مهندسی: در هر رشته درباره ساخت محصول و موارد مرتبط بحث میکند.

مهندسی نرم افزار: درباره ساخت محصول نرم افزاری و موارد مرتبط بحث میکند.

انواع محصولات نرم افزاری ←

- **محصولات کاربردی:** Applcation برای انجام وظایف خاص بکار میرود. (سیستم حسابداری ،بازی ،photoshop،office)
- **محصولات سیستمی:** برای عموم کاربرد ندارند (windows،کامپایلر #c)

System Life Cycle
چرخه حیات سیستم

مراحل تولید محصولات نرم افزاری

1-شناخت سیستم و شناخت امکان سنجی
1-شناخت سیستم دستی موجود
2- بررسی اینکه کدام بخشها از سیستم دستی قابل ماشینی شدن است و کدام بخشها نیست
3-ارائه هزینه انجام پروژه بصورت تخمینی

2-تحلیل: (Analysis) مهندسی نیازها انجام میشود (نیازها شناسایی شده و به صورت مهندسی مورد بحث ،بررسی و مدل سازی قرار میگیرد)

3-طراحی: (Design) طرح و نقشه تولید محصول مکانیزه (محصول ماشینی) ارائه میشود.

4-پیاده سازی: (Implementation) در این مرحله کدنویسی ،ایجاد پایگاه داده ،خرید و نصب تجهیزات سخت افزاری و .....انجام میشود.

5-تست یا آزمایش محصول: توسط یک تیم مستقل برای رفع اشکالات احتمالی انجام میشود.

6-نگهداری و توسعه سیستم: هرسیستم مکانیزه بایستی نگهداری شود و در صورت نیاز توسعه داده شود(افزودن امکانات جدید)

اجزاء مدل نرم افزاری

1-کلاس: (Class) قالب ایجاد شی است و به یک رده از اشیاء ،پدیده ها و....اطلاق میشود.

1-صفات (Properties) صفت یک ویژگی از کلاس است که مقداری را در خود ذخیره میکند و یا بیان میکند.

2-متدها (methods) یک رفتار از یک کلاس را بیان میکند.

انواع کلاس

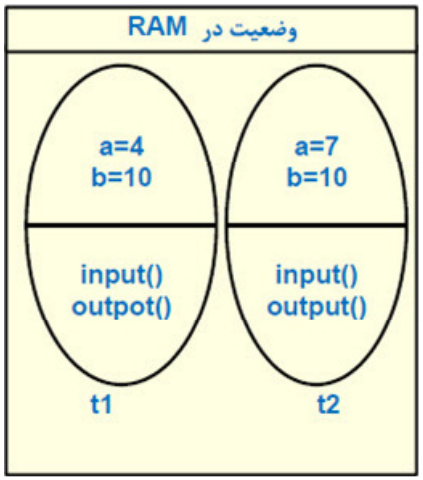
1-کلاسهای فیزیکی: با چشم قابل رویت هستند(قالب شیرینی،قالب گچ بری)

2-کلاسهای انتزاعی (مفهومی): با چشم قابل رویت نیستند و فرض میشود که وجود دارند (کلاس دانشجو)

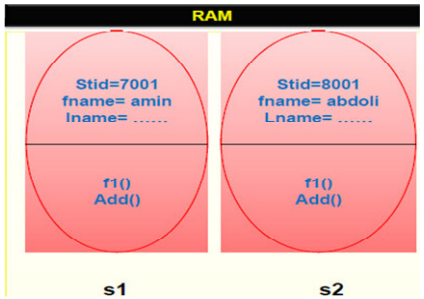
2-شی: (Object)

نمونه ای از کلاس است (توسط کلاس ساخته میشود)
اشیاء کامپیوتری در RAM ساخته میشوند و همان جا نیز از بین میروند.
مقادیر صفات باعث تمایز بین اشیاء میشود.
از یک کلاس میتوان N شی ساخت.
کلاسها کد نویسی میشوند.
هنگام ساخت یک شی توسط یک کلاس همه صفات و متدهای کلاس مورد نظر به شی منتقل میشود.

```
#include <iostream.h>
#include <conio.h>
class test{
          int a ;
          int b ;
    public:
      void input(){
          cout<<"Enter number:\n";
          cin>>a>>b;
      }
      void output() {
          cout<<"a*b"<<a*b;
      }
} ;
// end of class definition
int main(){
        test t1,t2;
        t1.input();
        t2.input();
        t2.output();
        t1.output();
        getch();
        return 0;
}
```




```
Student   s1 , s2 ;
s1.stid = 7001;
s1.fname = 'Amin';
s2.stid = 7001;
s2.fname = 'Abdoli';
s1.Add();
s2.f1();
```



روشهای مهندسی نرم افزار

1-روشهای قدیمی (ساختیافته ،آبشاری،....)(SSAD) معمولا بر اساس نیازهای شناسایی شده برنامه ها تهیه میگردند.

2-روشهای شی گرا: (Object Oriented)

ایجاد یک مدل کوچک نرم افزاری از مدل واقعی به طوری که تعامل اجزای این مدل ،نیازهای شناسایی شده را پاسخ دهد .بعنوان مثال در یک سیستم آموزشی اشیایی از کلاس دانشجو ،درس،استاد،فضای آموزشی ،کارمند آموزش .....با هم در تعاملند تا نیازهایی مانند ثبت نام ،انتخاب واحد ،صدور کارنامه و.... برآورده شود.

ارکان شی گرایی

1-کپسوله سازی: (Encapsulation)

1-اختصاصی: (Private) صفت یا متدی که با این سطح دسترسی تعریف میشود فقط در کد نویسی کلاس خودش قابل دسترسی است.

2-محافظت شده: (Protected) در کد نویسی کلاس خودش و در کد نویسی کلاسهای فرزند خودش قابل دسترسی است.

3-عمومی: (Public) در کد نویسی کلاسهای فرزند کلاس خودش و برنامه های کاربردی دیگر از جمله تابع main() فابل دسترسی خواهد بود.

نکته:همین سطوح دسترسی در خود کلاسها نیز استفاده میشوند.

2-وراثت: (Inheritance)

صفات و متدهای مشترک از یک گروه کلاس مرتبط به هم استخراج شده ودر یک کلاس بالاتر قرار میگیرند تا اهداف وراثت محقق شود.

1-کاهش حجم کدنویسی:

2-سازماندهی یک گروه کلاس مرتبط به هم:

هنگام ساخت شی توسط یک کلاس فرزند ،علاوه بر صفات و کمتدهای کلاس فرزند ،صفات و متد های والد کلاس فرزند ،صفات و متد های والد والد کلاس فرزند و .....به آن شی منتقل میشوند.

3-چند ریختی: (Polymorphism)

انجام چند عمل با یک عنوان را چند ریختی میگویند،هدف از بکار گیری چند ریختی کاهش تعداد عناوین است.

بطور مثال با تعریف مجدد عملگر + میتوان علاوه بر جمع دوعدد ،دو رشته را بهم چسباند.

پیاده سازی وراثت

```
class person {
   .
   .
   .
};

class student: public person {
        student پیاده سازی کلاس
};

class worker: public person {
        worker پیاده سازی کلاس
};

class employee: public person {
        Employee پیاده سازی کلاس
};
```

در تولید نرم افزار علاوه بر هدف تولید ،کاهش زمان و هزینه دو عامل مهم برای تولید هستند.

ویژه گیهای نرم افزاری و سخت افزاری

Hardware(manufacture) استفاده از ماشین برای ساخت کالا یا مواد (چیزی که از قبل وجود نداشته است)

Software(Develop) توسعه چیزی برای بهبود و پیشرفت دادن آن (سیستم دستی توسعه ماشینی داده میشود)

به قطعات نرم افزاری از قبل آماده Component گفته میشود.(عمدتا در محیطهای ویژوال بصورت ocx)

نرم افزار فقط برنامه های کامپیوتری نیست ،بلکه تمام مستند سازی ها و داده های پیکره بندی (داده راهنما برای رسیدن به داده اصلی MetaData) را شامل میشود که برای درست کارکردن این برنامه ها ضروری اند.

مهندسی نرم افزار یک نضام مهندسی(بکارگیری تئوریها ،روشهاو ابزار ها در جای مناسب و در نظر گرفتن محدودیتهای سازمانی و عملیاتی)که با تمام جنبه های نرم افزاری محصول(مدیریت پروژه های نرم افزاری) از مراحل اولیه تعین مشخصات سیستم تا نگهداری سیستم سرو کار دارد.

انواع محصولات نرم افزاری

1-محصولات کلی: سیستمهای مستقلی که توسط یک سازمان تولید کننده ایجاد و به بازار عرضه میشود(DBMS،واژه پردازها)

2-محصولات سفارشی: سیستمهایی که توسط مشتری سفارش داده میشوند(سیستمهای کنترل ترافیک هوایی،...)

از نظریات متخصصین (Software Myths) در راهبری و تولید نرم افزار استفاده نمیشود بلکه به واقعیات توجه میشود.

فرآیند نرم افزار:(Software Process) مجموعه ای از فعالیتها و نتایج مربوط به آنهاست که یک محصول نرم افزاری را تولید می نماید.

فرآیند نرم افزار

تعیین مشخصات نرم افزار: وظایف نرم افزار و محدودیتهای عملیات آن را تعیف می نماید.

توسعه نرم افزار: تولید نرم افزاری با مشخصات تعیین شده

اعتبار سنجی: تضمین اینکه آیا خواسته های مشتری را برطرف می نماید.

تکامل نرم افزار: افزودن امکان جدید به محصول تولید شده

مدل فرآیند

فرایند — تعریف پاسخی برای what(چه چیزی را نیاز داریم) . مانند زبانهای امری C

فازها — تولید How (چطور به انجام برسانیم.مانند زبانهای منطقی و امری که هم what و هم How را پشتیبانی میکنند (prolog)

ابزارها — پشتیبانی با تمرکز بر عملیات (بتوانیم امکان جدید به محصول تولید شده اضافه کنیم)

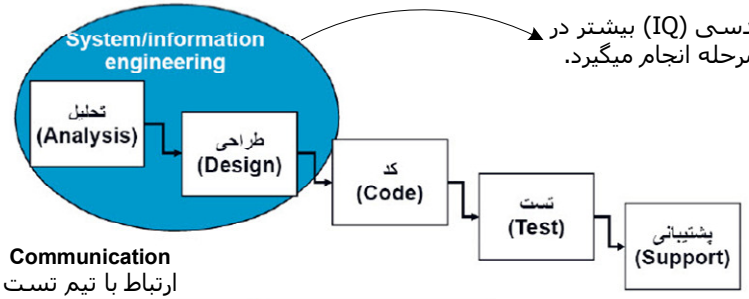
متدها (روشها)

مدلهای فرآیند

مدل خطی-ترتیبی(آبشاری)

در این مدل که یک فرآیند طراحی متوالی است مراحل تولید محصول نرم افزاری بصورت پشت سرهم انجام میشود ،برگشت از مرحله فعلی به مرحله قبلی با مشکلاتی همراه است.

معایب: 1:پروژه واقعی به ندرت جریان ترتیبی این مدل را دنبال میکند . 2:کاربر نمیتواند در یک مرحله به وضوح نیازهای خود را بیان کند. 3:مشتری برای دریافت یک نسخه از پروژه در روزهای آخر کاری کم حوصله است.



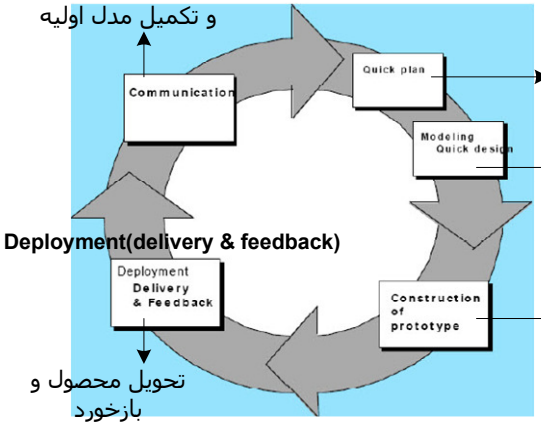
کار مهندسی (IQ) بیشتر در این دو مرحله انجام میگیرد.

