دانشگاه پیام نور مشهد

جزوه درس مهندسی نرم افزار ۱

مصطفی جهانگیر

www.mjahangir.ir

mjahangirf@gmail.com

به نام آن که جان را فکرت آموخت

جزوه درس

مهندسی نرم افزار ۱

مصطفی جهانگیر

mjahangirf@gmail.com

www.mjahangir.ir

خرداد ۱۳۹۳

فهرست مطالب

جلسه ۲– نرم افزار و مهندسی نرم افزار

۱. ماهیت نرم افزار

۱,۱. ماهیت و تعریف نرم افزار

امروزه نرم افزار نقشی دو گانه دارد:

- نرم افزار نوعی محصول است.
- در عین حال وسیله نقلیه ای برای تحویل یک محصول است.

نرم افزار عبارت است از:

- دستورالعمل ها که هنگام اجرا، ویژگی، عملکرد و کارایی مطلوب را فراهم می سازند.
- ساختمان های دادهایی که برنامه ها را قادر به پردازش مناسب داده ها کنند.
- اطلاعات توصیفی در هر دو قالب کپی سخت و مجازی که راه اندازی و استفاده از برنامه ها را شرح دهند.

۱,۲. خصوصیات نرم افزار

نرم افزار بیشتر یک عنصر منطقی است تا یک عنصر سیستمی فیزیکی. بنابراین نرم افزار با سخت افزار تفاوت چشمگیری دارد:

- نرم افزار، مهندسی و بسط داده می شود و چیزی نیست که به معنای کلاسیک کلمه ساخته شود.
- نرم افزار فرسوده نمی شود.
- گرچه صنعت در حال حرکت به سوی مونتاژ قطعات است، اکثر نرم افزارها همچنان به صورت سفارشی ساخته می شوند.

۱,۳. نرم افزار، مهندسی و بسط داده می شود

تفاوت های بسط سخت افزار و نرم افزار:

- فاز ساخت برای سخت افزار باعث بروز مشکلات کیفیتی می شود که برای نرم افزار وجود ندارند یا به راحتی قابل رفع است.
- هر دو عمل وابسته به انسان هستند ولی رابطه ی میان انسان و کاری که انجام می شود، کاملا متفاوت است.
- هر دو عمل مستلزم ساخت یک محصول هستند ولی روش ها متفاوت است.
- نرم افزارها ساخته نمی شوند، بلکه مهندسی می شوند.

۱,۴. نرم افزار فرسوده نمی شود

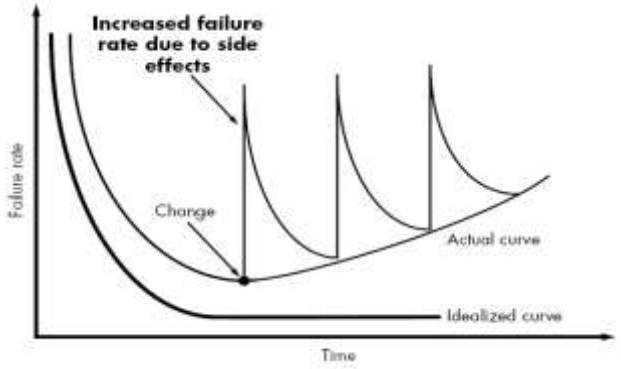
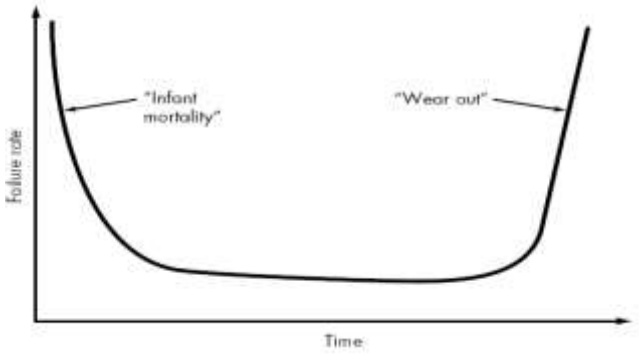
منحنی شکست سخت افزار

- منحنی وانی نمودار آهنگ شکست را به صورت تابعی از زمان برای سخت افزار نشان می دهد.
- منحنی وانی، نشان می دهد که سخت افزار در ابتدای عمر آهنگ شکست شدیدی دارد.
- این عیوب، تصحیح و آهنگ شکست در یک دوره زمانی به مقداری ثابت نزول می کند.

- با گذشت زمان سخت افزار شروع به فرسایش کرده و دوباره آهنگ شکست شدت می گیرد.

منحنی های شکست واقعی و ایده آل نرم افزار

- نرم افزار نسبت به ناملایمات محیطی که باعث فرسایش آن می شود نفوذ پذیر نیست.
- بنابراین در تئوری، منحنی شکست برای نرم افزار باید شکل منحنی ایده آل را به خود بگیرد.
- منحنی ایده آل نسبت به منحنی واقعی مدل های شکست نرم افزار، بسیار ساده تر است.
- ولی، معنای آن واضح است، نرم افزار هیچ گاه دچار فرسایش نمیشود بلکه زوال می یابد!
- نرم افزار در دوران حیات خود دستخوش تغییر می شود.
- با اعمال این تغییرات احتمال دارد که برخی عیوب جدید وارد شوند و باعث خیز منحنی آهنگ شکست می شود.
- و پیش از آن که منحنی بتواند به آهنگ شکست منظم اولیه خود برسد، تغییر دیگری درخواست می شود که باعث خیز دوباره منحنی می شود.

منحنی های شکست واقعی و ایده آل نرم افزار	منحنی شکست سخت افزار
	

۱,۵. نرم افزارها به صورت سفارشی ساخته می شوند

در جهان سخت افزار، استفاده مجدد از قطعات بخشی طبیعی از فرایند مهندسی است. در مهندسی نرم افزار این امر به تازگی مورد توجه قرار گرفته است.

۲. دامنه های کاربرد نرم افزار

۲,۱. انواع نرم افزارها

نرم افزارهای	توضیح
سیستمی	مجموعه ای از برنامه هاست که برای سرویس دهی به برنامه های دیگر نوشته شده اند.
کاربردی	برنامه های مستقلی که یک نیاز تجاری مشخص را بر طرف می کند.
مهندسی- علمی	نرم افزارهای علمی توسط الگوریتم هایی مشخص می شوند که اعداد و ارقام را پردازش می کنند. کاربردهای نوین در حیطه مهندسی و علمی از الگوریتم های عددی مرسوم فراتر رفته اند.
تعبیه شده	در حافظه فقط خواندنی جای دارند و برای کنترل محصولات و سیستم های مربوط به بازارهای صنعتی و مصرفی به کار می رود.
خط تولید	برای فراهم آوردن یک قابلیت خاص جهت استفاده توسط بسیاری از مشتریان مختلف طراحی می شوند.
کاربردی تحت وب	این گروه از نرم افزارهای شبکه ای شامل مجموعه ی گسترده ای از برنامه های کاربردی می باشد.

هوش مصنوعی	برای حل مسائل پیچیده ای که به روش های عددی قابل حل نیستند، از الگوریتم های غیر عددی استفاده می کنند. سیستم های خبره، تشخیص الگوها، شبکه های عصبی مصنوعی، اثبات قضایا و بازی مثال هایی از کاربرد این گروه هستند.

۲,۲. مسائل جدید کاربردهای نرم افزار

مسائل جدید	توضیح
کار با کامپیوتر در جهانی باز	چالشی که مهندسان فرا روی خود خواهند داشت، توسعه ی سیستم ها و برنامه های کاربردی است که برقراری ارتباط میان کامپیوترهای شخصی، دستگاه های همراه و سیستم های اداری را از طریقه شبکه های گسترده میسر می سازند.
کد منبع باز	تمایل رو به رشدی است که منجر به توزیع کدهای منبع سیستم ها و برنامه های کاربردی شده است به طوری که افراد بسیاری بتوانند در توسعه آن سهیم شوند.

۲,۳. نرم افزارهای قدیمی

- سیستم های نرم افزاری قدیمی چند دهه قبل ساخته شده اند و پیوسته اصلاح شده اند تا تغییرات به عمل آمده درخواست های تجاری و سکوهای محاسباتی را پاسخگو باشند.
- ازدیاد این گونه سیستم ها باعث دردسر برای سازمان های بزرگی می شود که نگهداری از آن ها را پر هزینه و تکامل بخشیدن به آن ها را خطرناک می دانند.

۲,۴. دلایل تکامل سیستم های قدیمی

- نرم افزار باید برای برآورده ساختن نیازهای محیط های جدید کامپیوتری یا فناوری های جدید اصلاح گردد.
- نرم افزار باید بهبود یابد تا خواسته های تجاری جدید را پیاده سازی کند.
- نرم افزار باید گسترش داده شود تا با سایر سیستم ها یا بانک اطلاعاتی جدیدتر، قابلیت همکاری را داشته باشد.
- نرم افزار باید دوباره معماری شود تا در یک محیط شبکه نیز قادر به ادامه ی حیات باشد.

۳. ماهیت برنامه های کاربردی تحت وب

۳,۱. صفات برنامه های کاربردی تحت وب

امروزه برنامه های تحت وب به ابزارهای کامپیوتری پیچیده ای تکامل یافته اند که نه تنها عملکردی مستقل را در اختیار کاربر نهایی قرار می دهند بلکه با بانک های اطلاعاتی و برنامه های کاربردی تجاری یکی شده اند. در اکثریت وسیع برنامه های تحت وب این صفت ها مشاهده می شود.

صفات	توضیح
میزان تمرکز شبکه	برنامه های تحت وب روی یک شبکه قرار دارند و باید نیازهای جامعه ای متنوع از کلاینت ها را برآورده سازند.
همروندی	ممکن است یک باره تعداد بسیاری از کاربران به برنامه های تحت وب دستیابی داشته باشند.
بار غیر قابل پیش بینی	تعداد کاربران برنامه های تحت وب از روزی به روز دیگر ده یا صد برابر شوند.

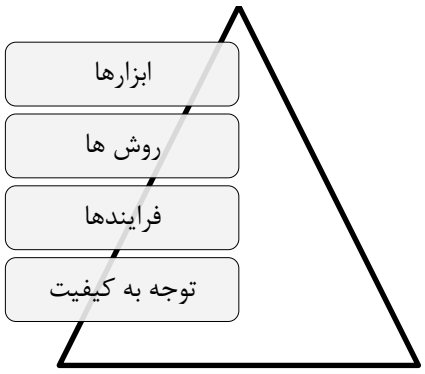
کارایی	اگر کاربر یک برنامه تحت وب باید یک مدت طولانی منتظر بماند، ممکن است تصمیم بگیرد به جای دیگری برود.
قابلیت دسترسی	گرچه انتظار ۱۰۰ در صد قابلیت دسترسی، غیر منطقی است، کاربران برنامه های تحت وب پر طرفدار غالبا تقاضای دسترسی ۲۴ ساعته در ۷ روز هفته و ۱۲ ماه سال را دارند.
داده محوری	عملکرد اصلی بسیاری از برنامه های تحت وب استفاده از ابر رسانه ها برای ارائه متون به کاربران نهایی است.
حساس به محتویات	کیفیت و ماهیت زیبا شناختی محتویات از جمله مهمترین عوامل تعیین کننده کیفیت در برنامه های تحت وب است.
تکامل پیوسته	برخلاف نرم افزارهای کاربردی سنتی، برنامه های تحت وب پیوسته در حال تکامل هستند.
بی واسطگی	نیاز اجباری برای رساندن سریع نرم افزار به بازار است.
امنیت	برای محافظت از محتویات حساس معیارهای امنیتی قوی ای باید پیاده سازی شود.

۴. مهندسی نرم افزار

۴,۱. نکات کلیدی در ساخت نرم افزار

- پیش از آن که برای مسئله راهکاری بیابید آن را درک کنید.
- یکی از فعالیت های محوری در مهندسی نرم افزار طراحی است.
- کیفیت و قابلیت نگهداری هر دو نتیجه طراحی خوب هستند.

۴,۲. لایه های مهندسی نرم افزار



۵. فرایند نرم افزار

۵,۱. فرایند نرم افزار

فرایند مجموعه ای از فعالیت ها، کنش ها و وظایف است که هنگام ایجاد یک محصول کاری اجرا می شوند.

- یک فعالیت، کوششی است در جهت رسیدن به هدفی گسترده
- یک کنش شامل مجموعه ای از وظایف است که یک محصول کاری عمده را تولید می کند.

در حیطه مهندسی نرم افزار فرایند یک روش انطباق پذیر است که تیم نرم افزار به کمک آن می توانند مجموعه ای مناسب از کنش ها و وظایف کاری را برگزینند.

۵,۲. فعالیت های چارچوبی

فعالیت	توضیح
ارتباطات	هدف ارتباطات درک اهداف طرف های ذینفع برای پروژه و جمع آوری خواسته هایی است که می توانند ویژگی ها و قابلیت های نرم افزار را تعیین کنند.
برنامه ریزی	با توصیف وظایف فنی که قرار است اجرا شوند، خطرات احتمالی، منابعی که مورد نیاز خواهد بود، محصولات کاری که باید تولید شوند و زمان بندی کاری، مهندسی نرم افزار را مشخص می کند.
مدل سازی	مهندسی نرم افزار با ایجاد مدل هایی جهت درک بهتر خواسته ها و طراحی که به این خواسته ها برسد همین کار را می کند.
ساخت	این فعالیت تولید کدها و آزمون لازم برای آشکار کردن خطاهای موجود در کدها را با هم تلفیق می کند.
استقرار	نرم افزار به مشتری تحویل داده شده که محصول تحویل داده شده را ارزیابی و براساس ارزیابی بازخوردی ارائه کند.

۵,۳. فعالیت های چتری

این فعالیت ها در سرتاسر فرایند نرم افزار رخ می دهد و کانون توجه آن ها اساسا مدیریت پروژه، پیگیری و کنترل است.

کنترل و پیگیری پروژه های نرم افزاری	مدیریت ریسک	تضمین کیفیت نرم افزار	بازبینی های فنی	اندازه گیری
	مدیریت پیکربندی نرم افزار	مدیریت قابلیت استفاده مجدد	تهیه و تولید محصول کاری	

۵,۴. تفاوت های مدل های فرایند

مدل های فرایند در ابعاد زیر با یکدیگر تفاوت دارند:

- جریان کلی فعالیت ها، کنش ها و وظایف و بستگی آن ها به یکدیگر
- درجه تعریف کنش ها و وظایف در هر فعالیت چارچوبی
- درجه شناسایی محصولات کاری و نیاز به آن ها
- شیوه اعمال فعالیت های تضمین کیفیت
- درجه کلی جزئیات به کار رفته در توصیف فرایند
- درجه دخالت مشتری و طرف های ذینفع در پروژه
- سطح استقلال داده شده به تیم نرم افزار
- درجه توصیف نقش ها و سازماندهی تیم

۵,۵. مدل فرایند تجویزی

- مدل فرایند تجویزی بر جزئیات تعریف، شناسایی و کاربرد فعالیت ها و وظایف تاکید دارد.
- هدف آن ها بهبود بخشیدن به کیفیت سیستم، بالا بردن قابلیت مدیریت پروژه، قابل پیش بینی کردن تاریخ تحویل و هزینه ها و راهنمایی تیم مهندسان نرم افزار را برای کارهای لازم است.

۵,۶. مدل فرایند چابک

- مدل فرایند چابک بر سرعت تاکید دارد و مجموعه ای از اصول را دنبال می کند که به یک روش غیر رسمی تر برای فرایند نرم افزار منجر می شود.
- این مدل بر قابلیت مانور و انطباق پذیر تاکید دارد و برای انواع بسیاری از پروژه ها به ویژه مهندسی برنامه کاربردی تحت وب مناسب است.

۶. مهندسی نرم افزار در عمل

۶,۱. اصول کلی در مهندسی نرم افزار

- اصول کلی که دیوید هوکر در مهندسی نرم افزار به عنوان یک کلیت است نام برد عبارتند از:

دلیل وجود سیستم	ساده نگه داشتن	حفظ چشم انداز	آنچه که شما تولید می کنید دیگران مصرف کنند.	آینده نگری	برنامه ریزی پیشاپیش برای استفاده مجدد	تفکر

۷. پندارهای باطل نرم افزاری

۷,۱. پندارهای باطل مدیریتی

- ما از قبل کتابی داریم که آکنده از استانداردها و روال های لازم برای ساختن نرم افزارهاست. آیا این کتاب آن چه را که افراد من باید بدانند در اختیارشان قرار نخواهد داد؟
- اگر از برنامه عقب بیفتیم، می توانیم بر تعداد برنامه نویسان بیفزاییم و عقب ماندگی را جبران کنیم (یورش مغولی)
- اگر تصمیم به برون سپاری یک پروژه نرم افزاری به شرکت دیگری بگیرم می توانم خودم را آسوده سازم و بگذارم تا آن شرکت آن را بسازد.

۷,۲. پندارهای باطل مشتریان

- بیانی کلی از اهداف، برای شروع به نوشتن برنامه ها کفایت می کند، جزئیات را بعدا می توانیم پر کنیم.
- نیازهای پروژه پیوسته در حال تغییر است، ولی این تغییرات را می توان در نرم افزار جای داد، زیرا نرم افزار انعطاف پذیر است.

۷,۳. پندارهای باطل سازندگان

- هنگامی که برنامه را نوشتیم و برنامه کار کرد، دیگر کار تمام است.
- تا هنگامی که برنامه را اجرا نکرده ام، راهی برای ارزیابی کیفیت آن ندارم.
- تنها چیز قابل تحویل برای یک پروژه موفق، برنامه ای است که کار کند.
- مهندسی نرم افزار، ما را وادار می سازد که مستندات حجیم و بیهوده تهیه کنیم و از سرعت ما می کاهد.

جلسه ۳– مدل های فرایند بخش ۱

۱. مدل فرایند کلی

۱،۱. فرایند نرم افزار

فرایند نرم افزار چیست؟

- مراحل تولید نرم افزار
- نقشه راه: گام‌هایی را که باید برای نیل به یک هدف برداشت، تعیین می‌کند. نقشه راه یعنی مطالعه و مشاهده همه ابعاد ممکن انجام یک کار با همه جزئیات به نحوی که به همه سوالات پاسخ منطقی ارایه شود و اهداف پیش بینی شده محقق گردد.

