

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

UML چیست؟

مدرس : مهندس افشین رفوآ

دوره آموزش UML

UML چیست؟

یک زبان مدلسازی همه منظوره استاندارد زیرمجموعه مهندسی نرم افزار است که توسط **Object management group** ایجاد شد. در حال حاضر آخرین ورژن این زبان ویرایش **2.4.1** است که در سال **۲۰۰۷** مشخصات آن منتشر شد. پیش نویس خصوصیات و امکانات ویرایش 1 آن در سال **1997** به **OMG** ارائه شد. با استفاده از **UML** می توان تقریباً هر گونه برنامه کاربردی که ممکن است بر روی هر ترکیبی از سخت افزار، سیستم عامل، زبان برنامه نویسی و شبکه اجرا شود را الگوسازی نمود. طراحی بر پایه مفاهیم شی گرای باعث می شود که به طور پیش فرض با محیطها و زبانهای برنامه نویسی شی گرا سازگاری و همخوانی کامل داشته باشد، با این حال می توان از آن به منظور مدلسازی برنامه های غیر شی گرا مانند برنامه هایی که با زبانهای بیسیک، کوبول نوشته می شوند نیز استفاده کرد.

UML یک زبان مدلسازی نسل سوم است و روشی باز برای توصیف ویژگی ها، نمایش گرافیکی، ساختن و مستندسازی اجزای یک سیستم نرم افزاری در حال توسعه می باشد. از یو ام ال برای فهمیدن، طراحی، مرور، پیکربندی، نگهداری و کنترل اطلاعات سیستم های نرم افزاری استفاده می شود.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

معرفی UML

مدرس : مهندس افشین رفوآ

دوره آموزش UML

معرفی UML

UML یک زبان استاندارد برای تعریف، نمایش گرافیکی، ساختن و مستندسازی اجزای یک سیستم نرم افزاری کمک گرفت.

این زبان در سال 1997 هنگامی که پیش نویس مشخصات آن به گروه مدیریت شی (OMG) ارائه شد، با ویرایش 1 پا به عرصه گذاشت.

OMG سعی بر هر چه بهتر کردن این زبان دارد. در زیر توضیحاتی را درباره ی این زبان مشاهده می کنید:

1. **UML** سرواژه ی **Unified Modeling Language** می باشد.

2. **UML** از دیگر زبان های رایج برنامه نویسی مانند **C++، Java** و **COBOL** متفاوت است.

3. **UML** یک زبان تصویری، نمایشی است که از آن جهت مدل سازی و ساخت برنامه ی کار نرم افزار استفاده می شود.

بنابراین **UML** را می توان به عنوان یک زبان دیداری مدل سازی همه منظور تلقی کرد که توسط آن سیستم نرم افزاری نمایش، تعریف، ساخته و مستندسازی می شود. اگرچه **UML** بیشتر جهت مدل سازی سیستم ها نرم افزاری بکار می رود، اما می توان از آن در زمینه های دیگر مانند مدل سازی جریان پردازش در یک واحد تولید بهره گرفت.

UML به خودی خود یک زبان برنامه نویسی نیست اما ابزاری است که با استفاده از نمودارهای آن می توان به زبان های مختلف کد نوشت. **UML** یک رابطه ی مستقیم با تجزیه و تحلیل، طراحی شی گرا دارد. پس از کمی متعارف سازی، **UML** به ی استاندارد **OMG** تبدیل شده است.

اهداف UML

مفایم شی گرا بسیار قبل تر از **UML** معرفی شدند. در آن زمان هیچ روشی برای سازمان دهی و تحکیم برنامه نویسی شی گرا وجود نداشت. **UML** برای این منظور پا به عرصه گذاشت.

UML برای اهداف مختلفی تعبیه و عرضه شد، اما مهمترین آن تعریف یک زبان همه منظوره ی مدل سازی بود که تمامی طراحانی که مدل سازی انجام می دهند بتوانند از آن استفاده کنند.

نمودارهای **UML** تنها توسط برنامه نویسان یا توسعه دهندگان استفاده نمی شود، بلکه تمامی طراحان و همچنین مردم عادی که علاقه مند به مدل سازی و فهم سیستم هستند می توانند از آن استفاده کنند. این سیستم می تواند یک سیستم نرم افزاری یا غیر نرم افزاری باشد.

UML را نمی توان به عنوان یک روش تولید نرم افزار کامل دانست. این زبان شامل فرایند مرحله به مرحله تولید نرم افزار نیست، بلکه **UML** زبانی است که تقریباً تمام شیوه های تولید نرم افزار از آن استفاده می کنند. در پایان باید گفت که **UML** یک مکانیزم **modeling** ساده برای مدل سازی تمامی سیستم های کاربردی در محیط پیچیده ی امروز محسوب می شود.

مدل یا الگوی ذهنی از UML (Conceptual model)

برای اینکه بفهمیم مدل ذهنی از **UML** چیست، ابتدا می بایست تعریف روشن و واضحی از آن ارائه کنیم و همچنین علت نیاز به آن را مورد بررسی قرار دهیم.

1. یک **conceptual model** را می توان یک مدل در نظر گرفت که از مفاهیم و رابطه ی میان آن مفاهیم تشکیل شده است.

2. مدل ذهنی اولین گامی است که پیش از ترسیم نمودار **UML** بایستی به آن پرداخت. این مدل در درک موجودیت ها و نحوه ی تعامل بین آن ها در جهان واقع کمک می کند.

از آنجایی که **UML** سیستم های زمان واقعی (**real time system**) تعریف می کند، بایستی ابتدا یک مدل ذهنی از آن ترسیم/ایجاد کرده و از آن جا باقی مراحل را دنبال کرد. لازم است برای درک کامل مدل ذهنی، سه المان اصلی زیر را فراگیرید:

1. بخش های اصلی و تشکیل دهنده ی **UML**

2. قوانین لازم برای ربط دادن این بخش های تشکیل دهنده

3. مکانیزم های رایج **UML**

مفاهیم شی گرا

می توان گفت که **UML** دنباله روی طراحی، تجزیه و تحلیل شی گرا می باشد.

یکی شی در واقع دربردارنده ی داده ها و متدهایی است که آن داده ها را کنترل می کنند. داده های ذخیره شده در شی، در واقع وضعیت آن را نمایش می دهد. کلاس یک شی تعریف می کند و سلسه مراتبی که بر اساس آن سیستم واقعی مدل سازی می شود را شکل می دهد. سلسله مراتب به صورت وراثت نشان داده می شود و کلاس ها بر اساس نیاز به هم ربط داده می شوند.

اشیا موجودیت های واقعی اطراف ما هستند. مفاهیم پایه ای مانند **abstraction**، کپسوله سازی (**encapsulation**)، وراثت (**inheritance**) و چند ریختی (**polymorphism**) را می توان به وسیله ی **UML** نمایش داد.

از این رو می توان گفت که **UML** چنان قدرتمند است که می تواند تمامی مفاهیم موجود در تجزیه، تحلیل و طراحی شی گرا را نمایش دهد. نمودارهای **UML** تنها نشانگر مفاهیم پایه ای شی گرا هستند. از این رو می بایست پیش از فراگیری **UML**، با مفاهیم شی گرا آشنایی پیدا کرد.

در زیر برخی از مفاهیم اساسی دنیای شی گرا شرح داده شده است:

1. شی: شی نشانگر یک موجودیت و عضو اساسی می باشد.

2. کلاس: کلاس طرحی از یک شی است.

3. انتزاع (**abstraction**): نشانگر رفتار یک موجودیت در دنیای واقعی است.
4. کپسوله سازی (**encapsulation**): مکانیزم متصل کردن داده ها به یکدیگر و پنهان سازی آن ها از دنیای خارج است.
5. وراثت (**inheritance**): مکانیزم ایجاد کلاس های جدید از یک کلاس موجود می باشد.
6. چندریختی (**polymorphism**): مفهوم چندریختی ویژگی است که به رابطه ها امکان می دهد تا برای گروهی از عملیات ها مورد استفاده قرار گیرند.

تجزیه و تحلیل oop

تجزیه و تحلیل شی گرا را می توان یک نوع تحقیق و بررسی یا به طور دقیق تر بررسی دقیق اشیا دانست. طراحی یعنی همکاری و رابطه ی بین اشیا شناسایی شود.

از این رو درک تحلیل و طراحی شی گرا از اهمیت بالایی برخوردار است. لازم است توجه داشته باشید که مهم ترین هدف تجزیه و تحلیل شی گرا شناسایی اشیا است که می بایست طراحی شود. گفتنی است که این تجزیه و تحلیل برای سیستم موجود نیز انجام می شود. حال یک تجزیه و تحلیل موثر صرفاً زمانی امکان پذیر می باشد که ما بتوانیم اشیا را شناسایی کنیم. پس از شناسایی اشیا، رابطه ی بین آن ها را شناسایی کرده و در نهایت **design** را تولید می کنیم.

می توان هدف تجزیه و تحلیل شی گرا را در موارد زیر خلاصه نمود:

1. شناسایی اشیا سیستم.
2. شناسایی رابطه ی میان آن ها.
3. ایجاد یک طرح که بتوان به وسیله ی زبان های شی گرا به فایل های اجرایی تبدیل نمود.

در کل سه گام وجود دارد که طی آن مفاهیم شی گرا اعمال و پیاده سازی می شوند. این گام ها به ترتیب زیر می باشند:

OO Analysis --> OO Design --> OO implementation using OO languages

پیاده سازی شی گرا به وسیله ی زبان های شی گرا → طراحی شی گرا → تجزیه و تحلیل شی گرا

حال سه گام بالا را به تفصیل شرح می دهیم:

1. در این مرحله (تحلیل شی گرا) هدف اصلی که دنبال می شود، شناسایی اشیا و توصیف آن ها به شیوه ی صحیح است. اگر این اشیا به صورت کارآمد شناسایی شوند، گام بعدی طراحی آسان می باشد. اشیا می بایست همراه با مسئولیت های آن ها شناسایی شوند. مسئولیت ها در واقع کارها یا وظایفی هستند که شی انجام می دهد. هر شی ای مسئولیت های خاص خود را داشته و وظایف مربوط به خود را انجام می دهد. هنگامی که این مسئولیت ها، هماهنگ شده و رابطه ی بین آن ها شکل می گیرد، مقصود اصلی سیستم برآورده می شود.
2. مرحله ی دوم طراحی یا **design** شی گرا می باشد. در این مرحله تاکید بر روی نیازها و برآورده کردن آن نیازهاست. همچنین در این گام اشیا با توجه به رابطه ی قبلا تعیین شده همکاری می کنند. با تکمیل رابطه ی بین آن ها، طراحی نیز کامل می شود.
3. مرحله ی سوم پیاده سازی شی گرا است. در این مرحله طراحی به واسطه ی زبان های شی گرا نظیر **Java**، **C** و **C++** پیاده سازی می شود.

نقش UML در طراحی شی گرا (OO design)

همان طور که پیش تر گفته شد، **UML** یک زبان مدل سازی سیستم های نرم افزار و غیر نرم افزاری می باشد. اگرچه **UML** برای سیستم های غیر نرم افزاری نیز بکار می رود، تاکید آن (کاربرد اصلی آن) بر روی مدل سازی نرم افزارهای کاربردی شی گرا می باشد. بیشتر نمودارهای **UML** که تاکنون درباره ی آن ها بحث شد، به منظور مدل سازی مفاهیم و جنبه های مختلف همچون **static** یا **dynamic** بکار می رفتند. حال می گوئیم جنبه هر چه می خواهد باشد، اجزا چیزی به جز اشیا نیستند.

اگر نمودار کلاس، نمودار شی، نمودار همکاری (**collaboration diagram**) و نمودارهای تعامل را با دقت مورد بررسی قرار دهیم، متوجه می شویم که تمامی آن ها بر اساس اشیا طراحی می شوند.

درک رابطه ی بین طراحی شی گرا و **UML** بسیار حائز اهمیت می باشد. طراحی شی گرا با توجه به نیاز به نمودارهای **UML** تبدیل می شوند. پیش از درک کامل **UML**، می بایست بر مفاهیم شی گرا اشراف پیدا کرد. پس از با

موفقیت پشت سر گذاشتن تحلیل و طراحی شی گرا، گام بعدی بسیار آسان می باشد. خروجی تحلیل و طراحی شی گرا ورودی نمودارهای UML می باشد.

Tahildadeh

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

بخش های اساسی و تشکیل دهنده ی UML

مدرس : مهندس افشین رفوآ

دوره آموزش UML

بخش های اساسی و تشکیل دهنده ی UML

از آنجایی که UML سیستم های بی وقفه و زمان واقعی (real time) تعریف می کند، ایجاد یک مدل ذهنی (conceptual model) پیش از هر کاری از اهمیت بالایی برخوردار است. جهت فراگیری مدل ذهنی UML، لازم است بر المان های اساسی آن اشراف داشته باشید:

1. اجزا و المان های اصلی تشکیل دهنده ی UML

2. قواعد به هم متصل کردن و ربط دادن این اجزا

3. مکانیزم و سازوکارهای معمول UML

این مبحث تمامی اجزای اصلی زبان مدل سازی UML را شرح می دهد. اجزای اصلی این زبان عبارتند از:

1. اشیا (Things)

2. رابطه ها (Relationships)

3. نمودارها (Diagrams)

1 اشیا (Things)

اشیا (thing) مهمترین اجزا UML هستند. اشیا می توانند:

1. ساختاری (structural)

2. رفتاری (behavioral)

3. grouping (مربوط به گروه بندی)

4. annotational (مربوط به نشان گذاری و حاشیه نویسی)

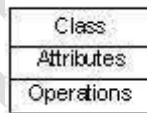
باشند.

Structural things

بخش ایستا (static) مربوط به مدل را تعریف می کند. Things نشانگر المان های فیزیکی و ذهنی هستند. در زیر توضیحات مختصری درباره ی اشیا ساختاری/structural things مشاهده می کنید:

Class (کلاس)

کلاس نشانگر مجموعه ای از اشیاست که دارای مسئولیت های مشابه هستند.



Interface (رابط)

رابط یا interface یک سری عملیات تعریف می کند که مسئولیت هر کلاس را مشخص می نماید.



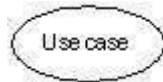
Collaboration (همکاری)

Collaboration همکاری یا تعامل بین المان ها را تعریف می کند.



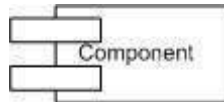
Use case (مورد کاربرد)

Use case نشانگر یک سری عملیات است که توسط سیستم برای نیل به هدف خاصی اجرا می شود.



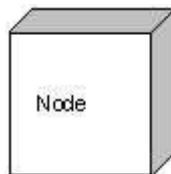
Component (مولفه)

مولفه نشانگر یا تعریف کننده ی بخش فیزیکی سیستم می باشد.



Node (گره)

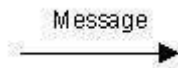
Node را می توان یک عنصر فیزیکی در نظر گرفت که در زمان اجرا بوجود می آید.



Behavioral things

Behavioral thing متشکل هست از بخش های پویا (**dynamic part**) مدل های UML. **Behavioral thing** ها خود به مولفه های زیر تقسیم می شوند:

1. **interaction** (تعامل): Interaction یک رفتار یا behavior است که از تعدادی پیغام تشکیل می شود. این پیغام ها در راستای نیل به هدف خاصی بین المان ها رد و بدل می شوند.



2. **state machine** (ماشین وضعیت): زمانی بکار می آید که وضعیت یک شی در چرخه ی حیاتش (**life cycle**) از اهمیت خاصی برخوردار باشد. **state machine** دنباله ای از وضعیت های مختلف است که یک شی در پاسخ به رخدادهای (**event**) متفاوت تجربه می کند. رخدادها عوامل خارجی هستند که تغییر در وضعیت (شی) از آن ها ناشی می شود.



Grouping thing (شی گروه بندی)

شی گروه بندی عبارت است از یک مکانیزم که وظیفه ی گروه بندی المان ها یک مدل UML را بر عهده دارد. تنها یک شی گروه بندی وجود دارد و آن Package می باشد.

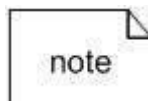
Package : تنها وسیله ای که می توان با استفاده از آن Behavioral thing و Structural thing را گروه بندی کرده و به صورت یک بسته درآورد.



Annotational things (شی حاشیه نویسی)

یک سازوکار برای ضبط description (توصیفات)، comment (توضیحات)، remark (نظرات) المان های مدل UML می باشد. Note تنها ابزار حاشیه نویسی می باشد.

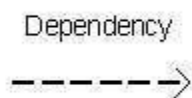
Note : یک شی یا ابزار است که امکان نمایش (render) توضیحات، محدودیت ها (constraint) مربوط به المان UML را فراهم می آورد.



Relationship (2) (رابطه)

Relationship یکی دیگر از اجزای اساسی زبان مدل سازی UML می باشد. Relationship نشان می دهد چگونه المان ها با یکدیگر رابطه دارند و به هم متصل هستند، این رابطه نیز قابلیت برنامه را تعریف می کند. در کل چهار نوع relationship وجود دارد که به شرح زیر می باشد.

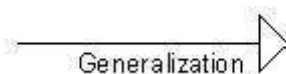
Dependency (رابطه ی وابستگی): **Dependency** یک رابطه بین دو شی است که در آن دو شی به هم وابسته هستند و تغییر در یکی سبب تغییر در دیگری می شود.



Association (رابطه ی انجمنی): **association** در اصل یک سری پیوند است که عناصر یک مدل **UML** را به هم متصل می کند. این نوع رابطه همچنین بیان می کند چه تعداد شی در آن رابطه شرکت دارند.



Generalization (رابطه وراثت): **Generalization** یک نوع رابطه هست که یک المان اختصاصی را به یک المان عمومی ربط می دهد (بیانگر رابطه ی جز به کل می باشد). در واقع رابطه ی وراثت را در جهان اشیا توصیف می کند.



Realization (رابطه ی تحقق): **Realization** را می توان یک رابطه تعریف کرد که در آن دو المان به هم وصل هستند. یک رابطه **Realization** از عنصر مبدا (عنصر محقق سازی) به عنصر مقصد (عنصر خصوصیت) نشان می دهد که عنصر مبدا حداقل همه عملیات عنصر مقصد را پشتیبانی می کند. (عنصر مبدا می تواند صفات یا پیوندهای عنصر مقصد را پشتیبانی کند). این رابطه را می توان در **interface** سراغ گرفت.

3) نمودارهای UML

نمودارهای **UML** خروجی یا همان نتیجه ی نهایی کل این مبحث محسوب می شود. تمامی المان ها، رابطه ها باهم ترکیب شده و یک نمودار کامل **UML** و نمودار هم در نتیجه سیستم را تشکیل می دهد.

آن نموداری که به عنوان خروجی مشاهده می کنید و تمامی عناصر ذکر شده در تکمیل آن دخیل بوده و شرکت دارند، مقصود نهایی بوده و مهم ترین بخش پروسه می باشد.

UML در مجموع 9 نمودار به عنوان خروجی عرضه می کند. در مباحث بعدی جزئیات مربوط به این نه نمودار را به تفصیل شرح خواهیم داد.

1. نمودار کلاس (**class**)

2. نمودار شی (**object**)

3. نمودار مورد کاربرد (**use case**)

4. نمودار توالی (**sequence**)

5. نمودار همکاری (**collaboration**)

6. نمودار فعالیت (**activity**)

7. نمودار وضعیت (**statechart**)

8. نمودار **deployment** (اسقرار)

9. نمودار اجزا (**component**)

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

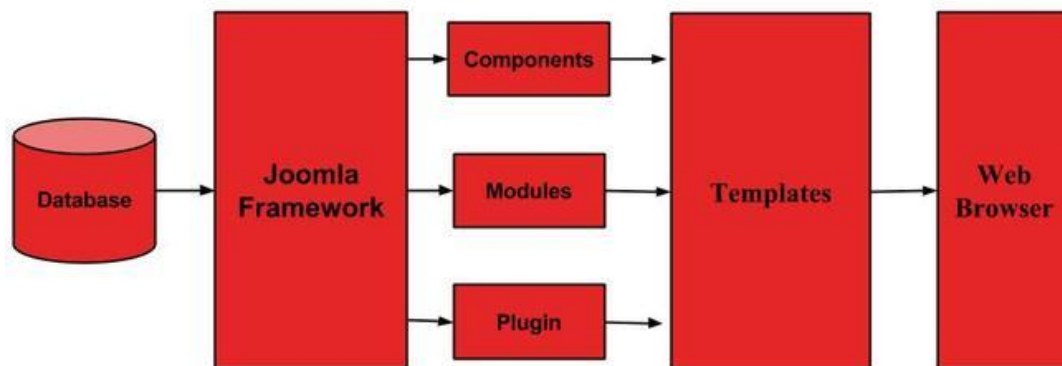
معماری Joomla

مدرس : مهندس افشین رفوآ

دوره آموزشی جوملا

معماری Joomla

جوملا یک برنامه ی تحت وب مبتنی بر الگوی طراحی **Model-View-Controller** می باشد. در این مبحث، معماری جوملا را به تفصیل شرح خواهیم داد.



معماری جوملا به لایه های زیر بخش پذیر می باشد:

1. **database**

2. **Joomla Framework**

3. **Components**

4. **Modules**

5. **Plugin**

6. **Templates**

7. **Web Server**

لایه ی **Database – database** یا پایگاه داده به یک مجموعه ی بزرگ از اطلاعات گفته می شود که قابل ذخیره سازی، دستکاری و سازمان دهی به شکل خاص می باشد. تمامی اطلاعات مربوط به کاربر و دیگر محتویات ضروری سایت در این لایه ذخیره می شوند. به عبارت بهتر اطلاعات لازم برای مدیریت سایت همگی داخل جداول پایگاه داده ذخیره و نگهداری می شود.

با توجه به آنچه گفته شد، لایه ی پایگاه داده انعطاف پذیری و قابلیت توسعه را برای این سیستم مدیریت محتوا تضمین می کند.

Joomla Framework – فریم ورک جوملا یک مجموعه متشکل از برنامه یا نرم افزارهای متن باز (**open source**) و رایگان است که سیستم مدیریت محتوا ذکر شده بر پایه ی آن بنا شده. این فریم ورک به منظور انعطاف پذیری بالا طراحی شده و همچنین به پکیج های مازول مانندی بخش بندی می شود که در توسعه ی مجزای هر یک بسیار موثر می باشد.

این لایه ی در واقع دربرگیرنده ی توابع درون ساخته، افزونه ها و کتابخانه های جوملا می باشد.

