فصل ۱۹: طراحی شیء گرا(OOD-Object Oriented Design)

مدل تحلیل ایجاد شده با استفاده از تحلیل شیء گرا را به یک مدل طراحی تبدیل می کند که بـه عنـوان نقشـه راهنمـایی برای ساخت نرم افزار عمل می کنـد. بـا ایـن وجـود، وظیفـه مهنـدس نـرم افـزار مـی توانـد دشـوار باشـد. Gamma [GAM95] و همکاران وی تصویر خوبی از OOD را ارایه می دهند:

طراحی نرم افزارهای شیء گرا دشوار است و طراحی نرم افزار شیء گرا با قابلیت اسـتفاده مجـدد از آن هـم دشـوارتر. بایـد اشیای مرتبط را بیابید، آنها را در کلاس‌هایی مناسب دسته بندی کنید و روابط کلیدی میان آنها را مشـخص کنیـد. طراحـی شما باید مختص مساله مورد نظر باشد، ولی در عین حال آنقدر عمومیت داشته باشد که مسایل و خواسته های آینـده را نیـز پاسخگو باشد. باید از طراحی دوباره پرهیز کنید یا حداقل آن را به کمترین میزان برسانید. طراحان شیء گـرای کـار آزمـوده معتقدند که دستیابی به طراحی انعطاف پذیر و قابل استفاده مجدد، برای بار اول اگر غیر ممکن نباشد، بسـیار دشـوار اسـت. پیش از آن که طراحی پایان یابد، معمولاً سعی می کنند از آن چند بار استفاده به عمل آورند و آن را هر بار اصلاح کنند.

برخلاف مدل‌های طراحی نرم افزار سنتی، OOD منجر به یک طراحی می شود که شامل سـطوح متفـاوتی از پیمانـه هـا است. مولفه های سیستم اصلی به صورت زیرسیستم یا **پیمانه** سطح سیستمی سازماندهی می شـوند. داده هـا و عملیـاتی که داده ها را دستکاری می کنند، در اشیاء- یک شکل پیمانه ای که مولفه سازنده سیستم‌های OO است- **بسته بنـدی** می شوند. به علاوه، OOD باید سازمان داده های مربوط به صفات و جزییات رویه‌ای هر یک از عملیات را توصـیف کنـد. اینها نشانگر مولفه های داده ها و الگوریتمی سیستم OO بوده در پیمانه‌ای کردن سیستم سهم دارد.

طراحی با استفاده از روش‌های شیء گرا

در فصل ۱۳ با مفهوم **هرم طراحی برای نرم افزارهای سنتی** آشنا شـدیم. **چهار لایه طراحی شامل داده هـا، معماری، واسط و سطح مولفه‌ها** تعریف و مورد بحث قرار گرفت. برای سیستم‌های شیء گرا نیز می توان یـک هـرم طراحی تعریف کرد، ولی لایه ها در آن قدری تفاوت دارند. در شکل ۱، چهار لایه هرم طراحی OO عبارتند از:

لایه زیرسیستم: شامل نمایشی از هر یک از زیر سیستم‌هاست که نرم افزار را قادر می سازد تا به خواسـته هـای تعیـین شده توسط مشتری دست پیدا کند و زیر ساخت فنی را که خواسته های مشتری را پشتیبانی می‌کنند، پیاده سازی کنند.

لایه کلاس‌ها و اشیاء: شامل سلسله مراتبی از کلاس‌هاسـت کـه امکـان ایجـاد سیسـتم را بـا اسـتفاده از تعمـیم‌هـا و تخصصی کردن هدفمند فراهم می آورد. این لایه همچنین شامل نمایشی از کلیه اشیا است.

لایه پیغام رسانی: شامل جزییاتی از طراحی است که هر کلاس را قادر به برقراری ارتباط با مشارکت کننده هایش مـی سازد. این لایه واسطهای داخلی و خارجی را برای سیستم برقرار می سازد.

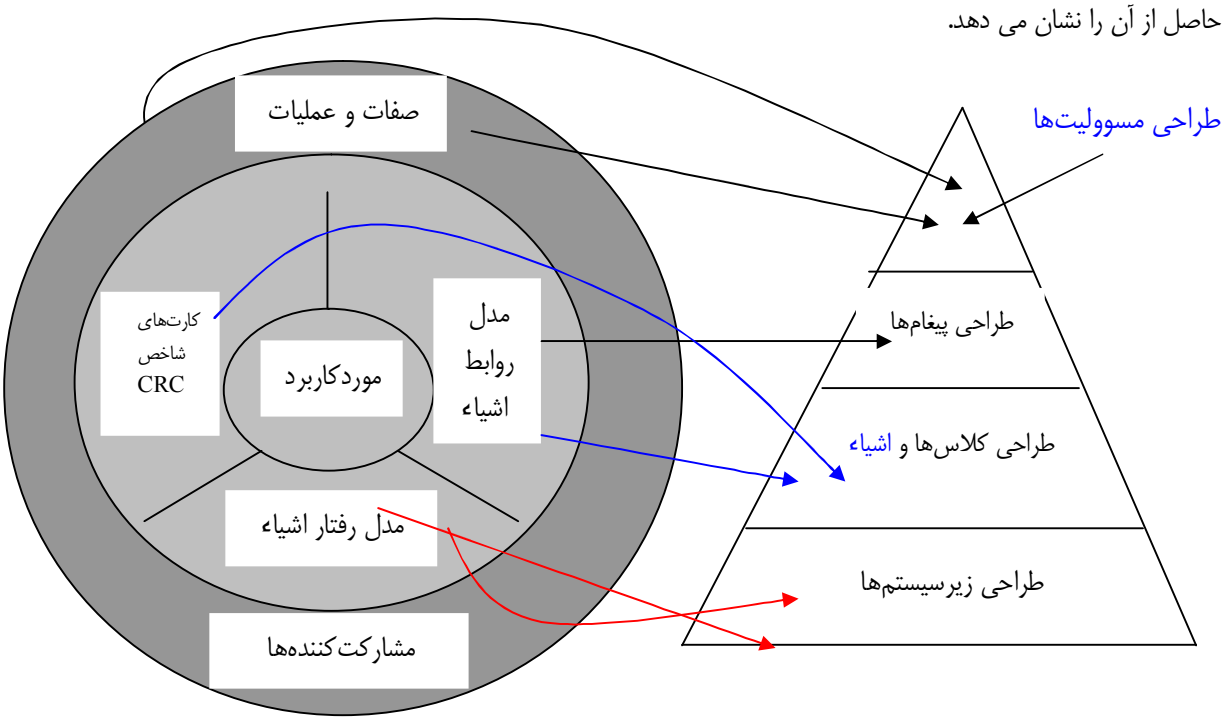
لایه مسوولیت‌ها: شامل طراحی ساختمان داده ها و الگوریتم برای کلیه صفات و عملیات مربوط به هر شیء می‌شود.

هرم طراحی انحصاراً بر طراحی محصول یا سیستمی مشخص تأکید دارد. ولی لازم به ذکر است که لایه دیگری از طراحـی نیز وجود دارد و این **لایه مبنا**یی تشکیل می دهد که هرم بر آن استوار است. لایه مبنا بر **طراحی اشـیای دامنـه** (کـه بعداً در همین فصل **الگوهای طراحی** نامیده خواهند شد) تأکید دارد. اشیای دامنه نقشی کلیدی در ایجاد زیرساختی بـرای سیستم OO ایفا می کنند. زیرا فعالیت‌های واسـط انسـان- کـامپیوتر، مـدیریت وظـایف و مـدیریت داده هـا را پشـتیبانی می‌کنند. اشیای دامنه را می توان برای افزودن اطلاعات بیشتر به خود برنامه کاربردی نیز به کار برد.

تبدیل مدل تحلیل OO به مدل طراحی OO

در OOD نیز همانند طراحی نرم افزار سنتی، هنگامی طراحی داده ها اجرا می شود که صفات موجود باشند؛ طراحی واسط زمانی صورت می پذیرد که مدل پیغام رسانی توسعه یافته باشد و طراحی در سطح مولفه ها (رویه‌ای) برای طراحی عملیات انجام می شود. لازم به ذکر است که معماری یک طراحی OO بیشتر با مشارکت میان اشیاء سر و کار دارد تا با جریان کنترل میان مولفه های سیستم.

گرچه میان مدل‌های طراحی سنتی و OO شباهت وجود دارد، تصمیم گرفتیم لایه های هرم طراحی را تغییر نام دهیم تا ماهیت طراحی OO را به طور صحیحی منعکس کند. شکل ۱ رابطه میان مدل تحلیل OO (فصل ۱۸) و مدل طراحی حاصل از آن را نشان می دهد.



شکل ۱: رابطه بین مدل‌های تحلیل و طراحی شیءگرا

طراحی زیرسیستم‌ها با در نظر گرفتن خواسته های مشتری (که در موارد کاربرد ارایه شدند) و رویدادها و حالت‌هایی به دست می آید که قابل مشاهده هستند(مدل رفتار اشیاء). طراحی کلاس‌ها و اشیاء از توصیف صفات، عملیات و مشارکت‌های موجود در مدل CRC تصویر برداری می شود. طراحی پیغام‌ها توسط مدل روابط میان اشیاء به دست می آید و طراحی مسوولیت‌ها با استفاده از صفات، عملیات و مشارکت‌های شرح داده شده در مدل CRC به دست می آید.