چرا فرایند نرم افزار؟

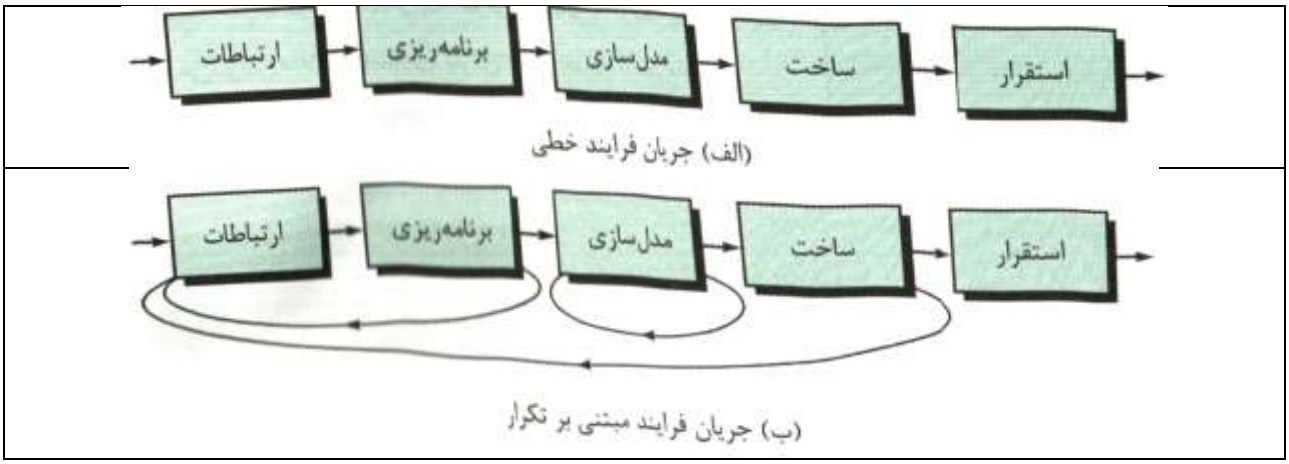
- باعث ثبات، کنترل و سازماندهی می شود.
- جلوگیری از بی برنامگی و آشوب و بهم ریختگی

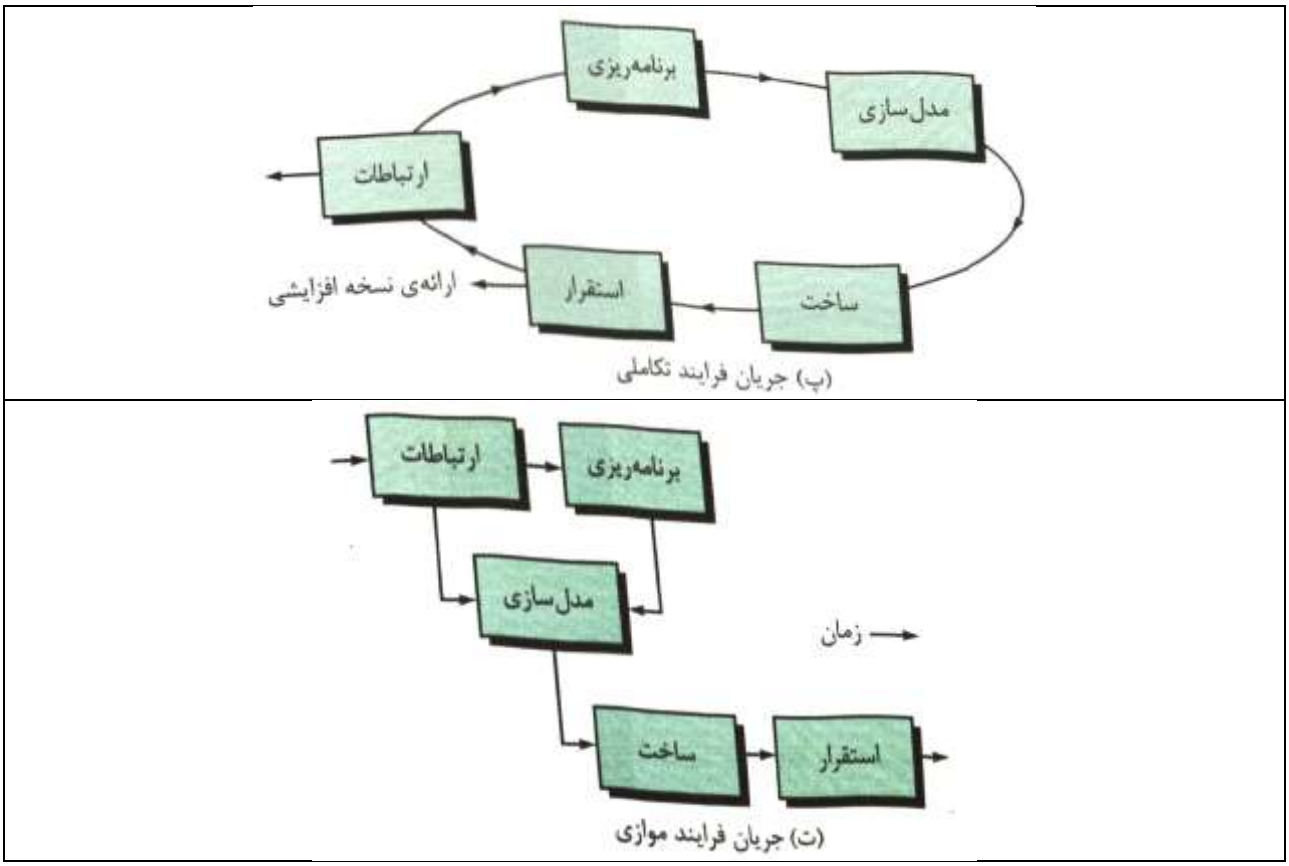
حاصل کار چیست؟

- برنامه ها، داده ها و مستندات

۱،۲. جریان فرایند

جریان فرایند شرح می دهد که فعالیت های چتری و چارچوبی، از نظر زمانی چگونه سازماندهی می شوند.





۱,۳. نمونه ای از کنش های فعالیت چارچوبی ارتباطات

در پروژه ای کوچک

- برقراری تماس تلفنی با مشتری
- بحث درباره خواسته های مشتری و یادداشت برداری
- سازماندهی و خلاصه سازی یادداشت ها
- ارسال به ایمیل مشتری جهت بازبینی
- فعالیت چارچوبی ارتباطات در پروژه های کوچک

در پروژه ای بزرگ

- شروع (inception)
- استخراج (elicitation)
- شناخت (elaboration)
- مذاکره (negotiation)
- تعیین مشخصات (specification)
- اعتبارسنجی (validation)

۲. الگوهای فرایند

۲,۱. الگوهای فرایند چه هستند؟

- راهکارهای اثبات شده برای مشکلات موجود در فرایند نرم افزار
- برای جلوگیری از تکرار دوباره اشتباهات
- اختراع نکردن دوباره چرخ
- شامل توصیف مشکل، توصیف محیط مشکل و ارایه راهکار

۲,۲. قالب الگوهای فرایند

- روشی سازگار برای توصیف راهکارهای مساله در حطیه فرایند نرم افزار

اجزای قالب	توضیح	مثال
نام الگو	به الگو نامی با معنا داده می شود.	Technical Review
نیروها	محیطی که الگو در آن مشاهده می شود و مواردی که ممکن است بر راهکار مساله تاثیر بگذارند.	
نوع الگو	الگوی مرحله ای: مساله ای مرتبط با یک فعالیت چارچوبی را برای فرایند تعریف می کند. الگوی وظیفه ای: مساله ای مرتبط با یک کنش یا وظیفه کاری نرم افزاری را تعریف می کند. الگوی فازی: یک سری فعالیت های چارچوبی را تعریف می کند که درون فرایند رخ می دهد.	الگوی مرحله ای: Establishing communication الگوی وظیفه ای: Requirement Gathering الگوی فازی: Prototyping
حیطه اولیه	شرایطی که الگو در آن کاربرد دارد. پیش از آغاز الگو، چه فعالیت های سازمانی یا تیمی قبلا رخ داده است؟	الگوی Planning مستلزم آن است که مشتریان و مهندسان نرم افزار یک ارتباط مبتنی بر همکاری برقرار کرده باشند.
مساله	مساله خاصی که قرار است توسط این الگو حل شود.	
راهکار	چگونگی پیاده سازی موفق الگو را توصیف می کند.	
حیطه حاصل	شرایطی که ماحصل پیاده سازی موفق الگوست.	
الگوهای مرتبط	فهرستی از همه الگوهای فرایند تهیه می کند که با این الگو ارتباط مستقیم دارند.	الگوی مرحله ای Communication، شامل الگوهای وظیفه ای Project Team می شود.
کاربردها و مثال ها	موارد خاصی را مشخص می کند که الگو در آن ها قابل اعمال است.	Communication در آغاز هر پروژه نرم افزاری لازم الاجرا است.

چند نمونه الگوی فرایند

- Customer Communication
- Requirement Extraction
- Scenario Description
- Scope Isolation

۳. مدل های فرایند تجویزی (سنتی)

۳,۱. مدل های فرایند تجویزی (سنتی) چه هستند؟

- این مدل ها در ابتدا برای نظم بخشیدن به آشوب موجود در توسعه نرم افزار پیشنهاد شدند.
- این مدل های سنتی به میزان معینی به کار مهندسی نرم افزار ساختار بخشیده اند و راهنمای اثربخشی برای تیم های نرم افزاری فراهم ساخته اند.
- مدل های فرایند تجویزی، مجموعه ای از عناصر تجویز شده و جریان کار تجویز شده را تعریف می کنند.
- اگر فرایند درست باشد، نتایج خودشان درست خواهند بود. (تاکاشی اوسادا)

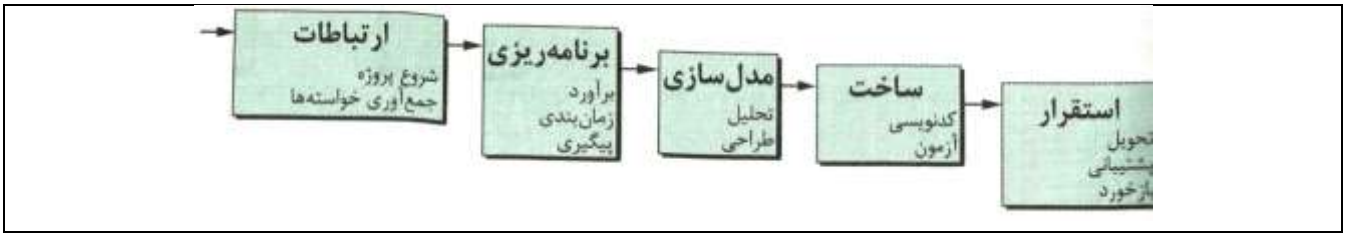
۴. مدل های فرایند آبشاری

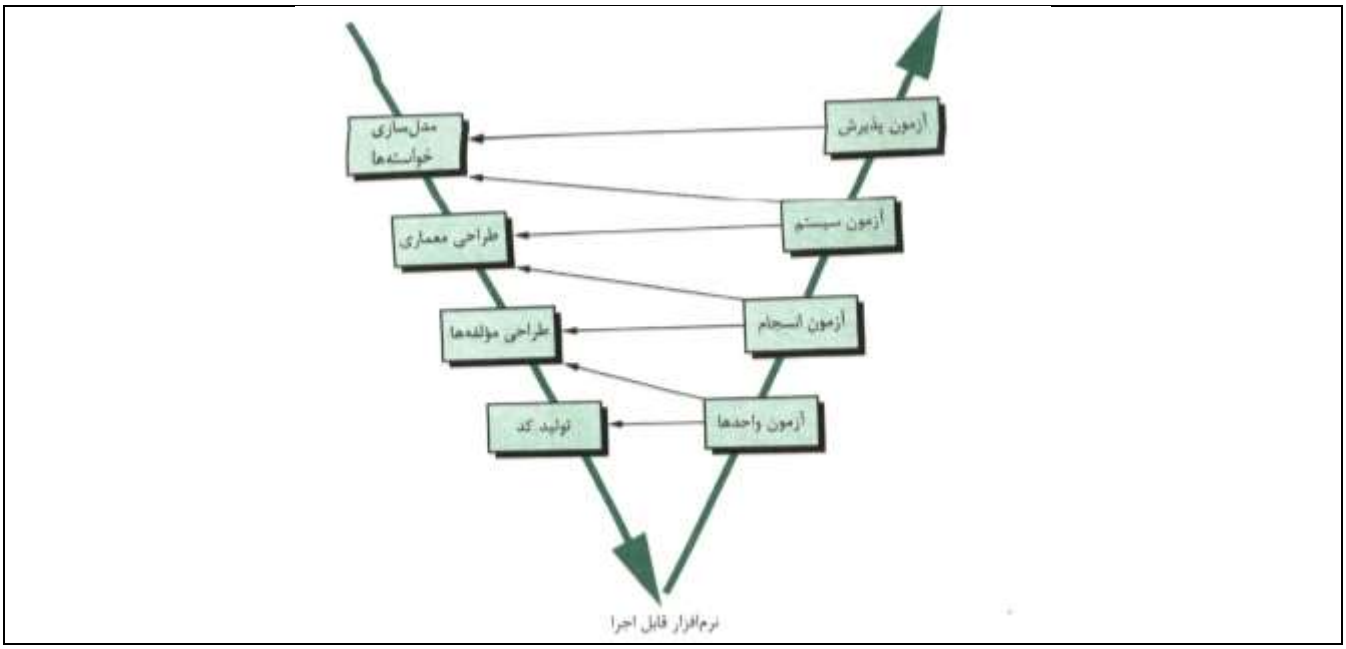
۴,۱. انواع مدل های آبشاری

انواع	توضیح
مدل خطی	روشی سیستماتیک و ترتیبی برای توسعه نرم افزار که با مشخص کردن خواسته ها توسط مشتری آغاز می شود و از طریق برنامه ریزی، مدل سازی، ساخت و استقرار پیش می رود و در پشتیبانی مستمر نرم افزار کامل شده و به اوج می رسد.
مدل V	چگونگی ارتباط کنش های وارسی و اعتبار بخشی با کنش های قبلی مهندسی را نشان می دهد.

۴,۲. کاربرد

- کاربرد: زمانی که خواسته های مربوط به یک مساله به خوبی شناخته شده اند.





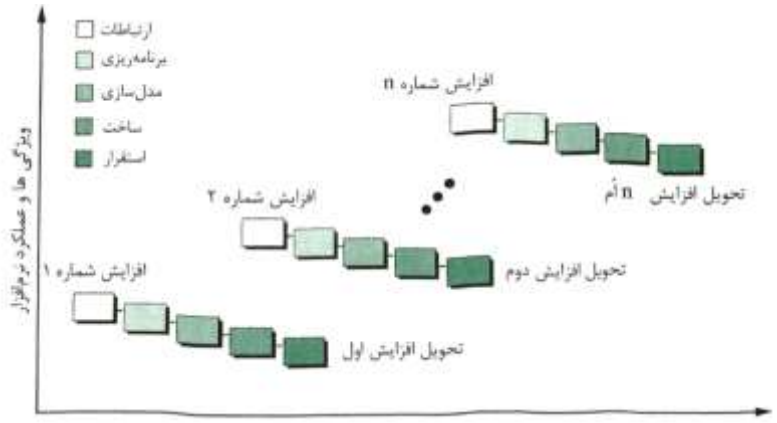
۴,۳. مشکلات مدل های آبشاری

- پروژه های واقعی به ندرت جریان ترتیبی پیشنهادی این مدل ها را دنبال می کنند.
- عدم امکان بیان کلیه نیازمندی ها توسط مشتری در مرحله اول
- عدم حوصله مشتری تا صبر کند یک نسخه از برنامه را آخرین روزهای پروژه ببیند.
- برخی از اعضای تیم پروژه باید منتظر سایر اعضای تیم بمانند تا وظایف وابسته انجام شود.

۵. مدل های فرایند افزایشی

۵,۱. مدل های افزایشی

- وضعیت های فراوانی وجود دارد که در آن ها خواسته های اولیه نرم افزار به خوبی تعریف شده اند، ولی ممکن است نیاز به فراهم کردن سریع مجموعه محدودی از عملکردهای نرم افزار برای کاربران و سپس پالایش و بسط بر اساس آن عملکردها در نسخه های بعدی نرم افزار ضرورت پیدا کند.
- مدل افزایشی، عناصر مدل ترتیبی خطی را با جریان های فرایند خطی و موازی تلفیق می کند.
- مدل افزایشی یک سری نرم افزار تحویل می دهد که هر کدام یک گام نامیده می شوند و این گام ها هر یک نسبت به سلف خود عملکرد بیشتری در اختیار مشتری قرار می دهند.



۵,۲. نمونه ای از مدل های افزایشی

- نسخه ۱: مدیریت فایل، تولید و ویرایش مستندات
- نسخه ۲: قابلیت های پیشرفته ویرایشی و تولید مستندات
- نسخه ۳: چک کردن املا و دستور
- نسخه ۴: قابلیت های پیشرفته صفحه آرایی

۶. مدل های فرایند تکاملی

۶,۱. چرا مدل های تکاملی؟

- نرم افزارها نیز همانند همه سیستم های پیچیده دیگر، در اثر مرور زمان تکامل می یابند.
- مهلت های زمانی محدود بازار، کامل کردن یک محصول نرم افزاری مفهومی را غیر ممکن می سازند، ولی یک نسخه محدود را باید وارد بازار کرد تا فشارهای رقابتی یا کاری را مرتفع سازد.
- در بسیاری موارد، زمان رساندن محصول به بازار، مهم ترین خواسته مدیریتی است. اگر زمان مقرر برای ارائه به بازار از دست برود، خود پروژه نرم افزاری ممکن است دیگر بی معنا شود.
- در این وضعیت ها، مهندسان نرم افزار به مدل فرایندی نیاز دارند که به طور مشخص برای محصول طراحی شده باشد و با گذشت زمان تکامل می یابد.
- هدف مدل های تکاملی توسعه نر افزارهایی با کیفیت بالا به شیوه ای افزایشی یا تعاملی است.

