سر فصل

۱ – مقدمه

- تعریف نرم افزار
- تعریف مهندسی نرم افزار

۲ – فرآیند های تولید نرم افزار

- آبشاری
- تکاملی
- تدریجی
- رسمی
- مارپیچ

۳ – مدیریت پروژه

- برنامه ریزی پروژه
- زمان بندی پروژه
- مدیریت ریسک
  - شناسایی ریسک
  - تحلیل ریسک
  - برنامه برخورد با ریسک
  - نظارت

۴ – خواسته های نرم افزار

- خواسته های عملکردی و غیر عملکردی
- خواسته های کاربر
- خواسته های سیستم
- سند خواسته های نرم افزار

۵ – فرآیند مهندسی خواسته ها

- مطالعات امکان سنجی
- اعتبار سنجی خواسته ها
- مدیریت خواسته ها

۶ – مشخصات سیستم های حاتی

- قابلیت اتکا
  - اعتماد
  - دسترسی
  - امنیت
  - حفاظت

۷ – وارسی و اعتبار سنجی

- برنامه ریزی اعتبار سنجی
- بازبینی نرم افزار
- تحلیل ایستای نرم افزار

۸ – تست نرم افزار

- تست سیستم
- تست قطعه
- طراحی موارد تست
- خودکار سازی تست

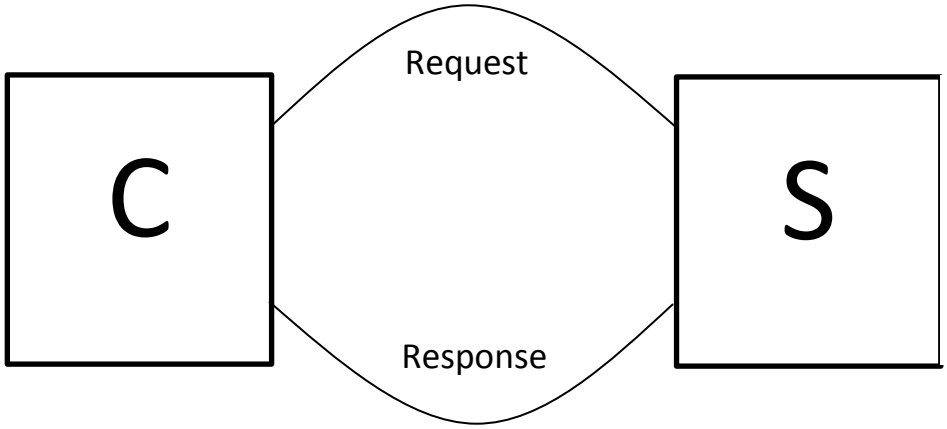
۹ – مدل های سیستمی

- مدل رفتار
- مدل داده
- مدل ساخت یافته
- مدل های شی گرا
  - نمودار کلاس
  - نمودار حالت
  - نمودار فعالیت
  - نمودار توالی

۱۰ – برآورد هزینه

زبان ماشین	زبان اسمبلی	زبان های ساخت یافته	زبان های شی گرا
صفر و یک	ADD R۱,R۲	تابع ، رویه	Oop
			C++
			BP

زبان تحت ویندوز	زبان های تحت وب	زبان های مستقل از پلتفورم
VC++	HTML	Java
Delphi	Java Script	.Net
V Basic	VBScript	



Php -> c

Asp -> VB

Cold fusion

مراحل شناخت زبان های تحت وب

Clint : HTML → CSS → java script → XML → Ajax

Server : ASP / PHP

محل اجرای برنامه های Server

- Web Server
  - Window
    - PWS → Home Edition
    - IIS → Server → professional Editon
  - Linux →→→Apache

نظریه مهندسی نرم افزاردر سال ۱۹۶۸ در کنفرانسی تحت عنوان بحران نرم افزار یا Software Crysis مطرح شد .

در آن زمان تولید نرم افزار های به صورت غیر رسمی بود . (سلیقه ای) بنابراین نرم افزار های تولید شده غالبا دارای اشکالات زیر بودند :

۱ – انجام پروژه های بزرگ ، سال های طولانی طول می کشید .

۲ – هزینه های واقعی تولید نر افزار بیش از مقدار تخمین زده شد بود .

۳ – قابل اطمینان و اعتماد نبودند .

۴ – نگهداری آن ها دشوار بود و کارایی نرم افزار ها پایین بود .

مهندسی نرم افزار

مهندسی نرم افزار عبارت است از پایه ریزی و استفاده از اصول و قواعد معتبر به منظور بدست آوردن نرم افزار قابل اعتماد ، کارا و مقرون به صرفه است

نرم افزار

طبق تعریف یان سامرویل نرم افزار عبارت است از برنامه های کامپیوتری و اسناد مربوط به آن . منظور از اسناد ، فایل پیکر بندی سیستم ، مستندات ساختاری سیستم و مستندات استفاده کاربران است .

نرم افزار

۱ – نرم افزار عمومی : سیستم هایی که توسط یک شرکت تولید کننده در موضوعی خاص تولید شده و به بازار عرضه شده باشد . مانند : واژه پرداز

۲ – نرم افزار های سفارشی : سیستم هایی که بر اساس نیاز های خاص یک سازمان و یا یک شرکت تولید کننده سفارش داده می شود .

نرم افزار ها از نگاهی دیگر به دو دسته کلی دیگر تقسیم بندی می شودند : ۱ – کد بسته ۲ – متن باز

فرآیند نرم افزار

مجموعه ای از فعالیت ها و نتایج مربوط به آن ها که منجر به تولید نرم افزار می شود را فرآیند توید نرم افزار گویند .

برخی از فعالیت ها تولید نرم افزار می تواند توسط ابزار ها یا (Case) ها صورت گیرد .

Case : Computer Aided Software Engineering

مصاحبه های تحلیلی → Word

برای کد نویسی → V Stadoi۲۰۱۰

برای مدل سازی → UML ← Rational Rose

برنامه ریزی و زمان بندی → My Project

My Visio

فرآیند های مختلفی در تولید نرم افزار وجود دارد اما تعدادی از آنها که در همه آنها مشترک هستند عبارتند از :

۱ – تعیین مشخصات Specification

۲ – طراحی و پیاده سازی Development

۳ – اعتبار سنجی Validation

۴ – نگهداری و تکامل Evolution

۱ – تعیین مشخصات

در این فعالیت مشخص می شود که سیستم بایستی چه خدماتی ارائه بدهد (مهندسی نیازمندی های و خواسته ها)

خطا در این فعالیت باعث بروز اشکال در فعالیت های دیگر از قبیل طراحی و پیاده سازی می شود .

مراحل مهندسی خواسته ها عبارتند از :

- امکان سنجی

امکان سنجی می بایستی سریع و ارزان باشد . نتیجه ی این مرحله مشخص می کند که تحلیل تفصیلی و یا تعیین مشخصات سیستم صورت بگیرد .

از جنبه های زیر یک تحلیلگر می بایستی سیستم را امکان سنجی کند :

۱ – امکان سنجی مالی : مشخص شدن برائت مالی کار فرما جهت انجام و ادامه تکمیل پروژه ، که این امکان سنجی به تجربه شخص تحلیلگر باز می گردد .

۲ – امکان سنجی فنی : امکان سنجی از لحاظ تکنیکی و از لحاظ پیاده سازی مورد بررسی قرار می گیرد .

۳ – امکان سنجی نیروی انسانی : بحث بر سر این که آیا فرد برای نوشتن برنامه وجود دارد و آیا این که شخص کاربر توانایی کار با نرم افزار را دارد و یا خیر

۴ – مرغون به صرفه بودن و ضرورت آنها

- استخراج و تحلیل داده ها

در این مرحله می بایستی نیازمندی های مشتری از طریق مشاهده سیستم (دستی یا مکانیزه) و یا مصاحبه با کاربران استخراج گردد .

برای این کار می توان از ۳ طریق عمل کرد :

۱ – دیدگاه : خود را در جایگاه افراد مختلف قرار دهیم و سیستم را از لحاظ افراد بررسی کنیم

۲ – سناریو : تعدادی زیر عمل که اگر پشت سر هم اتفاق بیفتد عمل کامل می شود مانند : سناریو بانک

۳ – اتناگرافی : تحلیل (فرهنگ سازمانی) به معنای این که خود شخص در سازمان حضور پیدا کند و شخص تحلیل گر به جای کارمند قرار بگیرد

- ثبت مشخصات نرم افزار

در این مرحله کلیه ی نیازمندی های سیستم می بایستی به صورت استاندارد مستند سازی شود .

- اعتبار سنجی

بررسی خواسته ها از نظر واقعی بودن (شدنی هست یا نیست) ، سازگاری (تناقض میان خواسته ها وجود دارد یا خیر) ، تمامیت (آیا همه چیزی بوده است که مشتری از ما خواسته است یا خیر ؟)

۲ – طراحی و پیاده سازی

در این فعالیت دو زیر فعالیت عمده ی طراحی و پیاده سازی صورت می گیرد .

طراحی ⟵ مدل سازی سیستم ⟵ ERD ، Class ، DFD ، ....

پیاده سازی ⟵ برنامه نویسی

مدل سازی جنبه های مختلفی سیستم را نمایش می دهد : ۱ – فرآیند ها (DFD) ۲ – داده ها (ERD)

۳ – اعتبار سنجی

بررسی می گردد که سیتم پیاده شده آیا با مشخصات یکسان است یا خیر و وارسی می شود تا اگر نرم افزار اشکالی دارد کشف و سپس رفع گردد .

تست نرم افزار می تواند در هنگام پیاده سازی نیز صورت می گیرد .

