مهندسی نرم افزار

فهرست مطالب

مهندسی نرم افزار..... ۱

فصل اول مقدمه‌ای برای مهندسی نرم‌افزار..... ۴

مقدمه..... ۴

تعریف چند مفهوم..... ۴

نرم‌افزار چیست؟..... ۴

مهندس نرم‌افزار کیست؟..... ۵

مهندسی نرم‌افزار چیست؟..... ۵

مشکلات کنونی نرم‌افزار..... ۵

ویژگی‌های نرم‌افزار..... ۶

کاربردهای نرم‌افزار..... ۸

کاربرد های نرم افزار..... ۸

۱ـ نرم‌افزارهای سیستمی (System Software):..... ۸

۲ـ نرم‌افزارهای کاربردی (Application Software):..... ۸

۳ـ نرم‌افزارهای علمی / مهندسی (Engineering/Scientific software):..... ۸

۴ـ نرم‌افزارهای توکار ـ تعبیه شده (Embedded Software):..... ۸

فصل دوم مهندسی سیستم و مهندسی نیازمندی‌ها..... ۹

بخش اول: مهندسی سیستم..... ۹

۱ـ سیستم چیست؟..... ۹

۲ـ سیستم کامپیوتری چیست؟..... ۱۰

۳ـ مهندسی سیستم چیست؟..... ۱۰
فصل چهارم مفاهیم شیء گرایی و UML..... ۱۱
بخش دوم: مدل سازی با UML..... ۱۱
ویژگی‌های UML..... ۱۱
تاریخچه‌ی UML..... ۱۲
نمودارهای UML..... ۱۳
شمول..... ۲۷
توسعه..... ۲۸

فصل اول

مقدمه‌ای برای مهندسی نرم‌افزار

مقدمه

دربخش اول این فصل یک سری مفاهیم درارتباط با ماهیت نرم‌افزار بیان می‌شود. سپس مشکلات کنونی نرم‌افزار و ویژگی‌ها و کاربردهای آن در پایان همین بخش مورد بررسی قرار می‌گیرند.دربخش دوم این فصل،لایه‌های مهندسی نرم‌افزار وارتباط آن‌ها با متدولوژی توسعه نرم‌افزار بیان می‌شود، سپس به چاچوب فرآیند و فعالیت‌های چتری نرم‌افزار می‌پردازیم. در بخش سوم که مهم‌ترین قسمت این فصل نیز هست مدل‌های فرآیندی مختلف را ارزیابی کرده و ویژگی‌ها، نقاط ضعف و قوت هر یک از این مدل‌ها را مورد بررسی قرار می‌دهیم. در قسمت آخر این فصل که مربوط به بهبود فرآیند نرم‌افزار می‌باشد، در مورد مفاهیم مربوط به آن و مدل‌های مختلف م ورد استفاده برای بهبود فرآیند نرم‌افزاری به منظور تضمین کیفیت فرآیند نرم‌افزار را مورد بحث رار می‌دهیم.

تعریف چند مفهوم

قبل ازاینکه به بحث درارتباط با مهندسی نرم افزار بپردازیم خوب است که مفهوم کلیدی را که در مهندسی نرم افزار نقش اساسی دارند بهتر بشناسیم برای شناخت بهترهرفرآیند مهندسی ابتدا باید محصولی را که قراراست طی فرآیند مهندسی به دست بشناسیم ، سپس فرد و یا افرادی را که دربه دست آمدن آن محصول تحت یک چارچوب مشخص فعالیت می کنند ، بشناسیم در پایان باید در مورد خود فرآیند مهندسی نرم افزار که منجر به ولید آن محصول می شود ، اطلاعات کامل را کسب کنیم.

نرم‌افزار چیست؟

نرم‌افزار،دستورالعمل‌ها بابرنامه کامپیوتری می باشدکه درصورتی که به درستی اجراشود می‌تواندخصوصیت‌ها، عملکردهاوکارآیی مطلوب و خواسته شده از خودش را به شکل مطلوب در اختیار ما قرار بدهد. علاوه بر آن نرم‌افزار شامل ساختمان داده‌اند می‌باشد که از طریق آن برنامه‌ها قادربه درستکاری ومدیریت اطلاعات به شکل مطلوب ومورد نیازمی‌باشند. همچنین همواره به همراه یک نرم‌افزار مستنداتی وجود دارند که در مورد عملکرد و نحوه استفاده از برنامه‌هایی که باید اجرا شوند توضیحات مناسبی در اختیار کاربران قرار می‌دهند.

مهندس نرم‌افزار کیست؟

مهندس نرم‌افزار شخصی است که نرم‌افزاری راتولید و پشتیبانی می‌کند تا کاربران نرم‌افزار قادر به استفاده از آن به شکل مطلوب باشند. نرم‌افزار تولید شده باید مطابق با نیازهای کاربران ساخته شود و بتواند نیازهای آن‌ها را برطرف کند.

مهندسی نرم‌افزار چیست؟

به مجموعه روش‌ها، فن‌آوری‌ها و ابزارهایی که در فرآیندهای سازگارپذیر و چابک برای رسیدن به محصولی با کیفیت بالا در تولید نرم‌افزار مورد استفاده قرار می‌گیرند و مبتنی بر اصول مهندسی می‌باشند، مهندسی نرم‌افزار می‌گویند.

نرم‌افزار هم می‌تواند محصول باشد و هم وسیله‌ای برای تحویل محصول (اطلاعات). برخلاف بسیاری از افراد که مهندسی نرم‌افزار را همان برنامه‌نویسی کامپیوتر می‌دانند، باید متذکر شویم که برنامه‌نویسی تنها بخش کوچکی از فرآیند مهندسی نرم‌افزار می‌باشد.

مشکلات کنونی نرم‌افزار

از بدو پدید آمدن نرم‌افزار تاکنون، همواره نرم‌افزار با مشکلاتی مواجه بوده است. این مشکلات به طور خلاصه عبارتند از:

۱ـ رشد فناوری: در طی دهه‌های اخیر، سخت‌افزار با رشد چشم‌گیری مواجه بوده است و سرعت رشد آن روز به روز در حال افزایش می‌باشد. این رشد سریع این الزام را بر روی نرم‌افزار ایجاد می‌کند تا برای تطبیق خود با سخت‌افزار به صورت دائم در حال رشد باشد که این خود باعث بروز مشکلاتی در عرصه نرم‌افزار می‌شود.

۲ـ تغییرات مدام بازار و نیازهای مشتری: یکی دیگر از مشکلات نرم‌افزار، رشد سریع نیازهای موجود در بازار و به وجود آمدن تغییرات زیاد در آن‌ها می‌باشد که باعث فشردگی زیاد در عرصه تولید نرم‌افزار شده است. به عبارتی دیگر هرچه یک نرم‌افزار زودتر به بازار بیاید احتمال موفقیت آن بیشتر خواهد بود و همین امر باعث ایجاد فشار بر روی زمان تولید نرم‌افزار و کوتاهتر کردن فرآیند آن می‌شود.

۳ـ گستردگی نرم‌افزار و عدم امکان تغییر سریع در آن: یکی دیگر از مشکلات نرم‌افزار علاقمند شدن عموم مردم به استفاده از آن درزندگی روزمره‌ی خود می‌باشد.درواقع طی دهه‌ی نود نرم‌افزار علاوه برمصارف صنعتی،به حوزه‌ی مصرفی خانه‌ها و استفاده شخصی افراد نیز وارد شد. همه‌گیر شدن نرم‌افزار سبب می‌شود تا امکان تغییر در نرم‌افزار و تعویض آن با نرم‌افزار جدید با مشکل روبرو شود و نتوان به سرعت این کار را انجام داد.

۴_ برقراری امنیت و تضمین کیفیت نرم‌افزار: با پا به عرصه نهادن اینترنت در دهه‌ی نود میلادی به خانه‌های کاربران، نیاز به ایجاد نرم‌افزارهایی که امکان برقراری ارتباط بین کامپیوترها و مدیریت اطلاعات و نیز حفاظت اطلاعات را داشته باشند بیش از پیش احساس شد. درحال حاضر بارشدروزافزون تهدید و حملات کامپیوتری حفاظت از نرم‌افر بسیار ضروری است. این کار باعث می‌شود که اقدامات بسیاری در زمینه‌ی امنیت نرم‌افزار صورت گیرد. همچنین نرم‌افزار تولیدی باید دارای کیفیت و قابل اعتماد باشد.

۵_ قابلیت توسعه سامانه‌های قدیمی و نرم‌افزارهای کنونی: وجود سامانه‌های قدیمی در سازمان‌ها خود می‌تواند باعث بروز مشکلات فراوانی در حوزه‌ی نرم‌افزار شود. این‌که چطور می‌خواهیم آن‌ها را بر اساس نیازهای کنونی به روز کنیم و چگونه از آن‌ها پشتیبانی کنیم و نیز نگهداری آن‌ها جزو بحث‌های امروزی حوزه‌ی نرم‌افزار می‌باشد. همچنین در مورد نرم‌افزارهای کنونی نیز باید قابلیت توسعه آن‌ها مورد بررسی قرار گیرد.

