زبان مدلسازی یکپارچه (UML)

علی‌رغم مزیتهای عمده‌ای که کارشناسان برای صنعت نرم‌افزار ایران برمی‌شمرند (از جمله نیروی انسانی مستعد و ارزان)، این صنعت در کشور ما با افت شدید کیفیت روبرو است. نه شرکتهای نرم‌افزاری معدود کشور، چندان به دنبال بکارگیری استانداردها و روشهای علمی طراحی و تولید نرم‌افزار هستند و نه فرهنگ مصرف ما کیفیت‌طلب است. مشتریان بزرگ و مشکل‌پسند (که معمولاً شرکتهای بزرگ دولتی هستند) نیز طالب خرید از خارج هستند و عدم کیفیت نرم‌افزارهای ساخت داخل را بهانه می‌کنند. یکی از مهمترین اقداماتی که دولت بایستی در جهت اصلاح این بازار آشفته انجام دهد، ترویج استانداردهای مدلسازی نرم‌افزار است. مدلهای استاندارد، تعریف دقیقتر نرم‌افزار را ممکن ساخته و از سوءتفاهمات جلوگیری می‌کنند. این مدلها همچنین ارزیابی نرم‌افزار را تسهیل نموده و ابزاری برای سوق دادن صنعت نرم‌افزار بسوی رعایت معیارهای کیفی و اصول مهندسی نرم‌افزار فراهم می‌آورند. مقالهٔ زیر به معرفی یکی از مهمترین این استانداردها پرداخته است:

زبان مدلسازی یکنواخت:

زبان مدلسازی یکنواخت یا Unified Modeling Language (UML)، یك زبان مدلسازی است که برای تحلیل وطراحی سیستمهای شی‌گرا بکار می‌رود. UML اولین بار توسط شرکت Rational ارائه شد و پس از آن از طرف بسیاری از شرکت‌های کامپیوتری و مجامع صنعتی و نرم‌افزاری دنیا مورد حمایت قرار گرفت؛ به طوریکه تنها پس از یك سال، توسط گروه Object Management Group، به عنوان زبان مدلسازی استاندارد پذیرفته شد. UML تواناییها و خصوصیات بارز فراوانی دارد که می‌تواند به طور گسترده‌ای در تولید نرم‌افزار استفاده گردد. در ادامهٔ این مقاله ابتدا به تاریخچهٔ UML و در ادامه به معرفی، ویژگیها و نمودارهای آن پرداخته می‌شود و در پایان، روند حرکت به سمت UML و اهمیت آن برای ایران، بررسی خواهد شد.

تاریخچهٔ UML :

دیدگاه شی‌گرایی (Object Oriented) از اواسط دهه ۱۹۷۰ تا اواخر دهه ۱۹۸۰ در حال مطرح شدن بود. در این دوران تلاشهای زیادی برای ایجاد روشهای تحلیل و طراحی شی‌گرا صورت پذیرفت. در نتیجهٔ این تلاشها بود که در طول ۵ سال یعنی ۱۹۸۹ تا ۱۹۹٤، تعداد متدولوژیهای شی‌گرا از کمتر از ۱۰ متدولوژی به بیش از ۵۰ متدولوژی رسید. تکثر متدولوژیها و زبانهای شی‌گرایی و رقابت بین اینها به حدی بود که این دوران به عنوان "جنگ متدولوژیها" لقب گرفت. از جمله متدولوژیهای پرکاربرد آن زمان می‌توان از Booch، OOSE، OMT، Fusion، Coad-Yourdan، Shlayer-Mellor وغیره نام برد. فراوانی و اشباع متدولوژیها و روشهای شی‌گرایی و نیز نبودن یك زبان مدلسازی استاندارد، باعث مشکلات فراوانی شده بود. از یك طرف کاربران از متدولوژیهای موجود خسته شده بودند، زیرا مجبور بودند از میان روشهای مختلف شبیه به هم که تفاوت کمی در قدرت و قابلیت داشتند یکی را انتخاب کنند. بسیاری از این روشها، مفاهیم مشترك شی‌گرایی را در قالبهای مختلف بیان می‌کردند که این واگرایی و نبودن توافق میان این زبانها، کاربران تازه‌کار را از دنیای شی‌گرایی زده می‌کرد و آنها را از این حیطه دور می‌ساخت. عدم وجود یك زبان استاندارد، برای فروشندگان محصولات نرم‌افزاری نیز مشکلات زیادی ایجاد کرده بود.

