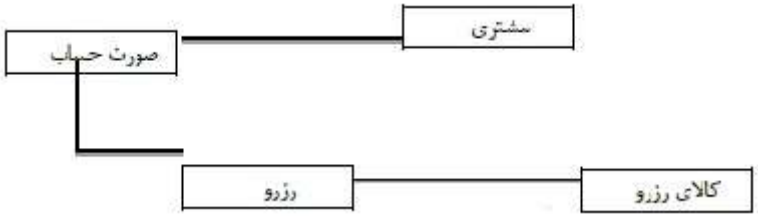
نمونه اولیه دورانداختنی:

این نوع نمونه درواقع خود اصل سیستم نیست و نمونه شبیه‌سازی آن می‌باشد و بعد از پایان ساخت آن و ارزیابی آن بطوریکه مشخص می‌شود که آیا نیازمندی‌های متقاضی تأمین می‌گردد یا خیر؟ و آیا دور انداخته می‌شود و سیستم دوباره از نقطه صفر شروع به ساخته‌شدن می‌نماید یا خیر؟

درحالی‌که در نمونه‌های معمولی در صورت تائید نمونه اولیه از جانب مدیران و تحلیلگران کاربر روی آن ادامه می‌یابد تا اینکه نرم‌افزار اصلی ساخته شود.

الگوها:

الگوها در راستای استفاده مجدد از کد استفاده می‌شود. درواقع هرگاه به مجموعه‌ای از کلاس مربوط، مواجه گشتیم می‌توان از همین مجموعه کلاس در سیستم دیگری نیز استفاده نمود. در اینجا مجموعه کلاس را می‌توان به‌عنوان یک الگو در نظر گرفت؛ مانند ایده‌ی چرخ که ابتدا برای ساخت کاری مورداستفاده قرار گرفت. مثلاً در سیستم رزرو غذا در یک رستوران کلاس‌هایی مانند مشتری، رزرو کالا یا خدمات رزرو شده و صورت‌حساب وجود دارد. اگر دقت نمایید همین کلاس‌ها را می‌توان در سیستم رزرو دیگری مانند رزرو بلیت هواپیما و سرویس تاکسی‌تلفنی استفاده کرد.



شکل ۱۴

انواع الگوها:

۱. الگوهای تحلیل
۲. الگوهای طراحی
۳. الگوهای معماری

جمع‌آوری نیازمندی‌ها (Requirements Gathering)

نیازمندی‌ها عبارت است از کارهایی که قرار است سیستم انجام دهد و یا ویژگی‌هایی که سیستم باید داشته باشد. در این مرحله مشخص می‌شود که سیستم چه‌کاری باید انجام دهد. نیازمندی‌ها را از دید کاربران سیستم (Business User) تعریف می‌کنیم که به آن Business Requirements گفته می‌شود. ولی در مرحله طراحی نیازمندی‌ها از دید توسعه‌دهندگان سیستم بیان می‌شود که به آن System Requirements گفته می‌شود. درنهایت تمامی نیازمندی‌ها جمع‌آوری‌شده و در غالب تعریف نیازمندی‌ها ارائه می‌شود.

انواع نیازمندی‌ها:

۱. نیازمندی‌های اساسی (Functional Requirements) شامل فرآیند یا اطلاعاتی است که سیستم باید داشته باشد و در حقیقت مشخص‌کننده عملکرد سیستم است.
۲. نیازمندی‌های غیراساسی (Non-Functional Requirements) شامل خصوصیات رفتاری است که سیستم باید داشته باشد؛ مانند کارایی بالا و امکان دسترسی از طریق اینترنت. در حقیقت ویژگی‌های مختلف سیستم مانند ویژگی‌های عملیاتی، کیفی، امنیتی، فرهنگی و سیاسی از طریق این نیازمندی‌ها مشخص می‌کنیم.

Operational Requirements: محیط‌های فیزیکی و تکنیکی که سیستم در آن عمل خواهد کرد.