مدل نمونه سازی (Prototyping)

بر ساخت نمونه اولیه محصولات نرم افزاری تاکید دارد و در مراحل بعدی نمونه اولیه کامل میشود.بعبارت دیگر در هر مرحله مقداری محصول نرم افزاری توسعه داده میشود.

معایب: 1:مشتری به آنچه از سیستم که کار میکند نگاه میکند و به مسائلی مانند کیفیت یا امکان پشتیبانی از سیستم فعلا نادیده گرفته شده توجه نمیکند و برای کمتر هزینه کردن میخواهد همان نمونه را کامل کنیم و تحویل دهیم. 2:عموما از زبان پیاده سازی نامناسب و الگوریتمهای ناکارا به علت سرعت در تولید نمونه استفاده میشود و بعد از مدتی علت استفاده فراموش میشود و بازگشتی برای تصحیح آن انجام نمیشود.

Communication
ارتباط با تیم تست و تکمیل مدل اولیه



Quick plan
برنامه ریزی ساده (تحلیل)

Modeling (quick Design)
طراحی ساده

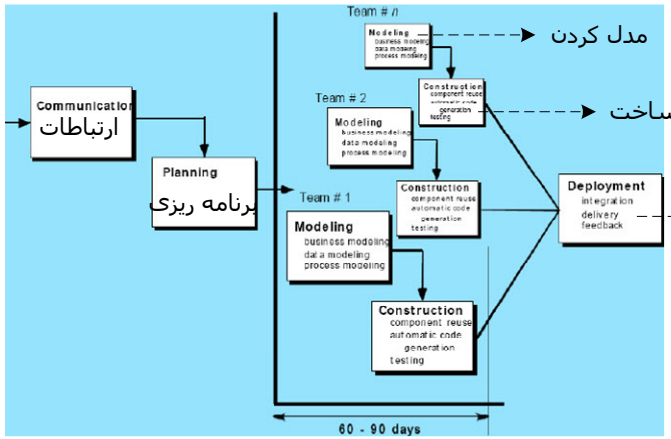
Construction of prototype
نمونه اولیه ساخته یا تکمیل میشود

Deployment(delivery & feedback)
تحویل محصول و بازخورد

مدل نمونه سازی سریع (RAD)

تقریبا مشابه روش مدل سازی نمونه است با این تفاوت که بخشهای مستقل پروژه توسط تیمهای مختلف به موازات هم توسعه پیدا میکنند.در این روش یک برنامه ریزی حداقل برای تولید سریع نمونه اولیه انجام میشود و توسعه Plan در حین تولید محصول صورت میگیردو برنامه ریزی مختصر اجازه میدهد که نرم افزار خیلی سریع تولید شود ،تغییرات و توسعه بعدی در آن بر اساس نیازمندیها بطور ساده انجام میشود.

معایب: 1:در پروژه های بسیار بزرگ نیاز به منابع انسانی زیاد است 2:کاربردهای غیر قابل تقسیم برای RAD مناسب نیستند 3:اگر احتمال بروز خطا بالا باشد RAD مناسب نیست (چون تیمهای مختلف با این خطاها درگیر میشوند و کنترل ساده نیست)



مدل کردن

ساخت

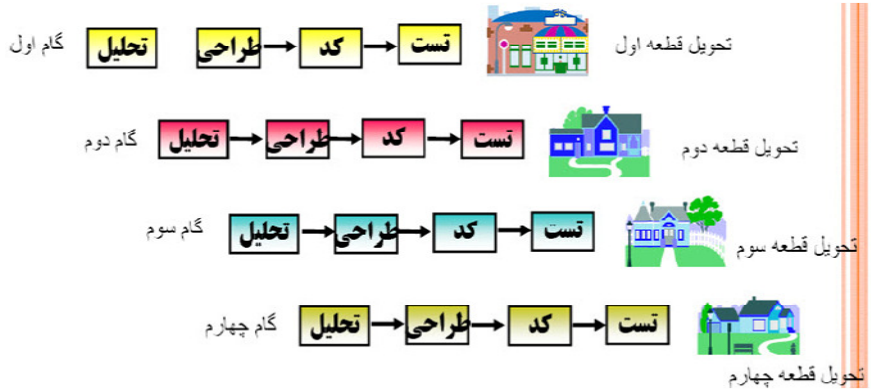
استقرار

Modeling	Deployment	Construction
Business model	Integration	Component reuse
Data model	Delivery	Automatic code
Process model	Feedback	Generation
		Test

مدل گام به گام

در این روش در هر مرحله یک بخش تقریبا مستقل از محصول نرم افزاری (تحلیل ،طراحی ، پیاده سازی ،آزمایش و تحویل) داده میشود.برای پروژه های قابل اعمال است که دارای قطعات مستقل باشند.

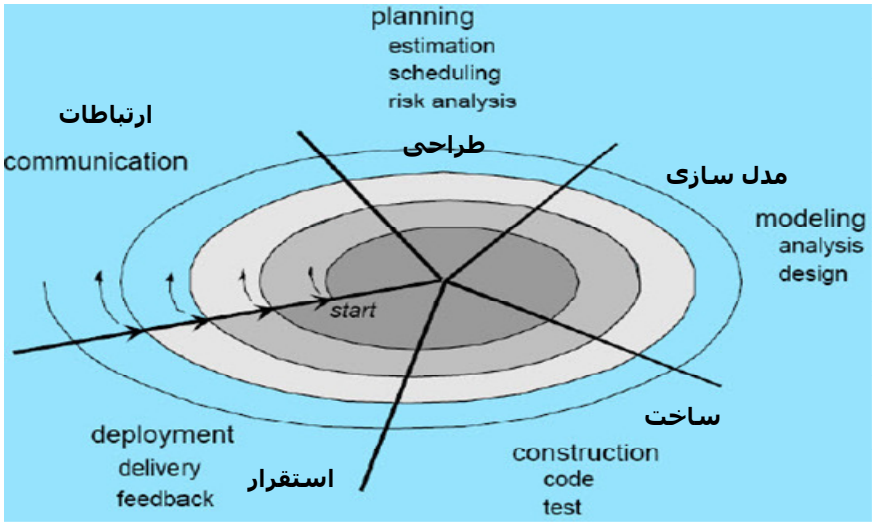
معایب : معایب این روش مانند معایب مدل RAD میباشد.



مدل مارپیچی

برای پروژه های گران و پیچیده استفاده میشود ،بصورت تکراری مراحل تولید محصول را توسعه میدهد و در هر تکرار وسعت مراحل تولید افزایش می یابد .

معایب: 1-سختی قانع کردن کاربر برای کنترل روش تکاملی 2:نیاز و اتکاء زیاد به مهارت ارزیابی 3:تغییر در دیدگاه مدیریت و شک به فرآیند مارپیچ



Communication	Deployment	Construction	Modeling	Planning
ارتباط و ارزیابی	تحویل Delivery	کد نویسی Code	تحلیل Analysis	تخمین زدن Estimation
	گرفتن بازخورد Feedback	تست Test	طراحی Design	زمانبندی Scheduling
				تحلیل ریسک Risk analysis

تبدیل رسمی (Formal Transformation)

مدل کردن بصورت ریاضی و تبدیل به نرم افزار با روشهای ریاضی مانند استفاده زبانهای منطقی برای تولید Firewall

روش مبتنی بر مولفه (Component -based method)

فرض بر این است که بخشهایی از سیستم فعلا موجود است و از طریق مونتاژ قطعات از قبل آماده سیستم توسعه داده میشود.

Agile Development Method

گروهی از روشهای توسعه نرم افزار است مبتنی بر تکرار و افزایش که نیازمندیها (what) و راه حل ها (who) در آن از طریق همکاری بین تیمهای خود سازمانده (خود مدیریت) ارائه میشود.

روش برنامه نویسی افراطی (Extreme Programming)

با تاکید بر توجه بیش از حد به برنامه نویسی سعی در افزایش کیفیت محصول و پاسخگویی مناسب به نیازهای مشتری دارد و به نوعی Agile Development است.

CASE(Computer-Aided Software Engineering)

مهندسی کامپیوتر به کمک کامپیوتر که سرعت ،کارائی و عملکرد مناسب خواهد داشت.

Upper CASE فقط در تحلیل و طراحی بکار میروند.

Lower CASE برای پشتیبانی از پیاده سازی و تست طراحی شده اند.

متدولوژی های(روشمندی در تولید) مهندسی نرم افزار

یک رهیافت برای توسعه نرم فزار است که نرم افزاری کیفی با هزینه مناسب را طراحی می نماید،که با استفاده از مدلهای مختلف فرایند به تولید محصول میپردازد.

ساخت یافته (Structural)

مبتنی بر نیاز کار میکند.

عملکرد گرا(Function Oriented)

رویکردی بین رویکرد ساخت یافته و شی گرا میباشد که ایده های هر دو رویکرد در آن استفاده شده است ،فاصله زمانی ظهور و مرگ این رویکرد بسیار کوتاه است.(مناسب برای سیستمهای بی درنگ Real time)

شی گرا (Object Oriented)

این روش مبتنی بر مدل کار میکند ،یک مدل کوچک نرم افزاری از روی مدل واقعی ساخته میشود که تعامل اجزای این مدل که همان اشیاء کلاسها هستند نیازهای مورد نظر را براورده میکنند.

سیستم

مجموعه ای هرفمند از مولفه های (اجزاء)مرتبط به هم است که با هم کار میکنند تا هدفی را رآورده سازند.

مهندسی سیستم (System Engineering)

فعالیتی برای تعیین مشخصات ،اعتبارسنجی،استقرار و نگهداری کل سیستم است.

مهندسین سیستم به غیر از نرم افزار با سخت افزار و تعامل سیستم با کاربران و محیط آن سروکار دارند.

فرآیند مهندسی سیستم

تعریف خواسته های سیستم (Requirement Definition)

خواسته های عملکردی انتزاعی عملکردهای اساسی که باید توسط سیستم انجام گیرد در یک سطح انتزاعی تعریف میشوند مانند ذخیره بانک اطلاعات (ذخیره جدول)

ویژه گیهای سیستم جزء خواص غیر عملکردی سیستم میباشد (شامل کارایی ،امنیت و غیره)

خواصی که سیستم نباید از خود نشان بدهد مانند ایجاد محدودیت در نرم افزار

طراحی سیستم

-مشخص مینماید که عملکرد سیستم چگونه باید توسط مولفه های مختلف سیستم انجام شود و فعالیتهای آن عبارتند از(تقسیم بندی خواسته ها،تشخیص زیر سیستم ها،انتساب خواسته ها به زیر سیستم ،مشخص کردن عملکرد زیر سیستم ،تعریف واسطهای زیر سیستم)

توسعه زیر سیستم (Requirement Definition)

-زیر سیستم هایی که در مرحله طراحی شناسایی شده انددر مرحله توسعه سیستم ،پیاده سازی میشوند.

-گاهی فرایند توسعه (Development Process) تمام زیر سیستمها را از اول توسعه میدهد.

-گاهی برای سیستم های اقتصادی از زیر سیستم های آماده (COTS) استفاده میشود.Commercial Of The Self سیستمهای آماده ای هستند که معمولا ارزانتر از توسعه سیستم میباشند و ممکن است دقیقا خواسته ها را براورده نکند.

جامعیت زیر سیستم (System Integration)

-صحت ،درستی و کامل بودن سیستم از نظر عملکرد در تمام لحظات حیات زیر سیستم را جامعیت زیر سیستم گویند .از نظر تکنیکی و مدیریتی جامعیت تدریجی مناسب تر است و در اینحالت در هر زمان یک زیر سیستم در سیستم جامعیت پیدا میکند.

-در رهیافت BigBang تمام زیر سیستمها همزمان جامعیت پیدا میکنند.

-جامعیت تدریجی به دو دلیل از BigBang مناسبتر است.

1-نمیتوان توسعه زیر سیستمهای مختلف را همزمان به اتمام رساند.

2-جامعیت تدریجی ،هزینه یافتن محل خطا را کاهش میدهد.

نصب سیستم (System Installation)

-سیستم درطی نصب در محیطی قرار میگیرد که برای آن طراحی شده است.

تکامل سیستم (System Evolution)

-طول عمر سیستم های بزرگ و پیچیده معمولا بالاست و در طی چرخه حیات این سیستمها خواسته های اصلی سیست تغییر میابد و خواسته های جدید مطرح میشود.

