

• آشنایی با UML 2

مقدمه

شاید برای شما هم این سؤال پیش آمده باشد که چه تغییر مهمی در UML رخ داده است که پس از UML 1.5، UML 2.0 عرضه شد؟ آیا اضافه شدن دیاگرام‌های جدید (مثل Timing Diagram) یا بهبود دیاگرام‌های موجود (مانند افزودن امکانات بیشتر به Sequence Diagram) موجب این ارتقاء قابل توجه شده است؟ حقیقت این است که آنچه که موجب این ارتقاء نسخه قابل توجه از 1 به 2 شده است، فراتر از این جزئیات است. آنچه که تولید مدل‌گرا (Driven Development Model) نامیده می‌شود، که بر پایه سطح تجرد بالاتر و استفاده بیشتری از تولید خودکار کد نسبت به روش‌های سنتی قرار دارد، اثر قابل توجه خود در بهبود کیفیت نرم‌افزار و بهره‌وری تولید نشان داده است. از آنجاییکه نقش زبان مدل‌سازی برای موفقیت MDD بسیار مهم است، یک تجدید نظر عمده در زبان استاندارد UML انجام شده است که منجر به عرضه UML 2.0 گردیده است. در عین حال که چندین قابلیت جدید مدل‌سازی اضافه شده است – مانند قابلیت بیان دقیق‌تر معماری نرم‌افزار – خصوصیت غالب این بازبینی عمده، زیاد کردن دقت قابلیت تعریف زبان است که سطح بالاتری از خودکارسازی را فراهم می‌کند. در ادامه شرح خواهیم داد که UML2.0 چگونه به این موارد دست یافته است و سایر جنبه‌های مهم آنرا نیز بیان خواهیم کرد. همان‌گونه که می‌دانید UML بوسیله تولیدکنندگان بزرگ ابزارهای مدل‌سازی پذیرفته و پشتیبانی می‌شود، و بصورت یک بخش ضروری از دانش مهندسی نرم‌افزار درآمده است و در دانشگاه‌ها نیز تدریس می‌شود. همچنین نقش مهمی در مدل‌سازی نرم‌افزارهای پیچیده ایفا می‌کند. اما با وجود همه این مزایا همچنان مقاومت‌هایی در برابر استفاده از UML وجود دارد. دلایل زیادی برای این وضعیت وجود دارد، لکن یکی از مهمترین آنها این است که مدل‌های نرم‌افزار ممکن است در بعضی موارد بسیار نادقیق باشند و ارزش کاربردی هر مدلی با میزان دقت و صحت آن تناسب مستقیم دارد. چنانچه شما نتوانید به یک مدل از یک سیستم نرم‌افزار اعتماد کنید، بدتر از حالتی است که مدلی وجود نداشته باشد، زیرا ممکن است منجر به تصمیم‌گیری غلط شما شود. بنابراین بهترین راه‌حل افزایش ارزش مدل‌های نرم‌افزاری کم کردن فاصله میان آنها و سیستمی است که آنرا مدل کرده‌اند. جالب است بدانید – همانطور که در ادامه بیان خواهیم کرد – در مهندسی نرم‌افزار بیش از سایر رشته‌های مهندسی این کاهش فاصله امکان‌پذیر است.

Model-Driven Development

راه‌حل این معما اتصال دقیق یک مدل به معادل پیاده‌سازی نرم‌افزاری آن با استفاده از یک یا چند تبدیل مدل خودکار است. شاید بهترین مثال یک کامپایلر باشد، که یک برنامه که به زبان سطح بالا نوشته شده است را به متناظر سطح ماشین آن برنامه تبدیل می‌کند. مدل، در این حالت، نقش آن زبان سطح بالا را ایفا می‌کند که جزئیات غیر ضروری را نمایش می‌دهد. جالب است توجه کنید در هیچ یک از رشته‌های مهندسی دیگر نمی‌توانند این ارتباط قوی بین مدل و فرآورده مهندسی آن ایجاد کنند. زیرا فرآورده‌ای که شما آنرا مدل می‌کنید نرم‌افزار است و نه سخت‌افزار. یک مدل از هر نوع از فرآورده‌های فیزیکی (بعنوان مثال، یک اتومبیل، ساختمان، پل و موارد دیگر) هنگام مدل‌کردن نیاز به یک سری حذف جزئیات و مجردسازی است که بصورت غیر دقیق انجام می‌شود و هنگام ساخت یک فرآورده مهندسی از روی مدل مجرد نیز لازم است یک سری تبدیل‌های غیر دقیق انجام شود. ماهیت این تبدیل‌ها به دلیل عدم دقتی که دارند ممکن است مدل‌ها را به یک موجودیت ناکارا یا حتی مزاحم تبدیل کنند. حال آنکه در نرم‌افزار این تبدیل‌ها، بطور کلی، می‌توانند بصورت کاملاً دقیق و قاعده‌مند انجام شوند. پتانسیلی که در ورای این ترکیب قدرتمند تجرد و خودکارسازی وجود دارد منجر به ظهور تکنولوژی مدل‌سازی جدیدی به همراه روش‌های تولید متناظر با آن شده