۶,۲. مشکل مدل های تکاملی

- تهیه نمونه اولیه به دلیل قطعی نبودن تعداد چرخ های لازم برای ساخته شدن محصول، برای برنامه ریزی پروژه ایجاد مشکل می کند. زیرا اکثر تکنیک های براورد پروژه بر پایه چیدمان خطی فعالیت ها استوارند، لذا به خوبی در این الگو نمی گنجند.

۶,۳. انواع مدل های تکاملی

مدل نمونه سازی اولیه	مدل مارپیچی (حلزونی)	مدل توسعه همروند

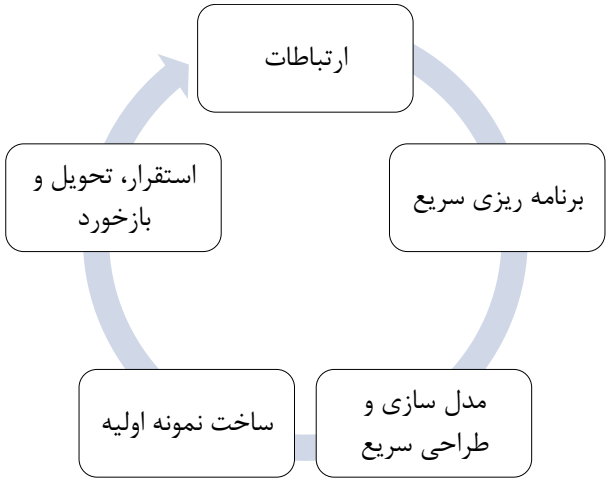
جلسه ۴– مدل های فرایند بخش ۲

۱. مدل نمونه سازی اولیه

۱,۱. کاربرد مدل

- هنگامی که مشتری شما خواسته ای مشروع دارد، ولی جزئیات چندانی در اختیار شما قرار نداده است
- وقتی سازنده از بازدهی یک الگوریتم، قابلیت تطابق با یک سیستم عامل خاص مطمئن نباشد.
- گرچه از تهیه نمونه اولیه می توان به عنوان یک مدل فرایند مستقل استفاده کرد، از آن بیشتر به عنوان تکنیکی استفاده می شود که می توان در حیطه هر کدام از مدل های فرایند ذکر شده پیاده کرد.

۱,۲. ساختار مدل



مراحل	توضیح
ارتباطات	جمع آوری خواسته ها – ملاقات مشتری و سازنده
برنامه ریزی سریع	تعیین اهداف کلی نرم افزار – شناسایی خواسته ها
مدل سازی و طراحی سریع	ارائه آن دسته از ویژگی‌های نرم افزار که به چشم مشتری می آید (مثل روش های ورود اطلاعات)
ساخت نمونه اولیه	ساخت یک نمونه اولیه
استقرار، تحویل و بازخورد	ارزیابی نمونه اولیه و استفاده از آن برای پالایش خواسته های نرم افزار

۱,۳. مشکلات مدل نمونه سازی اولیه

- مشتریان چیزی را می بینند که ظاهرا یک نسخه کاری از نرم افزار است ولی نمی دانند که این نمونه اولیه با موم سره بندی شده است. نمی دانند که به لحاظ شتابی که در به کارگیری داشته ایم، کیفیت کلی نرم افزار و قابلیت نگهداری درازمدت مدنظر نبوده است.
- مهندسان نرم افزار، غالبا برای بکارگیری هر چه سریع تر نمونه اولیه، در پیاده سازی دقیق آن کوتاه می آیند.

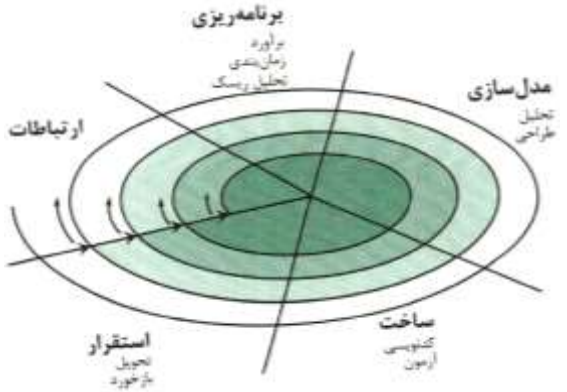
۲. مدل مارپیچی (حلزونی)

۲,۱. کاربرد مدل

- مدل مارپیچی یک مدل فرایند نرم افزاری تکاملی است که ماهیت تکراری مدل ساخت نمونه اولیه را با جنبه های کنترلی و سیستماتیک مدل ترتیبی خطی (آبشاری) تلفیق می کند.
- با استفاده از مدل مارپیچی، نرم افزار به صورت یک سری نگارش های تکاملی توسعه می یابد.

۲,۲. ساختار مدل

- نخستین مدار مارپیچ ممکن است منجر به ایجاد مشخصه ای از محصول گردد، ولی عبورهای بعدی در اطراف مارپیچ برای ایجاد یک نمونه اولیه و سپس نسخه های پیچیده تری از نرم افزار بکار رود.
- هر بار گذر از ناحیه برنامه ریزی، منجر به تنظیم دوباره طرح پروژه می شود.



۳. مدل توسعه همروند (concurrent Model)

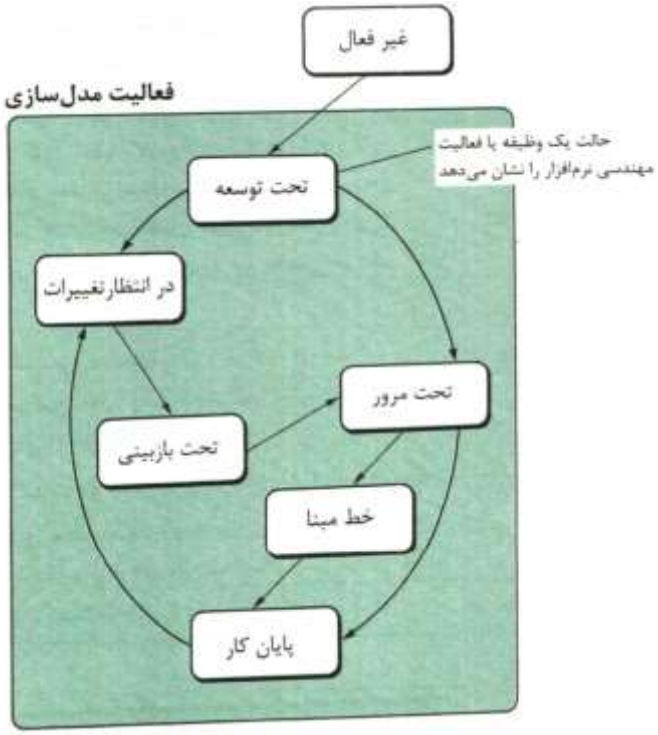
۳,۱. کاربرد مدل

- این مدل به تیم نرم افزار این امکان را می دهد عناصر تکراری و همروند هر کدام از مدل‌های فرایند توصیف شده را ارائه نماید.
- برای مثال فعالیت مدل سازی تعریف شده برای مدل مارپیچی با توجه به وظایف زیر قابل حصول است: ساخت نمونه اولیه، تحلیل و طراحی.
- این مدل غالبا برای پروژه های مهندسی ای مناسب است که تیم های مهندسی متفاوتی در آن نقش دارند.

۳,۲. ساختار مدل

- شکل زیر طرحی از یک فعالیت با مدل توسعه همروند ارائه می دهد. این فعالیت- مدل سازی- ممکن است در هر زمان مفروض، در یکی از حالت های ذکر شده باشد. (حالت، رفتاری از سیستم است که از بیرون قابل مشاهده است).
- به طور مشابه، فعالیت های دیگر مانند ارتباطات و ساخت را می توان به شیوه ای مشابه نمایش داد. همه فعالیت ها به صورت همروند وجود دارند ولی در حالت های متفاوت قرار می گیرند.

- مدل توسعه همروند، یک سری رویداد تعریف می کند که باعث گذار از حالتی به حالت دیگر برای هر یک از فعالیت های مهندسی نرم افزار می شوند.



۴. تو سعه مبتنی بر مولفه ها (Component-Based)

۴,۱. توضیح مدل

- مولفه های نرم افزاری آماده توسط عده ای از فروشندگان این مولفه ها توسعه داده می شوند، عملکرد مورد نظر را با واسطهایی مناسب فراهم می آورند، به طوری که مولفه را می توان به خوبی در سیستم در حال ساخت الحاق کرد.
- این روش بسیاری از خصوصیات مدل مارپیچی را در بر می گیرد، ماهیتی تکاملی دارد و برای ایجاد نرم افزار به رویکردی تکراری نیاز دارد.
- در این مدل، برنامه های کاربردی از به هم پیوستن مولفه های نرم افزاری آماده ساخته می شوند.
- فعالیت های مدل سازی و ساخت با شناسایی مولفه های کاندیدا اغاز می شود. این مولفه ها را می توان به صورت پیمانه های نرم افزاری سنتی یا کلاس ها و پکیج های شی گرا شی گرا طراحی کرد.
- این مدل استفاده مجدد از نرم افزار را میسر می سازد که می تواند به کاهش زمان چرخه توسعه و نیز کاهش هزینه های پروژه دست پیدا کند.

۵. مدل روشهای رسمی (Formal Methods Model)

۵,۱. توضیح مدل

- مجموعه ای از فعالیت ها که به مشخص کردن ریاضی و رسمی نرم افزار کامپیوتری منجر می شود.
- مدل روش های رسمی گرچه عمومیت ندارد اما نوید نرم افزاری عاری از نقص را می دهد.

- توسعه ی این مدل ها در حال حاضر وقت گیر است و به آموزش گسترده نیاز مند است.
- ارتباط با مشتری که دید فنی ندارد دشوار است.
- مناسب برای ساخت نرم افزارهای ایمنی-حیاتی مثل نرم افزارهای دستگاه های پزشکی و هوافضا یا نرم افزارهایی که در صورت بروز خطا در آن ها دستخوش زیان های اقتصادی کلان می شوند.

۶. توسعه نرم افزار به روش جنبه گرا (Aspect Oriented)

۶,۱. توضیح مدل

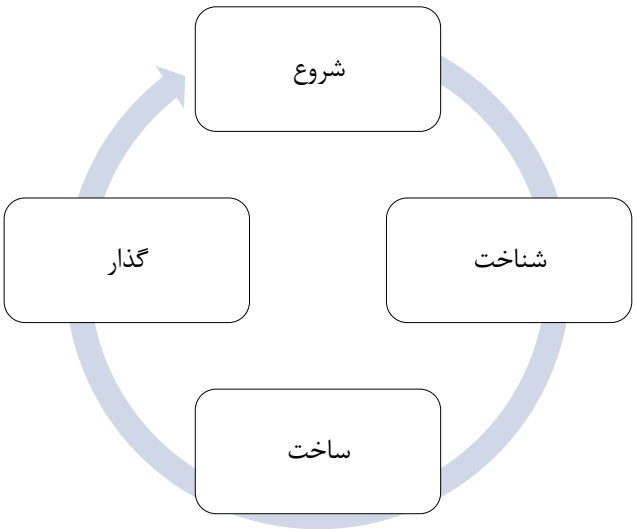
توسعه نرم افزار به روش جنبه گرا یا برنامه نویسی جنبه گرا یک الگوی مهندسی نسبتا جدید است که رویکردی فرایندی و روش شناختی برای تعریف، مشخص سازی، طراحی و ساخت جنبه ها ارایه می کند.

۷. فرایند یکپارچه (Unified Process)

۷,۱. توضیح مدل

- در فرایند یکپارچه اهمیت برقراری ارتباط با مشتریان و روش های ساده و روان برای توصیف دیدگاه مشتریان (Use Case) یک سیستم به خوبی درک می شود.
- در این فرایند بر اهمیت نقش معماری نرم افزار تاکید می شود و به معمار کمک می شود تا به اهداف درستی از قبیل قابلیت درک و استفاده مجدد توجهی خاص داشته باشد.
- در این فرایند، یک جریان مبتنی بر تکرار و افزایشی پیشنهاد می شود.
- نتیجه این فرایند، زبان مدل سازی یکپارچه (UML) است که حاوی یک نمادگذاری قدرتمند برای مدل سازی و توسعه سیستم های شی گرا است.

۷,۲. ساختار مدل



مراحل	توضیح
شروع	○ شامل هر دو فعالیت برقراری ارتباط با مشتریان و برنامه ریزی می شود. از طریق همکاری با ذینفعان، خواسته های نرم افزار شناسایی می شود و یک معماری تقریبی برای سیستم پیشنهاد می شود. خواسته های نرم افزار از طریق یک مجموعه Use Case مقدماتی توصیف می شود. در برنامه ریزی، منابع شناسایی می شوند، ریسک های اصلی ارزیابی می شوند و یک برنامه زمانی تدوین می شود.
شناخت	○ شامل فعالیت های برقراری ارتباط و مدل سازی می شود. ○ در این مرحله Use Case های مقدماتی که به عنوان بخشی از مرحله شروع ایجاد شده اند، پالایش یافته و بسط داده می شود. ○ پنج نمای متفاوت نرم افزار بسط داده می شوند: مدل Use Case، مدل خواسته ها، مدل طراحی، مدل پیاده سازی و مدل استقرار.
ساخت	○ مرحله ساخت با استفاده از مدل معماری به عنوان ورودی، مولفه های نرم افزاری را که هر کدام از Use Case ها را عملیاتی می کنند، ایجاد می کند یا آنها را توسعه می دهد. ○ به موازاتی که مولفه ها پیاده سازی می شوند، آزمون واحدها برای هر کدام طراحی و اجرا می شوند. سپس آزمون های انسجام و پذیرش انجام می شود.
گذار	○ نرم افزار برای آزمون بتا به کاربران نهایی داده می شود و بازخورد به دست آمده از کاربران هم نقایص و هم تغییرات لازم را گزارش می کند. ○ اطلاعات پشتیبانی از قبیل جزوات راهنمای کاربران، دستورالعمل اشکال زدایی و روال های نصب ایجاد می شود.

۸. مدل های فرایند تیمی و شخصی

۸,۱. توضیح مدل

- بهترین فرایند نرم افزاری، فرایندی است که به کسانی که کار می کنند نزدیک باشد.
- در یک شرایط ایده ال باید فرایندی را ایجاد کرد که به بهترین وجه بر نیازهای شما تطبیق یابد.

۸,۲. فرایند نرم افزار شخصی (Personal Software Process)

در مدل فرایند های نرم افزار شخصی پنج فعالیت چارچوبی تعریف می شود:

مراحل	توضیح
برنامه ریزی	○ خواسته ها شناسایی می شود و براورد منابع و تعیین اندازه پروژه انجام می شود. ○ وظایف لازم برای توسعه نرم افزار تعیید و زمان بندی پروژه انجام می شود.
طراحی سطح بالا	○ مشخصات خارجی برای هر کدام از مولفه هایی که قرار است تعیین شود و طراحی مولفه انجام می شود. ○ نمونه های اولیه ساخته می شوند، همه مسائل و مشکلات ثبت و پیگیری می شوند.
مرور طراحی سطح بالا	○ روش های وارسی رسمی برای یافتن خطاهای طراحی اعمال می شوند. ○ معیارهای مربوط به وظایف مهم و نتایج کار نگهداری می شوند.

توسعه	○ طرای سطح بالا پالایش و بازبینی می شود. کدها تهیه، بازبینی، کامپایل و آزموده می شوند.
پایان کار	○ با استفاده ازمعیارها وموازین جمع آوری شده، اثر بخشی فرایند تعیین می شود.

۸,۳.فرایند نرم افزار تیمی (Team Software Process)

برای انجام پروژه های نرم افزاری در پایه ی صنعتی را تیمی از دست اندرکاران انجام می دهند. هدف فرایند نرم افزار تیمی تشکیل تیم پروژه ی خود هدایت گر است. فعالیت های زیر را برای فرایند نرم افزار تیمی تعریف می کنند.

- تشکیل تیم خود هدایت گری که کار خود را برنامه ریزی و پیگیری می کند،اهداف را تعیین می کند و خود به تعیین فرایند و طرح ها اقدام می کنند.
- نشان دادن شیوه ی راهبردی و ایجاد انگیزه در تیم ها به مدیران و چگونگی کمک به انها در حفظ حداکثر کارایی.
- شتاب بخشیدن به بهبود فرایند نرم افزاری.
- فراهم ساختن دستور بهسازی برای سازمان های بالغ.
- تسهیل آموزش دانشگاهی مهارت های تیمی در سطح صنعتی.

۹. محصول و فرایند

- اگر فرایند ضعیف باشد، محصول نهایی نیز ضعیف خواهد بود. ولی اتکای بیش از حد بع فرایند نیز خطرناک است.
- افراد به همان اندازه که از محصول نهایی احساس رضایت می کنند، از فرایند خلاق نیز احساس رضایت می کنند.
- یک نرم افزار نویس حرفه ای خلاق نیز باید به همان اندازه که از محصول نهایی احساس رضایت می کند، از فرایند نیز راضی باشد.

جلسه ۵– توسعه چابک بخش ۱

۱. مقدمه

۱,۱. توسعه چابک چیست؟

توسعه چابک تلفیقی از فلسفه و مجموعه ای از دستورالعمل های توسعه است:

- جلب رضایت مشتری
- تحویل افزایشی نرم افزار از ابتدای پروژه
- تیم های پروژه کوچک با انگیزه بالا
- روش های غیر رسمی
- حداقل محصولات کاری مهندسی نرم افزار
- سادگی کلی در توسعه نرم افزار
- برقراری ارتباط فعال و پیوسته میان توسعه دهندگان و مشتریان

۱,۲. ویژگی های توسعه چابک

ویژگی ها	توضیح
اهمیت	○ محیط کاری جدید سریع و پیوسته در حال تغییر
انجام دهنده	○ تشکیل یک تیم چابک خود سازمانده با مشارکت مهندسان نرم افزار و سایر ذینفعان پروژه (مدیران، مشتریان و کاربران) ○ تیمی که سازماندهی اش بر عهده خودش است و سرنوشت اش را خود کنترل می کند. ○ تیم چابک، به ارتباطات و همکاری میان همه افراد تیم رسیدگی می کند.
مراحل کار	○ فعالیت های چارچوبی پایه (ارتباطات، برنامه ریزی، مدل سازی، ساخت و استقرار) ولی حداقلی و کمینه
محصول کار	یک نرم افزار عملیاتی که به موقع تحویل مشتری شود.