لایه ی **Component** – کامپوننت ها یا مولفه ها اساسا برنامه های کوچکی هستند که با الحاق آن ها به جوملا می توان نیازهای اضافی بر سازمان سایت را پاسخ گفت. این لایه خود به دو بخش تقسیم می شود: 1.

Site Administrator 2. هر بار که صفحه ای بارگذاری می شود، در واقع یک کامپوننت فراخوانی شده که بدنه ی اصلی سایت را نمایش می دهد (**render** می کند). مدیریت جنبه های مختلف کامپوننت بر عهده ی بخش **Admin** می باشد. زمانی که کاربری یک صفحه از سایت را باز می کند، یک درخواست به سرور ارسال می شود. آن بخش کامپوننت که مربوط به سایت است نمایش صفحه ی مورد درخواست را بر عهده دارد. کامپوننت ها در اصل الحاقات اصلی سایت محسوب می شوند.

لایه ی **Module** – ماژوال یک افزونه است که برای نمایش صفحات در جوئلا بکار می رود. همچنین به وسیله ی آن می توان داده های جدید را از کامپوننت خواند. ماژوال ها اغلب ظاهری گادر مانند دارند، به عنوان مثال می توان به ماژوال **login** اشاره کرد. برای مدیریت ماژول ها بایستی به **module manager** در بخش **admin** سایت مراجعه کرد. با لینک شدن ماژول ها به کامپوننت ها، محتوا از جمله تصاویر جدید به نمایش در می آید. **Plugin** – پلاگین نیز یک نوع افزونه ی جوئلا محسوب می شود، بدین معنا که به انعطاف پذیری جوئلا افزوده و با اضافه کردن قابلیت های جدید می تواند **framework** جوئلا را توسعه دهد. پلاگین ها حاوی تکه هایی هستند که به منظور فعال سازی رخدادها بکار می رود. از پلاگین ها غالباً برای فرمت دهی خروجی یک کامپوننت یا ماژول، زمانی که یک صفحه به نمایش در می آید، استفاده می شود. آن قسم از کاربرد پلاگین که مربوط به **event** ها می باشد، با فعال شدن رخداد به ترتیب اجرا می شوند. با پلاگین می توان در کدهای متعلق به کامپوننت ها یا هسته ی جوئلا، البته بدون ویرایش ساختار کلی آن ها، تغییراتی را ایجاد کرد.

Template – قالب یا **Template** ظاهر وب سایت را تعیین می کند. **template** خود به دو دسته تقسیم می شود: 1. **Front-end** 2. **Back-end**. قالب سمت **backend** برای کنترل ناحیه ی مدیریتی و **Admin** سایت طراحی شده، درحالی که قالب سمت **front-end** به منظور نمایش وب سایت به کاربران مورد استفاده قرار می گیرد. ساخت **Template** بسیار آسان بوده و این امکان را به شما می دهد تا ظاهر سایت را طبق سلیقه ی خود سبک دهی نمایید.

Web Browser – مرورگر وب یک خدمت رسان است که بستر تعامل کاربر با سایت را فراهم می آورد. به عبارت بهتر صفحات وب را در اختیار سرویس گیرنده قرار می دهد. مرورگر وب برای برقراری امکان ارتباط بین سرویس گیرنده (**client**) و سرویس دهنده (**server**) از پروتکل **HTTP** بهره می گیرد.



بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

انواع مدل سازی (Modeling types)

مدرس : مهندس افشین رفوآ

دوره آموزش UML

انواع مدل سازی (Modeling types)

از آنجایی که در UML، هر نمودار ویژه ی مدل سازی خاصی در UML بکار می رود، می بایست بین مدل های مختلف تفاوت قائل شد. سه نوع مدل سازی در UML کاربرد دارد که به شرح زیر می باشد:

مدل سازی ساختاری (Structural modeling)

مدل سازی ساختاری در واقع تصویری از ویژگی های ایستای (static feature ها) یک سیستم ارائه می دهد. این نوع مدل سازی نمودارهای زیر را شامل می شود:

1. نمودارهای کلاس ها (Class diagrams)

2. نمودارهای اشیا (object diagram)

3. نمودارهای توزیع و استقرار (deployment)

4. نمودارهای بسته بندی (Package)

5. نمودار ساختار ترکیبی (Composite structure)

6. نمودار اجزا (component diagram)

مدل ساختاری در واقع مبنا یا چارچوب سیستم را نمایش می دهد و این چارچوب جایی است که تمامی دیگر اجزا و مولفه ها در آن جای می گیرند (وجود دارند). بنابراین نمودار کلاس، نمودار مولفه و نمودارهای استقرار

همگی بخشی از مدل سازی ساختاری هستند. تمامی نمودارهای ذکر شده، المان ها و سازوکار سرهم بندی آن ها را نمایش می دهد.

مدل ساختاری هیچگاه رفتار پویای (**dynamic behavior**) سیستم را شرح نمی دهد. نمودار کلاس پر کاربرد ترین نمودار ساختاری می باشد.

مدل سازی رفتاری (behavioral modeling)

مدل رفتاری، تعامل (**interaction**) در سیستم را توصیف می کند یا به عبارتی، همان طور که از اسم آن پیدا است، تعامل بین نمودارهای ساختاری را نشان می دهد. مدل سازی رفتاری، ماهیت پویای سیستم را بیان می کند. مدل رفتاری از نمودار زیر تشکیل می شود:

1. نمودارهای فعالیت (**activity diagram**)

2. نمودارهای تعامل (**interaction diagram**)

3. نمودارهای مورد کاربرد (**Use case**)

تمامی موارد نام برده در فهرست بالا، توالی پویای روند (**flow**) در یک سیستم را نشان می دهد.

مدل سازی معماری (Architectural Modeling)

مدل معماری چارچوب کلی یک سیستم را نمایش می دهد. این مدل دربردارنده ی المان های رفتاری و ساختاری سیستم می باشد. مدل معماری را می توان طرح یا برنامه ی کار کل سیستم در نظر گرفت. نمودار **Package** (بسته بندی) تحت (**architectural modeling**) مدل سازی معماری قرار می گیرد.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

نشان گذاری های پایه در UML

مدرس : مهندس افشین رفوآ

دوره آموزش UML

نشان گذاری های پایه در UML (Notation)

یکی از دلایل محبوبیت UML، در نشان گذاری های نموداری (diagrammatic notation) آن نهفته است. همه ی ما می دانیم که UML یک زبان مدل سازی یکپارچه برای نمایش و به تصویر کشیدن، مشخص کردن، ساختن و مستندسازی اجزا یا مولفه های سیستم های نرم افزاری و غیر نرم افزاری می باشد. گفتنی است که در اینجا آنچه از همه مهمتر است، فراگیری و درک Visualization (نمایش تصویری) می باشد.

نشان گذاری UML مهمترین عنصر در مدل سازی به حساب می آید. استفاده ی مناسب و کارآمد از نشان گذاری در ساختن یک مدل کامل و معنی دار از اهمیت خاصی برخوردار است. یک مدل، اگر در نمایش دادن و شرح مقصود نهایی خود موفق نباشد، کاملاً ناکارآمد و بیپوده خواهد بود.

از این رو، یادگیری نشان گذاری (notation) می بایست از همان ابتدای امر مورد تأکید قرار گیرد. برای اشیا (things) و رابطه ها (relationship)، طبیعتاً از نشان گذاری های متفاوت بهره گرفته می شود و نمودارهای UML نیز در نهایت از همین نشان گذاری های اشیا و رابطه ها ایجاد می شوند. توسعه پذیری یکی دیگر از ویژگی ها و قابلیت های مهم می باشد که UML را قدرتمند و انعطاف پذیر می سازد.

فصل حاضر نشان گذاری های کلی UML را به تفصیل شرح می دهد. این بخش در واقع افزونه ای برای مبحث قبلی می باشد، بنابراین برای یادگیری آن می بایست درس پیشین را فراگرفته باشید.

اشیا ساختاری (Structural things)

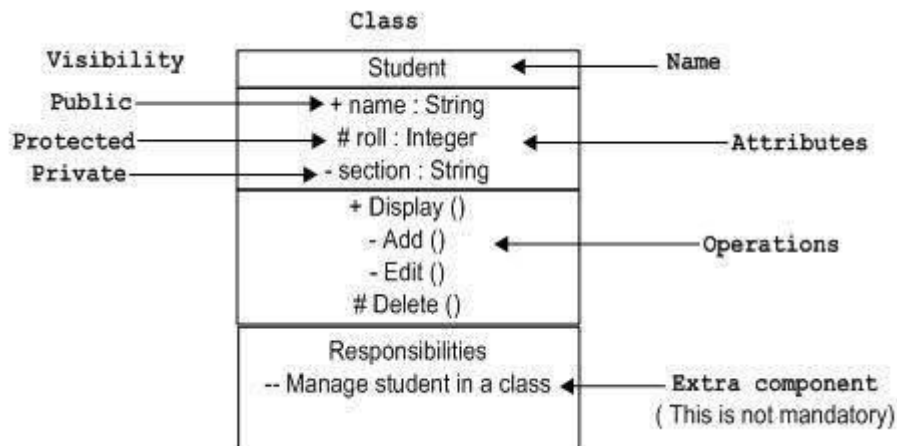
نشان گذاری های ترسیمی (**graphical notations**) پرکاربردترین در **UML** می باشد. اینها به نوعی نقش اسم را در مدل های **UML** ایفا می کنند. در زیر فهرستی از اشیا ساختاری را مشاهده می کنید:

1. کلاس ها
2. شی
3. **interface** (رابط)
4. **collaboration**
5. **Use case**
6. **active class** ها
7. **Component** ها (مولفه ها)
8. **node** ها (گره ها)

نشان گذاری کلاس

نمودار زیر، کلاس **UML** را به نمایش می گذارد. نمودار حاضر به چهار بخش تقسیم شده:

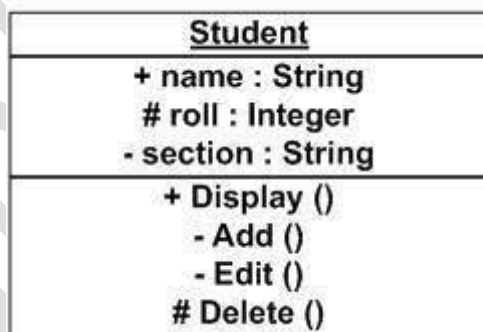
1. بالاترین بخش نام کلاس را مشخص می کند.
2. دومین بخش خصیصه ها یا **attribute** های کلاس را نمایش می دهد.
3. بخش سوم عملیاتی که کلاس مورد نظر قادر به انجام آن ها است را توصیف می کند.
4. آخرین بخش نیز اختیاری بوده و جهت نشان دادن مولفه های اضافی بکار می برد.



کلاس ها برای نمایش اشیا بکار می روند. اشیا می توانند هر چیزی که دارای رفتار و خاصیت هایی (property) (ها) هستند، باشند.

نشان گذاری شی (object notation)

شی نیز همانند کلاس نمایش داده می شود. تنها تفاوت، همان طور که در تصویر زیر می بینید، در اسم آن هاست.

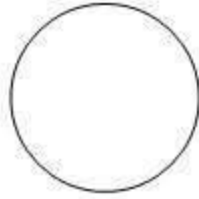


به این خاطر که شی در واقع پیاده سازی عملی یک کلاس هست، که به عنوان نمونه ای از آن کلاس شناخته می شود، مورد استفاده ی آن با کلاس یکسان می باشد.

نشان گذاری رابط (interface notation)

همان طور که در تصویر زیر مشاهده می کنید، **interface** توسط یک دایره نمایش داده می شود. این نمودار دارای یک اسم می باشد که در زیر آن درج می گردد.

Interface



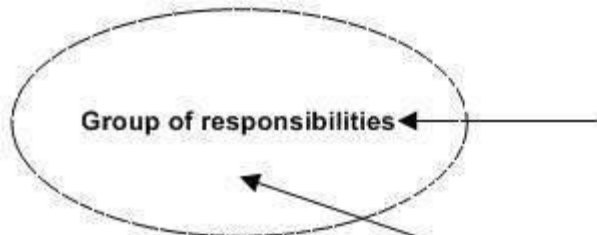
StudentApplication ← Name

مورد استفاده ی رابط (interface)، توصیف قابلیت بدون پیاده سازی می باشد. رابط دقیقا شبیه به یک قالب (template) می باشد که در آن کاربردهای مختلف تعریف می شود، اما در آن از پیاده سازی (implementation) خبری نیست. هنگامی که کلاس رابط را پیاده سازی می کند، همراه با آن و با توجه به نیاز قابلیت ها را نیز implement می کند.

Collaboration Notation (نشان گذاری همکاری)

Collaboration همان طور که در تصویر زیر نظاره گر آن هستید، توسط یک **ellipsis** نقطه چین (شکل بیضی نقطه چین) نمایش داده می شود. در داخل این شکل بیضی مجموعه مسئولیت ها درج می گردند.

Collaboration

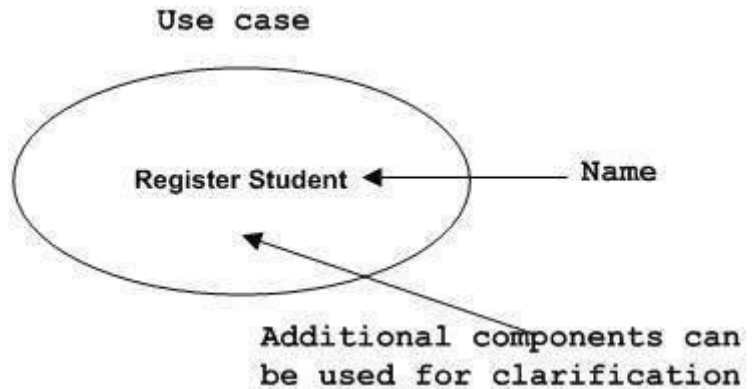


Additional components can be used for clarification

نمودار **collaboration** (همکاری) این مسئولیت ها را به نمایش می گذارد. معمولا مسئولیت ها زیر مجموعه ی یک گروه قرار می گیرند.

نشان گذاری Use case

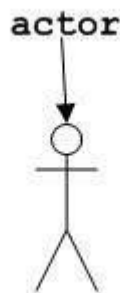
Use case به صورت یک شکل بیضی ترسیم می شود که اسمی داخل آن درج شده. در این نمودار همچنین ممکن است مسئولیت های اضافی بر سازمان جهت روشن سازی نوشته شود.



Use case در اصل به منظور نمایش دادن (و شناسایی) قابلیت ها و کاربردهای سطح بالای یک سیستم بکار می رود.

Actor Notation

می توان گفت که **actor** یک موجودیت داخلی یا خارجی است که با سیستم تعامل می کند.

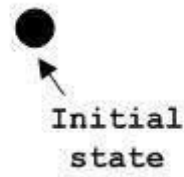


Actor در یک نمودار **Use case** جهت توصیف موجودیت های داخلی و خارجی مورد استفاده قرار می گیرد.

نشان گذاری Initial State

Initial State در واقع وضعیت اولیه یا شروع یک فرآیند را نمایش داده و تعریف می نماید. این نشان گذاری در

تقریباً تمامی نمودارها بکار می رود.



همان طور که از اسم آن پیدا است، کاربرد این نشان گذاری در مشخص کردن آغاز یک پروسه یا شروع فرایند خلاصه می شود.

نشان گذاری Final State

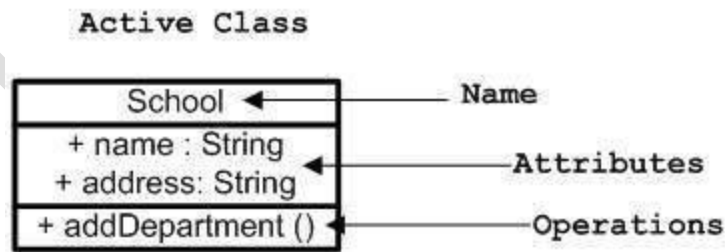
Final State همان طور که از اسم آن می تواند استنباط کرد، پایان یک پروسه یا فرایند را توصیف می کند. این نشان گذاری تقریباً در تمامی نمودارها برای مشخص کردن نقطه ی پایان پروسه بکار می رود.



بنابراین کاربرد **final state notation** در نمایش پایان فرایند خلاصه می شود.

نشان گذاری Active class

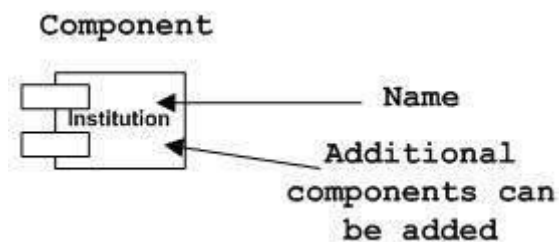
Active class بسیار شبیه به یک کلاس است که خط حاشیه ی ممتد آن را محصور می کند. این نوع نشان گذاری به طور معمول جهت به نمایش گذاشتن رفتار و عملکرد همروند (**concurrent behavior**) سیستم بکار می رود.



بنابراین مورد استفاده ی **Active class** در نمایش همروندی سیستم نهفته است.

نشان گذاری Component

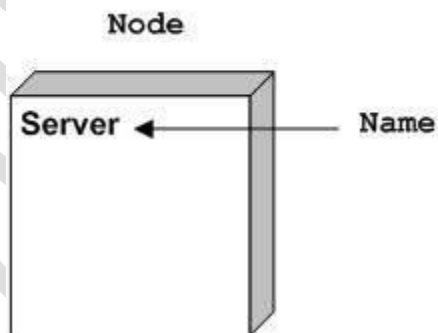
Component یا جز و یا مولفه در UML به صورت زیر و با یک اسم درج شده در داخل آن نمایش داده می شود. در صورت نیاز، می توان المان های اضافی بر سازمان را در هر جایی که لازم بود اضافه نمود.



Component به منظور نمایش دادن بخش های سیستم که نمودار های UML برای آن ترسیم و ایجاد می شود، بکار می رود.

Node notation

در UML، گره یا **Node** را به همراه اسم آن با یک جعبه ی مربع شکل نمایش می دهیم. یک node نشانگر جز فیزیکی (**physical component**) سیستم می باشد.



بنابراین با **node** می توان مولفه های فیزیکی سیستم همچون **server**، **network** و غیره را نمایش داد.

Behavioral things (بخش های پویا و دینامیک مدل های UML)

بخش های دینامیک یکی از مهمترین المان ها در UML هستند. UML با قابلیت ها و امکانات قدرتمندی که دارد، قادر است به آسانی بخش های پویای یک سیستم نرم افزاری یا غیر نرم افزاری را نمایش دهد. این قابلیت ها شامل **interaction** و **state machine** می باشد.

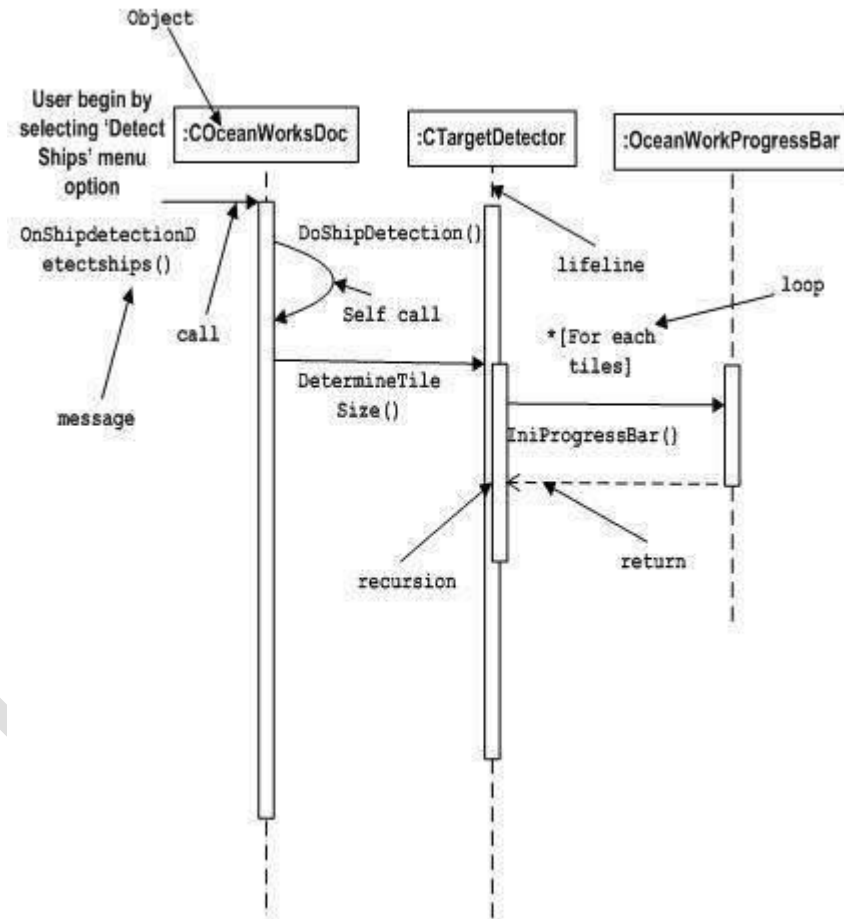
Interaction خود به دو نوع تقسیم می شود:

1. **Sequential** یا ترتیبی (توسط نمودار توالی **sequence** نمایش داده می شود)

2. **collaborative** یا مشترک (توسط نمودار همکاری **collaboration** به نمایش گذاشته می شود)

Interaction notation

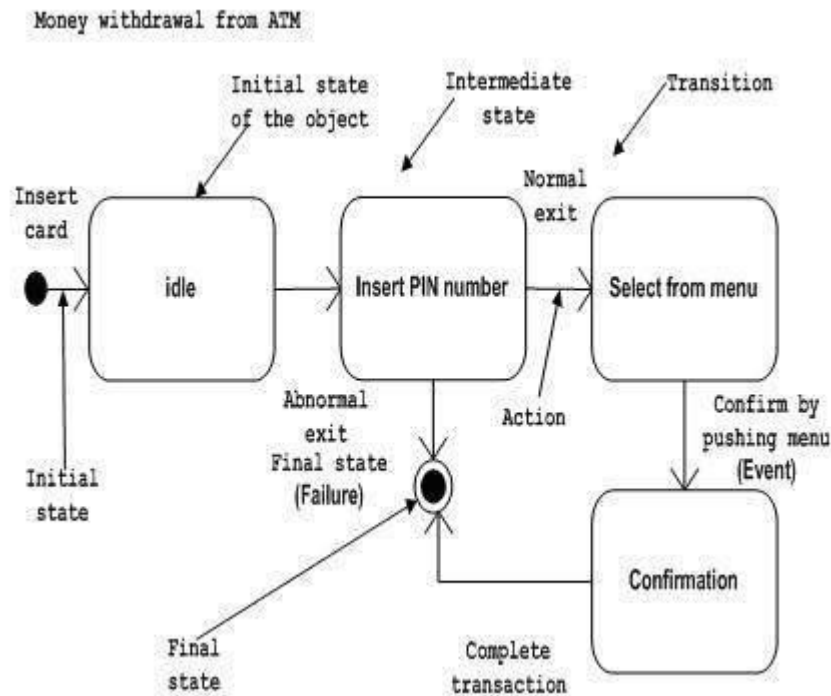
Interaction در واقع همان تبادل پیام بین دو جز یا مولفه ی **UML** می باشد. نمودار زیر نشان گذاری های مختلف که در یک **interaction** شرکت داشته و بکار می رود را نمایش می دهد.