نگرانی‌های حوزه نرم‌افزار عبارتند از:

۱) چرا هزینه‌های سخت‌افزارها این‌قدر بالا می‌باشد؟

۲) چرا زمان و تلاش بیاری را برای نگهداری و برطرف کردن مشکلات نرم‌افزارهای موجود هزینه می‌کنیم؟

۳) چرا هزینه تولید نرم‌افزارها این‌قدر بالا می‌باشد؟

ویژگی‌های نرم‌افزار

نرم‌افزار ویژگی‌های منحصر به فردی دارد که باعث تمایز آن از سایر محصولات فیزیکی (که طی فرآیندهای مهندسی به دست می‌آیند) می‌شود. تولید یک نرم‌افزار بیش از اینکه یک فرآیند فیزیکی باشد یک فرآیند منطقی است. بنابراین برای به دست آوردن بینش نسبت به نرم‌افزار (و نیز فهم کامل مهندسی نرم‌افزار) لازم است که تفاوت‌ها را بشناسیم.

۱ـ نرم‌افزار توسعه و یا مهندسی می‌شود، نه این‌که به سبک کلاسیک ساخت و تولید شود.

شباهت بین فرآیند توسعه نرم‌افزار وفرآیند ساخت و تولید محصول فیزیکی، در نیاز مبرم آن‌ها به طراحی و پیاده‌سازی خوب و همکاری و هماهنگی مناسب بین اعضای تولید کننده آن‌ها می‌باشد. اما چیزی که باعث بروز تفاوت در میان این دو می‌شود ماهیت متفاوت این فعالیت‌ها بایکدیگر است.برای توسعه‌ی نرم‌افزاربیشترنیازمندفعالیت‌های فکری ومنطقی هستیم،درحالیکه درتولید محصولات فیزیک فعالیت‌های فیزیکی بیشتر به کار می‌آیند. در واقع مدیریت پروژه‌های نرم‌افزاری بسیار متفاوت از مدیریت پروژه‌های تولیدی می‌باشد. به عبارت دیگر هزینه تولید و توسعه نرم‌افزار بیشتر بر روی فعالیت‌های مهندسی متمرکز می‌باشد در حالی که هزینه تولید و توسعه‌ی محصول فیزیکی بیشتر بر روی مواد خام و اولیه و تولید آن‌ها متمرکز است.

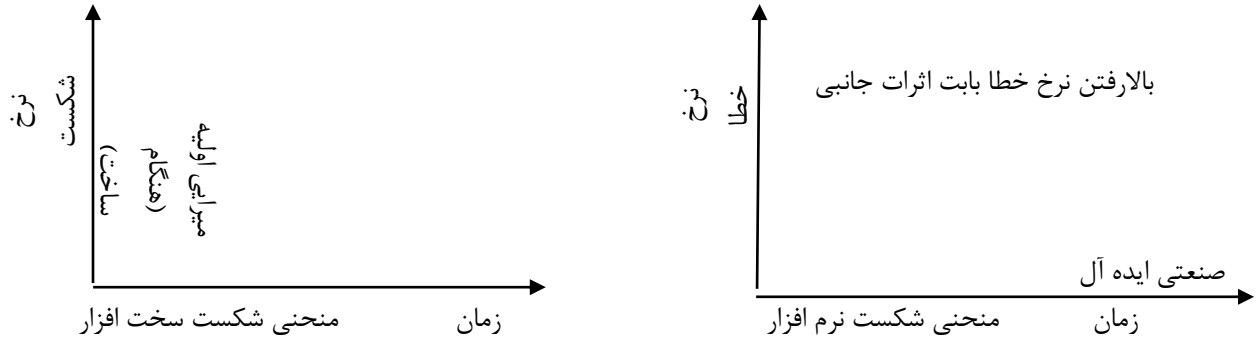
نکته ۳: به دلیل تمرکز بیشتر تولید و توسعه نرم‌افزار بر روی فعالیت‌های مهندسی، سرعت تغییر در این فعالیت‌ها نسبت به فعالیت‌های مربوط به توسعه محصول فیزیکی بسیار سریع‌تر است.

۲ـ نرم‌افزار دور انداختنی نیست.

یکی از تفاوت‌های عمده بین سخت‌افزار ونرم‌افزار در این است که سخت‌افزار پس از طی شدن یک بازه زمانی از مدت زمان استفاده آن، دچار استهلاک شده و دیگر قابل استفاده نیست. در واقع به علت استفاده زیاد از آن ممکن است خللی در کارآیی سخت‌افزار به وجود بیاید. به عبارت دیگر سخت‌افزارها تاریخ مصرف دارند و بعد از مدتی استفاده کم‌کم دورانداختنی می‌شوند. اما نرم‌افزار در حالت ایده‌آل بعد ازمدتی که ازتولید آن گذشت و اشکالات مربوط به آن برطرف شد دیگر برای همیشه و بدون اشکال قابل استفاده می‌باشد. اما تغییر در نیازهای کاربر باعث بروز خلل در کارآیی یک نرم‌افزار می‌شود و با گذشت زمان به علت بروز این تغییرات، نرم‌افزار دیگر قادر نخواهد بود نیازهای کاربر را برآورده کند. بنابراین برای

برآورده کردن نیازهای جدید کاربر نیازمند اعمال تغییرات در نرم‌افزار می‌باشیم که این اعمال تغییرات باعث بروز مشکلاتی در نرم‌افزار می‌شود و ممکن است اعمال این تغییرات با خطا مواجه شود. به طور خلاصه تفاوت اصلی نرم‌افزار و سخت‌افزار را می‌توان این‌گونه بیان کرد که:

سخت‌افزار دورانداختنی است وبعد ازمدتی استفاده به دلیل استهلاک کارآیی خودش راازدست می‌دهددر حالی که نرم‌افزار دورانداختنی نیست و روبه زوال می‌رود. یعنی به علت بروز تغییر در نیازهای کاربران نیازمند اعمال تغییر در آن هستیم که همین اعمال تغییر مداوم باعث رو به زوال رفتن نرم‌افزار می‌شود.



نمودار

شکل منحنی شکست نرم‌افزار

۳ـ اگرچه صنعت امروزی در جهت تولید بر اساس مونتاژ قطعات سخت‌افزاری و تولید جداگانه هر کدام در مکان و زمان مختلف حرکت می‌کند اما نرم‌افزار براساس نیاز مشتری و به صورت سفارشی ساخته می‌شود.

یکی دیگرازتفاوت‌های اساسی نرم‌افزاروسخت‌افزاردراین است که درتولیدسخت‌افزاردرحال حاضرازقابلیت استفاده ازمؤلفه‌های آماده بسیار استفاده می‌شود.درواقع ماهیت سخت‌افزار این امکان رابه مامی‌دهدتاهریک ازاجزای یک محصول را(حتی درمکان‌های متفاوت)به صورت جداگانه بسازیم ودرنهایت آن‌هارا بایکدیگر مونتاژکنیم: درحوزه نرم‌افزاراین ایده به تازگی مطرح واستفاده ازمؤلفه‌های آماده جهت ساخت نرم‌افزار مرسوم شده است. همچنین درحوزه نرم‌افزار به دلیل تاکید آن برروی تولیدبه صورت سفارشی،ایده‌های نوینی پدیدارشده‌اندکه باعث بروزخلاقیت و نوآوری بیشتری در تولید نرم‌افزارشده است. پس به طورخلاصه تفاوت سوم نرم‌افزار و سخت‌افزار را این‌گونه می‌توان بیان کرد که:

نکته ۴: امکان استفاده از مؤلفه‌های آماده در سخت‌افزار بیشتر از نرم‌افزار می‌باشد ولی نرم‌افزار قابلیت سفارشی‌سازی بیشتری نسبت به سخت‌افزار دارد.

کاربردهای نرم‌افزار

امروزه کاربردهای مختلفی برای نرم‌افزاردرحوزه‌های مختلف به وجود آمده است که این امرموجب پیچیدگی بیشتر فرآیند مهندسی نرم‌افزار می‌شود.قبل ازاین‌که به بحث درموردکاربردهای مختلف نرم‌افزار بپردازیم که این کاربردها اکثراً از نظر محتوا با یکدیگر متفاوت می‌باشند خوب است که اندکی راجع به عوامل مؤثر بر روی محتوای نرم‌افزارها بحث کنیم. محتوای نرم‌افزار از طریق دو عامل زیر تعیین می‌شود:

۱ـ محتوای اطلاعات: محتوای اطلاعات شامل شکل، مفهوم و محتوای مربوط به اطلاعات ورودی و خروجی می‌شود.

۲ـ قطعیت اطلاعات: این معیار، به قابلیت پیش‌بینی سفارش و زمان‌بندی اطلاعات اشاره دارد.

در ادامه به تشریح برخی از کاربردهای نرم‌افزار به طور مختصر می‌پردازیم.

کاربرد های نرم افزار

۱ـ نرم‌افزارهای سیستمی (System Software):

نرم‌افزارهای سیستمی مجموعه‌ای از برنامه‌ها هستند که برای سرویس دادن به سایر برنامه‌ها نوشته می‌شوند. بعضی از نرم‌افزارهای سیستمی مانند کامپایلرها، ویرایشگرها، برنامه‌های مدیریت فایل و... ساختار اطلاعاتی پیچیده، ولی تعیین شده‌ای دارند. (نرم‌افزار زمانی تعیین شده می‌باشد که بتواند ترتیب و زمان ورودی‌ها، پردازش‌ها و خروجی‌های قابل پیش‌بینی را پردازش کند) و بعضی دیگر از این نرم‌افزارها مانند مؤلفه‌های سیستم‌عامل، محرک‌ها، نرم‌افزارهای شبکه و پردازشگرهای ارتباطی مقدار زیادی اطلاعات تعیین نشده را پردازش می‌کنند.