Fishman و Kemerer [FIC92] ده مولفه مدلسازی طراحی را پیشنهاد می کنند که می توان آنها را برای مقایسه روش‌های طراحی شیء گرا و سنتی به کار برد:

۱. نمایش سلسله مراتب پیمانه ها
۲. مشخص سازی تعاریف داده ها
۳. مشخص سازی منطق رویه ای

۴. نشان دادن دنباله ای از پردازش انتها به انتها
۵. نمایش حالت‌ها و گذارهای اشیاء
۶. تعریف کلاس‌ها و سلسله مراتب‌ها
۷. انتساب عملیات به کلاس‌ها
۸. تعریف مشروح کلاس‌ها
۹. تعیین مشخصات اتصالات پیغام رسانی
۱۰. شناسایی سرویس‌های انحصاری

مسایل طراحی

Bertrand و Meyer [MYE90] پنج ملاک برای قضاوت در خصوص توانایی روش‌های طراحی برای دستیابی به پیمانه‌ای بودن پیشنهاد می‌کند و آنها را به طراحی شیء گرا رابط می دهد:

- **تجزیه پذیری:** میزان سهولتی که روش طراحی به طراح کمک می کند تا مساله ای بزرگ را بـه چنـد مسـاله کوچکتر تجزیه کند که راحت تر قابل حل باشند؛
- **ترکیب پذیری.** حدی که روش طراحی اطمینان می دهد تا مولفه های برنامـه (پیمانـه هـا) پـس از طراحـی و ساخته شدن، در ایجاد سیستم‌های دیگر قابل استفاده باشند؛
- **درک پذیری:** سهولت درک یک مولفه از برنامه بدون رجوع به اطلاعات دیگر یا پیمانه های دیگر؛
- **تداوم:** توانایی ایجاد تغییرات کوچک در برنامه، به طوری که این تغییرات خودشان را با تغییرات متنـاظر در یـک یا چند پیمانه نشان دهنده؛
- **محافظت:** خصوصیتی از معماری که انتشار اثرات جانبی حاصل از خطا را در یک پیمانه کاهش می‌دهد.

Meyer [MEY90] پیشنهاد می کند که پنج اصل طراحی زیر را می توان برای معماری پیمانـه ای بـودن بـه دسـت آورد:

۱- واحدهای پیمانه ای بودن زیانی؛
۲- واسطهای معدود؛
۳- واسطهای کوچک (اتصال ضعیف)؛
۴- واسطهای مشخص؛
۵- مخفی سازی اطلاعات(انسجام بالا)

دورنمای OOD

در دهه ۱۹۸۰ و اوایل دهه ۱۹۹۰، گستره وسیعی از روش‌های تحلیل و طراحی OO، پیشنهاد و به کار گرفتـه شـدند. ایـن روش‌ها مولد نشانه گذاری، اصول طراحی و مدل‌های مربوط به OOD نوین شدند. نگـاهی اجمـالی بـه روش‌هـای اولیـه OOD خالی از لطف نبوده و می‌تواند آموزنده نیز باشد:

روش Booch: همان طور که در فصل ۱۸ گفته شد، این روش شامل یک **فرآیند توسعه میکرو** ویک **فرآیند توسعه ماکرو** است. در زمینهٔ طراحی، توسعهٔ ماکرو شامل یک فعالیت طراحـی معمـاری اسـت کـه اشـیای مشـابه را در افرازهـای معماری جداگانه، گروه‌بندی می‌کند؛ اشیاء را از نظر سطح انتزاع، لایه‌بندی می‌کنـد، سـناریوهای مـرتبط را شناسـایی؛ یـک

نمونه اولیه برای طراحی ایجاد و نمونه اولیه را با اعمال آن روی سناریوهای کاربرد، اعتبارسنجی می‌کند. توسعه میکرو، مجموعه‌ای از **قواعد** را تعیین می‌کند که حاکم بر استفاده از عملیات و صفات هستند و سیاست‌های خاص دامنه را برای مدیریت حافظه، کنترل خطا و عملکردهای زیرساختی دیگر وضع می‌کنند؛ سناریوهایی را توسعه می‌دهد که معانی این قواعد و سیاست‌ها را تبیین می‌کند؛ برای هر سیاست یک نمونه اولیه می‌سازد؛ نمونه اولیه را دستکاری و پالایش می‌کند و هر سیاست را مورد بازبینی قرار می‌دهد. به طوری که **تصویر معماری سیستم آشکار گردد**[BOO94].

روش Rambough. تکنیک مدلسازی اشیا (OMT) شامل یک فعالیت طراحی است که اجرای طراحی در سطح انتزاع متفاوت را تشویق می‌کند. طراحی سیستم بر آرایش قطعاتی تأکید دارد که برای ساخت محصول یا سیستم کامل مورد نیاز است. مدل تحلیل، به زیرسیستم‌هایی افراز می‌شود که به پردازنده‌ها و وظایف اختصاص داده می‌شوند. راهبردی برای پیاده‌سازی مدیریت داده‌ها تعریف شده منابع سرتاسری و راهکارهای کنترلی مورد نیاز برای دستیابی به آنها شناسایی می‌شوند.

طراحی اشیاء بر آرایش مشروح یک شیء منفرد تأکید دارد. عملیات از مدل تحلیل انتخاب می‌شوند و برای هر عمل الگوریتم‌هایی تعیین می‌شوند. ساختمان داده‌های مناسب برای صفات و الگوریتم‌ها ارایه می‌شوند. کلاس‌ها و صفات کلاس‌ها به شیوه‌ای طراحی می‌شوند که دستیابی به داده‌ها را بهینه کرده بازدهی کار کامپیوتری را بهبود می‌بخشند. برای پیاده‌سازی روابط میان اشیاء یک مدل پیغام‌رسانی ایجاد می‌شود[RUM91].

روش Jacobson. فعالیت طراحی برای OOSE (مهندسی نرم‌افزار شیءگرا)، نسخه ساده‌ای از یک روش شیءگرایی مقدماتی است که آن را نیز Jacobson ابداع کرده است. مدل طراحی بر قابلیت پیگیری مدل تحلیل OOSE تأکید دارد. نخست، مدل تحلیل ایده‌آلی انتخاب می‌شود که در محیط جهان واقعی بگنجد. سپس اشیای طراحی اصلی، موسوم به بلوک ایجاد می‌شوند و به عنوان بلوک‌های واسط، بلوک‌های موجودیت و بلوک‌های کنترل گروه‌بندی می‌شوند. ارتباط میان بلوک‌ها در حین اجرا تعیین می‌شود و بلوک‌ها به صورت زیر سیستم سازماندهی می‌شود [JAC92].

روش Coad و Yourdon. این روش برای OOD با مطالعه چگونگی انجام کار طراحی توسط **طراحان کارآمد شیءگرا** توسعه یافته است. این روش طراحی نه تنها به کاربرد می‌پردازد، بلکه به زیرساخت کاربر نیز توجه دارد و بر نمایش چهار مولفه اصلی سیستم، یعنی **دامنه مساله، تعامل با انسان، مدیریت وظایف و مدیریت داده‌ها** تأکید دارد[COA91].

روش Wirfs-Brock. این روش طیف پیوسته‌ای از وظایف را تعریف می‌کند که در آن، تحلیل بلافاصله منجر به طراحی می‌شود. پروتکل‌های مربوط به هر کلاس با پالایش پروتکل‌های میان اشیاء ایجاد می‌شوند. هر عمل (مسوولیت) و پروتکل (طراحی واسط) در سطحی از جزییات طراحی می‌شودکه قابل پیاده‌سازی باشند. مشخصه‌های مربوط به هر کلاس (تعریف مسئولیت‌های خصوصی و جزییات مربوط به عملیات) و هر زیرسیستم شناسایی همهٔ کلاس‌های بسته‌بندی شده و تعامل میان زیرسیستم‌ها) تهیه می‌شود[WIR90].