تست ها را می توان به انواع مختلفی تقسیم بندی کرد که عبارتند از :

- تست واحد : منظور از تست واحد همان تست تابع است و می بایستی توسط برنامه نویس صورت بگیرد .
- تست ماژول ها : به تعدادی از توابع که ارتباط منطقی با یکدیگر داشته باشند ماژول گویند و این تست توسط برنامه نویس صورت می گیرد .
- تست زیر سیستم : چند ماژول که با یکدیگر جامعیت پیدا کنند زیر سیستم گویند . در تست زیر سیستم باید بر اساس ورودی های داده شده ارزیابی گردد .
- تست سیستم : جامعیت چند زیر سیستم را تست سیستم گویند این تست توسط تولید کننده صورت می گیرد چنانچه این تست توسط کاربران نهایی صورت بگیرد به آن تست پذیرش گویند . تست پذیرش آن قدر ادامه پیدا می کند تا کار فرما و مجری به توافق برسند .

۴ – تکامل و نگهداری

در این فعالیت سیستم راه اندازی شده دچار مشکل گردد توسط یک تیم نگهدارنده بایستی اشکال رخ داده را رفع کند .

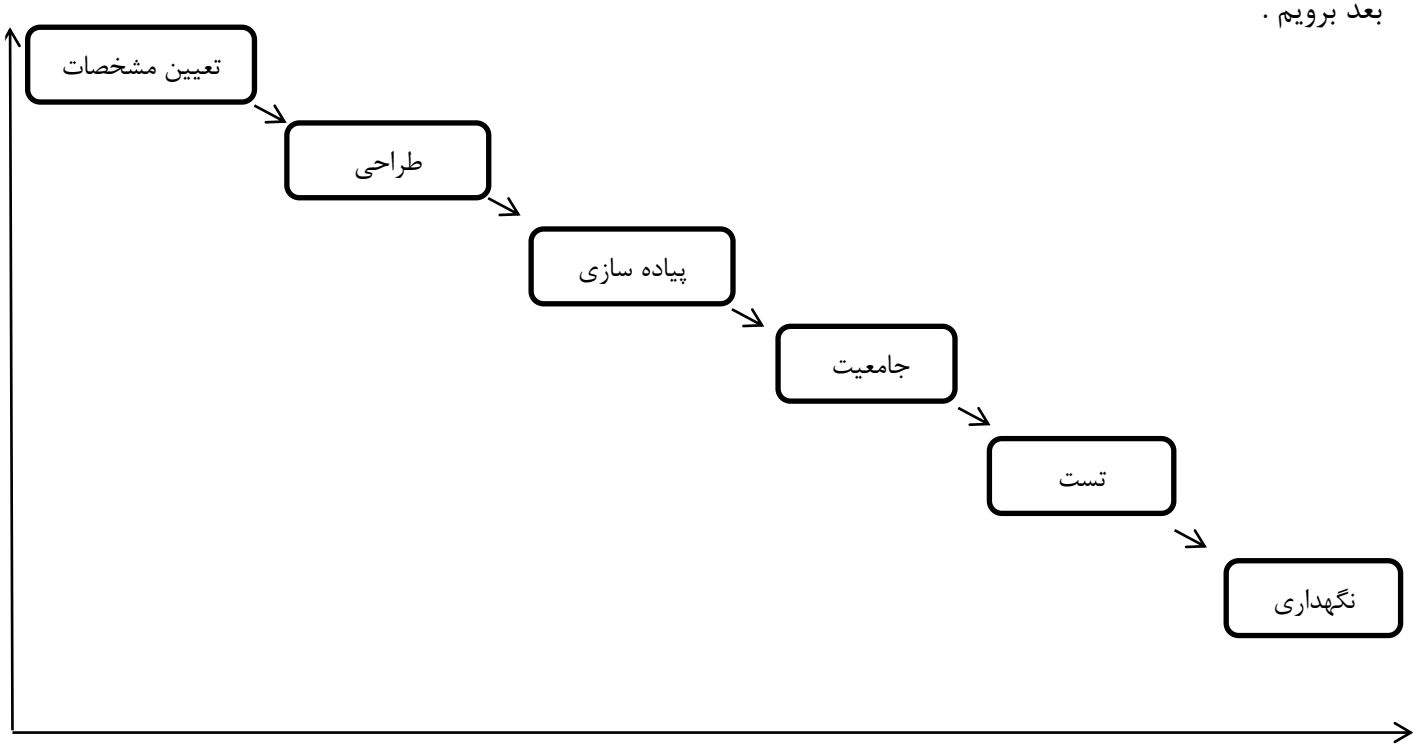
متدولوژی های تولید نرم افزار

- آبشاری
- تکاملی
- رسمی
- استفاده از مجدد
- تدریجی

همانطور که قبلا بیان شد فرآیند عبارت است از مجموعه ای از فعالیت ها که انجام آنها منجر به ایجاد یک نرم افزار می شود .

۱ – مدل آبشاری

در این روش برای تولید نرم افزار می بایستی یکبار فعالیت ها یا مرحله ای از پروژه به پایان برسد تا بتوانیم به مرحله ی بعد برویم .



نکته : در این روش راهی به مراحل و یا فعالیت های قبل وجود ندارد .

- حسن این روش در نزدیک بودن تخمین هزینه و زمان با واقعیت می باشد .
- یکی از اشکالات این روش آن است که کارفرما با مشتری تا پایان انجام پروژه نرم افزار مشاهده نمی کند و اگر اشکالی داشته باشد در انتهای کار ، کار فرما متوجه می شود .

  برای کم کردن اثر مشکل بالا در مرحله ی تعیین مشخصات در این روش پیشنهاد می شود که مجری از سیتسم پوسته یا دمویی را به مشتری ارائه دهد .

  پوسته (دمو) ارائه شده توسط کارفرما ارزیابی شده و نظرات وی برای پالایش خواسته ها مورد استفاده قرار می گیرد .

  مشکل اثر فوق در این است که کار فرما فکر می کند که این پوسته همان نرم افزار اصلی است .
- این روش تولید نرم افزار مناسب سیستم هایی است که در ابتدای کار کلیه ی خواسته ها و نیازهایش مشخص باشد .

۲ – تکاملی

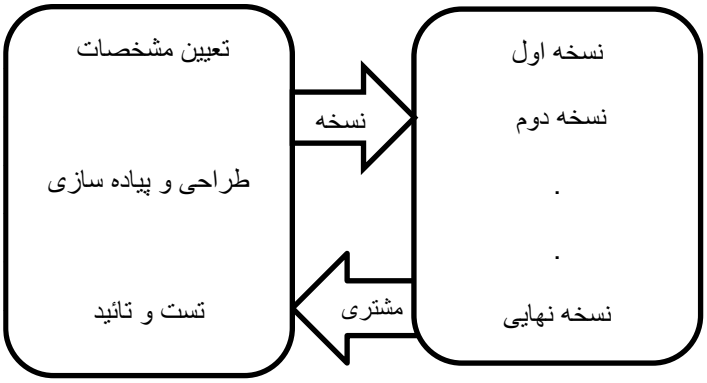
در این روش ، سیتم در ابتدا تولید می گردد و به مرور تکمیل شده تا به سیستم نهایی برسیم . علت تکمیل تدریجی این نرم افزار در مشخص نبودن کلیات نیاز های سیستم می باشد و بر خلاف روش آبشاری چند فعالیت می توان به صورت موازی صورت گیرد و تقدم و تاخر فعالیت ها نیز مورد نظر گرفته نمی شود .

دو نوع تولید نرم افزار به مدل تکاملی پیشنهاد شده است :

۱ – تولید سیستم در روی نیاز هایی تاکید دارد که مشخص و شفاف خواهد بود و با ارائه ورژن ها و یا نسخه های متعدد به مشتری نیازهای مبهم مشخص شده و سیستم نهایی بدست می آید .

۲ – تولید نسخه یا نسخه ی اولیه بر روی خواسته ها تاکید دارند که به خوبی درک نشده اند . و می توان گفت اشکال این متدولوژی در تخمین زمان و هزینه است . روش تکاملی در جایی کارآمد است که خود کار فرما ، یک گروه IT داشته باشد که اگر نرم افزار با مشکل رو به رو شد خود گروه IT آنرا رفع کنند .

۳ – رسمی



مدل رسمی برای تولید سیستم های حیاتی مناسب هستند . سیتم حیاتی سیستمی است که خطا در آن باعث خسارت مالی و جانی می گردد مانند سیستم تزریق انسولین و با ATM ها . بنابراین در این گونه سیستم ها خطا و اشتباه به حداقل برسد . برای کلیه نرم افزار ها می توان خصوصیتی با نام قابلیت اتکا تعریف کرد .

هر چقدر قابلیت اتکاء یک نرم افزار افزایش یابد زمان و هزینه ی آن افزایش می یابد

قابلیت اتکاء دارای وجوه زیر است :

۱ – قابلیت اعتماد

قابلیت اتکاء را می توان توانایی سیستم در ارائه دقیق سیستم ها دانست

مثال : برداشت پول از طریق ATM . اگر مبلغ ۱۰۰هزار تومان بخواهیم ولی ۹۰ هزار تومان با ما بدهد اعتماد خود را نسیت به سیستم ATM از دست خواهیم داد .

۲ – قابلیت دسترسی

توانایی سیستم در ارائه ی سرویس در زمان خاص

مثال : در سیستم ATM هرگاه که نیاز به ATM داشته باشیم آن دستگاه باید درست کار کند . و مثال دیگر را می توان به سیستم ثبت نام تشبیه کرد

۳ – امنیت

سیستم به محیط خود آسیب نرساند

مثال : مانند یک کارخانه ی مواد شیمیایی که اگر مواد نشت کردند نتواند روی دیگر اجزای کارخانه تاثیر گذار باشد . در سیستم ATM اگر بیش از ۳ بار رمز را اشتباه وارد کرد کارت را به داخل خود بکشد .