است که با عنوان تولید مدل گر (model-driven development) شناخته می‌شود. ویژگی اصلی MDD این است که مدل‌ها به فرآورده اصلی طراحی نرم‌افزار بدل گشته‌اند، که منجر به انتقال تمرکز بیشتر از کد برنامه به مدل می‌شود. با دقت تر شدن مدل‌ها (که UML 2.0 این قابلیت را فراهم می‌آورد) سطح خودکارسازی تولید کد از روی مدل افزایش پیدا می‌کند و مزایای MDD بیشتر نمایان می‌شوند. لازم به ذکر است که پیش از این هم در عمل MDD مورد استفاده قرار گرفته است، لکن در حال حاضر به دلیل رشد و افزایش قابلیت تکنیک‌ها و استانداردها (مانند UML 2) این امکان بیشتر فراهم شده است و MDD بیشتر مورد توجه قرار گرفته است.

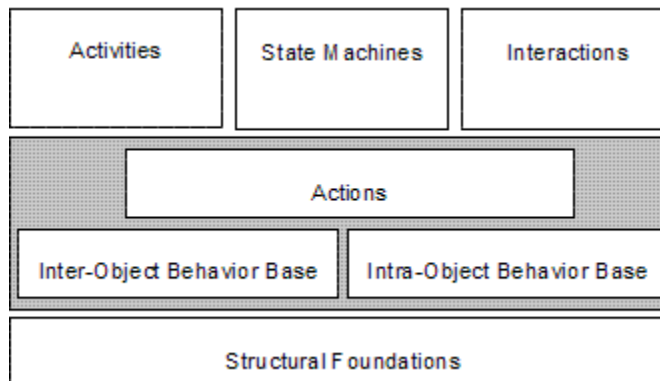
اهم موارد جدید در UML 2.0

توسعه‌های جدیدی که در UML 2.0 انجام شده است را می‌توان در این پنج دسته عمده گروه‌بندی کرد که در ادامه به ترتیب اهمیت بیان شده‌اند.

افزایش قابل توجه میزان دقت در تعاریف زبان

که نتیجه نیاز به پشتیبانی از خودکارسازی سطح بالاتری است که برای MDD لازم است. لازمه خودکارسازی رفع ابهام و عدم دقت از مدل‌ها (و در نتیجه از زبان مدل‌سازی) است تا برنامه‌های کامپیوتری بتوانند مدل‌های را تبدیل به کد کنند. یکی از اقداماتی که به منظور کمینه کردن ابهامات و افزایش دقت مدل انجام شده است استفاده از metamodel است. این مدل خصوصیات هر عنصر مدل‌سازی UML و ارتباط آن با سایر مفاهیم مدل‌سازی را تعریف می‌کند. این metamodel با استفاده از یک زیرمجموعه از عناصر UML - که بیشتر مفاهیم Class Diagram هستند و اصطلاحاً Meta-Object Facility (MOF) نامیده می‌شوند - تعریف شده است و بوسیله یک مجموعه از محدودیت‌های رسمی که به زبان Object Constraint Language (OCL) نوشته شده است پشتیبانی می‌شود. بطور خلاصه میزان دقت تعاریف زبان UML 2.0 با روش‌های زیر بطور قابل توجهی افزایش یافته است :