Interaction جهت نمایش ارتباط و تعامل بین اجزا مختلف سیستم استفاده می شود.

State machine Notation

State machine به منظور شرح وضعیت های مختلف یک **component** در چرخه ی حیات (**life cycle**) آن بکار می رود. **Notation** ها در نمودار زیر نشان و شرح داده شده اند:



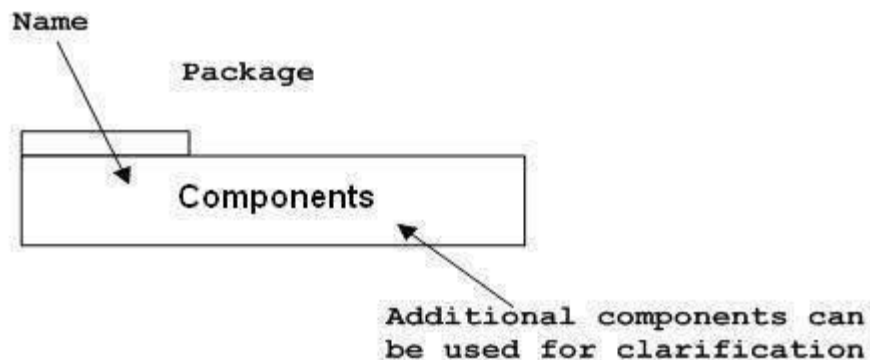
State machine وضعیت های مختلف سیستم و انتقال بین وضعیت ها را نمایش می دهد؛ به عبارتی دیگر ماشین وضعیت برای توصیف وضعیت های مختلف **component** یک سیستم بکار می رود. وضعیت یا **state** می تواند **Active**، **idle** یا بسته به شرایط هر چیز دیگری باشد و بین این ها تغییر کند.

Grouping things (المان های گروه بندی)

سازمان دهی مدل های **UML**، یکی از مهم ترین جنبه های طراحی (**design**) محسوب می شود. در **UML** تنها یک المان ویژه ی گروه بندی وجود دارد و آن هم **Package** می باشد.

package annotation

نشان گذاری **package** در زیر به تصویر کشیده شده است. این گونه نشان گذاری برای دربرگیری دیگر اجزا و **component** های سیستم کاربرد دارد.

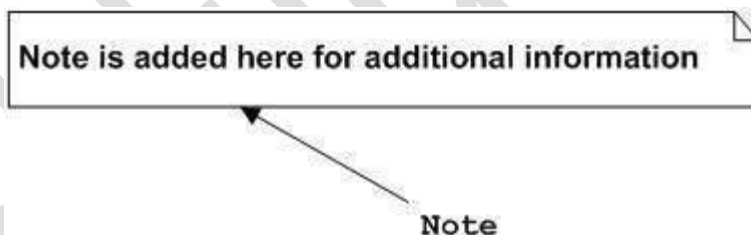


Annotational things (المان های حاشیه نویسی)

در هر نموداری، آنچه بالاترین اهمیت را به خود اختصاص می دهد، تشریح المان ها و قابلیت های آن ها می باشد. **UML** برای این منظور **notes notation** را ارائه می دهد.

Note Notation

این نشان گذاری را می توانید در تصویر زیر مشاهده نمایید. با استفاده از این نشان گذاری می توان تمامی اطلاعات مورد نیاز یک سیستم را فراهم نمود.



Relationships (رابطه ها)

یک مدل کامل قلمداد نمی شود مگر اینکه روابط بین المان های آن به درستی شرح داده شود. این **Relationship** یا رابطه است که معنی یک مدل **UML** را تکمیل می کند. در زیر انواع روابط موجود در **UML** نام برده و شرح داده شده است:

1. **Dependency** (وابستگی)

2. **Association** (انجمنی)

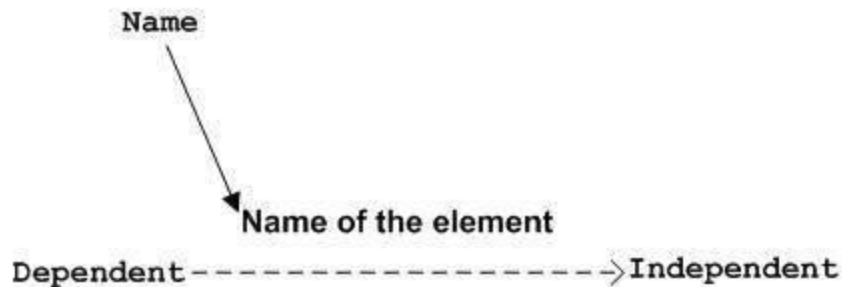
3. **Generalization** (رابطه ی وراثت)

4. **Extensibility** (توسعه پذیری)

Dependency notation

Dependency یکی از جنبه های مهم در المان های **UML** می باشد. این نشان گذاری المان های وابسته را و جهت وابستگی را مشخص می کند.

در تصویر زیر، **Dependency** به وسیله ی یک پیکان نقطه چین مانند نمایش داده شده است. نوک پیکان به المان مستقل اشاره دارد و طرف دیگر آن عنصر وابسته را مشخص می کند.



با توجه به آنچه گفته شد، رابطه ی **Dependency**، وابستگی بین دو المان سیستم را نشان می دهد.

Association notation

Association چگونگی ارتباط بین المان های یک نمودار **UML** را شرح می دهد. به عبارت ساده تر،

Association مشخص می کند چه تعداد المان در یک تعامل (**interaction**) شرکت دارند.

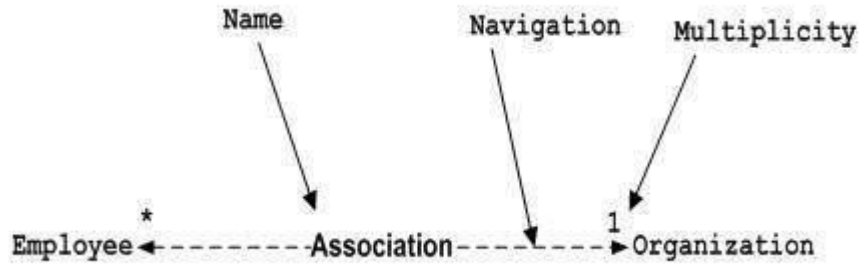
Association توسط خط نقطه چین با یا بدون پیکان در دو طرف خط نمایش داده می شود. دو طرف خط نشانگر

دو المان متصل به هم می باشد. یکی از association های مهم که در تحلیل سیستم وجود دارد تناظر

چندتایی یا **Multiplicity** می باشد : در واقع **multiplicity** تعداد اشیای مرتبط از یک کلاس را با اشیا کلاس

دیگر بیان می کند.

Multiplicity را در تصویر زیر مشاهده می کنید که نشان می دهد چه تعداد شی به هم مرتبط هستند.



با توجه به آنچه گفته شد، **Association** رابطه بین دو المان را در یک سیستم نمایش می دهد.

Generalization Notation

Generalization رابطه ی وراثت در دنیای شی گرا را تشریح می کند یا به عبارتی دیگر بیانگر رابطه ی پدر و فرزندی بین اشیا می باشد.

Generalization در نمودار به صورت یک خط که در سر آن یک پیکان تو خالی قرار می گیرد، نمایش داده می شود. در یک طرف عنصر پدر و در طرف دیگر این خط ممتد المان فرزند نمایش داده می شود.



با استناد به آنچه در بالا گفته شد، **Generalization** رابطه ی پدر-فرزندی بین دو المان در یک سیستم را توصیف می کند.

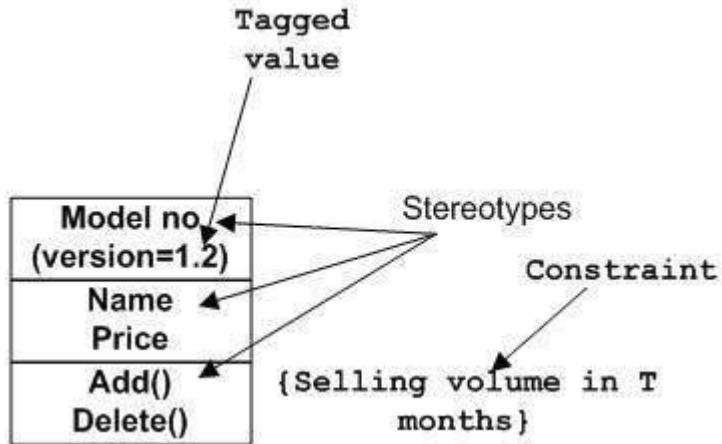
Extensibility Notation

تمامی زبان ها (برنامه نویسی یا مدل سازی) دارای مکانیزم هایی هستند که امکان بسط و توسعه ی قابلیت های آن را فراهم می آورد، از جمله می توان به **semantics**، **syntax** اشاره کرد. **UML** مکانیزم هایی دارد که قابلیت توسعه پذیری به آن می دهد. این سازوکارها عبارتند از:

1. **Stereotype** ها یا کلیشه ها (بیانگر المان های جدید هستند)

2. **tagged values** (نشانهگر خصیصه های جدید می باشد)

3. **Constraint** یا محدودیت (نشان دهنده ی قیود می باشد)



Extensibility notation زمینه ی بسط و افزایش قدرت زبان را مهیا می سازد. **Extensibility notation** ها عملاً المان های اضافی بر سازمان یا الحاقی هستند که تعدادی رفتار و عملکرد اضافی سیستم را معرفی می کند. این رفتارهای اضافه بر سازمان توسط **notation** های متعارف موجود تحت پوشش قرار نمی گیرد.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

نمودارهای رایج در UML

مدرس : مهندس افشین رفوآ

دوره آموزش UML

نمودارهای رایج در UML

در فصل های پیشین درباره ی اجزای بنیادین و دیگر المان های ضروری UML بحث کردیم. حال می بایست بیاموزیم از المان های نام برده در کجا استفاده کنیم.

المان ها مانند Component هایی هستند که می توانند به شیوه های مختلف به یکدیگر متصل شده و یک تصویر کامل را شکل دهند که در نهایت تحت عنوان نمودار یا دیاگرام شناخته می شود. بنابراین درک نمودارهای مختلف برای پیاده سازی دانش مورد نظر در سیستم های واقعی بسیار مهم می باشد.

برای فهم تمام سیستم های پیچیده، نیاز به تعدادی نمودار جامع داریم. این نمودارها کمک شایانی به درک ما می کند. بنابراین اگر به اطراف خود دقت کنیم، متوجه می شویم که نمودارها یک مفهوم جدید نیستند بلکه به شکل های مختلف و در زمینه های مختلف بکار می روند.

نمودارهای UML با این هدف ایجاد می گردند که درک و فهم عمیق تر و آسان تری از سیستم مورد نظر حاصل گردد. با این حال، این امکان وجود ندارد که تمامی جنبه های سیستم را در قالب یک نمودار تحت پوشش قرار داد. به این خاطر زبان مدل سازی UML چندین نوع نمودار ارائه می دهد که به واسطه ی آن ها می توان کلیه ی جنبه های یک سامانه ی پیچیده را تحت پوشش قرار داد.

شما می توانید مجموعه نمودارهای اختصاصی ایجاد نمایید و نیازهای خود را برطرف سازید. نمودارها طی فرایندی تکرارشونده و گام به گام ایجاد می شوند.

در کل دو دسته ی اصلی نمودار وجود دارد که این دو دسته خود به زیردسته هایی تقسیم می شوند:

1. نمودارهای ساختاری (structural diagrams)

2. نمودارهای رفتاری (behavioral diagrams)

نمودارهای ساختاری

نمودارهای ساختاری (به انگلیسی structural diagrams) جنبه های static سیستم مورد نظر را نمایش می دهد. جنبه های ایستا یا استاتیک سامانه در واقع نشانگر آن بخش هایی از یک نمودار است که ساختار اصلی و در نتیجه پایدار سیستم را تشکیل می دهد.

بخش های static توسط کلاس ها، interface ها، اشیا، component ها و node ها نمایش داده می شوند. در کل چهار نوع نمودار ساختاری داریم که در زیر فهرست شده:

1. نمودار کلاس (class diagram)

2. نمودار شی (object diagram)

3. نمودار اجزا (component diagram)

4. نمودار استقرار و توزیع (deployment diagram)

نمودار کلاس (class diagram)

نمودارهای کلاس، رایج ترین و پرکاربرد ترین نوع نمودار در UML تلقی می شوند. نمودار کلاس شامل کلاس ها، interface ها (رابط)، association ها و collaboration می باشد.

نمودارهای کلاس در اصل دید یا نمایه ی شی گرایی (object oriented view) یک سیستم که به ذات static هست (ماهیت آن ایستا است) را نمایش می دهد.

Active class در یک نمودار کلاس جهت نمایش همروندی (concurrency) در سیستم مورد نظر بکار می رود.

نمودار کلاس ذات شی گرایی یک سیستم را نیز به تصویر می کشد. از این رو نمودار مزبور را معمولاً برای توسعه بکار می روند. این نمودار همچنین پرکاربرد ترین نوع در زمان ساخت سیستم محسوب می شود.

نمودار شی (object diagram)

نمودار شی را می توان نمونه ای از نمودار کلاس در نظر گرفت. بنابراین این نوع نمودارها شباهت بیشتری به سناریوهایی که در آن ها یک سیستم پیاده سازی می شود دارد.

نمودارهای شی، درست مانند نمودار کلاس، مجموعه اشیا و رابطه ی بین آن ها را به تصویر می کشد و علاوه بر آن نمای ایستا یا **static view** از سیستم مورد نظر ارائه (نشان) می دهد.

نمودار شی کاربردی مشابه نمودار کلاس دارد، با این تفاوت که نمودار شی برای ساخت نمونه ی اولیه (**prototype**) از یک سیستم از دیدگاه یا چشم انداز کاربردی مورد استفاده قرار می گیرد.

نمودار اجزا (component diagram)

نمودار های اجزا، همان طور که اسم آن اشاره دارد، مجموعه اجزای سیستم و رابطه ی بین آن ها را به تصویر می کشد. این اجزا خود متشکل از کلاس ها، **interface** ها (،)، **collaboration** ها می باشند.

نمودارهای اجزا یک دید اجرایی (**implementation view**) یا پیاده سازی از سیستم مد نظر را نمایش می دهند.

در طی مرحله ی طراحی، اجزای نرم افزاری (کلاس ها، رابط یا **interface** ها و غیره ..) یک سیستم، با توجه به رابطه ای که بین این اجزا وجود دارد، به گروه های مختلف تقسیم می شوند. حال این گروه ها تحت عنوان **component** در **UML** به ما شناسانده می شوند.

در نهایت باید گفت که مورد استفاده ی نمودارهای مزبور در نمایش تصویری اجرا و پیاده سازی سیستم خلاصه می شود.

نمودار استقرار و توزیع (Deployment Diagram)

نمودار استقرار را می توان یک مجموعه **node** (گره) و رابطه ی بین آن ها در نظر گرفت. این گره ها در واقع موجودیت های فیزیکی هستند که اجزا بر روی آن مستقر می شوند.

سخت افزار بکار رفته در پیاده سازی سیستم و همچنین محیط های اجرا و سایر اجزایی که باید بر روی این سخت افزار قرار گیرند را توصیف می کند.

با توجه به آنچه گفته شد، نمودارهای استقرار به منظور نمایش گرافیکی دید استقرار و بکارگیری یک سیستم استفاده می شود. این نمودار به طور معمول توسط تیم **deployment** مورد استفاده قرار می گیرد.

توجه: اگر به تمامی توضیحات و کاربردهای شرح داده شده توجه دقیق داشته باشید، پی می برید که همگی نمودارهای ذکر شده به نحوی با هم رابطه دارند. بدین صورت که نمودارهای اجزا وابسته به کلاس ها و رابط ها هستند که خود بخشی از نمودار کلاس/نشی می باشد و نمودار استقرار نیز به اجزایی وابسته است که در کنار هم نمودار اجزا (**component diagram**) را تشکیل می دهد.

نمودارهای رفتاری (Behavioral Diagrams)

هر سیستمی می تواند دو جنبه داشته باشد، **static** (ایستا) و **dynamic** (پویا). یک سیستم تنها زمانی کامل محسوب می شود که هر دو جنبه به طور کامل پوشش داده شود.

نمودارهای رفتاری تصویری از جنبه ی پویای (**dynamic aspect**) سیستم ارائه می دهد. اگر بخواهیم جنبه ی پویای یک سیستم را دقیق تر توضیح دهیم، باید بگیم که جنبه ی پویای سیستم همان بخش های در حال تغییر و حرکت سیستم می باشد.

به طور کلی، **UML** پنج نوع نمودار رفتاری ارائه می دهد که در زیر آن ها را مشاهده می کنید:

1. نمودار مورد کاربرد (**Use case diagram**)
2. نمودار توالی (**sequence diagram**)
3. نمودار همکاری (**collaboration diagram**)
4. نمودار حالت (**Statechart**)
5. نمودار فعالیت (**Activity diagram**)

Use case diagram

نمودارهای مورد کاربرد از **use case** ها، **actor** ها و رابطه ی آن ها تشکیل می شود. این دست نمودارها در واقع دید مورد کاربرد از یک سیستم را به تصویر می کشد. به عبارتی روشن تر؛ کارکرد ارائه شده توسط یک

سیستم را در قالب **actor** و اهداف آنها که به صورت مورد کاربرد نمایش داده می‌شوند و وابستگی بین موردهای کاربرد را مدل سازی می‌کند.

منظور از **use case**، یک قابلیت (**functionality** = عملیاتی که سیستم قادر به انجام آن هاست) خاص از سیستم می‌باشد.

با استناد به آنچه گفته شد، نمودار **use case** رابطه ی بین قابلیت ها و کنترلگرهای داخلی/خارجی آن ها را توصیف می‌کند. این کنترلگرها در **UML** تحت عنوان **actor** شناخته می‌شوند.

Sequence diagram (نمودار توالی)

نمودار توالی (به انگلیسی **sequence diagram**) زیرمجموعه ی **interaction diagram** قرار می‌گیرد. بع عبارتی روشن تر؛ نمودار توالی یکی از نمودارهای **Interaction** می‌باشد که روندی در یک **Use case** را مرحله به مرحله نشان می‌دهد. همان طور که می‌توان از اسم آن فهمید، این نمودار با توالی سروکار دارد که از دنباله یا رشته ی پست سرهم از پیام ها که از یک شی به شی دیگر سرازیر می‌شوند (جریان دارند)، تشکیل می‌شود.

تعامل بین اجزای یک سیستم، به خصوص از دیدگاه اجرایی و پیاده سازی، بسیار مهم می‌باشد.

از این رو می‌توان گفت که **sequence diagram** جهت نمایش دنباله ای از فراخوانی ها در یک سیستم در راستای آماده سازی شرایط لازم برای اجرای عملیات خاص بکار می‌رود.

Collaboration diagram (نمودار همکاری)

نمودار همکاری فرم دیگری از یک **interaction diagram** را ارائه می‌کند. نمودار ذکر شده صورت و ترتیب ساختاری سیستم، همچنین پیام های فرستاده/دریافت شده را نمایش می‌دهد. صورت ساختاری از **object** ها و **link** ها تشکیل می‌شود.

نمودار همکاری شباهت بسیاری به نمودار توالی دارد ، اصلی ترین تفاوت آنها در شمای ظاهری آنها می باشد. نمودار همکاری بیشتر بر روی رابطه بین اشیا تاکید دارد، این درحالی است که یک نمودار توالی اعمال اشیا را در یک توالی زمانی نشان می دهد و بر حسب زمان تنظیم و مرتب می شود.

در نمودار همکاری دید و نمای متفاوتی از روند عملیات **Use Case** ارائه می شود. در این نمودار مشاهده ارتباط بین اشیا بسیار سهل تر است .

Statechart diagram (نمودار حالت)

این نمودار همانطور که از نام آن مشخص است حالت های مختلفی که یک شی در آن قرار می گیرد را مدل سازی می نماید. به عبارتی دیگر این نمودار تصویری از چرخه حیات شی (Object life cycle) را به تصویر می کشد .

Activity Diagram (نمودار فعالیت)

نمودار فعالیت برای توصیف و شرح گام به گام گردش یا جریان کار تجاری و عملیاتی **component** های سیستم استفاده می شود. نمودار فعالیت توصیفگر گردش کنترل در سرتاسر سیستم می باشد. بنابراین این نمودار از **link** ها و **object** هایی تشکیل می شود. جریان می تواند همروند، ترتیبی یا **branched** و منشعب باشد. **Activity** ها چیزی به جز همان کارکردها و عملیاتی که سیستم انجام می دهد، نیستند.

همان طور که تشریح شد، نمودار فعالیت (**activity diagram**) جهت نمایش گرافیکی جریان کنترل ها در یک سیستم استفاده می شود.

UML State Machine Diagram (نمودار ماشین وضعیت)

این نمودار برای نمایش وضعیت های مختلف سیستم و انتقال بین وضعیت ها بکار می رود.

توجه: ماهیت پویای یک سیستم را به سختی می توان نمایش داد. به این خاطر **UML** ویژگی و امکاناتی عرضه داشته که به تصویر کشیدن پویاشناسی یک سیستم را از زوایای مختلف آسان ساخته است. دو نمودارهای توالی و همکاری هم ریخت یا **isomorphic** هستند و از این رو می توانند بدون اینکه اطلاعاتی را از دست بدهند به راحتی از یکی به دیگری تبدیل شوند. این امر همچنین درباره ی نمودار **statechart** و **activity** حکم می کند.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

نمودار کلاس (class diagram)

مدرس : مهندس افشین رفوآ

نمودار کلاس (class diagram)

نمودار کلاس یک نمودار **static** یا ایستا می باشد. این دیاگرام دید ایستا (**static view**) از یک برنامه را ارائه می دهد. تنها مورد استفاده ی نمودار کلاس در نمایش گرافیکی، توصیف و مستندسازی جنبه های مختلف یک سیستم خلاصه نمی شود، بلکه کاربرد دیگری دارد و آن هم ساخت کدهای قابل اجرا یک نرم افزار کاربردی می باشد.