اولین تلاشهای استانداردسازی از اکتبر ۱۹۹٤ آغاز شد، زمانی که آقای Rumbaurgh صاحب متدولوژی OMT به آقای Booch در شرکت Rational پیوست و این دو با ترکیب متدولوژیهای خود، اولین محصول ترکیبی خود به نام "روش یکنواخت" را ارائه دادند. در سال ۱۹۹۵ بود که با اضافه شدن آقای Jacobson به این دو، روش یکنواخت ارائه شده با روش OOSE نیز ترکیب شد واین خود سبب ارائهٔ UML نسخهٔ ۰.۹ در سال ۱۹۹٦ گردید. سپس این محصول به شرکتهای مختلفی در سراسر جهان به صورت رایگان ارائه شد و استقبال شدید شرکت‌ها از این محصول و تبلیغات گسترده شرکت Rational، سبب آن شد که گروه OMG، نسخهٔ ۱.۰ UML را به عنوان زبان مدلسازی استاندارد خود بپذیرد. تلاشهای تکمیلی UML استاندارد ادامه پیدا کرد و نسخهٔ ۱.۱ آن در سال ۱۹۹۷ و نسخه ۱.۳ آن در

سال ۱۹۹۹ ارائه گردید.

UML چیست؟

UML یا زبان مدلسازی یکنواخت، زبانی است برای مشخص کردن (Specify)، مصورسازی (Visualize)، ساخت (Construction) و مستندسازی (Documenting) سیستمهای نرمافزاری و غیر نرمافزاری و نیز برای مدلسازی سیستمهای تجاری. اما چرا مدل و مدلسازی؟

ایجاد یك مدل برای سیستمهای نرمافزاری قبل از ساخت یا بازساخت آن، به اندازه داشتن نقشه برای ساختن یك ساختمان ضروری و حیاتی است. بسیاری از شاخه‌های مهندسی، توصیف چگونگی محصولاتی که باید ساخته شوند را ترسیم می‌کنند و همچنین دقت زیادی می‌کنند که محصولاتشان طبق این مدلها و توصیفها ساخته شوند. مدلهای خوب و دقیق در برقراری یك ارتباط کامل بین افراد پروژه، نقش زیادی می‌توانند داشته باشند. شاید علت مدل کردن سیستمهای پیچیده این باشد که تمامی آن را نمی‌توان یکباره مجسم کرد، بنابراین برای فهم کامل سیستم و یافتن و نمایش ارتباط بین قسمتهای مختلف آن، به مدلسازی می‌پردازیم. UML زبانی است برای مدلسازی یا ایجاد نقشه تولید نرمافزار.

به عبارت دیگر، یك زبان، با ارائه یك فرهنگ لغات ویك مجموعه قواعد، امکان می‌دهد که با ترکیب کلمات این فرهنگ لغات و ساختن جملات، با یکدیگر ارتباط برقرار کنیم. یك زبان مدلسازی، زبانی است که فرهنگ لغات و قواعد آن بر نمایش فیزیکی و مفهومی آن سیستم متمرکزند. برای سیستمهای نرمافزاری نیاز به یك زبان مدلسازی داریم که بتواند دیدهای مختلف معماری سیستم را در طول چرخهٔ تولید آن، مدل کند.

فرهنگ واژگان و قواعد زبانی مثل UML به شما می‌گویند که چگونه یك مدل را بسازید و یا چگونه یك

مدل را بخوانید. اما به شما نمی‌گویند که در چه زمانی، چه مدلی را ایجاد کنید. یعنی UML فقط یك زبان نمادگذاری (Notation) است نه یك متدولوژی. یك زبان نمادگذاری شامل نحوة ایجاد و نحوة خواندن یك مدل می‌باشد، اما یك متدولوژی بیان می‌کند که چه محصولاتی باید در چه زمانی تولید شوند و چه کارهایی با چه ترتیبی توسط چه کسانی، با چه هزینه‌ای، در چه مدتی و با چه ریسکی انجام شوند.