۲ـ نرم‌افزارهای کاربردی (Application Software):

این حوزه شامل نرم‌افزارهایی می‌باشدکه برای حل یک مشکل خاص تجاری ایجاد شده‌اند. نرم‌افزارهای این حوزه داده‌هایی که از فرآیندهای تجاری به دست می‌آید را به صورتی پردازش می‌کنند که تصمیم‌گیری فنی / مدیریتی با عملیات تجاری را تسهیل کنند.

۳ـ نرم‌افزارهای علمی / مهندسی (Engineering/Scientific software):

این حوزه شامل نرم‌افزارهایی می‌باشد که برای حل مشکلات مربوط به رشته‌های مهندسی و یا علمی به کار می‌روند. نمونه‌های متعارف آن‌ها عبارتند از سیستم‌های زمین‌شناسی، تحلیل استرس در شاتل فضایی و نمونه‌های دیگر مشابه آن‌ها.

۴ـ نرم‌افزارهای توکار ـ تعبیه شده (Embedded Software):

نرم‌افزارهای تعبیه شده، نرم‌افزارهایی هستند که برای انجام یک وظیفه خاص در محصولات و یا سیستم‌های دیگر تعبیه می‌شوند. هدف از ایجاد آن‌ها پیاده‌سازی و کنترل خصوصیت‌ها و کارکردهایی برای سیستم‌ها و یا کاربران نهایی می‌باشد. نرم‌افزارهای تعبیه شده گاهی ممکن است که

کارکردهای محدود وداخلی داشته باشند(مانندکنترل صفحه یک مایکروفر) وگاهی اوقات ممکن است کارکردهای تأثیرگذار و حیاتی داشته باشند (مانند سیستم کنترل چرخ‌های اتومبیل و یا کنترل سوخت).

فصل دوم
مهندسی سیستم و مهندسی نیازمندی‌ها

بخش اول: مهندسی سیستم

قبل از این که بحث در مورد مهندسی سیستم را شروع کنیم خوب است چندین تعریف را در این زمینه بیان کنیم:

۱ـ سیستم چیست؟

سیستم عبارت است ازاجزای به هم پیوسته و مرتبط بایکدیگرکه به طوری ت نظیم شده‌اند که یک کل مجزا از تک‌تک اجزاء را به وجود می‌آورند. به بیان ساده‌تر، سیستم عبارت است ازمجموعه‌ای از عناصر به‌هم پیوسته، ولی مجزا از یکدیگر که در راستای تحقق یک هدف واحد در کنار یکدیگر به فعالیت می‌پردازند.

۲ـ سیستم کامپیوتری چیست؟

سیستم کامپیوتری مجموعه‌ای از عناصر می‌باشد که در کنار یکدیگر سازمان‌دهی شده‌اند تا هدف از پیش تعریف شده‌ای را با پردازش اطلاعات تأمین نمایند. این هدف ممکن است برخی از اعمال کسب‌وکار را حمایت کند یا محصولی را توسعه دهد که قابلیت فروش به عنوان سرمایه را دارا می‌باشد. برای رسیتن به این هدف بهتر است که عناصر گوناگون یک سیستم کامپیوتری را بشناسیم.

این عناصر عبارتند از:

۱ـ نرم‌افزار: برنامه‌های کامپیوتری، ساختمان داده‌ها و مستندات مربوط به آن می‌باشند که برای بهبود کارآیی سیستم روش منطقی، روال و یا کنترل را عرضه می‌نمایند.

۲ـ سخت افزار: طیف گسترده‌ای از وسایل مختلفی را دربر می‌گیرد که برای ا هداف مختلف در قسمت‌های مختلف سازمان مورد استفاده قرار می‌گیرند. سخت‌افزار می‌تواند وسایل الکترونیکی که قابلیت محاسباتی را فرهم می‌کنند. ابزار ارتباطی (همانند سوییچ‌های شبکه و...) که جریان یافتن داده‌ها را تأمین می‌کنند و با ابزارهای الکترومکانیکی (همانند سنسورها) که تعامل با محیط را برقرار می‌نمایند، را شامل می‌شود.

۳ـ افراد: کاربرها واپراتورهای سخت‌افزارونرم‌افزارراشامل می‌شود.مهندسین نرم‌افزار نیز جزئی از افراد درون سیستم به حساب می‌آیند.

۴ـ پایگاه داده: مجموعه‌ای عظیم و سازمان‌یافته از اطلاعات که توسط نرم‌افزار قابل دستیابی است.

۵ـ مستندات: مستندات یک سازمان یا سیستم می‌باشد که شامل اطلاعات توضیحی همانند کتابچه‌های راهنما، جزوات، سایت‌های وب و... می‌شوند. این مستندات نحوه استفاده از سیستم و یا امور مختلف مربوط به سیستم را نشان می‌دهند.

۶ـ روال‌ها: مجموعه‌ای از مراحل هستند که نحوه استفاده از هر کدام از عناصر سیستم و یا قسمتی از سیستم را تشریح می‌کند.

۳ـ مهندسی سیستم چیست؟

مهندسی سیستم فرآیندی است که تمرکز خود را صرف مجموعه‌ای از عناصر تحلیل، طراحی و سازمان‌دهی آن‌ها در قالب یک سیستم می‌کند که نتیجه‌ی این کار تولید یک محصول، ارائه خدمت یا سرویس یا یک فناوری مناسب برای تبدیل و کنترل آن‌هاست.

نکته ۱: اگر محدوده‌ی کار آن تمرکز بر روی کسب کار باشد، مهندسی فرآیند کسب‌وکار نامیده می‌شود و اگر توسعه‌ی یک محصول مدنظر باشد، این فرآیند مهندسی محصول نامیده می‌شود. در ادامه به توضیح این دو خواهیم پرداخت.

فصل چهارم

مفاهیم شیء گرایی و UML

در بخش اول این فصل مفاهیم شیءگرایی مورد بررسی قرار می‌گیرند تاریخچه‌ی شیء گرایی، مفاهیم مرتبط با آن و ویژگی‌ها و مزایای شیءگرایی دراین بخش موردبحث قرارمی‌گیرند.دربخش دوم این فصل مدل‌سازی ازطریقUML مورد بررسی قرار می‌گیرد. تاریخچه‌ی به وود آمدنUMLبه همراه ویژگی‌ها ونمودارهای مختلف آن موردبررسی قرارمی‌گیرند.در بخش سوم این فصل RUP که یک محصول مبتنی بر متدولوژی شیءگرایی می‌باشدمورد بحث قرار می‌گیرد. در بخش چهارم نیز متدولوژی‌های نرم‌افزاری مورد بحث قرار می‌گیرند.

بخش دوم: مدل سازی با UML

UMLیازبان مدل‌سازی یکنواخت،زبانی است برای مشخص کردن (Specift)،مصورسازی (Visualize)، ساخت (Construction) و مستندسازی (Documenting) سیستم‌های نرم‌افزاری و غیر نرم‌افزاری. همچنین از آن برای مدل‌سازی سیستم‌های ت جاری نیز استفاده می‌شود.

ایجاد یک مدل برای سیستم‌های نرم‌افزاری قبل از ساخت یا بازساخت آن ، به اندازه داشتن نقشه برای ساختن یک ساختمان ضروری و حیاتی است. بسیاری از شاخه‌های مهندسی، توصیف چگونگی محصولاتی که بایدساخته شوند راترسیم می‌کنند و همچنین دقت زیادی می‌کنند که محصولاتشان طبق این مدل‌ها و توصیف‌ها ساخته شوند.مدل‌های خوب ودقیق دربرقراری یک ارتباط کامل بین افراد پروژه، نقش زیادی می‌تواننداشته باشند علت اصلی مدل کردن سیستم‌های پیچیده این باشدکه نمی‌توان به یکباره کل سیستم را تجسم کرد و ممکن است سیستم دارای ابهامات بسیاری باشد. لذا برای رفع این ابهامات و نیز برای فهم کامل سیستم و یافتن و نمایش ارتباط بین قسمت‌های مختلف، از مدل‌سازی استفاده می‌کنیم.

UML را می‌توان زبانی برای مدل‌سازی یا ایجاد نقشه تولید نرم‌افزار دانست. به عبارت دیگر، یک زبان مدل‌سازی، زبانی است که فرهنگ لغات و قواعد آن برنمایش فیزیکی و مفهومی آن سیستم متمرکز می‌باشند. از همین رو برای سیستم های نرم‌افزاری نیاز به یک زبان مدل‌سازی داریم که بتواند دیدهای مختلف معماری سیستم را در لو چرخه‌ی تولید آن، مدل کند.

فرهنگ واژگان و قواعد زبانی UMLبه شما می‌گویندکه چگونه یک مدل رابسازید ویا چگونه یک مدل را بخوانید. اما به شما نمی‌گویند که در چه زمانی، چه مدلی را ایجاد کنید.