۴ – محافظت

حفاظت سیستم از ورودی ها خود

**پروژه : نرم افزار تحویلی پروژه های طراحی نرم افزار و پایگاه داده باید مبحث حفاظت اطلاعات را دارا باشد بنابر این موارد زیر پیاده سازی می شود :

۱ – امکان شناسه کاربری و کلمه عبور داشته باشد

۲ – اجازه ی استفاده از امکانات نرم افزار بر اساس نقش

نقش : به میزان دسترسی هر فرد در سیستم نقش آن فرد در سیستم گفته می شود .

۳ – ثبت ماوقع و آنچه اتفاق افتاده است

مانند هنگامی که نمره دانشجو تغییر می کند باید مشخص باشد که چه کسی نمره ی وی را تغییر داده است .

به این ۳ امکان بالا AAA گویند . Authentication به معنی تصدیق Authorization به معنی اجازه و Accounting به معنی کاربر است .

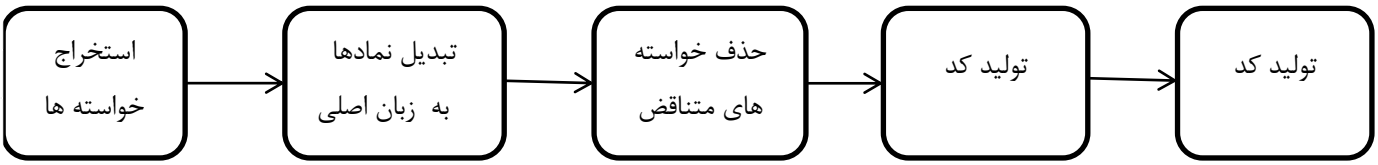
در مدل رسمی خواسته ها و نیازمندی های نرم افزار به صورت نمادها ارائه می شود .

ارائه خواسته ها و نیازمندی ها به زبان طبیعی می تواند دارای اشکالات زیر باشد :

۱ – ابهام : امکان تفاسیر مختلف از نیاز ها

۲ - تناقض : امکان وجود نیازهایی که در تضاد یکدیگرند .

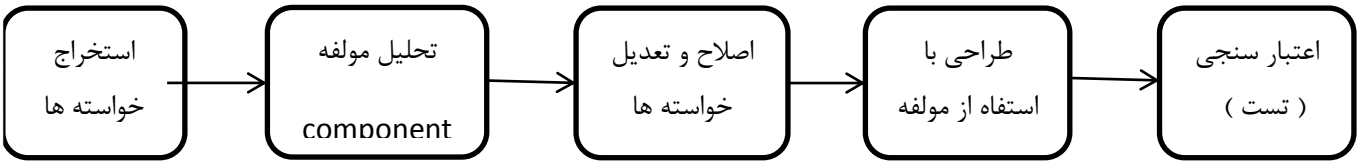
با ارائه ی خواسته ها به یک زبان قراردادی می توانیم مشکلات فوق را رفع کنیم (حذف ابهام ها و حذف تناقض ها) بر این اساس تولید سیستم نرم افزار به روش رسمی می تواند شامل مراحل زیر باشد



تولید نرم افزار به روش رسمی هزینه ی بالایی دارد .

۴ – توسعه مبتنی بر استفاده مجدد

مراحل این روش به صورت زیر است



هدف استفاده از این روش استفاده از مولفه های موجود و یا تغییر آن ها می باشد . مطابق با این روش سیستم سریعتر تولید می شود .

در بخش تحلیل مولفه مجری سیستم می بایست Component های موجود را بر اثر خواسته های مشتری جست و جو و بررسی نماید (معمولا مولفه ای پیدا نمی شود که تمامی نیاز های مشتری را پوشش دهد)

با توجه به عدم پوشش مولفه در مرحله اصلاح و تعدیل خواسته ها می بایستی تا آنجایی که می شود خواسته ها و مولفه ها با یکدیگر سازگار باشند .

حسن این روش در کاهش هزینه و زمان تولید نرم افزار است و اشکال این روش آن است که مولفه های استفاده شده در اختیار و کنترل سازمان نمی باشد (می توان گفت ممکن است برنامه دارای باگ باشد و یا از لحاظ امنیت مشکل دارد زیرا نمی دانیم از چه ماژول تابع هایی استفاده شده است)

۵ – تدریجی

این روش برای انجام پروژه های نرم افزاری بزرگ به کار می آید این روش می تواند ترکیبی از روش های گفته شده باشد. برای اجرای این روش می بایستی سیستم را به فاز ها و یا مراحل مختلف دسته بندی کنیم .

در این فرآیند می بایستی موارد زیر را انجام گیرد :

۱ – مشتری طرح کلی نیازها و خواسته های خود را ارائه دهند .

۲ – اهمیت و الویت بندی خواسته ها از طرف مشتری مشخص شده باشد .

۳ – پروژه را فاز بندی کنیم (در انتهای هر فازی می بایست عملکردی از سیستم جامع راه اندازی گردد .)

۴ – پس از فاز بندی پروژه تعریف دقیق تر خواسته های هر فاز باید انجام گیرد .

۵ – هر فاز می تواند توسط یک روش با توجه به ماهیت یک زیر سیستم تولید گردد (مثلا یک زیر سیستم را از روش آبشاری و زیر سیستم دیگر را از روش رسمی و زیر سیستم دیگر را از روش .....)

۶ – پس از تکمیل هر فاز می بایستی زیر سیستم راه اندازی گردد .

از جمله حسن های این روش می توان به موارد زیر اشاره کرد :

۱ – به دلیل تقطیع سیستم به زیر سیستم های امکان احتمال خرابی پروژه کمتر است .

۲ – نیازی نیست که مشتری تا پایان سیستم صبر کند تا کل سیستم را تحویل بگیرد بلکه سیستم به مرور با تحویل زیر سیستم ها و جامعیت آنها تحویل داده می شود .

۳ – مشتری می تواند با کار بر روی یک زیر سیستم ، می تواند زیر سیستم های دیگر را بهتر تعریف کند .

۴ – از آنجایی که زیر سیستم ها با توجه به الویت تحویل داده می شوند لذا بخش های مهم تر ، بیشتر تست می شوند .

مدیریت پروژه

با توجه به تعریف مهندسی نرم افزار نیاز است برای تولید نرم افزاری مقرون به صرفه بر روی پروژه ی تولیدی در مدیریتی واحد صورت گیرد . شخص مدیر دارای وظایفی از قبیل :

۱ – نوشتن پیشنهاد نامه

۲ – برنامه ریزی

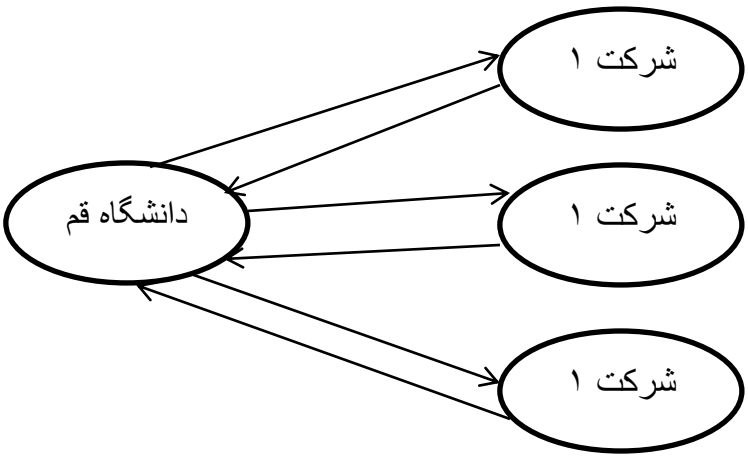
۳ – زمان بندی

۴ – مدیریت ریسک

نوشتن پیشنهاد نامه

پیشنهاد نامه یا Proposal سندی که در پاسخ یک RFP (Request For Proposal) می بایستی مدیریت تهیه نماید .

سند RFP سندی است که توسط سازمان های نرم افزار تدوین می شود .



تولید کنندگان واقعی RFP مهندسان نرم افزارند که به سفارش صاحبان سیستم تهیه می کنند .

RFP شامل موارد زیر است :

۱ – معرفی سازمان (مشتری)

- تاریخچه ی مختصر
- واحد ها و طبقه بندی سازمان
- تخصص های موجود
- معرفی سیستم های نرم افزار ی و سخت افزاری

۲ – دستور العمل ها

- زمان بندی منتهی به قرارداد (با چه افرادی در چه زمانی تماس گرفته شود / مهلت ارسال پیشنهاد نامه / جلسه دفاع از پیشنهاد نامه)
- انتظارات در مورد قرارداد (حقوق)
- انتظارات فنی (متدولوژی – زبان – پایگاه و .....)