-تکامل سیستم به دلایل زیادی هزینه بر است مانند (تغییرات از نظر اقتصادی و تکنیکی باید تحلیل شود ،تغییر در یک سیستم بر روی زیر سیستمها تاثیر میگذارد و....)

نیازمندیها (Requirements)

-بیشتر نیاز های کاربر مورد نظر میباشد

-تعریف نیاز کاربر **(user requirement definition)**

-مشخثات نیازهای سیستم **(system requirement specification)**

نیازهای عملکردی و غیر عملکردی

نیازهای عملکردی (Functional Requirements)

خدماتی که سیستم باید ارائه نماید مانندسرویس ثبت نام که توسط وب سایت سایت سازمان سنجش ارائه میشود.

نیازهای عملکردی (Non-Functional Requirements)

-محدودیت در خدماتی که توسط سیستم پیشنهاد میشود. و به ویژه گیهایی از سیستم مانند قابلیت اعتماد ،زمان پاسخ و محل استقرار مربوط میشود.

نیازهای دامنه کاربرد

ویژه گیهای دامنه کاربرد سیستم را منعکس مینماید مثلا افزایش ساعت آموزش در سیستم آموزشی توسط کاربر

امنیت یک نیاز غیر عملکردی است اما وقتی با جزئیات بیشتری توسعه می یابد منجر به نیاز عملکردی است(مانند امکان تایید کاربر)

نیازهای عملکردی کاربر بصورت کلی توصیف میشوند اما نیازهای عملکردی سیستم عملکرد تفضیلی سیستم را توصیف مینمایند.

نیازمندیهای غیر عملکردی نسبت به نیازمندیهای عملکردی ،حیاتی تر (Critical) میباشند.

برای بدست آوردن نیازهای دامنه کاربرد با کاربران و نیازمندیهای آنها کاری نداریم.

خوانندگان نیازها به سه نیازمندی مرتبط هستند و از هم تفکیک میشوند (نیازهای کاربر ،نیازهای سیستم ،مشخصات طراحی نرم افزار)

شکست در براورده شدن خواسته های عملکردی منجر به نقص سیستم میشود ولی شکست در خواسته های غیرعملکردی ممکن است کل سیسنم را بلااستفاده نماید.

نیازهای سیستم (System Requirements) نیازهای سیستم بیان میکنند که سیستم چه کارهایی را باید انجام دهد.

سند نیازها (Requirements Documents) بیان رسمی از احتیاجات توسعه دهنده سیستم است.

کاربرانی که با سند نیازهای نرم افزار در ارتباطند عبارتند از 1-مشتریان 2-مدیران 3-مهندسین سیستم 4-مهندسین تست سیستم 4-مهندسین نگهداری سیستم

فرایند مهندسی نیازها (Requirements Engineering) شامل کلیه فعالیتهای مورد نیاز برای ایجاد و نگهداری سند نیازهای سیستم است.

چهار فعالیت عمومی مهندسی نیازها

1-مطالعه امکان سنجی (Feasibility Study)

شامل براورد اطلاعات ،جمع آوری آنها و نوشتن گزارش است که تعیین میکند آیا فرایند توسعه سیستم ارزشمند است یا خیر.

2-استخراج و تحلیل گزارشها (Requirements Elications and Analysis)

توسعه دهندگان نرم افزار با مشتریان در ارتباطند تا سرویسهای سیستم و محدودیتهای عملیاتی سیستم مشخص شود.

به افرادی که در فرایند استخراج و تحلیل نیازها درگیر میشوند Stackeholder میگویند. از مشکلات تحلیل نیازها و stakeholder ها اینست که (نمیدانند چه چیزهایی را حقیقتا نیاز دارند ،خواسته های خود را به زبانهای متفاوتی بیان میکنند ،ممکن است نیازهای متضادی داشته باشند ،عوامل سازمانی و سیاسی ممکن است نیازها را تحت تاثیر قرار دهد،نیازها ممکن است در حین فرایند تحلیل دچار تغییر شوند)

3-مشخصات نیازها (Requirement Specification)

4-اعتبار سنجی نیازها (Requirement Validations)

بررسی میشود که آیا نیازهای کشف شده ،سیستم مورد نظر مشتری را تعریف میکند یا خیر.

هزینه برطرف کردن خطای نیازمندی پس از راه اندازی یک سیستم ، 100 برابر هزینه برطرف کردن خطای پیاده سازی است.

تکنیکهای اعتبارسنجی نیازها

مرور نیازها :نیازها بطور منظم توسط تیم مرورگر تحلیل شود.

ساخت نمونه اولیه:به کمک مدل اجرایی سیستم نیازمندیها چک میشود

تولید موارد تست :تولید تست هایی برای نیازها

فرآیند استخراج و تحلیل

1-کشف نیازها 2-دسته بندی نیازها 3-مذاکره و اولویت بندی نیازها 4-مستند سازی نیازها

کشف نیازها (خواسته ها)

دیدگاه ها (Viewpoints)

پیش زمینه یا پس زمینه در مورد موضوعی را دیدگاه میگویند.

مصاحبه (Interview)

مصاحبه بسته (Closed Interview) هر stakeholder به پرسشهای از پیش تعیین شده پاسخ میدهد.

مصاحبه بسته (Open Interview) پرسشهای از پیش تعیین شده وجود ندارد .در ارتباط با stakeholder نیازها شناسایی میشود.

مصاحبه ها ،تکنیکهای مناسبی برای استخراج اطلاعات درباره نیازهای سازمانی دامنه کاربرد نمیباشد.

سناریو (Scenarios)

چگونگی روند (داستان)انجام یک واقعه (عملیات) را میگویند.

توصیف وضعیت سیستم در ابتدا (پیش شرایط precondition)

توصیف جریان عادی رویدادها در سناریو (پروسه انجام عملیات سناریو)

توصیف اشتباهات احتمالی

اطلاعاتی راجع به دیگر فعالیتهای همزمان

توصیف وضعیتی که سناریو خاتمه میابد.(شرایط پسین postcondition)

مشاهده فرایند از دور (Ethnography)

تکنیک مشاهده برای درک نیازمندیهای سازمانی و اجتماعی است و تحلیلگر خودش را در محیطی که سیستم باید در انجا کار کند قرار میدهد و کار روزانه افراد را مشاهده و یادداشت میکند.

1-نیازها ی واقعی (نه نیازهای موجود در تعریف فرایندها)

2-نیازهایی که از همکاری سایر افراد و آگاهی از فعالیتهای آنها بدست می آید.

اتناگرافی برای کشف دو نیاز مفید است

تحلیل ساخت یافته

ساخت نمونه اولیه

مورد کاربرد (Use Case)

-هر سرویس (عملیات) ارائه شده در یک سیستم با یک Usecase بیان میشود.

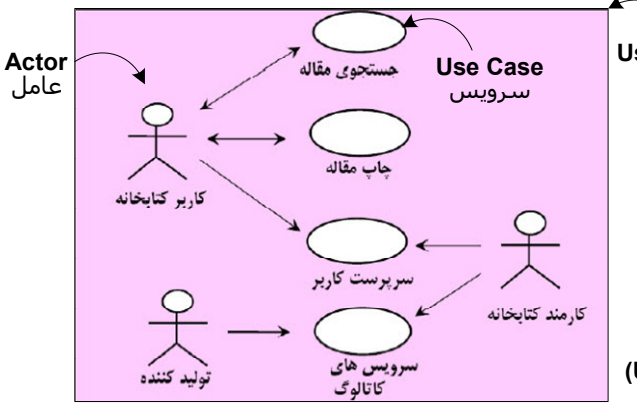
-یک سرویس (use case) از تعامل اشیاء کلاسهای مختلف با یکدیگر شکل میگیرد .بعنوان مثال سرویس ثبت نام در یک سیستم آموزشی از تعامل اشیاء کلاسهای دانشجو ،درس ،استاد ، فضای آموزشی و غیره شکل میگیرد.

نمودار کاربرد (Use Case Diagram) مجموعه سرویسهای ارائه شده در یک سیستم با نمودار (دیاگرام) usecase نشان داده میشود.

نمودار کلاس (Class Diagram) برای هر use case یک نمودار کلاس وجود دارد و مجموعه کلاسهای مرتبط را یک نمودار کلاس گویند.

نمودار توالی (Sequence Diagram) بعد از نمودار کلاس برای یک Use Case ارائه میشود و ترتیب توالی تعامل اشیاء کلاسها را با تکیه بر بعد زمان ،نشان میدهد.

نمودار همکاری (Collaboration Diagram) همان نمودار توالی است منهای بعد زمان



1 (Use Case Diagram) → 2 (Use Case) → 3 (Class Diagram) → 4 (Sequence Diagram) → 5 (Collaboration Diagram)

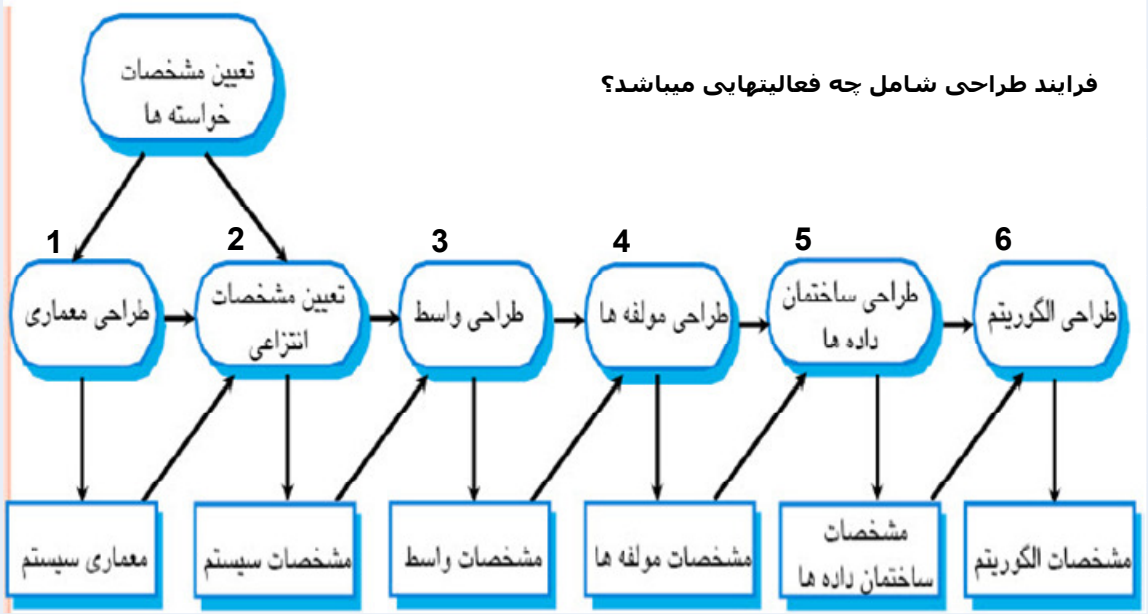
عامل (Actor) 1-انسان 2-ماشین (سیستم نرم افزاری و سخت افزاری) 3-زمان (مانند آلارم)

اصول و مفاهیم طراحی

-نیازهای مطرح شده در مرحله تحلیل ،در مرحله طراحی با دیدگاه انتزاعی مطرح میشود بعنوان مثال ذخیره سازی اطلاعات یک نیاز در مرحله تحلیل است که در مرحله طراحی ابزارهای انتزاعی مانند جدول و غیره برای ذخیره سازی آن مطرح میشود.

-سه ویژگی برای ارزیابی یک طرح خوب عبارت است از 1:مدل تحلیل باید در طراحی قابل پیگیری باشد 2:طراحی باید برای توسعه دهندگان نرم افزار (کدنویس ،پشتیبان) قابل فهم باشد 3:طراحی باید تصویرکاملی از نرم افزار ارائه دهد.

فرایند طراحی شامل چه فعالیتهایی میباشد؟



1: براساس مشخصات ،نیازها به شکل انتزاعی ارائه میشوند. طراحی کلاسها و روابط بین آنها جزعی از طراحی معماری میباشند.

2: بخشهای مختلف معماری دارای چه ویژگیها و مشخصاتی میباشند.

3: مجموعه متدهای public یک کلاس تشکیل واسط یک کلاس را میدهند و از طریق این واسط میتوان به اشیاء ایجاد شده از آن کلاس دسترسی داشت.