نمودار کلاس خصیصه ها (**attribute**) و عملیات (**operation**) یک کلاس و همچنین محدودیت ها (**constraint**) اعمال شده بر روی سیستم را توصیف می کند. نمودارهای کلاس به طور گسترده به منظور مدل سازی سیستم های شی گرا بکار گرفته می شوند، زیرا نمودار کلاس تنها دیاگرامی است که می تواند توسط زبان های شی گرا نگاشت (**map**) شود.

نمودار کلاس مجموعه ای از کلاس ها، **interface** ها، **association** ها، **collaboration** ها و محدودیت ها را به نمایش می گذارد. نمودار کلاس زیرمجموعه ی نمودار ساختاری (**structural diagram**) می باشد.

کاربرد نمودار کلاس

هدف از بکار بردن نمودار کلاس مدل سازی دید ایستا (**static view**) از یک برنامه ی کاربردی می باشد. یادآور می شویم که نمودارهای کلاس تنها دیاگرام هایی هستند که توسط زبان های شی گرا قابل نگاشت بوده و بنابراین به طور گسترده در زمان ساخت مورد استفاده قرار می گیرند.

نمودارهای UML همچون نمودار فعالیت (activity diagram)، نمودار توالی (sequence diagram) تنها قادر به ارائه ی جریان توالی (sequence flow) برنامه ی کاربردی می باشند، اما نمودار کلاس کمی تفاوت دارد و بر اساس همین تفاوت پرکاربردترین نمودار UML در جامعه ی کدنویسان تلقی می شود.

هدف از بکاربردن نمودار کلاس در زیر به صورت خلاصه شرح داده شده:

1. تحلیل و طراحی دید static یک برنامه ی کاربردی.
2. شرح وظایف سیستم.
3. پایه ی نمودارهای component و deployment.
4. مهندسی معکوس (reverse engineering) و رو به جلو (forward engineering).

نحوه ی ترسیم نمودار کلاس

نمودارهای کلاس محبوب ترین نمودارهای UML هستند که برای ساخت نرم افزارهای کاربردی مورد استفاده قرار می گیرند. از این حیث یادگیری روال ترسیم نمودار کلاس از اهمیت بسیار بالایی برخوردار می باشد. نمودارهای کلاس خاصیت های متعددی دارند که حین ایجاد و ترسیم نمودار می بایست مورد توجه قرار داد، اما در اینجا نمودار از دید سطح بالا در نظر گرفته می شود.

نمودار کلاس در اصل یک نمایش گرافیکی از دید static سیستم مورد نظر بوده و جنبه های مختلف برنامه ی کاربردی مورد نظر را به تصویر می کشد. بنابراین مجموعه ای از کلاس ها یک سیستم کل را تشکیل می دهند.

لازم است به هنگام ترسیم یک نمودار کلاس، نکات زیر را یادآور شوید:

1. اسم نمودار کلاس بایستی معنی دار بوده و جنبه ی مورد نظر سیستم را توصیف کند.
2. تمامی المان ها می بایست به ضمیمه ی روابط آن ها از پیش شناسایی و مشخص شوند.
3. مسئولیت تمامی کلاس ها (متغیرهای عضو/خصیصه و متدهای آن کلاس) بایستی به روشنی مشخص و تعریف شود.

4. حداقل تعداد خاصیت ها یا متغیرهای عضو (**property**) ویژه ی هر کلاس بایستی مشخص شود. از خاصیت های غیر ضروری بایستی اجتناب کرد زیرا در صورت وجود خاصیت های غیر لازم نمودار بی دلیل پیچیده می شود.

5. هر جایی که فکر می کنید لازم است باید از **note** ها برای توصیف جنبه هایی از نمودار استفاده کنید تا بدین وسیله در انتهای فرایند ترسیم، نمودار برای توسعه دهنده/کد نویس قابل فهم باشد.

6. سرانجام، پیش از آماده سازی و ارائه ی نسخه ی نهایی، نمودار باید بر روی یک ورقه ی سفید ترسیم شده و هر تعداد دفعه که لازم بود آن را تکرار کنید تا خروجی صحیح درآید.

نمودار زیر مثالی از **order system** یک برنامه ی کاربردی است؛ تنها یک جنبه ی خاص از کل یک برنامه را توصیف می کند.

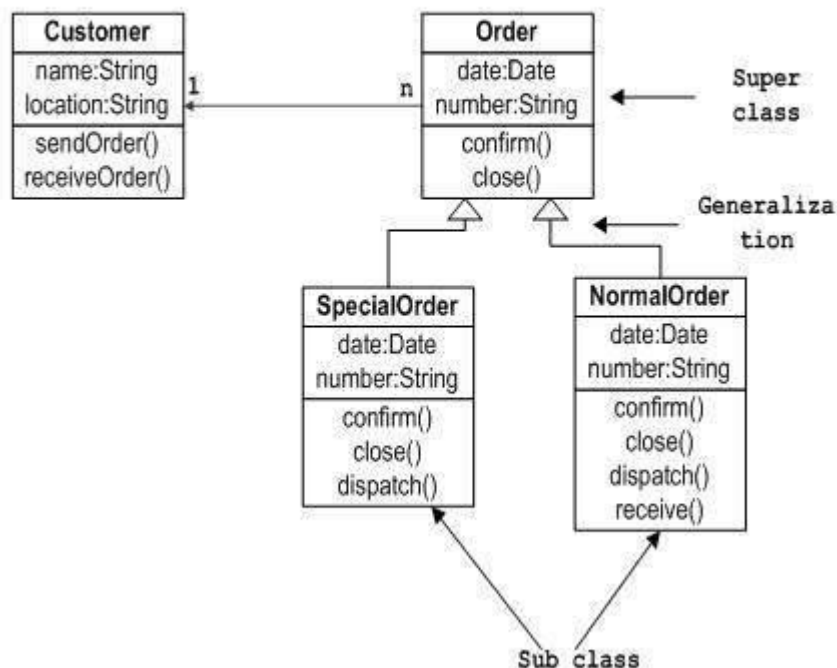
1. پیش از هر چیز، **Order** و **Customer** به عنوان دو المان اصلی سیستم مورد نظر شناسایی شده و این دو در یک رابطه ی یک به چند مشارکت دارند زیرا یک **customer** (مشتری) می تواند چندین **order** (سفارش) داشته باشد.

2. کلاس **Order** یک کلاس انتزاعی می باشد و دارای دو کلاس واقعی (از طریق رابطه ی وراثت) **NormalOrder** و **SpecialOrder** می باشد.

3. دو کلاس مشتق شده تمامی **property** های کلاس **Order** را به ارث برده اند. بعلاوه دارای توابع اضافی نظیر **dispatch()** و **receive()** هستند.

بنابراین نمودار کلاس با توجه به تمامی نکات ذکر شده در بالا ترسیم شده:

Sample Class Diagram



کجاها از نمودارهای کلاس استفاده می شود؟

باتوجه به آنچه قبلا شرح داده شد، نمودار کلاس یک دیاگرام **static** محسوب می شود و کاربرد آن در مدل سازی دید **static** یک سیستم می باشد. دید ایستا یا **static view** درواقع با استفاده از رابطه ها، اشیا، خصیصه ها و عملیات بر روی ساختار ایستا یک سیستم متمرکز شده و آن را تاکید می کند؛ بایستی گفت که دید ایستا **vocabulary** سیستم را توصیف می کند.

نمودار کلاس همچنین پایه و فونداسیون نمودارهای اجزا (**component**) و استقرار (**deployment**) محسوب می شود. نمودارهای کلاس تنها در نمایش گرافیکی دید ایستای سیستم کاربرد ندارد، بلکه این نوع نمودار در ایجاد کدهای قابل اجرا برای مهندسی معکوس و روبه جلو هر سیستمی مورد استفاده قرار می گیرد.

به طور معمول نمودارهای **UML** را نمی توان مستقیما توسط زبان های شی گرا نگاشت نمود، اما در این میان نمودار کلاس یک استثنا تلقی می شود.

نمودار کلاس نگاشت را توسط زبان های شی گرا همچون **Java**، **C++** و غیره .. پیاده سازی کرده و نمایش می دهد، از این رو با توجه تجربه ی عملی کسب شده می توان گفت که نمودار کلاس به طور معمول برای ساخت برنامه های کاربردی به وسیله ی زبان های شی گرا مورد استفاده قرار می گیرد.

موارد استفاده ی نمودار کلاس را می توان به صورت خلاصه نام برد:

1. توصیف دید ایستای سیستم.
2. نمایش همکاری بین المان های دید ایستا (ساختار ایستای سیستم را مورد تاکید قرار داده و نمودارهای ساختار ترکیبی، کلاس تحت پوشش آن قرار می گیرند).
3. شرح عملیاتی (**functionality**) که توسط سیستم قابل اجرا می باشد.
4. ساخت نرم افزارهای کاربردی توسط زبان های شی گرا.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

نمودار اشیا

مدرس : مهندس افشین رفوآ

نمودار اشیا (object diagram)

نمودارهای شی خود از نمودارهای کلاس مشتق می شوند. پس نتیجه می گیریم که نمودارهای شی به نحوی وابسته به نمودارهای کلاس هستند.

نمودارهای شی در واقع نمونه ای از یک نمودار کلاس ارائه می کند. مفاهیم پایه ای بین دو نمودار ذکر شده، مشترک می باشد. نمودارهای شی همچنین دید ایستایی (static view) از یک سیستم فراهم می نماید، اما این دید ایستا یک تصویر یا نسخه ی فوری از سیستم مورد نظر در یک برهه ی زمانی خاص ارائه می دهد.

نمودار شی جهت نمایش مجموعه ای از اشیا و رابطه بین آن ها به عنوان یک نمونه بکار می رود.

جهت پیاده سازی عملی یک نمودار، ابتدا می بایست مورد کاربرد آن را درک کرد. مورد استفاده (هدف و کاربرد) نمودارهای شی به گونه ای مشابه نمودارهای کلاس می باشد.

تفاوت بین دو نمودار مزبور این است که دیاگرام کلاس یک نمودار انتزاعی (abstract diagram) متشکل از کلاس ها و رابطه ی بین آن ها ارائه می نماید، در حالی که نمودار شی یک نمونه از کلاس مورد نظر در برهه ی زمانی خاص که دارای ماهیت واقعی (concrete nature) هست فراهم می کند، بدین معنا که نمودار شی شباهت بیشتری به رفتار واقعی سیستم دارد. در اینجا مقصود به تصویر کشیدن دید ایستا از یک سیستم در یک برهه یا زمان خاص می باشد.

بنابراین موارد استفاده و مقصود از بکاربردن نمودار شی را می توان به ترتیب زیر خلاصه بیان نمود:

1. جهت مهندسی معکوس (reverse engineering) و رو به جلو (forward engineering).

2. روابط بین اشیا در یک سیستم.

3. دید ایستا از تعامل یا **interaction**.

4. فهم رفتار شی و رابطه ی آن ها از یک چشم انداز یا دیدگاه کاربردی.

نحوه ی ترسیم نمودار شی (object diagram)

یادآور می شویم که نمودار شی در واقع نمونه ای از نمودار کلاس ارائه می دهد، بدین معنا که نمودار شی از نمونه ای از اشیا بکار گرفته شده در یک نمودار کلاس تشکیل می شود.

از این رو هر دو نمودار از المان های پایه ای یکسان تشکیل می شوند ولی در شکل های متفاوت (یکی در قالب انتزاعی و دیگری به صورت واقعی). در نمودار کلاس کلیه ی عناصر به منظور نمایش طرح کلی (**blue print**) در قالب انتزاعی هستند، در حالی که همین المان ها در نمودار شی دارای ماهیت **concrete** و واقعی بوده و اشیا حقیقی را به تصویر می کشد.

برای نمایش یک سیستم خاص، اغلب تعداد نمودارهای کلاس محدود می باشد. اما چنانچه نمودارهای شی را در نظر بگیریم، در آن صورت قادر خواهیم بود تعداد نامحدودی نمونه که به ذات منحصر بفرد هستند، داشته باشیم. بنابراین تنها آن نمونه هایی در نظر گرفته می شوند که بر روی سیستم تاثیرگذار هستند.

با استناد به توضیحات فوق، می توان نتیجه گرفت که یک نمودار شی مجرد نمی تواند تمامی نمونه های مورد نیاز را نمایش دهد یا به عبارتی دقیق تر قادر نخواهد بود کلیه ی اشیا سیستم را مشخص کند. راه حل آن به صورت زیر می باشد:

1. در ابتدای امر بایستی سیستم را مورد بررسی و تحلیل قرار داده، نتیجه گرفت و مشخص کرد که کدام نمونه ها دارای داده ها، اطلاعات و **association** (رابطه ی انجمنی و تناظر) می باشد.

2. در مرحله دوم می بایست تنها آن نمونه هایی را در نظر گرفت که عملیات و قابلیت ها (**functionality**) را تحت پوشش قرار می دهد.

3. در نهایت، از آنجایی که تعداد نمونه ها نامحدود می باشد، توصیه می کنیم بهینه سازی را پیاده سازی نماییم.

قبل از ترسیم نمودارهای شی، لازم است نکات زیر را فهمیده و بخاطر داشته باشید:

1. نمودارهای شی از اشیا تشکیل می شوند.

2. پیوند یا لینک را در نمودار شی جهت متصل کردن (ربط دادن) اشیا بکار می بریم.

3. اشیا و پیوندها دو المانی هستند که در کنارهم نمودار شی را می سازد.

پس از پرداختن به این مقوله، باید موارد زیر را پیش از اقدام به ترسیم نمودار مورد توجه قرار داد:

1. نمودار شی می بایست دارای اسم معنی دار باشد که مقصود یا هدف اصلی دیاگرام مورد نظر را به صورت صریح بیان کند.

2. بایستی تمامی المان های مهم آن شناسایی شوند.

3. رابطه انجمنی (**association**) و تناظر میان اشیا باید روشن شود.

4. باید مقادیر المان های مختلف نمایش داده شده و در نمودار شی لحاظ (**include**) شود.

5. در جاهایی که احساس می کنید، توضیح بیشتر نیاز است یادداشت و نکات (**note**) لازم را درج نمایید.

نمودار زیر نمونه ای از یک دیاگرام شی می باشد. نمودار حاضر همان **Order management system**

(سامانه ی مدیریت سفارش) را نمایش می دهد که در مبحث قبلی (نمودار کلاس) مورد بررسی قرار دادیم.

این نمودار نمونه ای از سیستم مورد نیاز را در زمان خرید ارائه می دهد. نمودار مزبور شامل اشیا زیر می باشد:

1. **Customer**

2. **order**

3. **SpecialOrder**

4. **NormalOrder**

شی **customer** با سه آبجکت دیگر دارای رابطه ی انجمنی (**O1**، **O2** و **O3**) می باشد (با آن ها مرتبط هست).

این آبجکت ها با دو شی **SpecialOrder** و **NormalOrder** (**S1**، **S2** و **N1**) دارای رابطه ی تناظر (متصل)

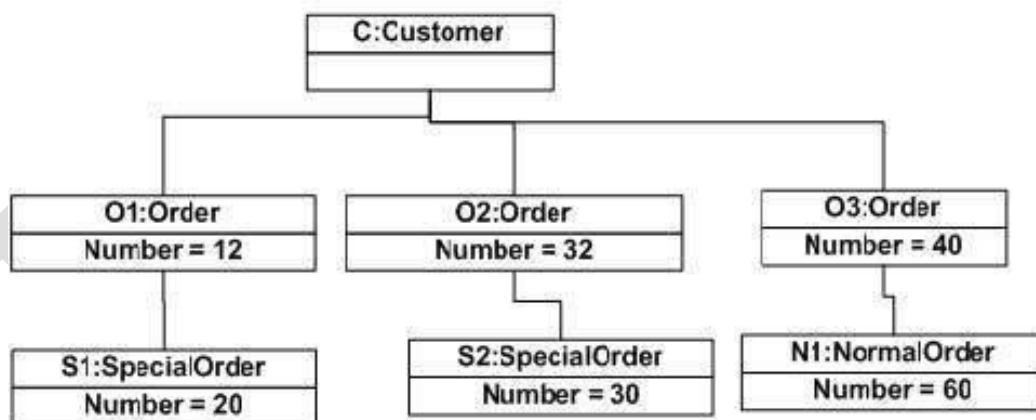
هستند. **Customer** با سه عدد مختلف (12، 32 و 40) در زمان مشخص و در نظر گرفته شده، سفارش می دهد.

Customer ممکن است در آینده تعداد **order** ها (سفارشات خود) را افزایش دهد که در آن صورت نمودار این قابلیت را دارد که تغییرات اعمال شده را نمایش دهد. حال اگر به سه شی **order**، **special order** و **normal order** نگاه کنید، متوجه می شوید که اشیا نام برده دارای مقادیری مختص به خود هستند.

مقادیر اشیا **order** به ترتیب 12، 32 و 40 می باشد. این امر بیانگر موقتی بودن مقادیر اشیا (اینکه این مقادیر متعلق به زمانی خاصی هستند؛ زمانی که خرید یا **purchase** صورت می گیرد) زمانی که نمونه به تصویر کشیده می شود، می باشد.

همین امر درباره ی اشیا **special order** و **normal order** حکم می کند که مقادیر آن ها به ترتیب 20، 30 و 60 می باشد. در صورت تغییر زمان خرید، این مقادیر نیز تغییر می کنند. نمودار زیر با در نظر گرفتن تمام نکات یاد شده رسم گردیده است:

Object diagram of an order management system



چه زمانی از نمودار شی استفاده می کنیم؟

نمودار شی را می توان یک عکس لحظه ای از سیستم در حال کار (فعال) در زمان مشخص تصور کرد. حال به منظور روشن سازی مفهوم آن می توان یک قطار در حال حرکت را مثال زد.

اگر عکسی از قطار در حال حرکت تهیه نمایید، یک تصویر ایستا از آن قطار دریافت می کنید که موارد زیر را شامل می شود:

1. وضعیت آن قطار را در حال حرکت می بینید.
2. تعدادی مسافر که بر آن سوار هستند و در صورت گرفتن عکس در زمان های مختلف تعداد آن ها تغییر می کند.

بنابراین می توان این عکس لحظه ای از قطار را یک شی در نظر گرفت که موارد ذکر شده در بالا، مقادیر آن هستند. این قضیه درباره ی کلیه ی سیستم های ساده و پیچیده ی حقیقی صادق است.

در زیر موارد کاربرد نمودار شی را مشاهده می کنید:

1. تهیه ی نمونه ی اولیه (ایجاد **prototype**) از سیستم مورد نظر.

2. مهندسی معکوس.

3. مدل سازی ساختارهای داده ای پیچیده.

4. فهم و درک عمیق سیستم از دیدگاه کاربردی.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

نمودارهای اجزا

مدرس : مهندس افشین رفوآ

نمودارهای اجزا (component diagram)

نمودارهای اجزا هم از نظر ماهیت، هم از نظر رفتار متفاوت هستند. این دست نمودارها اغلب برای مدل سازی جنبه های فیزیکی یک سیستم بکار می روند.

حال این سوال مطرح می شود که جنبه های فیزیکی که از آن ها بحث شد، چی هستند؟ جنبه های فیزیکی همان فایل های اجرایی، کتابخانه ها، فایل ها و سند های (document) یک سیستم هستند که در یک گره (node) قرار می گیرند.

از این رو می توان گفت که نمودارهای اجزا برای نمایش گرافیکی سازمان دهی، ترتیب و رابطه ی میان اجزا و مولفه های سیستم مورد بهره وری قرار می گیرد. این دست نمودارها برای ایجاد سیستم های قابل اجرا نیز مورد استفاده می گیرند. به عبارتی این نمودارها چگونگی تقسیم سیستم به مولفه های آن و وابستگی بین مولفه های سیستم را توصیف می کند.

مورد استفاده ی نمودار component

همان طور که قبلاً توضیح داده شد، نمودار اجزا یکی از دیاگرام های موجود در زبان مدل سازی UML محسوب می شود. هدف و مورد استفاده از آن نیز با تمامی نمودارهای نام برده متفاوت می باشد. دیاگرام component درباره ی عملیات قابل اجرا توسط سیستم و قابلیت های آن شرح نمی دهد، بلکه وظیفه ی آن توصیف مولفه هایی است که در کنار هم آن قابلیت ها را می سازند.

بنابراین نمودارهای جز برای نمایش گرافیکی مولفه های فیزیکی یک سیستم بکار می رود. این مولفه ها عبارتند از کتابخانه ها، پکیج ها، فایل ها و غیره ...

می توان تعریف دیگری از نمودار اجزا ارائه نمود: نموداری که پیاده سازی ایستا از سیستم را به تصویر می کشد. پیاده سازی ایستا نمایشگر ترتیب و سازمان دهی مولفه های سیستم در یک زمان مشخص می باشد.

یک نمودار **component** مجرد قادر به نمایش کل (تمامی اجزای) یک سیستم نیست، از این رو برای نشان دادن کل سیستم بایستی از یک مجموعه متشکل از چندین دیاگرام بهره گرفت.

بنابراین موارد استفاده از نمودار نام برده را می توان به صورت زیر خلاصه کرد:

1. نمایش گرافیکی اجزا و مولفه های سیستم.

2. ساخت فایل های اجرایی با بهره گیری از مهندسی روبه جلو و معکوس.

3. توصیف ترتیب (سازمان دهی) و روابط میان اجزای سیستم.

نحوه ی ترسیم نمودار اجزا

یادآور می شویم که نمودار **component** اجزای فیزیکی یک سیستم را توصیف می کند. این اجزا شامل فایل ها، فایل های اجرایی (**executable** ها)، کتابخانه ها و غیره ... می باشند.

از این رو هدفی که با رسم نمودار دنبال می شود، با هدفی که دیگر دیاگرام ها برای نیل به آن ایجاد می شوند کاملا متفاوت است. نمودار **component** در در مرحله ی پیاده سازی برنامه ی کاربردی استفاده می شوند.

