مدرسان شریف

فصل اول

« مقدمه‌ای برای مهندسی نرم‌افزار »

مقدمه

این فصل یکی از مهم‌ترین فصول کتاب می‌باشد که در سال‌های اخیر همواره مورد توجه طراحان سؤال کنکور بوده است. زمانی که از درس مهندسی نرم‌افزار در کنکور ۸ سؤال می‌آمد این فصل همواره دو سؤال در بین سؤالات کنکور داشت. اما به نظر می‌رسد از سال ۹۰ که تعداد سؤالات به ۶ تقلیل یافت، این فصل بیش از ۱ سؤال در کنکور نداشته باشد. به هر حال مباحث موجود در این فصل برای فهم سایر موارد نیز بسیار ضروری می‌باشد لذا به داوطلبان توصیه می‌کنیم که این فصل را به خوبی مورد مطالعه قرار دهند.

در بخش اول این فصل یک سری مفاهیم در ارتباط با ماهیت نرم‌افزار بیان می‌شود. سپس مشکلات کنونی نرم‌افزار و ویژگی‌ها و کاربردهای آن در پایان همین بخش مورد بررسی قرار می‌گیرند. در بخش دوم این فصل، لایه‌های مهندسی نرم‌افزار و ارتباط آن‌ها با متدولوژی توسعه نرم‌افزار بیان می‌شود، سپس به چارچوب فرآیند و فعالیت‌های چتری نرم‌افزار می‌پردازیم. در بخش سوم که مهم‌ترین قسمت این فصل نیز هست مدل‌های فرآیندی مختلف را ارزیابی کرده و ویژگی‌ها، نقاط ضعف و قوت هر یک از این مدل‌ها را مورد بررسی قرار می‌دهیم. در قسمت آخر این فصل که مربوط به بهبود فرآیند نرم‌افزار می‌باشد، در مورد مفاهیم مربوط به آن و مدل‌های مختلف مورد استفاده برای بهبود فرآیند نرم‌افزاری به منظور تضمین کیفیت فرآیند نرم‌افزار را مورد بحث قرار می‌دهیم.

تعریف چند مفهوم

قبل از این که به بحث در ارتباط با مهندسی نرم‌افزار بپردازیم خوب است که سه مفهوم کلیدی را که در مهندسی نرم‌افزار نقش اساسی دارند بهتر بشناسیم. برای شناخت بهتر هر فرآیند مهندسی ابتدا باید محصولی را که قرار است طی فرآیند مهندسی به دست بیاید بشناسیم، سپس فرد و یا افرادی را که در به دست آمدن آن محصول تحت یک چارچوب مشخص فعالیت می‌کنند، بشناسیم. در پایان باید در مورد خود فرآیند مهندسی نرم‌افزار که منجر به تولید آن محصول می‌شود، اطلاعات کامل را کسب کنیم.

نرم‌افزار چیست؟

نرم‌افزار، دستورالعمل‌ها یا برنامه کامپیوتری می‌باشد که در صورتی که به درستی اجرا شود می‌تواند خصوصیت‌ها، عملکردها و کارآیی مطلوب و خواسته شده از خودش را به شکل مطلوب در اختیار ما قرار بدهد. علاوه بر آن نرم‌افزار شامل ساختمان داده‌ای می‌باشد که از طریق آن برنامه‌ها قادر به دستکاری و مدیریت اطلاعات به شکل مطلوب و مورد نیاز می‌باشند. همچنین همواره به همراه یک نرم‌افزار مستنداتی وجود دارند که در مورد عملکرد و نحوه استفاده از برنامه‌هایی که باید اجرا شوند توضیحات مناسبی در اختیار کاربران قرار می‌دهند.

مهندس نرم افزار کیست؟

مهندس نرم‌افزار شخصی است که نرم افزاری را تولید و پشتیبانی می‌کند تا کاربران نرم‌افزار قادر به استفاده از آن به شکل مطلوب باشند. نرم‌افزار تولید شده باید مطابق با نیازهای کاربران ساخته شود و بتواند نیازهای آن‌ها را برطرف کند.

مهندسی نرم افزار چیست؟

به مجموعه روش‌ها، فن‌آوری‌ها و ابزارهایی که در فرآیندهای سازگارپذیر و چابک برای رسیدن به محصولی با کیفیت بالا در تولید نرم‌افزار مورد استفاده قرار می‌گیرند و مبتنی بر **اصول مهندسی** می‌باشند، مهندسی نرم‌افزار می‌گویند.

نکته۱: نرم‌افزار هم می‌تواند محصول باشد و هم وسیله‌ای برای تحویل محصول (اطلاعات)

نکته۲: بر خلاف بسیاری از افراد که مهندسی نرم‌افزار را همان برنامه‌نویسی کامپیوتر می‌دانند، باید متذکر شویم که برنامه‌نویسی تنها بخش کوچکی از فرآیند مهندسی نرم‌افزار می‌باشد.

مثال۱: کدام‌یک از گزینه‌های زیر درست می‌باشد؟

۱) برنامه‌نویسی مهم‌ترین بخش مهندسی نرم‌افزار می‌باشد.

۲) مستند سازی مهم‌ترین بخش مهندسی نرم‌افزار می‌باشد.

۳) هدف اصلی انجمن‌های نرم‌افزاری تلاش برای توسعه‌ی فناوری‌هایی می‌باشدکه نگهداری و توسعه‌ی نرم‌افزار با کیفیت بالا را سریع‌تر، ارزان‌تر و راحت‌تر می‌کند.

۴) هیچ‌کدام

☑ **پاسخ:** گزینه «۳» با توجه به مدل‌های مختلف فرآیندی اهمیت بخش‌های مختلف نرم‌افزاری می‌تواند دچار تغییر شود. بنابراین گزینه ۱ و ۲ نادرست می‌باشند.

مشکلات کنونی نرم‌افزار

از بدو پدید آمدن نرم‌افزار تاکنون، همواره نرم‌افزار با مشکلاتی مواجه بوده است. این مشکلات به طور خلاصه عبارتند از:

۱ـ رشد فناوری: در طی دهه‌های اخیر، سخت‌افزار با رشد چشم‌گیری مواجه بوده است و سرعت رشد آن روز به روز در حال افزایش می‌باشد. این رشد سریع این الزام را بر روی نرم‌افزار ایجاد می‌کند تا برای تطبیق خود با سخت‌افزار به صورت دائم در حال رشد باشد که این خود باعث بروز مشکلاتی در عرصه نرم‌افزار می‌شود.

۲ـ تغییرات مداوم بازار و نیازهای مشتری: یکی دیگر از مشکلات نرم‌افزار، رشد سریع نیازهای موجود در بازار و به وجود آمدن تغییرات زیاد در آن‌ها می‌باشد که باعث فشردگی زیاد در عرصه تولید نرم‌افزار شده است. به عبارتی دیگر هر چه یک نرم‌افزار زودتر به بازار بیاید احتمال موفقیت آن بیشتر خواهد بود و همین امر باعث ایجاد فشار بر روی زمان تولید نرم‌افزار و کوتاهتر کردن فرآیند آن می‌شود.

۳ـ گستردگی نرم‌افزار و عدم امکان تغییر سریع در آن: یکی دیگر از مشکلات نرم‌افزار علاقمند شدن عموم مردم به استفاده از آن در زندگی روزمره‌ی خود می‌باشد. در واقع طی دهه‌ی نود نرم‌افزار علاوه بر مصارف صنعتی، به حوزه‌ی مصرفی خانه‌ها و استفاده شخصی افراد نیز وارد شد. همه‌گیر شدن نرم‌افزار سبب می‌شود تا امکان تغییر در نرم‌افزار و تعویض آن با نرم‌افزار جدید با مشکل روبرو شود و نتوان به سرعت این کار را انجام داد.