ویژگی‌های UML

UML دارای ویژگی‌های بارز فراوانی است که در این قسمت به آنها می‌پردازیم:

۱ـ UML قابل فهم برای برنامه‌نویسان و حتی ماشین می‌باشد. در واقع یکی از ویژگی‌های اصلی آن تسهیل کردن ارتباطات بین اعضای پروژه و یا بین تولیدکنندگان یا ذی‌نفعان می‌باشد.

۲ـ UML زبان پیچیده‌ای می‌باشد زیرا از یک طرف هم باید نمودارهایی در آن فراهم باشد که در هر موقعیتی و با هر ترتیبی قابل استفاده باشد و از طرف دیگر این زبان ترکیبی از زبان‌های مختلف می‌باشد، که برای حفظ سازگاری و جمع کردن خصوصیات مثبت آنها ناگزیر از پذیرش این پیچیدگی می‌باشد.

۳ـ UML یک زبان برنامه‌نویسی بصری نمی‌باشد. اما مدل‌های بدست امده از آن را می‌توان مستقیماً به انواع زبان‌های مختلف ارتباط دارد. یعنی امکان نگاشت از UML به کد زبان‌های برنامه‌نویسی مثل Java و C++ و بالعکس می‌باشد. به این کار مهندسی رو به جلو و مهندسی معکوس می‌گویند.

۴ـ در مقایسه با زبان‌های مدل‌سازی دیگر مثل ER و زبان فلوچارتی DR، زبان UML نمودارهای قوی‌تر و قابل فهم‌تری را ارائه می‌دهد که شامل تمامی مراحل چرخهٔ حیات تولید نرم‌افزار (تحلیل نیازمندی‌ها، طراحی، پیاده‌سازی و آزمایش) می شود.

۵ـ یکی دیگر از ویژگی‌های مهم UML این است که مستقل از متدولوژی با فرآیند تولید نرم‌افزار می‌باشد و این بدان معنی است که شما برای استفاده از UML، نیاز به استفاده از یک متدولوژی خاص ندارید و می‌توانید طبق متدولوژی قبلی خود عمل کنید با این تفاوت که مدل‌هایتان را با UML نمایش می‌دهید.

۶ـ ازویژگی‌هایUMLمی‌توان به پشتیبانی ازمفاهیم سطح بالای شیءگرایی مثل همکاری (Collaboration)،چارچوب (Framework)، الگو (Pattern)،ومؤلفه (Component) اشاره کرد. همچنین UML با استفاده از یک سری مکانیزم‌های گسترش‌پذیر امکان می‌دهد که بتوان زبان‌های مدل‌سازی جدیدتری (با گسترش مفاهیم پایه‌ای موجود) ایجاد کرد.

تاریخچه‌ی UML

UML یک زبان استاندارد برای نمایش، ایجاد و مستندسازی سیستم‌های نرم‌افزاری مبتنی بر روش‌های شیءگرا می‌باشد. قبل از UML نیز روش‌های شیءگرایی متعددی توسط افراد مختلف برای مدل‌سازی سیستم‌های شیءگرا ارائه شده بود. اتفاقی که باعث ایجاد UML شد بدین صورت بود که Rumbough، طراح متدلوژی OMT به شرکت Rational که متعلق به Booch بود پیوست و آنها تلاش خود را برای ایجاد یک زبان مدل‌سازی شیءگرای متحدالشکل بکارگرفتند. ترکیب دوم تدولوژی وایجاد زبان UML اعتبار ویژه‌ای به آنها بخشید. درسال ۱۹۹۵، شرکت Rational آماده بود تا اولین مستندات UML (نسخه ۰,۸) را ارائه نماید. اما در این اقدام ناگهانی امتیاز شرکت Jacobson را که مالک Objectory بود، خریداری نمود پس از اقدام، شرکت Rational با ترکیب سه متدولوژی سطح بالا قادر به ارائه یک استاندارد در روش‌های شیءگرا بود.

درسال۱۹۹۷،UMLبعنوان یک زبان استانداردمدل‌سازی شیءگرا ازطرف گروه OMG(Object Management Group) پذیرفته شد. مهمترین قابلیت این زبان ارائه مدل‌هایی بصورت نمودار برای کل چرخه حیات نرم‌افزار است و می‌تواند بصورت یک زبان ارتباطی بین تمام گروه‌های یک

تیم پروژه استفاده شود. از قابلیت‌های دیگر آن این می‌باشد که سازگاری خود را با اغلب روش‌های متداول مانند OMT، Booch و OOSE حفظ نموده است.

از دید مستندسازی، UML قادر است کل چرخه حیات سیستم را در قالب نمودارهایی بصورت کلی و قابل فهم ارائه نماید که می‌تواند مستقل ازمتدولوژی ساخت ارائه شودهرچند که برخی ازمتدولوژی‌ها دیاگرام‌های خاص خودرادارند. اما باتوجه به نزدیکی متدولوژی‌های شیءگرا و شباهت نمودارهای آنها می‌توان UML را در بسیاری از متدولوژی‌ای شیء گرا استفاده نمود.

شرکت هایی مانند Unisys, Rational / Oracle/ IBM, Microsoft, HP و... از شرکت‌هایی هستند که از UML استفاده کرده و آن را پشتیبانی می‌نمایند.

نمودارهای UML

درفصل سوم بسیاری از این نمودارها را ک ه مرتبط با تحلیل شیءگرا بودند را معرفی کردیم. در این قسمت تنها به معرفی مختصر این نمودارها می‌پردازیم.

۱ـ نمودار کلاس

این نمودار، کلاس‌ها، واسطه‌ها و همکاری و روابط بین آنها را نمایش می‌دهد. و از نمودارهای اصلی و مرکزی UML می‌باشد که بیان کننده ساختار ایستای سیستم نرم‌افزاری می‌باشند.

نکاتی در مورد نمودار کلاس:

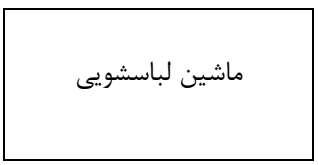
ـ کلاس‌ها به وسیله صفات و عملیات خود شناسایی می‌شوند. صفات معرف ماهیت کلاس هستند و عملیات معرف رفتار کلاس می باشند.

ـ مسئولیت هر چیزی است که کلاس می‌داند و با قادر به انجام آن می‌باشد. به عبارت دیگر می‌توان گفت مسئولیت همان صفات و عملیات مرتبط با یک کلاس هستند.

ـ همکارها کلاس‌هایی می‌باشند که اطلاعات مورد نیز یک کلاس را برای تکمیل مسئولیت آن کلاس فراهم می‌کند. به طور کلی، همکاری بر روی درخواست اطلاعات یا درخواست انجام بعضی از اعمال دلالت دارد.

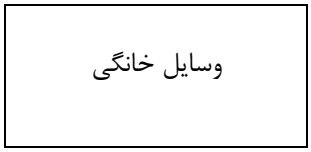
مجسم سازی یک کلاس

همان گونه که قبلا توضیح داده شد، برای نمایش کلاس از نماد مستطیل در UML استفاده می شود. نام کلاس در قسمت بالای مستطیل آورده می شود. به شکل توجه کنید.



شکل - آین کلاس UML

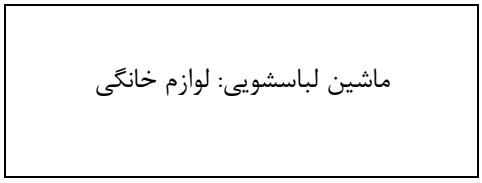
مفهوم دیگی که در UML وجود دارد، بسته است که می تواند نقش یک کلاس را بازی کند. همان گونه که قبلا هم توضیح داده شده، یک بسته مکانیزمی برای حل و مدیریت پیچیدگی سیستم می باشد و هدف آنها سازمان دهی عناصر یک نمودار است. در UML همانطور که دیدید، یک بسته به صورت یک پرونده گوشه دار به نمایش در می آید. شکل یک بسته را نشان می دهد.



شکل - یک بسته UML

اگر کلاس ماشین لباشویی، بخشی از یک بسته است (در اینجا وسایل خانگی یک بسته است)، در نتیجه آن را می توان به صورت ماشین لباسشویی : وسایل خانگی نشان داد.

علامت : نام بسته (در سمت چپ) را از نام کلاس در (سمت راست) جدا می کند. این نوع از نام کلاس، به نام مسیر نامیده می شود. به شکل دقت کنید.



شکل - یک کلاس با نام مسیر

صفات

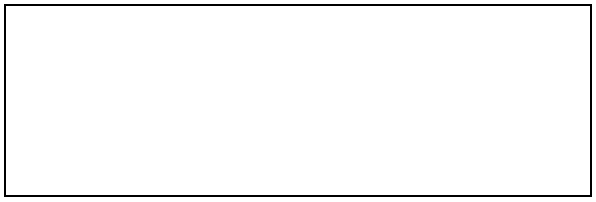
یک صفت یک ویژگی از یک کلاس است. صفت، محدوده مقادیری که صفت در داخل اشیای آن کلاس ممکن ست به خود بگیرد را توضیح می دهد (یعنی نمونه ها). یکی کلاس ممکن است از صفر تا چند صفت داشته باشد. نام صفات در زیر خط جدا کننده نام کلاس و صفات، آورده می شود. همانگونه که در شکل نشان داده شده است، هر شی کلاس دارای یک مقدار خاص برای هر صفت می باد. تصویر مثالی را در این ارتباط نشان می دهد. توجه کنید که پس از نام شی علامت (:) آورده شده و بلافاصله نام کلاس آمده است، نام کلاس زیر خط کشیده شده است.