ویژگیهای UML :

UML دارای ویژگیهای بارز فراوانی است که در این قسمت به آنها می‌پردازیم. UML یك زبان مدلسازی است اما چیزی فراتر از چند نماد گرافیکی است. بطوریکه در ورای این نمادها، یك سمانتیك (معناشناسی) قوی وجود دارد، بطوریکه یك تولیدکننده می‌تواند مدلهایی تولید کند که تولیدکننده‌های دیگر و یا حتی یك ماشین آن را بخواند و بفهمد. بنابراین یکی دیگر از نقش‌های مهم UML "تسهیل ارتباط" بین اعضای پروژه و یا بین تولیدکنندگان مختلف می‌باشد. این ارتباط بسیار مهم است. شاید دلیل اصلی اینکه تولید نرم‌افزار به صورت فریبنده‌ای دشوار است، همین عدم ارتباط مناسب بین اعضای پروژه باشد و اگر در تولید نرم‌افزار، بین اعضای پروژه گزارشهای هفتگی و مداوم وجود داشته باشد، بسیاری از این دشواریها برطرف خواهد شد.

البته این را هم باید در نظر گرفت که UML کمی پیچیده است و این به خاطر آن است که سعی شده است نمودارهایی فراهم شود که در هر موقعیتی و با هر ترتیبی قابل استفاده باشند. دلیل دیگر پیچیدگی از آنجا ناشی می‌شود که UML ترکیبی است از زبانهای مختلف، که برای حفظ سازگاری و جمع کردن خصوصیات مثبت آنها، ناگزیر از پذیرش این پیچیدگی می‌باشد.

UML موفقیت طرح را تضمین نمی‌کند، اما در عین حال خیلی چیزها را بهبود می‌بخشد. به عنوان مثال استفاده از UML، تا حد زیادی، هزینه‌های ثابتی نظیر آموزش و استفاده مجدد از ابزارها را در هنگام

ایجاد تغییر در سازمان و طرحها کاهش می‌دهد.

مساله دیگر اینکه، UML یك زبان برنامه‌نویسی بصری (visual) نیست، اما مدلهای آن را می‌توان مستقیماً به انواع زبانهای مختلف ارتباط داد. یعنی امکان نگاشت از مدلهای UML به کد زبانهای برنامه‌نویسی مثل Java و VC وجود دارد که به این عمل "مهندسی روبه‌جلو" می‌گویند. عکس این عمل نیز ممکن است؛ یعنی این امکان وجود دارد که شما بتوانید از کد یك برنامه زبانی شی‌گرا، مدلهای UML معادل آن را بدست آورید. به این عمل "مهندسی معکوس" می‌گویند. مهندسی روبه‌جلو و معکوس از مهمترین قابلیتهای UML به شمار می‌روند، البته نیاز به ابزار Case مناسبی دارید که از این مفاهیم پشتیبانی‌کنند.

اگر با زبانهای مدلسازی دیگر کار کرده باشید، برای کار با UML مشکل چندانی نخواهید داشت. اما برای شروع کار با UML به عنوان اولین زبان مدلسازی، بهتر است فقط با نمودارهای خاصی کار کنید. برای این کار بهتر است ابتدا با نمودارهای مورد کاربرد و تعامل کار کنید و پس از مدتی کار و آشنا شدن با ویژگیهای اولیة آن، به یادگیری و استفاده از نمودارها واجزای دیگر بپردازید. در مقایسه با زبانهای مدلسازی دیگر مثلER و زبان فلوچارتی DR، زبان UML نمودارهای قویتر و قابل‌فهمتری را ارائه می‌دهدکه شامل تمامی مراحل چرخة حیات تولید نرم‌افزار (تحلیل، طراحی، پیاده‌سازی و تست) می‌شود.

یکی دیگر از ویژگیهای مهم UML این است که مستقل از متدولوژی یا فرایند تولید نرم‌افزار می‌باشد و این بدان معنی است که شما برای استفاده از UML، نیاز به استفاده از یك متدولوژی خاص ندارید و می‌توانید طبق متدولوژی‌های قبلی خود عمل کنید با این تفاوت که مدلهایتان را با UML نمایش می‌دهید. البته مستقل‌بودن از متدولوژی و فرایند تولید، یك مزیت برای UML می‌باشد؛ زیرا بسیاری از انواع پروژه‌ها و سیستمها نیاز به متدولوژی خاص خود دارند. اگر UML در پی پیاده کردن همة اینها بر می‌آمد، یا بسیار پیچیده می‌شد و یا استفاده خود را محدود می‌کرد. البته متدولوژیهایی براساس

UML در حال شکل‌گیری می‌باشند.