برقراری امنیت و تضمین کیفیت نرم‌افزار: با پا به عرصه نهادن اینترنت در دهه‌ی نود میلادی به خانه‌های کاربران، نیاز به ایجاد نرم‌افزارهایی که امکان برقراری ارتباط بین کامپیوترها و مدیریت اطلاعات و نیز حفاظت اطلاعات را داشته باشند بیش از پیش احساس شد. در حال حاضر با رشد روزافزون تهدیدها و حملات کامپیوتری حفاظت از نرم‌افزار بسیار ضروری است. این کار باعث می‌شود که اقدامات بسیاری در زمینه‌ی امنیت نرم‌افزار صورت گیرد. همچنین نرم‌افزار تولیدی باید دارای کیفیت و قابل اعتماد باشد.

قابلیت توسعه سامانه‌های قدیمی و نرم‌افزارهای کنونی: وجود سامانه‌های قدیمی در سازمان‌ها خود می‌تواند باعث بروز مشکلات فراوانی در حوزه‌ی نرم‌افزار شود. این که چطور می‌خواهیم آن‌ها را بر اساس نیازهای کنونی به روز کنیم و چگونه از آن‌ها پشتیبانی کنیم و نیز نگهداری آن‌ها جزو بحث‌های امروزی حوزه‌ی نرم‌افزار می‌باشد. همچنین در مورد نرم‌افزارهای کنونی نیز باید قابلیت توسعه آن‌ها مورد بررسی قرار گیرد.

مثال۲: کدام‌یک از گزینه‌های زیر دیگر جزو نگرانی‌های حوزه‌ی نرم‌افزار به حساب نمی‌آید؟

۱) چرا هزینه‌ی سخت‌افزارها این قدر بالا می‌باشد؟

۲) چرا در اندازه‌گیری پیشرفت پروژه‌های کنونی مشکل وجود دارد؟

۳) چرا نمی‌توانیم همه خطاها را قبل از اینکه محصول را به مشتری ارائه بدهیم به دست بیاوریم؟

۴) چرا زمان زیادی طی می‌شود تا تولید نرم‌افزار به پایان برسد؟

☑ **پاسخ:** گزینه «۱» با توجه به رشد روزافزون سخت‌افزار، دیگر نباید نگران قیمت زیاد سخت‌افزار که در گذشته بر روی نرم‌افزار تأثیر می‌گذاشت باشیم.

علاوه بر سه مورد فوق **سایر نگرانی‌های حوزه نرم‌افزار عبارتند از:**

۴) چرا زمان و تلاش بسیاری را برای نگهداری و برطرف کردن مشکلات نرم‌افزارهای موجود هزینه می‌کنیم؟

۵) چرا هزینه تولید نرم‌افزارها این قدر بالا می‌باشد؟

ویژگی‌های نرم‌افزار

نرم‌افزار ویژگی‌های منحصر به فردی دارد که باعث تمایز آن از سایر محصولات فیزیکی (که طی فرآیندهای مهندسی به دست می‌آیند) می‌شود. تولید یک نرم‌افزار بیش از اینکه یک فرآیند فیزیکی باشد یک فرآیند منطقی است. بنابراین برای به دست آوردن بینش نسبت به نرم‌افزار (و نیز فهم کامل مهندسی نرم‌افزار) لازم است که تفاوت‌ها را بشناسیم.

۱ـ نرم‌افزار توسعه و یا مهندسی می‌شود، نه این که به سبک کلاسیک ساخت و تولید شود.

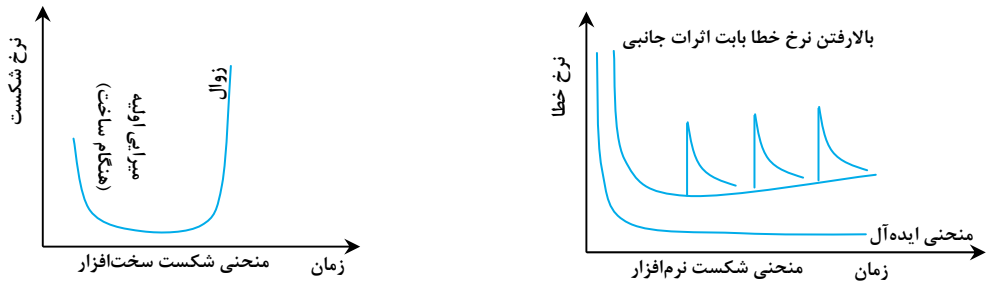
شباهت بین فرآیند توسعه نرم‌افزار و فرآیند ساخت و تولید محصول فیزیکی، در نیاز مبرم آن‌ها به طراحی و پیاده‌سازی خوب و همکاری و هماهنگی مناسب بین اعضای تولیدکننده آن‌ها می‌باشد. اما چیزی که باعث بروز تفاوت در میان این دو می‌شود ماهیت متفاوت این فعالیت‌ها با یکدیگر است. برای توسعه‌ی نرم‌افزار بیشتر نیازمند فعالیت‌های فکری و منطقی هستیم، در حالیکه در تولید محصولات فیزیکی فعالیت‌های فیزیکی بیشتر به کار می‌آیند. در واقع مدیریت پروژه‌های نرم‌افزاری بسیار متفاوت از مدیریت پروژه‌های تولیدی می‌باشد. به عبارت دیگر هزینه تولید و توسعه نرم‌افزار بیشتر بر روی **فعالیت‌های مهندسی** متمرکز می‌باشد در حالی که هزینه تولید و توسعه‌ی محصول فیزیکی بیشتر بر روی **مواد خام و اولیه و تولید آن‌ها** متمرکز است.

نکته۳: به دلیل تمرکز بیشتر تولید و توسعه نرم‌افزار بر روی فعالیت‌های مهندسی، سرعت تغییر در این فعالیت‌ها نسبت به فعالیت‌های مربوط به توسعه محصول فیزیکی بسیار سریع‌تر است.

۲ـ نرم‌افزار دور انداختنی نیست.

یکی از تفاوت‌های عمده بین سخت‌افزار و نرم‌افزار در این است که سخت‌افزار پس از طی شدن یک بازه زمانی از مدت زمان استفاده آن، دچار استهلاک شده و دیگر قابل استفاده نیست. در واقع به علت استفاده زیاد از آن ممکن است خللی در کارآیی سخت‌افزار به وجود بیاید. به عبارت دیگر سخت‌افزارها تاریخ مصرف دارند و بعد از مدتی استفاده کم‌کم دور انداختنی می‌شوند. اما نرم‌افزار در حالت ایده‌آل بعد از مدتی که از تولید آن گذشت و اشکالات مربوط به آن برطرف شد دیگر برای همیشه و بدون اشکال قابل استفاده می‌باشد. اما تغییر در نیازهای کاربر باعث بروز خلل در کارآیی یک نرم‌افزار می‌شود و با گذشت زمان به علت بروز این تغییرات، نرم‌افزار دیگر قادر نخواهد بود نیازهای کاربر را برآورده کند. بنابراین برای برآورده کردن نیازهای جدید کاربر نیازمند اعمال تغییرات در نرم‌افزار می‌باشیم که این اعمال تغییرات باعث بروز مشکلاتی در نرم‌افزار می‌شود و ممکن است اعمال این تغییرات با خطا مواجه شود. به طور خلاصه تفاوت اصلی نرم‌افزار و سخت‌افزار را می‌توان این گونه بیان کرد که :

سخت افزار دور انداختنی است و بعد از مدتی استفاده به دلیل استهلاک کارآیی خودش را از دست می‌دهد در حالی که نرم‌افزار دور انداختنی نیست و رو به زوال می‌رود. یعنی به علت بروز تغییر در نیازهای کاربران نیازمند اعمال تغییر در آن هستیم که همین اعمال تغییر مداوم باعث رو به زوال رفتن نرم‌افزار می‌شود.