۳ – نیاز ها و خصوصیات سیستم نرم افزاری

** پروژه : ۱ – نوشتن RFP (Word)

۲ – Proposal (Word)

۳ – برنامه ریزی

پشنهاد نامه شامل موارد زیر است :

۱ – معرفی شرکت تولید کننده نرم افزار

۱ – تاریخچه

۲ – مقررات سازمان

۳ – مدل و روش های تولید نرم افزار

۴ – تخصص ها

۵ – سابقه کار و تجربه در زمینه پروژه مربوط (رزومه)

۶ – زیر مجموعه شرکت های موجود در شعبه های مختلف

۲ – بررسی مساله و پیشنهاد راه حل ها و اشاره به نقاط قوت و ضعف آنها

۳ – توصیف خدمات نگهداری و پشتیبانی پس از تولید

۴ – توصیف نحوه ی آموزش

۵ – ابزار های پیاده سازی

۶ – زمان بندی بر اساس

۷ – قیمت قرارداد

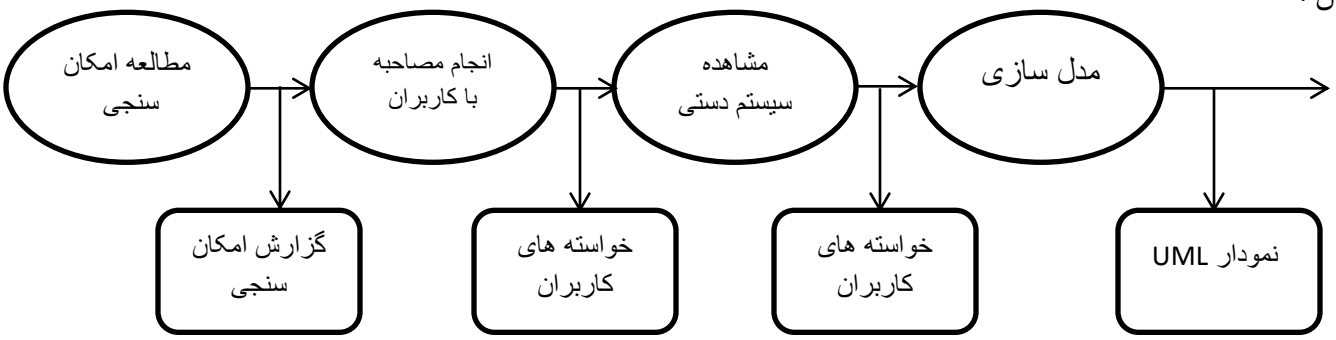
فاز	عنوان	مدت زمان
اول	تحلیل	۲ هفته
دوم	مدل سازی و طراحی پایگاه	۴ هفته
سوم	پیاده سازی نرم افزار ورودی اطلاعات	۶ هفته

برنامه ریزی

مدیر پروژه موظف است هنگام برنامه ریزی پروژه مجموعه ای از نقاط عطف را تعیین کند . نطقه عطف ، نقطه ی پایانی یک فعالیت از فرآیند نرم افزار است که خروجی مشخص دارد .

پس از تعیین نقاط عطف پروژه ، می بایستی فعالیت زمانی را تخمین بزند .

مثال :



پس از استخراج فعالیت ها و تخمین زمان انجام هر فعالیت می بایستی زمان بندی آنها به صورت نمودار هایی ارائه می شود . معمولا از دو نمودار شبکه ای و گانت برای این کار استفاده می شود .

مثال :

مشخصه فعالیت	پیش نیاز	عنوان	زمان تخمین
A	----------	تحلیل	۴
B	A	فرم ورود کالا	۳
C	A	حواله انبار	۳
D	B	فرم ویرایش کالا	۶
E	D	فرم جست و جوی کالا	۶
F	C	گزارش خروجی کالا	۲
G	E,F	تست	۴

برای رسم نمودار شبکه ای به صورت زیر می بایستی عمل کنیم :

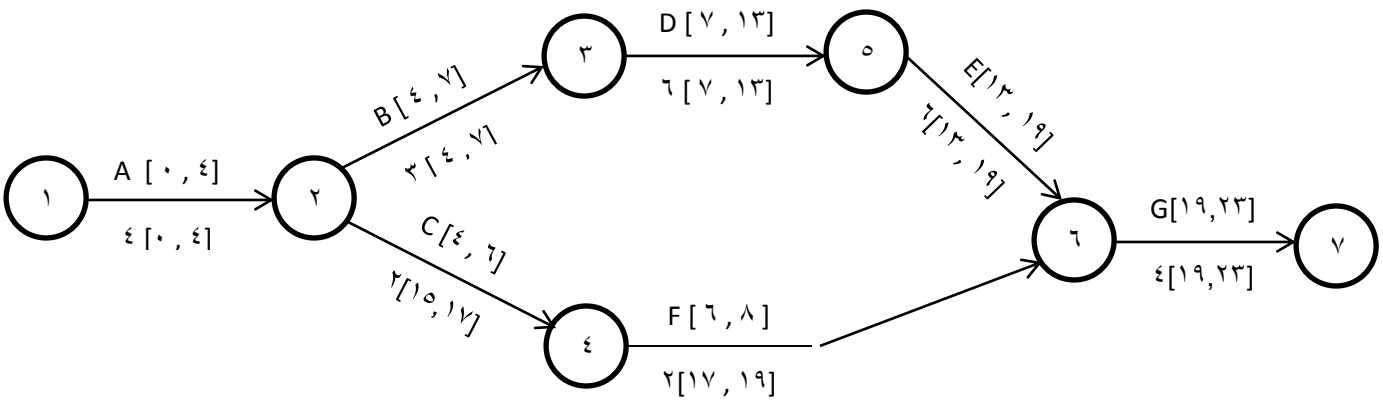
نمودار شبکه شبیه یک گراف است که دارای تعدادی گره و یال می باشد که بر روی هر یال بر چسب هایی به صورت زیر نسبت داده می شود .

۱ – فعالیت ۲ – زمان تخمین فعالیت ۳ – زودترین زمان شروع فعالیت

۴ – زودترین زمان پایان فعالیت ۵ – دیرترین زمان شروع فعالیت ۶ – دیرترین زمان پایان فعالیت

رسم :

الف) رسم گراف (شبکه) با توجه به رعایت پیشنیازها

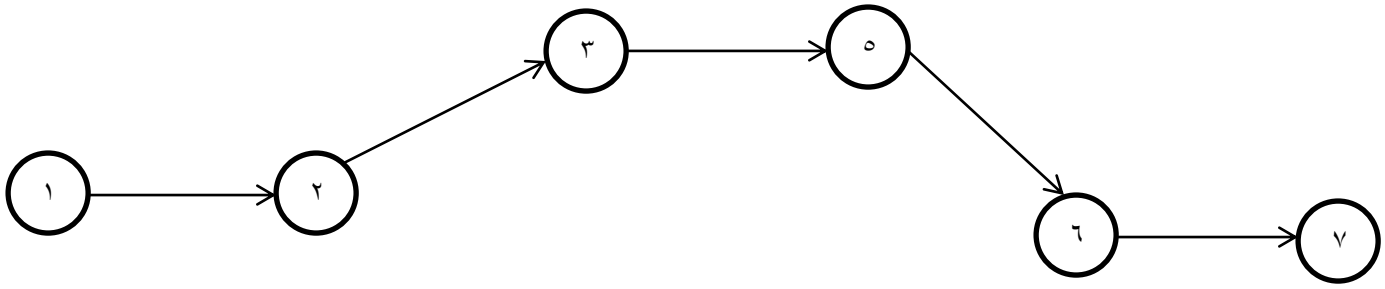


ب) پیمایش گراف از ابتدا تا انتها و مشخص کردن زودترین زمان شروع و زودترین زمان پایان برای هر یال . و با این پیمایش در انتها زمان پروژه بدست می آید .

ج) پیمایش گراف از انتها تا ابتدا و مشخص کردن زمان های دیرترین شروع و دیرترین پایان .

مسیر بحرانی

مسیری است که بر روی آن فعالیت ها نمی توانند تاخیر در انجام داشته باشند . کمترین تاخیر در هر فعالیت این مسیر باعث تاخیر در تحویل پروژه می گردد .



روش دیگر برای ارائه زمان بندی استفاده از نمودار گانت است که می توان هنگام تعریف در فعالیت منابعی را به آن انتصاب کرد .

	فروردین				اردیبهشت				خرداد			
	۱	۲	۳	۴	۱	۲	۳	۴	۱	۲	۳	۴
مطالعه امکان سنجی												
مصاحبه به کاربران												
مشاهده سیستم دستی												
طراحی پایگاه												
پیاده سازی فرم ورود												

مدیریت ریسک

ریسک شرایط نا مطلوبی است که در اثر وقوع آن زمان بندی و کیفیت پروژه نرم افزاری تحت تاثیر قرار می گیرد .

در یک دسته بندی می توان انواع ریسک زیر را دسته بندی کرد :

۱ – ریسک پروژه : منظور خطراتی است که زمان بندی پروژه را تحت تاثیر قرار می دهد .

۲ – ریسک محصول : ریسکی که کیفیت یا کارایی نرم افزار را تهدید می کنند .

۳ – ریسک های کاری : ریسک هایی که سازمان تولید کننده نرم افزار را تحت تاثیر قرار می دهند .

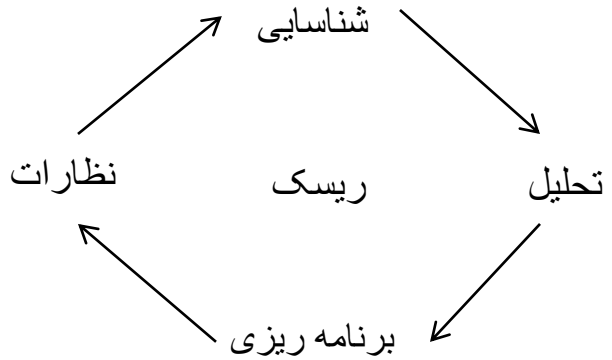
مثال : ریسک ترک برنامه نویس ماهر :

ریسک پروژه : زیرا ممکن است شخص جایگزین نتواند کار را به موقع تحویل دهد .

ریسک محصول : ممکن است شخص جایگزین خبرگی شخص اول را نداشته باشد .

ریسک کاری : زیرا شرکت تولید کننده یک نیروی با تجربه را از دست می دهد .