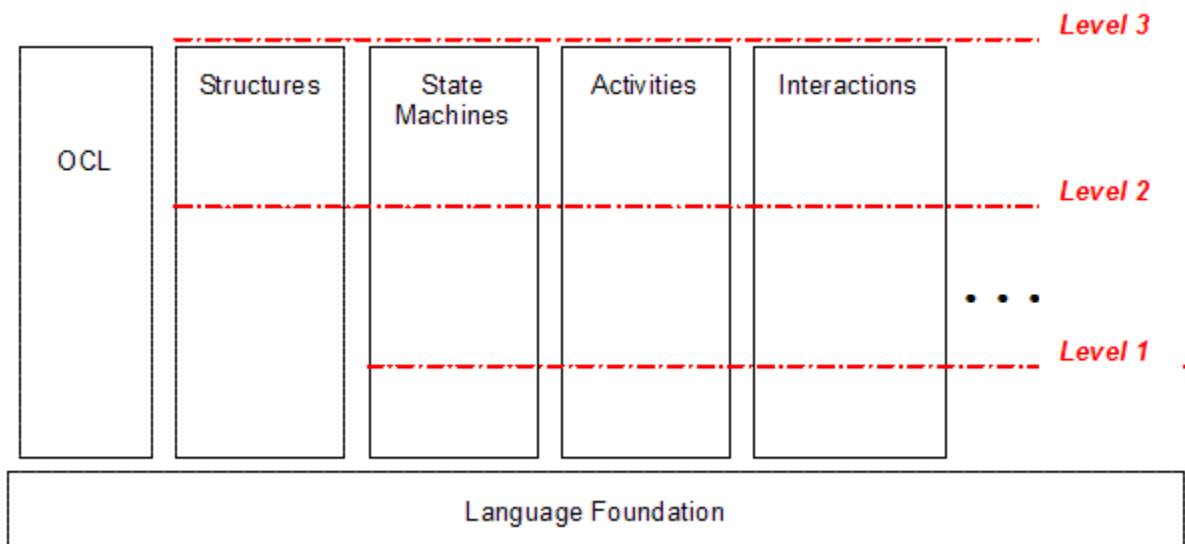
- یک سازماندهی مجدد عمده در فراساختار metamodel
- توصیف‌های معنایی توسعه یافته و دقیق‌تر
- یک چارچوب معنایی پویا و شفاف به منظور پرکردن خلاءهایی که در این زمینه وجود داشت



شکل 1 - چارچوب معنایی UML 2.0

بهبود سازماندهی زبان

با پیمانه‌ای (Modular) کردن زبان، علاوه بر اینکه برای کاربران جدید امکان شناخت و استفاده ساده تر خواهد بود، همکاری میان ابزارها را نیز تسهیل می‌کند. در واقع معماری بهتری برای زبان ایجاد شده است. با توجه به افزایش دقت UML 2.0، تعاریف زبان بزرگ‌تر شده‌اند و در نتیجه چنانچه از همان ساختار و معماری UML 1 برای آن استفاده می‌شد، فهم و استفاده از آن را بسیار مشکل می‌ساخت. به منظور مقابله با مشکل پیچیدگی زبان، UML 2.0 بصورت پیمانه‌ای درآمده است تا امکان استفاده انتخابی از پیمانه‌های زبان فراهم شود. شکل کلی این ساختار در شکل 2 نمایش داده شده است. همانطور که مشاهده می‌شود از یک پایه که شامل عناصر به اشتراک گذاشته شده است (مانند کلاس‌ها و روابط association)، که بر روی آن مجموعه‌ای از زیر-زبان‌ها یا عناصر زبان وجود دارد. هر کدام از این زیر-زبان‌ها برای مدل‌سازی یک قالب یا جنبه‌ی بخصوص مناسب هستند. این عناصر در Error! Reference source not found. نمایش داده شده‌اند. این عناصر افقی زبان بطور معمول از یکدیگر مستقل هستند و بنابراین شما می‌توانید آنها را بصورت مستقل استفاده کنید (برخلاف UML 1 که بعنوان مثال activity diagram بطور کامل بر روی state diagram قرار گرفته بود).



