بنام خدا

مهندسی نرم افزار 1

جلسه دوم

مدرس: فردین شاپوری

ضوابط ارزیابی نرم افزار

- هدف: تولید برنامه با کیفیت خوب
- عوامل موثر در خوب بودن نرم افزار:
  - عوامل خارجی: توسط کاربر نرم افزار تشخیص داده می شود. (اهداف)
  - عوامل داخلی: برای متخصصین کامپیوتر قابل درک است. (ابزار رسیدن به هدف)

عوامل خارجی:

1- صحت برنامه (correctness):

برنامه دقیقا وظایف خود را بدرستی انجام دهد و مطابق نیازهای تعیین شده از سوی متقاضی باشد و تمامی نیازهای وی را جوابگو باشد.

2- استحکام (Robustness):

توانایی برنامه در جوابگویی به کاربر در شرایط غیر عادی. یک برنامه مستحکم می تواند در شرایط خارج از تعیین مشخصات و خصوصیات سیستم به نحوی مطلوب جوابگو باشد.

عوامل خارجی (ادامه):

3- قابلیت توسعه (Extendibility):

این عامل بمنظور تغییر در تعیین مشخصات مورد توجه قرار می گیرد. قابلیت توسعه در برنامه هایی که مدلسازی آنها به محیط واقعی مساله نزدیکتر است و فاصله معنایی کمتری دارد بیشتر وجود دارد.

4- قابلیت استفاده مجدد (Reusability):

توانایی استفاده مجدد از قسمتهایی از برنامه در تولید برنامه های جدیدتر است. این امر بیشتر در رابطه با عدم تمرکز در طراحی مشاهده می شود. (منظور از عدم تمرکز وجود واحدبندی و یکپارچه نمودن برنامه است)

5- قابلیت حمل (Portability):

قابل اجرا بر روی سیستم عامل ها و سخت افزارهای گوناگون

عوامل خارجی (ادامه):

6- سازگاری (Compatibility):

جهت ترکیب و وجود رابطه بین محصولات نرم افزاری است مانند:

- استاندارد درقالب فایلها
- استاندارد در ساختمان داده ها
- استاندارد در واسطهای کاربری

7- کارایی (Efficiency):

سرعت بالا و مصرف حافظه پایین

عوامل داخلی:

مهمترین عامل <u>واحدبندی</u>

الگوها یا مدلهای فرایند موجود برای مهندسی نرم افزار

جهت برطرف کردن بحران نرم افزاری، الگو ها یا مدلهای فرایند مناسبی برای تولید نرم افزار ارایه گردیده است.

سه عامل کلیدی وجود دارند که یک مدیر بتواند، یک فرایند تولید نرم افزار را کنترل کند و برای مجری یک پایه و اساس جهت ساختن یک نرم افزار با کیفیت بالا ایجاد کند:

1- روشهای مهندسی نرم افزار (methods): "چگونگی" انجام کار را از نظر فنی برای ساخت نرم افزار مهیا می کنند. روشها طیف وسیعی از مشاغل زیر را در بر می گیرد: زمانبندی و تخمین پروژه – تجزیه و تحلیل نیازمندیهای سیستم و نرم افزار – طراحی ساختمان داده ها – معماری سیستم نرم افزاری – نوشتن روالها – تست کردن و نگهداری

2- ابزار مهندسی نرم افزار (tools):

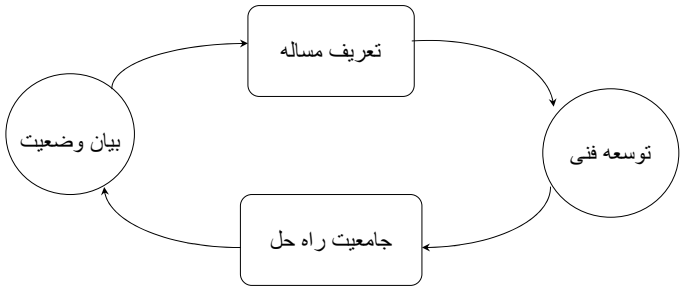
ابزارهای مهندسی نرم افزار، پشتیبانی تمام خودکار یا نیمه خودکار از روشها و فرایند را فراهم می کنند مانند CASE ها (Computer Aided Software Engineering)

3- فرایند (process):

مانند چسبی است که روشها و ابزار را به هم پیوند می زند. فرایند، ترتیبی که روشهای فنی بایستی بکار برده شوند را تعریف می کند. مستندات، گزارشها، مدلها، داده ها، فرمها و ... را که باید تولید شوند، مشخص می کند و محک پیشرفت پروژه را برای مدیریت مشخص می کند.