گرچه اصطلاحات و مراحل فرآیند برای هر یک از این روش‌های OOD متفاوت است، ولی کل فرآیند OOD یکی است. مهندس نرم‌افزار برای اجرای مهندسی شیءگرا باید مراحل زیر را اجرا کند:

1. **توصیف کلیه زیرسیستم‌ها و اختصاص دادن آن به پردازنده‌ها و وظایف.**
2. **انتخاب یک راهبرد طراحی برای پیاده‌سازی مدیریت داده‌ها، پشتیبانی واسطها و مدیریت وظایف.**
3. **طراحی یک راهکار کنترلی مناسب برای سیستم.**

۴. **اجرای طراحی اشیاء با ایجاد یک نمایش رویه‌ای برای هر عمل و ساختمان داده‌ها برای صفات کلاس‌ها.**
۵. **اجرای طراحی پیغام‌ها با استفاده از مشارکت میان اشیاء و روابط میان اشیاء.**
۶. **ایجاد مدل طراحی.**
۷. **بازبینی مدل طراحی و تکرار آن تا حد کفایت.**

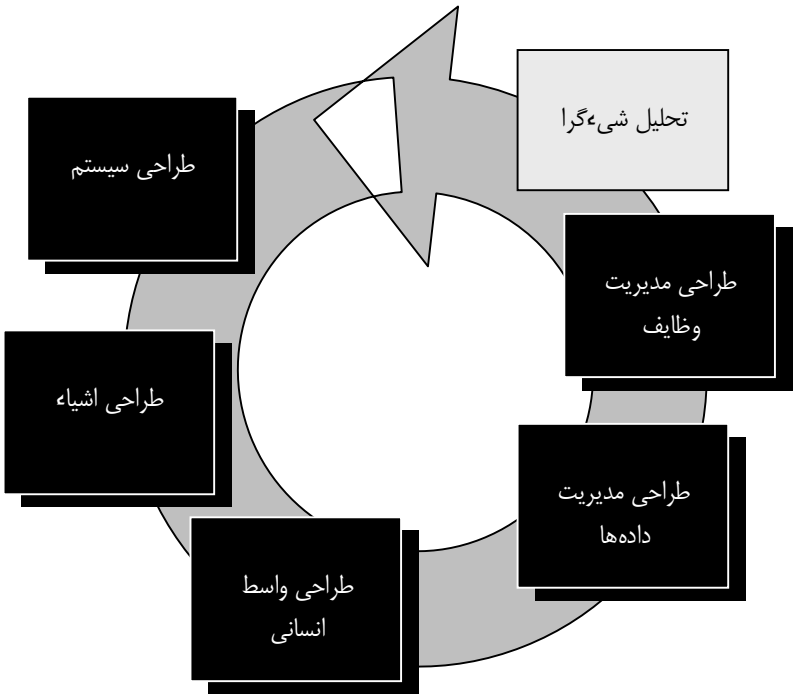
روش یکنواخت برای OOD

Grady، Booch، Rumbough و Jacobson بهترین ویژگی‌های روش‌های تحلیل و طراحی شیءگرای خود را کنار هم قرار دادند و روشی یکنواخت حاصل شد. نتیجه کار که زبان مدلسازی یکنواخت، UML، خوانده می‌شود، در سرتاسر صنعت، کاربردی گسترده یافته است.

در فرایند مدلسازی تحلیل (فصل ۱۸)، دیدگاه‌های مدل کاربر و مدل تحلیل ارایه می‌شود. این دیدگاه‌ها شناختی از سناریوهای استفاده (که راهنمایی برای مدلسازی رفتاری هستند) ارایه داده و با شناسایی و توصیف عناصر ساختاری ایستای سیستم، دیدگاه‌هایی از مدل پیاده‌سازی و مدل رفتاری ارایه می‌کند.

UML در دو فعالیت طراحی عمده سازماندهی می‌شود: **طراحی سیستم و طراحی اشیاء.**

هدف اصلی طراحی سیستم UML، نمایش دادن معماری نرم‌افزار است. جریان فرآیند از تحلیل به طراحی در شکل زیر نیز نشان داده شده است. در سرتاسر فرآیند طراحی UML، دیدگاه مدل کاربر و دیدگاه مدل ساختار، به نمایش طراحی بسط داده می‌شود.



شکل ۲: فریان فرایند طراحی شیءگرا

فرایند طراحی سیستم

فرآیند طراحی سیستم شامل فعالیت‌های زیر می‌شود:

- **افزار مدل تحلیل به زیرسیستم‌ها**
- **شناسایی همزمانی که توسط مساله دیکته می‌شود**
- **تخصیص زیرسیستم‌ها به پردازنده‌ها و وظایف**
- **توسعه یک طراحی برای واسط کاربر**
- **انتخاب یک راهبرد پایه برای پیاده‌سازی مدیریت داده‌ها**
- **شناسایی منابع سرتاسری و راهکاریهای کنترلی مورد نیاز برای دستیابی به آنها**
- **طراحی یک راهکار کنترلی مناسب برای سیستم از جمله مدیریت وظایف**
- **در نظر گرفتن چگونگی پرداختن به شرایط مرزی**
- **بازبینی و در نظر گرفتن مطالعات**

افراز مدل تحلیل

یکی از اصول بنیادی تحلیل (فصل ۱۱)، افراز سیستم است. در طراحی سیستم‌های OO، مدل تحلیل را افراز می‌کنیم تا مجموعه‌های منسجمی از کلاس‌ها، روابط و رفتارها را تعیین کنیم. این عناصر طراحی به صورت یک زیرسیستم بسته‌بندی می‌شوند.

زیرسیستم‌هایی که تعریف (و طراحی) می‌شوند، باید با ملاک‌های زیر نیز مطابقت کنند:

- زیرسیستم باید دارای یک واسط کاملاً تعریف شده باشد که ارتباط با بقیهٔ سیستم، از طریق آن انجام شود.
- به استثنای تعداد کوچکی از ((کلاس‌های ارتباطات))، کلاس‌های درون یک زیرسیستم باید فقط با کلاس‌های موجود در همان زیرسیستم مشارکت کنند.
- تعداد زیرسیستم‌ها نباید زیاد شود.
- زیرسیستم را می‌توان برای کاهش دادن پیچیدگی افراز کرد.

هنگامی که سیستمی به صورت زیرسیستم افراز می‌شود، یک فعالیت طراحی دیگر، موسوم به لایه‌بندی نیز رخ می‌دهد. هر لایه از سیستم OO شامل یک یا چند زیرسیستم است و سطح متفاوتی از انتزاع را برای قابلیت عملیاتی نشان می‌دهد که برای دستیابی به عملکردهای سیستم مورد نیاز است. در اکثر موارد، سطح انتزاع برحسب میزان وابستگی پردازش به زیرسیستمی سنجیده می‌شود که در معرض دید کاربر نهایی قرار دارد.

برای مثال، یک معماری چهار لایه‌ای ممکن است شامل این موارد شود:

۱- **لایه ارایه (زیرسیستم مرتبط با واسط کاربر)؛**
۲- **لایه کاربرد (زیرسیستمی که پردازش مرتبط با کاربرد را انجام می‌دهند)؛**
۳- **لایه قالب‌بندی داده‌ها (زیرسیستمی که داده‌ها را برای پردازش آماده می‌کند)؛**
۴- **لایه بانک اطلاعاتی (زیرسیستم مرتبط با مدیریت داده‌ها).**

هر لایه به طور عمیقتر وارد سیستم می‌شود و پردازشی را نشان می‌دهد که بیشتر خاص محیط است.

همزمانی و تخصیص زیرسیستم‌ها

جنبه پویای مدل رفتار اشیاء، شاخصی از همزمانی میان کلاس‌ها (یا زیرسیستم‌ها) فراهم می‌آورد. اگر کلاس‌ها (یا زیرسیستم‌ها) در یک زمان فعال نباشند، نیازی به پردازش همزمانی نیست. این بدان معنا است که کلاس‌ها (یا زیرسیستم‌ها) را می‌توان روی یک سخت‌افزار مشترک پیاده‌سازی کرد. از طرفی دیگر، اگر کلاس‌ها (یا زیرسیستم‌ها) باید در یک زمان، روی رویدادها عمل کنند، آنها را همزمان در نظر می‌گیرند. هنگامی که زیرسیستم‌ها همزمان هستند، دو گزینه برای تخصیص داریم: ۱. تخصیص هر زیرسیستم به یک پردازنده مستقل یا ۲. تخصیص زیرسیستم‌ها به یک پردازنده مشترک و فراهم آوردن پشتیبانی همزمانی از طریق ویژگی‌های سیستم عامل.

مؤلفه مدیریت وظایف

Coad و Yourdon برای طراحی اشیایی که وظایف جاری را مدیریت می‌کنند، راهبرد زیر را پیشنهاد کرده‌اند:

- **خصوصیات وظیفه تعیین می‌شود.**
- **وظیفه هماهنگ‌سازی و اشیای مرتبط با آن تعریف می‌شود.**
- **هماهنگ‌سازی و دیگر وظایف با یکدیگر مجتمع می‌شوند.**

هنگامی که خصوصیات وظیفه تعیین شد، صفات و عملیاتی از شیء که برای هماهنگی و برقراری ارتباط با وظایف دیگر مورد نیاز هستند، تعیین خواهند شد. الگوی اصلی وظایف(برای یک شیء) به شکل زیر است:

نام وظیفه – نام شیء
توصیف – شرحی که هدف شیء را بیان می‌کند.
اولویت – شامل مثلاً کم، متوسط و زیاد
سرویس – لیستی از عملیات و مسوولیت‌های شیء
هماهنگی – شیوه رفتار شیء
برقراری ارتباط از طریق – مقادیر داده‌های ورودی و خروجی مربوط به وظیفه

مؤلفه واسط کاربر

گرچه مؤلفه واسط کاربر درحیطه دامنه مساله پیاده‌سازی می‌شود، خود واسط، یک زیرسیستم بسیار مهم برای اکثر برنامه‌های کاربردی مدرن به شمار می‌رود. مدل تحلیل OO (فصل ۱۸) شامل سناریوهای استفاده (موارد کاربرد) و شرحی از نقش‌هایی است که کاربر هنگام تعامل با سیستم باید ایفا کند. این‌ها به عنوان ورودی فرآیند طراحی واسط کاربر عمل می‌کنند.