لازم به ذکر است که نمودار اجزا بایستی با فاصله ی زمانی زیاد قبل از این مرحله رسم می شود تا از این طریق جزئیات پیاده سازی به درستی در نظر گرفته و به صورت گرافیکی نمایش داده شود.

در ابتدا سیستم با بکارگیری نمودارهای مختلف **UML** طراحی می شود، سپس با آماده شدن تمامی اجزا، دیاگرام **component** جهت ارائه ی گرافیکی جزئیات پیاده سازی سیستم ترسیم می شود.

رسم این نمودار از اهمیت ویژه ای برخوردار است، زیرا بدون آن امکان پیاده سازی موثر اپلیکیشن وجود ندارد. یک نمودار اجزا که به صورت کارآمد ترسیم شده باشد نیز برای دیگر جنبه ها همچون کارایی برنامه و

نگهداشت آن بسیار حائز اهمیت می باشد.

بنابراین قبل از اقدام به ترسیم نمودار می بایست، مولفه های زیر را به طور صریح و روشن شناسایی نمایید:

1. فایل های مورد استفاده در سیستم.

2. کتابخانه ها و دیگر اجزای مربوط به نرم افزار کاربردی مورد نظر.

3. رابطه ی بین اجزای سیستم.

پس از شناسایی اجزای سیستم، توصیه می کنیم نکات زیر را رعایت نمایید:

1. یک اسم معنی دار برای مولفه ای که قرار است نمودار برای آن رسم می شود، انتخاب نمایید.

2. پیش از رسم نمودار، با استفاده از ابزار موجود، یک طرح ذهنی از آن نمودار آماده سازید.

3. هر جایی که لازم دیدید با استفاده از یادداشت گذاری (**note** ها) نکاتی را جهت تشریح نمودار درج نمایید.

در زیر یک نمودار اجزا مشاهده می کنید که برای سیستم مدیریت سفارش ترسیم شده. در اینجا اجزای تشکیل

دهنده سیستم، فایل ها هستند. نمودار مربوطه فایل های برنامه و رابطه ی میان آن ها را نمایش می دهد.

لازم به ذکر است که نمودارهای **component** اغلب دربردارنده ی فایل های **dll**، کتابخانه ها، پوشه ها و غیره

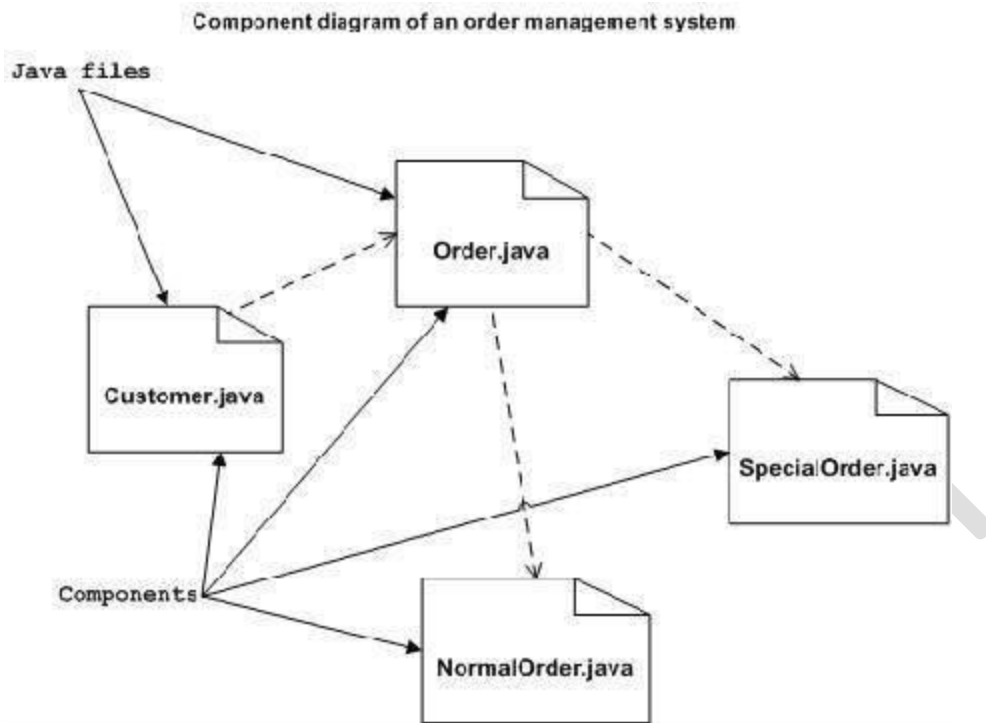
... می باشد.

همان طور که مشاهده می کنید، در دیگرام حاضر چهار فایل مختلف شناسایی شده و رابطه میان آن ها تشریح

گردیده است. نمودار اجزا را نمی توان با دیگر دیگرام های **UML** که تاکنون نام بردیم مقایسه کرده یا برابر

دانست. این نمودار از لحاظ موارد استفاده با دیگر نمودارها کاملا تفاوت دارد.

نمودار یاد شده با رعایت کلیه ی نکات مزبور ترسیم گردیده است:



کجا از نمودار اجزا استفاده می شود؟

قبلا هم گفتیم که نمودارهای **component** به منظور نمایش گرافیکی پیاده سازی دید یا **view** (مفاهیم یو ام ال در قالب رده یا دسته‌هایی به نام دید (**view**) طبقه‌بندی می‌شوند. هر "دید" در واقع زیرمجموعه‌ای از ساخت‌های مدلسازی است که یک جنبه از سیستم را نمایش می‌دهند) ایستا یک سیستم بکار می‌رود. این دید مفاهیم مربوط به حوزه برنامه کاربردی و مفاهیم داخلی ابداع شده به عنوان بخشی از پیاده سازی برنامه کاربردی را مدل سازی می‌کند. این **view**، به این خاطر ایستا نامیده می‌شود زیرا رفتارهای وابسته به زمان سیستم را توصیف نمی‌کند. اجزای تشکیل دهنده دید ایستا عبارتند از کلاسها و روابط (ارتباط و تعمیم) و وابستگی‌های (مانند **realization** و **usage**) بین آنها. دید ایستا در قالب نمودارهای کلاس نمایش داده می‌شود.

نمودار اجزا نوع خاصی از دیاگرام های **UML** می باشند که برای نیل به اهداف مختلف مورد استفاده قرار می گیرد.

این دیاگرام ها نمایی از اجزای فیزیکی یک سیستم ارائه می دهند؛ به عبارتی روشن تر نمودارهای اجزا ترتیب و سازمان دهی (organization) مولفه ها در یک سیستم را توصیف می کند.

Organization را می توان مکان قرارگیری اجزا در یک سیستم تصریح نمود. این اجزا به منظور رفع نیازهای سیستم به گونه ای خاص سازمان دهی می شوند.

قبلا هم توضیح دادیم که آن اجزای تشکیل دهنده کتابخانه، فایل ها، فایل های اجرایی (executable) هستند. پیش از پیاده سازی برنامه می بایست این مولفه ها را سازمان دهی کرد. سازمان دهی اجزا به عنوان بخشی از پروسه ی اجرای پروژه، به صورت مجزا طراحی می شوند. نمودارهای اجزا، به خصوص از دیدگاه پیاده سازی، بسیار مهم می باشند. از این رو تیم پیاده سازی برنامه باید دانش و اطلاعات کافی درباره ی جزئیات مولفه داشته باشند.

موارد کاربرد دیاگرام اجزا یا **component** را می توان بدین ترتیب فهرست کرد:

1. مدل سازی اجزا سیستم

2. مدل سازی شمای پایگاه داده (database schema).

3. مدل کردن فایل های اجرای یک برنامه یا نرم افزار کاربردی.

4. مدل سازی کد منبع (source code) سیستم.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

Deployment Diagram

مدرس : مهندس افشین رفوآ

دوره آموزش [UML](#)

نمودار استقرار (Deployment Diagram)

نمودار استقرار یا توزیع و یا بکارگیری در **uml** ، معماری یک سیستم متکی به رایانه را به صورت فیزیکی نمایش می دهد.

این نمودار قادر به نمایش رایانه و دستگاه های مربوط به آن و نیز ارتباطاتی که این دستگاه ها با هم دارند و نیز نرم افزاری که روی هر ماشین قرار دارد، می باشد. هر رایانه توسط یک مکعب نمایش داده می شود و ارتباط آن نیز با رایانه های دیگر توسط خطوط ارتباطی ارائه می گردند. تعریف دیگری که می توان ارائه نمود بدین صورت است: سخت افزار بکار رفته در پیاده سازی سیستم و همچنین محیط های اجرا و سایر مولفه هایی که بایستی بر روی این سخت افزار مستقر شوند را شرح می دهد. بنابراین نمودار استقرار جهت توصیف دید ایستای استقرار یک سیستم بکار می رود. نمودارهای استقرار از گره ها (**node**) و رابطه ی بین آن ها تشکیل می شود.

اهداف استفاده از نمودار استقرار

خود اسم **Deployment** هدف و مورد استفاده ی نمودار را شرح می دهد. نمودارهای استقرار برای توصیف اجزای سخت افزاری که اجزای نرم افزاری بر روی آن قرار می گیرد، بکار می رود. نمودارهای اجزا و استقرار بسیار به هم نزدیک (با هم مرتبط) هستند.

نمودارهای اجزا جهت شرح مولفه ها مورد استفاده قرار گرفته و نمودارهای استقرار به منظور نمایش نحوه ی قرارگیری (توزیع و مستقر شدن) آن اجزا و مولفه ها بر روی سخت افزار بکار می رود.

UML در اصل ویژه ی تمرکز بر روی مصنوعات و اجزای یک سیستم بکار می رود. اما این دو نمودار، دیاگرام های ویژه ای هستند که تمرکز آن ها بر روی اجزای نرم افزاری و سخت افزاری می باشد.

از این رو می توان گفت که بیشتر نمودارهای **UML** به منظور مدیریت اجزا یا مولفه های منطقی مورد بهره وری قرار می گیرند، اما نمودار استقرار به صورت اختصاصی برای تمرکز بر روی توپولوژی (جانمایی) سخت افزاری یک سیستم بکار گرفته می شود. این دست نمودارها اساسا برای مهندسين سیستم تعبيه و طراحی شده است و اغلب توسط آن ها مورد استفاده قرار می گیرد.

مقصود از بکارگیری نمودارهای استقرار را می توان به صورت زیر شرح داد:

1. نمایش گرافیکی توپولوژی و جانمایی سخت افزاری سیستم.
2. توصیف اجزا و مصنوعات سخت افزاری یک سیستم که اجزا نرم افزاری بر روی آن قرار می گیرد.
3. توصیف گره های پردازش زمان اجرا (**runtime processing node**).

نحوه ی ترسیم نمودار استقرار

نمودار **deployment** دید استقرار از یک سیستم را نمایش می دهد. این دیاگرام با نمودار اجزا (**component**) مرتبط می باشد. از آنجایی که اجزا (**component**) توسط نمودارهای **deployment** بر روی سخت افزار پیاده و توزیع می شوند، نمودار استقرار از گره ها تشکیل می شود. گره ها همان اجزا سخت افزاری فیزیکی هستند که برای نصب و استقرار برنامه ی کاربردی مورد استفاده قرار می گیرد. نمودارهای استقرار برای مهندسين نرم افزار بسیار پرکاربرد می باشد. یک نمودار کارآمد استقرار، از آنجایی که پارامترهای زیر را تحت کنترل دارد، از اهمیت بالایی برخوردار است:

1. کارایی
2. مقیاس پذیری
3. قابلیت نگهداشت
4. قابلیت حمل و نصب آسان

پیش از اقدام به ترسیم نمودار استقرار می بایست مصنوعات و مولفه های زیر را شناسایی نمود:

1. گره ها (node)

2. رابطه ی میان گره ها

دیگرام نمونه ی زیر یک نمای استقرار از سیستم مدیریت سفارش (order management system) ارائه می نماید. گره های این سیستم به ترتیب زیر می باشند:

1. نمایشگر

2. مدم

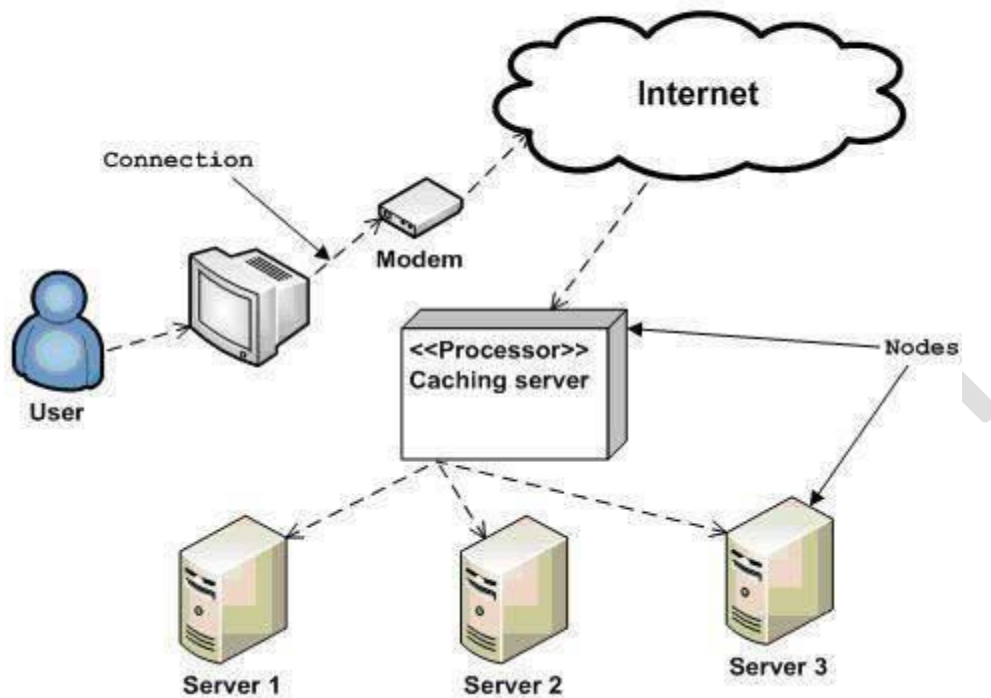
3. سرویس دهنده ی موقت (caching server)

4. سرویس دهنده

نرم افزار مورد نظر یک برنامه ی کاربردی تحت وب می باشد که بر روی محیط خوشه ای (clustered environment) که سرویس دهنده ی server1، server2 و server3 در آن شرکت دارند، مستقر می شود. کاربر از طریق اینترنت به برنامه ی مورد نظر متصل می شود. کنترل از caching server به محیط خوشه ای جریان دارد.

حال نمودار استقرار با رعایت نکات فوق بدین شکل ترسیم می شود:

Deployment diagram of an order management system



کجا از نمودار استقرار استفاده می شود؟

نمودارهای استقرار عمدتاً توسط مهندسين سيستم بكار مي رود. اين نمودارها جهت توصيف اجزا و مصنوعات سخت افزاري سيستم و نيز توزيع و رابطه ي انجمنی میان آن ها مورد استفاده قرار مي گيرد.

اگر بخواهيم آن را دقيق تر توضيح بدهيم، بايد بگوييم كه نمودارهای استقرار را مي توان به صورت اجزا/گره های سخت افزاري كه مولفه های نرم افزاري بر روی آن ها مستقر مي شوند تصور کرده، سپس به تصوير کشيد.

نرم افزارهای کاربردی به منظور مدل سازی فرایندهای تجاری (business process) پیچیده تولید و توسعه داده می شوند. اما نرم افزارهای کاربردی به تنهایی قار به رفع و برآورده ساختن نیازهای تجاری نیستند.

نیازهای تجاری را می توان در قابلیت پشتیبانی از تعداد روز افزون کاربران و زمان پاسخ دهی سریع و غیره ... خلاصه نمود.

جهت رفع نیازهایی از این دست، مصنوعات سخت افزاري می بایست به صورت کارآمد و کم هزینه طراحی شده باشند.

امروزه نرم افزارهای کاربردی به ذاته بسیار پیچیده هستند. نرم افزارهای کاربردی می توانند مستقل، مبتنی بر وب، توزیع شده، مبتنی بر **mainframe** (بزرگ رایانه) باشند، به این خاطر طراحی موثر و کارآمد اجزا سخت افزاری بسیار مهم می باشد.

موارد استفاده ی نمودارهای استقرار بدین ترتیب قابل شرح می باشد:

1. مدل سازی جانمایی و توپولوژی سخت افزاری یک سیستم.
2. مدل سازی سیستم های نهفته (**embedded system**) = در واقع رایانه هایی هستند که برای کنترل یک سیستم بزرگ و مشخص طراحی شده اند و مخصوصا در زمان هایی که محدودیت هایی در مورد پردازش همزمان وجود دارد به کار می روند).
3. مدل سازی جزئیات سخت افزاری برای سیستم های سرویس گیرنده/سرویس دهنده.
4. مدل سازی جزئیات سخت افزاری برنامه های توزیع شده.
5. مهندسی معکوس و رو به جلو.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

Use Case Diagrams

مدرس : مهندس افشین رفوآ

دوره آموزش [UML](#)

نمودار مورد کاربرد (Use Case Diagrams)

در مدل سازی سیستم آنچه بسیار مهم است، نمایش رفتار پویای آن می باشد. رفتار پویا یا به انگلیسی **dynamic behavior** بیانگر رفتار و عملکرد سیستم زمانی که آن سیستم در حال اجرا (عملیات) می باشد، است. نمودارهای رفتاری بر آنچه بایستی در سیستم مدل سازی شده اتفاق بیافتد تاکید دارد.

رفتار ایستا (**static behavior**) به تنهایی قادر به مدل سازی سیستم نیست بلکه برای این منظور می بایست رفتار پویا را نیز لحاظ نمود. حتی می توان گفت که رفتار پویا دارای اهمیت بیشتری می باشد. در زبان مدل سازی یکپارچه ی **UML**، به طور کلی پنج دیاگرام برای مدل سازی ماهیت پویای سیستم در دست داریم که نمودار **Use case** یکی از آن ها می باشد. حال از آنجایی که نمودار مورد کاربرد (**use case**) ذاتا پویا می باشد، باید تعدادی عامل خارجی و داخلی برای برقراری تعامل و برهمکنش دخیل باشند.

این **agent** های داخلی و خارجی (عوامل) تحت عنوان عملگر یا **actor** معرفی می شوند. نمودارهای مورد کاربرد از عملگرها، موارد کاربرد (**use case**) و رابطه ی میان آن ها تشکیل می شود. نمودار نام برده جهت مدل سازی سیستم ها و **subsystem** های برنامه ی کاربردی استفاده می شود. یک نمودار **use case** مجرد تنها قادر به نمایش گذاشتن یک قابلیت یا عملیات قابل اجرای سیستم می باشد.

از این رو به منظور ارائه ی تصویری جامع از کل سیستم، ملزوم به استفاده از چندین نمودار **use case**

هستیم.

همان طور که تصریح شد، نمودار مورد کاربرد کارکرد ارائه شده توسط یک سیستم را در قالب عملگرها (Actor) و اهداف آنها که به صورت مورد کاربرد نشان داده می‌شوند و نیز وابستگی بین موردهای کاربرد را مدل می‌نماید.

مقصود از بکاربردن نمودار use case نمایش گرافیکی جنبه های پویای یک سیستم می باشد. اما این تعریف بیش از حد عمومی بوده و هدف این نمودار را به طور دقیق تشریح نمی کند. چهار نمودار دیگری که در UML مورد استفاده قرار می گیرند (statechart، collaboration، sequence و activity) نیز همین کاربرد را دارند.

به این خاطر هدف دقیق تری که برای نیل به آن طراحی شده را مورد بررسی و پژوهش قرار می دهیم تا دلیل تفاوت آن از دیگر دیاگرام ها مشخص گردد.

نمودارهای use case به منظور شناسایی نیازهای که قرار است توسط سیستم مورد نظر برآورده شود و همچنین موارد استفاده ی آن، مورد استفاده قرار می گیرد. این نیازها غالباً مربوط به طراحی هستند. بنابراین زمانی که سیستمی برای شناسایی قابلیت ها و عملیات قبل اجرای آن مورد تجزیه و تحلیل قرار می گیرد، دیاگرام های use case آماده سازی شده و عملگرهای (actor) آن شناسایی می شوند.

پس از اینکه گام اول تکمیل شد، نمودار use case برای ارائه ی نمای خارجی مدل سازی می شود.

اکنون می توان موارد استفاده ی نمودارهای use case را به طور خلاصه تشریح نمود:

1. شناسایی موارد کاربرد سیستم و نیازهایی که قرار است برآورده سازد.
2. ارائه ی نمای خارجی از سیستم.
3. شناسایی عوامل داخلی و خارجی که سیستم را تحت تاثیر قرار می دهند.
4. نمایش تعامل و interaction مابین موارد کاربرد در قالب عملگرها (کارکرد ارائه شده توسط یک سیستم را در قالب بازیگران (Actor) و اهداف آنها که به صورت مورد کاربرد نمایش داده می‌شوند و وابستگی بین موردهای کاربرد، مدلسازی می‌کند.)

نمودار **use case** نمایی از عملگرها که در تعامل با سیستم نقش هایی را ایفا می کنند، ارائه می دهد.

نمودارهای **use case** برای تجزیه و تحلیل سطح بالای موارد استفاده ی سیستم بکار می روند. بنابراین هنگامی که نیازهای سیستم تجزیه و تحلیل می شوند، تمامی قابلیت ها در نمودارهای **use case** نمایش داده می شوند. با توجه به آنچه گفته شد، **use case** ها چیزی به جز قابلیت ها و کارکردهای سیستم نیستند که به صورت سازمان یافته رسم می شوند. دیگر جز متعلق به نمودار **use case**، عملگر یا **actor** ها هستند. **actor** را می توان عملگرهایی نامید که با سیستم تعامل برقرار می کند.

این **actor** ها می توانند یک کاربر انسان، برنامه های داخلی یا خارجی باشند.

پس پیش از اقدام به ترسیم نمودار **use case**، لازم است آیتم های زیر را شناسایی نمود:

1. قابلیت ها و کارکردها که به صورت **use case** نمایش داده شوند.

2. **actor** (عملگرها)

3. رابطه ی میان **use case** ها و **actor** ها

نمودار **use case** جهت نمایش موارد استفاده و کارکردهای عملی سیستم مورد استفاده قرار می گیرد. بعد از شناسایی آیتم های بالا، رهنمودها و دستورالعمل های زیر را برای ترسیم نمودار کارآمد **use case** رعایت نمایید:

1. اسم **use case** یا مورد کاربرد از اهمیت ویژه ای برخوردار است. اسم بایستی طوری انتخاب شود که عملیات قابل اجرا توسط سیستم را بیان نماید.