Performance Requirement: سرعت، حجم و قابلیت اطمینان سیستم

Security Requirement: چه افرادی و تحت چه شرایطی توانایی دسترسی به سیستم و اطلاعات را دارا می‌باشند.

Cultural & Political Requirement: فاکتورهای فرهنگی و سیاسی و قوانینی که بر سیستم تأثیرگذارند. درنهایت کلیه نیازمندی‌های شناخته‌شده را در غالب گزارش تعریف نیازمندی‌ها ارائه می‌کنیم.

نیازمندی‌های غیراساسی شناخته‌شده در سیستم به شرح زیرند:

۱. نیازمندی‌های عملکردی
   a. سیستم باید تحت محیط Windows عمل کند.
   b. سیستم باید توانایی خواندن و نوشتن فایل‌های Word, Html را داشته باشد.
   c. سیستم باید توانایی مشاهده تصاویر با فرمت BMP, GIF و JPEG را داشته باشد.
۲. نیازمندی‌های کیفی
   a. زمان پاسخ حداقل باید ۷ ثانیه باشد.
۳. نیازمندی‌های امنیتی
   a. مورد امنیتی خاص در سیستم موجود نیست.
۴. نیازمندی‌های فرهنگی و سیاسی

مراحل فرآیند تحلیل

- فهم سیستم موجود (As Is System)
- شناسایی نقاط بهبود و پیشرفت سیستم (System Improvement)
- تشخیص و توسعه نیازمندی‌ها برای سیستم جدید (To-Be System)
- در فرآیند تحلیل، احتیاج به تکنیک‌هایی برای جمع‌آوری نیازمندی‌ها (Gathering Requirements)
- به‌منظور جمع‌آوری اطلاعات و تکنیک‌هایی برای تحلیل نیازمندی‌ها (Analyzing Requirement)

تکنیک‌های تحلیل نیازمندی‌ها:

- اتوماتیک کردن فرآیند چارت BPA
- بهبود فرآیند تجارت BPI
- مهندسی مجدد فرآیند تجارت BPR

BPA

در این تکنیک ما اقدام به بررسی سیستم نموده و مشخص می‌کنیم که در کدام قسمت‌های سازمان می‌توان کارها را کامپیوتری انجام داد. در این مرحله در عملیات و فرآیندهای سازمان تغییری به وجود نمی‌آوریم. تکنیک‌های BPA زمان زیادی را صرف فهم سیستم موجود می‌نمایند. تا مشخص کنند در سیستم آینده چه فرآیندهای تجاری در سیستم قابلیت اتوماتیک سازی را دارند و مشخص شود کدام قسمت‌های سازمان باید توسط سیستم‌های اطلاعاتی پیاده‌سازی شوند.

تکنیک‌های BPA

تحلیل مشکل (Problem Analysis): در این تکنیک با استفاده از کاربران و مدیران سیستم به تشخیص سیستم جاری (AS-Is System) می‌پردازیم و مشخص می‌کنیم که مشکل را به چه صورت در سیستم آتی (To-Be Sytem) حل کنیم. بدین معنی که از کاربران و مدیران سازمان می‌خواهیم که مشکلات سازمان را برای ما تشریح کرده و به ما بگویند به چه طریقی می‌توان این مشکلات را حل نمود. در اینجا تمرکز بر روی حل مشکلات است نه خود مشکل. در پایان این قسمت راه‌حل مشکلات پیدا می‌شود.

تحلیل علت ریشه: (Root Cause Analysis): در اینجا تمرکز بر روی خود مشکل است و تحلیلگر سعی در پیدا کردن ریشه مشکل دارد. به‌عنوان‌مثال فرض کنید یکی از لامپ‌ها مرتب می‌سوزد شما می‌توانید هر دفعه که این لامپ می‌سوزد آن را با یک نمونه سالم تعویض کنید و یا دنبال علل سوختن مکرر لامپ بروید. برای این کار تحلیلگر ابتدا لیستی از مشکلات سیستم جاری تهیه‌کرده و سپس آن‌ها را اولویت‌بندی می‌کند. سپس به بررسی مهم‌ترین آن‌ها پرداخته و سعی در پیدا کردن ریشه مشکل دارد. معمولاً با پیدا کردن ریشه یکی از مشکلات، تعدادی از مشکلات دیگر نیز حل می‌شود.