هنگامی که بازیگر و سناریوی آن تعریف شدند، سلسله مراتبی از فرمان‌ها تعیین می‌شود. این سلسله مراتب فرمان‌ها، گروه‌های اصلی منوی سیستم (منوی میله‌ای، یا جعبه ابزار)، و کلیه عملکردهای فرعی را تعریف می‌کنند که از طریق یک گروه منوی سیستم (پنجره‌های منو) در دسترس هستند. سلسله مراتب منوها به طور تکراری مورد پالایش قرار می‌گیرد تا اینکه کلیه موارد کاربرد را بتوان با حرکت در سلسله مراتب عملکردها پیاده‌سازی کرد.

از آنجا که گسترهٔ وسیعی از محیط‌های توسعهٔ واسط‌های کاربری از قبل وجود دارند، نیازی به طرحی عناصر GUI نیست. کلاس‌های قابل استفاده مجدد (با صفات و عملیات مناسب) از قبل برای پنجره‌ها، نمادها(Icons)، عملیات ماوس و گستره وسیعی از امور تعاملی دیگر وجود دارد. برای پیاده‌سازی، فقط کافی است از اشیایی که دارای خصوصیات مناسب هستند، نمونه‌برداری شود.

مؤلفه مدیریت داده‌ها

مدیریت داده‌ها شامل دو زمینه کاری متمایز است:

۱- مدیریت داده‌هایی که در خود برنامه کاربردی اهمیت بحرانی دارند و

۲- خلق زیر ساختی برای ذخیره‌سازی و بازیابی اشیاء.

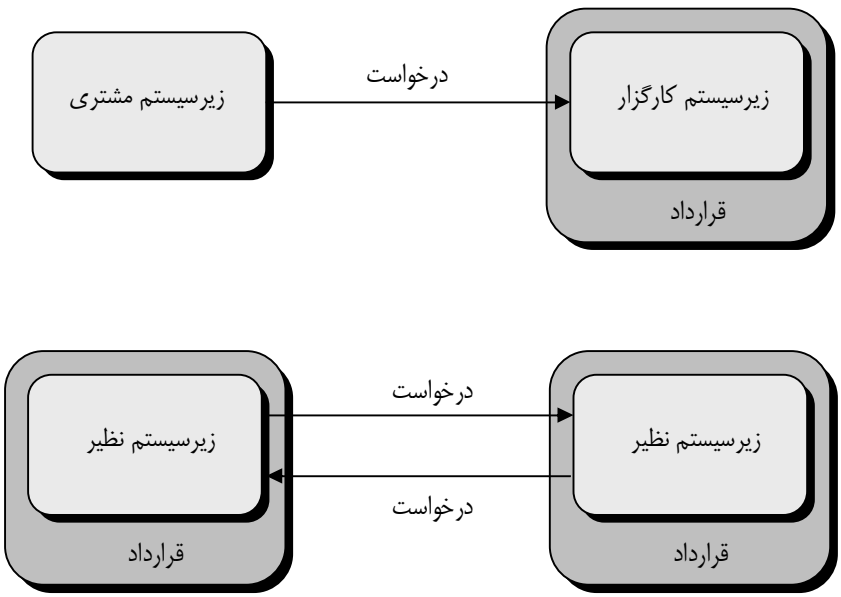
به طور کلی، مدیریت داده‌ها به شیوه‌ای لایه‌ای طراحی می‌شود. ایده اصلی، جداسازی خواسته‌های سطح پایین برای دستکاری ساختمان داده‌ها، از خواسته‌های سطح بالا برای در دست گرفتن صفات سیستم است.

مؤلفه مدیریت منابع

منابع متفاوتی در دسترس یک سیستم یا محصول OO قرار دارند و در بسیاری از موارد، زیرسیستم‌ها بر سر این منابع به رقابت می‌پردازند. منابع سیستمی سرتاسری می‌توانند نهادهای خارجی (مثل گرداننده دیسک، پردازنده، یا خط ارتباطی) یا انتزاعی (مثلاً یک بانک اطلاعاتی یا شیء) باشند. ماهیت منبع هرچه که باشد، مهندس نرم‌افزار باید یک راهکار کنترلی برای آن طراحی کند. Rumbaugh و همکاران وی [RUM91] پیشنهاد می‌کنند که هر منبعی باید به یک شیء نگهبان تعلق داشته باشد. شیء نگهبان، در واقع مراقب و حافظ منبع بوده و دستیابی به آن را کنترل و از تضاد تقاضاها جلوگیری می‌کند.

برقراری ارتباط میان زیرسیستم‌ها

هنگامی که همهٔ زیرسیستم‌ها مشخص شدند، لازم است مشارکت‌های موجود بین آنها نیز تعیین گردد. مدلی که ما برای مشارکت شیء با شیء به کار می‌بریم، به کل زیرسیستم‌ها نیز قابل بسط است. شکل زیر یک مدل مشارکت را نشان می‌دهد.



شکل ۳: مدلی از مشارکت میان زیر سیستم‌ها

فرآیند طراحی اشیاء

سیستم طراحی OO را با توجه به استعاره‌ای که در این متن به کار گرفته شد، می‌توان به عنوان نقشهٔ پلان یک خانه در نظر گرفت. نقشهٔ پلان، هدف از ساخت هر اتاق و ویژگی‌های معماری، اتصال اتاق‌ها به یکدیگر و به محیط خارج را مشخص می‌سازد. اکنون زمان آن فرا رسیده که جزییات مورد نیاز برای ساخت هر اتاق تهیه شود. در حیطهٔ OOD، **طراحی اشیاء** بر **اتاق ها** تأکید دارد.

در این مرحله است که اصول و مفاهیم پایه‌ای مرتبط با طراحی در ساختار مؤلفه‌ها مطرح می‌شوند. ساختمان داده‌های محلی (برای صفات) تعیین و الگوریتم‌ها (برای عملیات) طراحی می‌شوند.

توصیف اشیاء

توصیف طراحی یک شیء (نمونه‌ای از یک کلاس یا زیرکلاس) می‌تواند یکی از دو شکل زیر انجام گیرد[GOL83]:

۱. **توصیف قرارداد** که واسط شیء را با تعریف هر پیغامی که شیء می‌تواند دریافت کند و عملی که شیء به هنگام دریافت آن پیغام اجرا می‌کند، برقرار می‌سازد، یا

۲. **توصیف پیاده‌سازی** مربوط به عمل درخواست شده توسط پیغام را نشان می‌دهد. جزییات پیاده‌سازی شامل اطلاعاتی درباره بخش خصوصی شیء می‌شود، یعنی جزییاتی درباره ساختمان داده‌هایی که صفات شیء و جزییات رویه‌ای توصیفگر عملیات را توصیف می‌کنند.

طراحی الگوریتم‌ها و ساختمان داده‌ها

تنوع نمایش‌های موجود در مدل تحلیل و طراحی سیستم، مشخصه‌ای برای کلیه عملیات و صفات فراهم می‌آورند. الگوریتم‌ها و ساختمان‌های داده‌ای با استفاده از روشی طراحی می‌شوند که قدری با روش‌های طراحی داده‌ها و طراحی در سطح مؤلفه‌های بحث شد، تفاوت دارد.

برای پیاده‌سازی مشخصات مربوط به هر عمل، الگوریتمی ایجاد می‌شود. در بسیاری از موارد، الگوریتم یک دنباله محاسباتی یا رویه‌ای است که می‌توان آن را به صورت یک پیمانه نرم‌افزاری مستقل پیاده کرد. ولی، اگر مشخصات عملی پیچیده باشد، ممکن است نیاز به پیمانه کردن عمل باشد. برای این منظور می‌توان از تکنیک‌های سنتی طراحی در سطح مؤلفه‌ها استفاده کرد.

ساختمان داده‌ها همزمان با الگوریتم‌ها طراحی می‌شوند. از آنجا که عملیات، صفات یک کلاس را دستکاری می‌کنند، طراحی ساختمان داده‌ها که صفات را به خوبی منعکس کند، طراحی الگوریتم‌های عملیات مربوط را بسیار دشوار می‌سازد.