از دیگر ویژگیهای UML می‌توان به پشتیبانی از مفاهیم سطح بالای شی‌گرایی مثل Collaboration، Framework، Pattern و Component اشاره کرد. همچنین UML با استفاده از یك سری مکانیزمهای گسترش‌پذیر امکان می‌دهد که بتوان زبانهای مدلسازی جدیدتری (با گسترش مفاهیم پایه‌ای موجود) ایجادکرد.

نمودارهای UML :

در این بخش به معرفی نمودارهای UML می‌پردازیم وعلاقمندان به آشنایی بیشتر را، دعوت به مطالعه مراجع معرفی شده، می‌نماییم:

نمودار کلاس (Class Diagram):

این نمودار،کلاسها، واسطها و همکاری و روابط بین آنها را نمایش می‌دهد. و نمودار اصلی و مرکزی UML می‌باشد. که بیان‌کننده ساختار ایستای سیستم نرم‌افزاری می‌باشد.

نمودار اشیاء (Object Diagram):

این نمودار، اشیاء سیستم و روابط بین آنها را نمایش می‌دهد. در واقع یك تصویر لحظه‌ای از نمودار کلاس می‌باشد.

نمودار موردکاربرد (Usercase Diagram):

این نمودار، تعامل کاربران خارجی و سیستم را مدل می‌کند و از جهاتی شبیه نمودار سطح صفر DFD می‌باشد که جنبه‌های رفتاری سیستم را نمایش می‌دهد. این نمودار نقطه ورودی برای تمامی نمودارهای دیگری است که به تشریح نیازمندیها و معماری و پیاده‌سازی سیستم می‌پردازند.

نمودارهای تعامل (Interaction Diagram):

این نمودارها، بیان کننده تعامل هستند که شامل اشیاء مختلف و روابط بین آنها و همچنین پیغامهایی که بینشان رد و بدل می‌شود می‌باشند. این نمودارها جنبه‌های پویای یك سیستم را مدل می‌کنند و خود بر دو نوعند: نمودار توالی(Sequence Diagram) که ترتیب زمانی تعامل‌ها را نشان می‌دهد و نمودار همکاری(Collaboration Diagram) که تاکید بر نمایش ساختاری تعامل‌ها دارد.

نمودارحالت (Statechart Diagram):

این نمودار، بیان‌کننده جنبه‌های رفتاری سیستم می‌باشد و در واقع توصیف رسمی یك کلاس بوده که شامل حالات، انتقال بین حالات، رخدادها و فعالیتها می‌باشد. از این نمودارها برای نمایش دادن چرخه حیات اشیاء یك کلاس خاص نیز می‌توان استفاده کرد.

نمودار فعالیت(Diagram Activity):

این نمودار، نوع خاصی است از نمودار حالت، که انتقال جریان از یك فعالیت به فعالیت دیگر را نمایش می‌دهد. این نمودار جنبه‌های پویای یك سیستم را نمایش می‌دهد. در واقع حالات این نمودار، گامهای ترتیبی انجام یك عمل را نمایش می‌دهند.

نمودار اجزاء(Component Diagram):

از جمله نمودارهای پیاده‌سازی می‌باشد و سازماندهی و روابط بین مجموعه‌ای از اجزاء را نمایش می‌دهد. این نمودار، جنبه های ایستای پیاده‌سازی یك سیستم را مدل می‌کند.

نمودار به‌کارگماری(Deployment Diagram):

پیکربندی گره‌های پردازشی زمان اجرا را نمایش می‌دهد. که برای مدل کردن جنبه‌های ایستای به‌کارگماری یك معماری بکار می‌رود. همچنین نمایش‌دهندهٔ اجزای استفاده‌شده زمان اجرا مثل کتابخانه‌های DLL، فایل‌های اجرایی، کدهای مبدا و روابط بین آنها می‌باشد.