۱,۳. کی به وجود آمد؟

در سال ۲۰۰۱ توسط کنت بک و ۱۶ نفر دیگر از سازندگان نرم افزار (اتحادیه چابک) بیانیه ای صادر کردند:

ما با انجام توسعه نرم افزار و کمک به دیگران در انجام این کار به کشف راههای بهتری نائل آمده ایم و به این نتیجه رسیده ایم که:

- افراد و تعامل ها را بر فرایندها و ابزارها
- نرم افزار عملیاتی را بر مستندات جامع
- همکاری با مشتری را بر مذاکره قرارداد
- و پاسخ به تغییر را بر دنبال کردن یک برنامه برتری دهیم.

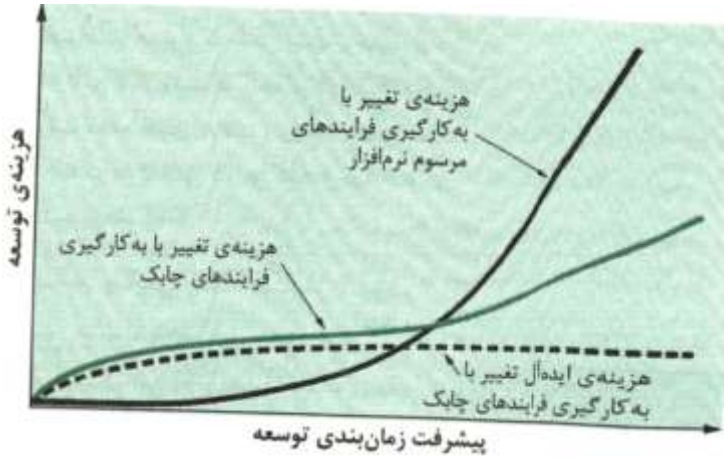
۲. چابکی چیست؟

این روزها هنگام توصیف یک فرایند نرم‌افزار مدرن، چابکی واژه‌ای است که به‌وفور به گوش می‌رسد. تیم چابک تیمی فرز و چالاک است که قادر است به تغییرات پاسخ مناسب بدهد. تغییر چیزی است که در توسعه‌ی نرم‌افزار، بسیار با آن مواجه می‌شویم. تغییرات در نرم‌افزارهایی که در حال ساخته‌شدن هستند، تغییرات در اعضای تیم، تغییرات به دلیل فن‌آوری جدید، تغییرات از هر نوع که ممکن است بر محصول در حال ساخت یا محصولی که محصول نهایی را ایجاد می‌کند، تأثیر گذار باشند. هر آنچه در نرم‌افزار انجام می‌دهیم باید خود حاوی ویژگی‌هایی برای پشتیبانی از تغییرات باشد؛ چیزی که قلب و روح نرم‌افزار است. تیم چابک می‌داند که نرم‌افزار توسط افرادی توسعه می‌یابد که در قالب تیمی کار می‌کنند و مهارت‌های این افراد و توانایی ایشان در همکاری، هسته‌ی اصلی موفقیت پروژه است.

نکته: مرتکب این اشتباه نشوید که چابکی این مجوز را به شما می دهد که بدون برنامه ریزی به راهکار برسید. یک فرایند لازم است و انضباط ضروری است.

۳. چابکی و هزینه های تغییر

در نمودار زیر هزینه تغییرات به عنوان تابعی از زمان در توسعه نرم افزار نمایش داده شده است:



۴. فرایند چابک چیست؟

۴,۱. فرض های کلیدی در فرایند چابک

- پیش بینی این که کدام خواسته های نرم افزاری باقی خواهند ماند و کدام یک از آن ها تغییر می کند، دشوار است. پیش بینی این که اولویت های مشتری با پیشرفت پروژه چگونه تغییر می کند نیز به همان اندازه دشوار است.
- برای بسیاری از انواع نرم افزارها، طراحی و ساخت در بین هم انجام می شوند. یعنی هر دو فعالیت را باید به طور موازی انجام داد، به طوری که مدل های طراحی به هنگام ایجاد، به اثبات برسند. پیش بینی این که چه مقدار طراحی مورد نیاز است، قبل از به کارگیری ساخت برای به اثبات رساندن طراحی، دشوار است.
- تحلیل، طراحی، ساخت و آزمون ممکن است (از دیدگاه برنامه ریزی) به آن اندازه که ما دوست داریم، قابل پیش بینی نباشد.

۴,۲. اصول چابکی

در پیمان چابک، ۱۲ اصل برای کسانی که می خواهند به چابکی دست پیدا کنند، تعریف شده است.

- جلب رضایت مشتری از طریق تحویل زودهنگام و پیوسته نرم افزارهای ارزشمند، بیشترین اولویت را نزد ما دارد.
- پذیرا بودن تغییرات در خواسته ها حتی در اواخر فرایند توسعه.
- در فرایندهای چابک، تغییرات برای مزایای رقابتی مشتریان تحت کنترل هستند.
- تحلیل پیوسته نرم افزارهای کاری از دو هفته گرفته تا دو ماه، که بازه های زمانی کوتاه تر باید در اولویت قرار داده شوند.
- دست اندرکاران و افراد تجاری باید در سرتاسر پروژه هر روز با هم کار کنند.
- سپردن پروژه به افراد با انگیزه، فراهم سازی محیط و پشتیبانی مورد نیاز آن ها و اطمینان کردن به آن ها در انجام کارها.
- اثربخش ترین و موثرترین روش انتقال اطلاعات به درون و بیرون تیم توسعه، گفتگوی رو در رو است.
- نرم افزار کاری، میزان اصلی در سنجش پیشرفت است.
- فرایندهای چابک، توسعه پایدار را ارتقا می بخشند. حامیان، سازندگان و کاربران باید قادر به حفظ سرعت ثابت در پیشرفت کار باشند.
- توجه پیوسته به اعتلای فنی و طراحی خوب، باعث بهبود افزایش چابکی می شود.
- سادگی – هنر به حداکثر رساندن کارهایی که انجام نمی شوند- ضروری است.
- بهترین معماری ها، خواسته ها و طراحی ها از تیم های خودسازماندهی شده ظهور می کند.
- تیم در بازه های منظم، بازخوردی از میزان بهبود اثربخشی خود ارائه می دهد و سپس رفتار خود را مطابق این بازخورد تنظیم می کند.

۴,۳. نکات مهم

- گرچه فرایندهای چابک با آغوش باز به استقبال تغییرات می روند، هنوز هم بررسی دلایل تغییرات اهمیت دارد.
- نرم افزاری که کار کند مهم است، ولی فراموش نکنید که انواع صفات کیفیتی از جمله قابلیت اطمینان، قابلیت استفاده و قابلیت نگهداری را هم باید داشته باشد.
- روش های چابک بیشتر خاصیت خود را مرهون دانش نهفته در تیم است نه در دانش نوشته شده روی کاغذ.

۴,۴. سیاست توسعه چابک

توسعه چابک در مقابل روش های سنتی

چابک‌ها مواردی حدی را ذکر می‌کند. «روش‌شناسان سنتی، یک مشت آدم‌های فاقد خلاقیت هستند که ترجیح می‌دهند مستنداتی بدون نقص تهیه کنند تا اینکه سیستمی کاری ارائه دهند که نیازهای تجاری را برآورده کند.» او در نقطه‌ی مقابل، موفقیت اردوگاه مهندسی نرم‌افزار سنتی را چنین توصیف می‌کند: «روش شناسان سبک بال یا «چابک» یک مشت نفوذگر با استعدادند که وقتی تلاش می‌کنند اسباب بازی‌های خود را بزرگ کنند و به نرم‌افزارهایی در سطح شرکت‌ها تبدیل کنند، کلی ذوق‌زده می‌شوند.»

نکته: نباید بین چابکی و مهندسی نرم افزار یکی را انتخاب کنید بلکه یک رویکرد مهندسی نرم افزار انتخاب کنید که چابک باشد.

۴،۵. عوامل انسانی

در توسعه چابک بر استعدادها و مهارت های افراد و شکل دهی به فرایند بر اساس افراد و تیم های موجود تاکید دارد. خصوصیات کلیدی عوامل انسانی چابک عبارتند از:

خصوصیات	توضیح
رقابت	○ رقابت شامل استعداد ذاتی، مهارت های خاص مرتبط با نرم افزار و آگاهی کلی از فرایندی است تیم برای استفاده برگزیده است.
کانون توجه مشترک	○ گر چه ممکن است اعضای تیم چابک وظایف متفاوتی را به انجام برسانند و مهارت های متفاوتی را وارد پروژه کنند، همه آن ها باید هدف واحد، تحویل نسخه جدیدی از نرم افزار به مشتری در زمان مقرر را کانون توجه خود قرار دهند.
همکاری	○ برای ایجاد اطلاعاتی که همه طرف های ذی نفع را در فهم کار تیم یاری دهد، و انتشار دادن اطلاعات که برای مشتری ارزش تجاری در بر داشته باشد، اعضای تیم باید همکاری کنند.
توانایی تصمیم گیری	○ تیم باید خودمختاری داشته باشد یعنی اجازه تصمیم گیری برای مسائل فنی و پروژه.
توانایی حل مساله با حمل ابهام	○ تیم چابک پیوسته ناگزیر از مقابله با ابهام است و پیوسته در معرض تغییر قرار دارد. ○ در برخی موارد، تیم باید پذیرای این واقعیت باشد که مساله ای که امروز در حال جل کردن آن است، ممکن است فردا مساله ای باشد که دیگر نیازی به حل آن نباشد.
احترام و اطمینان متقابل	○ تیم چابک باید تیمی قوام یافته باشد که اطمینان و احترامی را از خود به نمایش می گذارد که به واسطه آن اعضای تیم چنان به هم پیوسته می شوند که کلیت حاصل چیزی بیش از مجموع اجزای تشکیل دهنده باشد.
خودسازماندهی	○ تیم چابک، خودش را سازماندهی می کند تا کارها به انجام برسد. ○ تیم، فرایند را سازماندهی می کند تا به بهترین نحو در محیط محلی اسکان یابد. ○ تیم، زمان بندی کاری را سازماندهی می کند تا به بهترین نحو، تحویل یک نسخه از نرم افزار را امکان پذیر سازد. ○ تیم خود سازمانده کنترل کارهایش را بر عهده دارد. این تیم خودش به تعهداتش عمل می کند و طرح هایی برای انجام آن ها تعریف می کند.

جلسه ۶– توسعه چابک بخش ۲

۱. برنامه نویسی XP

۱,۱. برنامه نویسی XP

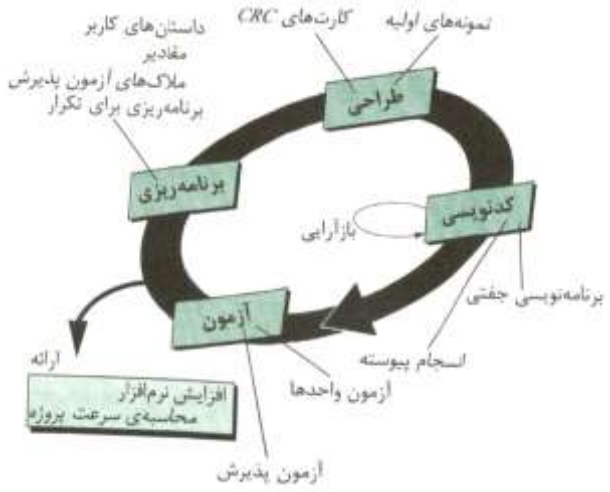
- XP: پر کاربردترین رویکرد در توسعه نرم افزار به روش چابک
- IXP: XP صنعتی که پالایشی از XP است برای استفاده در سازمان های بزرگ است.

۱,۲. ارزش های XP

مبنای همه کارهای انجام شده در XP:

- ارتباطات اثر بخش میان ذی نفعان
- سادگی یا توجه به نیازهای فوری و نه نیازهای آینده
- بازخورد از نرم افزار، مشتری و سایر اعضای تیم نرم افزاری
- جرات، جسارت و انضباط برای پایبندی به XP
- احترام با دنبال کردن این ارزش ها

۱,۳. فرایند XP



مراحل	توضیح
برنامه ریزی	○ گوش سپردن برای جمع آوری خواسته ها که اعضای تیم را قادر به شناخت حیطه تجاری مناسب برای نرم افزار سازد. ○ گوش سپردن، به ایجاد مجموعه ای از داستان ها می انجامد که خروجی لازم، برای نرم افزاری که قرار است ساخته شود، ویژگی ها و عملکردها را توصیف می کند. ○ هر داستان مشابه (Use Caseها) توسط مشتری نوشته می شود و روی یک کارت شاخص قرار داده می شود.

طراحی	○ در XP، استفاده از کارت های CRC، سازوکاری موثر برای تفکر درباره نرم افزارهای شی گرا محسوب می شود. ○ در XP تاکید از روی اهمیت طراحی برداشته می شود که البته همه با آن موافق نیستند. در واقع، مواقعی پیش می آید که باید بر طراحی تاکید شود.
کد نویسی	○ اکثر تیم های نرم افزاری پر هستند از آدم های تک رو، اگر می خواهید برنامه نویسی جفتی با بازدهی بالا داشته باشید، باید این روحیه را در آن ها تغییر دهید.
آزمون	○ آزمون های پذیرش XP، از داستان های کاربر به دست می آید.

۱,۴. انتقادات به XP

- متغیر بودن خواسته ها، زیرا مشتری عضو فعالی از تیم XP است لذا تغییراتی که در خواسته ها به عمل می آید به صورت غیر رسمی تقاضا می شود.
- نیازهای متناقض مشتریان در پروژه های با چند مشتری
- خواسته ها به صورت غیر رسمی بیان می شوند.
- فقدان طراحی رسمی

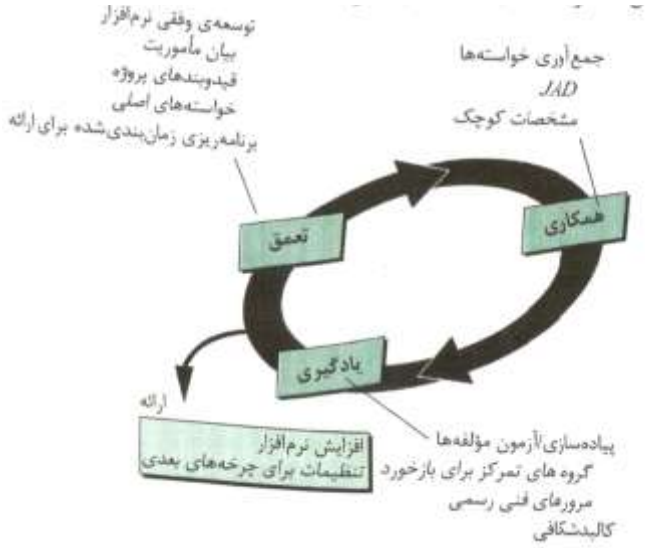
۱,۵. برنامه نویسی جفتی در توسعه چابک

- برنامه نویسی جفتی یک شیوه برنامه نویسی است که دو نفر به صورت همزمان پشت یک کامپیوتر می‌نشیند و شروع به برنامه نویسی می‌کنند.
- فردی که کد نویسی می‌کند به "driver" مشهور است و فردی که بررسی می‌کند به "navigator".
- فرد "navigator" (کسی که کیبورد در دست او نیست) باید کامپیوتری برای خود داشته باشد، نه برای توسعه بلکه برای آزمایش نظرات خود، مرور مستندات و غیره.
- فرد "driver" (کسی که کیبورد در دست اوست) باید کد نویسی کند.
- برنامه نویسی جفتی کیفیت کد را افزایش می‌دهد.
- برنامه نویسی جفتی تمرکز تیم را افزایش می‌دهد (به عنوان مثال وقتی که فرد پشت سر شما می‌گوید که " هی، آیا واقعا این کار در این اسپرنت لازم است؟"
- به طور شگفت آوری اکثر افراد نسبت به برنامه نویسی جفتی موضع تدافعی می‌گیرند. ولی همین که یک بار آن را امتحان می‌کنند دوست دارند که آن را یاد بگیرند.
- بهتر است که جابجایی جفت‌ها مرتبا صورت بگیرد.

۲. سایر مدل های فرایند چابک

توسعه وفقی نرم افزار (ASD)	اسکرام (Scrum)	روش توسعه سیستم های پویا (DSDM)	کریستال	توسعه ویژگی محور (FDD)
	توسعه نرم افزار ناب (LSD)	مدل سازی چابک (AM)	فرایند یکپارچه چابک (AUP)	

۳. توسعه وقفی نرم افزار (ASD)



- کاربرد: ساخت نرم افزارهای و سیستم های پیچیده
- فلسفه: همکاری انسانی و خود سازماندهی تیمی
- همکاری موثر با مشتری تنها در صورتی امکان پذیر خواهد بود که هر گونه نگرش «ما و آن ها» را کنار بگذارید.
- ASD بر یادگیری به عنوان عنصری کلیدی در دستیابی به تیم «خود سازمانده» تاکید دارد.

۴. اسکرام (Scrum)

۴,۱. ساختار اسکرام

- این روش در اوایل دهه ۱۹۹۰ شکل گرفت.
- اسکرام برای پروژه هایی که تغییرات در آنها زیاد است، بسیار مناسب می‌باشد.
- اسکرام شامل مجموعه ای از الگوهای فرایند می شود که بر اولویت های پروژه، واحدهای کاری قطعه بندی شده، ارتباطات و بازخورد پیاپی از مشتری تاکید دارد.
- اسکرام چارچوبی برای توسعه چابک است که با تعریف فرآیندها، نقش‌ها و آرتیفکت‌های مشخص به تیم‌های نرم افزاری کمک می‌کند تا چابک شوند.
- در اسکرام، نرم افزار به صورت مرحله به مرحله توسعه پیدا می کند که به این مراحل، اسپرینت گفته می‌شود. هر اسپرینت معمولا بین ۲ تا ۴ هفته طول می‌کشد و در انتها یک نرم افزار کار کننده تحویل مشتری می‌گردد.