بررسی الگوها یا مدلهای فرایند تولید نرم افزار

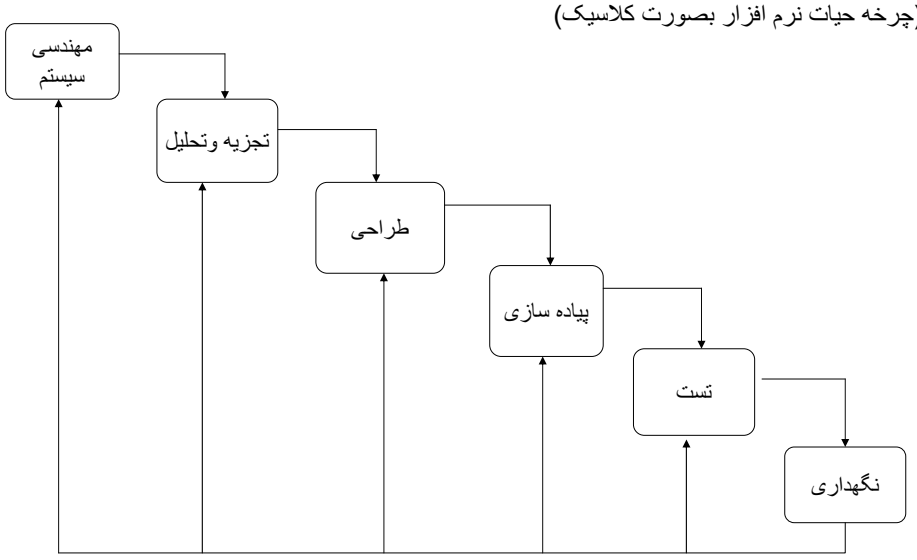
- مدل فرایند نرم افزار، یک بازنمایی (نمایش) انتزاعی (abstract) از فرایند است. شرح فرایند را از تعدادی دیدگاه خاص نمایش می دهد.
- یک مدل فرایند برای مهندسی نرم افزار بر اساس ماهیت و طبیعت پروژه و کاربرد آن، روشها و ابزار مورد استفاده و کنترل ها و مستنداتی که لازم است ارایه شود، انتخاب می شود.
- تمام کارهای توسعه و ارایه یک نرم افزار را می توان یک حلقه حل مشکل طبق شکل زیر توصیف کرد:



- چهار مرحله در شکل فوق:
  - وضع موجود: نمایانگر وضعیت موجود امور است.
  - تعریف مساله: لیست نیازها و خواسته ها و مشکلات و کاستی ها (مشکل یا مشکلاتی که باید حل شود)
  - توسعه فنی: مشکل را از طریق بکارگیری فنآوری حل می کند.
  - یکپارچه سازی راه حل: راه حل هایی که در طی فرایند توسعه بدست می آیند، یکپارچه کرده و نتایج حاصل (مثل اسناد، برنامه ها، داده ها، عملکرد کسب و کار جدید، محصول جدید) را به کسانی که راه حل را در مرحله اول تقاضا کرده اند، ارایه می دهد.
- در اینجا چندین الگوی موجود تولید و توسعه نرم افزار (مدل فرایند) را بررسی می کنیم:

1- مدل آبشاری (خطی-ترتیبی)

(چرخه حیات نرم افزار بصورت کلاسیک)



- مهندسی سیستم و تجزیه و تحلیل آن:

از آنجایی که نرم افزار همیشه قسمتی از یک سیستم بزرگتر است، کار از مشخص کردن نیازمندیهای کل سیستم آغاز می شود و سپس زیر مجموعه ای از این نیازمندیها را به نرم افزار نسبت می دهیم.

از آنجایی که نرم افزار ما مجبور به داشتن ارتباط با مولفه های دیگر سیستم از قبیل سخت افزار، مردم و گاه پایگاه داده هاست، داشتن این دیدگاه از سیستم یک امر اساسی است.

(این مرحله شامل جمع آوری نیازها در سطح سیستم و کمی هم تجزیه و تحلیل و طراحی در سطح بالاست)

- تجزیه و تحلیل نیازمندیهای نرم افزار:

جمع آوری نیازمندیها مشخصا مربوط به نرم افزار.

برای درک ماهیت برنامه هایی که باید ساخته شود، مهندس نرم افزار (تحلیلگر) بایستی موارد زیر را استخراج کند:

- دامنه اطلاعات
- عملیات مورد نظر (ورودیها، خروجیها و پردازش ها)
- واسط ها

نیازمندیهای سیستم و نرم افزار مستندسازی شده و با مشتری بازنگری می شود.