ماشین لباسشویی
نام تجاری نام مدل شماره سریال ظرفیت

شکل - یک کلاس با صفاتش

UML، این امکان را می دهد تا اطلاعات اضافی تری برای صفات، معین کنید. در نماد کلاس می توانید نوع هر صفت را مشخص کنید. به طور مثال انواع ممکن شامل : رشته ای، مقدار اعشاری، عدد و بولین (و انواع دیگر). برای مشخص نمودن نوع یک صفت، با استفاده از علامت (:) نام صفت را از نوع آن جدا می کنیم. همچنین می توانید مقدار پیش فرضی برای یک صفت مشخص کنیم. شکل ۶-۳ این طریقه نمایش صفات را نشان می دهد.

ماشین لباسشویی: ماشین لباسشویی من
نام تجاری:" خوب شو" نام مدل: " آناهیتا" شماره سریال: " GL۵۷۷۷۴ " ظرفیت: ۱۶

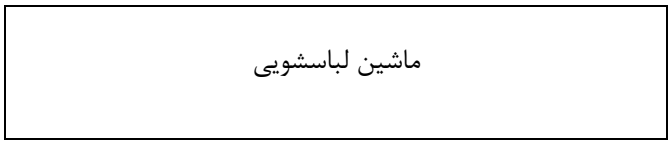


شکل - یک شی دارای یکمقدار مشخص برای صفات کلاسش است.

شکل - یک صفت همانند مقدار پیش فرضش می تواند نوعش را نشان دهد.

اعمال

یک عمل چیزی است که یک کلاس می تواند انجام دهد، یا آنکه شما (یا کلاس دیگری) می توانید برای کلاس انجام دهید عمل توسط یک خط از مجموعه صفات جدا می شود شکل .



نام تجاری نام مدل شماره سریال ظرفیت
اضافه کردن لباسها() برداشتن لباسها() اضافه کردن پودر() روشن کردن()

شکل - فهرست اعمال یک کلاس زیر خط جدا کننده صفات از اعمال قرار می گیرد.

همان گونه که می توانید برای صفات خاصه، اطلاعات اضافی تعریف کنید، برای اعمال نیز می توانید اطلاعات اضافی تری معرفی کنید. در پرانتزی که به دنبال عمل می آید، می توانید پارامترهایی را که براساس آنها عمل، کار می کند، به همراه نوع پارامتر بیاورید. یکی انواع عمل، تابع است که بعد از پایان گرفتن عمل یک مقدار برگرداند. برای یک تابع می توانید مقداری که بر می گرداند و نوع آن را نشان دهید.

قطعات اطلاعاتی درباره یک عمل، امضای اعمال نامیده می شود. تصویر ۸-۳ این مطلب را نشان می دهد.

صفات، اعمال و مجسم سازی

تاکنون سعی کردیم که کلاسها را به صورت مجزا نشان دهیم و صفحات و اعمال آنها را داخل کلاسها بیاوریم. در دنیای واقعی، مجبور هستیم بیشتر از یک کلاس را در آن واحد نشان دهیم. به همین منظور وقتی در حال نمایش کلاسها هستیم، نیازی نیست که هر بار، کلیه صفات و اعمال هر کلاس را رسم کنیم. چون وقت زیادی را می گیرد، به جای آن فقط نام کلاس را می گذاریم و قسمتهای صفات و اعمل را خالی می گذاریم. تصویر این موضوع را نشان می دهد.

ماشین لباسشویی

نام تجاری نام مدل شماره سریال ظرفیت
اضافه کردن لباسها(c:string) برداشتن لباسها(c:string) اضافه کردن پودر(d:integer) روشن کردن(): Boolean

شکل - امضا برای یک عمل

ماشین لباسشویی

شکل - در عمل شما همیشه تمام صفات و عملیات شی را نشان نمی دهید.

گاهی اوقات بهتر است که بعضی (نه همه) صفات یا اعمال را نشان دهیم. برای این منظور، وقتی چند صفت یا عمل را نشان دادید، بقیه آن را با «...» ادامه دهید که به این کار حذف کردن یک کلاس گویند. تصویر آن را نشان می دهد.

اگر فهرستی طولانی از صفات و اعمال دارید، می توانید از یک کلیشه برای سازمان دهی این صفات یا اعمال استفاده کنید؛ به نحوی که فهرست را خواناتر کند. یک کلیشه راهی جهت توسعه اصطلاحات و فرهنگ لغات UML می باشد؛ ضمن آنکه امکان ایجاد عناصر جدید را فراهم می آورد، به نحوی که مختص یک مساله خاص هستند که شما سعی در حل آن دارید. همان گونه که درفصلهای قبلی توضیح داده شد، یک کلیشه را داخل

یک جفت « » نانمی دهیم. برای فهرست صفات، می توانید از یک کلیشه به عنوان سر صفحه یا عنوان جهت مجموعه ای از صفات استفاده کنیـد، همان گونه که در تصویر آمده است.

ماشین لباسشویی
نام تجاری ....
اضافه کردن لباسها() ....

شکل - یک سه نقطه اشاره به این موضوع دارد که صفات و اعمال نوشته شده تمام مجموعه نمی باشد.

«ماشین لباسشویی»
اطلاعات شناسایی نام تجاری نام مدل شماره سریال «اطلاعات ماشین» ظرفیت
«اطلاعات لباسها» اضافه کردن لباسها()

برداشتن لباسها()
اضافه کردن پودر()
«مرتبط با ماشین»
روشن کردن()

شکل - می توانید از یک کلیشه برای سازمان دهی فهرستی از صفات و اعمال استفاده کنید.

مسئولیتها و محدودیتها

نماد کلاس این امکان را فراهم می آورد تا بتوان نوع دیگری از اطلاعات درباره کلاس را مشخص نمود. می توانید مسئولیتهای یک کلاس را در ناحیه زیر فهرست اعمال قرار دهید. یک مسئولیت توصیفی است از آنچه کلاس مجبور به انجام آن استو به عبارت دیگر آنچه که صفات و اعمال آن کلاس سعی در انجام آن دارند. مثلاً ماشین لباسشویی مسئولیت گرفتن لباسهای چرک به عنوان ورودی و تبدیل آنها به لباسهای شسته و تمیز به عنوان خروجی دارد.

در نماد کلاس، مسئولیتها درناحیه زیر اعمال قرار می گیرد.

ماشین لباسشویی
اطلاعات شناسایی نام تجاری نام مدل شماره سریال «اطلاعات ماشین» ظرفیت
«اطلاعات لباسها» اضافه کردن لباسها() برداشتن لباسها()

اضافه کردن پودر() «مرتبط با ماشین» روشن کردن()
مسئولیت ها: گرفتن لباس های چرک به عنوان ورودی و تبدیل آنها به لباس های شستشو شده و تمیز به عنوان خروجی.

در آیکن کلاس، می توانید مسئولیتهای کلاس را زیر ناحیه فهرست اعمال بنویسید.

ایده کلی در اینجا آن است که اطلاعات مورد نیاز و کافی یک کلاس را دریک شکل غیر مبهم و واضح قراردهیم. مشخص نمودن مسئولیتهای کلاس، یک راه غیر رسمی برای حذف ابهامات می باشد.

حال باید محدودیت را به شکل اضافه کنیم. برای این کار از یک جفت کروشه استفاده می کنیم. متن داخل کروشه یک یا چند قاعده کلاس را مشخص می کند. مثلاً، فرض کنید در کلاس ماشین لباسشویی، شما می خواهید ظرفیت ماشین را که دارای مقادیر ۱۶، ۱۸ یا ۲۰ پوند است، به مشخصات این کلاس اضافه کنید (محدودیت صفت ظرفیت کلاس ماشین لباسشویی)، باید بنویسید [پوند ۲۰ یا ۱۸ یا ۱۶ = ظرفیت] که آن را نزدیک نماد کلاس ماشین لباسشویی در نظر می گیریم. نحوه انجام این کار را نشان می دهد.

[ظرفیت = ۱۶ یا ۱۸ یا ۲۰ پوند]

ماشین لباسشویی
نام تجاری نام مدل شماره سریال ظرفیت
اضافه کردن لباسها () برداشتن لباسها ()

اضافه کردن پودر ()
روشن کردن ()

شکل - یک جفت کروشه محدودیت صفت ظرفیت ماشین را برای سه مقدار نش ان می دهد.

نکته :

UML راه دیگری را برای اضافه نمودن محددویتهای ارائه نموده است به نحوی که امکان دقیق تر تعاریف فراهم می آید. این راه زبان کاملی بـه نـام زبان محدودیت شی (OCL) می باشد یک ابزار پیشرفته و گاهی اوقات مفید OCL دارای قوانین، اصطلاحات و اپراتورهای خاص خود است.