BPI

در این تکنیک ما اقدام به ایجاد تغییراتی در سازمان نموده تا بتوانیم از منافع تکنولوژی‌های جدی استفاده نموده و عملکرد سازمان را بهبود ببخشیم. توسط BPI عملکرد سازمان بهتر می‌شود و می‌توانیم بازدهی را توسط انجام دادن درست کارها (Doing Things Right) و اثربخشی را توسط انجام کارهای درست (Doing The Right Things) بهبود بخشیم.

تکنیک‌های BPI

تحلیل مدت: در این متد ما زمان انجام فرآیند را در سیستم جاری بررسی می‌کنیم. تحلیلگر کار خود را با مشخص کردن کل و میانگین مدت‌زمانی که برای انجام یک مجموعه از فرآیندهای تجاری که به یک ورودی خاص اجرا می‌شوند شروع می‌کند. سپس مدت اجرای هرکدام از این فرآیندها را

مشخص می‌کند. درصورتی‌که مدت اجرای هرکدام از این فرآیندها تفاوت فاحشی با کل زمان لازم برای اجرای مجموعه فرآیندها داشته باشد، آنگاه این فرآیند احتیاج به تغییر دارد. به‌عنوان‌مثال در فرآیند پرداخت وام ممکن است کل فرآیندها دو ماه زمان ببرد. پس از تحلیل متوجه می‌شویم که کارهایی که در این فرآیند انجام می‌شود مانند بررسی وام‌گیرنده، میزان اعتبار وام‌گیرنده و ... همگی طی ۲ هفته انجام پذیرد.

تکنیک‌های جمع‌آوری نیازمندی‌ها:

- مصاحبه (Interview)
- پرسشنامه (Questionaires)
- تحلیل اسناد (Document Analysis)
- مشاهده (Observation)

مصاحبه

مراحل مصاحبه به شرح زیر می‌باشد:

- انتخاب افراد برای مصاحبه
- طراحی سؤالات
- آماده شدن برای مصاحبه
- انجام مصاحبه و هدایت آن
- کارهای بعد از مصاحبه

انتخاب افراد برای مصاحبه:

در ابتدا باید لیستی از افراد که می‌خواهیم با آن‌ها مصاحبه کنیم تهیه‌کرده، هدف مصاحبه با هر شخص را مشخص می‌کنیم و زمان آن را تعیین کنیم که با چه کسی، در چه زمانی و با چه هدفی باید مصاحبه کنیم؟

افرادی که باید برای مصاحبه انتخاب شوند بر اساس اینکه تحلیلگر به چه اطلاعاتی نیاز دارد مشخص می‌شود. در ابتدا باید با مدیران ارشد و سپس با مدیران میانی و درنهایت با کارمندان مصاحبه شود که این ترتیب می‌تواند مرتب تکرار شود؛ زیرا در ابتدا باید یک درک کلی از فرآیندهای سازمان پیدا شود و بعد با مصاحبه با کارمندانی که این فرآیندها را انجام می‌دهند از چگونگی مراحل داخل فرآیندها مطلع شد.