- طراحی:

طراحی نرم افزار یک فرایند چند مرحله ای است که روی سه مشخصه متفاوت از برنامه تاکید دارد:

- ساختمان داده ها
- معماری نرم افزار
- جزئیات رویه ها

در فرایند طراحی، نیازمندیها تبدیل به نمایشی از نرم افزار می شوند، تا قبل از به کد در آوردن قابل ارزیابی باشند.

طراحی نیز بایستی مانند نیازمندیها مستندسازی گردد.

- پیاده سازی:

طراحی بایستی به صورتی که برای ماشین قابل فهم باشد در بیاید. اگر طراحی در حد جزئیات باشد، پیاده سازی می تواند بسیار سریع و بصورت مکانیکی انجام شود.

- تست (آزمون):

پس از پیاده سازی تست برنامه آغاز می شود.

تست تاکید دارند بر:

- روال منطقی داخل نرم افزار: اطمینان از اینکه تمامی دستورالعملها تست شده اند.
- عملیات خارجی برنامه: اطمینان از اینکه یک ورودی تعریف شده، نتایج مورد نظر را ایجاد می کند.

- نگهداری:

نرم افزار بدون شک پس از تحویل به مشتری دچار تغییر می شود. (به استثناء نرم افزارهای توکار)

تغییرات به علت:

- خطاها
- نرم افزار بایستی با تغییرات در محیط بیرونی تطبیق پیدا کند.
(مثل یک OS جدید یا دستگاه جانبی جدید، قواعد و تغییر قوانین سازمانی)
- مشتری خواستار توسعه در عملیات یا بالا بردن کارایی است.

* نگهداری نرم افزار تمام مراحل قبل را روی یک برنامه موجود اجرا می کند.

- این الگو یا مدل فرایند، قدیمی ترین الگوی مهندسی نرم افزار می باشد.

معایب این روش:

- پروژه های واقعی به ندرت ماهیت ترتیبی دارند.
- اغلب در شروع کار برای مشتری بیان صریح همه نیازمندیها مشکل است. با توجه با اینکه مدل فرایند آبشاری بصورت خطی-ترتیبی است، ممکن است جهت تشخیص دقیق تمام نیازمندها، ناچارا برگشت به عقب داشته باشیم و این زمان و هزینه تولید و توسعه را افزایش می دهد.
- مشتری باید صبور باشد. نسخه عملیاتی نرم افزار، غالبا تا اواخر پروژه در دسترس قرار نمی گیرد. یک اشتباه بزرگ که تا اجرای برنامه مشخص نشده باشد می تواند تولید فاجعه کند.
- خطای شناسایی شده در مراحل آخر، ممکن است به مراحل اول مرتبط باشد و این بازگشت به عقب، هزینه ها را بسیار بالا می برد.
- با توجه با اینکه مراحل بصورت خطی دنبال می شود، این امکان پیش می آید که برخی از اعضای تیم باید منتظر باشند تا دیگر اعضا کارشان تکمیل شود.

* البته این مدل با قابلیت انعطاف بیشتر در الگوهای دیگر بکار می رود و از یک برخورد غیر سیستماتیک خیلی بهتر است.

محاسن این روش:

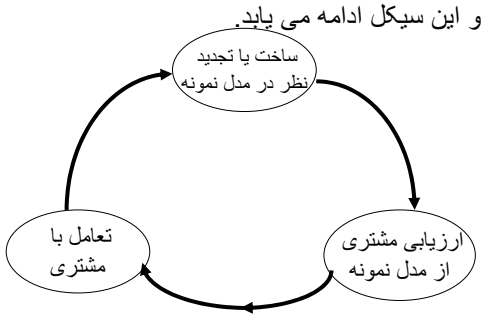
- وقتی از ابتدا، نیازها مشخص باشد، با توجه به طبیعت این مدل (یعنی خطی-ترتیبی) انتظار می رود با بکارگیری این الگو ، یک نسخه کاری یا عملیاتی تحویل مشتری داده شود. بنابراین وقتی در طی مراحل تولید، نیازها در حال تغییر می باشد، بهتر است از این مدل استفاده نشود.

2- مدل نمونه سازی

- معمولا مشتری یکسری اهداف کلی را برای نرم افزار تعریف می کند. اما جزئیات پردازش ورودی و یا خروجی های مورد نیاز را مشخص نمی کند. به تعبیری نیازمندیهای مشتری مشخص و واضح نیست.
- در موارد دیگر تولید کننده از کارایی یک الگوریتم یا مناسب بودن سیستم عامل یا نحوه تعامل انسان با سیستم (واسط کاربر) مطمئن نیست.
- در این موارد روش نمونه سازی به کار می آید.
- نمونه سازی فرایندی است که تولید کننده را قادر به ایجاد مدل از نرم افزار مورد نظر می کند.
- یکسری فرمها، پنجره ها و صفحات تولید می شود که شکل و شمایل نرم افزار را نمایش می دهد. این نمونه می تواند روی کاغذ باشد یا روی کامپیوتر و از طریق محیط های ویژوال در مدت زمان کوتاهی تولید شود.