4: قطعات نرم افزاری مورد نیاز طراحی میشود(components) مثلا .ocx , .dll

5: چه نوع ساختمان داده ای مورد نیاز است. انتخاب ساختمان داده بر اساس ماهیت مسئله انجام میشود مثلا در محیطهایی که نیاز به جستجوی زیاد دارند آرایه خوب است و اگر حذف و اضافه زیاد باشد link list بهتر است.

6: براساس ساختمان داده انتخاب شده الگوریتم مناسب ارائه میشود (مباحث طراحی الگوریتم)

مفهوم طراحی معماری: (Architectural Design)

فرایند شناسایی زیر سیستمها و ایجاد چهارچوبی برای کنترل زیر سیستمها و ارتباط دادن آنها را طراحی معماری گویند.

طراحی معماری 1:اولین مرحله فعالیت طراحی است 2:پیوند بین مرحله تحلیل و طراحی است 3:اجزاء اصلی سیستم و ارتباطات بین آنها را مشخص میکند (مانند کلاسها)

ویژه گیهای طراحی معماری

انتزاع (Abstraction)

انتزاع به یک شخص اجازه میدهد بدون توجه به جزئیات بی اهمیت سطح پایین ،بر یک مسئله تمرکز کند.

پالایش (Refinement)

یک راهبرد طراحی بالا به پایین است که برنامه با پالایش پیاپی توسعه می یابد .در هرمرحله یک یا چند دستورالعمل به چند دستورالعمل جزعی تر تجزیه میشود (ازکل به جزء رسیدن)

پیمانه ای کردن (Modularity)

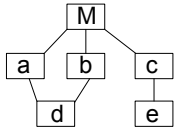
معماری نرم افزار میتواند پیمانه ای باشد .یعنی نرم افزار به چند مولفه جداگانه و متمایز تقسیم میشود که غالبا پیمانه نامیده میشود.

قابلیت خوانایی نرم افزار یکپارچه (تک پیمانه ای پایین است. با افزایش تعداد پیمانه ها ،هزینه تولید پیمانه نرم افزار کاهش میابد ولی هزینه جامعیت INTEGRITY پیمانه ها افزایش میابد.

معماری نرم افزار (SOFTWARE ARCHITECTURE) ساختار سلسله مراتبی مولفه های برنامه ،شیوه تعامل این مولفه ها با یکدیگر و ساختمان داده هایی است که مورد استفاده مولفه ها قرار میگیرد.

پیمانه ها

سلسله مراتب کنترلی



fan in: تعداد سلولهایی که یک پیمانه را مستقیما کنترل مینماید.(پیمانه a,b پیمانه ی را کنترل میکنند)

fan out: تعداد سلولهایی که توسط یک پیمانه دیگر مستقیما کنترل میشوند.(پیمانه a,b توسط پیمانه M کنترل میشوند)

super ordinate: پیمانه ای که یک پیمانه دیگر را کنترل مینماید.

sub ordinate: پیمانه ای که توسط پیمانه دیگرکنترل میشود.

طراحی پیمانه های کارآمد

استقلال عملیاتی

توسعه پیمانه هایی که حداکثر یک عمل را انجام میدهند و فقط یک واسط دارند. نگهداری و آزمایش پیمانه های مستقل آسانتر است.

Cohesion: هر پیمانه تنها یک وظیفه را انجام دهد و تعامل اندکی با بخشهای دیگر برنامه داشته باشد.

Coupling: میزان اتصال و ارتباط بین پیمانه ها را گویند (هدف کمترین میزان coupling است)

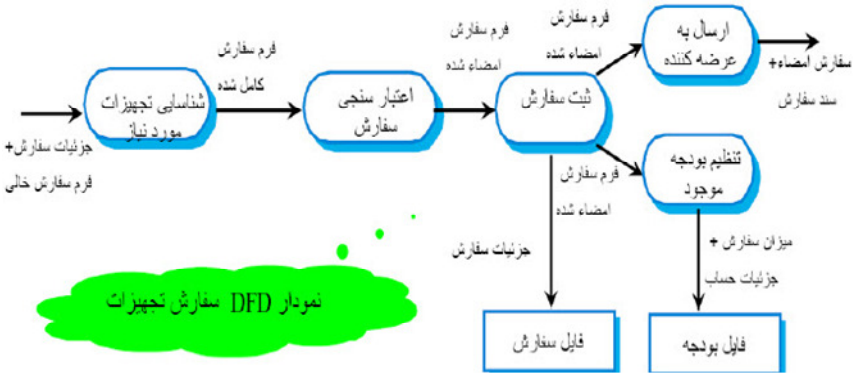
مدلهای سیستم (System Models)

مدل رفتاری (Behavioral Model) برای توصیف رفتار کلی سیستم بکار میروند.

مدل جریان داده (Data Flow Model)

پردازش داده ها در سیستم را نشان میدهد.(مانند سیستمهای تجاری)

این نمودارها نشان میدهند که داده ها چگونه از طریق دنباله ای از مراحل پردازش جریان می یابند.



مدل ماشین حالت (Machine State Model)

پاسخگویی سیستم به رویدادها را نشان میدهد.(سیستمهای بلادرنگ)

رویدادهایی از سیستم را نشان میدهد که موجب انتقال از حالتی به حالت دیگر میشوند ،در هر زمان ماشین در حالتی از سیستم قرار دارد و با دریافت یک محرک از حالتی به حالت دیگر میرود.

اگر نیاز به بیان جزئیات بیشتری از سیستم در مدل ماشین حالت بود از دیکشنری داده ها استفاده میشود.

مشکل مدل ماشین حالت رشد سریع تعداد حالتهاست که میتوان برای حل این مشکل از ابر حالت استفاده نمود و چند حالت مجزا را بسته بندی نمود.

مدل داده ای (Data Model)

بیشتر در رابطه با بخش پایگاهی یک محصول نرم افزاری است .مراحل مختلف تولید یک محصول نرم افزاری امروزه بصورت شی گرا انجام میشود ولی بخش پایگاهی آن بصورت رابطه ای پیاده سازی میگردد.

مدل داده ای برای توصیف ساختار منطقی داده های پردازش شده توسط سیستم بکار میرود.

مدل نهاد-رابطه-صفت (Entity-Relation-Attribute) ERA معمولترین روش مدل داده ای میباشد.

شی گرا:حدادی شی از کلاس دانشجو است.

ER حدادی نمونه ای از موجودیت دانشجو است.

مدل ساختاری (ساخت یافته) (Structural Model)

مبتنی بر نیازها است. در این روش از مدل ها (Driving Process) ،قوانین (Rules) ،و راهنماها (Guideline) برای تولید سیستم استفاده میشود.

معایب: 1:نیازمندیهای غیر عملکردی را مدل نمی نماید. 2:حجم مستند سازی بالاست 3:مدیریت ریسک بسیار پایین است 4:دارای جزئیات زیاد و غیر ضروری اند و فهم مسئله را مشکل میکنند.

مدل اشیاء (Object Model)

در کلاسهای مبتنی بر شی روابط بین کلاسهای اشیاء و خود اشیاء مدل میشود.

دیکشنری داده ها کلیه قراردادها در دیکشنری قرار دارند.دیکشنری داده ها توصیف کاملی از نهادها ،روابط و صفات موجود در این مدل است.

طراحی شی گرا

UML (Unified Modeling Language)

شامل ضوابط و قوانین تولید محصول نرم افزاری به شیوه شی گرا و ابزار نرم افزاری آن Rational Rose میباشد. (زبان مدل سازی یکنواخت)

توسعه شی گرا

در این روش یک مدل کوچک نرم افزاری بر مبنای مدل واقعی ساخته میشود و تعامل اجزای این مدل با یکدیگر نیازهای کاربر را پوشش میدهند . اجزاء این مدل نرم افزاری اشیاء با کلاسهای مختلف هستند ،بنابراین شناسایی کلاسهای اشیاء یکی از مهمترین فعالیتها است.

- **Object-Oriented Analysis (OOA)** با توسعه مدل شی گرا ی دامنه کاربرد سروکار دارد.
- **Object-Oriented Devlopment (OOD)** با توسعه مدل سیستم برای پیاده سازی نیازها سروکار دارد.
- **Object-Oriented Programing (OOP)** با پیاده سازی طراحی نرم افزار با زبانهای شی گرا سروکار دارد.

ویژه گیهای طراحی شی گرا

- اشیاء نمونه ای از نهادهای (کلاسهای)سیستم هستند.
- اشیاء مستقل بوده و وضعیت آنها بصورت کپسوله (Encapsulate) میباشد.
- ارتباط بین اشیاء از طریق ارسال پیام (message Passing) انجام می گیرد
  - 1-سرویس یا متد درخواست میشود.
  - 2-کپی اطلاعات مورد نیاز برای اجرای سرویس و نتایج اجرای سرویسها
- اشیاء ممکن است توزیع شده باشند.(در کامپیوترهای مختلف منشر میشوند)
- اشیاء ممکن است بصورت موازی یا ترتیبی اجرا(ایجاد) شوند.

یعنی یک شی از شی دیگر (t1) میخواهد که متد ()calculate را با پارامترهایی که به آن میدهد call کند.

Message

t1.calculate()	1
50,80	2

کلاس کارمند در UML

نام کلاس

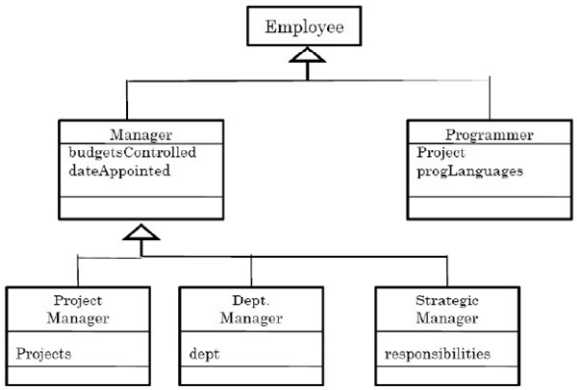
صفات

متدها

ارتباطات همگام و ناهمگام

- درارتباطات همگام ،شی فراخوان باید منتظر بماند تا شی گیرنده پیام را ترجمه کند ،سرویسها و داده های درخواستی را شناسایی کند تا به درخواست پاسخ گوید
- در ارتباطات ناهمگام ،اشیاء بصورت فرآیندهای همزمان یا (Thread) پیاده سازی شده و شی فراخوان منتظر پایان عملیات درخواست سرویس نبوده و به کار خود ادامه میدهد.

وراثت (تعمیم) در UML

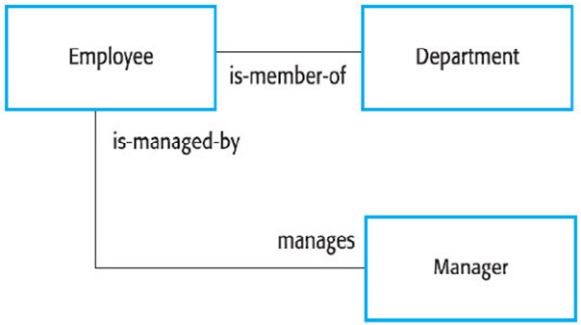


- در زبان uml به رابطه وراثت، تعمیم (Generalization) گویند.
- رابطه تعمیم بین کلاسها با ▷ نشان داده میشود.
- در این زبان وراثت روبه بالاست نه رو به پایین (جهت فلش)

ارتباط تناظر (Association Relation)

ارتباط تناظر بین دو کلاس خود یک کلاس است .(کلاس تناظر) بعنوان مثال ارتبط قرارداد بین کلاس بازیکن و کلاس تیم یک کلاس تناظر است .