شکل ۱. شکل منحنی شکست نرم افزار

۳ـ اگرچه صنعت امروزی در جهت تولید بر اساس مونتاژ قطعات سخت‌افزاری و تولید جداگانه هر کدام در مکان و زمان مختلف حرکت می‌کند اما نرم‌افزار بر اساس نیاز مشتری و به صورت سفارشی ساخته می‌شود.

یکی دیگر از تفاوت‌های اساسی نرم‌افزار و سخت‌افزار در این است که در تولید سخت‌افزار در حال حاضر از قابلیت استفاده از مؤلفه‌های آماده بسیار استفاده می‌شود. در واقع ماهیت سخت‌افزار این امکان را به ما می‌دهد تا هر یک از اجزای یک محصول را (حتی در مکان‌های متفاوت) به صورت جداگانه بسازیم و درنهایت آن‌ها را با یکدیگر مونتاژ کنیم. در حوزه نرم‌افزار این ایده به تازگی مطرح و استفاده از مؤلفه‌های آماده جهت ساخت نرم‌افزار مرسوم شده است.

همچنین در حوزه نرم‌افزار به دلیل تاکید آن بر روی تولید به صورت سفارشی، ایده‌های نوینی پدیدار شده‌اند که باعث بروز خلاقیت و نوآوری بیشتری در تولید نرم‌افزار شده است. پس به طور خلاصه تفاوت سوم نرم‌افزار و سخت افزار را این‌گونه می‌توان بیان کرد که:

نکته۴: امکان استفاده از مؤلفه‌های آماده در سخت‌افزار بیشتر از نرم‌افزار می‌باشد ولی نرم‌افزار قابلیت سفارشی‌سازی بیشتری نسبت به سخت‌افزار دارد.

مثال ۳: کدام‌یک از گزینه‌های زیر در ارتباط با تفاوت سخت‌افزار و نرم‌افزار نادرست می‌باشد؟

۱) نرم‌افزار رو به زوال می‌رود در حالی‌که سخت‌افزار فرسوده می‌شود.

۲) بروز خلاقیت و نوآوری در سخت‌افزار بیش از نرم‌افزار می‌باشد.

۳) استفاده از قطعات قابل استفاده مجدد در سخت‌افزار رواج بیشتری نسبت به نرم‌افزار دارد.

۴) سرعت تغییرات در سطح نیازمندی‌های مشتری در نرم‌افزار به مراتب بیشتر از سخت‌افزار می‌باشد

☑ **پاسخ:** گزینه «۲» بروز خلاقیت و نوآوری در نرم‌افزار به دلیل این که نرم‌افزار یک فعالیت مهندسی است و بیشتر متکی بر تلاش افراد می‌باشد بیش از سخت‌افزار است.

کاربردهای نرم‌افزار

امروزه کاربردهای مختلفی برای نرم‌افزار در حوزه‌های مختلف به وجود آمده است که این امر موجب پیچیدگی بیشتر فرآیند مهندسی نرم‌افزار می‌شود. قبل از این که به بحث در مورد کاربردهای مختلف نرم‌افزار بپردازیم که این کاربردها اکثراً از نظر محتوا با یکدیگر متفاوت می‌باشند خوب است که اندکی راجع به عوامل مؤثر بر روی محتوای نرم‌افزارها بحث کنیم. محتوای نرم‌افزار از طریق دو عامل زیر تعیین می‌شود:

۱ـ محتوای اطلاعات: محتوای اطلاعات شامل شکل، مفهوم و محتوای مربوط به اطلاعات ورودی و خروجی می‌شود.

۲ـ قطعیت اطلاعات: این معیار، به قابلیت پیش‌بینی سفارش و زمان‌بندی اطلاعات اشاره دارد.

در ادامه به تشریح برخی از کاربردهای نرم‌افزار به طور مختصر می‌پردازیم.

۱ ـ نرم‌افزارهای سیستمی (System Software):

نرم‌افزارهای سیستمی مجموعه‌ای از برنامه‌ها هستند که برای سرویس دادن به سایر برنامه‌ها نوشته می‌شوند. بعضی از نرم‌افزارهای سیستمی مانند کامپایلرها، ویرایشگرها، برنامه‌های مدیریت فایل و ... ساختار اطلاعاتی پیچیده، ولی تعیین شده‌ای دارند. (نرم‌افزار زمانی تعیین شده می‌باشد که بتواند ترتیب و زمان ورودی‌ها، پردازش‌ها و خروجی‌های قابل پیش‌بینی را پردازش کند) و بعضی دیگر از این نرم‌افزارها مانند مؤلفه‌های سیستم عامل، محرک‌ها، نرم‌افزارهای شبکه و پردازشگرهای ارتباطی مقدار زیادی اطلاعات تعیین نشده را پردازش می‌کنند.

۲ ـ نرم‌افزارهای کاربردی (Application Software) :

این حوزه شامل نرم‌افزارهایی می‌باشد که برای حل یک مشکل خاص تجاری ایجاد شده‌اند. نرم‌افزارهای این حوزه داده‌هایی که از فرآیندهای تجاری به دست می‌آید را به صورتی پردازش می‌کنند که تصمیم‌گیری فنی/مدیریتی یا عملیات تجاری را تسهیل کنند.

۳ ـ نرم‌افزارهای علمی/مهندسی (Engineering/Scientific software):

این حوزه شامل نرم‌افزارهایی می‌باشد که برای حل مشکلات مربوط به رشته‌های مهندسی و یا علمی به کار می‌روند. نمونه‌های متعارف آن‌ها عبارتند از سیستم‌های زمین‌شناسی، تحلیل استرس در شاتل فضایی و نمونه‌های دیگر مشابه آن‌ها .

۴ ـ نرم‌افزارهای توکار ـ تعبیه شده (Embedded Software):

نرم‌افزارهای تعبیه شده، نرم‌افزارهایی هستند که برای انجام یک وظیفه خاص در محصولات و یا سیستم‌های دیگر تعبیه می‌شوند. هدف از ایجاد آن‌ها پیاده‌سازی و کنترل خصوصیت‌ها و کارکردهایی برای سیستم‌ها و یا کاربران نهایی می‌باشد. نرم‌افزارهای تعبیه شده گاهی ممکن است که کارکردهای محدود و داخلی داشته باشند (مانند کنترل صفحه کلید یک مایکروفر) و گاهی اوقات ممکن است کارکردهای تأثیرگذار و حیاتی داشته باشند (مانند سیستم کنترل چرخ‌های اتومبیل و یا کنترل سوخت).

۵ ـ نرم‌افزارهای خط تولیدی(Product-Line Software) :

نرم‌افزارهایی هستند که با هدف فراهم آوردن قابلیت‌های مشخصی برای استفاده توسط کاربران زیاد با سلیقه‌ها و نیازهای مختلف مورد استفاده قرار می‌گیرند. این نرم‌افزارها ممکن است یا بر روی بازار محدود و مخصوصی متمرکز شوند (مانند محصول کنترل انبار) و یا بازار هدف گسترده‌تری را نشانه بگیرند (مانند نرم‌افزار پردازش متن،گرافیک کامپیوتری، برنامه‌های تفریحی، برنامه‌های چندرسانه‌ای، مدیریت پایگاه داده، کاربردهای مالی، تجاری و شخصی)

۶ـ نرم‌افزارهای مبتنی بر وب (Web-application):

نرم‌افزارهای مبتنی بر وب می‌توانند از یک صفحه ساده وب که حاوی کدهای HTML ساده می‌باشد تا یک نرم‌افزار کاربردی تجارت الکترونیکی پیچیده و حساس را شامل شوند. در حال حاضر به خاطر کاربردهای تجارت الکترونیکی، این نرم‌افزارها علاوه بر داشتن خصوصیت‌هایی به عنوان یک صفحه وب باید دارای قابلیت یکپارچه شدن با پایگاه داده‌های شرکت‌ها و سایر برنامه‌های کاربردی تجاری نیز باشند.