شکل 2- معماری زبان UML 2.0

علاوه بر این عناصر افقی زبان بصورت سلسله‌مراتبی در سه سطح سازمان‌دهی شده‌اند که سطح بالاتر قابلیت‌های بیشتری نسبت به سطح پایین‌تر دارد. به این ترتیب زبان از یک بعد دیگر نیز پیمانه‌ای است و شما را قادر می‌سازد که حتی در یک واحد زبان هم یک زیرمجموعه‌ی بخصوص را انتخاب کنید. این ساختار معماری زبان شما را قادر می‌سازد که تنها یک زیر مجموعه از UML را بیاموزید و بکار ببرید که مناسب کار شما است و به مرور زمان و کسب تجارب بیشتر می‌توانید با عناصر قدرتمندتر زبان آشنا شوید و در هنگام نیاز از آنها استفاده کنید.

جدول 1- عناصر زبان UML 2.0

Language Unit	Purpose
Actions	Foundation) modeling of fine-grained actions)
Activities	Data and control flow behavior modeling

Classes	Foundation) modeling of basic structures)
Components	Complex structure modeling for component technologies
Deployments	Deployment modeling
General Behaviors	Foundation) common behavioral semantic base) modeling and time
Information Flows	Abstract data flow modeling
Interactions	Inter-object behavior modeling
Models	Model organization
Profiles	Language customization
State Machines	Event-driven behavior modeling
Structures	Complex structure modeling
Templates	Pattern modeling
Use Cases	Informal behavioral requirements modeling

بهبود قابل توجه در توانایی برای مدل کردن سیستم‌های نرم‌افزاری بزرگ

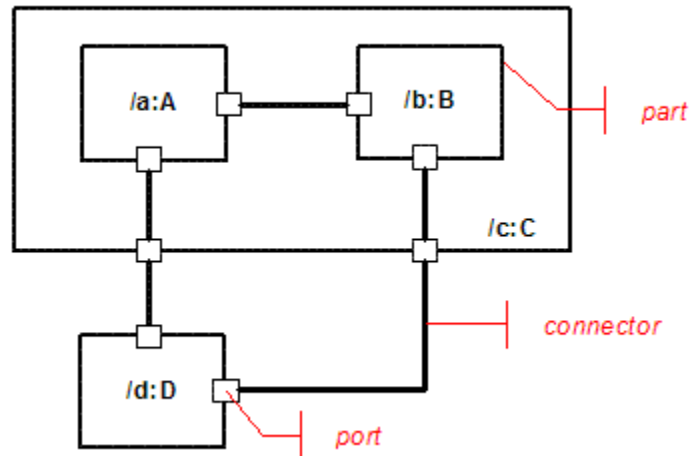
برخی از نرم‌افزارهای کاربردی مدرن تجمیع برنامه‌های کاربردی مستقل را در قالب سیستمی از سیستم‌ها نمایان می‌سازند و این روندی است ادامه دار که منجر به پیچیده‌تر شدن سیستم‌ها خواهد شد. برای پشتیبانی از چنین روندهایی، قابلیت‌های جدید انعطاف‌پذیر سلسله‌مراتبی به زبان اضافه شده است تا از مدل‌سازی نرم‌افزار در سطوح دلخواه پیچیدگی پشتیبانی کند. بطور کلی تعداد قابلیت‌های جدید اضافه شده به UML 2.0 کم است تا از بزرگ شدن زیاد زبان جلوگیری شود و بخش عمده‌ای از این قابلیت‌های جدید مدل‌سازی، گسترشی بر استفاده از قابلیت‌های جدید است که به شما امکان مدل‌سازی سیستم‌های نرم‌افزاری بزرگ را می‌دهد. علاوه‌براین، این گسترش‌ها با استفاده از یک راهکار پایه‌ای یکسان بدست آمده‌اند: استفاده بازگشتی از مجموعه یکسانی از مفاهیم در سطوح مختلف تجرد. بدین معنی که شما می‌توانید عناصر مدل‌سازی مربوط به یک گونه بخصوص را با یکدیگر ترکیب کنید تا واحدهایی ایجاد شود که بعنوان بلوک‌های سازنده در سطح بعدی تجرد مورد استفاده قرار گیرند. این وضعیت قابل مقایسه با روشی است که در برنامه‌نویسی procedure ها می‌توانند تا چندین سطح در داخل یکدیگر فراخوانی شوند(Procedure Call Nested). بطور مشخص، قابلیت‌های مدل‌سازی زیر بدین روش توسعه یافته‌اند:

- ساختارهای پیچیده (Complex Structures)
- فعالیت‌ها (Activities)
- تعامل‌ها (Interactions)
- ماشین‌های حالت (State machines)

سه تای اول از لیست بالا 90٪ قابلیت‌های جدید UML 2.0 را شکل می‌دهند.