قرارداد یک شی از کلاس بازیکن با یک شی از کلاس تیم در واقع خود یک شی از کلاس تناظر قرار داد است .پس خط تناظر بین کلاسها خود یک کلاس است (مانند جدول رابطه بین دو موجودیت)



فرآیند طراحی شی گرا

تعریف حالتهای کاربرد سیستم

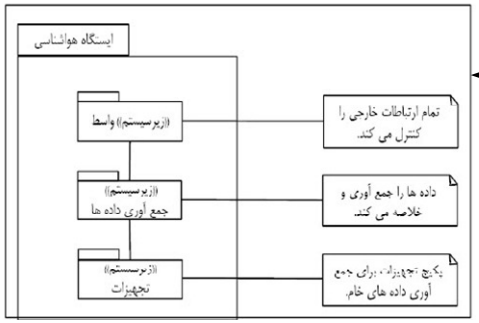
مدلهای کاربرد

نسخه ای از مدل use case (نمودار آن)در مرحله طراحی برای مدل کردن ارتباط محصول نرم افزاری با دنیای بیرون استفاده میشود که به آن Design Use Case Model گفته میشود.
اولین مرحله در فرایند طراحی نرم افزار درک روابط بین نرم افزار در حال توسعه و دنیای بیرونی آن است این درک به کمک مدل کاربرد سیستم بدست می اید.
در روش RUP مدل سازی تعامل یک سیستم با زیر سیستم هایش با use case model انجام میشود.

توصیف Use Case

هر مورد کاربرد را میتوان با زبان طبیعی توصیف کرد با توصیف موارد کاربرد (use case description) اشیاء سیستم بسادگی قابل شناسایی است.

طراحی معماری سیستم



طراحی معماری سیستم

پس از آنکه ارتباط بین سیستم و محیط اطرافش شناسایی و درک شد از این اطلاعات در طراحی معماری سیستم استفاده میشود.

شناسایی کلاسهای اشیاء سیستم (object Identification)

شناسایی اشیاء سخت ترین بخش فرایند طراحی شی گرا است که تنها بر اساس تجربه ،مهارت و دانش طراحان بدست می آید.

شناسایی کلاسها

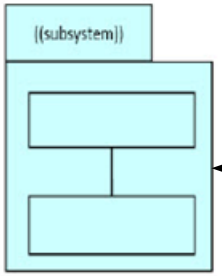
1:استفاده از راهکارهای گرامری در توصیف زبان طبیعی (اسامی بعنوان کلاس و صفات ،افعال بعنوان متد
2:استفاده از نهادهای ملموس (tangible) در دامنه کاربرد.
3:استفاده از روشهای رفتاری و اختصاص هر رفتار به هر بخش از سیستم
4:استفاده از تحلیل مبتنی بر سناریو، در توصیف هر سناریو ،صفات و متدهای شی شناسایی میشود.

توسعه مدلهای طراحی با استفاده از زبان UML

انواع مدل طراحی (Design Model) مدلهای طراحی اشیاء کلاسهای شی و ارتباط بین آنها را نشان میدهد.

مدلهای ایستا: (Static Model) برای بیان ساختار سیستم بکار میرود.
نمودارهای کلاس ،نمودارهای شی ،نمودار اجزاء و نمودار های استقرار برای مدل کردن ایستای سیستم بکار میرود.

مدلهای پویا(Dynamic Model) برای بیان رفتار و نحوه عملکرد سیستم بکار میرود.
نمودارهای Use Case ،نمودارهای فعالیت ،نمودار حالت ،نمودار توالی و همکاری جهت ارائه مدلهای پویا ارائه شده اند.



زیر سیستم

مدلهای زیر سیستم(Subsystem Model)

هر زیر سیستم بصورت یک (package) نمایش داده میشود. هر پکیج دارای ساختاری ایستا میباشد.

مدلهای توالی و همکاری (Sequence & Collaboration Model)

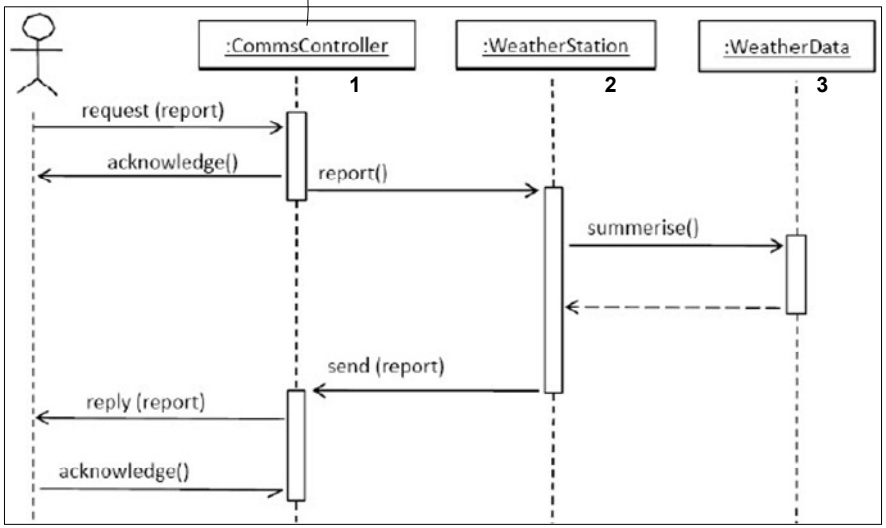
این مدلها ترتیب تعامل بین اشیاء کلاسها را نشان میدهند.

مدل ماشین حالت (Statechart Machine Model)

حالتهای یک شی از یک کلاس (چگونگی تغییر حالت اشیاء در پاسخ به رویدادها) را بیان میکند.

نام شی قبل از کلاس ذکر نشده است

نمودار توالی



قبل از نمودار توالی برای Use Case ،باید نمودار کلاس برای هر سه کلاس ذکر شده تهیه و رسم گردد.

Bay

شناسایی واسطهای شی (مجموعه متدهای عمومی)

تعیین مشخصات واسط شی:(Object Interface Specification)

متدهای عمومی که معمولا در همه کلاسها وجود دارند معرفی میشوند تا توسعه دهندگان سیستم در طراحی کلاسها آن را مد نظر قرار دهند و این متدها در کلاسهای مختلف به موازات هم ایجاد میشوند.

فعالیتی از فرایند طراحی است که واسط قطعات مختلف طراحی مشخص میگردد بطوریکه اشیاء و قطعات بتوانند موازی طراحی شوند.

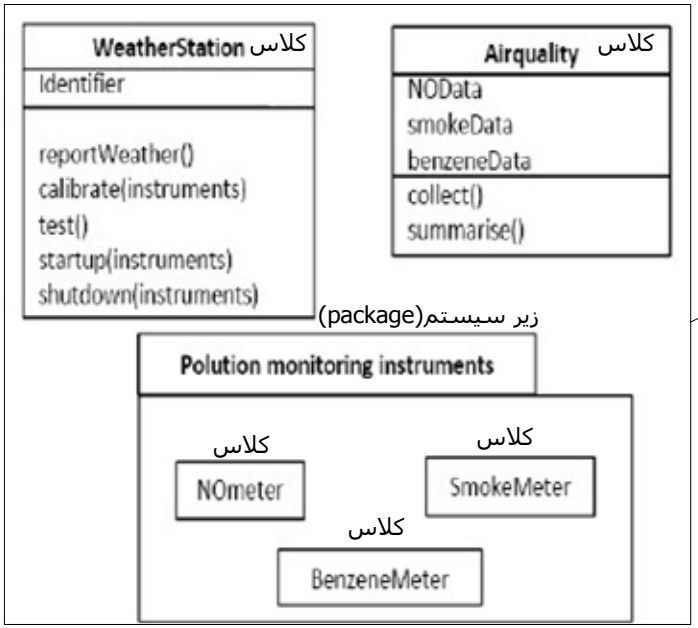
نمایش واسط در UML همانند نمایش کلاس است که فاقد بخش صفات است.

بین اشیاء و واسطها لزوما ارتباط یک به یک برقرار نیست زیرا شی ممکن است چندین واسط داشته باشد.

تکامل طراحی

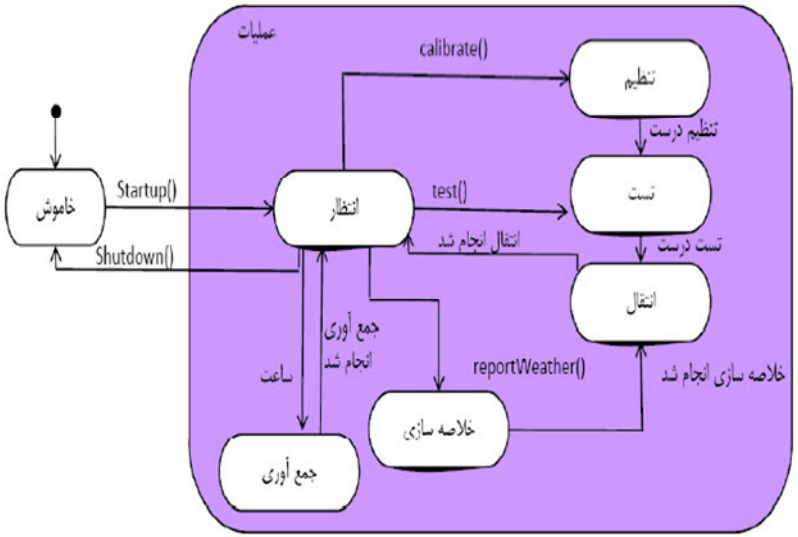
مونتاژ قطعات طراحی شده

از آنجا که ارتباطات بین اشیاء Loosely Coupled بوده و نمایش حالت شی بر طراحی تاثیر ندارد ،تغغیر در اشیاء موجود در سیستم یا اضافه نمودن شی جدید هزینه کمی دارد.



تکامل طراحی

نمودار حالت شی از کلاس ایستگاه هواشناسی در سیستم هواشناسی



توصیف واسط کلاس ایستگاه هواشناسی با جاوا

```
interface WeatherStation {
        public void WeatherStation () ;
        public void startup () ;
        public void startup (Instrument i) ;
        public void shutdown () ;
        public void shutdown (Instrument i) ;
        public void reportWeather ( ) ;
        public void test () ;
        public void test ( Instrument i ) ;
        public void calibrate ( Instrument i) ;
        public int getID () ;

} //WeatherStation
```

با مشخص شدن این واسط ، طراح کلاس، متدهای عمومی مشخص شده در آن کلاس را لحاظ میکند.

(Unified Modeling Language) UML

1

انتزاع:مدل کوچک نرم افزاری

- مدل کردن شامل ساخت یک انتزاع از یک واقعیت است.
- انتزاع ها برای ساده سازی بکار میروند به این علت که
  - جزئیات غیر مرتبط را حذف میکنند.
  - فقط جزئیات مرتبط را بیان میکنند
- مرتبط بودن یا غیر مرتبط بودن وابسته به هدف (منظور)از مدل سازی است.
- بعنوان مثال نقشه یک شهر یک مدل کوچک (انتزاع) از شهر است
- طراحی و تحلیل =مدل سازی

2

چرا مدل نرم افزاری؟

- محصولات نرم افزاری بزرگ بسیار پیچیده هستند
  - ویندوز XP بیش از 40 میلون خط کد نویس دارد.
  - برای انجام پروژه های بزرگ نرم افزاری ،تیمهای توسعه دهنده باید مجهز به دانش توسع نرم افزار بصورت مهندسی باشند.
- کد نویسی بدون مدل سازی براحتی توسط توسعه دهندگان قابل فهم نیست.
- نیاز به روشی برای ارائه (نمایش) سیستم های پیچیده داریم.
- مدل سازی میتواند روشی برای نمایش ساده سیستمهای پیچیده باشد.

3

چه کاری باید اول انجام شود؟

- کد نویسی یا مدل سازی؟ بستگی دارد به؟

مهنسی مستقیم**(Forward Engineering)**

تولید کد از مدل برای پروژه های جدید**(Greenfield project)**

مهنسی معکوس **(Reverse Engineering)**

ساخت مدل از کد و مهندسی مجدد پروژه **(Reengineering project)**

مهندسی رفت و برگشت **(Roundtrip Engineering)**

-در اینحالت مهندسی مستقیم و معکوس بصورت ترکیبی انجام میشود و فی مابین دو مهندسی ذکر شده قبلی میباشد.

برای حالاتی که نیازمندیهای تکنولوژی و زمانبندی بطور مداوم در حال تغییر است مناسب است.

4

UML سندی است شامل قوانین و ضوابط مستند سازی تولید محصول نرم افزاری بصورت شی گرا

- از UML برای بصری سازی (visualize) ،ویژه سازی (Specify) ،ساخت (construct) استفاده میشود و مدلی از سیستمهای پیچیده را ارائه میکند.
- UML سندی برای یکپارچه سازی نظرات آقایان رامبو ،جکسون و بوش در حوزه مهندسی نرم افزار به شیوه شی گرا میباشد.
- با استفاده از 20% از ابزارهای UML میتوان 80% اغلب سیستمها را مدل سازی کرد.