۷ـ نرم‌افزارهای هوش مصنوعی (Artificial Intelligence Software) :

نرم‌افزارهایی هستند که با استفاده از روش‌های هوش مصنوعی به حل مسائل پیچیده که توسط تجزیه و تحلیل و محاسبات مستقیم قابل حل نیستند، می‌پردازند. از نمونه کاربردهای این نرم‌افزارها می‌توان در روباتیک، سیستم‌های خبره، سیستم‌های مبتنی بر یادگیری ماشین و شبکه‌های عصبی مصنوعی اشاره کرد. اما علاوه بر این چالش‌ها و کاربردهای مختلف نرم‌افزار، چالش‌ها و کاربردهای جدیدی در حوزه نرم‌افزار پدید آمده است که آن‌ها را در ادامه‌ی همین کاربردها و با شماره‌های ۸ و ۹ عنوان می‌کنیم.

۸ ـ محاسبات همه‌جاحاضر(Ubiquitous Computing) :

رشد سریع شبکه‌های بی‌سیم ممکن است ما را به سمت محاسبات توزیع شده واقعی هدایت کند. چالش جدیدی که برای مهندسی نرم‌افزار پدید آمده است، توسعه‌ی سیستم و نرم‌افزار کاربردی برای دستگاه‌های کوچک، رایانه‌های شخصی و سیستم‌های بنگاهی برای ارتباط در شبکه به صورت گسترده، می‌باشد.

۹ ـ نرم‌افزارهای متن باز(Open source software) :

امروزه گرایش زیادی به سمت در اختیار دادن کدهای منبع مربوط به کاربردهای سیستمی مختلف (نظیر سیستم عامل‌ها، پایگاه داده و محیط‌های توسعه) برای مشتریانی که می‌توانند اصلاحاتی با توجه به اقتضاهای محلی خودشان در آن‌ها اعمال کنند، به وجود آمده است. چالشی که در اینجا برای مهندسی نرم‌افزار مطرح می‌شود این است که کدهایی را تولید کنند که خود تعریف باشند تا زمانی که آن‌ها را در اختیار فرد جدید قرار می‌دهیم به راحتی و بدون نیاز به کمک فرد دیگری بتواند از آن‌ها استفاده کند.

مثال۴: کدام‌یک از گزینه‌های زیر در ارتباط با نوع نرم‌افزار نادرست می‌باشد؟

۱) نرم‌افزار Word جزو نرم‌افزار خط تولید به حساب می‌آید.

۲) نرم‌افزار کنترل سیستم ماکروفر جزو نرم‌افزارهای تعبیه شده به حساب می‌آید.

۳) نرم‌افزار حساب‌داری هلو جزو نرم‌افزارهای علمی مهندسی به حساب می‌آید.

۴) نرم‌افزار پردازش تصویر جزو نرم‌افزارهای کاربردی به حساب می‌آید.

☑ **پاسخ:** گزینه «۴» نرم‌افزار پردازش تصویر جزو نرم‌افزارهای هوش مصنوعی به حساب می‌آید.

مثال۵: کدام دسته از نرم‌افزارها، ارتباط مستقیم با حل مشکلات تجاری ندارند؟

۱) نرم‌افزار سیستمی ۲) نرم‌افزار کاربردی ۳) نرم‌افزار تحت وب ۴) نرم‌افزار خط تولید

☑ **پاسخ:** گزینه «۱» نرم‌افزار سیستمی بیشتر برای سرویس دادن به سایر نرم‌افزارها نوشته می‌شود و ارتباط مستقیمی با نیاز تجاری ندارد.

افسانه‌های نرم‌افزاری

افسانه‌های نرم‌افزاری را می‌توان در روزهای ابتدایی و تصوراتی که مربوط به آغاز کار شروع عمر مهندسی نرم‌افزار می‌باشد، ردیابی کرد. این افسانه‌ها معمولاً دارای یک سری خصیصه‌ی ثابت می‌باشند، مثلاً ممکن است به صورت شهودی احساس شوند، یا حتی ممکن است در فرضیه معقول به نظر برسند اما چیزی که همواره در مورد آن‌ها صدق می‌کند این است که همواره در عمل با شکست مواجه شده‌اند و هیچ‌گاه درست از آب درنیامده‌اند.

این افسانه‌ها به سه دسته‌ی افسانه‌های مدیریتی، افسانه‌های مشتری و افسانه‌های توسعه‌دهنده تقسیم می‌شوند که در ادامه به توضیح هر کدام از آن‌ها می‌پردازیم.

۱ـ افسانه‌های مدیریتی

مدیران مسئول نرم‌افزار معمولاً تحت فشار بودجه، مجبور به تمام کردن نرم‌افزار سر موعد زمانی و بهبود کیفیت نرم‌افزار می‌باشند. این مدیران ممکن است دارای یک سری فرضیاتی باشند که شاید از نظر منطقی درست باشند اما در عمل ثابت شده است که اینها نادرست هستند.

این فرضیات یا افسانه‌ها عبارتند از:

ـ در صورتی که دارای کتابی باشیم که پر از استانداردها و رویه‌ها برای ساختن نرم‌افزار است، تیم نرم‌افزاری از نظر دانش مورد نیاز تأمین می‌باشد و دیگر به چیزی نیاز ندارد.

حتی اگر چنین کتابی باشد که تمام موارد کیفی در آن گنجانده شده باشد، آیا مهندسین نرم‌افزار به آن دسترسی دارند؟ آیا از آن استفاده می‌کنند؟ آیا این کتاب کامل هست؟ آیا برای این پروژه قابل استفاده می‌باشد؟ آیا می‌توان هم به کتاب رجوع کرد و هم نرم‌افزار را به موقع آماده کرد؟ در بسیاری از موارد پاسخ این سؤالات منفی خواهد بود و بودن و داشتن چنین کتابی کمکی به توسعه‌ی بهتر و با کیفیت‌تر نرم‌افزار نمی‌کند.

ـ در صورتی که از زمان‌بندی عقب بیفتیم، با اضافه کردن چند برنامه‌نویس می‌توانیم عقب ماندگی را جبران کنیم.

مهندسی نرم‌افزار یک فرآیند مکانیکی نیست که با اضافه کردن نیروی کاری بتوانیم کار آن را سرعت ببخشیم. حتی در بسیاری از موارد اضافه کردن نیروی کاری به پروژه، باعث عقب افتادن کارها می‌شود. علت این امر آن است که با اضافه شدن نیروی کاری نیازمند صرف زمانی برای آموزش به آن‌ها می‌باشیم، که همین امر سبب می‌شود تا حد بسیار زیادی از بهره‌وری نیروهای کاری کنونی کاسته شود (زیرا که وظیفه‌ی آموزش بر عهده‌ی آن‌ها می‌باشد). می‌توان در یک حالت از پیش برنامه‌ریزی شده و به خوبی هدایت شده، افراد را به پروژه اضافه کرد.

ـ درصورتی که یک نرم‌افزار را به شخص سومی واگذار کنیم و آن‌را برون سپاری (out sourcing) کنیم، مسئولیت مدیر پروژه به صفر کاهش می‌یابد.