2- مدل نمونه سازی (ادامه)

- مدل نمونه سازی طبق شکل زیر ابتدا با جمع آوری نیازمندیها شروع می شود. مشتری و تولید کننده همدیگر را ملاقات نموده و اهداف کلی خود را در مورد نرم افزار بیان می نمایند، هرچه که مورد نیازاست شناسایی می کنند و حوزه هایی را مطرح می کنند که در آن تعریف بیشتر ضروری است. سپس طراحی سریع رخ می دهد. این طراحی روی نمایش آن جنبه هایی از نرم افزار متمرکز می شود که برای مشتری مشهودند (مثل فرم های ورودی و خروجی). این طراحی سریع منجر به ساخت یک نمونه می شود . نمونه توسط مشتری ارزیابی شده و دوباره نظرات مشتری در طی تعامل دریافت شده و این سیکل ادامه می یابد.



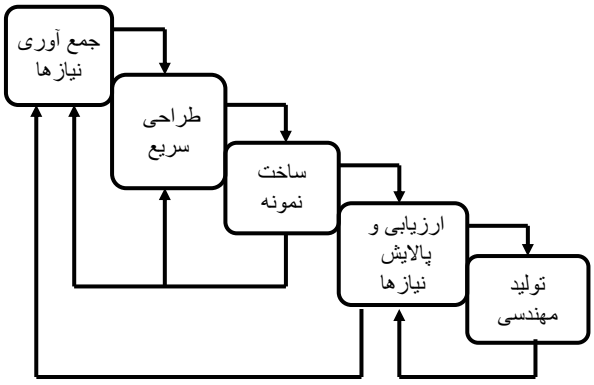
2- مدل نمونه سازی (ادامه)

معایب این روش:

- مشتری یک نسخه کاری از نرم افزار را می بیند و نمی داند که این نمونه بدون در نظر گرفتن کیفیت کلی نرم افزار و یا قابلیت نگهداری دراز مدت ساخته شده. وقتی به او بگوئیم که این محصول بایستی دوباره ساخته شود مشتری ناراحت می شود.
- تولید کننده اغلب برای اینکه نمونه را سریع به پایان برساند به خیلی از مسایل توجه نمی کند. یک سیستم عامل نامناسب یا زبان برنامه نویسی نامناسب، چون در دسترس و شناخته شده هستند بکار می روند. یک الگوریتم ناکارامد فقط برای نمایش قابلیت انجام یک کار بکار می رود. بعد از مدتی تولید کننده با این انتخابها آشنا شده و دلایل نامناسب بودن آنها را فراموش می کند.
- توقف این سیکل مشکل است. مشتری دوست دارد دائم یکسری ویژگیها اضافه شود.
- مشتری فکر می کند تولید نرم افزار به سادگی همین نمونه (ماکت) می باشد. بنابراین در زمان تحویل نرم افزار و هزینه انجام پروژه، ممکن است با مشتری با مشکل مواجه شویم.
- ممکن است مشتری وقت لازم را برای تعامل و ارزیابی نگذارد یا مشتری بسادگی در دسترس نباشد.

- بنابراین با اینکه مدل فرایند نمونه سازی یک الگوی موثر در مهندسی نرم افزاراست بایستی مسایلی را در نظر گرفت. از ابتدا قوانینی را تعریف کرد. مشتری و تولید کننده بایستی هر دو موافقت کنند که این نمونه به عنوان مکانیزمی برای تعریف نیازها تعریف شده است. پس از آن بایستی تمام آن یا قسمتی از آن دور انداخته شود و نرم افزار مورد نظر با در نظر گرفتن کیفیت و قابلیت نگهداری تولید شود.

- از این مدل زمانی مناسب است استفاده کنیم که نیازهای مشتری واضح و مشخص نیست و ابهام در خواسته های کاربر وجود دارد. همچنین اگر می خواهیم واسط کاربر مناسبی داشته باشیم یا کارایی یک الگوریتمی و ... را بررسی کنیم، نمونه سازی می تواند استفاده شود.

2- مدل نمونه سازی (ادامه)

وقتی نمونه نهایی ساخته شد و اهداف لازمه مثل تعیین لیست خواسته ها و نیازهای مشتری / کاربران حاصل گردید، می توان از یک مدل فرایند دیگر استفاده کرد و نرم افزار را بصورت مهندسی تولید کرد.