گرچه انواع مختلفی از عملیات وجود دارند، با این وجود می‌توان آنها را به ۳ گروه تقسیم کرد:

۱. عملیاتی که به طریقی داده‌ها را دستکاری می‌کنند (مثل اضافه کردن، حذف کردن، قالب‌بندی، گزینش)؛

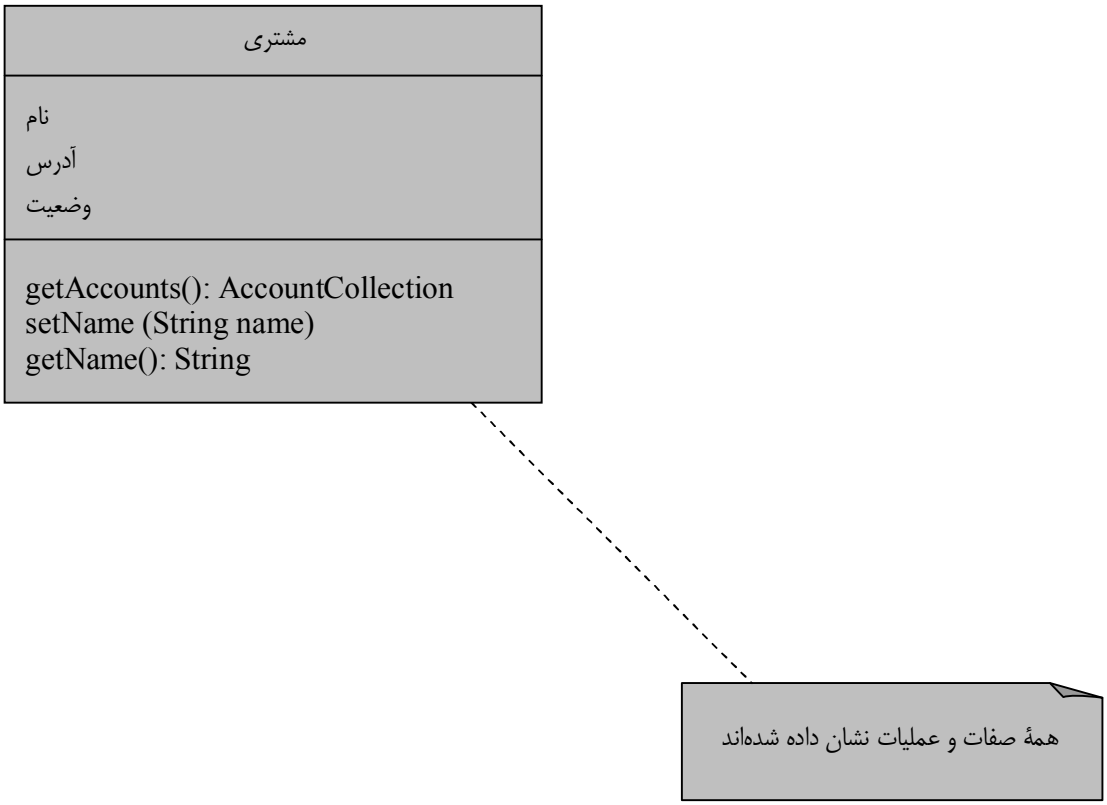
۲. عملیاتی که یک کار محاسباتی انجام می‌دهند؛

۳. عملیاتی که شیء را از لحاظ رخ دادن یک رویداد کنترلی مورد نظارت قرار می‌دهند.

مدل کلاس‌ها

مدل کلاس، توصیفی از کلاس‌ها و روابط میان آنها در یک سیستم است. این مدل، رفتار پویای سیستم، مثلاً رفتار تک تک اشیاء را توصیف نمی‌کند. نخستین عنصر نمودار کلاس‌ها، شرحی از تک تک کلاس‌ها است. شکل ۴ چگونگی توصیف یک کلاس را نشان می‌دهد. این کلاس به مشتریان یک بانک مربوط می‌شود.

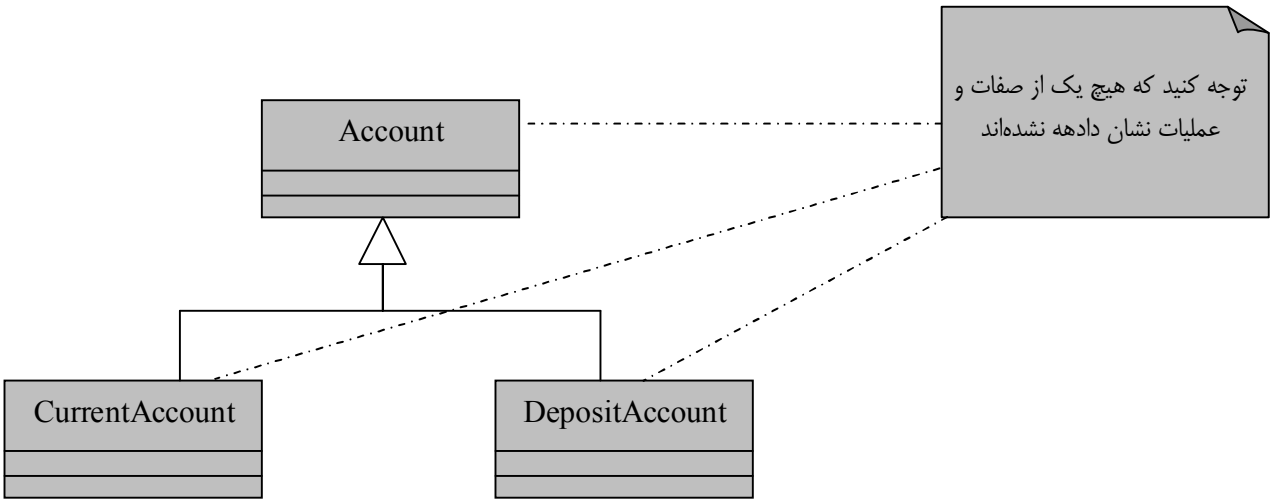
این شکل بسیار ساده است، زیرا تنها یک کلاس دارد. این مدل شامل نام کلاس (Customer)، نام برخی صفات آن (مثلاً صفت address یعنی نشانی مشتری است) و لیستی از عملیات (مثلاً getName، نام مشتری را برمی‌گرداند) است. پس هر مستطیلی که نشانگر یک کلاس است شامل قسمتی برای نام کلاس، قسمتی برای صفات اشیای تعریف شده توسط آن کلاس و قسمتی برای لیست عملیات مرتبط با این شیء است.



شکل ۴: مثالی از بخش از یک کلاس توصیف شده در UML

تعمیم

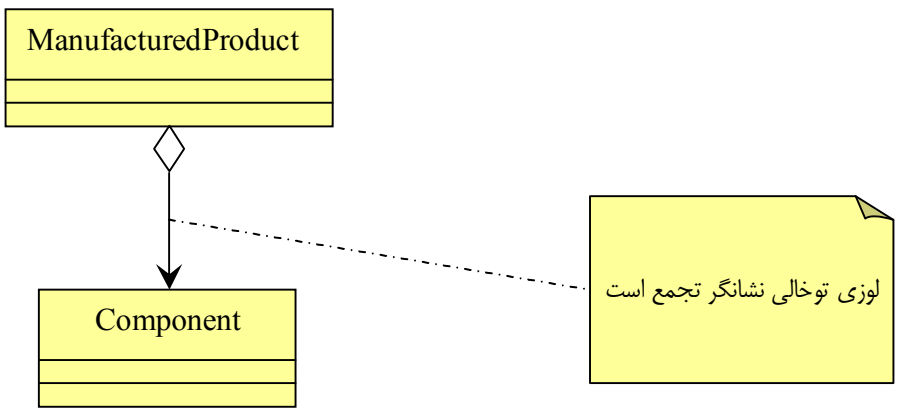
این رابطه میان کلاس X و کلاس Y زمانی برقرار است که کلاس Y نمونه خاص از کلاس X باشد. برای مثال، بین کلاس Account که نشانگر یک حساب بانکی عمومی است و یک حساب جاری Current Account که نمونه خاصی از یک حساب است، رابطه **تعمیم** وجود دارد. شکل ۵ چگونگی نمایش این رابطه را در یک نمودار کلاس‌های UML نشان می‌دهد.



شکل ۵: مثالی از یک تعمیم در UML.

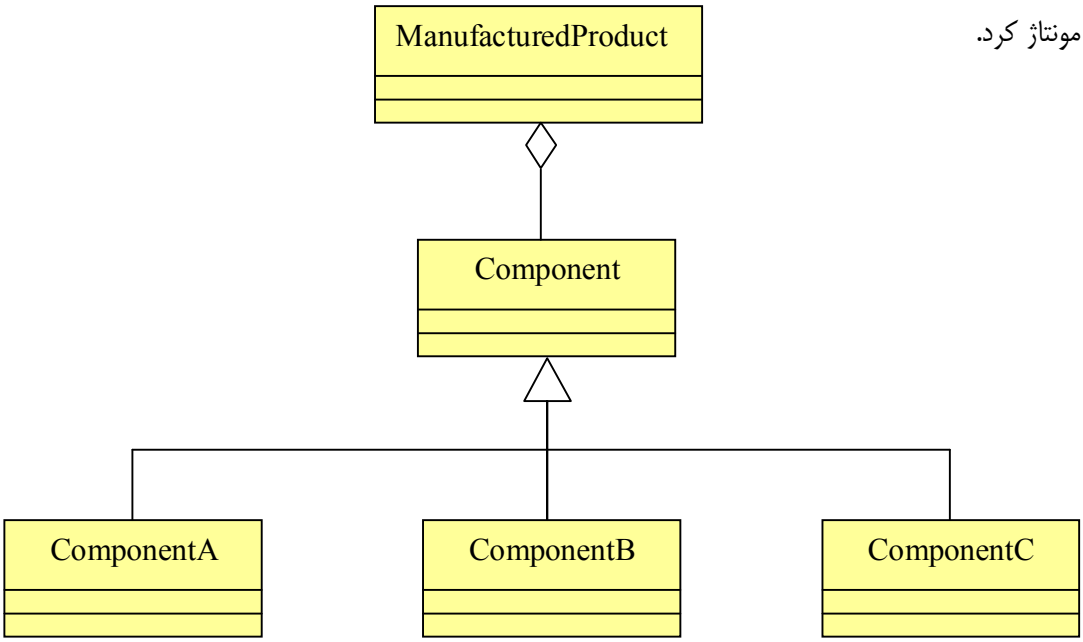
تجمع و ترکیب

روابط مهم دیگر عبارتند از **تجمع** و **ترکیب**. دو رابطه وجود دارند که نشان می‌دهند یک کلاس اشیایی تولید می‌کند که بخشی از یک شیء است که توسط کلاس دیگری تعریف شده است. برای مثال، سیستمی برای یک تولیدکننده باید داده‌های مربوط به اقلامی را نگهداری کند که تولید می‌شوند و اینکه از چه مواردی ساخته می‌شوند. مثلاً یک کامپیوتر از قطعاتی مثل دستگاه اصلی، حافظه جانبی، کارت‌های حافظه و غیره ساخته می‌شود. کامپیوتر از قطعات تشکیل می‌شود و در یک سیستم شیءگرا که برای پشتیبانی تولید به کار برده می‌شود، یک رابطهٔ تجمعی میان کلاس توصیف کننده محصول تولید شده و هر یک از قطعات آن وجود دارد. بنابراین می‌گوییم یک رابطه تجمع وجود دارد. شکل ۶ چگونگی نمایش این رابطه **تجمع** را در نمودار کلاس UML نشان می‌دهد.