حتی زمانی که نرم‌افزار را به شخص سوم برون‌سپاری می‌کنیم، مدیر پروژه باید راهکارهایی برای تضمین کیفیت آن و کنترل پروژه انجام دهد. در واقع بدون اتخاذ این سیاست‌ها و بدون تأیید فرسنگ‌شمارهایی (milestones) برای کنترل کیفیت پروژه نمی‌توان به یک نرم‌افزار با کیفیت دست یافت.

۲ـ افسانه‌های مشتری

مشتریان خواهان نرم‌افزار می‌توانند طیف گسترده‌ای را شامل شوند. مشتری می‌تواند یک فرد کاملاً فنی و آشنا به فناوری اطلاعات، یک فرد بازاری و یا هر فرد یا سازمان دیگری با هر سطح سواد و دانشی، باشد. در بسیاری از موارد، مشتری یک سری باورهای غلط در مورد نرم‌افزار دارد که علت اصلی آن این است که مدیر پروژه‌ی نرم‌افزار و مهندس‌های نرم‌افزار تلاش کمی برای اصلاح این باورهای غلط کرده‌اند. این باورها یا افسانه‌های غلط، سبب می‌شود تا توقعات نادرست برای مشتری ایجاد شود که همین امر موجب نارضایتی او از محصول می‌شود. در ادامه به معرفی تعدادی از این باورهای غلط می‌پردازیم.

ـ یک توصیف کلی از اهداف سیستم برای شروع به نوشتن برنامه کفایت می‌کند، باقی جزئیات در حین نوشتن می‌توانند اضافه شوند.

اگرچه بیان کلی و جامع از نیازمندی‌ها همواره امکان‌پذیر نمی‌باشد، اما بیان ابهام‌آمیز از اهداف نیز فاجعه‌انگیز است و منجر به شکست می‌شود. نیازمندی‌های بدون ابهام (که معمولاً به صورت تکاملی یافت می‌شوند) تنها تحت ارتباطات مؤثر و مداوم بین مشتری و مهندس نرم‌افزار به دست می‌آیند.

ـ نیازمندی‌های نرم‌افزار به سرعت در حال تغییر هستند، این تغییرات می‌توانند به سرعت اعمال شوند چون نرم‌افزار انعطاف‌پذیر است.

این امر که نیازمندی‌ها به سرعت تغییر می‌کنند امر درستی است، اما اثر این تغییرات با توجه به زمانی که به وقوع می‌پیوندند متفاوت می‌باشد. به طور کلی هر چه نیازمندی در مراحل ابتدایی‌تر تغییر کند، هزینه‌های ناشی از آن کمتر خواهد بود. بنابراین باید تلاش کرد که مشتری تغییرات دلخواه خود را تا جایی که امکان دارد در مراحل ابتدایی کار اعمال نماید.

۳ـ افسانه‌های توسعه دهنده

علاوه بر مشتری و مدیر پروژه، برنامه‌نویسان نیز یک سری باورهای غلط دارند که در طول عمر نسبتاً طولانی برنامه‌نویسی شکل گرفته است، اما به مرور زمان عکس آن‌ها ثابت شده است.

ـ پس از طراحی نرم‌افزار و اجرا شدن آن، کار ما تمام است.

تحقیقات صنعتی نشان داده‌اند که بین ۶۰ تا ۸۰ درصد تمام تلاشی که بر روی نرم‌افزار شکل می‌گیرد بعد از زمانی است که برای اولین بار به مشتری تحویل داده می‌شود. با تحویل نرم‌افزار به مشتریان، بسیاری از اشکالات نرم‌افزاری یافت می‌شوند که همین امر سبب طولانی شدن پروسه‌ی تولید نرم‌افزار می‌شود.

ـ قبل از این‌که نرم‌افزار نوشته شود راهی برای ارزیابی کیفیت آن وجود ندارد.

با بازبینی نرم‌افزار بعد از تمام شدن هر یک از فعالیت‌های چارچوبی می‌توان از کیفیت آن اطمینان حاصل کرد. در واقع بازبینی فنی، یک فیلتر کیفی است که باعث می‌شود به صورت مؤثری کلاس‌های خطای مربوط به نرم‌افزار را پیدا کنیم.

ـ تنها محصول کاری که باید در یک پروژه موفق تحویل داده شود برنامه‌ای است که کار کند.

محصول کاری تنها یکی از قلم‌های اطلاعاتی مربوط به پیکربندی نرم‌افزار می‌باشد. پیکربندی نرم‌افزار شامل قلم‌های پیکربندی مختلفی نظیر محصولات کاری (مدل‌ها، مستندات، طرح‌ها) می‌شود که این اقلام می‌توانند به موفق شدن فرآیند مهندسی نرم‌افزار کمک شایانی کنند و نیز برای پشتیبانی نرم‌افزار بسیار حایز اهمیت می‌باشند.

ـ مهندس نرم افزار برای ما مستنداتی را فراهم می‌کند که لازم نیستند و باعث کندتر شدن کارها می‌شوند.

مهندسی نرم‌افزار در مورد تولید مستندات نمی‌باشد، مهندسی نرم‌افزار در مورد تولید محصول با کیفیت می‌باشد. کیفیت بیشتر باعث کاهش دوباره کاری‌ها می‌شود و کاهش دوباره کاری‌ها باعث کمتر شدن هزینه‌ها و سریع‌تر انجام شدن کارها می‌شود. این مستندات می‌توانند به با کیفیت‌تر شدن محصول کمک شایانی بکنند.

لایه‌های مهندسی نرم‌افزار

مهندسی نرم‌افزار یک فرآیند لایه‌ای می‌باشد. به عبارت دیگر مهندسی نرم‌افزار از چهار لایه تشکیل شده است که لایه‌ی زیرین آن لایه‌ی کیفیت می‌باشد که بیانگر این است که در هر یک از لایه‌های مهندسی نرم‌افزار، کیفیت نرم‌افزار باید تضمین شود. در شکل ۲ این چهار لایه نشان داده شده است.

۱ـ لایه کیفیت: مدیریت جامع کیفیت (Total Quality Management)، شش سیگما (Six Sigma) و فلسفه‌های مشابه باعث به وجود آمدن فرهنگی برای پیشرفت مداوم فرآیند می‌شوند که این فرهنگ ما را به سمت توسعه‌ی روش‌های کاراتر مهندسی نرم‌افزار هدایت می‌کند. در واقع تمرکز بر کیفیت بستری است که مهندسی نرم‌افزار را پشتیبانی می‌کند.

۲ـ لایه فرآیند: پایه و اساس مهندسی نرم‌افزار لایه فرآیند می‌باشد. لایه‌ی فرآیند مهندسی نرم‌افزار مانند ریسمانی است که لایه‌های فناوری را در کنار یکدیگر نگه داشته است و باعث تحویل به موقع و با کیفیت خواسته شده نرم‌افزار می‌شود. به عبارت دیگر این لایه چارچوبی را تعریف می‌کند که برای تحویل کارهای مربوط به فناوری مهندسی نرم‌افزار باید برقرار باشد.

فرآیند نرم‌افزار پایه‌ای را برای کنترل مدیریت پروژه‌های نرم‌افزار بنا می‌کند که این پایه را در جاهایی ایجاد می‌کند که روش‌های فنی مورد نیاز باشد و یا محصولات کاری تولید شده یا نقاط عطفی پایه‌گذاری شود و یا این که کیفیتی تضمین شده و تغییراتی به درستی مدیریت شود.