فرآیند مدیریت ریسک



۱ – شناسایی ریسک

۲ – تحلیل ریسک

۳ – برنامه ریزی برای برخورد با ریسک

۴ – نظارت

همانطور که مشاهده می شود فرآیند فوق تکراری است و این فرآیند تا پایان پروژه ادامه دارد .

۱ – شناسایی ریسک

در این مرحله باید خطراتی را که پروژه را تهدید می کنند را تخمین زده شوند . شناسایی بر اساس تجربه مدیر و استفاده از نظر افراد تیم بدست مب آید

الف) ریسک افراد : ممکن است یکی از افراد تیم مریض شود

ممکن است شخص دیگری به دلیل پایین بودن حقوق شرکت را ترک کند .

ب) ریسک خواسته ها : تغییر خواسته ها باعث طراحی مجدد می گردد .

مشتری تغییر خواسته ها را درک نمی کند .

ج) ریسک ابزار عملکرد ضعیف پایگاه داده

کد تولید شده بهینه نباشد .

د) ریسک برآورد از نظر زمان می توان به تخمین بیش از اندازه اشاره کرد

قیمت پیش بینی شده کم در نظر گرفته شده باشد .

۲ – تحلیل ریسک

در این مرحله احتمال و اثر ریسک بررسی می گردد (طبق تجربه مدیر) . در این مرحله مدیر پروژه موظف است برای کلیه ریسک های شناسایی شده احتمال وقوع و اثر آن را بر اساس پارامترهای زیر مشخص کند

- احتمال وقوع
  - خیلی کم (کمتر از ۱۰ درصد)
  - کم (۲۵- ۱۰ درصد)
  - متوسط (۵۰ – ۲۵ درصد)
  - زیاد (۷۵ – ۵۰ درصد)
  - خیلی زیاد (۷۵ الی ۱۰۰ درصد)
- اثر وقوع
  - فاجعه برانگیز
  - جدی
  - قابل تعمل
  - بی ارزش

مثال : زمان مورد نیاز برای تولید کم برآورد شده است احتمال : زیاد

اثر : جدی

مثال : مشکلات اقتصادی اجازه ی ادامه ی کار را نمی دهد احتمال : کم

اثر : فاجعه بر انگیز

۳ – برنامه ریزی برای برخورد با ریسک

در این مرحله ریسک های کلیدی انتخاب شده و راهبردهایی برای مقابله با آنها تعریف می گردد (با توجه به تجربه ی مدیر)

راهبرد های اتخاذ شده می تواند موارد زیر باشند :

الف) اجتناب از ریسک : این راهبرد به دنبال آن است که احتمال وقوع ریسک را کاهش دهد

به عنوان مثال : جایگزین مولفه های معیوب به جای مولقه های تست شده .

ب) کمینه سازی : در این راهبرد سعی بر کاهش اثر ریسک است (ریسک کاهش بیماری کارکنان) سازماندهی پروژه به طوری که افراد مختلف در پروژه از کار یکدیگر مطلع باشند

ج)برخورد با ریسک: در این راهبرد مدیر منتظر رخ دادن ریسک است ولی از قبل می بایستی برنامه در نظر گرفته شود .

به عنوان مثال اگر ریسک در تخمین هزینه پروژه است ، می توان منتظر این ریسک بود و مستنداتی را آماده کرد تا کار فرما متوجه این امر شود .

۴ – نظارت

در این مرحله مدیر موظف است افزایش و یا کاهش احتمال و اثر ریسک را نظارت نماید و همچنین اگر برنامه ریزی برای هر ریسک در نظر گرفته است ملاحظه نماید که آن برنامه اجرا می شود یا نه

استخراج نیازمندی ها

روش های مختلف استخراج نیازمندی عبارتند از :

۱ – دیدگاه

۲ – سناریو

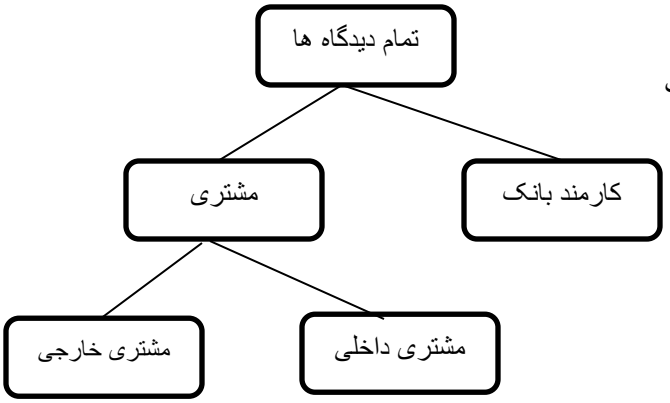
۳ - اتناگرافی

در روش دیدگاه برای تدوین خواسته ها می بایستی مراحل رید را طی کنیم :

۱ – شناسایی دیدگاه ها

اولین گام کشف دیدگاه هایی است که شخص از سیستم دریافت می کند به عنوان مثال در سیستم بانک می توان دیدگاه های زیر را در نظر گرفت : مشتری ، مدیر ، کارمند

۲ – سازماندهی دیدگاه ها



در این مرحله می بایستی دیدگاه های مرتبط به هم را به صورت سلسله مراتبی مرتب نماییم . می بایستی هنگام سازمان دهی توجه داشت که خدمات مشترک در سطوح بالاتر قرار گیرند

۳ – مستند سازی دیدگاه

که شامل توصیف و بیان جزئی خدمات می باشد

روش سناریو

سناریو عبارت است یک سری عملیات پشت سر هم و یا متوالی که انجام آنها منجر به یک عمل خاص می شود به عنوان مثال دانشجو در سیستم آموزش دارای سناریو ثبت نام در نیم سال تحصیلی می باشد . این سناریو توالی اعمال زیر را دارد

برداشت وجه
دلایل : بهبود خدمات مشتری و کاهش کاغذ بازی مشخصات : پس از دریافت مبلغ در صورت موجودی وجه تحویل داده شود . دیدگاه : مشتری

۱ – دریافت برگه ثبت نام

۲ – انتخاب دروس نمی سال جدید

۳ – تائید و و امضاء استاد راهنما

۴ – تائید و مهر دانشکده

۵ – تائیده ی اداره آموزش

۶ – دریافت برگ ثبت نام

توسط دانشجو

هر سناریو شامل موارد زیر می باشد :

۱ – توصیف حالت سیستم در حالت آغاز سناریو

۲ – توصیف جریان عادی رویداد ها در سناریو

۳ – توصیف اشتباهات یا خطاهای احتمالی و چگونگی اداره ی آنها

۴ – توصیف سیستم پس از کامل شدن سناریو

برای مدل سازی سناریو ها در UML از نمودار Use Case استفاده می کنیم

** پروژه : رسم تعدادی Use Case های پروژه

روش دیگر برای استخراج نیازمندی ها اتناگرافی می باشد . این روش برای سیستم هایی پیشنهاد می شود که مشتریان قادر به بیان سیستم دستی نمی باشند .

در این روش تحلیل گر موظف است در محیط سیتسم به عنوان یک کارمند مشغول به کار شود . با این کار خواسته های ضمنی سیستم کشف شده و نهایتا تحلیل گر می بایستی سناریو آنها را تدوین کند .

مدل سازی

خواسته های کاربر به زبان طبیعی بیان می شود . یک روش برای توصیف گرافیکی این خواسته ها از طریق مدل سازی می باشد . مدل سازی می تواند ابعاد مختلف سیستم را مدل کند که برخی از این ابعاد عبارتند از :

۱ – بعد خارجی

۲ – بعد رفتاری

۳ – بعد ساختاری یا بعد معماری سیستم

بعد خارجی حیطه یا محیط سیستم را مدل سازی می کند و در بعد رفتاری گردش اطلاعات و ترتیب فعالیت ها توصیف می گردد . در بعد ساختاری ساختمان داده سیستم بیان می گردد .

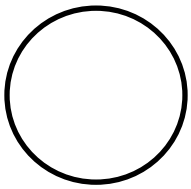
مدل سازی
- ساخت یافته: DFD ERD
- شیء گرا: Use Case Activity Class

مدل DFD یا " Data Flow Diagram " یا نمودار جریان (مدل سازی فرآیند)

توسط DFD گردش داده ها در سیستم توصیف می شود این نمودار به ساختمان داده ی داده ها کاری ندارد بلکه تاکیید بر نوع پذیرش آنها دارد .

DFD شامل نماد های زیر است :

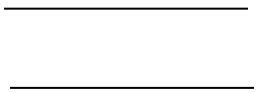
۱ – فرآیند : اعمالی که بر روی داده ها صورت می گیرد



۲ – جریان داده ها : برای توصیف گردش داده در DFD

از این عنصر استفاده می کنیم .

۳ – انبار داده ها : برای نمایش مخزن داده های برنامه



۴ – نهاد های خارجی : افراد یا اشیائی که در خارج از سیستم هستند و اطلاعات را به سیستم وارد یا از سیستم دریافت می کنند .

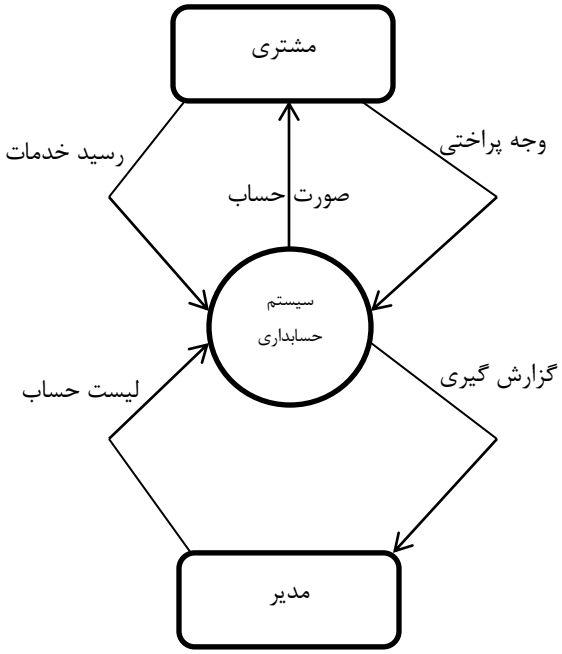
** پروژه : DFD پروژه ها رسم و تحویل داده شود .