5

انواع دیاگرامها

ساختاری(Structural) برای بیان ساختار ایستا سیستم استفاده میشود.

(Class Diagram) برای هر (Use Case یک نمودار کلاس مستقل ارائه میشود.

- ساختار ایستا سیستم را با استفاده از اشیاء،صفات و ارتباطات توصیف میکند.
- شامل مجموعه ای از کلاسها و ارتباطات بین آنها میباشد.
- یک واسط مجموعه ای از متد ها است که سرویس خاصی در یک کلاس را مشخص میکند.

(Component Diagram) اجزاء قطعات نرم افزاری

- به مجموعه ای قطعات نرم افزاری و روابطشان گویند.
- به درک فیزیکی از یک گروه منطقی از عناصر میگویند (کلاسها ،اینتر فیسها)

(Deploy Diagram) دیاگرام استقرار

- به مجموعه ای از گره ها (nodes) و روابطشان گویند (زمانهای اجراء و بیان منابع محاسباتی)
- برای مشخص کردن جایگاه فیزیکی قطعات نرم افزاری و سخت افزاری در سیستم استفاده میشود)

رفتاری (Behavioral) نحوه عملکرد و رفتار سیستم را بیان میکند.

(Use Case Diagram) دیاگرام مورد کاربرد

1:رفتار عملیاتی سیستم از دیدگاه کاربر را توصیف میکند. 2:شامل اهداف کاربر میباشد 3:قابل درک و مشاهده از طریق موجودیتهای خارج از سیستم(عاملها actors)

(Interaction Diagram) دیاگرام محاوره ای

1:رفتار پویا بین عاملها و سیستم و بین اشیاء سیستم را توصیف میکند. 2:روی پیامهای مبتنی بر ترتیب زمانی تمرکز دارد(توالی) 3:دیاگرامهای همکاری همان دیاگرام توالی است منهای بعد زمان

(Statechart Diagram) دیاگرام حالت

تغییر حالت یک شی از یک کلاس را در طول حیات خود نشان میدهد.

(Activity Diagram) دیاگرام فعالیت

- جریان کاری سیستم را همانند یک فلوچارت مدل میکند.
- روی کنترل جریان از یک فعالیت به فعالیت دیگر تمرکز دارد.

1

فرایند توسعه (ساخت محصول) (Development Process)

شناسایی مدلها (بصورت سطح بالا) **(Use Case Diagram)**

شناسایی اشیاء مهم و ارتباطات آنها **(Class & Object Diagram)**

ایجاد سناریو برای موارد کاربرد **(Sequence & Collaboration Diagram)**

توسعه (تعمیم) سناریوها برای توصیف رفتار **(State & Activity Diagram)**

پالایش (اضافه)کردن جزئیات پیاده سازی **(Component & Deployment Diagram)**

2

(Use Case View) بخشی از مدل مد نظر است

یافتن Use Caseها:چه سرویسهایی در سیستم ارائه میشود.(مورد استفاده)

Use Case

یک سرویس قابل ارائه در سیستم مورد نظر را نشان میدهد.

هر Use Case از تعامل اشیاء چندین کلاس با یکدیگر شکل میگرد.بعنوان مثال سرویس ثبت نام از تعامل اشیاء کلاسهای دانشجو ،درس ،استاد ،فضای آموزشی و...شکل میگیرد.

این کلاسها در قالب دیاگرام کلاس برای آن Use Case ارائه میشود.

یافتن Actor ها(عاملها ،بازیگرها)

Actor

هرکس یا هرچیزی که با سیستم موجود برهم کنش دارد عامل گفته میشود.
بعبارت دیگر به هر استفاده کننده از Use Case گویند.

عاملها

انسان (کاربران سیستم)

سیتمهای سخت افزاری یا نرم افزاری دیگری که با سیستم موجود در ارتباطند.

زمان (مثلا سرویس backup در یک زمان خاص)

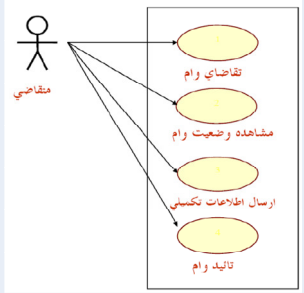
رسم Use Case Diagram :نمودار موارد استفاده

برای بیان تمام سرویسهای استفاده شده در سیستم بکار میرود.

هر Use Case یک کلاس دیاگرام دارد.

ترکیب کلی Use Case وActor

نمودار موارد کاربرد سیستم اعطای وام



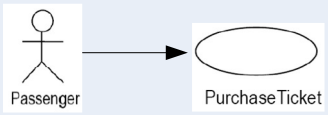
توصیف Use Case ها:شرح هر کدام از سرویسهایی که استفاده شده اند. (ارائه سناریو)

3

انواع ارتباطات در نمودار مورد استفاده

ارتباط Communiocation

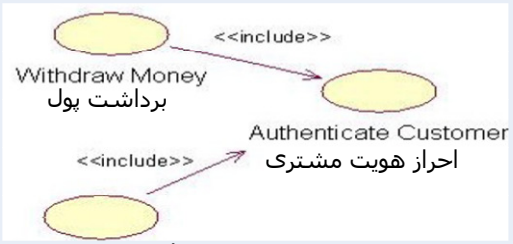
به ارتباط بین عامل (actor)و مورد استفاده (Use Case)گویند.

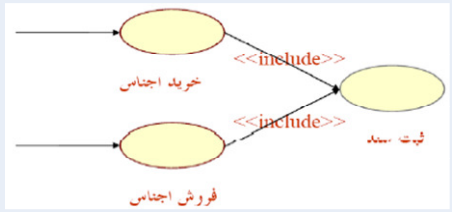


ارتباط Include

به ارتباط بین مورد استفاده هاگویند.

ارتباط Include به یک Use Case اجازه استفاده از عملیات مهیا شده توسط یک Use Case دیگر را میدهد.



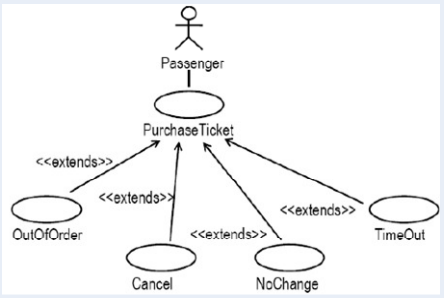


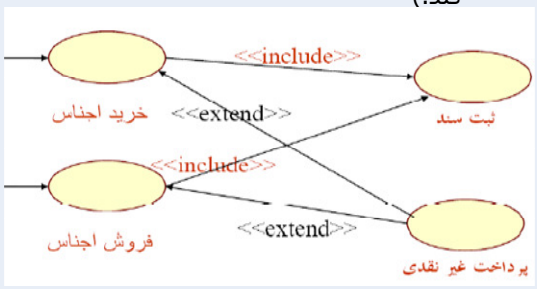
ارتباط Extend

به ارتباط بین عامل هاگویند.

یک ارتباط Extend به یک Use Case اجازه میدهد که بطور دلخواه عملیات مهیا شده توسط دیگر Use Caseها را بسط دهد.

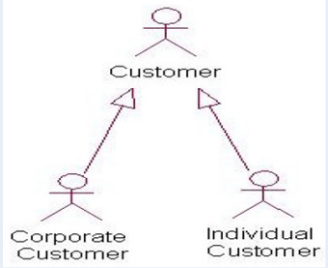
یک Use Case در حالات خاص میتواند بطور ویژه ارائه شود (به یک سرویس ویژه توسعه پیدا کند.)





ارتباط Generalization

ارث بری میان عاملها برای نشان دادن همانندی چندین عامل بکار میرود.



سناریو Use Case

تعریف مختصر

یک تعریف مختصر مرتبط با آنچه که Use Case انجام میدهد.

پیش شرایط

لیستی از شرایطی هستند که قبل از شروع کار Use Case باید بررسی شوند:بطور مثال یک Use Case از قبل اجراء شود یا آیا کاربر حق دستیابی مستقیم برای اجراء Use Case جاری را دارد یا نه؟

جریان رخدادهای اصلی

آنچه را که در اجرای عملیات در Use Case پیش خواهد آمد را مرحله به مرحله توصیف میکند .جریان رخدادها برآنچه سیستم انجام خواهد داد متمرکز میشود ولی به چگونگی انجام آن کاری ندارد.

جریان رخدادهای فرعی

در صورتی اجراء میشوند که در اجرای جریان اصلی مشکلی بوجود آید.مثلا در Use Case برداشت پول میزان درخواستی از میزان موجودی حساب بیشتر باشد.

شرایط پسین

شرایطی هستند که باید بعد از پایان اجرای Use Case دارای مقدار true باشند.مثلا ممکن است بعد از کامل شدن یک Use Case یک flag تنظیم شود.

مثال سناریوی Use Case

جریان رخدادهای اصلی:

۱ _ دانشجو فرم تکمیل شده حذف و اخذ واحدها را به کارمند آموزش تحویل می دهد .
۲ _ کارمند آموزش پیش نیاز درسها را در پرونده کارنامه های دانشجو کنترل می کند .
۳ _ کارمند آموزش بر مبنای معدل سقف واحدها را چک می کند .
٤ _ کارمند آموزش ظرفیت کلاسها را کنترل می کند .
٥ _ کارمند آموزش تداخل دروس را کنترل می کند .
٦ _ کارمند آموزش برگه حذف و اخذ را مهر تائید می زند .
۷ _ کارمند آموزش برگه تائید شده را به دانشجو تحویل می دهد .

جریان رخدادهای فرعی:

۲ _ در صورتی که پیش نیاز یا هم نیاز رعایت نشده باشد آن درس از لیست دروس حذف می شود .

۳ _ در صورتیکه معدل زیر ۱۲ باشد حداکثر ۱٤ واحد به دانشجو می دهد .
۳ _ در صورتیکه معدل بالای ۱۷ باشد حداکثر ۲٤ واحد به دانشجو می دهد .
۳ _ در صورت تائید مدیر آموزش هر تعداد واحدی به دانشجو می دهد .

برای به واقعیت پیوستن یک Use Case تعدادی شی از کلاسهای مختلف با هم تعامل و همکاری میکنند.

کلاسهای اشیاء یک Use Case در نمودار کلاس آن نمایش داده میشوند.برای هر Use Case یک نمودار کلاس ارائه میشود.

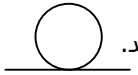
تعامل و همکاری اشیاء در یک Use Case از طریق نمودارهای محاوره ای(Intraction) نشان داده میشود که به دو نوع نمودارهای همکاری و توالی تقسیم میشود.

نمودار کلاس Class Diagram

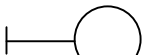
انواع کلاسها در UML (Stereotype)

کلاسهای کنترلی وظیفه کنترل عملیات را بر عهده دارند و معمولا کلاس بی صفت هستند و متد دارند و شی با این کلاس ایجاد نمیشود.

کلاسهای Entity مبنایی برای ایجاد Table ها و بانک اطلاعاتی سیستم هستند و به جدولی در پایگاه داده ها تبدیل میشوند.



کلاسهای Boundry در مز بین Actor و سیستم قرار دارند. مثلا فرم ثبت نام شی از کلاس باندری فرم است.(کلیه فرمها) یا گزارش عملکرد مالی شی از کلاس باندری گزارش است. و کلا این کلاسها شاخص فرمهای ورودی – خروجی و رابطها میباشند.



انواع ارتباطات در نمودار کلاس

- تعمیم
- تناظر
- تجمیع
- ترکیب
- وابستگی

موارد استفاده Class Diagram

ساختار سیستم را ارائه میکند.