طراحی سؤالات:

Closed-End: پرسش‌هایی هستند که در آن‌ها یک جواب خاص وجود دارد و اطلاعات دقیقی را برای ما مشخص می‌کند. مثلاً برای اینکه بپرسیم: آیا شما درخواست‌های زیادی را در روز پاسخ می‌دهید؟ باید بپرسیم: " چه تعداد درخواست‌ها را شما روزانه پاسخ می‌دهید؟"

Open-Ended: سؤالاتی هستند که در آن‌ها جواب خاص و دقیقی وجود ندارند. مثلاً اظهارنظر در مورد یک موضوع و یا اینکه، شما با چه مشکلاتی در سازمان روبرو هستید؟

Probing: سؤالاتی که در فهم مسئله کمک می‌کنند و جزئیات بیشتری را برای ما آشکار می‌کنند و معمولاً زمانی پرسیده می‌شود که قسمت‌هایی از صحبت‌های مصاحبه‌شونده برای مصاحبه‌کننده غیرشفاف است.

در ابتدای پروژه به علت اینکه مصاحبه‌کننده هیچ اطلاعی در مورد سیستم ندارد نمی‌تواند سؤالات دقیقی را طراحی کند، یعنی مصاحبه‌کننده از یک مدل ساختار استفاده می‌کند و سؤالات کلی در مورد سیستم می‌پرسد، پس از گذشت مدتی با آشناشدن با سیستم مصاحبه به شکل ساختار داده‌شده‌ای ادامه پیدا می‌کند و می‌توان بر روی موارد خاص تمرکز کرد.

آماده شدن برای مصاحبه:

خیلی از تحلیلگران تازه‌کار به این دلیل که تهیه کردن یک ساختار برای مصاحبه و استفاده از سؤالات Closed-End عملی دشوار و وقت‌گیر است از این کار اجتناب می‌کنند. غافل از اینکه درصورتی‌که اگر ساختار مصاحبه آماده نباشد کار بسیار دشوارتر شده و مجبور به مصاحبه دوباره می‌شویم و همچنین اکثر افراد از اینکه زیاد مورد مصاحبه قرار بگیرند ناراضی می‌باشند. برای آماده شدن جهت انجام مصاحبه باید اعمال زیر را انجام داد:

۱. آماده‌سازی طرح کلی مصاحبه
    a. تهیه لیست سؤالات
    b. پیش‌بینی پاسخ‌ها
۲. اولویت‌بندی سؤالات
۳. آماده شدن برای مصاحبه
    a. زمان‌بندی
    b. توضیح دلیل مصاحبه
    c. توضیح محدوده موردبحث

انجام و هدایت مصاحبه:

نکاتی که برای هدایت و انجام مصاحبه باید مدنظر داشت:

- در طول مصاحبه بی‌طرفی و بی‌غرضی خود را کاملاً نشان دهید.
- تمامی اطلاعات را ثبت کنید.
- از اینکه تمامی نکات و واژگان را درست متوجه شده‌اید اطمینان حاصل کنید.
- حقایق را ازنظرات جدا کنید.
- زمانی را برای مصاحبه‌شونده برای طرح سؤالات اختصاص دهید.
- ادب را در طول کل مصاحبه کاملاً رعایت کنید.
- شوخ‌طبع باشید.
- از گفتن اصطلاحات فنی خودداری کنید.
- در زمان معین‌شده به مصاحبه پایان دهید.
- نکات کلیدی را جمع‌بندی کنید.
- کوتاه و مختصر سخن بگویید
- راست‌گو باشید.
- به زبان اشاره و حالات مصاحبه‌شونده دقت کنید.

کارهای پس از مصاحبه:

پس از پایان مصاحبه اعمال زیر را انجام دهید:

- آماده‌سازی نکات مطرح‌شده در مصاحبه
- آماده‌سازی گزارش مصاحبه حداکثر تا ۴۸ ساعت پس از هر مصاحبه
- بررسی مصاحبه انجام‌شده برای مشخص کردن نکات نامعلوم و پرسش‌های جدید
- ارسال گزارش ارسال‌شده به مصاحبه‌شونده برای گرفتن تأییدیه
- تهیه گزارش مصاحبه
    - تائید مصاحبه
    - مصاحبه‌کننده
    - اهداف مصاحبه
    - جزئیات

معماری نرم‌افزار:

معماری به ساختار نرم‌افزار و راهکارهایی که آن ساختار جامعیت مفهومی را برای سیستم فراهم می‌آورد مربوط می‌شود. معماری درواقع ساختار سلسله مراتبی اجزای برنامه (ماژول‌ها- مؤلفه‌ها) همراه با روش برقراری ارتباط بین این مؤلفه‌ها را نشان می‌دهد. علاوه بر آن ساختمان داده‌ای که توسط این مؤلفه‌ها نیز استفاده می‌شوند نیز باید نمایش داده شود. به‌طورکلی مؤلفه‌ها می‌توانند آن‌قدر گسترده شوند تا عناصر عمده تیم و ارتباطات بین آن‌ها را نمایش دهد. مجموعه‌ای از الگوهای معماری چندلایه، Mediator موجب می‌شود تا مهندس نرم‌افزار بتواند مفاهیم سطح طراحی را مجدد استفاده کند.

طراحی معماری باید دارای خصوصیات زیر باشد:

۱. خصوصیات ساختاری: این جنبه از طراحی معماری مؤلفه‌های سیستم همراه با روشی که این مؤلفه‌ها را دسته‌بندی می‌کند و با یکدیگر ارتباط برقرار می‌نماید، تعریف می‌کند.
۲. خصوصیات عملکردی اضافه: توصیف طراحی باید مشخص نماید که این معماری چگونه نیازهای کارایی، قابلیت اطمینان، امنیت، سازگاری و دیگر خصوصیات تیم را تخمین و برآورده می‌کند.
۳. قابل‌استفاده در خانواده‌های مرتبط: طراحی معماری باید با الگوریتمی قابل تکرار شکل گیرد طوری که بتوان آن را به‌طور مشترک در طراحی خانواده‌ای از گروه‌های مشابه مورداستفاده قرار داد.

انواع مدل‌های نرم‌افزار (معماری)

۱. مدل‌های ساختاری: معماری را به‌صورت مجموعه‌ای سازمان‌یافته از مؤلفه‌های برنامه نشان داد.
۲. مدل‌های زمینه کاری: سطح مجردسازی طراحی را افزایش می‌دهد.
۳. مدل‌های پویا: جنبه‌های رفتاری معماری برنامه را مورد توجه قرار می‌دهند. نشان می‌دهد که چگونه ساختار یا پیکربندی تیم به‌صورت تابعی از وقایع خارجی تغییر می‌کند.

مدیریت پروژه

مدیریت پروژه شامل طرح‌ریزی، نظارت و کنترل افراد، فرآیند و رویدادهایی می‌شود که به‌موازات تکامل نرم‌افزار از مفهوم مقدماتی تا پیاده‌سازی عملی رخ می‌دهد. در نتیجه فعالیت‌های مدیریتی یک طرح پروژه ایجاد می‌شود که این طرح فرآیند و وظایفی را که باید انجام شود، افرادی که کار را انجام می‌دهند و راهکارهای ارزیابی خطرات، کنترل تغییرات و ارزیابی کیفیت را تعیین می‌کنند.

طیف مدیریت شامل عوامل زیر می‌باشد:

- افراد
- محصول
- پروژه
- فرآیند

منظور از افراد، منظور آموزش نرم‌افزار سازان بسیار ماهر و باانگیزه یا جمع‌آوری افراد موردنظر می‌باشد.

افراد دخیل:

- مدیران: مدیر فنی، مدیر تجاری
- تحلیلگران
- طراحان
- برنامه نویسان
- متخصصان فنی: شبکه، بانک اطلاعات، نیروی انسانی و ...
- مشتری
- کاربران

عوامل مؤثر بر مناسب بودن تیم:

- افراد باانگیزه و ماهر
- تعداد افراد متناسب با پروژه
- ساختار مدیریتی
- میزان پیچیدگی مسئله

روش‌های ساختاردهی تیم نرم‌افزاری (ازنظر مانتی):

