**خصوصیات یک مدل:**

1. دقیق Accurate
2. قابل فهم و ساده Understandable
3. سازگاری Consistent
4. انعطاف پذیری و تغییر Modifiable

**سه عامل مهم Fail شدن سیستم:**

1. سرریز هزینه ها
2. تاخیر در تحویل
3. مشخص نشدن نیازهای مشتری

**روشهای Object Oriented در رابطه با زبان مدل:**

1. Code Yourdar [OOA(Object Oriented Analyze) /OOD(Object Oriented Design)]
2. Booch (بیشتر در فرآیندهای تکرار شونده iterative استفاده میشود)
3. Object Modeling Technique (OMT در شرکتهای بزرگ استفاده میشود)
4. OOSE/Objectory
5. Fusion (نسل دوم است)

**خواص یک زبان مدل:**

1. Viewها (مجموعه ای از دیاگرامها هستند. هر سیستم ممکن است بیش از یک View داشته باشد)
2. Diagram (از عناصر گرافیکی تشکیل می شود و هدفش توصیف محتوای View است مثل Activity Diagram)
3. Model Element (عنصر گرافیکی که هدف اش نمایش مفاهیم مشترک Object Oriented است مثل Class ، پیامها ، رابطه ها)
4. General Mechanisms (مکانیزم های کلی است مثل توضیح اضافه به صورت دستی وارد میشود. Semantic است)
5. Model Driven Architecture:MDA (تلاش گروه OMG در استفاده از UML برای یک مدل قابل اجرا. ابزار مانند Rational Rose
6. **View ها و Diagramها**
7. **Usecase View:** از Usecase و Actor تشکیل شده. از دید خود سیستم ، Actor یک عنصر خارجی است که میخواهد از سیستم استفاده کند. این View یک View برای نمایش Functionality سیستم نسبت به Actor های خارجی است. کارهای ریز و Internal نباید در این View بیاید. Actor میتواند یک یوزر یا سیستم دیگر باشد.
8. **Logical View:** کلاس شامل متغیرها و متدها است. تنها متدی که ReturnValue ندارد Constructor است. اگر Void هم باشد ReturnValue دارد. کلاس متد دارد که با Variable Members کار میکند. کلاس ها ، Function ندارند. چون نمیتوانند به بیرون وابسته باشند. کار متد با اشیاء درون کلاس است. صدا کردن یک متد در یک شیء همن Message فرستادن برای اشیاء است. Behavior یک رفتار با شیء است. در رادیو عوض کردن ایستگاه (Tuning) یک Behavior است. کم و زیاد کردن صدا یک متد است و خاموش و روشن کردن Variable Members هستند. State وضعیت شیء را مشخص میکند و تغییر دادنState ، یک متد است. اندازه State یک Variable Member است. در Logical View به لایه های کلی و مشتری کار نداریم و به لایه های Internal می پردازیم (در مقابل Usecase است) دیاگرامها:
   1. **Interactive Diagram:** انتظار داریم Interactive بین اشیاء در زمان اجرای نرم افزار (نه در دنیای بیرون) مشخص شود. شامل سه نوع دیاگرام است:
      * **Sequence Diagram:** نشان دهنده Dynamic Collaboration بین تعدادی از اشیاء است. در این دیاگرام پشت سر هم کارهایی انجام میشود و جوابهایی گرفته میشود (مثلا برای برداشت پول از ATM ابتدا کارت وارد میکنیم ، سپس کلمه عبور میزنیم و ...) در این دیاگرام همه استثنائات را نمیتوان آورد. در این دیاگرام اشیاء را می آوریم و کاربران را نمی آوریم.
      * **Communication Diagram:** یک دیاگرام مستقل نیست. در دیاگرام Sequence به عنوان یک باکس می آید و شرح و ترتیب فراخوانی های بین اشیاء را نشان میدهد.
      * **Interactive Overview Diagram:** به دنبال ارتباطات High Level است. خیلی به ریز و درون ساختار کاری ندارد.
   2. **State Machine Diagram**: یک متمم یا توصیف اضافه برای کلاسها محسوب میشود. یعنی کلیه State های اشیاء موجود در سیستم باید دیده شود. یعنی ایتدا کلاس دیاگرام و سپس Object Diagram را میکشیم و Stateها را در این قسمت می آوریم. در این دیاگرام دایره سیاه نشان دهنده نقطه شروع است. فلش ها نشان دهنده تغییر حالت هستند و معمولا روی فلش ها هم گفته میشود بر اساس چه Event از این State به State بعد رفته ایم. خود State را در مستطیل میکشیم. در UML دو نوع State Machine داریم.
      * **Behavioral State Machine**
      * **Protocol State Machine**
   3. **Object Diagram:** بیشتر بحث Message داریم. نسبت به کلاس دیاگرام بیشتر سیستم را باز میکنیم. بحث Interaction بین مجموعه ای از Objectهای سیستم را میتواند توسط یک Dynamic Collaboration برای ما مشخص کند.
   4. **Class Diagram:** انواع ارتباطات بین کلاسها (Assocation, Dependency, Specialization, Generalization) است. پکیج به مجموعه ای از کلاسها گفته می شود.
   5. **(Sequence flow of action)Activity Diagram:** چه Action هایی در پی هم باید اجرا شوند. یکسری گره هایی به عنوان گره های تصمیم گیر Desicion وجود دارند میگویند طبق چه فرضیاتی چه کاری انجام شود. در این دیاگرام باید Trigger ها دیده شوند. Trigger اتفاقاتی هستند که بر اساس آنها کار باید انجام شود.
9. **:Implement View** مباحث مربوط به کدنویسی و Dependency ها که مخاطبش Developer ها هستند میباشد. دیاگرامها:
   1. **Component Diagram (Physical Structure of Code):** کل سیستم به قطعاتی (Component) تقسیم میشود. معمولا قطعات وظیفه های خاص و مستقلی دارند. در این دیاگرام ارتباط بین این قطعات مشخص میشود.
10. **Process View :** دو قسمت Process و Processor دارد. درباره اجرای مولفه های موازی Concurrent است. Throughput به معنی تعداد پردازه ها در واحد زمان است. Scalability به معنی قابلیت بزرگ شدن فعالیتهای سامانه میباشد. در محاسبه پیچیدگی سیستم وقتی Process View سیستم را میبینیم میتوانیم بفهمیم سیستم چقدر پیچیده است.
11. **Deployment View :** از لحاظ فیزیکی ارتباطات Device ها و نرم افزار را نشان میدهد که معمولا قطعات سیستمی چه منطقی و چه فیزیکی در کنار هم قرار میگیرند و قطعات به صورت Node نمایش داده میشود. دیاگرامها:
    1. **Deployment Diagram :** این دیاگرام Physical Structure و Node ها را میبیند. مثلا در یک سیستم Application Server و Database Server و Web Server و Printers که همه با TCP/IP به هم متصل هستند برای هر کدام یک Nodeمکعب میکشیم. نرم افزارهایی که روی هر Node سوار هستند را به صورت Package میکشند.
    2. **Composite Structure Diagram :** بحث نقش (Role) است. هر شیئی یک سری Role دارد. ممکن است تعامل بین Roleها خودش کلاس شود.
12. **Model Element ها :**