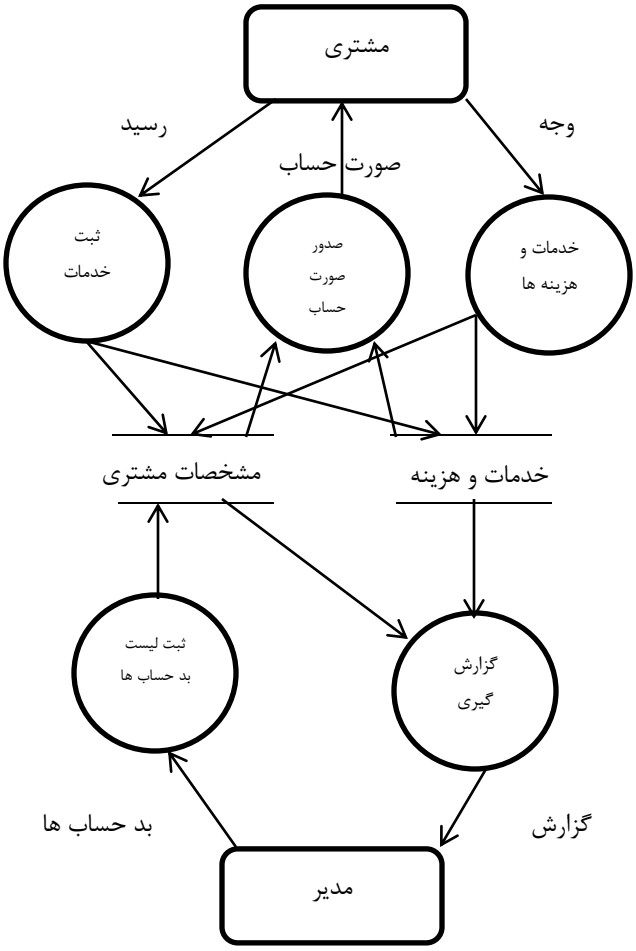
مثال : رسم DFD یک سیستم حسابداری شرکت خدماتی

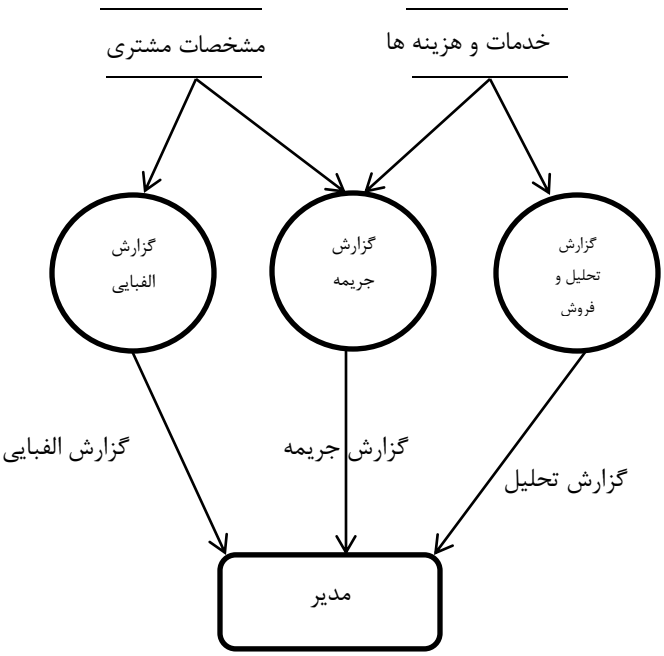
نمودار سطح صفر : در نمودارسطح

صفر فقط از یک تک فرآیند تشکیل

شده است







برخی از اصول ترسیم

۱ – نهاد خارجی : دارا ی نام هستند / معمولا در حاشیه نمودار قرار می گیرند .

۲ – فرآیند : دارای شمارنده اند به جز تک فرآیند سطح صفر / دارای نام مشخص هستند / فرآیند بدون خروجی (چاله سیاه) یا بدون ورودی (معجزه) نباید داشت .

۳ – جریان داده : به غیر از جریان هایی که مبداء یا مقصدشان یک انباره است می بایستی بقیه دارای نام باشند / حداقل یکی از دو جریان داده ی بایستی فرآیند باشد .

دیدگاه شیء گرا سعی دارد تا با نگرش خود به عناصر یک سیستم ، کل سیستم را شبیه سازی کند . دیدگاه شیء گرا بر مباحثی نظیر شیء ، کلاس ، وراثت ، دسترسی ، چند ریختی و ..... استوار است . در این دیدگاه دنیای مسئله به صورت مجموعه ای از اشیاء به هم مرتبط شناسایی شده و آنها را به کلاس های مجزا تفکیک می کند.

```
Class queue {
        Name   q[۱۰۰] ;
        Int    Lloc , Floc ;
        Public :
        Void   inti ( void ) ;
        Void  qput ( int ) ;
        int   qget ( void ) ; } ;
```

```
Void   queue  ::  init  ( void )  {
        Floc = Lloc = ۰ ; } ;
```

```
Void   queue  ::  qput  ( init    i ) {
        If (  Lloc==٠ )  {
                Cout << " queue  is call " ;
        Return ٠ ;}
        Lloc ++ ;
        q [ Lloc ] = i ;}
                ---------------------------------------------------------------------------------
Int     queue  ::  qget ( void ) {
        If ( Floc == Loc ) {
                Cout << " queue  is  empty " ;
        Return ٠;
        Floc ++ ;
        Return  q [ Floc ] ; }
                        ------------------------------------------------------------------
Main () {
        queue  a , b ;
        a . init () ;
        b . init ();
        a . qput (١٠) ;
```

```
b. qput (١٠) ;

a . qput (٢٠) ;

b. qput (٢٠) ;

cout << " a . qget() " ; } ;
```

نوع دیگری از کلاس ها

```
Class  queue  {

int  q [ ١٠٠ ] ;

int  Lloc , Floc ;

int ID ;

public :

queue  ( int  id ) ;

~ queue  ( void ) ;

Void  qput ( int i ) ;

int  qget ( void ) ; }
```

```
queue  :: queue  ( int  id ) {

Lloc = Floc = ٠ ;
```

```
        ID = id ;}

                ----------

Queue  :: ~ queue ( void ) {

        Cout << " queue destroy " ; }

        Defulte  value ;

                ----------

Void  xyout ( char *str  ,  int  x = -١ , int  y = -١ ) {

        If  ( x == -١ )          x = where  x  ;

        If  ( y == -١ )          y = where  y ;

        Goto   ( x, y ) ;

        cout  << " str " ; }

                ----------

Main () {

        Xyout ( " Hello " , ١٠ , ١٠ ) ;

        Xyout ( " test ") ; }
```

زبان های شی گرا این امکان را فراهم می سازد که یک تابع دارای مقادیر قراردادی برای پارامتر های ورودی خود داشته باشد . چنانچه در زمان فراخوانی تابع آرگومانی به تابع ارسال نشود آن مقدار قراردادی مورد استفاده قرار می گیرد .

مثال :

```
Void   F ( int  i = ۱ ) {
        i ++ ;
        cout << " i " ; }
```

```
Main () {
        F ( ۱۰ ) ;
        F ( ۰ ) ; }
```

ارث بری inheritance

امکانی است برای آن که خواص یک کلاس به کلاس دیگر انتقال می یابد . دو کلاس در ارث بری مطرح است :

۱ – کلاس مبنا parent supper

۲ – کلاس مشتق child subclass

هنگامی که یک کلاس خواص کلاس دیگر را به ارث می برد کلیه ی عناصر خصوصی کلاس پدر برای کلاس فرزند غیر قابل دسترسی است .

مثال :

```
Class  x {
        int i ;
```

```
int j ;

public :

void  get – ij ( void ) ;

void  put – ij ( void ) ; }
```

```
Class  y : class x {

int  k ;

public :

int get – k ( void ) ;

void  make – k ( void ) ; }
```

برای دسترسی به این عناصر کافی است دسترسی انها را محافظت شده در نظر بگیریم

شکل کلی ارث بری

```
Class  class name : access class name {

------------------      ------------------------ ;

------------------      ------------------------ ;

------------------      ------------------------ ;

------------------      ------------------------ ; }
```

اگر مقدار access به عنوان privet باشد در هنگام انتقال i و j که در x عمومی هستند به y مقدار خصوصی می گیرند .

کلیه عناصر پدر با همان نوع دسترسی به فرزند منتقل می شوند .