DD: (Decentralized democratic): تمرکززدایی دمکراتیک

CD: (Decentralization of control): تمرکززدایی کنترل‌شده

CC: (Centrally controlled): تمرکز کنترل‌شده (استبدادی)

DD: تیم مهندسی نرم‌افزار فاقد رهبری دائم است، در عوض هماهنگ‌کنندگان وظایف، برای مدتی کوتاه نصب، سپس جای خود را به افراد دیگر می‌دهند که ممکن است وظایف متفاوتی را هماهنگ

کنند. تصمیم‌گیری در مورد مشکلات و روش کار به‌صورت جمعی صورت گرفته و رابطه افقی بین افراد وجود دارد.

CD: تیم مهندسی نرم‌افزار دارای رهبری مشخص است که وظایف معینی را مشخص می‌کند و رهبران ثانوی مسئول وظایف فرعی هستند. حل مشکلات کماکان از طریق فعالیت‌های گروهی صورت گرفته ولی پیاده‌سازی راهکارها توسط رهبر، میان زیرگروه‌ها تقسیم می‌شود. رابطه بین افراد و زیرگروه‌ها به‌صورت افقی بوده اما بین سلسله مراتب مدیریت رابطه عمودی است.

CC: این نوع تیم دارای رهبری دائم است که هماهنگ‌سازی داخلی تیم، حل مشکلات سطح بالا و تصمیم‌گیری‌ها بر عهده‌ی وی می‌باشد. ارتباط میان رهبر و اعضای تیم کاملاً عمودی است.

الگوهای سازمان‌دهی تیم (آقای کنسانتین):

الگوی بسته: تیم در راستای یک سلسله‌مراتب سنتی از مسئولیت‌ها مشابه CC سازمان‌دهی می‌شود. چنین تیم‌هایی، هنگام ساخت نرم‌افزارهایی خوب هستند که نیاز به کارهای روتین و منظم باشد. ولی برای کارهای خلاقانه و نوآوری مناسب نیستند.

الگوی تصادفی: تیم ساختار سستی دارد و به ظرفیت تک‌تک افراد بستگی دارد. در صورت نیاز به کشفیات فنی یا نوآوری تیمی که از این الگو پیروی می‌کند بهترین نتیجه را دارد اما برای کارهای منظم و روتین این نوع تیم به دردسر می‌افتد.

الگوی باز: در این الگو تلاش می‌شود ساختار تیم به شیوه‌ای باشد که در حین دستیابی به کنترل‌های خاص الگوی بسته به حداکثر نوآوری الگوی تصادفی نیز دست‌یابیم. کارها از طریق همکاری زیاد و تصمیم گیرها مبتنی بر اجماع صورت می‌گیرد و برای مسائل پیچیده بسیار مناسب است اما لزوماً کارایی الگوهای قبل را ندارد.

الگوهای همزمان: اعضای تیم در این روش به زیرگروه‌های مجزا تقسیم و سازمان‌دهی می‌شوند تا بر روی قسمت‌های مختلف پروژه با الگوهای متفاوت کار کنند. ارتباط بین این زیرگروه‌ها کم و از طریق مدیر برقرار می‌باشد.

عوامل مؤثر بر الگوی سازمان‌دهی تیـم:

- دشواری مسئله‌ای که قرار اسـت حل شود
- اندازه‌ی برنامه‌ی حاصل برحسب تعداد خطوط کد یا تعداد توابع (بزرگی پروژه)

- میزان زمـان کنار هم ماندن اعضای تیم
- میزان قابلیت پیمانه‌ای کردن مسئله، یعنی تقسیم مسئلـه به بخش‌های کوچک‌تر
- میزان کیفیت و اطمینان موردنظر
- قطعیـت تاریخ تحویـل
- میزان ارتباطـات موردنیاز برای پروژه

عوامل مسمومیت تیم (عوامل شکست تیم):