۲ـ نمودار اشیاء:

این نمودار، اشیاءسیستم و روابط بین آنها را نمایش می‌دهد. در واقع یک تصویر لحظه‌ای از نمودار کلاس می‌باشد.

۳ـ نمودار مورد کاربرد:

این نمودار، تعامل کنش‌گران سیستم با سیستم را مدل می‌کند. این نمودار می‌تواند به عنوان نقطه‌ی ورودی برای تمامی نمودارهای دیگر که به تشریح نیازمندی‌ها و معماری و پیاده‌سازی سیستم می‌پردازند، مورد استفاده قرار بگیرد.

نکاتی در مورد نمودار مورد کاربرد:

- این نمودار نحوهٔ ارتباط کاربر نهایی با سیستم را تشریح می‌کند.
- مهم‌ترین کاربردهای این نمودار استخراج واسط کاربری نرم‌افزار می‌باشد.
- این نمودار دنباله‌ای از عملیاتی است که یک سیستم انجام می‌دهد تا یک نتیجه قابل مشاهده و ارزشمند برای استفاده کننده از سیستم فراهم نماید. کامل بودن آن نسبی است.
- با استفاده از مدل کاربری می‌توان رفتار سیستم را از دیدگاه کاربران بیان نمود.
- برای شناسایی نیازمندی‌های کارکردی یا وظیفه‌مندی یک سیستم به کار می‌رود.

کار با نمودار مورد کاربرد

مورد کاربرد یک مفهوم قدرتمند برای کمک به تحلیل گر است تا از طریق آن چونگی رفتار یک سیستم را مورد بررسی قـرار دهد. مورد کاربرد، به گرد آوری نقطه نظرات کاربران و نیازهای آنان کمک می کند. در این فصل سعی بر آن داریم که آموخته های فصول قبلی راجع به مورد کاربرد را به کار بگیریم.

چون نمودارهای مورد کاربرد، وسیله قدرتمندی می باشند، لذا اگر از UML هم استفاده شود در نتیجه موارد کاربرد باز هم قدرتمندتر خواهند شد. مجسم سازی این امکان را به شما می دهد که بتوانید موارد کاربرد را به کاربران نمایش دهید، به شکلی که آنها اطلاعات بیشتری را به شما منتقل نمایند. این یک واقعیت در زندگی روزمره است که معمولاً کاربران، بیشتر از آن چیزی که می دانند، می توانند بیان نمایند. مورد کاربرد کمک می کند تا این موضوع حل شود. همچنین امکان ارائه تصویری بصری از نیازهای کاربران را به شما می دهد. ضمن آنکه ارائه تصویر بصری اجازه ترکیب نمودارهای کاربرد با سایر نمودارها را نیز فراهم می آورد.

یکی از اهداف روند تحلیل سیستم، ایجاد مجموعه ای از کاربردهاست. در واقع هدف آن است که بتوانیم کوارد کاربرد را فهرست نموده به نحوی که بتوانیم به این مجموعه دسترسی داشته باشیم و این مبنایی برای گرفتن نقطه نظرات کاربران در جهت چگونگی کارکرد سیستم می باشد. در ضمن، این مجموعه به عنوان پایه ای برای گرد آوری نیازهای اطلاعاتی سیستم در زمان ارتقا آن خواهدبود.

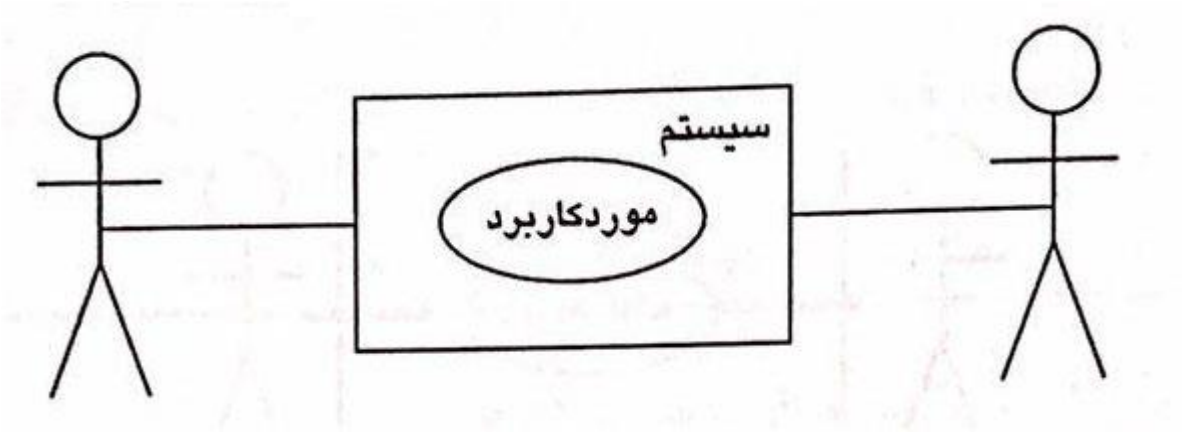
نمایش یک مدل مورد کاربرد

یک کنش گر آغاز کننده یک مورد کاربرد است و یک کنش گر (احتمالا، آغاز کننده ولی نه لزوماً) چیزی ارزشمند از مورد کاربرد دریافت می کند. نمایش گرافیکی نمودار کاربرد بسیار صریح است. نمایش مورد کاربرد توسط یک بیضی ارائه می گردد. از یک آدمک برای نمایش کنش گر استفاده می شود. کنش گر آغاز کننده در سمت چپ مورد کاربرد قرار می گیرد. نام مورد کاربرد یا داخل یا زیر بیضی نوشته می شود. یک خط تناظر، کنش گر را به مورد کاربرد متصل می کند، درست شبیه خطی که کلاسهای تناظر را به هم متصل می کرد.

یکی از خصوصیات بارز و مزایای مهم تحلیل مورد کاربرد آن است که مرز بین سیستم و دنیای خارج، از این طریق قابل نمایش است. کنش گران، به ویژه خارج از سیستم هستند، در حالی که موارد کاربرد داخل سیستم می باشد. از یک مستطیل، (با نام سیستمی که در داخل مستطیل قرار می گیرد)

برای محدوده و مرز سیستم استفاه می کنیم. مستطیل، مورد کاربردی سیستم را در داخل خودش نگه می دارد.

کنش گران، موارد کاربرد و خطوط ارتباطی یک مدل مورد کاربرد را می سازند.



شکل - در یک مدل مورد کاربرد یک آدمک نمایش گر یک کنش گر است یک بیضی مورد کاربرد را نشان می دهد و خط تناظر ارتباط بین کنش گر و مورد کاربرد را نشان می دهد.

مثال : ماشین تهیه نوشابه

حال از نمادهای آورده شده در بالا برای مثالی که در فصل قبل گفته شد استفاده می کنیم. همان گونه که به یاد دارید، مورد کاربرد را برای ماشین تهیه نوشابه توسعه دادیم. مورد کاربرد «خرید نوشابه» به همراه «پرکردن دستگاه» و «جمع کردن پول» در داخل سیستم قرار می گیرند. کنش گران، عبارت از مشتریان، نماینده تهیه کننده و جمع کننده می باشند. شکل مدل مورد کاربرد UML را برای ماشین تهیه نوشابه نشان می دهد.

دنبال کردن مراحل در سناریوها

هر مورد کاربرد مجموعه ای از سناریوهاست و هر سناریو توالی مراحل می باشد. همان گونه که مشاهده می کنید، این مراحل در نمودار ظاهر نشده اند. مراحل همانند، یادداشتها نیستند که به موارد کاربرد متصل شوند. شفافیت در هر نمودار از اصل اولیه است. درا ینجا چسباندن یادداشتها به هر مورد کاربرد باعث شلوغی نمودار می گردد. پس چگونه و کجا اطلاعات این مراحل را نگهداری کنیم ؟

نمودارهای مورد کاربرد معمولاً بخشی از مستندات طراحی هستند که هم مشتریان و هم تیم توسعه دهندگان به آن مراجعه مـی نمایند. هر نمودار دارای رویدادهای مربوط به خودش است. هر سناریو از هر مورد کاربرد نیز دارای رویدادهای مختص به خـودش می باشد. مجموعه فعالیتها و عوامل موجود در یک سناریو به این شکل است :

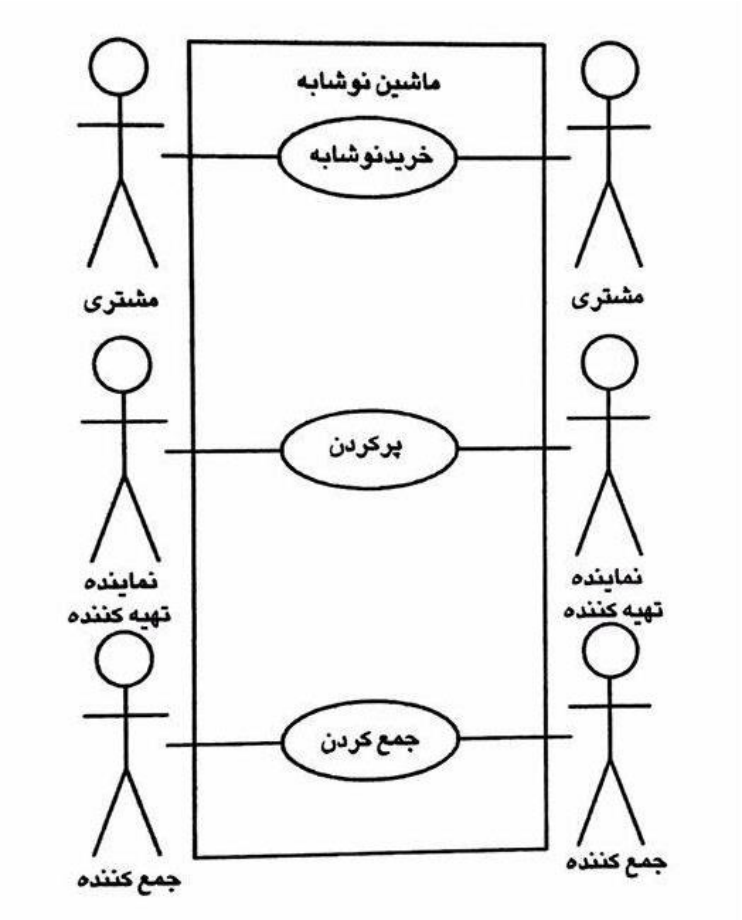
- کنش گر که مورد کاربرد را آغاز می کند.