مثال :

```
Class  x {
        Protected :
        int  i ;
        int j ;
        public :
        void  get – ij ( void ) { cin >> i >> j ;}
        void  put – ij ( void ) { cout << " i " << " j " ; }
class y : public  x {
        int k ;
        public :
        int get – k ( void ) { return k ; )
        void make –k { k = i * j }
class ۲ : public y {
        public :
```

```
        void   F  ( void )  { i = ۲ , j =۳ ; }

main () {

        Y   o ;

        Y   o۲ ;

        O . put – ij ( );

        O . make – k ( ) ;

        Cout << o . get – k ( ) ;

        Oz . F( ) ;

        Oz . put – ij ( ) ;
```

مدل سازی با استفاده از UML (unified modeling language) زبان مدل سازی یک پارچه

با استفاده از UML می توان بر اساس مفاهیم شی گرایی طراحی کنیم . زبان UML دارای یک سری نماد می باشد که از چند روش طراحی شیء گرا بدست آمده است (یکپارچه شده است) و دارای تعدادی نمودار است از قبیل :

۱ – نمودار کاربر use case

۲ – نمودار کلاس class diagram

۳ – نمودار رفتار

- نمودار فعالیت activity
- نمودار حالت state chart

۴ – نمودار تعامل

- نمودار توالی sequence
- نمودار همکاری collocation

۵ – نمودار پیاده سازی

- نمودار اجزاء component
- نمودار استقرا deployment

در UML مدل شی گرا ممکن است ارتباط های زیر برقرار باشد :

۱ – تعمیم

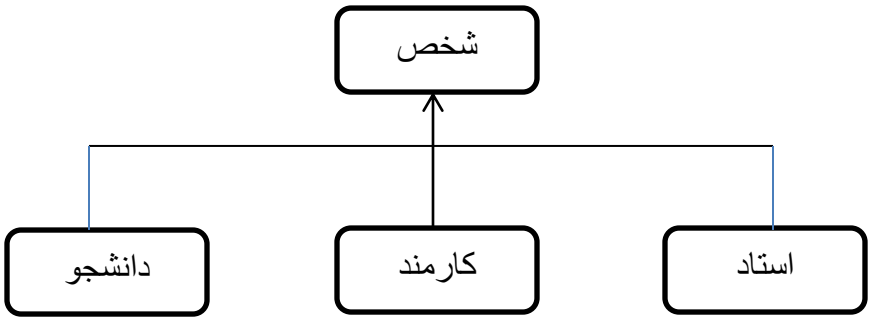
۲ – وابستگی

۳ – تناظر

۱ – تعمیم

تعمیم ارتباطی است که بین یک چیز عمومی با چند نوع خاص تر از آن برقرار می گردد این نوع ارتباط یکی از پایه ای ترین ارتباطات شی گراست که در اکثر نمودار های UML کاربر دارد .

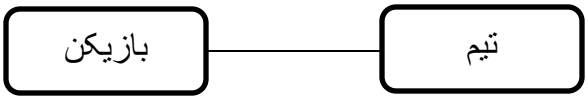
این ارتباط برای پرهیز از تکرار صفات ، اعمال و ارتباط هایی بین کلاس ها در یک نمودار می باشد .



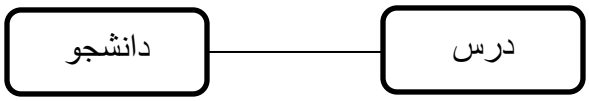
برای این منظور می توان عناصر تکراری را در کلاس مجزا به عنوان کلاس تعمیم و عناصر غیر مشترک را در خود کلاس قرار دهد و با برقراری ارتباط تعمیم بین آنها کلید عناصر کلاس تعمیم را به صورت ارث به کلاس های اخص انتقال داد .

۲ – تناظر

زمانی که ۲ کلاس با یکدیگر ارتباط ساختاری دارند در این حالت ارتباط به عنوان تناظر شناخته می شود .



ممکن است بر روی ارتباط تناظر محدودیت هایی تعریف شود . این محدودیت ها بر اساس قواعد و قوانین مسئله تعریف گردد .



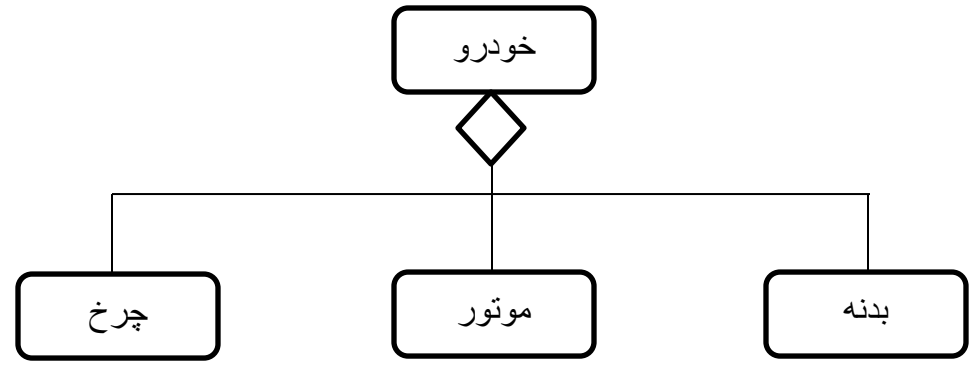
۳ – وابستگی

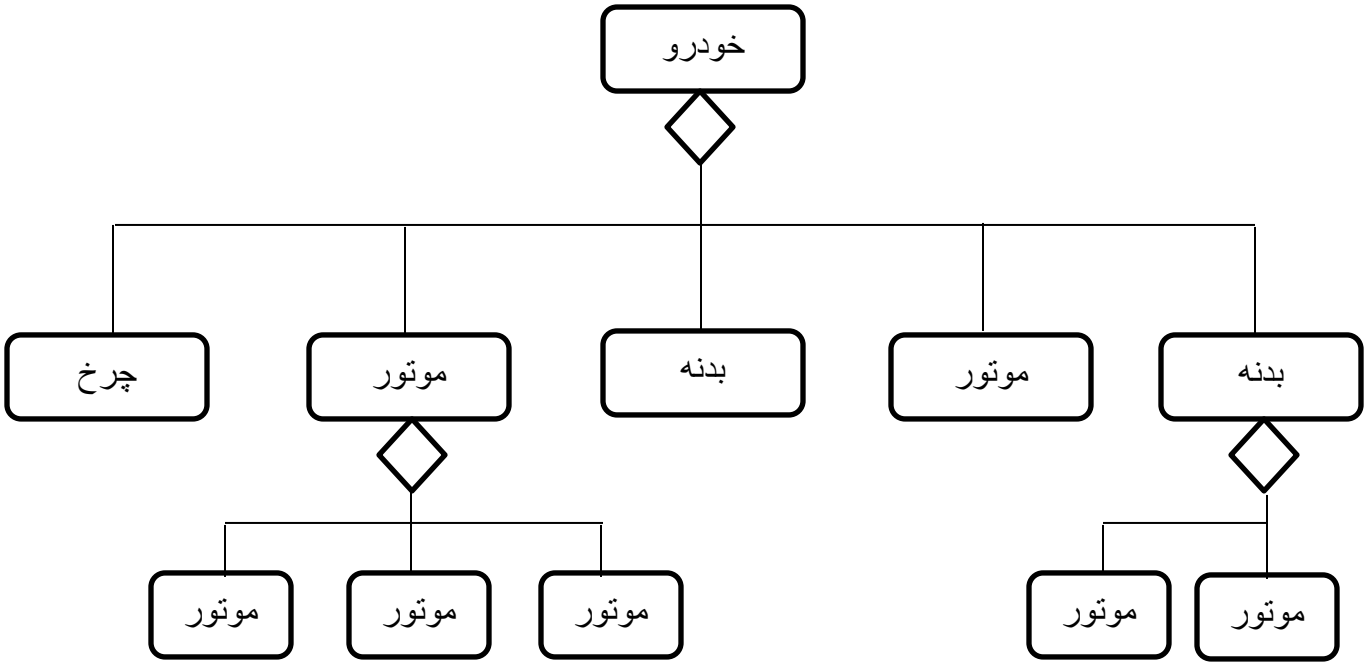
در این نوع ارتباط یک کلاس از کلاس دیگر استفاده می نماید مشخصا زمانی که کلاسی مانند y به عنوان پارامتر یکی از عمل های کلاس x به این کلاس ارسال می گردد می گوییم x و y وابستگی دارند .

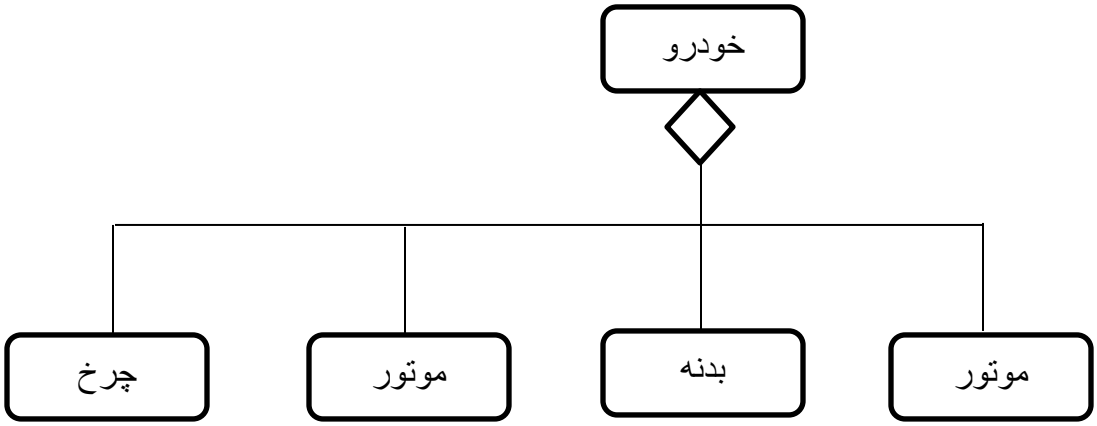
دو حالت خاص در یک ارتباط تناظر می توان در نظر گرفت که به علت متداول بودن به صورت مجزا نماد گذاری شده است . این دو حالت عبارتند از :

۱ – تجمیع ۲ – ترکیب

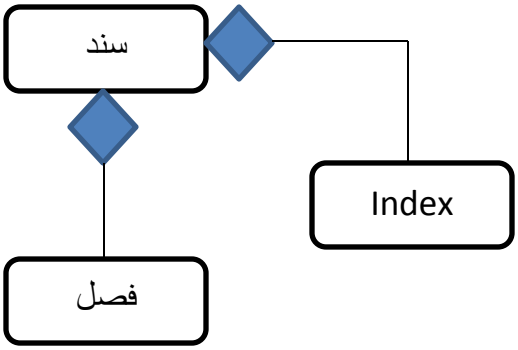
برخی از اشیاء از به هم پیوستن چند قسمت بوجود می آیند در این حالت برای آنکه نشان دهیم مجموع چند قسمت تشکیل یک کل را می دهد از ارتباط تجمیع استفاده می کنیم


خودرو
چرخ
موتور
بدنه


خودرو
چرخ
موتور
بدنه
موتور
بدنه
موتور
موتور
موتور
موتور
موتور


خودرو
چرخ
موتور
بدنه
موتور

ترکیب



ترکیب نوع خاصی از تجمیع است که در این نوع ارتباط با از بین رفتن شی کلی تر اشیاء جزئی اش نیز از بین می رود .