۴,۲. تیم های خود سازمانده در اسکرام

- اسکرام بر پایه تیم‌های خود سازمانده و میان کارکردی بنا نهاده شده است.
- تیم اسکرام خود سازمانده است یعنی رهبر تیم تصمیم نمی‌گیرد که کدام عضو تیم چه کاری را انجام دهد یا اینکه مسائل چگونه باید حل شوند. این امور توسط کل تیم تصمیم گیری می‌شوند.
- تیم اسکرام میان کارکردی است یعنی هر عضو تیم باید قادر باشد هر کاری انجام دهد و از شروع شکل گیری ایده تا پیاده سازی و تحویل آن مشارکت داشته باشد.

۴,۳. فعالیت های چارچوبی:

خواسته ها	تحلیل	طراحی	تکامل	تحویل

۵. ابزارهای فرایند چابک

ابزارها در جهت پشتیبانی از عناصر کلیدی کلیه مدل های فرایند چابک کمک می کنند:

استخدام افراد مناسب	همکاری تیمی	برقراری ارتباط میان طرف های ذی نفع	مدیریت غیر مستقیم

ابزارها	توضیح
ابزارهای اجتماعی	○ در مرحله استخدام شروع می شوند.
ابزارهای فنی	○ به تیم های توزیع شده کمک می کنند، تا حضور فیزیکی را شبیه سازی کنند.
ابزارهای فیزیکی	○ به افراد امکان کار در قالب کارگاهی را می دهند.
ابزارهای همکاری و ارتباطی	○ تسهیل ارتباطات بین اعضای تیم

جلسه ۷– اصول راهنما در مهندسی نرم افزار

۱. مقدمه

۱،۱. اصول راهنما چیست؟

مهندسی نرم افزار مجموعه ی وسیعی از اصول، مفاهیم، روش ها و ابزارهاست که باید در توسعه نرم افزار در نظر گرفت.

۱،۲. چرا اهمیت دارد؟

- فرآیند نرم افزار برای رسیدن به هدف یک نقشه راه در اختیار ما قرار می دهد.
- کار مهندسی جزئیات لازم برای طی این نقشه راه فراهم می کند.

۲. اصول هسته ای

۲،۱. اصول هسته ای

مجموعه ای از اصول هسته ای وجود دارد که راهنمای مهندسی نرم افزار است و به استفاده از یک فرایند نرم افزار با معنا و اجرای اثربخش روش های مهندسی نرم افزار کمک می کند. برخی از این اصول عبارتند از:

- فراهم ساختن ارزش برای کاربر نهایی
- حفظ سادگی
- حفظ چشم انداز(محصول و پروژه)
- دانستن این مطلب که دیگران آنچه را که شما ساخته اید مصرف می کنند و باید بتوانند آن را درک کنند
- نگاه به آینده
- برنامه ریزی برای استفاده مجدد
- تفکر!

۲،۲. اصول راهنمای فرایند مهندسی

- اصل ۱: چابک باشید.
- اصل ۲: در هر مرحله کیفیت را در کانون توجه قرار دهید.
- اصل ۳: آمادگی انطباق را داشته باشید و انعطاف پذیر باشید.
- اصل ۴: تیمی اثربخش و خودسازمانده تشکیل دهید.
- اصل ۵: سازوکارهایی برای برقراری ارتباط و هماهنگی برقرار کنید.
- اصل ۶: مدیریت تغییرات
- اصل ۷: ارزیابی ریسک
- اصل ۸: ایجاد محصولات کاری که برای دیگران ارزش فراهم می کنند.

۲,۳. اصول راهنمای کار مهندسی نرم افزار

- اصل ۱: تقسیم و حل؛ یک مسئله بزرگ را به مجموعه ای از مسائل کوچک تقسیم کنید تا حل آن راحت‌تر شود.
- اصل ۲: درک بکارگیری انتزاع ها؛ انتزاع شکل ساده از از یک سیستم پیچیده است.
- اصل ۳: تلاش برای سازگاری؛ مثلاً سازگاری رنگ، ظاهر و گزینه‌های منو در طراحی واسط کاربر
- اصل ۴: توجه ویژه به انتقال اطلاعات؛ مثلاً از بانک اطلاعاتی به کاربر نهایی
- اصل ۵: توسعه نرم افزاری که ساختار پیمانه‌ای اثربخش داشته باشد؛ هر سیستم پیچیده را به چند پیمانه به طور اثربخش طراحی کنید.
- اصل ۶: جستجو به دنبال الگوها؛
- اصل ۷: هرگاه که امکان دارد مسئله و راهکار آن را از چند دیدگاه متفاوت به نمایش بگذارید. مثلاً یک مدل از خواسته ها را می توان به بکارگیری دیدگاهی داده گرا، عملیات گرا، رفتار گرا، به نمایش گذاشت.
- اصل ۸: به خاطر داشته باشید که نرم افزار را نگهداری خواهید کرد.

۳. اصول راهنمای فعالیت های چارچوبی

۳,۱. اصول ارتباطی

- اصل ۱: گوش سپردن
- اصل ۲: خود را قبل از برقراری ارتباط آماده کنید.
- اصل ۳: یک نفر باید در جلسه ارتباطی فرایند ارتباط را مدیریت کند.
- اصل ۴: بهترین راه ارتباط رودررو است.
- اصل ۵: یادداشت بردارید و تصمیم گیری را مستند کنید.
- اصل ۶: تلاش برای همکاری
- اصل ۷: توجه خود را معطوف کنید. بحث خود را پیمانه‌ای کنید.
- اصل ۸: اگر چیزی واضح نبود یک تصویر را نمایش دهید.
- اصل ۹: الف) هنگامی که بر سر مبحثی به توافق رسیدید به مبحث دیگر بپردازید. - ب) اگر به توافق نرسیدید به مبحث دیگر بپردازید. - ج) اگر ویژگی یا قابلیتی واضح نیست و نمی توان در حال حاضر آن را واضح کرد باز هم به مبحث دیگر بپردازید.
- اصل ۱۰: مذاکره یک مسابقه یا بازی نیست، وقتی بهترین نتیجه را می دهد که هر دو طرف برنده باشند.

۳,۲. اصول برنامه ریزی

- اصل ۱: شناخت حوزه پروژه؛ اگر ندانید مقصد کجاست استفاده از نقشه راه غیرممکن است.
- اصل ۲: طرف های ذینفع را در فعالیت برنامه ریزی دخالت دهید.
- اصل ۳: این را بدانید که برنامه ریزی ماهیتی متغیر و منعطف دارد.
- اصل ۴: برآوردهای خود را بر اساس آنچه که می‌دانید و به آن اطمینان دارید انجام دهید.
- اصل ۵: همزمان با برنامه ریزی، ریسک را هم در نظر بگیرید.
- اصل ۶: واقع بین باشید. مردم هر روز صددرصد کار نمی‌کنند.

- اصل ۷: هنگام تعریف برنامه ریزی گرانولیته را تعیین کنید. منظور از گرانولیته سطحی از جزئیات است که در برنامه ریزی پروژه به آن پرداخته می شود.
- اصل ۸: تعیین کنید که چگونه می خواهید از کیفیت محصول اطمینان یابید.
- اصل ۹: چگونگی انجام دادن تغییرات را شرح دهید.
- اصل ۱۰: برنامه ریزی را به وفور پیگیری کنید و در صورت نیاز تنظیماتی به عمل آورید.

۳,۳.اصول مدل‌سازی

- اصل ۱: هدف اصلی تیم نرم افزاری ساخت نرم افزار است نه ایجاد مدل. مدل هایی را انتخاب کنید که به رساندن نرم افزار به مشتری در سریعترین زمان ممکن کمک کند.
- اصل ۲: سبک بار سفر کنید. یعنی مدل هایی بیش از نیاز خود ایجاد نکنید.
- اصل ۳: بکوشید ساده ترین مدلی را بسازید که مسئله یا نرم افزار را توصیف کند.
- اصل ۴: مدل ها را طوری بسازید که قابل تغییر باشند.
- اصل ۵: توانایی بیان صریح هدف هر مدل ایجاد شده را داشته باشید.
- اصل ۶: مدل هایی را توسعه دهید که با سیستم مورد نظر مطابقت دارد.
- اصل ۷: سعی کنید مدل های مفید بسازید ولی ساخت مدلهای کامل را فراموش نکنید.
- اصل ۸: در مورد قالب و نحو مدل، تعصب به خرج ندهید. اگر در انتقال مفاهیم مؤثر است نمایش در مرحله دوم اهمیت قرار دارد.
- اصل ۹: اگر غریزه شما می گوید مدلی درست نیست، هر چند که روی کار درست به نظر می رسد احتمالاً دلیلی برای این نگرانی دارید، پس به غریزه خود اطمینان کنید.
- اصل ۱۰: به محض اینکه توانستید بازخورد بگیرید.

۳,۴. اصول مدل‌سازی خواسته ها

- اصل ۱: دامنه اطلاعاتی یک مسئله، باید نمایش داده و درک شود.
- اصل ۲: عملکردهای نرم افزار باید تعریف شوند.
- اصل ۳: رفتار نرم افزار باید نمایش داده شود.
- اصل ۴: مدل هایی که اطلاعات قابلیت ها و رفتارها را تصویر می کنند باید به شیوه ای تقسیم بندی شوند که جزئیات را به گونه ای لایه ای یا سلسله مراتبی نمایش دهند.
- اصل ۵: وظیفه تحلیل باید از اطلاعات ضروری به سمت جزئیات پیاده سازی حرکت کند.

۳,۵.اصول مدل‌سازی طراحی

- اصل ۱: طراحی باید تا مدل خواسته قابل ردگیری باشد.
- اصل ۲: همواره معماری سیستمی را که قرار است ساخته شود در نظر داشته باشید.
- اصل ۳: طراحی داده ها به اندازه طراحی عملکردها اهمیت دارد.
- اصل ۴: واسط ها(چه درونی و چه بیرونی) باید با احتیاط طراحی شوند.
- اصل ۵: طراحی واسط کاربر باید مطابق با نیازهای کاربر نهایی نتظیم گردد.

- اصل ۶: طراحی در سطح مؤلفه ها باید مستقل از عملکرد باشد.
- اصل ۷: ارتباط مؤلفه ها باید با یکدیگر و با محیط خارجی باید در سطحی منطقی حفظ گردد.
- اصل ۸: نمایش های(مدل های) طراحی باید به آسانی قابل درک باشند.
- اصل ۹: طراحی باید به صورت تکراری توسعه یابد، در هر دوره از تکرار طراحی باید بکوشد تا سادگی بیشتر شود.

۴. اصول ساخت

فعالیت ساخت شامل مجموعه ای از وظایف کدنویسی و آزمون می شود که نتیجه آن نرم افزاری عملیاتی و آماده تحویل به مشتری است.

۴,۱. اصول آماده سازی

پیش از آن که حتی یک خط برنامه بنویسید اطمینان حاصل کنید که:

- می دانید که چه مسئله ای را قرار است حل کنید.
- اصول و مفاهیم طراحی پایه را می دانید.
- زبانی برای برنامه نویسی انتخاب کنید که نیازهای نرم افزار و محیطی را که قرار است در آن کار کند برآورده سازد.
- محیطی برای برنامه نویسی انتخاب کنید که ابزارهای لازم برای آسان تر کردن کار را در اختیارتان قرار دهد.

۴,۲. اصول برنامه نویسی

با شروع به کد نویسی، اطمینان حاصل کنید که:

- استفاده از برنامه نویسی جفتی را در نظر داشته باشید.
- ساختمان داده هایی را انتخاب کنید که نیازهای طراحی را برآورده کند.
- معماری نرم افزار را بشناسید و واسطه هایی سازگار با آن بسازید.
- منطق شرطی را تا حد امکان ساده نگه دارید.
- حلقه های تودرتو را به شیوه ای بنویسید که به آسانی قابل آزمون باشد.
- نام های با معنا برای متغیرها انتخاب کنید.
- کدهای خود را مستند سازی کنید.
- یک چیدمان بصری ایجاد کنید. مثلاً با تورفتگی و خطوط خالی که به فهم کدهای شما کمک کند.

۴,۳. اصول اعتبار سنجی

پس از به پایان رساندن اولین دور کد نویسی، حتماً:

- گشتی در میان کدها بزنید.
- آزمون واحدها را اجرا کنید و خطاهایی را که کشف می شوند، تصحیح نمایید.
- کدها را بازآرایی کنید.

۴,۴. قواعد آزمون

- قاعده۱: آزمون فرایند اجرای برنامه به قصد یافتن خطاهاست.
- قاعده ۲: یک مورد آزمون خوب باید خطاهای کشف نشده را با احتمال زیادی کشف کند.
- قاعده ۳: آزمون موفق، آزمونی است که خطای کشف نشده تا کنون را کشف کند.

۴,۵. اصول آزمون

- اصل ۱: همه آزمون ها تا خواسته های مشتری قابل رهگیری باشند زیرا هدف از آزمون نرم افزار کشف خطاهاست.
- اصل ۲: آزمون ها را باید مدتها قبل از شروع آزمون برنامه ریزی کرد.
- اصل ۳: اصل پارتو را در آزمون نرم افزار کاربرد دارد، یعنی اثر ۸۰ درصد از همه خطاهای کشف شده طی آزمون را احتمالاً در ۲۰درصد از کل مؤلفه های نرم افزار می توان پیدا کرد.
- اصل ۴: آزمون باید در مقیاس کوچک آغاز شود و به سمت مقیاس بزرگ پیش رود.
- اصل ۵: آزمون کامل امکان پذیر نیست.

۴,۶. اصول استقرار

- اصل ۱: انتظارات مشتری برای نرم افزار باید مدیریت شود.
- اصل ۲: پکیج تحویل کامل باید مونتاژ و آزمایش شود.
- اصل ۳: پیش از تحویل نرم افزار یک روال پشتیبانی باید مشخص کرد.
- اصل ۴: مواد آموزشی مناسب باید برای کاربران نهایی تهیه شود.
- اصل ۵: نرم افزار مشکل دار، ابتدا باید اصلاح و بعداً تحویل داده شود.

جلسه ۸– شناخت خواسته ها

۱. مقدمه

۱,۱. خواسته ها چیستند؟

آن چه که مشتری می خواهد و چگونگی تعامل کاربران نهایی با نرم افزار

۱,۲. چرا اهمیت دارد؟

پیش از شروع به طراحی و ساخت یک سیستم کامپیوتری، درک و شناخت آن چه که مشتری می خواهد اهمیت دارد.

۱,۳. محصول کار چیست؟

یک گزارش مکتوب از مساله برای همه طرف ها.

۲. مهندسی خواسته ها (Requirements Engineering)

۲,۱. مهندسی خواسته ها

- طیف گسترده ای از وظایف و فنونی که به شناخت خواسته ها می انجامد.
- مهندسی خواسته ها پلی است به سوی طراحی و ساخت
- مهندسی خواسته ها بنیادی محکم برای طراحی و ساخت فراهم می سازد. نرم افزار حاصل بدون آن به احتمال زیاد نیازهای مشتری را برآورده نخواهد کرد.

۲,۲. هفت وظیفه مهندسی خواسته ها

مراحل	توضیح
شروع	○ حوزه و ماهیت مساله ای را که باید حل شود، تعیین می کند. ○ شناختی پایه ای از مساله و شناسایی راهکار مطلوب مساله ○ برقراری ارتباط میان ذی نفعان و تیم نرم افزاری
استخراج	○ به طرف های ذی نفع کمک می کند تا آن چه را که مورد نیاز است، تعریف کنند. ○ استخراج اهداف و خواسته های نرم افزار از ذی نفعان ○ مشکلات مرحله استخراج عبارتند از: ○ مشخص نشدن مرزهای سیستم و نداشتن درک کامل از دامنه مساله ○ مبهم بودن اهداف ○ عدم شناخت کامل خواسته ها و تغییر خواسته ها با گذشت زمان ○ شناخت ضعیف از قابلیت ها و محدودیت های محیط کامپیوتری خود ○ ضعف در انتقال خواسته ها به تحلیلگران

شناخت	○ اطلاعات به دست آمده از مشتری در طول مراحل شروع و استخراج، بسط داده شده و در مرحله شناخت، پالایش می یابد. ○ هدف: توسعه مدلی پالایش یافته که جنبه های گوناگون عملکرد نرم افزار، رفتار و اطلاعات آن را مشخص می سازد.
مذاکره	○ به توافق رسیدن در مورد تضادهای زیر در یک فرایند مذاکره: ○ خواسته هایی که برآوردن آن ها بر اساس محدودیت های موجود امکان پذیر نباشد. ○ خواسته های متضاد مشتریان و کاربران متفاوت ○ راه حل: ○ رتبه بندی کردن نیازهای ذی نفعان ○ مورد بحث قرار دادن تضادها و مغایرت ها بر اساس اولویت ها ○ حذف، اصلاح و تلفیق برخی از خواسته ها جهت کسب رضایت همه ذینفعان
تعیین مشخصات	○ مشخصات عبارتند از: مستندات مکتوب، مدل های گرافیکی، مدل ریاضی رسمی، مجموعه ای از سناریوهای کاربرد و نمونه اولیه نرم افزار ○ برای تعیین مشخصات یک الگوی استاندارد به کار گرفته می شود. ○ میزان رسمیت و قالب تعیین مشخصات بسته به اندازه و پیچیدگی نرم افزار، متغیر است.
اعتبار سنجی	○ ارزیابی کیفی محصولات کاری تولید شده مهندسی خواسته ها ○ بررسی مشخصات تا اطمینان حاصل شود که همه خواسته های نرم افزار بدون ابهام بیان شده اند. ○ شناسایی و تصحیح ناسازگاری ها، جا افتادگی ها و خطاها ○ چک کردن محصولات کاری از نظر مطابقت با استانداردها
مدیریت	○ مجموعه ای از فعالیت ها که تیم پروژه را در شناسایی، کنترل، پیگیری خواسته ها و تغییرات به عمل آمده در آن ها در هر زمان از پروژه یاری می دهد.

۳. تدارک مقدمات کار

۳,۱. انواع ذی نفعان

هر کسی که به طور مستقیم یا غیر مستقیم از سیستم در حال توسعه بهره مند خواهد شد.