۳ـ لایه روش‌ها: روش‌های مهندسی نرم‌افزار شیوه‌هایی را برای ساخت نرم‌افزار فراهم می‌آورند. این روش‌ها شامل طیف وسیعی از وظایف همچون برقراری ارتباط، تحلیل نیازمندی‌ها، مدل‌سازی طراحی، ساخت برنامه، آزمایش و پشتیبانی می‌شود. این روش‌ها بر مبنای یک سری اصول خاص بنا شده‌اند که هر کدام از این اصول بر ناحیه‌ی خاصی از فناوری کنترل دارند.

۴ـ لایه ابزار: ابزار مهندسی نرم‌افزار پشتیبانی خودکار و یا نیمه خودکاری برای دو لایه‌ی قبلی (فرآیند و روش‌ها) فراهم می‌کند. در مهندسی نرم‌افزار تلاش ما این است که خروجی‌های حاصل از ابزارها را در بخش‌های مختلف با یکدیگر یکپارچه کنیم تا در نتیجه آن اطلاعاتی که توسط یک ابزار تولید می‌شود بتواند توسط ابزار دیگر مورد استفاده قرار گیرد.

شکل ۲. شکل لایه‌های نرم‌افزار

نکته ۵: CASE (Computer-aided software engineering) ابزارهای کمکی می‌باشند که به کمک آن‌ها می‌توانیم به صورت خودکار و یا نیمه خودکار بعضی از فعالیت‌های مهندسی نرم‌افزار را انجام دهیم. در واقع هدف اصلی آن تسهیل فعالیت‌های مختلف مهندسی نرم‌افزار می‌باشد.

مثال ۶: کدام‌یک از گزینه‌های زیر در ارتباط با لایه‌های مهندسی نرم‌افزار نادرست می‌باشد؟

۱) پایه و مبنای مهندسی نرم‌افزار کیفیت نرم‌افزار می‌باشد.

۲) در صورتی که خروجی یک ابزار توسط ابزار دیگر قابل استفاده باشد، می‌توانیم از ابزارهای مختلف برای قسمت‌های مختلف استفاده کنیم.

۳) کیفیت نرم‌افزار باید قبل از مدل فرآیندی آن مشخص شود.

۴) روش‌های مختلف مهندسی نرم‌افزار بعد از مشخص شدن مدل فرآیندی مشخص می‌شوند.

☑ **پاسخ:** گزینه «۱» پایه و اساس مهندسی نرم‌افزار فرآیند می‌باشد.

متدولوژی نرم‌افزار

تمامی پروژه‌های نرم‌افزاری از چهار لایه مذکور تشکیل شده‌اند. به عبارت دیگر هر پروژه‌ی نرم‌افزاری، مدل فرآیندی برای انجام پروژه‌اش دارد، همچنین روش‌های مختلفی را در قسمت‌های مختلف پروژه برای انجام فازهای تعریف شده در فرآیندها به کار می‌برند و ممکن است برای هر یک از این روش‌ها از ابزار خاصی استفاده کنند. افزون بر این، هر پروژه‌ی نرم‌افزاری راهکاری برای تضمین کیفیت یک نرم‌افزار دارد زیرا هدف اصلی مهندسی نرم‌افزار تولید نرم‌افزار با کیفیت بالا می‌باشد. متدولوژی در واقع نحوه ارتباط این چهار لایه را با یکدیگر مشخص می‌کند. در فصل دهم به تفصیل در مورد متدولوژی‌های مختلف تولید نرم‌افزار صحبت خواهیم کرد.

نکته ۶: در واقع می‌توان گفت متدولوژی است که مشخص می‌کند از چه مدل فرآیندی با چه روش‌هایی و با استفاده از چه ابزاری و تحت چه معیارها و استراتژی‌های کیفی استفاده کنیم تا به نرم‌افزار مورد نظر برسیم.

چارچوب فرآیند

چارچوب فرآیند پایه‌ای را برای فرآیند یک نرم‌افزار بنا می‌کند که باعث تشخیص تعدادی از فعالیت‌های چارچوبی می‌شود که در تمام پروژه‌های نرم‌افزاری قابل استفاده باشند بدون این که لازم باشد اندازه و پیچیدگی آن پروژه‌ها را در نظر بگیریم. به علاوه، چارچوب فرآیند شامل تعدادی فعالیت‌های چتری است که در طول مدت زمان اجرای فرآیند نرم‌افزار به کار می‌روند.

فعالیت‌های چارچوبی فرآیند

پنج فعالیت عمومی برای چارچوب فرآیند وجود دارند که در تعداد زیادی از پروژه‌های نرم‌افزار قابل استفاده می‌باشند. این فعالیت‌های چارچوبی عبارتند از:

۱ ـ ارتباطات: در این فعالیت پایه‌ی ارتباطات و همکاری با مشتری (و سایر ذی‌نفعان) کلید می‌خورد و جمع‌آوری نیازمندی‌ها و سایر فعالیت‌های مورد نیاز برای این جمع‌آوری‌ها در این مرحله انجام می‌شود. در واقع مهم‌ترین فعالیتی که در این مرحله انجام می‌گیرد مهندسی سیستم و مهندسی نیازمندی‌ها می‌باشد.

نکته ۷: این فعالیت چارچوبی بر خلاف سایر فعالیت‌های چارچوبی تا به انتهای توسعه‌ی نرم‌افزار ادامه دارد و ذی‌نفع باید دائماً با تیم پروژه در ارتباط باشد.

۲ ـ برنامه‌ریزی: هر کار مهندسی بدون برنامه‌ریزی قطعاً به شکست می‌انجامد. مهندسی نرم‌افزار نیز مانند هر فعالیت مهندسی دیگری نیازمند برنامه‌ریزی است. فعالیت برنامه‌ریزی، برنامه‌ای را برای فعالیت‌های مختلف بخش‌های مختلف مهندسی نرم‌افزار پایه‌ریزی می‌کند. این فعالیت، وظیفه‌های فنی که باید هدایت شوند، ریسک‌هایی که محتمل می‌شوند، منابع موردنیاز، واحدهای کاری که باید ایجاد شوند، و برنامه‌ی زمان‌بندی برای کارها را تشریح می‌کند.

۳ ـ مدل‌سازی: نیازمندی‌هایی که جمع‌آوری شده‌اند باید به زبانی مدل شوند که برای توسعه‌دهندگان و مشتری قابل فهم باشد. این فعالیت، شامل ایجاد مدل‌هایی می‌شود که به توسعه‌دهنده و مشتری کمک می‌کند تا نیازمندی‌های نرم‌افزار را بهتر درک کرده و از روی آن‌ها فعالیت طراحی را انجام دهند تا بدین طریق بتوانند به سمت پیاده‌سازی آن نیازمندی‌ها بروند. این فعالیت شامل دو عمل اصلی تحلیل و طراحی می‌شود. تحلیل شامل مجموعه‌ای از وظایف کاری می‌شود (مثل جمع‌آوری نیازمندی‌ها، استخراج، مذاکره، مشخصه‌سازی و اعتبار سنجی) که ما را به سمت ساختن مدل تحلیل پیش می‌برند. طراحی نیز وظایف کاری ر ا در بر می‌گیرد (مثل طراحی داده، طراحی معماری، طراحی واسط، و طراحی سطح مؤلفه) که مدل طراحی را می‌سازند. بعد از ایجاد مدل طراحی می‌توانیم به سمت پیاده‌سازی و تولید آن برویم.

۴ـ تولید: پس از تحلیل نیازمندی‌های نرم‌افزار نوبت به طراحی معماری نرم‌افزار، واسط، کلاس‌ها و داده‌های نرم‌افزار و نیز مؤلفه‌های مختلف آن می‌رسد و پس از طراحی مرحله تولید کد شروع می‌شود. در این فعالیت، تولید کد (خواه به صورت دستی و خواه به صورت خودکار) و آزمایش‌هایی که برای تصحیح خطاهای آن و نیز اعتبارسنجی آن مورد نیاز است، انجام می‌شود.