ساختارهای پیچیده (Complex Structures)

به این منظور عناصر پایه‌ای ساختاری که **part** نامیده می‌شوند ممکن است یک یا چند درگاه (**port**) داشته باشند که با استفاده از کانال‌های ارتباطی که اتصال‌دهنده (**connector**) نامیده می‌شوند (همانگونه که در شکل 3 نمایش داده شده است) به یکدیگر متصل شده‌اند. این ساختار تجمعی می‌تواند در داخل عناصر سطح بالاتر کپسوله شود که **port** های خود را دارند و می‌توانند با عناصر سطح بالاتری متصل شوند و به همین ترتیب عناصر سطح بالاتری می‌توانند ساخته شوند.



شکل 3- مفاهیم مدل‌سازی ساختارهای پیچیده

فعالیت‌ها (Activities)

Activity ها در UML برای مدل کردن انواع مختلفی از جریان مورد استفاده قرار می‌گیرد: جریان **signal** یا **data**، و نیز جریان‌های **algorithmic** یا **procedural**. در UML 1 یک محدودیت عمده برای **Activity** ها وجود داشت و آن هم این بود که آنها بر پایه ماشین حالت قرار گرفته بودند و بنابراین در حوزه معنایی ماشین‌های حالت محدود شده بودند. در UML 2.0 پایه ماشین حالت با یک چارچوب معنایی عمومی دیگر که تمام این محدودیت‌ها را حذف کرده است جایگزین شده است. (در واقع، پایه معنایی با استفاده از **petri net colored** های عمومی شده جایگزین شده است). علاوه بر این از برخی استانداردهای صنعتی برای مدل‌سازی فرآیندهای کسب و کار مانند **BPEL4WS** الهام گرفته است.

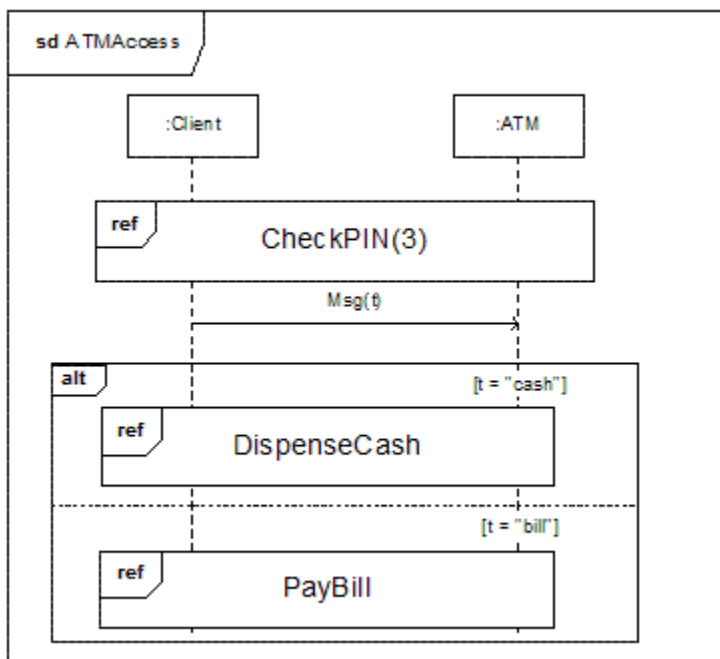
تعامل‌ها (Interactions)

تعامل‌های میان اشیاء در UML 1 یا با استفاده از **collaboration diagram** و یا با استفاده از **sequence diagram** نمایش داده می‌شدند. لکن متأسفانه دو قابلیت اساسی جا افتاده بودند:

1. امکان استفاده مجدد از توالی‌ها که ممکن بود در متن دنباله‌های دیگر تکرار شوند. بعنوان مثال، یک دنباله که هویت‌شناسی کاربر را انجام می‌دهد ممکن است در چندین جای مختلف یک برنامه کاربردی رخ دهد. بدون امکان بسته‌بندی این دنباله‌های تکرار شونده در عناصر جداگانه، لازم بود که آنها را چندین مرتبه بیان کنید که علاوه بر افزودن سربار اضافی، نگهداری مدل را نیز مشکل‌تر می‌کرد.