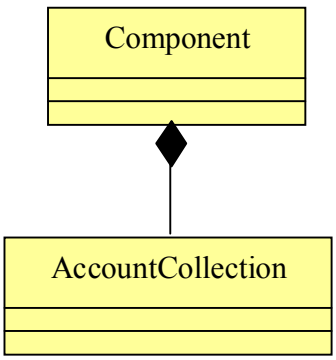
شکل ۶: رابطه تجمع در UML

در اینجا،خطی که لوزی توخالی به آن متصل است، نشان می‌دهد که کلاس، اشیایی را توصیف می‌کند که اشیای دیگر را با هم مجتمع می‌کنند. کلاسی که لوزی به آن متصل است، اشیایی را توصیف می‌کند که شامل اشیایی است که توسط کلاس‌هایی دیگر تعریف شده‌اند. در UML، روابط معمولاً به هم آمیخته‌اند. برای مثال در شکل ۱۲-۲۲، چند مؤلفه وجود دارند که با کلاس Component رابطه تعمیم دارند (شکل ۱۳-۲۲). دراین شکل Cmponent با چند کلاس اختصاصی‌تر مرتبط است که توصیفگر قطعاتی هستند که محصول رامی‌توان از آنها مونتاژ کرد.



شکل ۷: نمودار کلاس‌های UML که تعمیم و تجمع را نشان می‌دهد.

شکل خاصی از **تجمع** هست که به آن **ترکیب** گفته می‌شود. این رابطه زمانی استفاده می شود که یک شیء شامل چند شیء دیگر باشد و وقتی که شیء اصلی حذف شود، همهٔ اشیای موجود در آن نیز از بین می‌روند. برای مثال کلاس Customer که نشانگر مشتریان بانک است، با حساب‌های مشتری رابطه‌ای ترکیبی دارد، زیرا اگر مشتری حذف شود، همهٔ حساب‌های وی نیز حذف خواهد شد. این رابطه مشابه با رابطه تجمیع نشان داده می‌شود، ولی با یک لوزی توپر (شکل ۸)



شکل ۸: نمودار کلاس UML که ترکیب را نشان می‌دهد

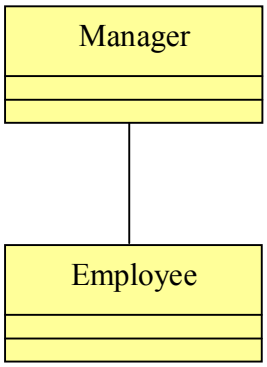
همبستگی‌ها

تجمع و ترکیب، مثال‌های خاصی از رابطه میان دو کلاس هستند. وقتی یک رابطه میان دو کلاس برقرار می‌شود که بین آن دو اتصالی برقرار باشد؛ این اتصال در UML به عنوان همبستگی شناخته می‌شود. برخی از مثال‌های همبستگی عبارتند از:

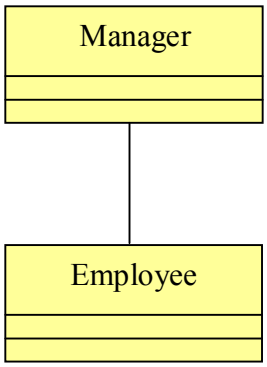
- کلاس Manager (مدیر) با کلاس Employee (کارمند) رابطه دارد، زیرا یک مدیر چند کارمند را مدیریت می‌کند.
- کلاس Flight (پرواز) با کلاس Plane (هواپیما) بستگی دارد، زیرا هواپیما یک پرواز خاص را به انجام می‌رساند.
- کلاس Computer با کلاس Message (پیغام) رابطه دارد، زیرا مجموعه‌ای از پیغام‌ها منتظرند تا توسط کامپیوتر پردازش شوند.
- کلاس BankStatement (صورت حساب) با کلاس Transaction (تراکنش) رابطه دارد، زیرا صورت حساب شامل جزییات هر تراکنش است.

از میان این روابط، فقط آخری رابطه تجمعی است. همهٔ روابط دیگر، همبستگی‌های ساده‌اند. این همبستگی‌ها در UML به صورت یک خط راست نشان داده می‌شوند. برای مثال، در شکل ۹ نخستین همبستگی نشان داده شده است.

همبستگی میان کلاس‌ها برحسب چندگانگی همبستگی و نام همبستگی نیز مستندسازی می‌شود. با در نظر گرفتن مثال‌های نشان داده شده در شکل ۹، نگاهی به چندگانگی خواهیم داشت.

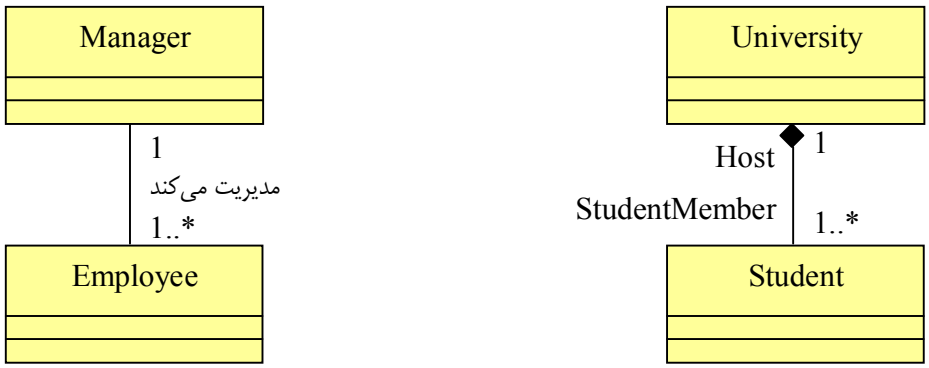


نمونه‌ای از رابط هواپیما در UML.



تعدد در نمودار کلاس UML.

شکل ۹: رابطه‌های همبستگی



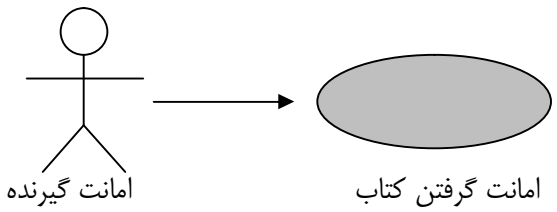
مستندسازی همبستگی.