2. اسم مناسبی برای **actor** ها انتخاب نمایید.

3. رابطه ها و همچنین رابطه های وابستگی (**dependency**) می بایست به صراحت در نمودار نمایش داده شوند.

4. لزومی ندارد تمامی رابطه ها را در نمودار نمایش دهید. آنچه از بالاترین اهمیت برخوردار است، به تصویر کشیدن کارکردهای سیستم می باشد.

5. هر جایی که لازم بود از یادداشت ها و **note** ها برای توضیح بخش های نمودار استفاده کنید.

در زیر نمونه ای از نمودار **use case** را مشاهده می کنید که تصویری از سیستم مدیریت سفارش ارائه می نماید. با مشاهده ی نمودار سه کارکرد سیستم مورد نظر (**Order**، **SpecialOrder** و **NormalOrder**) و یک عملگر یا **actor** که **customer** یا همان مشتری می باشد.

دو **use case** (مورد کاربرد) **NormalOrder** و **SpecialOrder** از **use case** ای به نام **Order** منشعب می شوند. نتیجتاً رابطه ی بین آن ها از نوع **extend** می باشد. نکته ی مهم دیگری که باید در نظر داشت، شناسایی **boundary** (مرز) سیستم است که در تصویر حاضر نمایش داده شده است. **Actor** که در این نمودار **customer** نام دارد بیرون از دیاگرام قرار گرفته زیرا یک کاربر خارجی سیستم محسوب می شود.

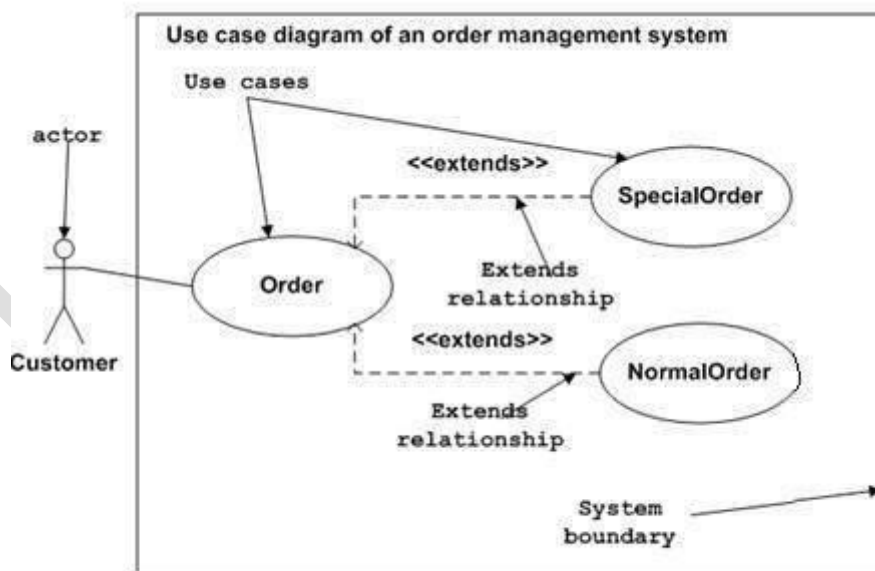


Figure: Sample Use Case diagram

کجا از نمودار use case استفاده می شود؟

همان طور که قبلاً شرح داده شد، در کل پنج نوع نمودار وجود دارد در زبان UML وجود دارد که با استفاده از آن ها نمای پویا از سیستم را مدل سازی می کنیم. تمامی مدل ها یک کارکرد خاص دارند. می توان گفت که این کارکردها در واقع نگاه به سیستم از زوایای مختلف است.

برای آشنایی با چگونگی عملکرد سیستم، بایستی انواع نمودار را مورد استفاده قرار داد. نمودار use case تنها یکی از مجموعه نمودار است و هدف آن شناسایی کارکردها و actor ها می باشد.

نمودارهای use case رخدادهای سیستم و جریان آن ها را نمایش می دهد، اما در به تصویر کشیدن نحوه ی پیاده سازی آن ها هیچ نقشی را ایفا نمی کند. نمودار use case را یک جعبه ی سیاه در نظر بگیرید که در آن فقط ورودی، خروجی و عملکرد جعبه سیاه برای ما مشخص می باشد.

این نمودارها در سطوح بالای طراحی بکار گرفته می شوند. سپس این طراحی سطح بالا بارها و بارها مورد بازبینی قرار گرفته و تصحیح می شود تا تصویری کاربردی و کامل از سیستم بدست آید. یک دیاگرام use case که به صورت کارآمد و سازمان یافته ترسیم شده باشد، precondition ها، post condition ها و استثنائات را نمایش می دهد. المان های نام برده همگی در ایجاد test case ها یا موارد آزمایش، به هنگام اجرای تست بکار می روند.

اگرچه use case ها گزینه های مناسبی برای مهندسی معکوس و روبه جلو نمی باشند، اما با کمی تنظیم می توان آن ها را برای مهندسی معکوس و روبه جلو بکار برد.

در مهندسی رو به جلو، می توان به کمک نمودارهای use case موارد آزمایش (test case) را شناسایی کرده و در مهندسی معکوس موارد کاربرد (use case) را جهت استخراج جزئیات کارکردها از یک برنامه ی از پیش ساخته و آماده استفاده کرد.

اهداف استفاده از نمودار use case در زیر شرح داده شده است:

1. تجزیه و تحلیل کارکردها (عملیاتی که قرار است انجام دهد و نیازهایی که برآورده سازد) و نیز طراحی سطح بالا.
2. مدل سازی بستر یا context سیستم.
3. مهندسی معکوس.
4. مهندسی رو به جلو.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

interaction diagram

مدرس : مهندس افشین رفوآ

دوره آموزش UML

نمودارهای برهمکنش (interaction diagram)

نمودارهای **interaction** زیر مجموعه ای از نمودارهای رفتاری (**behavior diagram**) هستند که بر گردش کنترل و داده‌ها بین چیزهای مختلف در سیستم مدل سازی شده تاکید دارند.

همان طور که از واژه ی **interaction** پیدا است، این نمودار به توصیف برهمکنش و تعامل میان المان های مختلف در مدل مورد نظر می پردازد. از این رو می توان گفت که **interaction** بخشی از رفتار پویا و داینامیک سیستم تلقی می شود.

این رفتار تعاملی (**interactive**) را توسط دو دیاگرام توالی (**sequence**) و همکاری (**collaboration**) در زبان مدل سازی **UML** به تصویر می کشند. دو نمودار نام برده از لحاظ هدف و مورد استفاده مشابه هستند. نمودار **sequence** بر ترتیب و توالی زمانی پیغام ها تاکید دارد و دیاگرام توالی بر روی ترتیب و سازمان دهی ساختاری اشیایی که پیغام ها را رد و بدل (ارسال/دریافت) می کنند، تمرکز می کند.

هدف از بکار بردن نمودار **interaction**

هدف از بکار بردن نمودارهای **interaction** را می توان نمایش گرافیکی و ارائه ی تصویری از رفتار تعاملی سیستم (برهمکنش بین اشیا سیستم) بیان کرد. به تصویر کشیدن **interaction** امر دشواری است و با تنها یک مدل قابل نمایش نمی باشد. راه حل استفاده از انواع مختلف مدل برای نمایش جنبه های مختلف **interaction** می باشد.

به این خاطر است که نمودارهای **sequence** و **collaboration** در کنار هم برای نمایش ماهیت پویا، اما از زاویه ای متفاوت، بکار می روند.

اهداف استفاده از نمودار **interaction** را می توان به ترتیب زیر شرح داد:

1. نمایش رفتار پویای سیستم.

2. توصیف جریان و گردش پیام در سیستم.

3. به تصویر کشیدن و توصیف ترتیب و سازمان دهی ساختاری اشیا.

4. شرح برهمکنش و تعامل بین اشیا (**object** ها).

نحوه ی ترسیم نمودار **interaction**

همان طور که قبلا گفته شد، مقصود اصلی استفاده از نمودارهای **interaction** نمایش جنبه های پویا و داینامیک یک سیستم می باشد، از این رو به منظور به تصویر کشیدن جنبه ی پویای سیستم ابتدا بایستی با مفهوم و معنی جنبه ی پویا و نیز نحوه ی نمایش گرافیکی آن آشنا شویم. جنبه ی پویای سیستم را می توان یک تصویر فوری (لحظه ای) از سیستم در حال اجرا در نقطه یا برهه ی زمانی مشخص در نظر گرفت.

در **UML** دو نوع دیاگرام وجود دارد که زیرمجموعه ی نمودار **interaction** محسوب می شوند: یکی نمودار توالی یا **sequence** و دیگری نمودار مشارکت، همکاری یا همان **collaboration** است. نمودار **sequence** توالی و ترتیب زمانی گردش یا جریان پیام از یک شی (**object**) به شی دیگر را نمایش داده و نمودار **collaboration** ترتیب و سازمان دهی ساختاری و همچنین رابطه ی بین اشیا که در گردش پیام مشارکت دارد را به نمایش می گذارد.

پیش از اقدام به رسم نمودار **interaction**، بایستی موارد زیر را شناسایی نمود:

1. اشییایی که در **interaction** شرکت دارند.

2. گردش و جریان پیام ها بین اشیا.

3. ترتیبی که پیام ها در آن جریان دارند.

4. ترتیب اشیا و رابطه ی بین آن ها.

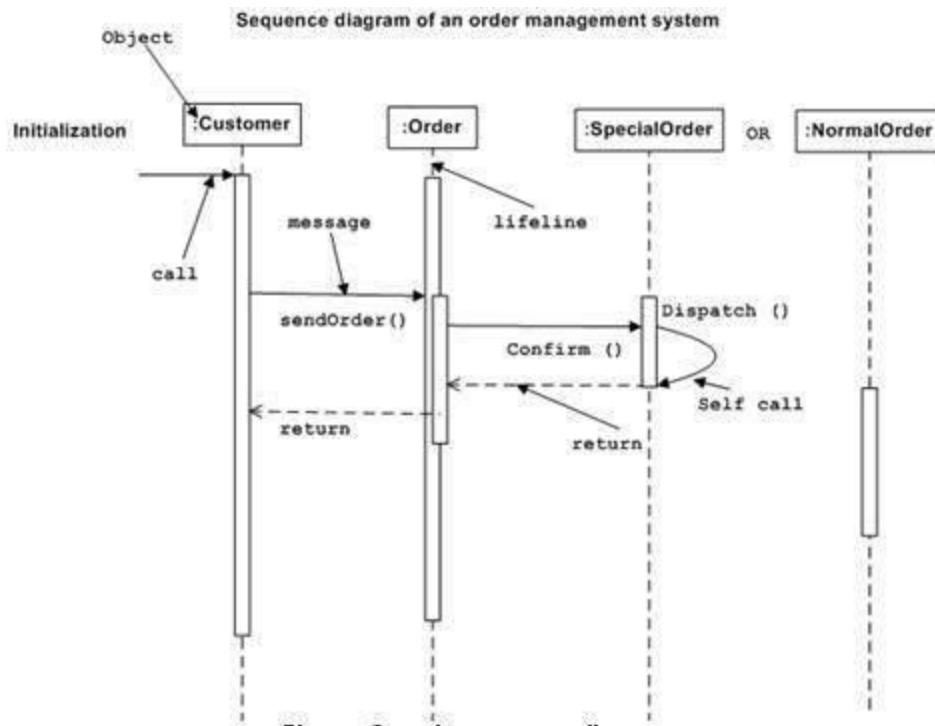
در زیر دو نمودار **interaction** می بینید که سیستم مدیریت سفارش را مدل سازی می کنند. اولین دیاگرام، یک نمودار **sequence** و دیگری یک نمودار **collaboration** می باشد.

نمودار Sequence

پیش تر گفته شد که نمودار **sequence** مشخص می نماید **object** ها چگونه با یکدیگر در قالب پیام هایی متوالی ارتباط برقرار می کنند و همچنین بیانگر طول عمر اشیا در رابطه با این پیامها می باشد. در دیاگرام حاضر **sequence** چهار شی شرکت دارند (**NormalOrder** و **SpecialOrder**، **Order**، **Customer**).

نمودار زیر توالی پیامی را برای شی **SpecialOrder** و **NormalOrder** نشان می دهد. حال باید توالی زمانی روند/گردش پیام ها را درک کرد. گردش پیام در واقع همان فراخوانی متدهای یک شی می باشد.

اولین فراخوانی مربوط به **sendOrder()** است که یکی از متدهای متعلق به شی **Order** می باشد. دومین فراخوانی، مربوط به متد **confirm()** است که یکی از رفتارها و عملیات قابل اجرا توسط شی **SpecialOrder** می باشد. باتوجه به آنچه گفته شد، نمودار حاضر عمدتاً فراخوانی متدهای اشیا را نشان می دهد. این دقیقاً همان اتفاقی است که در یک سیستم در حال اجرا رخ می دهد.

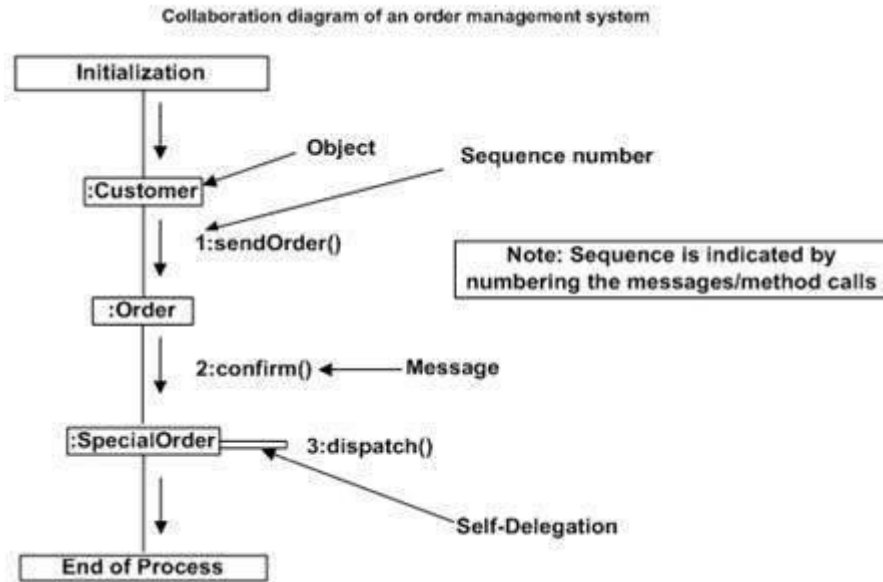


نمودار collaboration

دومین نموداری که زیر مجموعه ی نمودار **interaction** هستند، دیاگرام **collaboration** می باشد. دیاگرام **collaboration** بیشتر بر روی رابطه بین اشیا متمرکز می شود. در اینجا نمودار مربوطه ترتیب فراخوانی متدها را در قالب شماره هایی به صورت زیر نمایش می دهد. این شماره ها نشان می دهند متدهای مورد نظر چگونه یکی پس از دیگری صدا زده می شوند.

فراخوانی متدها در هر دو نمودار **sequence** و **collaboration** تقریباً یکسان می باشد نمودار همکاری شباهت بسیاری به نمودار توالی دارد، اصلی ترین تفاوت آنها در شمای ظاهری آنها می باشد. دیاگرام همکاری بیشتر بر روی ارتباط بین اشیا تاکید دارد، این درحالی است که دیاگرام ترتیب اعمال و کارهای اشیا را در یک توالی زمانی نشان می دهد و بر اساس زمان تنظیم و مرتب می شود .

برای انتخاب نمودارها، بایستی به نوع نیاز خود توجه کرد. اگر توالی زمانی مهم است، می بایست نمودار **sequence** را انتخاب کرد و اگر رابطه ی بین اشیا مهم است، در آن صورت باید از نمودار **collaboration** کمک گرفت.



موارد استفاده از نمودار interaction

پیشتر شرح دادیم که نمودارهای **interaction** برای توصیف ماهیت پویای سیستم بکار می رود. اکنون به سناریوهایی که در آن این نمودارها به صورت کاربردی مورد استفاده قرار می گیرد، خواهیم پرداخت. برای درک موارد کاربرد این دیاگرام بایستی با ماهیت این دو نمودار آشنا شویم.

هر دو نمودار برای ارائه ی نمای پویای سیستم بکار می رود. اما برای استفاده از آن باید هدف از بکاربردن هریک را تشریح نمود و درک کرد.

نمودار **sequence** ترتیب پیغام ها را که از یک شی به شی دیگر جریان دارد، نمایش می دهد. نمودار **collaboration** رابطه ی بین اشیای که در تعامل شرکت دارند، نمایش می دهد.

صرفاً یک نمودار قادر به توصیف جنبه ی پویای کل یک سیستم نیست، از این رو بایستی یک مجموعه نمودار را برای به تصویر کشیدن سیستم به صورت کامل مورد استفاده قرار داد.

حال اگر بخواهیم موارد کاربرد نمودارهای **interaction** را به طور خلاصه بیان کنیم:

1. مدل سازی گردش و جریان کنترل بر حسب زمان.
2. مدل سازی جریان کنترل بر اساس رابطه ی بین اشیا و ترتیب قرار گیری آن ها.
3. مهندسی معکوس.
4. مهندسی رو به جلو.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

Statechart diagram

مدرس : مهندس افشین رفوآ

دوره آموزش UML

نمودار وضعیت (Statechart diagram)

اسم نمودار **statechart** بیانگر موارد کاربرد آن می باشد. این نمودار همانطور که از نام آن پیداست حالت های مختلفی که یک شی در آن قرار می گیرد را مدل سازی می کند. در واقع این نمودار تصویری از چرخه حیات شی (**Object life cycle**) را به نمایش می گذارد. این وضعیت ها مختص به یک شی/مولفه ی (**object/component**) خاص از سیستم مورد نظر هستند.

نمودار **statechart** یک ماشین وضعیت (**state machine**) را توصیف می کند. ماشین وضعیت برای نمایش وضعیت های مختلف یک شی در سیستم و همچنین نمایش انتقال بین وضعیت ها بکار می رود. وضعیت های نام برده توسط رخدادهای (**event**) داخلی یا خارجی مدیریت و کنترل می شود.

نمودار **activity** زیرمجموعه ی نمودار **statechart** می باشد که در مبحث بعدی به شرح آن خواهیم پرداخت. از آنجایی که نمودار **statechart** وضعیت و حالات مختلفی که شی در آن قرار می گیرد را تعریف می کند، از آن برای مدل کردن چرخه ی حیات یک شی استفاده می شود.

موارد کاربرد

نمودار **statechart** یکی از پنج نمودار **UML** است که برای مدل سازی ماهیت پویای یک سیستم بکار می رود. وضعیت های مختلف یک شی را در طول چرخه ی حیات آن مشخص می کند. یادآور می شویم که این وضعیت ها توسط رخدادها تغییر می کنند. سیستم های واکنشی (**reactive system**) را می توان سیستمی تعریف کرد که نسبت به رویدادهای داخلی و خارجی واکنش نشان می دهد.

نمودار **statechart** گردش یا جریان کنترل (**control flow**) را از یک **state** به **state** دیگر شرح می دهد. **State** را می توان یک وضعیت در نظر گرفت که در آن یک شی وجود دارد و این وضعیت، هنگامی که رخدادی فعال شده و روی می دهد، تغییر می کند. با استناد به توضیحات بالا می توان گفت که مهم ترین مورد کاربرد نمودار **statechart**، مدل سازی چرخه ی حیات یک شی از زمان ایجاد تا پایان عمر آن شی است. نمودار **Statechart** برای مدل سازی مهندسی معکوس و رو به جلوی یک سیستم نیز بکار می رود. با این حال عمده ی استفاده ی آن، مدل سازی سیستم های واکنشی (**reactive**) می باشد.

موارد استفاده از نمودار حالت (**State Chart Diagram**)

1. مدل سازی جنبه ی پویای سیستم.
2. مدل سازی چرخه ی حیات سیستم واکنشی.
3. توصیف وضعیت های مختلفی که شی در طول چرخه ی حیات خود در آن ها قرار می گیرد.
4. تعریف یک **state machine** جهت مدل سازی وضعیت های مختلف یک شی.

نحوه ی ترسیم نمودار **statechart**

یادآور می شویم که نمودار **statechart** به منظور توصیف وضعیت هایی که شی در طول عمر خود در آن ها قرار می گیرد استفاده می شود. با توجه به آنچه گفته شد، تاکید بر روی تغییر وضعیت هایی قرار می گیرد که به مجرد اتفاق افتادن رخدادهای داخلی و خارجی، صورت می گیرد. لازم است که وضعیت اشیا را مورد تجزیه و تحلیل قرار داده و به دقت پیاده سازی نمود.

نمودارهای **statechart** برای توصیف وضعیت های مختلف شی از اهمیت بالایی برخوردار است. **State** ها را می توان وضعیت اشیا تعریف کرد که با فعال شدن رخدادهایی دستخوش تغییراتی قرار می گیرند.

پیش از اقدام به رسم نمودار **statechart**، بایستی نکات زیر را تشریح نمود:

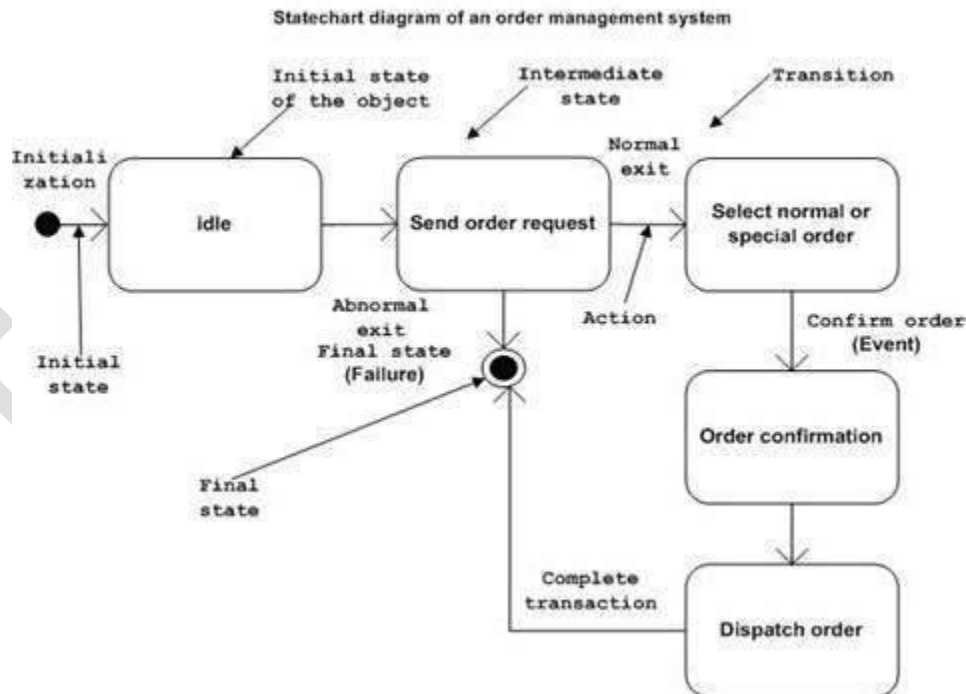
1. شناسایی اشیا که می بایست مورد تجزیه و تحلیل قرار گیرد.
2. مشخص کردن وضعیت ها.