البته این نمودارها تمام نمودارهای UML نیستند بلکه بسته به نیاز و با کمك ابزارهای Case میتوان نمودارهای دیگری نیز تعریف و استفاده کرد.

روند حرکت به سمت UML در جهان:

قبل از ارائه UML، زبان مدلسازی استانداردی وجود نداشت و استفاده‌کنندگان مجبور بودند از میان زبانهای مختلف موجود که هیچیك تقریباً کامل نبودند و تفاوتهایی با هم داشتند، یکی را انتخاب کنند. تفاوتهای زبانهای مدلسازی، چندان قدرت مدلسازی را افزایش نداده بود، اما در عوض باعث افول صنعت شی‌گرایی و سردرگمی کاربران شده بود. در چنین شرایطی طبیعی بود که استقبال زیادی از یك زبان مدلسازی استاندارد که ویژگیهای بارز زیادی داشت، بشود. بسیاری از شرکتها در همان اوایل کار به UML روی آوردند و تعداد دیگری نیز پس از تثبیت UML، آن را به عنوان استراتژی تولید ومستندسازی خود پذیرفتند.

OMG که کنسرسیومی است متشکل از ۷۰۰ شرکت معتبر آمریکا، از UML حمایت کرد و آن را به

عنوان زبان مدلسازی استاندارد خود اعلام کرد. البته علاوه بر استاندارد شدن، حمایت جداگانه شرکت‌های بزرگ دنیا مثل Hewlett-Packard، I-Logix، Microsoft، IBM، Oracle و بسیاری دیگر، خود سبب افزایش کاربرد آن در محافل صنعتی و نرم‌افزاری دنیا گردید. امروزه نیز با ارائه نسخه ۱.۳ و رفع مشکلات گذشته، روز به روز بر کاربران آن افزوده می‌شود.

روند حرکت به سمت UML در ایران:

در ایران حرکت برخی شرکتها به سمت UML سریعتر انجام شد؛ بطوریکه در همان زمان استاندارد شدن UML در سال ۱۹۹۷، شرکتهایی در ایران، این ابزار را به عنوان استاندارد خود پذیرفتند و از آن در تولید محصولات خود استفاده کردند.

یکی از مشکلات پذیرش این زبان در ایران، مقاومتهایی است که در رابطه با خود شی‌گرایی مطرح می‌شود. البته نظیر این مقاومتها در دنیا نیز وجود داشت و سرو صداهای بسیاری را سبب شد. اما تا قبل از ظهور UML و با ارائه متدهای فراوان شی‌گرایی، این مشکل تا حدودی حل شده بود.

با توجه به روند حرکت شتابان به سمت UML در دنیا و با توجه به اهمیت استانداردسازی برای صنعت نرم‌افزار کشور، حرکت هرچه‌سریعتر به سوی این فناوری در کشور توصیه می‌شود.

اهمیت ترویج UML در کشور:

در کشور ما شرکتهای نرم‌افزاری که روشهای علمی طراحی و مهندسی نرم‌افزار را به کار برند بسیار کمیاب هستند. در واقع رقابت بین شرکتها بیشتر بر سر کاهش قیمت است و نه بهبود کیفیت. ممکن است تصور شود عامل اصلی بروز این مشکل، فرهنگ مصرف غلط در کشور است و لذا حل مشکل نیز به دست مصرف‌کنندگان است. اما بایستی از خود پرسید که مصرف‌کنندگان چگونه خواهند

توانست يك محصول نرم‌افزاری را ارزیابی کرده و انتظارات خود را به طور دقیق مطرح نمایند؟ در این زمینه دولتها وظایف مهمی دارند و می‌توانند ابزارهای لازم را فراهم نمایند.

هرچند UML يك استاندارد برای تشخیص کیفیت نرم‌افزارها نیست ولی استانداردی برای مدلسازی نرم‌افزار است ولذا مراحل مختلف تعریف، طراحی و حتی تست نرم‌افزار را تسهیل نموده و کار تیمی و ارزیابی ناظران خارجی را آسان و ممکن می‌نماید. اگر استفاده از UML در تولید نرم‌افزار در کشور به يك فرهنگ تبدیل گردد، گام بزرگی به سوی دقت، کیفیت، مستندسازی و رعایت اصول مهندسی نرم‌افزار برداشته شده است.