مستندسازی نقش

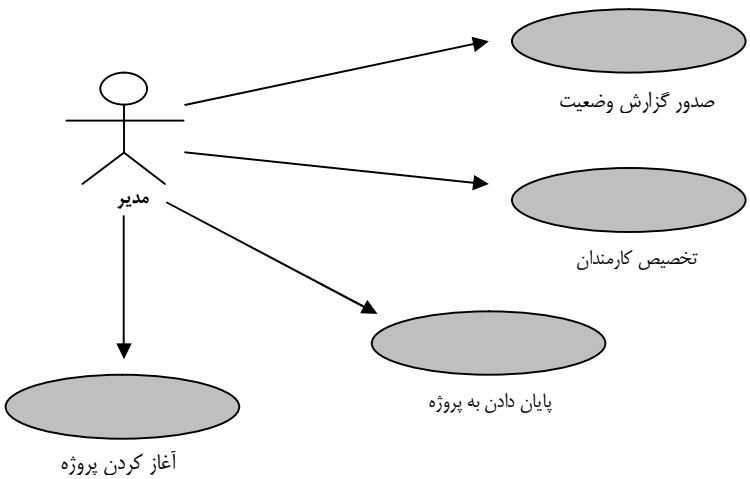
شکل ۱۰: رابطه‌های همبستگی با تعیین چندی

موارد کاربرد

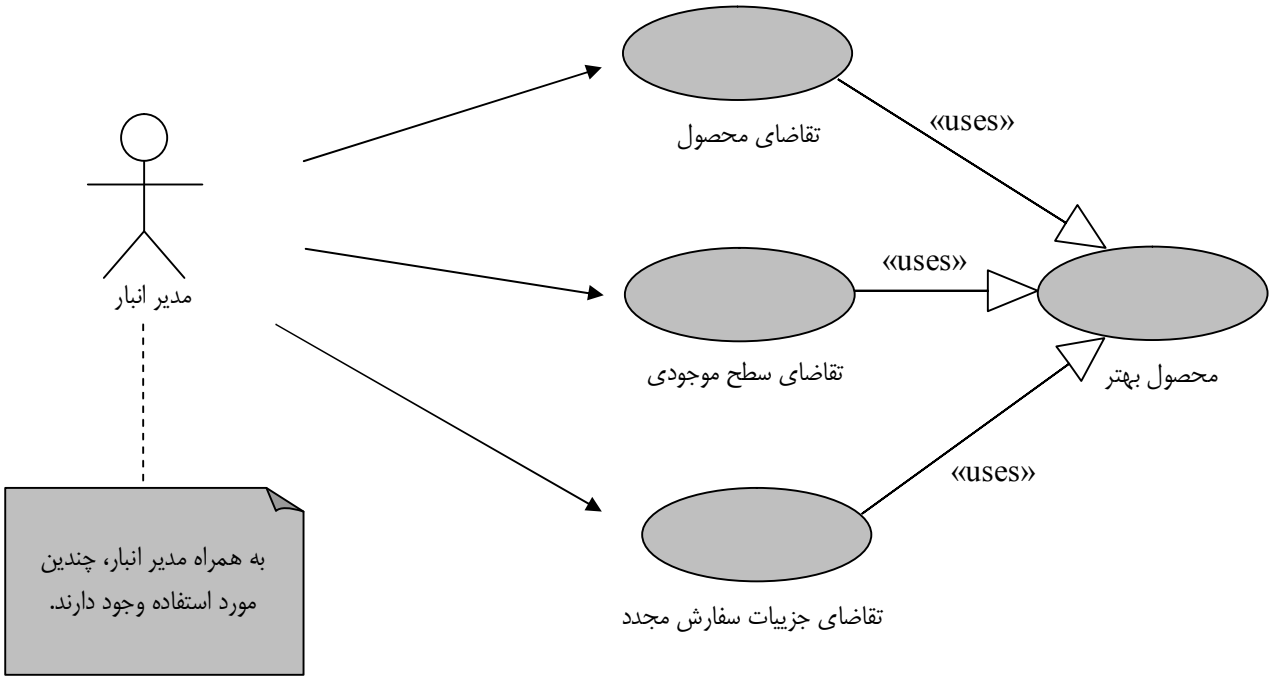
در فصل ۱۸ موارد کاربرد را مورد بحث قرار دادیم. در UML، مورد کاربرد بسیار ساده، برحسب بازیگر (Actor) و یک مورد کاربرد مستندسازی می‌شود. بازیگر، عاملی است که با سیستمی که در حال ساخته‌شدن است، تعامل می‌کند؛ مثلاً خلبان یک هواپیما، کسی که از کتابخانه کتاب به امانت می‌گیرد یا مدیر چند کارمند در یک شرکت. هر مورد کاربرد، عملی را که بازیگر انجام می‌دهد مستندسازی می‌کند؛ مثل تغییر دادن جهت حرکت هواپیما، امانت گرفتن کتاب از کتابخانه یا افزودن یک عضو جدید به تیم برنامه‌نویسی. نمونه‌هایی از مورد کاربردها در شکل‌های ۱۱ تا ۱۳ نشان داده شده است. این مثال، کاربری را نشان می دهد که در حال امانت گرفتن یک کتاب از کتابخانه است.



شکل ۱۱: یک مورد کاربرد ساده



شکل ۱۲: چند مورد کاربرد



شکل ۱۳: مثالی از موارد کاربرد دیگر

مشارکت‌ها

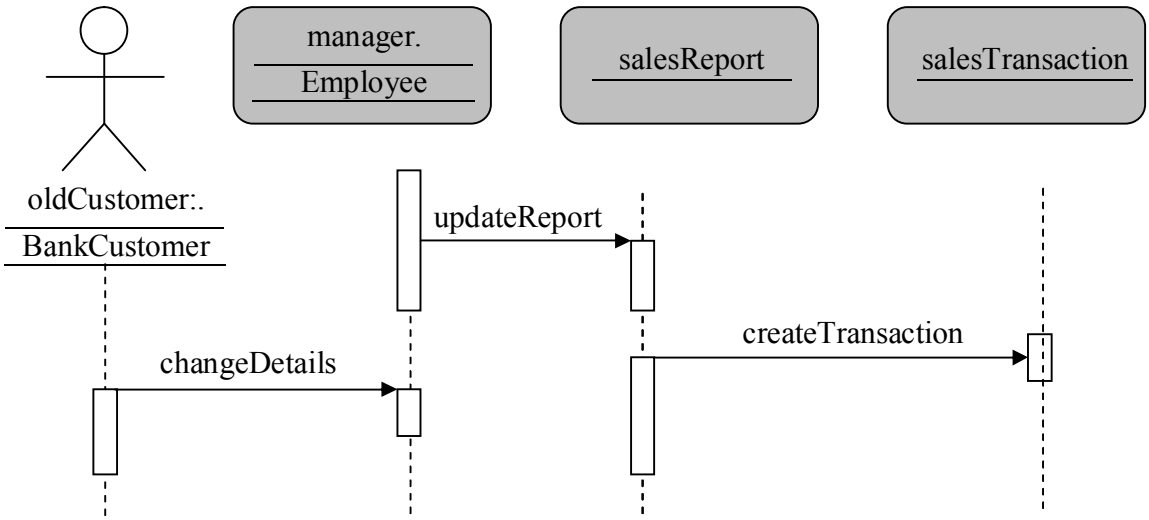
در فرایند اجرای یک سیستم شیءگرا، اشیای سیستم با یکدیگر تعامل می‌کنند. برای مثال، در یک سیستم بانکداری، شیء Account ممکن است پیغامی به یک شیء تعاملی ارسال نماید تا تراکنشی را ایجاد کند که در آن حساب رخ داده است. برای مثال، از حساب برداشت شده است. این نوع اطلاعات برای طراحی یک سیستم شیءگرا در حین فرآیند شناسایی و اعتبارسنجی کلاس‌ها مهم است. از این رو، UML دو نشانه‌گذاری متفاوت برای تعریف تعامل‌ها دارد. **نمودار توالی** (Sequence Diagram) و نمودار دیگری که به عنوان **نمودار مشارکت**(Collaboration Diagram) شناخته می‌شود و هم ارز نمودار توالی است؛ در واقع، آنها چنان شبیه یکدیگرند که ابزارهای Case غالباً می‌توانند یک نمودار را از روی دیگری ایجاد کنند. شکل‌های ۱۴ و ۱۵ نمونه‌هایی ساده از نمودار توالی را نشان می‌دهند.

در نمودار ۱۴، سه شیء وجود دارد که در یک تعامل شرکت دارند. اولی، شیء manager است که توسط کلاس Employee توصیف می‌شود. این شیء یک پیغام updateReport را به شیء salesReport ارسال می‌کند که سپس آن شیء پیغام‌های CreatTransaction را به شیء دیگری به نام salesTransaction ارسال می‌کند. در نمودار توالی، سه شیء موجودند که یکی از آنها (manager) دارای کلاس مشخص خود(Employee) است ولی بقیه خیر.

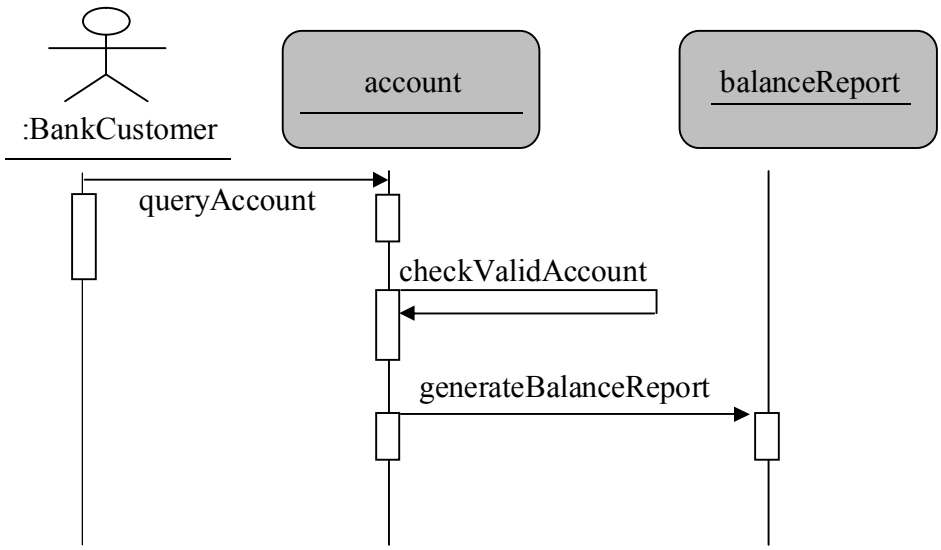
محتویات یکی از کادرهای موجود در نمودار توالی می‌تواند فقط شامل نام یک شیء به همراه نام کلاس آن که توسط دو نقطه (:) از هم جدا شده یا تنها نام کلاس و قبل از آن دو نقطه (:) باشد؛ در موردآخر، شیء بدون نام است.