- یک فضـای کاری آشفتـه کـه در آن اعضـای تیـم انرژی خـود را هدر داده و توجـه خود را معطـوف اهداف کاری نمی‌کنند.
- ناراحتی‌های زیـاد ناشی از عوامل فنی، کاری و پرسنلی که باعث اصطکاک میان اعضای تیم می‌شود.
- رویه‌های قطعه‌قطعه شده باهماهنگی ضعیـف یـا مدل فرآیندی کـه به‌خوبی مشخـص نشـده یا درسـت انتخاب‌نشده باشـد و باعـث ایجـاد مانـع گردد.
- تعریـف ناواضـح نقش‌ها که منجـر به فقدان مسئولیت‌پذیری و انگشت‌نما شدن می‌شود.
- قرار گرفتن پیوسته و مکـرر در معرض شکسـت که منجر به از بین رفتن اعتمادبه‌نفس و سوء رفتار می‌شود.

تعریف محصول: منظور تعیین اهداف محصول و دامنه کاربرد آن می‌باشد.

تعریف دامنه کاربرد محصول:

نخستین فعالیت مدیریت تعیین دامنه‌ی کاربرد محصول اسـت که باید در سطح فنی و مدیریتی نامبهم و قابل‌درک باشد. بیانی از دامنه کاربرد باید دارای حدومرز باشد؛ یعنی داده‌های کمی مثل تعداد کاربران هم‌زمان و حداکثر زمان پاسخگویـی مجاز به‌طور واضح بیان‌شده و محدودیت‌ها و شرایط تعیین‌کننده مثل هزینـه محصول ذکرشده و عوامل تعدیل آن شرح داده شود.

عوامل مؤثر بر دامنه کاربرد محصول:

محیـط: نرم‌افزاری کـه قـرار اسـت ساخـتـه شـود چگونه در یک سیستم بزرگ‌تر یعنی محیط کـاری قرار می‌گیرد و چـه شرایـط حدی بر آن اعمال می‌شود

اهداف اطلاعـات (ورودی و خروجـی): نرم‌افزار چـه اشیـاء داده‌ای را برای مشتـری تولید می‌کند و به چه اشیاء داده‌ای برای ورود احتیاج دارد

عملکرد و کارایی: به ایـن معناست که نرم‌افزار چه عملی برای تبدیل داده‌های ورودی به داده‌های خروجـی انجـام می‌دهد و چـه ویژگی‌هایی از کارایی وجود دارد که باید به آن‌ها توجه نمود.

تعریف فرآیند: به فاز کلی گفتـه می‌شود که مراحـل ساخـت نرم‌افزار را مشخـص می‌کند و در تمام مراحل لازم‌الاجراست اما مشکل اصلـی آن انتخـاب درسـت مدل تولیـد نرم‌افزار می‌باشد.

عوامل مؤثر بر انتخاب مدل:

- مشتریانی که محصول را درخواست کردند و کسانی که نرم‌افزار را تولید می‌کنند.
- شرایط کلی محصول
- محیط پروژه که نرم‌افزار در آن تولید می‌شود.

تعریف پروژه: شامل عوامل بحرانی موفقیت و یافتن روشی مبتنی بر عقـل برای طرح‌ریزی، نظارت و کنترل پروژه می‌باشد.

عوامل به خطر افتادن پروژه:

- افراد نرم‌افزاری، نیازهای مشتری را درک نکننـد.
- حوزه‌ی محصول به‌خوبی مشخص نشود.
- تغییرات به‌درستی مدیریت نشود.
- فناوری انتخاب‌شده تغییر کند.
- نیازهای تجاری تغییر کند.
- مهلت‌ها واقع‌بینانه نباشـد.
- کاربران مقاومت کننـد.
- حمایت های مالی از بین برود یا به‌طور مناسب به دست نیاید.
- تیم فاقد افراد با مهارت‌های لازم باشـد.
- سـازنـدگـان پـروژه و مدیـران از حداکثـر توانایی‌های خود اسـتفـاده نکنند.