نمودار کلاس

مجموعه از اشیاء که دارای صفات ، اعمال و ارتباطات یکسان هستند در یک کلاس قرار می گیرند . به عنوان مثال کلاس دانشجو دارای صفات : شماره دانشجویی ، نام ، نام خانوادگی و اعمال ثبت نام – حذف و اضافه – درخواست مرخصی دارا می باشد .

محدوده ی نمایش کلاس در UML به صورت زیر نشان می دهیم

نام کلاس
صفت (ها)
عمل (ها)

چند تایی کلاس

چند تایی در کلاس ها تعداد اشیاء تعریف شده از آن کلاس را در سیستم مشخص می کند .

۴۰ ... ۱نام کلاس

حداقل یک و حداکثر ٤٠ تا ٤٠ ... ١

یک یا ٤٠ تا ٤٠ ... ١

حداقل صفر و حد اکثر بی شمار * ... ٠

حداقل صفر و حد اکثر بی شمار *

حداقل ٣شی و حداکثر ٦ یا ٨ شی٨ ... ٣.٦

نکته : کلاس مجرد Abstract class به منظور استفاده از تعمیم این نوع کلاس ایجاد می شود و هیچ شی ء وجود ندارد صرفه به منظور خلاصه سازی و استفاده از خاصیت وراثت مورد استفاده قرار می گیرد .

چندتایی در صفات

چندتایی یا کاردینالیتی در صفات یک کلاس می توان در نظر گرفت

مثال : port [۲ ... *] : هر شی از این کلاس حداقل ۲ و حداکثر بی شمار port دارد .

فرمت کلی صفت ها در نمودار کلاس

Visibility Name [Cardinality] : Type = Initial Value { Property }

Visibility
- \+ صفت دارای دسترسی Public
- \- صفت دارای دسترسی Private
- # صفت دارای دسترسی Protected
- ~ صفت قابل رویت در دسته ی (Package) آن کلاس می باشد .

Type : بیان گر نوع و ماهیت صفت است مانند : float , string , Date

برخی از صفات را قصد داریم در لحظه ایجاد شی به صورت پیش فرض دارای مقدار خاصی باشد در این حالت می توانیم با مقدار دهی اولیه این موضوع را مشخص کنیم (با دیدن یک مقدار اولیه برنامه نویس باید یک Constrictor را ایجاد گند)

Changeable

Property Add only

Frozen

۱ – هیچ محدودیتی برای تغییر یافتن این صفت وجود ندارد (Changeable

۲ - برای صفاتی که چند تایی بیش از یک دارند مورد استفاده قرار می گیرند و بیان گر آن است که هر شی از این کلاس پس از گرفتن مقداری قابل حذف نیست بلکه فقط می تواند اضافه گردد (Add only)

مثال : درس پاس شده در کلاس دانشجو

۳ – پس از اولین باری که مقدار دهی می گردد هرگز قابل تغییر نمی باشد مانند شماره دانشجویی (Frozen)

فرمت کلی اعمال در نمودار کلاس

Visibility Name (Parma List) : Return – Type

فهرست پارامتر های Method با فرمت زیر در این بخش ذکر می شود :

Direction Name : Type = Default Value

Default Value : مقدار پیش فرض پارامتر را در صورت نداشتن آرگومان به تابعی منتقل می شود

نمودار Use case

با استفاده از Use case می توانیم نیاز های کاربر را مشخص کرده بدون آنکه نحوه ی انجام آن مشخص باشد زیرا کاربر علاقه مند است که نیازهایش برآورده شود و علاقه ای ندارد بداند چگونه و با چه روشی آنها انجام گیرند .

مثال : در سیستم بانک ، مشتری از سیستم انتظار دارد که بتواند از حساب خود مبلغی را به حساب شخصی دیگری انتقال دهد .

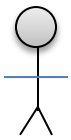
نکته : یک Use case می تواند شامل یک یا چند سناریو باشد .

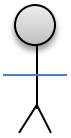
در نمودار Use case برای نمایش تعامل بین سیستم و خارج از سیستم از کنشگر یا Actor استفاده می شود .

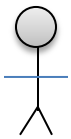
Actor : نقشی که کاربر در ارتباط با سیستم ایفا می کند (توجه به نقش است نه شخص) مثلا یک نفر ممکن است رئیس بانک و مشتری بانک نیز باشد .

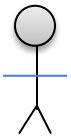
باید توجه داشت که Actor الزاما انسان نیست بلکه می تواند سیستم نرم افزار دیگری و یا یک سخت افزار باشد

نماد Actor :









جریان واقع یک Use case عبارت است از :

۱ – یک Use case کی و چگونه پایان می باید ؟

۲ – چه زمانی تعامل با کنشگر صورت می گیرد ؟

۳ – چه اشیائی در چه زمانی مورد مبادله قرار می گیرند ؟

۴ – چه روال های اصلی و چه روال های استثنائی وجود دارد ؟

مثال : سیستم بانک ، Use case : احراز اعتبار مشتری

الف) Use case با درخواست از مشتری برای وارد کردن Pin code کار را شروع می کند

ب) مشتری کد را وارد کرده و دکمه ی Enter را وارد کند .

ج) Pin code وارد شده توسط سیستم کنترل می شود تا از اعتبار مشتری اطمینان یابد .

د) در صورت اعتبار ، سیستم خوش آمد گویی می کند و Use case پایان می یابد

غیر از جریان اصلی فوق می توان چند جریان استثنائی نیز داشت :

۱ – کاربر در هر زمان که خواست بتواند کلید Cancel را زده و از ابتدا شروع کند

۲ – هنگامی که تعداد Pin code وارد شده را ۳ بار متوالی اشتباه وارد کرد سیستم تا یک روز سرویس ندارد

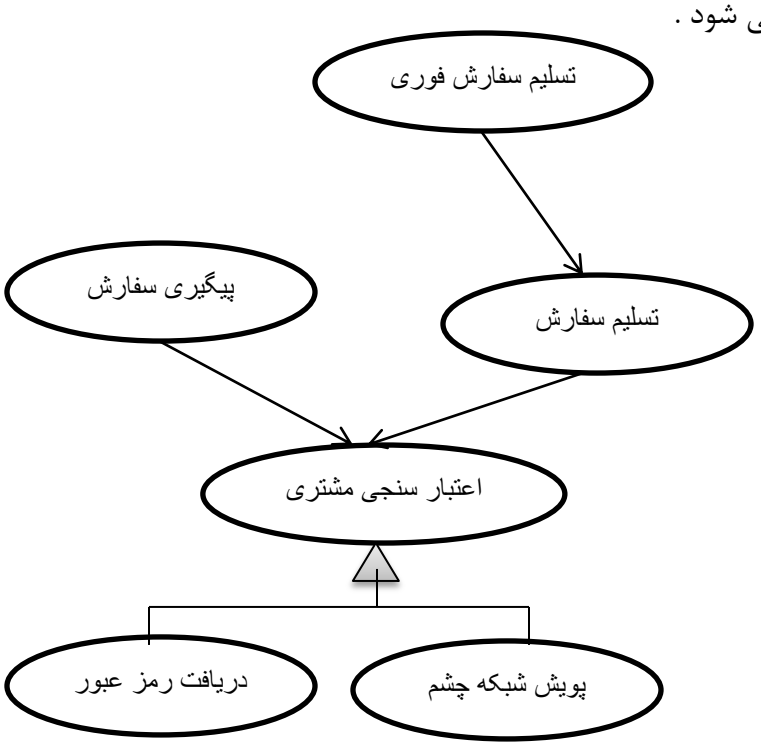
سازماندهی

برای مدیریت بهتر Use case ها میتوانیم آنها را دسته بندی کنیم واز طریق رابطهای زیرارتباط بین آنها را ایجادکنیم :

۱ – تعمیم (Generalization) : در Use case ها همانند تعمیم در کلاس ها است در این حالت نیز Use case فرزند رفتار هایی را از Use case وارث به ارث می برد

۲ - توسعه (Extend) : جریان های استثنائی که حالت خاصی را نشان می هند با استفاده از رابطه Extend به Use case که استثنائی آن به حساب می آید وصل می شود .

۳ - شمول (Include) : برای آنکه از تکرار Use case های یکسان در سیستم جلوگیری شود رفتاری که بین Use case مشترک است . در یک Use case جداگانه قرار گرفته و تمامی Use case های آن رفتار را به کار می برند با رابطه Include با آن مرتبط می شوند



پایان

تاریخ : ۰۸ / ۰۳ / ۱۳۹۱

خدمت استاد محترم : آقای رمزی

تهیه کننده : مسعود کوثری