- پیش شرطها برای مورد کاربرد

- مراحل در سناریو

- پس شرطها وقتی سناریو خاتمه می یابد.

- کنش گری که از مورد کاربرد استفاده و بهره می برد.



شکل - یک مدل مورد کاربرد برای ماشین تهیه نوشابه

همچنین می توانید فهرست مفروضات یک سنارسو را تهیه کنید (مثلاً در هر لحظه فقط یک مشتری از ماشین تهیه نوشابه استفاده می کند) و نیز می توانید توضیح کوتاهی از سناریو را با یک جمله فهرست نمایید.

درفصل قبل مورد کاربرد را معرفی کردیم و دیدیم که سناریوهای گوناگونی را می توان برای مورد کاربرد «خرید نوشابه» ارائه نمود. در توضیحاتتان می توانید فهرستی از سناریوها به صورت جداگانه داشته باشید ("نداشتن نوع خاص نوشابه" و "عدم داشتن پول کافی")، یا می توانید آنها را به عنوان استثنائاتی برای سناریوی اول در مورد کاربرد در نظر بگیرد. این کار دقیقا بستگی بیه شما و کاربران دارد.

نکته :

برای نمایش مراحل یک سناریو می تواند از روش دیگری، یعنی از نمودار فعالیت UML استفاده کرد. این موضوع را درفصلهای بعدی خواهیم دید.

مجسم سازی ارتباطات میان موارد کاربرد

در مثال فصلی قبلی به دو روش اشاره شد که موارد کاربرد می تواستند به یکدیگر مرتبط شوند. یک روش، شمول بود. این روش امکان استفاده از مراحل یک مورد کاربرد در مورد کاربردی دیگر را میداد. روش دیگر توسعه بود که به شما این امکان را می داد تا مورد کاربرد جدیدی را از طریق اضافه نمودن مراحلی به یک مورد کربرد موجود ایجاد نمایید.

دو نوع دیگر از ارتباطات عبارتند از تعمیم و گروه بندی. تعمیم ارتباطی است که بین یک چیزی عمومی یا چند نوع خاص تر از آن چیز عمومی برقرار می گردد و برای جلوگیری از تکرار صفات، اعمال و ارتباطات بین کلاسها در یک نمودار استفاده می شود. گروه بندی راهی ساده جهت سازمان دهی مجموعه ای از موارد کاربرد است.

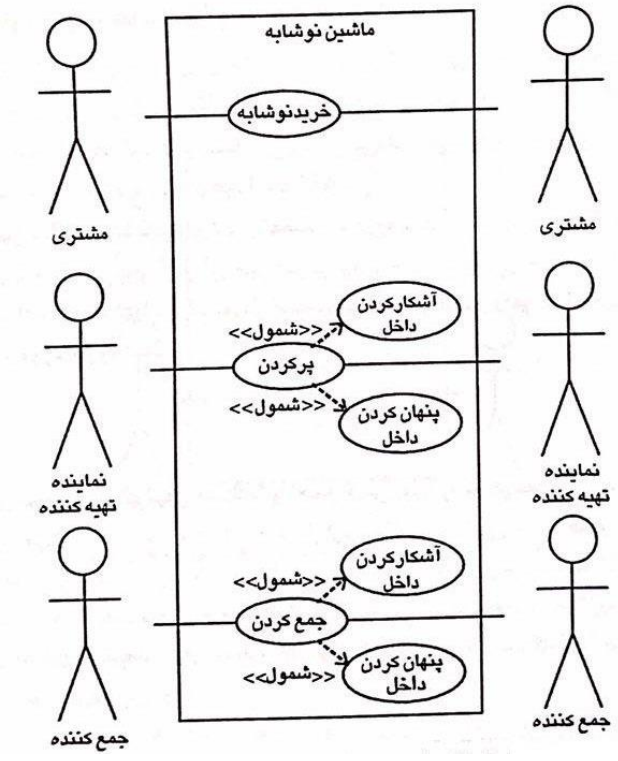
شمول

مثال مورد کاربرد «پرکردن دستگاه» و «جمع کردن پول» را که در فصل قبل آورد شده، یک بار دیکر بررسی می کنیم. هر دو مورد کاربرد با بازکردن قفل و باز کردن درب جلوگیری دستگاه شروع می شدند و هر دو نیز با بستن دربا ماشین و قفل نمودن دستگاه به پایان می رسیدند. آنگاه مورد کاربرد «آشکار کردن داخل» ایجاد شد تا دو مرحله اول را در داخل خود جا دهد و مورد کاربرد «پنهان کردن داخل» برای جادادن دو مرحله دیگر استفاده شد. موارد کاربرد «پرکردن دستگاه» و «جمع کردن پول» شامل این دو مورد کاربرد می شوند.

برای نمایش شمول، از نمادی استفاده می کنیم که برای وابستگی بین کلاسها استفاده می شد. یک خط چین، کلاسها را به هم مرتبط می کد، ضمن آنکه سمت پیکان به سوی کلاسی که وابسته بود، اشاره می کند. بالا خط، یک کلیشه اضافه می کنیم و کلمه «شمول» در داخل یک جفت گیومه قرار می گیرد. شکل ارتباط شمول را در یک مدل مورد کاربرد در ماشین تهیه نوشابه نشان میدهد.

نکته :

به خاطر داشته باشید که یک مورد کاربرد شامل شده هرگز متکی به خودش نیست. بلکه به عنوان بخشی از یک مورد کاربردی است که شامل آن می باشد.



شکل - مدل مورد کاربرد تهیه نوشابه با مشمول

در نماد متنی که مراحل توالی را دنبال می کند، موارد کاربرد شمول را مشخص می کنیم. مرحله اول در مورد کاربرد «پر کردن دستگاه» شامل (آشکار کردن داخل) خواهد بود.

توسعه

در مورد کاربرد «پر کردن دستگاه» پایه ای برای مورد کاربرد «دوباره پرکردن بر طبق فروش» می باشد. به جای اینکه ماشین نوشابه را دوباره پر کنیم، به طوری که انواع مختلف نوشابه ها با تعداد یکسانی از قوطی ها پر شود، نماینده تهیه کننده می تواند آن نوشابه ای را که فروش بهتری داشته است را بیشتر پر کند. مرد کاربرد جدید گفته می شود که مورد کاربرد اصلی را توسعه داده است، زیرا مراحل جدیدی به توالی مورد کاربرد اصلی اضافه کرده است. همچنین گاهی به مورد کاربرد اصلی، مورد کاربرد پایه نیز گفته می شود.

توسعه فقط می تواند در نقاط تخصیص داده شده خاصی در مراحل توالی مورد کاربرد پایه روی دهد. این نقاط توسعه نامیده می شود در مورد کاربرد «پر کردن دستگاه»، مراحل جدید، بعد از آنکه نماینده تهیه کننده درب ماشین را باز می کند، اتفاق می افتد و وقتی آماده پر کردن مخزنهای انواع نوشابه هاست، رخ می دهد. برای این مثال نقطه توسعه «پرکردن مخزنها» می باشد.