3. شناسایی رخدادها.

در زیر مثالی از یک نمودار **Statechart** را مشاهده می کنید که در آن وضعیت شی **Order** مورد تجزیه و تحلیل قرار گرفته است.

State اول، بیانگر وضعیت **idle** (آماده بکار و اولیه) می باشد که فرایند از آنجا آغاز می شود. شی پس از فعال شدن هر یک از رخدادهای **send request**، **confirm request** و **dispatch order** در وضعیت های مختلف قرار می گیرد. درحقیقت رخدادهای ذکر شده مسئول تغییر وضعیت شی **order** می باشند.

در طول چرخه ی حیات خود، یک شی (**order**) وضعیت های زیر را تجربه می کند که در این میان ممکن است خروج غیرمنتظره رخ دهد. این خروج غیره منتظره می تواند بر اثر وجود یک مشکل در سیستم روی دهد. پس از اینکه چرخه ی حیات سیستم کامل شد، تراکنش کامل زیر حاصل می گردد. وضعیت اولیه و پایانی شی نیز در نمودار زیر به نمایش گذاشته شده است:



نمودارهای state chart کجا کاربرد دارد؟

این نمودار همانند چهار نمودار ذکر شده در این آموزش، جنبه های پویای یک سیستم را به تصویر می کشد. اما این دیاگرام دارای ویژگی های اختصاصی برای مدل سازی جنبه ی پویای یک سیستم می باشد.

نمودار **Statechart**، وضعیت های یک مولفه (**component**) را شرح می دهد. این وضعیت ها تغییر می کنند و این تغییر وضعیت ها به ذاته پویا می باشند. بنابراین مقصود اصلی از بکاربردن نمودار **statechart** تعریف تغییراتی است که با فعال شدن رخدادهایی در وضعیت رخ می دهند. رخدادها عوامل داخلی و خارجی هستند که سیستم را تحت تاثیر قرار می دهند.

نمودارهای **statechart** جهت مدل سازی وضعیت ها و رخدادهایی تعبیه شده که بر روی سیستم مورد نظر عملیاتی را انجام می دهند. به هنگام پیاده سازی سیستم، بایستی وضعیت های مختلفی که یک شی در طی چرخه ی حیات خود در آن قرار می گیرد را توضیح داد، نمودار **statechart** نیز دقیقا به همین منظور بکار می رود. پس از اینکه وضعیت ها و رخدادهای آن شناسایی می شوند، آن ها را برای مدل سازی مورد استفاده قرار داده و این مدل ها حین پیاده سازی سیستم بکار می روند.

اگر با دقت به جنبه ی پیاده سازی کاربردی نمودار **Statechart** دقت کنیم، متوجه می شویم که مورد استفاده ی اصلی آن تجزیه و تحلیل وضعیت های شی است که توسط رخدادها دست خوش تغییر قرار می گیرند. این تجزیه و تحلیل در درک رفتار سیستم در طول اجرای آن بسیار مفید است.

به طور خلاصه می توان موارد استفاده ی این دیاگرام را بدین صورت نام برد:

1. جهت مدل سازی وضعیت های شی یک سیستم.
2. مدل کردن سیستم واکنشی. این سیستم ها خود از اشیا واکنشی یا **reactive** تشکیل می شوند.
3. شناسایی رخدادهایی که مسئول این تغییر وضعیت ها هستند.
4. مهندسی رو به جلو و معکوس.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

Activity diagram

مدرس : مهندس افشین رفوآ

دوره آموزش UML

نمودار فعالیت (Activity diagram)

نمودار **activity** یکی دیگر از نمودارهای مهم در UML می باشد که نمایی از جنبه ی پویای سیستم مورد نظر بدست می دهد.

دیاگرام های فعالیت که شامل **activity** ها و **state** ها و **transition** ها می باشند در راستای تعریف

جریان کاری مورد استفاده قرار می گیرند. در واقع وسیله ای برای تجزیه و تحلیل سطوح مختلف

محسوب می شود. در مرحله طراحی (**design phase**)، این دیاگرام کمک می کند تا عملیات ها را بهتر تعریف کنیم.

در این نمودار چگونگی جریان انجام یک کار یا فعل مشخص می شود.

دیاگرام مذکور را می توان یک نمودار گردش و روند کاری تعریف کرد که جریان کار را از یک فعالیت یا **activity** به فعالیت دیگر نشان می دهد. این **activity** را می توان یکی از افعال یا عملیات سیستم مد نظر تلقی کرد.

بنابراین جریان کنترل از یک فعل یا عمل به فعل یا عمل دیگری ترسیم می شود. این جریان می تواند ترتیبی

(**sequential**)، پخش یا شاخه شاخه شده (**branched**)، همزمان (**concurrent**) باشد. دیاگرام **activity** با

بهره گیری از المان های متعدد همچون **fork** (انشعاب)، **join** (پیوند) قادر است انواع جریان های کنترل را

مدیریت کند.

هدف از بکاربردن نمودار activity

هدف اصلی که نمودار **activity** دنبال می کند، بسیار شبیه به دیگر چهار دیاگرام در **UML** است، بدین معنا که سعی دارد رفتار پویا و داینامیک سیستم را به تصویر بکشد. چهار دیاگرام دیگر که در مقاله ی آموزشی حاضر درباره ی آن بحث شد، جریان پیام (**message flow**) را از یک شی به شی دیگر نمایش می دهد، در حالی که دیاگرام **activity** جریان پیام را از یک فعل یا عمل به فعل یا عمل دیگر نشان می دهد.

همان طور که قبلاً تشریح شد، **activity** همان فعل و عملی است سیستم قادر به انجام آن می باشد. نمودارهای **activity** صرفاً برای نمایش گرافیکی ذات **dynamic** سیستم بکار نمی رود، بلکه مورد استفاده ی فرعی نیز دارند و آن در ساخت سیستم های اجرایی (**executable system**) به وسیله ی مهندسی رو به جلو یا معکوس خلاصه می شود. تنها آیتمی که در نمودار **activity** نمایش داده نمی شود، بخش مربوط به پیام ها می باشد، بدین معنی که نمودار یاد شده هیچگونه جریان پیامی را از یک فعل به فعل دیگر نمایش نمی دهد. نمودار **activity** را گاهی به غلط **flowchart** یا نمودار روند و جریان کار نیز در نظر می گیرند. اگرچه نمودارهای **activity** ظاهری شبیه به **flowchart** دارند، اما نمی توان آن ها را صد در صد یک **flowchart** تلقی کرد.

دیاگرام مورد نظر این قابلیت را دارد که جریان های مختلف همانند موازی (**parallel**)، منشعب (**branched**)، همروند (**concurrent**) و منفرد (**single**) را به تصویر بکشد.

اهداف و موارد کاربرد این نمودار را می توان به ترتیب زیر شرح داد:

1. ترسیم جریان **activity** سیستم مورد نظر.

2. تشریح توالی و **sequence** از یک **activity** یا فعل به فعل دیگر.

3. توصیف جریان موازی، منشعب و همروند یک سیستم.

نحوه ی ترسیم نمودار activity

نمودارهای **activity** عمدتاً به صورت یک **flow chart** مورد استفاده قرار می گیرند که از افعال و عملیات قابل اجرا توسط سیستم تشکیل می شود. اما نمودار **activity**، به دلیل داشتن قابلیت های اضافی بر سازمان نظیر **branching flow**، **parallel flow** و **swimlane**، کمی با **flowchart** تفاوت دارد.

پیش از اقدام به رسم نمودار **activity** می بایست فهم دقیقی از المان های بکار رفته در نمودار مزبور کسب نمود. عنصر اصلی بکار رفته در نمودار **activity**، خود **activity** می باشد.

با توجه به آنچه گفته شد، **activity** یک وظیفه یا عمل است که سیستم انجام می دهد. پس از شناسایی **activity** های سیستم بایستی چگونگی رابطه ی آن ها با **constraint** ها (محدودیت ها)، **condition** ها را درک کرد.

بنابراین قبل از اینکه به رسم نمودار بپردازیم، لازم است المان های زیر را مشخص کنیم:

1. **activity** ها

2. **association** (رابطه ها)

3. **condition** ها

4. **constraint** ها (محدودیت ها)

پس از شناسایی پارامترهای فوق، باید یک طرح کلی از کل جریان را در ذهن خود ایجاد نمود، سپس این طرح ذهنی را در قالب نمودار **activity** به نمایش گذاشت.

در زیر نمونه ای از دیاگرام **activity** را مشاهده می کنید که سیستم مدیریت سفارش را به تصویر کشیده است. در این نمودار چهار **activity** شناسایی شده که با شرط هایی گره خورده (مرتبط) است. یک نکته که باید به آن توجه کرد این است که نمودار **activity** را نمی توان با کد برابر دانست و آن را جهت پیاده سازی کدها بکار برد. نمودار **activity** بیشتر به منظور درک جریان **activity** ها مورد استفاده قرار گرفته و توسط **business user** ها بکار می رود.

Activity های نمودار حاضر به ترتیب فهرست شده در زیر می باشد:

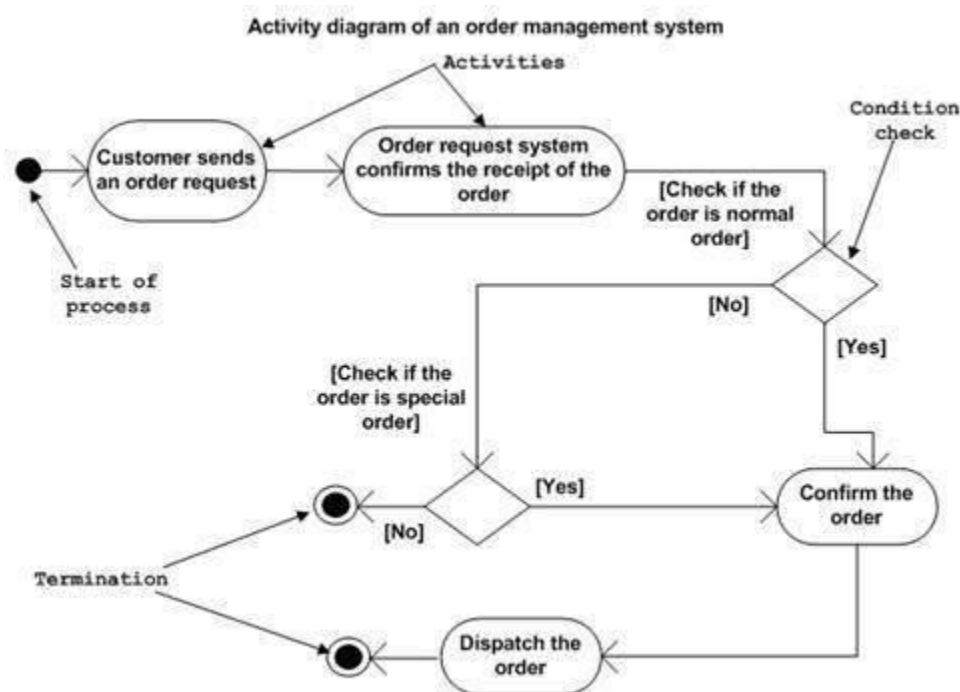
1. ارسال سفارش توسط مشتری

2. دریافت سفارش

3. تایید سفارش

4. ارسال سفارش

پس از دریافت درخواست سفارش، عملیات بررسی شرط اجرا شده تا مشخص شود آیا سفارش از نوع مخصوص است یا معمولی. با مشخص شدن نوع سفارش، **activity** مربوط به ارسال اجرا شده که نشانگر پایان پروسه می باشد.



کجا از این نمودار استفاده می شود؟

نمودار **activity** در هدف اصلی خود با چهار نمودار دیگر UML مشترک می باشد. مورد کاربرد ویژه ای که آن را از دیگر نمودارها متمایز می کند، مدل سازی جریان کنترل از یک **activity** به **activity** دیگر است. لازم به ذکر است که این جریان کنترل پیام ها را شامل نمی شود.

نمودار **activity** مناسب مدل سازی جریان **activity** سیستم می باشد. یک برنامه می تواند چندین سیستم داشته باشد. نمودار **activity** همچنین این سیستم ها را به تصویر کشیده و جریان را از یک سیستم به سیستم دیگر نمایش می دهد. این کاربرد را دیگر دیاگرام ها فراهم نمی کنند.

این سیستم ها می توانند پایگاه داده، صف های خارجی (**external queue**) یا هر سیستم دیگری باشند.

اکنون به جنبه های کاربردی نمودار **activity** می پردازیم. از توضیحاتی که در بالا برای دیاگرام مورد نظر ارائه شد می توان نتیجه گرفت که این نمودار نما یا دید سطح بالا از سیستم فراهم می نماید. **View** سطح بالایی که این دیاگرام عرضه می کند غالباً توسط کاربرانی مورد استفاده قرار می گیرد که دید تجاری نسبت به نمودار دارند یا اشخاصی هستند که دانش فنی ندارد.

نمودار **activity** عمدتاً به منظور مدل کردن **activity** ها یا افعالی کاربرد دارد که نیازهای تجاری را برآورده می سازند. بنابراین نمودار **activity** تاثیر خود را غالباً بر روی فهم تجاری می گذارد تا جزئیات پیاده سازی.

در زیر اهداف اصلی استفاده از نمودار **activity** را مشاهده می کنید:

1. مدل سازی جریان کاری به وسیله ی **activity** ها.
2. مدل سازی کارکردها و نیازهای تجاری.
3. درک سطح بالا از قابلیت ها و عملیات قابل اجرا توسط سیستم.
4. برای شناسایی **Use Case** ها
5. برای تشریح ارتباط میان **Use Case** ها
6. برای تشریح پیچیدگی و نمودار جریان کاری (**flowchart**) یک عمل در یک **Use Case**
7. برای توصیف جزئیات فرایندها در یک **Activity** سطح بالا

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

مروری کلی بر زبان مدل سازی UML

مدرس : مهندس افشین رفوآ

دوره آموزش UML

مروری کلی بر زبان مدل سازی UML

UML یک زبان مدل سازی همه منظوره می باشد. این زبان در ابتدا به منظور نمایش تصویری رفتار سیستم های نرم افزاری و غیر نرم افزاری پیچیده اختراع شده و ارائه گردید اما هم اکنون یک استاندارد **OMG** (گروه مدیریت آبجکت) تلقی می شود.

UML المان ها و **component** هایی ارائه می کند که قادر به برآورده ساختن نیازهای سیستم های پیچیده می باشد. **UML** از مفهیم پایه ای و متدولوژی شی گرا پیروی می کند، بدین معنی که با بهره گیری از زبان های تصویر نما همچون **UML** می توان آن ها را مدل سازی کرد.

نمودارهای **UML** از دیدگاه های مختلف نظیر طراحی، پیاده سازی، استقرار/توزیع و غیره ... ترسیم می شوند. در خلاصه، زبان **UML** را می توان یک زبان مدل سازی که قادر به نمایش جنبه هایی همانند معماری، رفتار و ساختار سیستم می باشد، تعریف کرد.

اشیا اساس و بنیاد دنیای شی گرا محسوب می شوند. اولین کاری که در تجزیه، تحلیل و طراحی شی گرا بایستی انجام داد، شناسایی کارآمد اشیا می باشد، پس از آن فقط بایستی مسئولیت ها را به اشیا تخصیص داد. در مرحله ی بعد، به طراحی با استفاده از خروجی تجزیه و تحلیل خواهیم پرداخت.

UML نقش مهمی در تحلیل و طراحی شی گرا ایفا می نماید. نمودارهای این زبان نیز طراحی را مدل سازی کرده و به صورت گرافیکی ارائه می کند. بنابراین می توان نتیجه گرفت که **UML** نقش بسیار مهمی را ایفا کرده و جایگاه بسیار مهمی دارد.

نشانه گذاری های زبان مدل سازی UML (UML notations)

Notation ها مهم ترین المان ها در مدل سازی محسوب می شوند. استفاده ی کارآمد و بجا از **notation** ها در جهت ایجاد یک مدل کامل و معنی دار کمک شایانی می کند. یک مدل تا زمانی که هدف و مورد کاربرد خود را به درستی به تصویر نکشد، کاملاً بی ارزش قلمداد می شود.

از این رو تاکید بر یادگیری **notation** ها از اهمیت خاصی برخوردار است. برای موارد و رابطه های مختلف طبیعتاً از **notation** های مختلف بهره گرفته می شود. نمودارهای UML نیز از همین نشانه گذاری های (**notation**) اشیا و رابطه ی بین آن ها تشکیل می شوند. توسعه پذیری (**extensibility**) یکی دیگر از ویژگی ها و قابلیت هایی است که به کارایی و انعطاف پذیری هرچه بیشتر UML کمک می کند.

نمودارهای UML

نمودارها قلب UML هستند. این نمودارها به طور کلی تحت عناوین ساختاری (**Structural**) و رفتاری (**behavioral**) رده بندی می شوند.

1. دیاگرام های ساختاری خود به نمودارهای ایستا (نمودارهایی که ساختار ایستای سیستم را نمایش می دهند) از قبیل نمودار کلاس، شی و غیره ... گروه بندی می شوند.

2. نمودارهای رفتاری نیز به دیاگرام های پویایی (نمودارهایی که تاکید برنمایش رفتار پویای سیستم با نشان دادن هماهنگی میان اشیا و تغییرات وضعیت داخلی اشیا دارند) نظیر توالی (**sequence**)، همکاری (**collaboration**) و غیره ... دسته بندی می شوند.

با توجه به آنچه گفته شد، توسط این نمودارها می توان ماهیت پویا و ایستای یک سیستم را به تصویر کشید.

نمودار کلاس (class diagram)

نمودار کلاس پرکاربردترین دیاگرام در UML می باشد که توسط جامعه ی شی گرا مورد استفاده قرار می گیرد. این نمودار به شرح اشیا در یک سیستم و رابطه ی بین آن ها می پردازد. دیاگرام کلاس از توابع (**function**) و متغیرهای عضو (**attribute**) تشکیل می شود.

یک نمودار کلاس تنها قادر به نمایش یک جنبه از سیستم مورد نظر می باشد درحالی که چندین نمودار کلاس می توانند نمایی ایستا از کل سیستم بدست دهند. همان طور که گفته شد نمودار کلاس در اصل نمایی ایستا از سیستم را ارائه می دهد.

نمودارهای کلاس تنها نمودارهای **UML** هستند که به طور مستقیم قابل نگاشت توسط زبان های شی گرا همچون **C#** می باشد. از این رو به طور متداول توسط جامعه ی برنامه نویسان و توسعه دهندگان مورد استفاده قرار می گیرد.

نمودار شی (object diagram)

نمودار شی درواقع نمونه ای از دیاگرام کلاس می باشد و به همین خاطر بسیاری از المان های پایه ای بین این دو نمودار یکسان و مشترک می باشد. دیاگرام های شی از اشیا و پیوندها (**link**) تشکیل می شوند؛ به بیان دیگر نشانگر یک دید کامل یا جزئی از ساختار سیستم مدل سازی شده در یک زمان مشخص است. نمودارهای شی برای پیش الگو سازی (**prototyping**)، مهندسی معکوس و نیز مدل سازی سناریوهای کاربردی مورد استفاده قرار می گیرد.

نمودار اجزا (component diagram)

نمودار **component** یکی از مجموعه دیاگرام هایی است که جهت توصیف و ارائه ی دید ایستا از یک سیستم بکار می رود. این نمودار چگونگی تقسیم سیستم به اجزای آن و وابستگی بین اجزای سیستم را تشریح می نماید. نمودار **component** متشکل از مصنوعات و اجزای فیزیکی از قبیل کتابخانه ها، فایل ها، پوشه ها و غیره ... می باشد.

این نمودار از دیدگاه پیاده سازی و اجرایی مورد استفاده قرار می گیرد. به منظور نمایش گرافیکی کل یک سیستم به مجموعه ای از نمودارهای **component** نیاز است. با استفاده از تکنیک مهندسی رو به جلو و معکوس می توان از نمودار مزبور فایل های اجرایی (**executable**) ایجاد نمود.

نمودار استقرار (deployment diagram)

نمودار های **deployment** سخت‌افزار بکار رفته در پیاده‌سازی سیستم و همچنین محیط‌های اجرا و سایر اجزایی که باید بر روی این سخت‌افزار قرار گیرند را شرح داده و به تصویر می‌کشد. این نمودارها عمدتاً مورد کاربرد مهندسين سیستم هستند.

نمودارهای **deployment** از گره‌ها (**node**) و رابطه‌ی میان آن‌ها شکل گرفته و ساخته می‌شود. یک نمودار **deployment** کارآمد، بخش جدایی‌ناپذیر چرخه‌ی توسعه‌ی نرم‌افزار کاربردی می‌باشد.

نمودار مورد کاربرد (Use case)

نمودار **use case** به منظور نمایش ماهیت پویای یک سیستم بکار می‌رود. نمودار ذکر شده کارکرد ارائه شده توسط یک سیستم را در قالب عملگرها (**Actor**) و اهداف آنها که به صورت **use case** نمایش داده می‌شوند و وابستگی بین موردهای کاربرد، مدل و مصورسازی می‌کند. این نمودار از اجزایی همچون **use case** ها، **actor** ها و رابطه‌ی آن‌ها تشکیل می‌شود.

دیاگرام‌ها **use case** در سطح بالا برای مصورسازی کارکردهای سیستم و نیازهایی که برطرف می‌سازد، بکار می‌رود.

در خلاصه می‌توان گفت که نمودار **use case** کارکردهای سیستم و جریان آن‌ها را مدل‌سازی می‌کند.