مهندسان پشتیبانی و نگهداری	مهندسان نرم افزار	مشاوران	کاربران نهایی	مشتریان	بازاریاب ها	مدیران سازمان

۳,۲. تدارک مقدمات کار

- شناسایی طرف های ذی نفع و شناخت دیدگاه های چندگانه ذینفعان مختلف
- تلاش برای همکاری بین خواسته های مختلف و متنوع ذینفعان
- پرسیدن نخستین سوالات: درخواست کننده کار، استفاده کننده از راهکار، مزیت های اقتصادی راهکار و قید و بندهای تاثیر گذار بر راهکار

۴. استخراج خواسته ها

- شامل شناخت، مذاکره و تعیین مشخصات
- همکاری در جمع آوری خواسته ها با برگزاری جلسات بین مهندسان نرم افزار و ذینفعان

۴,۱. استقرار عملکرد کیفیت (Quality Function Deployment-QFD)

- QFD، تکنیک تضمین کیفیتی است که نیازهای مشتری را به خواسته های فنی برای نرم افزار ترجمه می کند.
- تاکید بر به حداکثر رساندن رضایت مشتری از فرایند مهندسی نرم افزار
- خواسته های مطرح شده در QFD: خواسته های عادی، خواسته های مورد انتظار و خواسته های مهیج

۴,۲. سناریو های کاربرد (Use Scenario)

Use case برای توصیفی از چگونگی به کارگیری سیستم شامل محصولات کاری استخراج (Elicitation Work Product):

نمونه اولیه نرم افزار	مجموعه ای از سناریوهای کاربرد	فهرستی از خواسته ها	توصیفی از محیط فنی سیستم	فهرست ذینفعان	بیان حوزه سیستم یا محصول	بیان خواسته ها و امکان سنجی

۵. توسعه use case

- UC رفتار سیستم را تحت شرایط گوناگون در پاسخ به درخواست های کاربر توصیف می کند.
- UC داستانی است که با سبک و سیاق درباره چگونگی تعامل کاربر نهایی با سیستم تحت مجموعه شرایط معین.
- کنش گر هر چیزی است که با سیستم یا محصول ارتباط برقرار می کند و خارج از خود سیستم قرار دارد.
- کنش گر و کاربر نهایی الزما یکسان نیستند. یک کاربر ممکن است هنگام استفاده از سیستم چند نقش داشته باشد در حالی که کنش گر نماینده طبقه ای از موجودیت های خارجی است که فقط یک نقش دارند.

۵,۱. گام های توسعه use case

- تعریف مجموعهای از کنش گران که در داستان مشارکت دارند.
- تعیین اهداف کنش گران
- پیش شرط های مورد کاربرد
- استثنائات
- تغییرات در تعامل کنش گران
- اطلاعاتی که کنش گران به دست می آورند.

۶. ساخت مدل های خواسته ها (مدل تحلیل)

۶,۱. عناصر مدل خواسته ها

هدف: استفاده از چند حالت متفاوت نمایشی برای به تصویر کشیدن مدل خواسته ها برای نگریستن به مدل خواسته ها از دیدگاه های متفاوت

مراحل	توضیح
عناصر مبتنی بر سناریو	○ توصیف سیستم از دیدگاه کاربر با استفاده از رویکردی مبتنی بر سناریو ○ UC Diagram
عناصر مبتنی بر کلاس	○ هر سناریوی کاربرد نشان گر مجموعه ای از اشیاء است که با تعامل یک کنش گر با سیستم، دستکاری می شوند. ○ طبقه بندی این اشیا در قالب چند کلاس ○ Class Diagram
عناصر رفتاری	○ ترسیم رفتار یک سیستم ○ نمودار حالت: تصویر کشیدن حالت ها و رویدادهایی که باعث می شوند سیستم تغییر حالت دهد. ○ حالت: به هر شیوه رفتاری سیستم گفته می شود که ار بیرون قابل مشاهده باشد.
عناصر جریانگرا	○ انتقال اطلاعات با جریان یافتن در یک سیستم کامپیوتری صورت می پذیرد. ○ ورودی – پردازش - خروجی

۶,۲. الگوهای تحلیل

- الگوهای تحلیل راهکارهایی در یک دامنه کاربردی پیشنهاد می کند که در مدلسازی بسیاری از برنامه های کاربردی قابل استفاده مجدد است.

۷. مذاکره بر سر خواسته ها

هدف مذاکره نتیجه برد-برد شامل فعالیت های:

- شناسایی ذینفعان مهم
- تعیین شرایط برد برای ذینفعان
- مذاکره بر سر شرایط برد ذینفعان برای رساندن آن ها به شرایط برد- برد.

۸. اعتبار سنجی خواسته ها

- همخوانی خواسته ها با اهداف کلی نرم افزار
- لازم بودن یا اضافی بودن خواسته ها
- خالی بودن خواسته ها از ابهام
- تضاد یا هماهنگی خواسته ها با یکدیگر
- استفاده از الگوهای خواسته ها برای ساده سازی مدا خواسته ها

جلسه ۹– مدل سازی خواسته ها:سناریوها، اطلاعات و کلاس های تحلیل

۱. مقدمه

۱,۱. مدل سازی خواسته ها چیستند؟

در مدل سازی خواسته ها تلفیقی از شکل های متنی و نموداری برای به تصویر کشیدن خواسته ها استفاده می شود به شیوه ای که درک آن آسان تر باشد.

۱,۲. چرا اهمیت دارد؟

چون هر کدام از مدل ها خواسته ها را از بعدی متفاوت به نمایش در می آورند.

۲. تحلیل خواسته ها

۲,۱. تحلیل خواسته ها

- تحلیل خواسته ها به تعیین مشخصات خصوصیات عملیاتی نرم افزار منجر می شود.
- واسط نرم افزار با سایر عناصر سیستم را مشخص می کند.
- قید و بندهایی را که نرم افزار باید رعایت کند، تعیین می نماید.

۲,۲. انواع مدل ها

مدل	توضیح
مدل های مبتنی برسناریو	○ سیستم را از دیدگاه کاربر به نمایش می گذارند.
مدل سازی داده ها	○ فضای اطلاعاتی و اشیاء داده و روابط بین آنها را به نمایش می گذارد.
مدل های مبتنی بر کلاس	○ به تعریف اشیاء صفات و روابط می پردازد.
مدل های جریان گرا	○ که عناصر عملیاتی سیستم و چگونگی تبدیل داده ها توسط عناصر را به هنگام حرکت در سیستم نمایش می دهد.
مدل های رفتاری	○ چگونگی رفتار نرم افزار را به عنوان نتیجه ای از رویداد های بیرونی به تصویر می کشند.

۲,۳. اهداف مدل خواسته ها

- توصیف انچه که مشتری نیاز دارد.
- ایجاد مبنایی برای تهیه طراحی نرم افزار
- تعریف مجموعه ای از خواسته ها که پس از ساخته شدن نرم افزار بتوان آنها را اعتبار سنجی کرد.
- نکته: مدل خواسته ها به عنوان پلی میان توصیف سیستم و مدل طراحی

۲,۴. قواعد تحلیل

- مدل نباید خود را درگیر جزئیات کند. و باید سعی کند چگونگی کار کردن سیستم را توضح دهد.
- هر عنصر از مدل خواسته ها باید در کل چیزی به درک ما از خواسته های نرم افزار بیافزاید و دیدی از دامنه اطلاعاتی، عملکرد و رفتار سیستم فراهم سازد.
- جلب رضایت همه ی ذی نفعان.
- به حداقل رسانی ارتباطات میان مولفه ها در سرتاسر سیستم
- حفظ سادگی مدل تا حد امکان

۲,۵. تحلیل دامنه

- عبارت است از شناسایی، تحلیل، و تعیین مشخصات خواسته های رایج از یک دامنه ی کاربردی خاص معمولا برای استفاده مجدد در چندین پروژه که در همان دامنه کاربردی قراردارند.
- مانند: هوانوردی، بانکداری، بازی و پزشکی.

۲,۶. عناصر مدل تحلیل

مدل	توضیح
مدل های کلاس ها	○ نمودارهای کلاس ها - نمودارهای همکاری
مدل های مبتنی بر سناریو	○ مورد کاربردها - سناریوهای کاربردی
مدل های جریان گرا	○ DFD ها - مدل های داده ای
مدل های رفتاری	○ نمودارهای حالت - نمودارهای ترتیبی

۳. مدل سازی مبتنی بر سناریو

۳,۱. ایجاد یک مورد کاربرد مقدماتی

- مورد کاربرد شیوه استفاده یک کنش گر از نرم افزار برای رسیدن به هدفی مشخص است.
- UC ها صرفا به تعریف آنچه که در بیرون سیستم قرار دارد (کنش گر) و آنچه که باید توسط سیستم انجام شود کمک می کند.

۳,۲. پالایش یک مورد کاربرد مقدماتی

- آیا کنش گر در این نقطه کنش دیگری انجام می دهد؟
- آیا امکان خطا برای کنش گر در این نقطه وجود دارد؟
- آیا این امکان وجود دارد که کنش گر در این نقطه با رفتار دیگری مواجه گردد؟

۴. مدل های UML که مورد کاربرد را تکمیل می کنند.

۴,۱. نمودار فعالیت

- نمودار فعالیت کنش ها و تصمیم گیری هایی را که در اجرای یک عملکرد رخ می دهند را به نمایش می گذارد.

- مستطیل: نشان دادن عملکرد سیستم
- پیکان: نمایش جریان در سیستم
- لوزی: تصمیم گیری
- خطوط افقی تو پر: فعالیت های موازی

۴,۲. نمودارهای بخش بندی

- جریان کنش ها و تصمیم گیری ها را به نمایش می گذارد و نشان می دهد کدام کنش گران کدام کنش را انجام دهند.
- نمودار بخش بندی شکل دیگری از نمودار فعالیتهاست که به کمک آن می توان جریانِ فعالیت های توصیف شده در یک UC را به نمایش در آورد.
- و در عین حال نشان داد که کدام کنش گر یا کلاس تحلیل مسئولیت کنشِ توصیف شده توسط یک فعالیت است.

۵. مفاهیم مدل سازی دادها

۵,۱. مفاهیم مدل سازی داده ها

مفاهیم	توضیح
اشیاء داده	○ یک شی ءداده نمایشی است از هر گونه اطلاعات مرکب که توسط نرم افزار پردازش می شود. ○ منظور از اطلاعات مرکب چیزی است که دارای چند صفت یا خاصیت متفاوت باشد. ○ مثلا: پهنا که تنها یک مقدار دارد، شی ء داده معتبری نیست ولی بعد که شامل طول عرض و ارتفاع می شود به عنوان یک شی ء قابل تعریف است.
صفات داده ها	○ خواص یک شی ء داده را تعریف می کند. ○ صفات، یک شی ء داده ای را نام می برند، خصوصیات آن را توصیف می کنند و در برخی موارد، به شی ء دیگر ارجاع می دهند.
ارتباطات	○ اشیاء داده را به هم متصل می کند. ○ رابطه ها نشان گرِ شیوه ی اتصال اشیاء داده به یکدیگرند.

۶. مدل سازی مبتنی بر کلاس ها

۶,۱. مدل سازی مبتنی بر کلاس ها

- اشیایی را که سیستم دستکاری می کند.
- عملیاتی که در مورد اشیاء به کار می رود تا این دستکاری ها انجام شود
- روابط میان اشیاء
- همکاریهایی که بین کلاس ها رخ می دهد.

۶,۲. عناصر یک مدل مبتنی بر کلاس ها

- کلاس ها و اشیاء

- صفات
- عملیات
- مدل های همکاری- مسئولیت کلاس ها (CRC)
- نمودارهای همکاری
- پکیج ها

۶,۳. شناسایی کلاس های تحلیل

- مسئله ی واقعا دشوار، کشف کلاس های درست در وهله نخست است.
- می توانیم شناسایی کلاس ها را بررسی سناریوهای کاربری آغاز کنیم و مورد کاربرد ها را تجزیه کنیم.
- کلاس ها با خط کشیدن زیر عبارت های اسمی و وارد کردن آنها در یک جدول تعیین می شوند.
- خصوصیات انتخابی که در پرداختن به هر کدام از کلاس های بالقوه برای لحاظ کردن در مدل تحلیل باید از آن ها استفاده کرد.

مفاهیم	توضیح
اطلاعات نگهداری شده	○ به خاطر سپردن اطلاعات مربوط به کلاس بالقوه برای عملکرد سیستم ضروری باشد.
سرویس های لازم	○ کلاس بالقوه باید دارای مجموعه ای از عملیات قابل شناسایی باشد که بتوانند مقدار صفات آن را به طریقی تغییر دهند.
صفات چندگانه	○ کلاس با یک صفت بهتر است در قالب صفتی از یک کلاس دیگر نمایش داده شود.
اطلاعات نگهداری شده	○ به خاطر سپردن اطلاعات مربوط به کلاس بالقوه برای عملکرد سیستم ضروری باشد.
سرویس های لازم	○ کلاس بالقوه باید دارای مجموعه ای از عملیات قابل شناسایی باشد که بتوانند مقدار صفات آن را به طریقی تغییر دهند.
صفات چندگانه	○ کلاس با یک صفت بهتر است در قالب صفتی از یک کلاس دیگر نمایش داده شود.

۶,۴. مشخص کردن صفات

- صفات، کلاس انتخاب شده برای گنجاندن در مدل خواسته ها را توصیف می کند . یعنی کلاس را تعریف می کنند.
- صفات، مجموعه ای از اشیاء داده هستند که یک کلاسرا به طور کامل در حیطه ی مسئله تعریف می کنند.
- برای توسعه مجموعه ای با معنا از صفات برای یک کلاس تحلیل باید هرکدام از UCها را مطالعه کنید و آن چیزهایی را انتخاب کنید که به طور منطقی به کلاس تعلق دارند.

۶,۵. تعریف عملیات ها

عملیات ها رفتار شیء را تعریف می کنند.

- عملیاتی که داده هارا به طریقی دستکاری می کنند: اضافه و حذف کردن
- عملیاتی که محاسبه انجام می دهند.
- عملیاتی که در باره ی حالت یک شی ء تحقیق می کنند .
- عملیاتی که یک شی را برای وقوع یک رویداد کنترل کننده پایش می کنند.

برای تعریف عملیات ها تجزیه گرامری دوباره مطالعه می شود و این بار افعال جداسازی می شوند.

۶,۶. مدل سازی همکار-مسئولیت کلاس ها CRC

- CRC ابزاری ساده برای شناسایی و سازماندهی کلاس های مرتبط با خواسته های سیستم یا محصول را فراهم می سازد.
- CRC مجموعه ای از کارتهای شاخص استاندارد است که کلاس ها را به نمایش می گذارند.
- این کلاس ها به سه بخش تقسیم می شوند:
- در بالای کارت نام کلاس را می نویسید.
- در بدنه کارت فهرست مسئولیت های کلاس را در طرف راست و همکاران را در طرف چپ می نویسید.
- این کارت ها میتوانند واقعی یا مجازی باشند.
- مسئولیت ها صفات و عملیات مرتبط با کلاس هستند.
- همکاران کلاس هایی هستند که برای فراهم ساختن اطلاعات لازم برای کامل شدن یک مسئولیت توسط یک کلاس مورد نیاز هستند.
- هر همکاری یا به معنای درخواست برای اطلاعات یا تقاضای کنش است.

۶,۷. روابط بین کلاس ها

- رابطه شمول
- رابطه آگاهی داشتن از ...
- رابطه بستگی داشتن به ...
- همه کلاس هایی که بخشی از یک کلاس مجتمع هستند، از طریق رابطه شمول به آن کلاس مجتمع متصل اندمثلا: کلاس بدن بازیکن بخشی از بازیکن است/مثال ۱۹۳
- هنگامی که یک کلاس باید اطلاعات را از کلاس دیگری کسب کند رابطه اگاهی داشتن از... برقرار می شود.
- رابطه بستگی داشتن به ... به این معناست که دو کلاس دارای ارتباطی به جز آگاهی داشتن از ... و شمول هستند.

۶,۸. اجتماع و وابستگی

- در بسیاری از موارد دو کلاس به شیوه ای مشابه با ارتباط دو شی ء داده ای با هم ارتباط دارند.
- اجتماع رابطه میان کلاس هارا مشخص می کند.
- چندگانگی مشخص می کند که چند کلاس با چند کلاس دیگر ارتباط دارند.
- در بسیاری از موارد میان دو کلاس تحلیل یک رابطه کلاینت –سرور وجود دارد.
- در این گونه موارد کلاس کلاینت به نحوی به کلاس سرور وابسته است و یک رابطه وابستگی برقرار می شود.

۶,۹. پکیج های تحلیل

- بخش مهمی از مدل سازی تحلیل گروه بندی است.
- یعنی عناصر گوناگون مدل تحلیل مثل مورد کاربردها و کلاس های تحلیل طوری گروه بندی میشوند که یک پکیج از آن ها تشکیل شود.

جلسه ۱۰ – مدل سازی خواسته ها:جریان، رفتار، الگوها و برنامه های تحت وب بخش ۱

۱. راهبردهای مدل سازی خواسته ها

۱,۱. تحلیل ساخت یافته

- داده ها و فرایندهایی که این داده ها را تبدیل می کنند، موجودیت های مجزا در نظر گرفته می شوند.
- اشیای داده ای به شیوه ای مدل سازی می شوند که صفات و روابط میان آن ها را تعریف کند.
- فرایندهایی که اشیای داده ای را دستکاری می کنند، به شیوه ای مدل سازی می شوند که چگونگی تبدیل اشیای داده ای را به هنگام جریان یافتن آن ها در سیستم نشان دهند.

۱,۲. تحلیل شی گرا

- بر تعریف کلاس ها و شیوه همکاری آن ها با یکدیگر برای برآورده ساختن خواسته های مشتری تاکید دارد.

۲. مدل سازی جریان گرا

۲,۱. نمودار جریان داده ها DFD

- DFD و نمودارهای مرتبط با آن بخش رسمی از UML به شمار نمی روند ولی می توان از آن ها برای تکمیل نمودارهای UML بهره برد و دیدی اضافه از خواسته ها و جریان داده ها به دست آورد.
- DFD نمایی از سیستم بر اساس ورودی – فرایند- خروجی به دست می دهد. یعنی اشیاء داده ای به درون نرم افزار جریان پیدا می کنند، توسط عناصر پردازشی تبدیل می شوند و اشیاء داده ای حاصل به بیرون نرم افزار جریان پیدا می کنند.