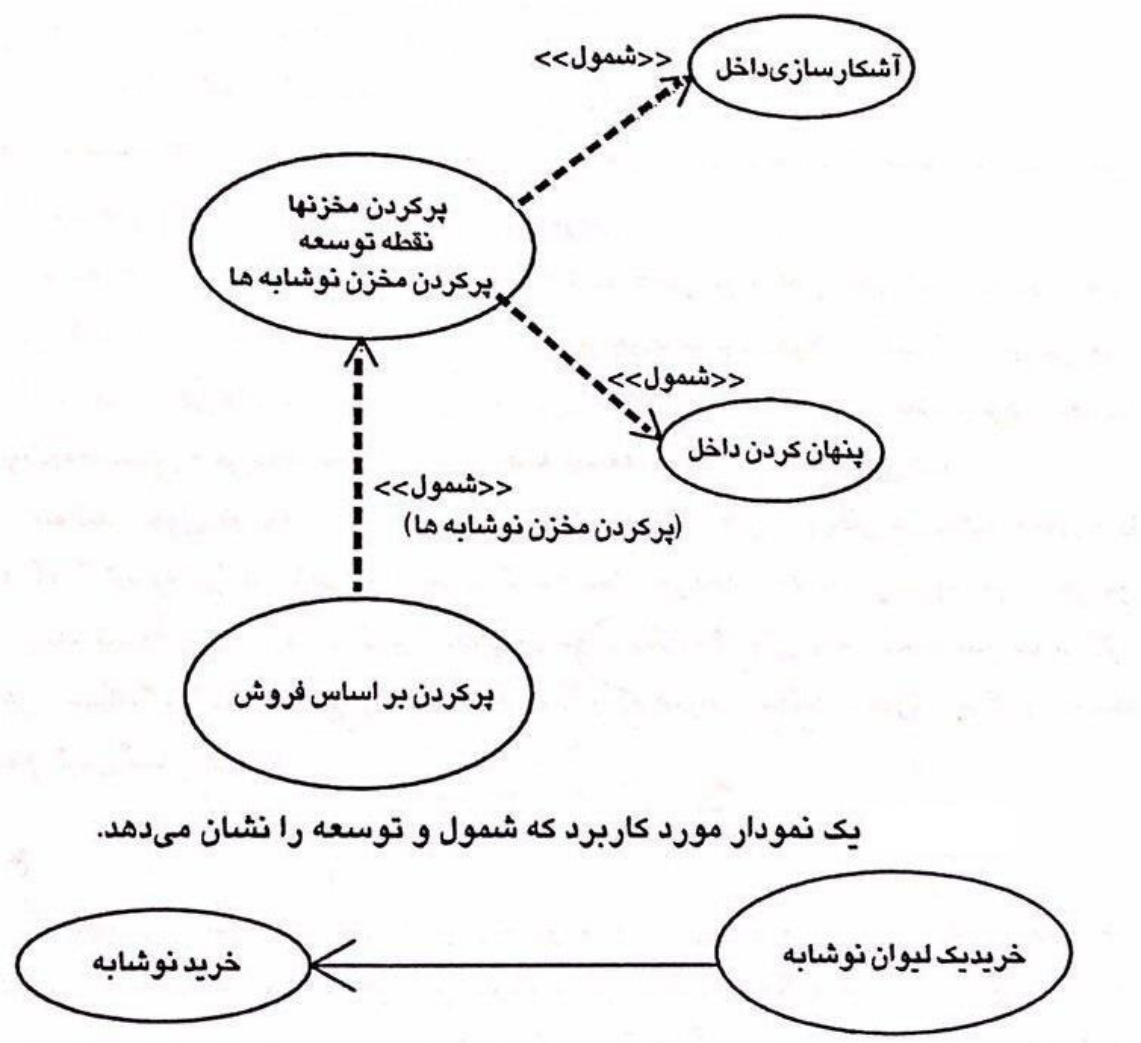
همانند شمول، توسعه را به وسیله یک خط وابسته (خط چین با پیکانی در نوک) به همراه یک کلیشه که «توسعه» را در داخل یک جفت گیومه نشان می دهد، مجسم می سازیم. در داخل مورد کاربرد پایه، توسعه زیر نام مورد کاربرد ظاهر می شود. شکل ۴-۷ ارتباط توسعه بیم کورد کاربرد «پر کردن دستگاه» و «دوباره پرکردن بر طبق فروش» را به همراه ارتباطات شمول، «پر کردن دستگاه» و «جمع کردن» نشان می دهد.

تعمیم کلاسها می توانند از کلاسهای دیگرارث ببرند.به همین صورت موارد کاربردنیز مـی تواننـد ازیکـدیگر ارث ببرنـد مـثلاً. درمیراث مورد کاربرد، مورد کاربرد فرزند،رفتار و معنی را از مورد کاربرد پدر به ارث می برد و آن را به رفتارهای خـود اضـافه می کند. هر کجا که از پدر استفاده می کنید، می توانید از فرزندان نیز استفاده کنید.

مثلاً، تصور کنید که مورد کاربرد «خرید یک لیوان نوشابه» از مورد کاربرد «خرید نوشابه» ارث می بـرد. فرزنـدان رفتارهـایی مثل «اضافه نمودن یخ» و یا «ترکیب انواع نوشابه» را اضافه می کنند.

می توانید تعمیم موارد کاربرد را همانند تعمیم کلاسها به مدل در می آورید. این کار را با یک خط توپری که دارای مثلث تـو خالی است و به پدر اشاره می کند، انجام دهیم به شکل ۵-۷ توجه کنید.

ارتباط تعمیم می توانند بین کنش گرها و موارد کاربرد وجود داشته باشد. مثلاً این امکان وجود دارد که هم نماینده تهیـه کننده و هم جمع کننده را به عنوان نماینده و کارگزار تهیه کننده نمایش دهید. اگر نماینـده را بـه نـام پرکننـده تغییـر نـام دهیم، پرکننده و جمع کننده هر دو فرزندان نماینده تهیه کننده هستند. همان گونه کـه در شـکل ۶-۷ بـه تصـویر کشـیده است.

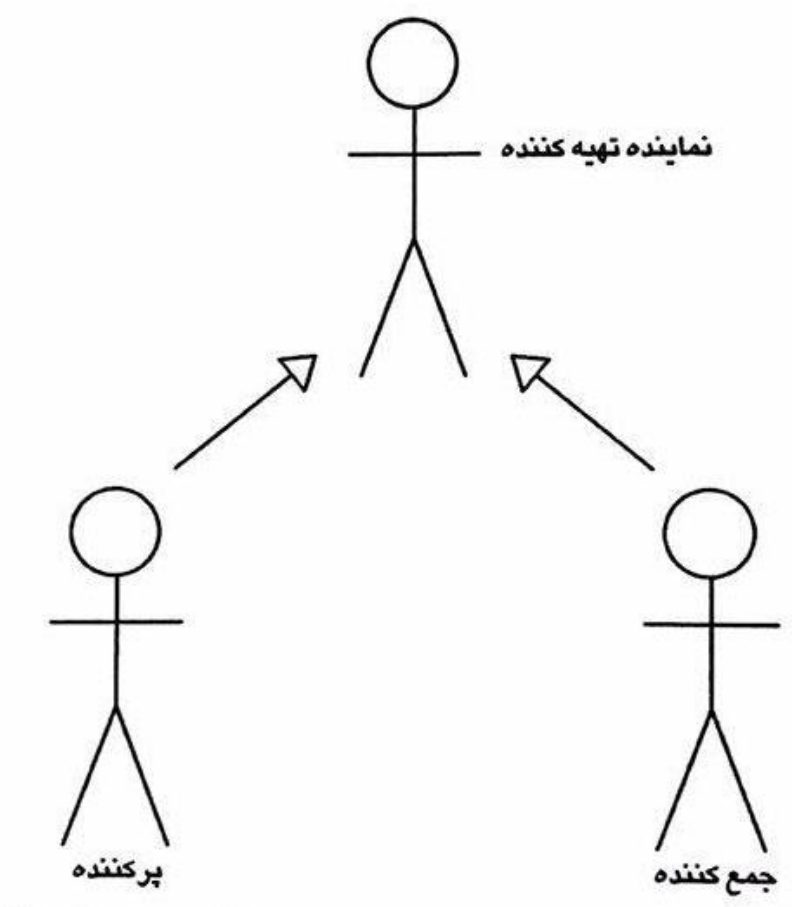


شکل - یک مورد کاربرد می تواند معنی و رفتار یک مورد کاربرد را ارث ببرد.

گروه بندی

در بعضی نمودارهی مورد کاربرد ممکن است تعداد زیادی موارد کاربرد داشته باشید و بخواهید آنها را سازمان دهی کنید. این موضوع زمانی اتفاق می افتد که یک سیستم، شامل تعدادی زیر سیستم باشد. امکان دیگر زمانی است که شما در حال مصاحبه با کاربران برای گرد آوری نیازهای سیستم هستید. هر نیاز احتمالا به صورت یک مورد کاربرد جداگانه به نمایش در می آید.

مستقیم ترین راه برای سازمان دهی آن است که موارد کاربرد مربوط به هم را در داخل یک بسته قرار دهیم. یک بسته همان گونه که قبلاً توضیح داده شد به صورت یک پوشه گوشه دار ظاهر خواهد شد. موادر کاربرد گروه بندی شده در داخل پوشه ظا هر خواهند شد.



شکل - همانند کلاسها و موارد کاربرد، کنش گران می توانند در یک ارتباط تعمیم باشند.

نمودارهای مورد کاربرد در روند تحلیل

در قسمتهای قبل، مثالی داده شد که با آن کار کردیم. حال وقت آن رسیده است که از موارد کاربرد در متن یک تحلیل سیستم استفاده شود.

مصاحبه با مشتریان روند کار را آغاز می کند. این مصاحبه ها نمودارهای کلاس را مشخص می کنند و پایه ای برای بالا رفتن دانش شما از محدوده و قلمروی سیستم (محیطی که قرار است مسائل آنجا را حل کنیم) خواهد بـود. بعـد از آنکـه شـما بـا اصطلاحات عمومی محیطی مشتری آشنا شدید و اتن را درک کردید، آماده صحبت با کاربران خواهید شد.