نمودار تعامل یا برهمکنش (interaction diagram)

نمودارهای **interaction** جهت ارائه‌ی نمایی از جنبه‌ی پویای سیستم بکار می‌رود. نمودارهای **sequence** و **collaboration** هر دو زیرمجموعه‌ی نمودارهای **interaction** هستند که برای نمایش گرافیکی ذات پویای سیستم بکار می‌روند. در واقع این نمودار بر گردش کنترل و داده‌ها بین چیزهای مختلف در سیستم مدل شده تاکید دارند.

نمودار **sequence** برای نمایش ترتیب زمانی بکار می‌رود یا به عبارتی دیگر نشان می‌دهد که اشیا چگونه با یکدیگر در قالب پیام‌هایی متوالی تعامل دارند و همچنین نمایشگر طول عمر اشیا نسبت به این پیام‌ها

می‌باشد. دیاگرام **collaboration** بیشتر بر روی رابطه بین اشیا تاکید دارد. در حالی که یک دیاگرام **sequence** اعمال اشیا را در یک توالی زمانی نشان می‌دهد و بر حسب زمان تنظیم می‌شود. برای نمایش کل یک سیستم معمولاً از ترکیبی از چندین نمودار **collaboration** و **sequence** استفاده می‌شود.

نمودار حالت (statechart diagram)

نمودارهای **statechart** یکی از پنج دیاگرامی است که برای مدل‌سازی جنبه‌های پویای سیستم بکار می‌رود. این نمودارها به منظور مدل‌سازی کل چرخه‌ی حیات یک شی بکار می‌روند. نمودار گفته شده همان طور که از اسم آن پیداست مدلی از حالت‌های مختلفی که یک شی در آن قرار می‌گیرد، ارائه می‌دهد. در واقع این نمودار تصویری از چرخه حیات شی (**object life cycle**) را به نمایش می‌گذارد. نمودار **activity** زیرمجموعه‌ی این نمودار محسوب می‌باشد.

State را می‌توان وضعیت یا حالتی که شی در زمان مشخص در آن قرار می‌گیرد تعریف کرد. وضعیتی که شی در آن به سر می‌برد با فعال شدن **event**‌هایی تغییر می‌یابد. این نمودار برای مهندسی معکوس و رو به جلو نیز بکار می‌رود.

نمودار activity

یکی دیگر از نمودارهای پرکاربرد و مهم **UML** که نمایشگر رفتار داینامیک یا پویای سیستم می‌باشد، نمودار **activity** است. این نمودار از عناصری نظیر **activity**‌ها (افعال سیستم)، پیوندها (**link**) و رابطه‌ها تشکیل می‌شود. نمودار ذکر شده همچنین انواع جریان‌ها همچون موازی (**parallel**)، واحد (**single**)، همروند (**concurrent**) و منشعب یا شاخه‌ای (**branched**) و غیره... را مدل‌سازی می‌کند.

نمودار مورد نظر کنترل جریان از یک **activity** به **activity** را بدون پیام‌ها نمایش می‌دهد. این نمودار برای توصیف قدم به قدم گردش کار تجاری و عملیاتی اجزا و مصنوعات سیستم استفاده می‌شود. این نمودار نمایشگر گردش کنترل در سراسر سیستم است.

بسم الله الرحمن الرحيم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

آموزش Placeholder در HTML 5

مدرس : مهندس افشین رفوآ

دوره آموزش UML

مرور کلی بر ویرایش 2.0 زبان مدل سازی UML

UML 2.0 یک بعد کاملاً متفاوتی از زبان مدل سازی یکپارچه ارائه می دهد، به گونه ای از نظر قابلیت بسیار پیچیده و غنی تر می باشد.

در این نسخه مستندسازی شکل دیگری به خود گرفته و در مقایسه با ویرایش 1.5 به طور قابل توجهی گسترش یافته است.

UML 2.0 امکاناتی به این زبان افزود که موارد کاربرد آن را نسبت به گذشته گسترده تر ساخته است.

UML 2.0 تعریف کاملاً مشخص و رسمی از semantics را ارائه می دهد. این امکان نوین را می توان جهت توسعه مدل ها مورد استفاده قرار داد، سپس سیستم های مربوطه را از این مدل ها ایجاد کرد.

ابعاد جدید UML 2.0

ساختار و مستندسازی زبان مدل سازی در ویرایش 2.0 به طور بنیادی مورد بازبینی و ویرایش قرار گرفت. در حال حاضر دو نوع سند در UML موجود می باشد:

1. Infrastructure ساختارهای اولیه ای که زبان مدل سازی بر پایه ی آن بنا شده را مشخص می کند. این بخش مستقیماً به کاربران UML مربوط نمی باشد، بلکه بیشتر توجه خود را به توسعه دهندگان ابزار مدل سازی متمرکز می کند. در واقع Infrastructure هسته ی metamodel که Superstructure بر پایه ی آن می باشد را تعریف می کند.

2. **Superstructure** که **notation** و **semantics** را برای نمودارها و المان های مدل آن ها تعریف می کند. **Superstructure** در واقع سازه های مورد استفاده ی کاربر را تعریف می کند. منظور از سازه های مورد استفاده کاربر آن المان هایی است که کاربر در سطوح اولیه بکار می برد. بنابراین تمرکز اصلی این بخش بر جامعه ی کاربران زبان **UML** می باشد.

نسخه ی ویرایش شده ی **UML** به منظور برآورده ساختن یک هدف که بازسازی و اصلاح **UML** می باشد، ایجاد شد. با ویرایش و بهبود این زبان هم اکنون قابلیت استفاده، تطبیق و پیاده سازی به مراتب آسان تر شده است.

Infrastructure زبان **UML** برای موارد زیر بکار می رود:

1. ارائه ی یک هسته ی **meta-language** که دارای قابلیت استفاده ی مجدد می باشد. **Infrastructure** به منظور تعریف خود **UML** و زیرساخت های آن بکار می رود.

2. ارائه ی مکانیزم هایی که امکان تنظیم زبان را فراهم می آورد.

Superstructure کارهای زیر را انجام می دهد:

1. ارائه ی پشتیبانی بهتر برای توسعه ی مبتنی بر **component**.

2. بهبود سازه هایی که برای تعریف معماری مورد استفاده قرار می گیرد.

3. ارائه ی گزینه های بهتر برای مدل سازی رفتار.

نکاتی که در بالا تشریح شد، قابلیت استفاده **UML** را افزایش داده و یک فهم دقیق تر از کاربرد آن ارائه می دهد.

یک بعد جدید دیگری نیز وجود دارد که در نسخه ی جدید پیشنهاد شده است. این یک پیشنهاد جدید برای **OCL (Object Constraint Language)** و **Diagram Interchange** می باشد. تمامی ویژگی های مذکور پکیج کامل **UML 2.0** را تشکیل می دهند.

نمودارهای مدل سازی در ویرایش UML 2.0

مدل سازی تعاملات و برهمکنش ها:

نمودارهای **interaction** که در نسخه ی 2.0 زبان مدل سازی **UML** ارائه شد، با ویرایش های قبلی کمی متفاوت است. گفتنی است که مفهوم پایه ای تغییر نکرده، بلکه قابلیت ها و امکانات جدید ارائه شده که کارایی نمودارها را در نسخه ی نوین بهبود می بخشد.

UML2.0 تعامل اشیا (**object interaction**) را به چهار شیوه ی زیر مدل کرده و مصورسازی می نماید:

1. نمودار **sequence** یک نمای وابسته به زمان از تعامل بین اشیا ارائه می دهد که اهداف رفتاری یک سیستم را برآورده می سازد. توالی زمانی شباهت بسیار زیادی به ویرایش قبلی دارد. یک تعامل می تواند در هر سطحی از انتزاع در **design** سیستم، طراحی شود: از تعامل **subsystem** گرفته تا تعامل در سطح نمونه. در واقع این نوع نمودار نشان می دهد که اشیا چگونه با یکدیگر در قالب پیامهایی متوالی ارتباط برقرار می کنند.

2. نمودار **communication** بر همکنش و تعامل بین اشیا و یا اجزا را در قالب توالی پیام های رد و بدل شده نمایش می دهد. این نمودار نمایشگر ترکیبی از اطلاعات به دست آمده از نمودارهای **class**، **sequence** و **use case** است و هر دوی ساختار ایستا و رفتار پویای سیستم را شرح می دهند. این نمودار را می توان نسخه ی ویرایش شده ی نمودار **collaboration** معرفی کرد.

3. نمودار کلی برهمکنش یا **interaction overview** یک دید کلی ارائه می دهد که در آن **node** ها نشانگر نمودارهای برهمکنش **interaction** هستند. نمودار ذکر شده همچنین یک دید سطح بالا از مجموعه تعاملات که در قالب یک توالی و ترتیب منطقی گنجانده شده فراهم می نماید که شامل منطق کنترل جریان برای پیمایش بین این برهمکنش ها و تعاملات می باشد.

4. نمودار **timing** یا زمان بندی گونه ویژه ای از نمودارهای برهم کنش هستند که بر روی محدودیت های زمانبندی تمرکز دارند. این نمودار در ویرایش 2 ارائه شد و بر محدودیت های زمانی اعمال شده بر پیام های ارسالی و دریافتی در طول یک برهمکنش تمرکز دارد.

از توصیفات بالا نتیجه می‌گیریم که اهداف تمامی نمودارها در ارسال/دریافت پیام‌ها خلاصه می‌شود. حال مدیریت این پیام‌ها به صورت داخلی در اشیا صورت می‌گیرد. از این رو اشیا نیز می‌توانند پیام‌ها را دریافت و ارسال کنند. در این بین سخن از جنبه‌ی دیگری به نام **interface** یا رابط به میان می‌آید. این رابط‌ها مسئول مبادله‌ی پیام‌ها (دریافت و ارسال پیام‌ها بین یکدیگر) می‌باشند.

با در نظر گرفتن توضیحاتی که تا کنون ارائه شد، این مسئله روشن می‌شود که تعاملات و برهمکنش‌ها در **UML 2.0** به گونه‌ای دیگر به تصویر کشیده می‌شوند و به همین منظور هم است که نمودارهای جدید به **UML** اضافه شدند. با این حال اگر نمودارهای جدید را مورد تجزیه و تحلیل قرار دهیم، پی می‌بریم که این نمودارها هم بر مبنای همان دیاگرام‌های نسخه‌ی قبلی پی‌ریزی و طراحی شده است. تنها تفاوت آن‌ها در ویژگی‌های جدیدی است که به نمودارهای مزبور اضافه شده که کارایی آن‌ها را بهبود بخشیده و آن‌ها را هدفمندتر ساخته.

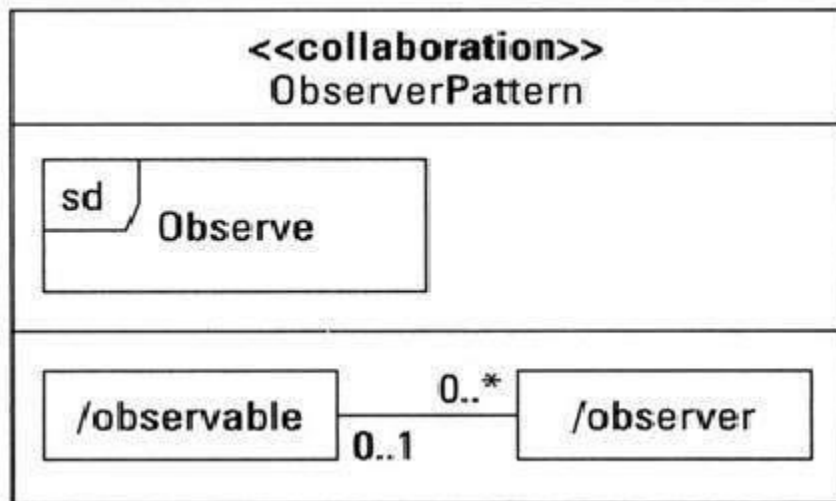
مدل‌سازی همکاری:

همان‌طور که پیش‌تر تشریح شد، **collaboration** جهت مدل‌سازی تعاملات و همکاری عادی بین اشیا بکار می‌رود. به عبارتی روشن‌تر، **collaboration** یک برهمکنش است که طی آن مجموعه‌ای از پیام‌ها توسط یک سری اشیا که نقش‌های از پیش تعیین شده دارند مدیریت می‌شود.

مهم‌ترین نکته قابل توجه، تفاوتی است که بین نمودار **collaboration** در دو نسخه‌ی 2.0 و قبلی وجود دارد. برای تمایز قائل شدن بین نمودار نام برده در ویرایش قبلی و کنونی، اسم آن از **collaboration** به **communication** تغییر یافته است.

Collaboration را می‌توان یک کلاس تعریف کرد که دارای **attribute**‌ها (خصیصه‌ها) رفتار یا عملیاتی می‌باشد. بخش‌های یک کلاس **collaboration** می‌تواند **user defined** بوده و برای نمودارهای **interaction** (نمودارهای **sequence**) و المان‌های ساختاری (نمودار **composite structure**) بکار گرفته شود.

نمودار زیر الگوی طراحی Observer را به صورت یک **collaboration** بین شی‌ها که نقش‌آیتم قابل مشاهده و تعدادی شی‌ها که نقش مشاهده‌گر را در آن ایفا می‌کند به تصویر کشیده است.



مدل سازی ارتباط:

نمودار **communication** کمی با نمودارهای **collaboration** که در ویرایش قبلی مورد استفاده قرار می گرفت متفاوت است. می توان گفت که نسخه ی اصلاح شده ی آن محسوب می شود، به گونه ای که تمرکز اصلی در این نمودار بر روی پیوند (**link**) بین اشیا قرار گرفته است.

این **link** در واقع یک پیوند قابل مشاهده و دیداری می باشد که در نمودار **sequence** نمی توان از آن سراغی گرفت. در نمودار **sequence** صرفا می توان پیام هایی که بین اشیا رد و بدل می شود را مشاهده کرد، حتی اگر هیچ پیوندی بین آن ها وجود نداشته باشد.

نمودار **communication** در اصل برای جلوگیری از رخداد این اشتباه (استفاده از نمودار شی توسط مدل ساز به عنوان پایه ی ترسیم مبادله ی پیام ها بدون وجود هیچ گونه پیوند میان اشیا) عرضه شد.

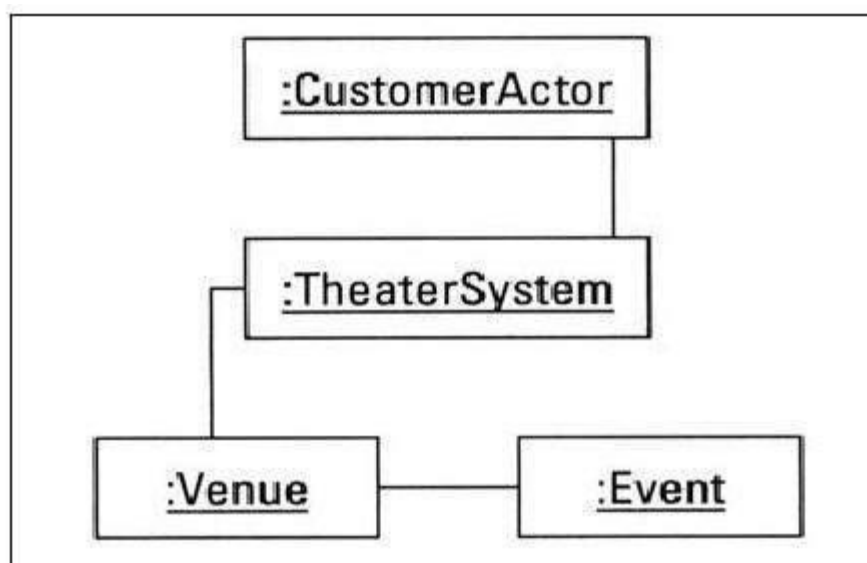
هریک از اشیا موجود در نمودار **communication** را **object lifeline** می نامند.

انواع پیام در نمودار **communication** با انواع پیام در نمودار **sequence** یکسان می باشد. نمودار

communication می تواند پیام های ایجاد شی، گم شده، یافت شده، همزمان، ناهمزمان و بازگشتی را مدل سازی

نماید.

نمودار زیر یک دیاگرام شی را به نمایش می گذارد که از سه شی و دو پیوند (link) تشکیل شده. این نمودار زیرساخت و پایه ی نمودار **communication** را شکل می دهد. همان طور که قبلا ذکر شد، هر شی ای که در نمودار **communication** قابل رویت می باشد، یک **object lifeline** خوانده می شود.



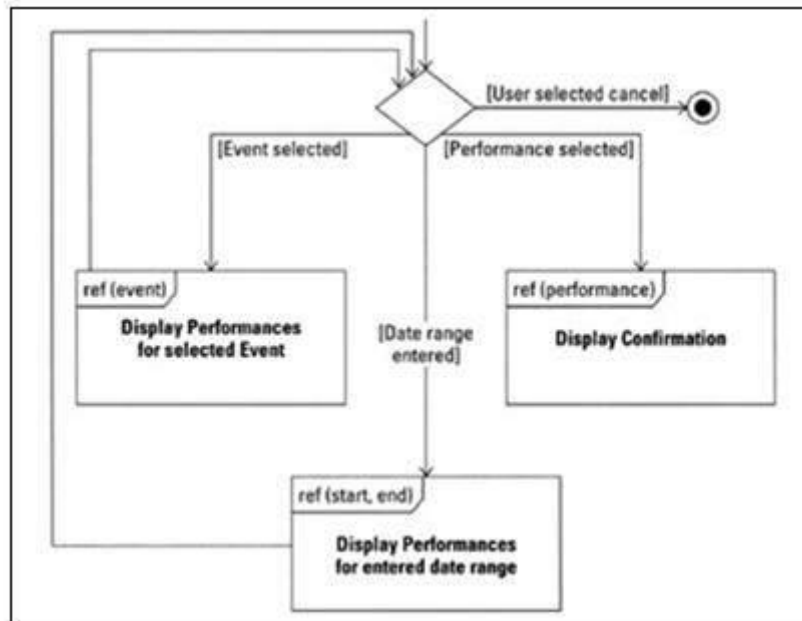
مدل سازی برهمکنش کلی (**interaction overview**):

نمودار **sequence** در اصل برای مدل سازی یک جنبه از برنامه ی کاربردی مورد نظر بکار می رود. براین اینکه بتوان کل یک برنامه را مصور سازی کرد، می بایست از چندین نمودار **sequence** بهره گرفت. از این رو حین مدل سازی یک جنبه از برنامه، ممکن است کل فرایند (تمامی جنبه های برنامه) را فراموش کرده و این منجر به بروز خطاهایی شود.

برای حل این مشکل، نمودار جدید **interaction overview** جریان کنترل را از دیاگرام **activity** و **messaging specification** را از نمودار **sequence** باهم ترکیب می کند.

نمودار **activity** از **activity** ها و جریان اشیا (**object flow**) برای توصیف فرایند استفاده می کند. نمودار **Interaction overview** از برهمکنش ها و نمونه های برهمکنش (**interaction overview**) استفاده می کند. **Lifeline** (اشیا) و **message** ها یافت شده در نمودار **sequence** تنها در **interaction** ها و نمونه های **interaction** یافت می شود. با این حال، **lifeline** هایی که در نمودار **interaction overview** مشارکت دارند، بایستی همراه با اسم نمودار فهرست شود.

دیاگرام زیر نمودار **interaction overview** را با مراحل تصمیم گیری که به صورت لوزی ترسیم شده و همچنین فریم ها و نقطه ی پایان نمایش می دهد.



مدل سازی نمودار زمان بندی (**timing diagram**):

اسم این نمودار مورد کاربرد آن را نیز تشریح می کند. نمودار مزبور زمان رخدادها را در طی چرخه ی حیات آن توصیف می کند.

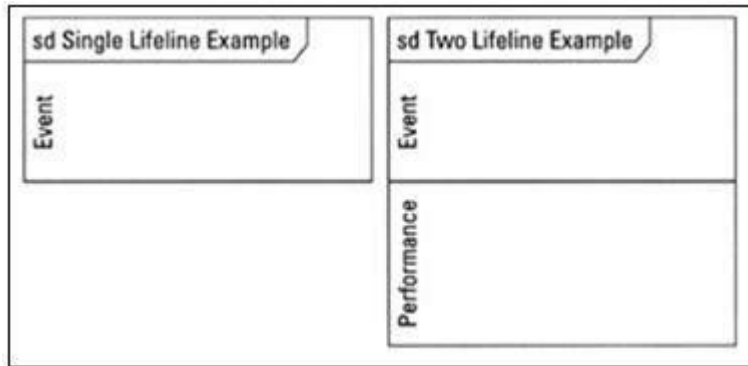
این نمودار تعامل بین اشیا را نشان داده و تاکید آن بر زمان می باشد. نمودار **timing** تلفیقی از دو نمودار **state machine** و **interaction** می باشد.

نمودار **timing** از دو خط زمانی (**timeline**) زیر استفاده می کند:

1. **state time line**

2. **general value time line**

نمودار **timing** یک فضای مستطیل شکل در ناحیه ی محتوا یک کادر یا **frame** ایجاد می کند. این ناحیه ی مستطیل شکل به صورت افقی ترازبندی شده و از راست به چپ خوانده می شود. ممکن است برای نمایش رابطه ی تعاملی بین آن ها، چندین **lifeline** به صورت پشته بر روی هم و در محدوده ی کادر انباشته شوند.



در پایان باید گفت که ویرایش 2.0 ویژگی‌های جدیدی به مجموعه امکانات زبان UML افزوده و بدین وسیله کارایی و موارد استفاده‌ی آن را افزایش داده است. در UML 2.0 دو دسته‌ی کلی وجود دارد: **superstructure** و **infrastructure**. اگرچه نمودارهای نوینی که در این نسخه ارائه شده دارای ویژگی‌ها و قابلیت‌های جدید می‌باشد، اما پایه و اساس آن را همان نمودارهای ورژن قبلی تشکیل می‌دهند.

UML 2.0 چهار نوع نمودار **interaction** ارائه می‌دهد: **sequence**، **communication**، **interaction overview** و در پایان دیاگرام **timing** که اختیاری می‌باشد. هر چهار نمودار مذکور از **frame notation** برای محصور کردن یک **interaction** استفاده می‌کنند. استفاده از **frame**‌ها این امکان را می‌دهد که **interaction**‌ها را مجدداً در قالب نمونه‌های **interaction** (**interaction occurrences**) استفاده کرد.

آموزشگاه کلیکر داده‌ها