۲,۲. نمایش نمودارهای DFD

- اشیاء داده ای با پیکان های برچسب دار و تبدیلات با دایره
- DFD به شیوه ای سلسله مراتبی نمایش داده می شوند، یعنی اولین مدل در جریان داده ها که سطح صفر نامیده می شود، کل سیستم را نمایش می دهد و نمودارهای بعدی در هر سطح بعدی جزئیات بیشتری را نشان می دهند.

۲,۳. ایجاد مدل جریان داده ها

- با پالایش DFD به سطوح بالاتری از جزئیات، می توانیم سیستم را به طور ضمنی از نظر عملیاتی تجزیه کنیم.
- هدف نمودارهای جریان داده ها، فراهم ساختن یک پل معنایی میان کاربران و سازندگان سیستم است.

۲,۴. ایجاد مدل جریان کنترل

- برای برخی انواع برنامه های کاربردی مدل داده ها و نمودار جریان داده ها همه آن چیزی است که برای به دست آوردن دیدی مناسب از خواسته های نرم افزار لازم است.

- ولی گروه بزرگی از برنامه های کاربردی بیشتر توسط رویدادها اداره می شوند تا داده ها و بیشتر اطلاعات کنترلی تولید می کنند تا گزارش و چیزهایی برای نمایش.
- این گونه برنامه ها علاوه بر مدل سازی جریان داده ها به مدل سازی جریان کنترل هم نیازدارند.
- یک آیتم کنترلی یا رویدادی به صورت مقداری بولی، یا شرطی پیاده سازی می شود.
- برای ایجاد مدل جریان کنترل، رفتار سیستم را با شناسایی حالت ها، چگونگی رسیدن به هر حالت و تعیین گذارهای میان حالت ها توصیف کنید.

۲,۵. مشخصات کنترل (CSPEC)

رفتار سیستم را به دوشیوه نمایش می گذارد.

- حاوی یک نمودار حالت است که رفتار را به صورت ترتیبی مشخص می کند. این نمودار چگونگی پاسخ دهی سیستم به رویدادها را به هنگام عبور از حالت های سیستم نشان می دهد.
- حاوی یک جدول فعال سازی فرایندها (PAT) (مشخصات ترکیبی رفتار) است. این نمودار اطلاعات موجود در نمودار حالت را در حیطه فرایندها و نه حالت ها، نشان می دهد.
- CSPEC، رفتار سیستم را توصیف می کند، ولی درباره کارکرد داخلی فرایندهایی که در نتیجه این رفتار فعال می شوند، هیچ اطلاعاتی نمی دهد.

۲,۶. مشخصات فرایندها (PSPEC)

- در توصیف همه فرایندهای مدل جریان که در سطح نهایی پالایش ظاهر می شوند کاربرد دارد.
- محتوای تعیین مشخصات فرایندها می تواند شامل متن روایی، توصیفی از زبان طراحی برنامه برای الگوریتم فرآیند، معادلات ریاضی ، جداول یا نمودار های فعالیت های UML باشد.
- با فراهم ساختن یک PSPEC برای همراه کردن با هر حباب در مدل جریان، می توانید مجموعه ای از ریز مشخصاتی ایجاد کنید که برای طراحی مولفه ای از نرم افزار، که آن حباب را پیاده سازی می کند، به عنوان دستورالعمل به کار می رود.

۳. ایجاد مدل رفتاری

۳,۱. ایجاد مدل رفتاری

- مدل های قبلی، نشان دهنده عناصر ایستای مدل خواسته ها
- مدل رفتاری نشان دهنده رفتار پویای سیستم، نمایش رفتار سیستم به صورت تابعی از زمان و رویدادهای مشخص و نمایش پاسخ نرم افزار به رویدادها و محرک های خارجی.

۳,۲. شناسایی رویدادها به کمک مورد کاربرد

- مورد کاربرد: نمایش دنباله ای از فعالیت ها که کنش گر و سیستم را شامل می شوند.
- هرگاه سیستم و کنش گری به تبادل اطلاعات بپردازند، یک رویداد رخ می دهد.

۳,۳. نمودارهای حالت

حالت هر کلاس در زمانی که به وظیفه خود عمل می کند.

- حالت انفعالی: حالت فعلی همه صفت های یک شی است.
- حالت فعال: وضعیت فعلی شی را به هنگام قرار گرفتن در معرض تبدیل یا پردازش مستمر نشان می دهد.

نمودار حالت: یک مولفه از مدل رفتاری که حالت های فعال هر کلاس و رویدادهایی را نشان می دهد که باعث تغییر در این حالت های فعال می شوند.

۳,۴. نمودارهای ترتیب

- یک مولفه از مدل رفتاری که نشان می دهد رویدادها چگونه باعث گذار یک شی به شی دیگر به عنوان تابعی از زمان می شوند.
- نشان دهنده کلاس های کلیدی و رویدادهایی که باعث جریان یافتن رفتار از کلاسی به کلاس دیگر می شوند.

۴. الگوهایی برای مدل سازی خواسته ها

۴,۱. الگوهایی برای مدل سازی خواسته ها

- سازوکارهایی هستند که هنگام مواجهه با مساله ای جدید، می توان آن ها را دوباره به کار برد.
- ذخیره الگوهای تحلیل در یک مخزن، به طوری که اعضای تیم بتوانند با جستجو، آن ها را یافته و دوباره استفاده کنند.

جلسه ۱۱ – مدل سازی خواسته ها:جریان، رفتار، الگوها و برنامه های تحت وب بخش ۲

۱. مدل سازی خواسته ها برای برنامه های تحت وب

۱,۱. مدل سازی خواسته ها برای برنامه های تحت وب

- کسانی که در وب کار می کنند، غالبا به تحلیل خواسته ها برای برنامه های تحت وب، با دیده تردید می نگرند و استدلال آن ها این است که فرایند کار در وب باید چابک باشد و تحلیل، کاری است که زمان می برد.
- تحلیل خواسته زمان می برد، اما حل مسئله اشتباه از آن هم بیشتر وقت می برد.

۱,۲. چقدر تحلیل کافی است؟

عوامل موثر بر میزان نیاز برنامه های تحت وب به تحلیل و مدل سازی:

- اندازه و پیچیدگی برنامه تحت وب
- تعداد طرف های ذینفع
- اندازه تیم برنامه نویسی
- میزان همکاری قبلی اعضای تیم برنامه نویسی
- میزان حیاتی بودن برنامه

با کوچکتر شدن پروژه، کم شدن تعداد ذینفعان، یکپارچگی بیشتر تیم توسعه و حیاتی نبودن پروژه، بهتر است تحلیل کمتری صورت بگیرد.

۱,۳. ورودی در مدل سازی خواسته ها

در مهندسی برنامه های تحت وب می توان از روش چابک استفاده کرد. اطلاعاتی که طی فعالیت برقراری ارتباط به دست می آیند. شامل هر چیزی از نامه های الکترونیکی غیر رسمی گرفته تا یک خلاصه پروژه مشروح با سناریوهای کاربرد جامع و مشخصات کامل محصول. ورودی ها عبارتند از:

- گروه های ذینفع و کاربر
- حیطه تجاری
- اهداف اطلاعاتی و کاربردی تعیین شده
- خواسته های عمومی برنامه های تحت وب
- سناریوهای کاربرد

۱,۴. خروجی های مدل سازی خواسته ها

تحلیل خواسته ها یک سازوکار منضبط برای موارد زیر فراهم می آورد:

- ارائه و ارزیابی محتوا و قابلیت های عملیاتی برنامه های تحت وب

- شیوه های تعامل فراروی کاربر و محیط
- زیرساخت قرار گرفتن برنامه های تحت وب

مدل	توضیح
مدل محتوا	○ محتوای متنی، گرافیکی، ویدیویی و صوتی برنامه تحت وب.
مدل تعامل ها	○ شیوه تعامل کاربران با برنامه تحت وب
مدل عملیاتی	○ عملیاتی که روی محتوای برنامه تحت وب انجام می شود و قابلیت های پردازشی مستقل از محتوا و ضروری برای کاربر را توصیف می کنند.
مدل گشت و گذار	○ راهبرد کلی گشت و گذار برنامه تحت وب
مدل پیکربندی	○ محیط و زیرساخت برنامه تحت وب

۱,۵. مدل محتوا برای برنامه های تحت وب

- حاوی عناصری ساختاری که دیدی مهم از خواسته های محتوایی برنامه تحت وب در اختیار قرار می دهد.
- شامل اشیای محتوایی و همه کلاس های تحلیل. اشیای محتوایی: متن، عکس، پویانمایی، ویدیو، صوت و ...
- اشیای محتوایی را می توان به طور مستقیم از UC و با بررسی سناریو برای ارجاعات مستقیم و غیر مستقیم به محتوا تعیین کرد.
- امکان گسترش محتوا، قبل از پیاده سازی برنامه های تحت وب، یا حین ساخته شدن، یا پس از عملیاتی شدن برنامه
- محتوا از طریق مرجع گشت و گذار در ساختار کلی برنامه های تحت وب گنجانده می شود.
- مثال هایی از اشیای محتوایی:

توضیح رویداد خبری	عکسی از یک رویداد ورزشی	پاسخی به سوال یک کاربر	نمایش پویا نمایی شده از لوگوی یک شرکت	یک قطعه ویدئو	صداگذاری

- برای هر محتوایی که از چند شیء محتوایی و آیتم داده ای تشکیل می شود، می توان یک درخت داده ایجاد کرد.
- درخت داده ها برای تعریف روابط سلسله مراتبی میان اشیاء محتوایی و فراهم ساختن ابزاری برای مرور محتوا توسعه می یابد.
- به طوری که ناسازگاری ها و جا افتادگی ها قبل از شروع طراحی کشف شوند.

۱,۶. مدل تعامل برای برنامه های تحت وب

- گفتگو میان کاربر نهایی و قابلیت عملیاتی، محتوا یا رفتار برنامه
- عناصر مدل تعامل: UC، نمودارهای ترتیب، نمودارهای حالت و نمونه های اولیه واسط کاربری
- در بسیاری از نمونه ها از UC برای توصیف تعامل در سطح تحلیلی استفاده می شود.
- اگر تعامل پیچیده و شامل کلاس چندگانه باشد، می توان از نمودار استفاده کرد.
- از آن جا که ابزارهای ساخت برنامه های تحت وب، بسیار زیاد و ارزان و دارای قدرت عملیاتی بالا هستند، بهترین کار، ایجاد نمونه اولیه واسط با استفاده از این گونه ابزارهاست.
- در نمونه اولیه باید پیوندهای اصلی مربوط به گشت و گذار دربرنامه پیاده سازی شود و چیدمان کلی صفحه به همان صورت که قرار است ساخته شود، به نمایش درآید.

۱,۷. مدل عملیاتی برای برنامه های تحت وب

ارائه گستره وسیعی از قابلیت های محاسباتی و دستکاری داده ها که انواع آن عبارتند از:

- قابلیت های عملیاتی که توسط برنامه به کاربر نهایی ارایه می شوند و او قادر به دیدن آن هاست.
- عملیاتی که در داخل کلاس های تحلیل قرار دارند و رفتارهای مرتبط با هر کلاس را پیاده سازی می کنند.
- استفاده از نمودار فعالیت

۱,۸. مدل پیکر بندی برای برنامه های تحت وب

- فهرست صفات کلاینت و سرور
- پیکربندی های پیچیده شامل: توزیع بار میان چند سرور، بانک های اطلاعاتی از راه دور و سرورهای چندگانه
- استفاده از نمودار استقرار UML

۱,۹. مدل گشت و گذار برای برنامه های تحت وب

چگونگی سیاحت گروه های کاربری از یک عنصر برنامه به عنصر دیگر. باید خواسته های کلی گشت و گذار را مورد توجه قرار دهیم. در همین راستا باید سوالاتی از خود بپرسیم:

- اولویت ارایه عناصر گشت وگذار
- تاکید بر عناصر خاص گشت و گذار
- چگونگی برخورد با خطاهای گشت و گذار
- نحوه انجام گشت و گذار (پیوند، جستجو یا ...)
- در دسترس بودن یک منو یا نقشه گشت و گذار کامل در مقابل پیوند ساده بازگشت یا اشاره گر جهت دار
- امکان ضبط گشت و گذار توسط کاربر برای استفاده در آینده

جلسه ۱۲ – مفاهیم طراحی

۱. مقدمه

۱,۱. طراحی چیست؟

- طراحی، نمایش یا مدلی از نرم افزار ایجاد می کند.
- ولی بر خلاف مدل خواسته ها، مدل طراحی جزئیات مربوط به معماری نرم افزار، ساختمان داده ها، واسط ها و مولفه های لازم برای پیاده سازی سیستم را فراهم می کند.

۱,۲. مراحل کار کدام است؟

- نمایش معماری سیستم یا محصول
- مدل سازی واسط هایی که نرم افزار را به کاربران نهایی، به سایر سیستم ها و دستگاه ها و مولفه های سازنده خودش مرتبط می کند.
- مدل سازی مولفه های نرم افزار

۲. طراحی در حیطه مهندسی نرم افزار

۲,۱. طراحی در حیطه مهندسی نرم افزار

- طراحی نرم افزار پس از تحلیل و مدل خواسته ها آغاز می شود.
- آخرین کنش در فعالیت مدل سازی است، که نرم افزار را برای ساخت آماده می کند.
- اهمیت طراحی در آن است که طراحی جایی است که در آن کیفیت نرم افزار تثبیت می شود.
- برگردان مدل خواسته ها به مدل طراحی

۲,۲. روش های طراحی

- طراحی داده های کلاس ها
- طراحی معماری
- طراحی واسط
- طراحی مولفه ها

۳. فرایند طراحی

۳,۱. فرایند طراحی

- طراحی نرم افزار، فرایندی مبتنی بر تکرار است.

- که از طریق آن خواسته ها به نقشه ای برای ساخت نرم افزار ترجمه می شوند.
- در آغاز، طراحی در سطح بالایی از انتزاع نمایش داده می شود.
- با تکرار شدن طراحی، پالایش های بعدی به نمایش های طراحی در سطوح پایین تر منجر می شود.

۳,۲. دستورالعمل های کیفیت نرم افزار

- به منظور تعیین کیفیت نمایش طراحی باید ملاک هایی فنی برای طراحی خوب وضع کنید.
- طراحی باید معماری ای را نشان دهد که با استفاده از سبک ها یا الگوهای معماری شناخته شده ایجاد شده باشند.
- طراحی باید پیمانه بندی شده باشد.
- طراحی باید حاوی نمایش های متمایزی از داده ها، معماری، واسط ها و مولفه ها باشد.
- طراحی باید به ساختمان داده ای منجر گردد که برای کلاس هایی که قرار است پیاده سازی شوند و از الگوهای داده ای قابل تشخیص بیرون کشیده می شوند، مناسب باشند.
- طراحی باید با بکارگیری نمادهایی ارائه شود که معنا و مفهوم را به خوبی برساند.

۳,۳. صفات کیفیت نرم افزار

صفات	توضیح
قابلیت عملیاتی (Functionality)	○ با تعیین مجموعه ویژگی ها و قابلیت های برنامه، عملیات کلی که تحویل می شوند و امنیت کل سیستم ارزیابی می شود.
قابلیت کاربرد (Usability)	○ با اندازه گیری عوامل انسانی، زیبایی شناسی کلی، سازگاری و مستندسازی سنجیده می شود.
قابلیت اطمینان (Reliability)	○ با اندازه گیری فراوانی و شدت شکست ها، صحت نتایج خروجی، میانگین زمان شکست، توانایی خلاصی یافتن از شکست و قابلیت پیش بینی برنامه تعیین می شود.
قابلیت پشتیبانی (Supportability)	○ ترکیبی است از توان بسط برنامه، قابلیت انطباق، قابلیت سرویس، قابلیت آزمایش، سازگاری، قابلیت پیکربندی، سهولت نصب سیستم و سهولت پیدا کردن مسائل.

۴. مفاهیم طراحی

۴,۱. انتزاع (Abstraction)

- هنگامی که راهکاری پیمانه ای برای مساله در نظر می گیرید، سطوح انتزاع متعددی ممکن است پیش آید.
- در بالاترین سطح انتزاع، راهکار در قالب عبارت های کلی بیان می شوند.
- در سطوح پایین تر انتزاع، توصیف مشروح تری از راهکار ارائه می شود.
- انتزاع فرایندی: دستورالعمل هایی که وظیفه ای مشخص و محدود دارند و جزئیات خاصی کنار گذاشته می شوند.
- انتزاع داده ای: مجموعه ای از داده ها با نام مشخص است که شی داده ای را توصیف می کند.

۴,۲. معماری (Architecture)

- معماری نرم افزار به ساختار کلی نرم افزار و شیوه هایی مربوط می شوند که این ساختار باعث یکپارچگی مفهومی در سیستم می گردد.
- یکی از اهداف مهندسی نرم افزار، به دست آوردن یک نمای معماری از سیستم است.
- این نما به عنوان چارچوبی عمل می کند که از طریق آن فعالیت های مشروح تر طراحی اجرا می شوند.

۴,۳. الگوها (Patterns)

- الگو عبارت است از کتیبه ای دارای نام که حامل جوهره راهکار اثبات شده برای مساله ای تکراری در حیطه معین و در میان دغدغه های گوناگون است.
- الگوی طراحی توصیفی است از یک ساختار طراحی که یک مساله طراحی را در حیطه ای خاص حل می کند.

۴,۴. جداسازی دغدغه ها (Separation of Concern)

- یک مفهوم طراحی است که پیشنهاد می کند هر مساله پیچیده ای را می توان بهتر حل کرد اگر به قطعاتی تقسیم گردد که هر یک را بتوان به طور مستقل، حل کرد.
- هر دغدغه، ویژگی یا رفتاری است که به عنوان بخشی از مدل خواسته ها برای نرم افزار مشخص می شود.
- با جداسازی دغدغه ها به قطعات کوچکتر، زمان و تلاش کمتری صرف حل مساله می شود.