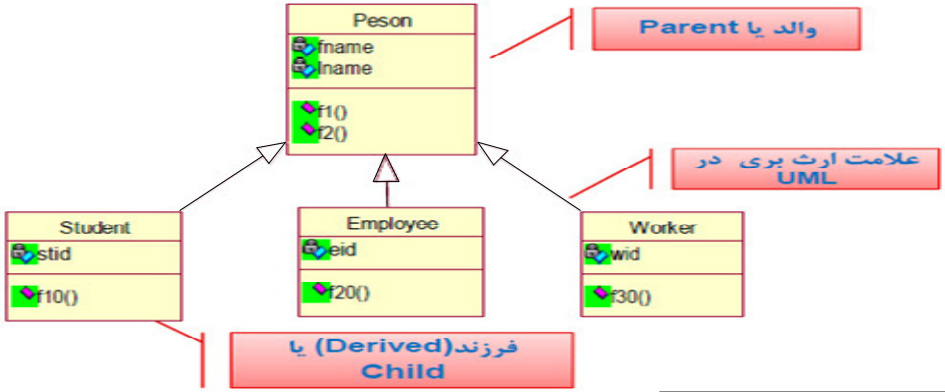
1:در ضمن تحلیل نیازمندیها برای مدل کردن

2:برای مدل کردن زیر سیستم ها و واسط ها

3:طراحی شی برای مدل کردن کلاسها

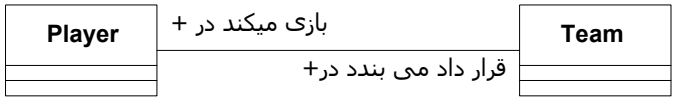
ارتباط تعمیم(Generalization)

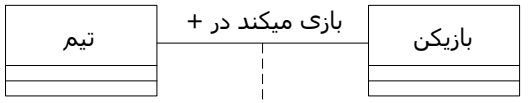
همان ارتباط وراثت است و شامل 1:کاهش حجم کد نویسی 2:سازماندهی یک گروه کلاس مرتبط به هم میشود.



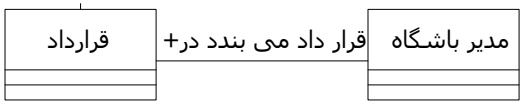
ارتباط تناظر (Association)

تعامل یک یا چند شی از یک کلاس با یک یا چند شی از کلاس دیگر را نشان میدهد.یک تناظر یک ارتباط معنایی بین کلاس ها است.



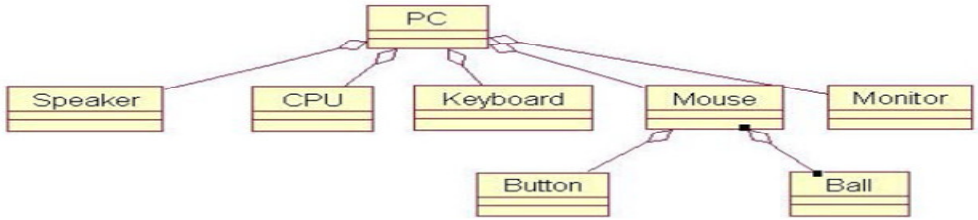


قرارداد یک شی از کلاس تیم با یک شی از کلاس بازیکن بعنوان یک شی جدید از کلاس تناظر قرارداد مد نظر قرار میگیرد.



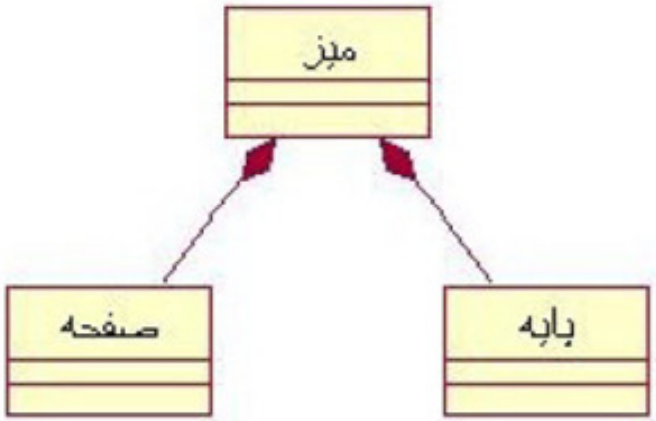
ارتباط تجمیع (Aggregation)

چند شی از چند کلاس مختلف با هم جمع شده و یک شی از کلاس جدید میسازند.در تجمیع ،هر کدام از اشیاء خارج از مجموعه تجمیع نیز قابل استفاده است.



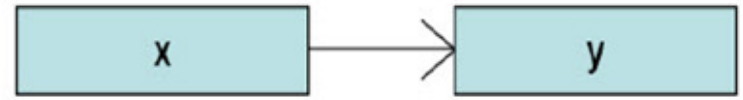
ارتباط ترکیب (Composition)

همان مفهوم تجمیع را بصورت قوی تر دارد.در ترکیب هر کدام از اشیاء خارج از مجموعه ترکیب قابل استفاده نیست.



ارتباط وابستگی (Dependency)

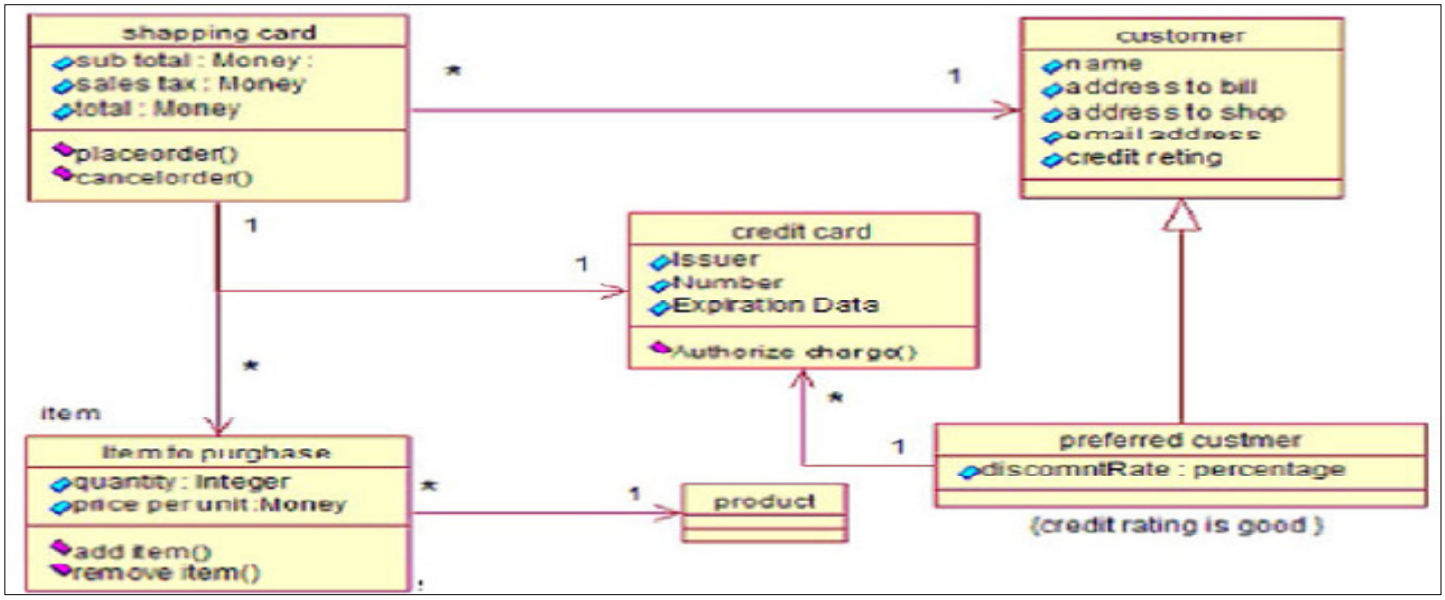
وقتی که میگوییم کلاس x به کلاس y وابسته است بدین معناست که اگر در هر یک از مشخصه های y تغییری صورت پذیرد میتواند باع ایجاد تغییری در x گردد.




```
Class y{…….}
Class x{
Void f1(y a,int b,float m)}
```

***=N**

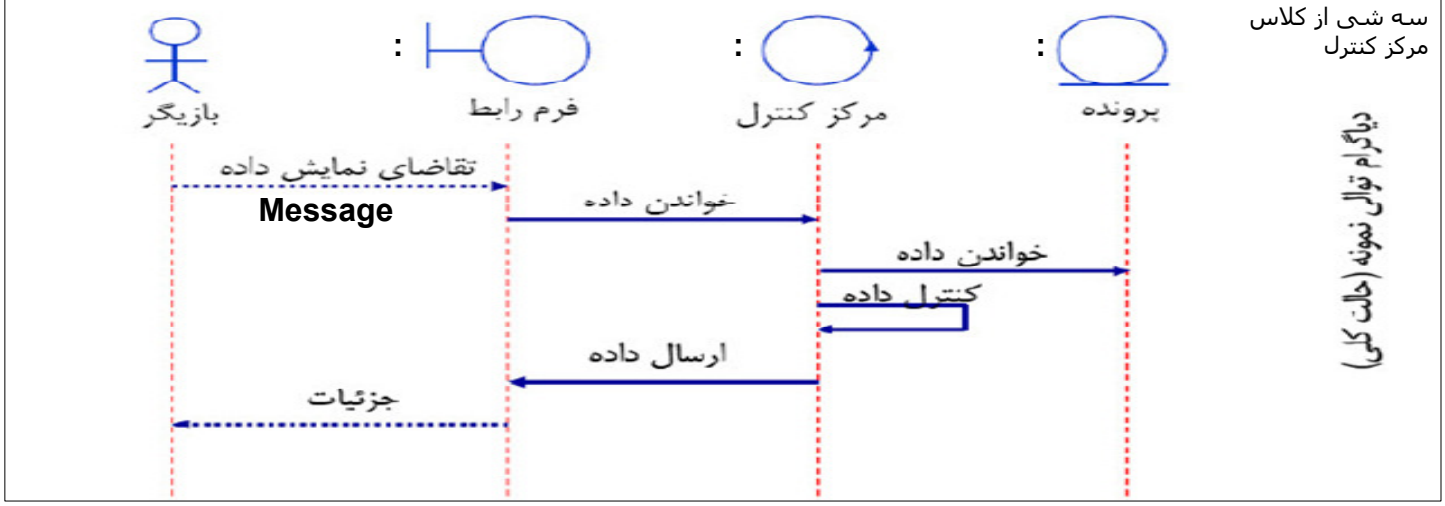
Class Diagram Ex.



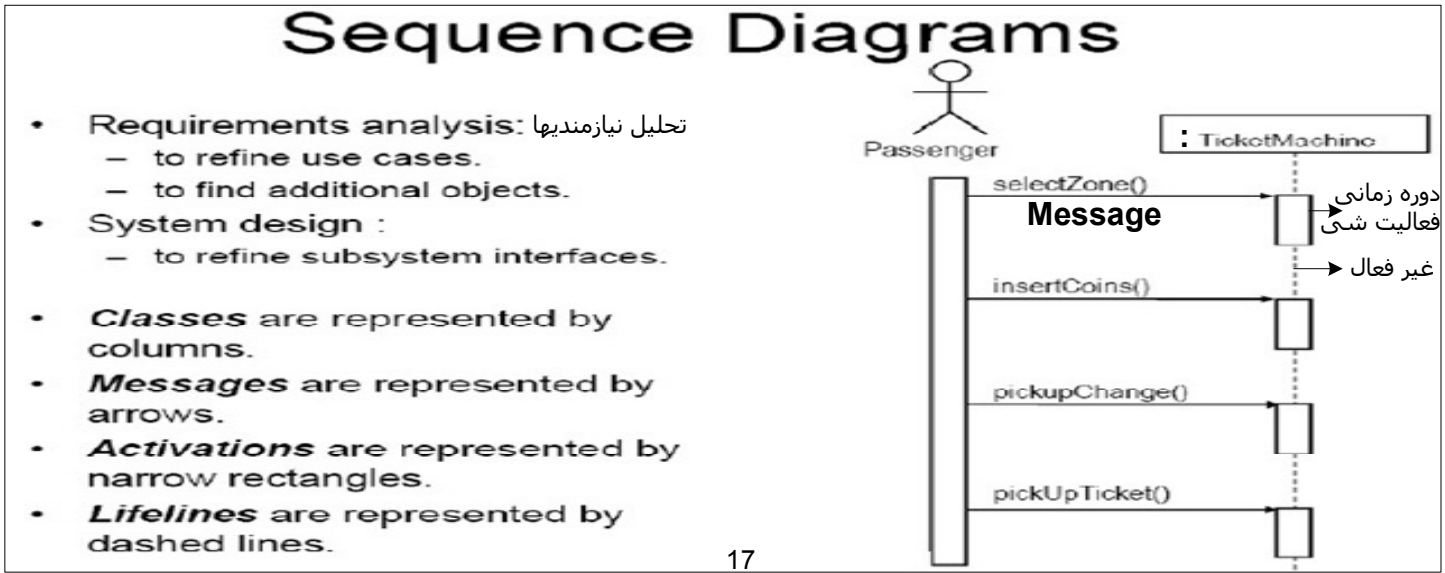
نام کلاس :نام شی

Sequence Diagram Ex.