۵ـ استقرار: در این فاز، نرم‌افزار (به عنوان یک موجودیت کامل یا به عنوان یک موجودیت ناکامل که به طور افزایشی کامل می‌شود) به مشتری تحویل داده می‌شود و مشتری با بررسی محصول دریافتی، بازخوردهای به دست آمده بر اساس همین ارزیابی‌ها را به تیم نرم‌افزاری ارائه می‌دهد.

بعد از این که نرم‌افزار به مشتری تحویل داده شد مرحله نگهداری آغاز می‌شود که در این فاز تاکید بر روی **نگهداری نرم‌افزار** می‌باشد. به طور کلی چهار نوع نگهداری داریم:

۱ـ نگهداری اصلاحی(Corrective) : در این نوع نگهداری اشکالاتی که در چرخه عمر نرم‌افزار به وجود آمده‌اند، شناسایی، بررسی و رفع می‌گردند.

۲ـ نگهداری تطبیقی(Adaptive) : در این نوع نگهداری، با توجه به بروز نیازمندی‌های جدید تغییراتی در نرم‌افزار برای تطبیق دادن آن با این نیازمندی‌ها ایجاد می‌شود.

۳ـ نگهداری بهبودی (Preeffective): در این نوع نگهداری قابلیت‌ها و خصوصیت‌های جدید برای بهبود کارآیی نرم‌افزار به آن اضافه می‌شود.

۴ـ نگهداری پیش‌گیرانه (Preventive): نرم‌افزار کامپیوتر در اثر تغییرات زیاد، کارآیی خود را از دست می‌دهد، از این رو مهندسی مجدد نرم‌افزار باید اجرا شود تا نرم‌افزار بتواند نیازهای کاربران نهایی خود را برآورده کند. به طور خلاصه نرم‌افزار بعد از دوره مشخصی بازنگری شده و اصلاحات لازم بر روی آن انجام می‌شود و در واقع نرم‌افزار جدیدی تولید می‌گردد. نگهداری متناظر با این نوع تغییر، نگهداری پیش‌گیرانه نامیده می‌شود.

مثال ۷: کدام‌یک از گزینه‌های زیر درست می‌باشد؟

۱) تحلیل و طراحی نرم‌افزار در فاز مدل‌سازی انجام می‌گیرد.
۲) برآورد نرم‌افزار در فاز ارتباطات صورت می‌گیرد.
۳) آزمایش نرم‌افزار در فاز استقرار صورت می‌گیرد.
۴) مورد ۱ و ۳

☑ **پاسخ:** گزینه «۱»

مثال۸: کدام دسته از نگهداری معمولا هزینه و زمان بیشتری را نسبت به سایر نگهداری‌ها دربر خواهد داشت؟

۱) نگهداری اصلاحی ۲) نگهداری تطبیقی ۳) نگهداری پیش‌گیرانه ۴) نگهداری بهبودی

☑ **پاسخ:** گزینه «۳» در این نوع نگهداری به دلیل این که حجم تغییرات اعمال شده بر روی نرم‌افزار بیش از سایر نگهداری‌ها می‌باشد، هزینه و زمان بیشتری در آن صرف خواهد شد.

فعالیت‌های چتری نرم‌افزار

فعالیت‌های چتری، فعالیت‌هایی هستند که در تمام مدت زمان چرخه عمر نرم‌افزار می‌توانند انجام شوند و به صورت چتری فعالیت‌های چارچوبی را پوشش می‌دهند. این فعالیت‌ها عبارتند از:

۱ـ کنترل و بازرسی پروژه نرم‌افزار: پیشرفت پروژه در برابر برنامه‌ریزی از قبل انجام شده برای پروژه، در این فعالیت مقایسه می‌شود تا در صورت لزوم و در مواردی که از زمان‌بندی عقب هستیم کارهایی برای جبران این عقب‌ماندگی انجام پذیرد.

۲ـ مدیریت ریسک: در این فعالیت ریسک‌های محتملی که ممکن است بر روی خروجی‌های پروژه و یا کیفیت محصول نهایی پروژه تأثیر ناگواری بگذارند شناسایی، مدیریت و در صورت امکان تقلیل می‌یابند.

۳ـ تضمین کیفیت نرم‌افزار: در این فعالیت، کارهایی را که برای تضمین کیفیت نرم‌افزار مورد نیاز می‌باشند تعریف و هدایت می‌کند. وظیفه اصلی این فعالیت همان‌طور که از نامش پیداست، تضمین کیفیت نرم‌افزار مطابق با کیفیت موردنظر مشتری می‌باشد.

۴ـ بازبینی رسمی فنی: تمامی محصولات کاری تولید شده در این فعالیت مهندسی نرم‌افزار ارزیابی می‌شوند تا در نتیجه آن خطاهای محتمل این محصولات کاری قبل از گسترش یافتن آن‌ها به مرحله یا عمل بعدی شناسایی و رفع شوند.

۵ـ اندازه‌گیری. در این فعالیت، اندازه‌ها، معیارها و شاخص‌های محصول، پروژه و فرآیند، تعریف و جمع‌آوری می‌شوند که با استفاده از این اطلاعات، مدیر و تیم پروژه قادر به تحویل نرم‌افزار با کیفیت بالا و مطابق با استانداردهای از پیش تعیین شده می‌باشند. در واقع این اطلاعات سبب مدیریت بهتر پروژه‌های نرم‌افزاری و تطابق دادن آن با استانداردها و به دست آمدن محصولی با کیفیت بالاتر می‌شود.

۶ـ مدیریت پیکربندی نرم‌افزار: تأثیرات اعمال تغییرات در طول تمام فرآیند نرم‌افزار مدیریت می‌شود.

۷ـ مدیریت قابلیت استفاده مجدد: ضوابطی برای محصول کاری که قابلیت استفاده مجدد دارد (که شامل تمام مؤلفه‌ها می‌شود) تعریف شده و نیز مکانیزمی برای به دست آوردن مؤلفه‌هایی با قابلیت استفاده مجدد پایه‌ریزی می‌شود.

۸ـ تولید و پیش تولید محصولات کاری: شامل فعالیت‌های مورد نیاز برای ساخت محصولات کاری از قبیل مدل‌ها، مستندات، فرم‌ها و لیست‌ها می‌شود.

مثال۹: کدام‌یک از فعالیت‌های زیر جزو فعالیت‌های چتری نرم‌افزار نمی‌باشد؟

۱) بازبینی رسمی فنی ـ مدیریت قابلیت استفاده مجدد ۲) مدیریت تغییرات ـ مدیریت ریسک

۳) مدیریت پیکربندی نرم‌افزار ـ کنترل و بازرسی پروژه‌ی نرم‌افزاری ۴) اندازه‌گیری ـ تضمین کیفیت نرم‌افزار

☑ **پاسخ:** گزینه «۲» مدیریت تغییرات جزو فعالیت‌های چتری نمی‌باشد.

مدل‌های فرآیندی

فرآیند نرم‌افزار مجموعه‌ای از فعالیت‌ها است که هدف آن توسعه یا تکمیل نرم‌افزار می‌باشد. در نتیجه بر اساس نوع نرم‌افزاری که قرار است توسعه داده شود و نیز محیط پروژه و کاربرد آن، مدل فرآیندی خاصی را برای رسیدن به هدف مطلوب باید انتخاب کنیم.