شکل ۱۴ نقش یک بازیگر را نیز در مشارکت نشان می‌دهد؛ در اینجا، بازیگر BankCustomer با ارسال پیغام changeDetails با مدیر شیء Employee تعامل می‌کند.

شکل ۱۵ مثال دیگری از یک نمودار توالی را نشان می‌دهد. در اینجا، بازیگری که توسط شیء بدون نام تعریف شده است به وسیله کلاس BankCustomer نمایش داده می‌شود، به شیء account پیغامی ارسال می‌کند که حساب را تقاضا می‌کند. این شیء چک می‌کندکه آیا حساب معتبر است و سپس پیغام generateBalanceReport به شیء balanceReport ارسال می‌کند که شامل داده‌هایی است که مشتری بانک درخواست کرده است.



شکل ۱۴: نمودار توالی ساده



شکل ۱۵: مثال دیگری از نمودار توالی

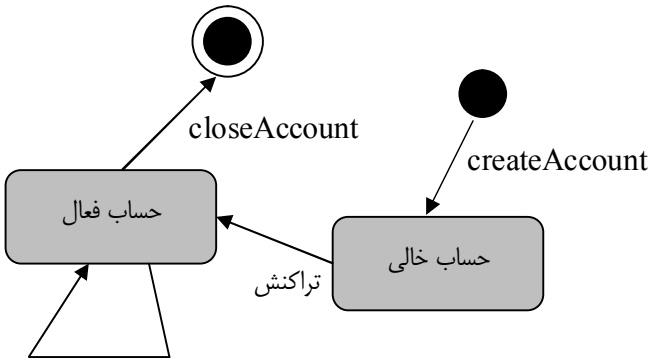
نمودارهای حالت

یکی دیگر از اجزای مهم UML، نمودار حالت است. این نمودار، حالت‌های گوناگونی را نشان می‌دهد که شیء می‌تواند آن حالت‌ها را دارا باشد و نشان می‌دهد که چگونه هر حالت به حالت دیگر گذار می‌کند. چنین نموداری شامل چند مؤلفه است:

- **حالت‌ها که به صورت کادرهایی با گوشه‌های گرد نشان داده می‌شوند.**
- **گذارهای میان حالت‌ها که به صورت خطوط پیکان‌دار نشان داده می‌شوند.**
- **رویدادها که باعث گذار میان حالت‌ها می‌شوند.**
- **علامت شروع که حالت اولیهٔ شیء را به هنگام ایجاد نشان می‌دهد.**
- **علامت توقف که نشان می‌دهد شیئی به پایان حیات خود رسیده است.**

نمونه‌ای از نمودار حالت در شکل ۱۶ نشان داده شده است.

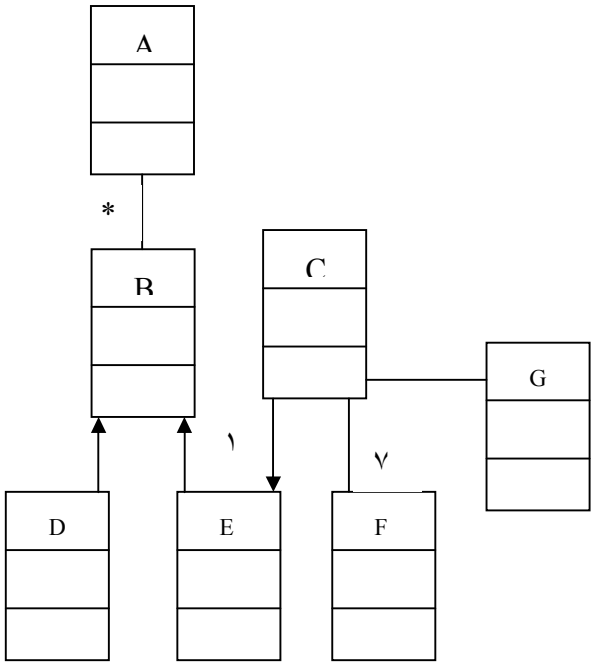
در اینجا چرخه حیات یک حساب بانکی نشان داده شده است. هنگامی که حساب ایجاد شد، به عنوان یک حساب خالی در نظر گرفته می‌شود. به محض آنکه یک تراکنش روی حساب رخ داد، حساب، فعال در نظر گرفته می‌شود. نمودار حالت نشان می‌دهد که وقتی حساب بسته شد، باید از بین برود.



شکل ۱۶: نمونه‌ای از نمودار حالت

تست‌های فصل ۱۹: طراحی شیءگرا

۱- با توجه به نمودار کلاس زیر گزینه صحیح را انتخاب کنید.



الف) E بخشی از C می‌باشد. E نوعی از B می‌باشد. یک شی از نوع G با چند شی از نوع C رابطه دارد.
ب) C بخشی از E می‌باشد. D نوعی از B می‌باشد. یک شی از نوع C با یک تا هفت نوع از شی F رابطه دارد.
ج) E بخشی از C می‌باشد. E نوعی از B می‌باشد. یک شی از نوع C با یک تا هفت نوع از شی F رابطه دارد.
د) C بخشی از E می‌باشد. B نوعی از D می‌باشد. یک شی از نوع C با یک تا هفت نوع از شی F رابطه دارد.

۲- چهار لایه تعریف شده برای طراحی شیءگرا مشابه لایه‌های تعریف شده در طراحی نرم‌افزار سنتی است.

الف) درست ب) غلط

۳- کدامیک از گزینه‌های زیر در مدل‌های طراحی شیءگرا وجود دارند ولی در مدل‌های طراحی سنتی نیستند؟

الف) نمایش سلسله مراتب پیمانه‌ها ب) مشخصات تعاریف داده‌ها
ج) مشخصات ارتباط پیغام‌ها د) مشخصات منطق رویه‌ای

۴- در طراحی شیء گرا به اتصال پایین پیمانه‌ها دست می‌یابیم که این امر شامل پنهان‌سازی اطلاعات بهتر نسبت به سایر روش‌ها است.

الف) درست ب) نادرست

۵- مراحل عمومی یکسانی برای طراحی شیءگرا بکار گرفته می‌شود بجز اینکه روش طراحی خاصی انتخاب می‌گردد.

الف) درست ب) نادرست

۶- نگرش UML به طراحی شیءگرا دارای دو فعالیت عمده است. این فعالیت‌ها کدامند؟

الف) طراحی معماری و طراحی اشیاء ب) طراحی واسط و طراحی پیغام
ج) طراحی واسط و طراحی سیستم د) طراحی سیستم و طراحی اشیاء

۷- کدامیک از فعالیت‌های زیر بخشی از فعالیت طراحی سیستم با نگرش UML به OOD است؟

الف) انتخاب استراتژی برای مدیریت داده‌ها ب) افراز مدل تحلیل به زیرسیستم‌ها

ج) طراحی واسط کاربر د) تمام موارد فوق

۸- اولین مرحله از طراحی سیستم در OOD، افراز مدل تحلیل به مجموعه‌ای از کلاس‌ها، روابط بین آنها، و رفتارها است. این کار را ..... می‌نامند؟

الف) سلسله مراتبی کلاس‌ها ب) ارتباطهای مشتری/ کارگزار

ج) زیرسیستم‌ها د) لایه‌های سیستمی

۹- وقتی زیرسیستم‌ها همروند (همزمان) هستند باید آنها را به پردازنده‌های مجزایی اختصاص داد.

الف) درست ب) نادرست

۱۰- واسط‌های کاربران معمولاً با استفاده از ابزارهای اتوماتیک ساخته می‌شود که شامل کلاس‌های قابل استفادهٔ مجدد است به طوری‌که پیاده‌ساز فقط باید اشیاء مناسب با محدودهٔ مسأله را در آن تعریف کند.

الف) درست ب) نادرست

۱۱- کدامیک از زمینه‌های زیر به عنوان بخشی از مدیریت داده مولفه‌های طراحی سیستم OOD است؟

الف) ایجاد زیرساختاری برای ذخیره و بازیابی اشیاء.

ب) مدیریت داده‌هایی که برای نرم‌افزار کاربردی حیاتی و حساس هستند.

ج) نرمال‌سازی صفات کلاس‌های داده.

د) موارد الف و ب

۱۲- هر قراردادی بین زیرسیستم‌ها دقیقاً از طریق یک پیغام که بین اشیاء زیر سیستم قرار دارد ایجاد می‌گردد.

الف) درست ب) نادرست

۱۳- طراح توصیف یک شیء یکی از دو صورت زیر می‌باشد:

الف) الگوی شیء یا شبه کد ب) توالی اپراتور یا گراف‌های صفات

ج) توصیف پروتکل و یا توصیف شیء د) گراف همکاری زیرسیستم یا گراف پروتکل

۱۴- در OOD عملیات با اعمال زیر پالایش می‌شوند.

الف) ایزوله کردن عملیات جدید در پایین‌ترین سطح انتزاع ب) تهیه تجزیه گرامری

ج) نوشتن خلاصه فرآیند د) تمام موارد

۱۵- طراحی الگوها برای طراحی نرم‌افزار شیءگرا قابل استفاده و بکارگیری نیست؟

الف) درست ب) نادرست

۱۶-طراحی‌های شیءگرا نیازی به استفاده از تکنیک‌های برنامه‌سازی شیءگرا در پیاده‌سازی ندارند.

الف) درست ب) نادرست