بر اساس نمودار کلاس، برای هر مورد کاربرد یک نمودار توالی ارائه می شود. در این نمودار چگونگی تعامل اشیاء کلاس ها، جهت شکل گیری مورد کاربرد(سرویس مورد نظر) نشان داده می شود. نمودار توالی شامل : object (شی)، Message (پیام) و زمان است.



Sequence Diagram Ex.



اشیاء کلاسها بعنوان سر ستون نمایش داده میشود.

با هر پیام در خواست خاصی مطرح میشود(مثلا درخواست اجرای متد ()selectzone برروی شی از کلاس Ticketmachine:

Summary of Sequence Diagram

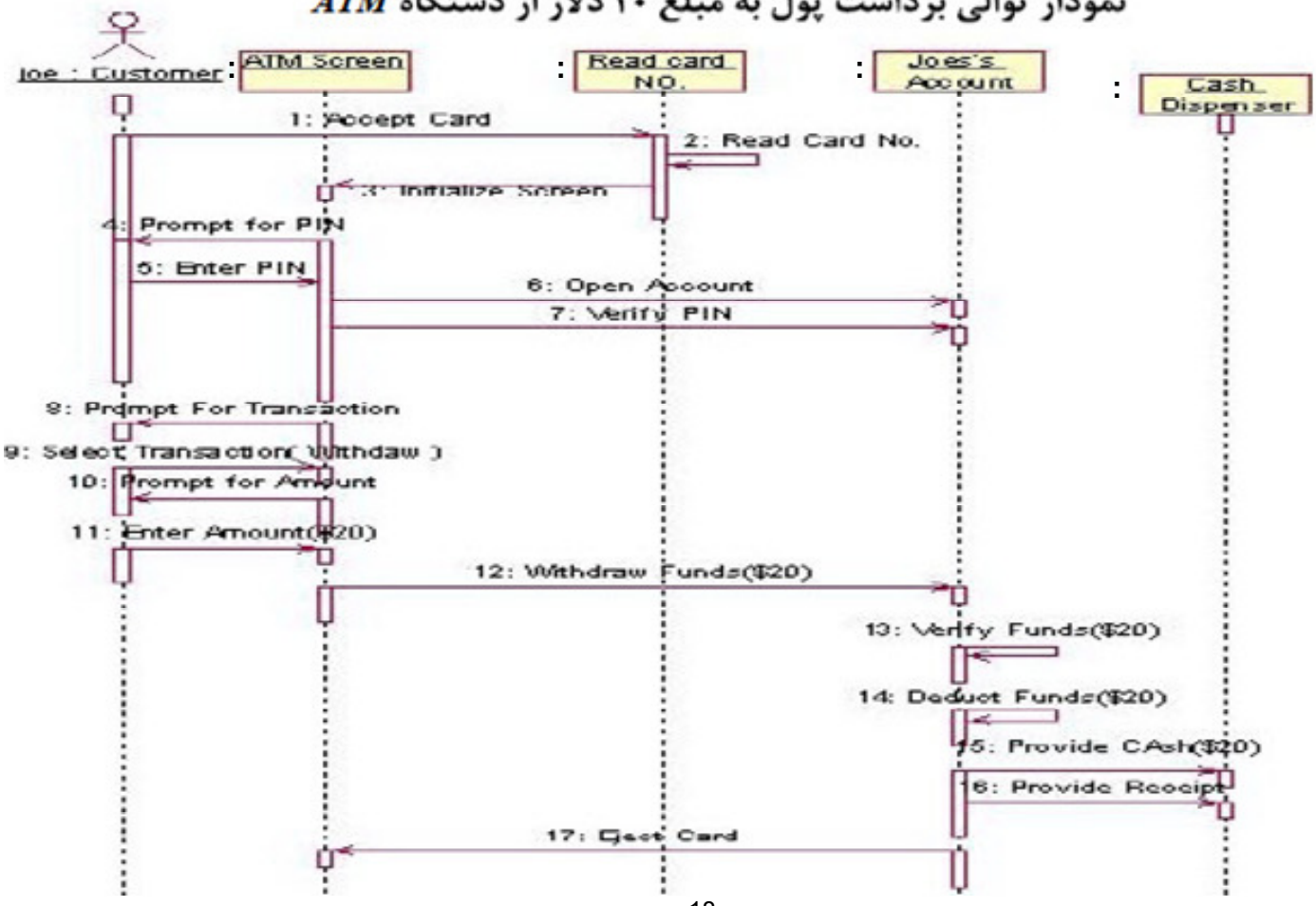
- UML sequence diagram represent behavior in terms of interactions.
- Useful to find missing objects.
- Time consuming to build but worth the investment.
- Complement the class diagrams (which represent structure).

UML دارای بعد زمان است (ترتیب و توالی پیام ها مشخص است) . مکملی برای نمودار کلاس است.

نمودار توالی(*Sequencce Diagram*)– مثال

شی از کلاس Customer

نمودار توالی برداشت پول به مبلغ ۲۰ دلار از دستگاه *ATM*



برداشت پول=Use Case شامل 4 کلاس است.

نمودار حالت (State Chart Diagram)

حالتهای مختلفی که یک شی در آن قرار میگیرد را مدل میکند.
تصویری از چرخه حیات شی (Object Life Cycle) را به نمایش میگذارد.
نسبت به سایر نمودارها مانند توالی و همکاری کمتر مورد استفاده قرار میگیرد.

موارد استفاده

اشیایی که دارای تعداد زیادی حالت هستند.
اشیایی که برای update کردن صفات خاصه خود شروط متنوعی دارند.
اشیایی که معمولا به صورت سخت افزاری هستند.
اشیایی که عملکرد بعدی آنها به عملکرد قبلی شان بستگی دارد.

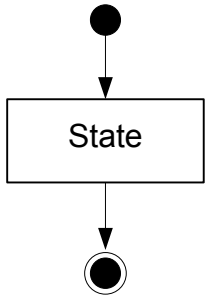
برای مدل کردن حالات یک شی در یک دوره زمانی استفاده میشود.

Start State: نقطه شروع چرخه حیات شی است.

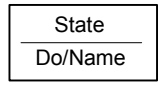
State: بیان کننده حالت یک شی است.

State Transition: مسیری برای عبور شی از یک حالت به حالت دیگر میباشد.

End State: بیان کننده نقطه پایانی چرخه حیات یک شی میباشد.یعنی طول عمرش در این نقطه به پایان میرسد.



میتوان جزئیات مختلفی را به نماد حالت اضافه کرد .برای این منظور نمدار حالت به دو قسمت تقسیم میشود.



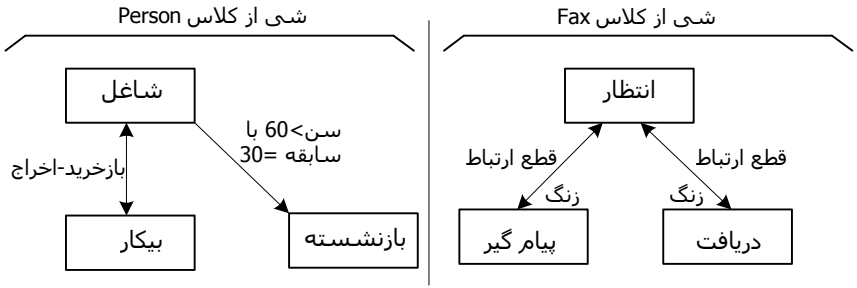
Name: نام حالت نوشته میشود.

State: متغییر های حالات برای نگهداری زمان یا شمارشگرها میباشد که معمولا سه متغییر مورد استفاده قرار میگیرد.

Entry: به مجموعه فعالیتهایی که در زمان ورود یک شی به یک حالت باید انجام گیرد گفته میشود.

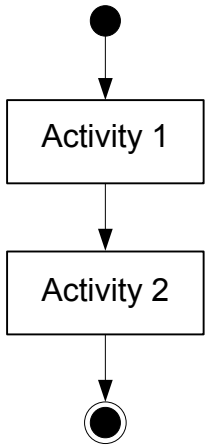
Exit: به مجموعه فعالیتهایی که در زمان خروج شی از یک حالت باید انجام شود گفته میشود.

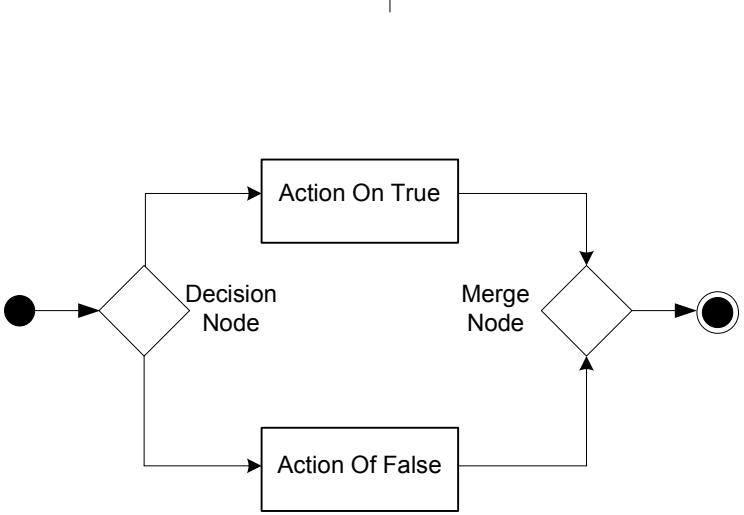
Do: حالت برای انجامم یک سری فعالیت ایجاد شده است.یعنی یک شی به یک حالت میرود تا یکسری عملیات را انجام دهد . این فعالیتها با نام Do در حالت قرار میگیرند.



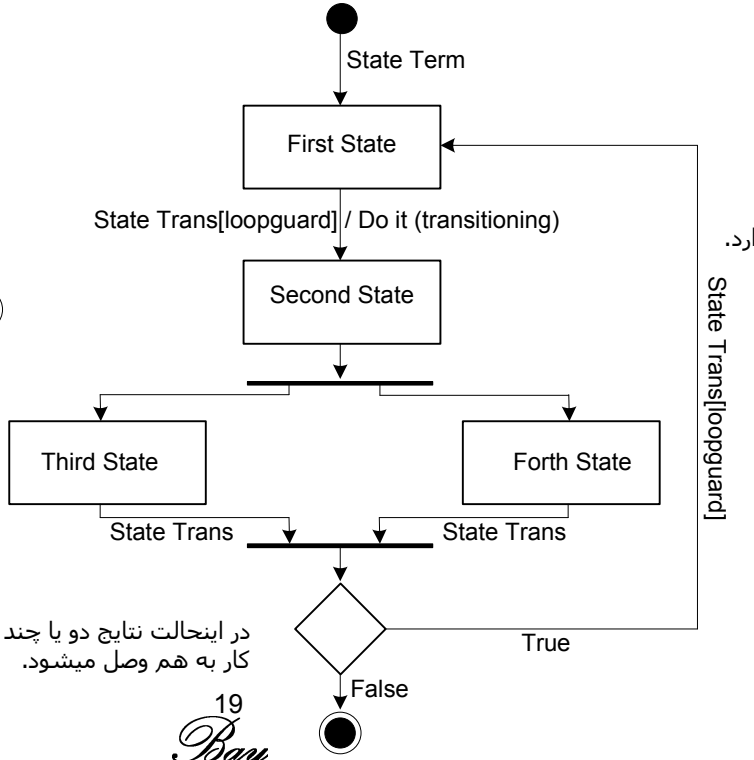
نمودار فعالیت (Activity Diagram)

جریان رفتاری داخل Use Case را به نمایش میگذارد.
مراحل یک عمل با پردازش را نشان میدهد.
نشان دهنده جنبه پویای سیستم است.
شکل ظاهری شبیه فلوچارت در زبان برنامه نویسی دارد.





در اینحالت مسیر ها به هم وصل میشوند و نه نتایج



در اینحالت نتایج دو یا چند کار به هم وصل میشود.