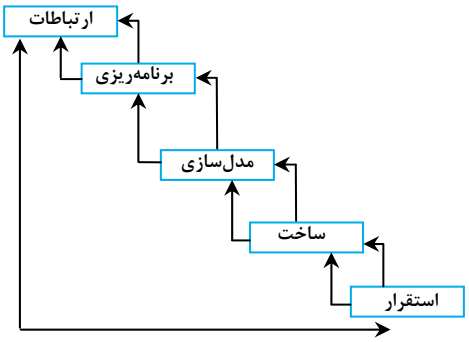
غالباً مدل‌های گوناگون فرآیندی، فازهای مختلفی را برای توسعه نرم‌افزار معرفی می‌کنند اما در حالت کلی، تمامی این فازها قابل انطباق با فعالیت‌های چارچوبی نرم‌افزار می‌باشند. در واقع تفاوت این مدل‌های فرآیندی در **میزان تأکید آن‌ها** بر بخش‌های متفاوت فازهای مختلف فعالیت‌های چارچوبی می‌باشد.

نکته۸: به طور خلاصه می‌توان گفت تفاوت مدل‌های فرآیندی با یکدیگر، در ترتیب اجرا و زمان‌بندی فعالیت‌های چارچوبی در حین فرآیند و نیز میزان تأکید هر کدام از این مدل‌ها بر روی هر کدام از فعالیت‌های چارچوبی می‌باشد.

۱ـ مدل آبشاری (Waterfall model): مدل آبشاری مدل سنتی و متداول است که به آن چرخه حیات (Life Cycle) هم می‌گویند. علت نام‌گذاری این مدل فرآیندی این است که می‌تواند به صورت گرافیکی در یک مدل آبشاری، فعالیت‌های بخش‌های مختلف آن نمایش داده شود. در این مدل، تولید نرم‌افزار مراحل مختلفی دارد که هر مرحله دارای ورودی، وظیفه و خروجی خاص خود می‌باشد. خروجی هر مرحله در این مدل، ورودی مرحله بعدی است.

این مدل با مرحله ارتباطات شروع می‌شود که در آن مراحل اولیه پروژه انجام می‌گیرد و سپس نیازمندی‌های پروژه جمع‌آوری می‌شود. مرحله دوم این مدل که برنامه‌ریزی نام دارد، پروژه برآورد و زمان‌بندی می‌شود. در مرحله سوم که به مدل‌سازی مشهور است، تحلیل و طراحی نرم‌افزار صورت می‌گیرد. مرحله چهارم، یعنی که مرحله ساخت، تولید کد و آزمایش‌های مربوط به صحت و اعتبارسنجی نرم‌افزار را بر عهده دارد و مرحله پایانی که استقرار نام دارد تحویل نرم‌افزار و پشتیبانی از آن را انجام می‌دهد.

خصوصیت اصلی این مدل این است که **هیچ گونه بازخوردی بین مراحل این مدل وجود ندارد.** مانند آب که نمی‌تواند در آبشار به عقب برگردد، در این مدل نیز بعد از ورود به یک فاز به فازهای قبلی نمی‌توان برگشت. این مدل زمانی می‌تواند کارآیی داشته باشد که نیازمندی‌ها قبل از طراحی به طور کامل و به خوبی استخراج شده باشند:



شکل ۳. **شکل مدل آبشاری**

در این مدل باید طراحی قبل از ساخت و تولید کد قبل از آزمایش به طور کامل و بدون نقص انجام گیرند. در واقع کدهای تولید شده در این مدل نباید هیچ گونه خطایی داشته باشند و از آن مهم‌تر این که **نیازمندی‌ها نیز در این مدل فرآیندی نباید تغییر کنند.** بدیهی است که در پروژه‌های بزرگ که مدت زمان انجام پروژه طولانی است و نیازمندی‌ها به صورت دائم در حال تغییر هستند، استفاده از این مدل چندان مناسب نیست. اما در پروژه‌های روتین که نیازمندی‌های آن به خوبی مشخص می‌باشند (مثل یک سیستم حسابداری) استفاده از این مدل به دلیل سهولت آن می‌تواند سودمند باشد.

استفاده از این مدل معایبی نیز دارد که عبارتند از:

۱ـ ماهیت پروژه‌های نرم‌افزاری بدین صورت است که به ندرت یک مرحله را به صورت کامل تمام کرده و وارد مرحله بعدی می‌شویم. بلکه همیشه برگشت به عقب وجود دارد که این امر در این مدل فرآیندی امکان‌پذیر نمی‌باشد.

۲ـ اساس این پروژه بر این بنا شده است که در همان ابتدای کار نیازها به خوبی از مشتری استخراج می‌شود که در بسیاری از پروژه‌ها این امر فرض غلطی است. بنابراین در پروژه‌هایی که به خصوص عدم قطعیت در آن‌ها بسیار زیاد است و نیازمندی‌ها دائما در حال تغییر می‌باشند، این مدل به هیچ وجه کارا نیست.

۳ـ در این مدل فرآیندی مشتری باید صبر زیادی داشته باشد. زیرا نسخه‌ی عملیاتی را دیر می‌بیند و بر این اساس باید تا پایان کار پروژه صبور باشد، در این بین، ایجاد یک عیب در برنامه تولید شده و یا تولید چیزی غیر از خواسته‌ی مشتری ممکن است فاجعه‌انگیز باشد.

مثال ۱۰: کدام یک از گزاره‌های زیر در ارتباط با مدل فرآیندی آبشاری نادرست می‌باشد؟

۱) در این مدل امکان تغییر نیازمندی‌ها پس از مرحله‌ی تحلیل وجود ندارد.

۲) این مدل فرآیندی برای پروژه‌های روتین که نیازمندی‌های آن مشخص می‌باشد بسیار مناسب است.

۳) در پروژه‌های شیءگرا می‌توان از این مدل استفاده کرد.

۴) مزیت اصلی استفاده از این مدل فرآیندی ساده بودن آن می‌باشد.

☑ **پاسخ:** گزینه «۳» در پروژه‌های شیءگرا به دلیل بروز تغییر فراوان در مراحل مختلف آن نمی‌توان از مدل آبشاری استفاده کرد. از این مدل بیشتر در متدولوژی ساخت یافته استفاده می‌شود.

۲ـ مدل‌های فرآیندی با ماهیت افزایشی

حالت‌های بسیاری وجود دارد که در آن نیازمندی‌های نرم‌افزار به طور معقول و مناسبی تعریف شده‌اند ولی قلمرو و اندازه‌ی کار مورد نیاز برای توسعه‌ی نرم‌افزار مانع توسعه‌ی آن به صورت خطی می‌شود. در واقع حجم کار این قدر زیاد است که نمی‌توان همه‌ی آن را یکباره انجام داد. به علاوه، یک سری نیازمندی‌ها بعد از کار کردن کاربر با نرم‌افزار مشخص می‌شوند که در ابتدای کار به طور کامل مشخص نیستند. در واقع این نیازمندی‌ها بعد از ارائه‌ی نسخه‌های بعدی نرم‌افزار می‌توانند گسترش یافته و کامل‌تر شوند. در این موارد، از مدل‌های فرآیندی با ماهیت افزایشی استفاده می‌شود تا بهتر بتوانیم پاسخگوی نیازمندی‌های کاربر باشیم.

مدل‌های افزایشی ـ مدل افزایشی (Incremental model)

آنچه که در این مدل فرآیندی بیش از همه به چشم می‌آید این است که به ساختن نرم‌افزار به چشم یک مدل گسسته از هم نگاه نمی‌کند. بلکه دیدی پیوسته به فرآیند ساخت نرم‌افزار دارد و به طور افزایشی در هر یک از افزایش‌های خود نسخه‌ی کامل‌تری از نرم‌افزار را ارائه می‌کند.