2 امکانات کافی برای مدل کردن جریان‌های پیچیده مختلف که در بازنمایی تعاملات سیستم‌های پیچیده رایج هستند. این قابلیت‌ها شامل تکرار کردن یک زیردنباله، مسیرهای اجرایی آلترناتیو، اجراهای همروند و مستقل از ترتیب، حلقه، شرط و موارد مشابه می‌شود.

مهمترین نوآوری در این زمینه معرفی تعامل‌ها (Interaction) بصورت یک واحد مدل‌سازی جداگانه نامگذاری شده است که امکان پارامتری کردن آنها نیز وجود دارد و بنابراین می‌توان هر سطح پیچیدگی دلخواهی از تعامل‌های میان اشیاء را در یک نمودار تعامل مدل کرد.

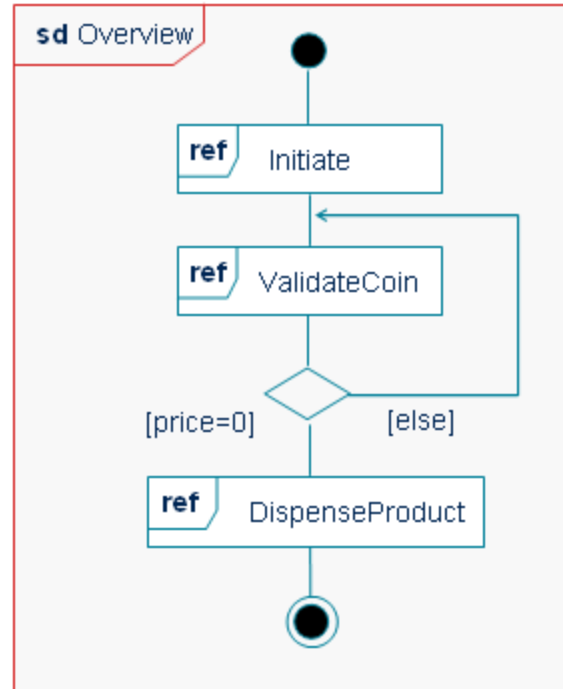


شکل 4 - مثالی از یک مدل تعاملی پیچیده

همانگونه که در شکل 4 نشان داده شده است شما می‌توانید این تعاملات بسته‌بندی شده را در تعاملات سطح بالاتر بصورت بازگشتی فراخوانی کنید.

ماشین‌های حالت (State Machine)

مهمترین قابلیت‌های ماشین‌های حالت در UML 2.0 اضافه شده است کاملاً مشابه موارد قبلی است. ایده اصلی این است که شما می‌توانید یک ماشین حالت پیچیده را کاملاً بصورت پیمانه‌ای مدل کنید که دارای نقاط مشخصی برای ورود و خروج است. به این ترتیب شما می‌توانید تجزیه داخلی یک ماشین حالت را بوسیله یک مجموعه از ماشین‌های حالت جداگانه و قابل استفاده انجام دهید. به این ترتیب توصیف یک الگوی رفتاری مشترک در چند حوزه مختلف به سادگی انجام می‌پذیرد.



شکل 5- نمونه‌ای از نمودار حالت در UML 2.0

بهبود پشتیبانی برای سفارشی سازی برای یک حوزه بخصوص
 تجارب عملی استفاده از UML ارزش مکانیزم‌های توسعه (Extension) آنرا نمایان ساخته است. در UML 1.x فقط از مکانیزم‌های توسعه stereotype و profile استفاده می‌شد، لکن در UML 2.0 مکانیزم توسعه جدید metamodel اضافه شده است که امکانات توسعه سطح بالاتری را فراهم می‌کند.
 تقویت، تطابق با اصول، روشنی و وضوح بیشتر برای مفاهیم مختلف مدل سازی
 زبان ساده تر و سازگارتر شده است. اقدامات جدید شامل تقویت و تثبیت مفاهیم و - در بعضی موارد - حذف مفاهیم تکراری،
 پالایش چندین تعریف و افزودن توضیحات متنی و مثال بوده است.

منبع: www.smhoseyni.com