۴,۵. پیمانه بندی (Modularity)

- پیمانه بندی متداول ترین نمود جداسازی دغدغه هاست.
- نرم افزار به مولفه های جداگانه و دارای نام تقسیم می شود که به آن پیمانه گفته می شود.
- از انسجام این پیمانه ها، خواسته های مساله برآورده می شود.

۴,۶. پنهان سازی اطلاعات (Information Hiding)

- از اصل پنهان سازی اطلاعات چنین بر می آید که پیمانه ها باید با تصمیم گیری های طراحی مشخص شوند که هر کدام از دید بقیه پنهان هستند.
- پیمانه ها باید طوری مشخص و طراحی شوند که اطلاعات (الگوریتم ها و داده های موجود) در یک پیمانه نتواند در دسترس پیمانه های دیگری قرار گیرد که به این اطلاعات نیاز ندارند.

۴,۷. استقلال عملیاتی (Functional Independence)

- مفهوم استقلال عملیاتی، نتیجه مستقیم جداسازی دغدغه ها، پیمانه بندی و مفاهیم پنهان سازی اطلاعات و انتزاع است.
- استقلال عملیاتی با توسعه پیمانه هایی که با عملکرد یگانه و دوری جستن از تعامل بیش از حد با سایر پیمانه ها به دست می آید.

۴,۸. پالایش (Refinement)

- پالایش مرحله ای یک راهبرد طراحی از بالا به پایین است.

- برنامه با سطوح پالایش پیاپی از جزئیات روالی توسعه داده می شود.
- یک سلسله مراتب، با تجزیه بیان ماکروسکوپی قابلیت عملیاتی به شیوه ای مرحله ای توسعه می یابد تا این که به دستورات زبان برنامه نویسی برسیم.
- پالایش در واقع همان فرایند تعیین جزئیات است.

۴,۹. بازآرایی (Refactoring)

- بازآرایی تکنیکی برای سازمان دهی مجدد به شمار می رود و طراحی یک مولفه را ساده می کند.
- بدون این که قابلیت عملیاتی رفتار آن را تغییر دهد.
- بازآرایی عبارت است از فرایند تغییر دادن سیستم نرم افزاری به گونه ای که رفتار خارجی کد (طراحی) تغییر نکند.
- و در عین حال، ساختار درونی آن بهبود یابد.

۵. مدل طراحی

۵,۱. مدل طراحی از دو بعد متفاوت

بعد	توضیح
بعد فرایندی	○ تکامل مدل طراحی را به موازات اجرای وظایف طراحی به عنوان بخشی از فرایند نرم افزار نشان می دهد.
بعد انتزاعی	○ سطح جزئیات را به موازات تبدیل هر عنصر از مدل تحلیل به یک مدل طراحی و سپس پالایش تکراری، نمایش می دهد.

۵,۲. عناصر مدل طراحی

بعد	توضیح
عناصر طراحی داده ها	○ طراحی داده ها، یک مدل از داده ها ایجاد می کند که در سطح بالایی از انتزاع نمایش داده می شود. (دیدگاه کاربر نسبت به داده ها) ○ این مدل داده ها سپس به نمایش هایی پالایش می شود که به تدریج جزئیات خاص پیاده سازی بر آن ها افزوده می شود. ○ ساختمان داده ها، پایگاه داده ها و انبار داده ها
عناصر طراحی معماری	○ طراحی معماری برای نرم افزار، هم ارز نقشه برای ساختمان است. ○ دیدی کلی از نرم افزار را ارائه می دهد.
عناصر طراحی واسط ها	○ طراحی واسط ها برای نرم افزار، مشابه با مجموعه ای از ترسیم های مشروح برای درها، پنجره ها و سایر امکانات خارجی برای یک خانه است. ○ جریان های اطلاعات به درون و بیرون سیستم و چگونگی برقراری ارتباط میان مولفه های تعریف شده به عنوان بخشی از معماری را به تصویر می کشند.
عناصر طراحی در سطح مولفه ها	○ طراحی در سطح مولفه ها برای نرم افزار، هم ارز مجموعه ای از ترسیم های مشروح و مشخصات مربوط به هر اتاق در خانه اند. ○ طراحی در سطح مولفه ها برای نرم افزار، توصیف کاملی است از جزئیات داخلی هر مولفه نرم افزار.
عناصر طراحی در سطح استقرار	○ عناصر طراحی در سطح استقرار چگونگی تخصیص یافتن زیر سیستم ها و قابلیت های عملیاتی نرم افزار را در داخل محیط کامپیوتری ای نشان می دهند که نرم افزار را پشتیبانی می کند

جلسه ۱۳ -طراحی معماری

۱. مقدمه

۱,۱. طراحی معماری چیست؟

- طراحی به عنوان یک فرایند چند مرحله ای است که در آن، نمایش هایی از ساختار برنامه و داده ها، خصوصیات واسط و جزئیات روال ها از روی خواسته های اطلاعاتی ساخته می شود.
- طراحی یک فعالیت اطلاعات-محور است.
- طراحی معماری نشان گر ساختمان داده ها و مولفه های برنامه ای است که برای ساخت یک سیستم کامپیوتری مورد نیازند.

۱,۲. چرا اهمیت دارد؟

- بدون داشتن نقشه، ساختمان ساخته نمی شود. برای نقشه از کلیت ساختمان شروع می شود بعد به جزئیات پرداخته می شود.

۱,۳. مراحل کار کدام است؟

- طراحی داده ها
- طراحی ساختار معماری سیستم
- انتخاب سبک معماری
- طراحی جزئیات معماری

۱,۴. محصول کار چیست؟

- ایجاد یک مدل معماری شامل معماری داده ها و ساختار برنامه
- توصیف خواص مولفه ها و روابط (تعامل ها)

۲. معماری نرم افزار

۲,۱. معماری چیست؟

- معماری، روش انسجام بخشی اجزای ساختمان برای تشکیل کلیتی یکپارچه است.
- معماری سیستم، یک چارچوب جامع است که شکل و ساختار آن سیستم را توصیف می کند.

۲,۲. معماری نرم افزار چیست؟

- معماری نرم افزار، برای یک برنامه یا سیستم برنامه نویسی، ساختارهایی از سیستم است که از مولفه های نرم افزار، خواص بیرونی قابل مشاهده این مولفه ها و روابط میان آن ها تشکیل می شود.
- معماری نرم افزار باید ساختار یک سیستم و شیوه همکاری داده ها و مولفه های عملیاتی با یکدیگر را مدل سازی کند.

۲,۳. اهمیت معماری در چیست؟

- نمایش معماری نرم افزار، برقراری ارتباط بین طرف های ذینفع در توسعه یک سیستم کامپیوتری را میسر می سازد.
- معماری، تصمیم گیری های زودهنگامی را که بر همه کارهای بعدی مهندسی نرم افزار تاثیر می گذارند، برجسته می سازد.
- معماری یک مدل کوچک و قابل درک از چگونگی ساخت یافتگی سیستم و چگونگی همکاری مولفه های آن تشکیل می دهد.

۲,۴. مدل معماری

- مدل معماری نمایی کلی از سیستم فراهم می آورد، به طوری که مهندس نرم افزار می تواند آن را به عنوان یک کل بررسی کند.

۲,۵. توصیف های معماری

- معماری برای افراد متفاوت، معانی متفاوت دارد. ذینفعان متفاوت، معماری را از دیدگاه های متفاوتی می بینند. که علت آن در مجموعه دغدغه های متفاوت است.
- یک توصیف معماری مجموعه ای از محصولات کاری است که نماهای متفاوتی از سیستم ارائه می دهد.

۳. ژانرهای معماری

۳,۱. ژانرهای معماری

- ژانر معماری، رویکرد معماری مشخصی را در ساختاری که قرار است ساخته شود، دیکته می کند.
- ژانر به معنای گروهی خاص در دامنه کلی نرم افزار است. در هر گروه با چند زیر گروه مواجه می شویم.
- مثلا در ژانر ساختمان سبک های خانه، آپارتمان، ساختمان اداری و ... را خواهیم دید.

هوش مصنوعی	تجاری	ارتباطی	سرگرمی	مالی	بازی ها	دولتی
صنعتی	پزشکی	نظامی	علمی			

۴. سبک های معماری

۴,۱. سبک های معماری

معمار از یک سبک معماری مانند عمارت ویلایی به عنوان سازوکار توصیفی برای خانه استفاده می کند تا خانه را از سایر سبک ها مانند آپارتمانی متمایز سازد. نرم افزار نیز یکی از چند سبک معماری را از خود نشان می دهد. هر سبک گروهی از سیستم ها را توصیف می کند که شامل موارد زیر است:

- مجموعه ای از مولفه ها که وظیفه ای برای سیستم به انجام می رسانند.
- مجموعه ای از ارتباط دهنده ها که برقراری ارتباط، هماهنگ سازی و همکاری میان مولفه ها را امکان پذیر می سازند.
- قید و بندهایی که تعیین می کنند مولفه ها را چگونه می توان با هم منسجم ساخت و سیستم را ایجاد کرد.

- مدل های معناشناختی که طراح به کمک آن ها می تواند خواص کلی سیستم را با تحلیل خواص بخش های سازنده آن درک کند.

۴,۲. طبقه بندی سبک های معماری

سبک	توضیح
معماری های داده محور	○ محور این نوع معماری را یک فایل یا بانک اطلاعاتی تشکیل می دهد که دستیابی به آن غالبا توسط مولفه های دیگری صورت می پذیرد که داده های موجود در این بانک اطلاعاتی را به روز، اضافه، حذف یا به طریقی دیگر، اصلاح می کنند.
معماری های جریان داده ها	○ این معماری هنگامی به کار برده می شود که قرار باشد داده های ورودی از طریق یک سری مولفه های محاسباتی و دستکاری، به داده های خروجی ابدیل شوند.
معماری های فراخوانی و بازگشت	○ به کمک این سبک، می توانید به ساختاری برای برنامه دست پیدا کنید که اصلاح و تغییر دادن ابعاد آن نسبتا آهسته باشد.
معماری شی گرا	○ مولفه های این سیستم، داده ها و عملیاتی را که باید برای دستکاری آن ها اجرا شوند، کپسوله می کنند. برای برقراری ارتباط و هماهنگ سازی میان مولفه ها از طریق مبادله پیام انجام می شود.
معماری لایه ای	○ در این سبک، تعدادی لایه های متفاوت تعریف می شود که هر یک عملیاتی را انجام می دهند و به طور تدریجی به دستورات ماشین نزدیک تر می شوند.

۴,۳. الگوهای معماری

- الگوهای معماری به مساله ای با کاربرد خاص در حیطه ای مشخص و تحت مجموعه ای از محدودیت ها و قید و بندها می پردازد.
- این الگو، یک راهکار معماری پیشنهاد می کند که می تواند به عنوان مبنایی برای طراحی معماری عمل کند.

۵. طراحی معماری

۵,۱. حیطه معماری

- حیطه معماری چگونگی تعامل نرم افزار با موجودیت های خارج از مرزهای آن را نمایش دهید.

۵,۲. زبان های توصیف معماری

- معماری یک خانه، دارای مجموعه ای از ابزارهای استاندارد است که ارائه و نمایش طراحی را به شیوه ای قابل فهم و عاری از ابهام امکان پذیر می سازد.
- زبان توصیف معماری (ADL) ابزاری معنایی و نحوی را برای توصیف معماری نرم افزار فراهم می آورد.

جلسه ۱۴ – طراحی در سطح مولفه ها

۱. مقدمه

۱,۱. طراحی در سطح مولفه ها چیست ؟

- در طراحی در سطح مولفه ها موارد ساختمان داده ها، الگوریتم ها، سازو کارهای ارتباطی و خصوصیات واسط های هر مولفه از نرم افزار تعریف می شود

۱,۲. چرا اهمیت دارد؟

- تضمین صحت و سازگاری با معماری داده ها و طراحی واسط.
- ارزیابی این که آیا ساختمان داده ها، واسط ها و الگوریتم ها کار می کنند یا خیر.

۱,۳. محصول کار چیست؟

- مستندات طراحی مولفه ها در قالب های گرافیکی، جدول و متن.

۱,۴. مراحل طراحی

- طراحی داده ها - طراحی معماری - طراحی واسط - طراحی مولفه ها

۲. مولفه چیست؟

۲,۱. مولفه چیست؟

- بخش پیمانه ای، قابل استقرار و قابل تعویض از یک سیستم
- مولفه ها معماری نرم افزار را تشکیل می دهند. مولفه ها باید قادر به برقراری ارتباط و همکاری با سایر مولفه ها و موجودیت های خارجی باشند.

۲,۲. دیدگاه شیء گرا

- مولفه ها مجموعه ای از کلاس هاست که با یکدیگر همکاری دارند. هرکلاس در داخل یک مولفه دارای جزئیات کامل است.
- انواع: کلاس های تحلیلی برای مولفه هایی که با دامنه مسئله مرتبطند و کلاس های زیرساختی برای مولفه هایی که سرویس ها را برای دامنه مسئله فراهم می سازند.

۲,۳. دیدگاه سنتی

- مولفه یک عنصر عملیاتی از برنامه است که شامل منطق پردازش، ساختمان داده های داخلی که برای پیاده سازی منطق پردازش لازم اند و واسطی که فراخوانی مولفه ها و تحویل داده ها به آن را میسر می سازند، است.

- مولفه ها در این دیدگاه =پیمانه
- انواع مولفه ها: مولفه کنترلی، مولفه دامنه مسئله که یک قابلیت عملیاتی کامل یا بخشی از آن را پیاده سازی می کند و مولفه زیرساختی که مسئول قابلیت های عملیاتی پشتیبان برای پردازش لازم در دامنه مسئله است.
- نکته: در طراحی مولفه های سنتی از عنصر جریان گرای مدل تحلیل استفاده می شود.

۲,۴. دیدگاه فرایندی

- در این دیدگاه کاتالوگی از مولفه ها در اختیار شما قرار داده می شود. مولفه های مورد نظر را از کاتالوگ انتخاب می کنید وآنها را در معماری خود جای می دهید.
- تاکید برالگوهای طراحی و قابلیت استفاده مجدد.

۳. طراحی مولفه های مبتنی برکلاس

۳,۱. اصول پایه طراحی

اصل	توضیح
اصل باز-بسته(ocp)	○ یک مولفه باید برای عمل بسط، باز و برای عمل اصلاح، بسته باشد. ○ یعنی باید مولفه را به قسمی مشخص کنید که بتوان آن را بسط داد (در دامنه عملیاتی مربوط)، بدون این که نیازی به انجام اصلاحات داخلی (در سطح کدها) باشد. ○ برای دستیابی به این هدف، انتزاع هایی ایجاد می کنید که میان قابلیت عملیاتی که باید بسط داده شود و خود کلاس طراحی به عنوان واسط عمل کند.
اصل جایگزینی لیسکوف (LSP)	○ زیر کلاس ها باید با کلاس های پایه خود جایگزین پذیر باشد. ○ یعنی مولفه ای که از یک کلاس پایه استفاده می کند، اگر به جای کلاس پایه کلاس مشتق آن به مولفه ارسال شود، مولفه باید به درستی عمل کند.
اصل وارونگی وابستگی (DIP)	○ به انتزاع ها متکی باشید و به عینیت ها متکی نباشید. ○ انتزاع ها نقاطی هستند که طراحی از آن ها بدون پیچیدگی زیاد، بسط و گسترش می یابد. ○ هرچه وابستگی یک مولفه به سایر مولفه های عینیت یافته بیشتر باشد، بسط و گسترش آن دشوار تر خواهد بود.
اصل جداسازی واسط ها(ISP)	○ داشتن واسط های خاص کلاینت بسیار بهتر از یک واسط چند منظوره است. ○ واسط های فراوانی وجود دارد که در آنها چند مولفه کلاینت از عملیات هایی استفاده می کنند که توسط یک کلاس سرور منفرد فراهم می آیند. ○ طبق این اصل باید برای سرویس دهی به هر دسته عمده از کلاینت ها یک واسط تخصص یافته ایجاد کنید.
اصل هم ارزی استفاده مجدد از نسخه ها (REP)	○ استفاده مجددسنگ بنای ارائه نسخه های جدید است.
اصل بستار مشترک (ccp)	○ کلاس هایی که با هم تغییر می کنند به هم تعلق دارند. ○ کلاس ها باید به طور مناسب در پکیج قرار گیرند.
اصل استفاده مجدد مشترک(CCP)	○ کلاس هایی که با هم دوباره استفاده نمی شوند، نباید در یک پکیج قرار گیرند.

۳,۲. یکپارچگی

- یعنی یک مولفه یا کلاس فقط صفات و عملیاتی را در خود پنهان سازی می کند که رابطه ای تنگاتنگ با یکدیگر و با خود کلاس یا مولفه دارند.

۳,۳.اتصال

- میزانی کیفی از درجه اتصال کلاس ها به یکدیگراست.
- با وابستگی بیشتر کلاس ها و مولفه ها به یکدیگر اتصال افزایش پیدا می کند که باعث پیچیده ترشدن سیستم می شود.
- با افزایش پیچیدگی، پیاده سازی، آزمون ونگهداری نرم افزار دشوارتر می شود.
- یک هدف مهم در طراحی مولفه حفظ اتصال، در حداقل سطح ممکن است.

۴. اجرای طراحی در سطح مولفه ها

۴,۱. مراحل

- همه کلاس های منتاظر با دامنه مسئله را شناسایی کنید.
- همه کلاس های طراحی متناظر با دامنه زیرساختار را شناسایی کنید. این کلاس ها عبارتند از: کلاس های واسط کاربر، کلاس های سیستم عامل و کلاس های مدیریت داده ها واشیاء.
- جزئیات همه کلاس هایی را که به عنوان مولفه های قابل استفاده مجدد به دست نمی آیند، تعیین کنید.
- منابع داده ای پایدار (فایل ها و بانک های اطلاعاتی) را توصیف و کلاس های لازم برای مدیریت آنها را تعریف کنید.
- نمایش های رفتاری مربوط به کلاس یا مولفه را بسط و توسعه دهید.
- نمودارهای استقرار را بسط دهید تا جزئیات پیاده سازی فراهم آید.
- نمایش طراحی در سطح مولفه هار بازآرایی کنید وهمواره راههای دیگر را مدنظر داشته باشید زیرا طراحی فرایندی مبتنی برتکرار است.