UML 2.0

مقدمه

شايد براي شما هم اين سؤال پيش آمده باشد كه چه تغيير مهمي در UML رخ داده است كه پس از UML 1.5، UML 2.0 عرضه شد؟ آيا اضافه شدن دياگرام هاي جديد (مثل TimingDiagram)يا بهبود دياگرام هاي موجودمانند افزودن امكانات بيشتر به SequenceDiagramموجب اين ارتقاء قابل توجه شده است؟ حقيقت اين است كه آنچه كه موجب اين ارتقاء قابل توجه نسخه، از ۱ به ۲ شده است، فراتر از اين جزئيات است. آنچه كه توليد مدل گرا (Model Driven Development) ناميده مي شود، كه بر پايه سطح تجرد بالاتر و استفاده بيشتري از توليد خودكار كد نسبت به روش هاي سنتي قرار دارد، اثر قابل توجه خود در بهبود كيفيت نرم افزار و بهره وري توليد نشان داده است. از آنجايي كه نقش زبان مدل سازي براي موفقيت MDD بسيار مهم است، يك تجديد نظر عمده در زبان استاندارد UML انجام شده است كه منجر به عرضه UML 2.0 گرديده است.

درعين حال كه چندين قابليت جديد مدلسازي اضافه شده است – مانند قابليت بيان دقيق‌تر معماري نرم افزار- خصوصيت غالب اين بازبيني عمده، زياد كردن دقت قابليت تعريف زبان است كه سطح بالاتري از خودكارسازي را فراهم مي‌كند. در ادامه شرح خواهيم داد كه UML2.0 چگونه به اين موارد دست يافته است و ساير جنبه‌هاي مهم آن را نيز بيان خواهيم كرد. همان گونه كه مي دانيد UML بوسيله توليدكنندگان بزرگ ابزارهاي مدلسازي پذيرفته و پشتيباني مي‌شود، و بصورت يك بخش ضروري از دانش مهندسي نرم‌افزار درآمده است و در دانشگاه‌ها نيز تدريس مي‌شود. همچنين نقش مهمي در مدلسازي نرم‌افزارهاي پيچيده ايفا مي‌كند. (توضيحات بيشتر در ساير مقالات سايت ميكرو رايانه) اما با وجود همه اين مزايا همچنان مقاومت هايي در برابر استفاده از UML وجود دارد. دلايل زيادي براي اين وضعيت وجود دارد، لكن يكي از مهمترين آنها اين است كه مدل‌هاي نرم‌افزار ممكن است در بعضي موارد بسيار نادقيق باشند و ارزش كاربردي هر مدلي با ميزان دقت و صحت آن تناسب مستقيم دارد. چنانچه شما نتوانيد به يك مدل از يك سيستم نرم‌افزار اعتماد كنيد، بدتر از حالتي است كه مدلي وجود نداشته باشد، زيرا ممكن

است منجر به تصميم‌گيري غلط شما شود. بنابراين بهترين راه‌حل افزايش ارزش مدل‌هاي نرم‌افزاري كم كردن فاصلة ميان آنها و سيستمي است كه آنرا مدل كرده‌اند. جالب است بدانيد - همانطور كه در ادامه بيان خواهيم كرد- در مهندسي نرم‌افزار بيش از ساير رشته‌هاي مهندسي اين كاهش فاصله امكان‌پذير است .

Model-Driven Development

راه‌حل اين معما اتصال دقيق يك مدل به معادل پياده‌سازي نرم‌افزاري آن با استفاده از يك يا چند تبديل مدل خودكار است. شايد بهترين مثال يك كامپايلر باشد، كه يك برنامه كه به زبان سطح بالا نوشته شده است را به متناظر سطح ماشين آن برنامه تبديل مي‌كند. مدل، در اين حالت، نقش آن زبان سطح بالا را ايفا مي‌كند كه جزئيات غير ضروري را نمايش مي‌دهد. جالب است توجه كنيد در هيچ يك از رشته‌هاي مهندسي ديگر نمي‌توانند اين ارتباط قوي بين مدل و فرآورده مهندسي آن ايجاد كنند. زيرا فرآورده‌اي كه شما آن را مدل مي‌كنيد نرم‌افزار است و نه سخت‌افزار.

يك مدل از هر نوع از فرآورده‌هاي فيزيكي (به عنوان مثال، يك اتومبيل، ساختمان، پل و موارد ديگر (هنگام مدل‌كردن نياز به يك‌سري حذف جزئيات و مجردسازي است كه بصورت غير دقيق انجام مي‌شود و هنگام ساخت يك فرآورده مهندسي از روي مدل مجرد نيز لازم است يك سري تبديل‌هاي غير دقيق انجام شود. ماهيت اين تبديل‌ها به دليل عدم دقتي كه دارند ممكن است مدل‌ها را به يك موجوديت ناكارا يا حتي مزاحم تبديل كنند. حال آنكه در نرم‌افزار اين تبديل‌ها، بطور كلي، مي‌توانند بصورت كاملا دقيق و قاعده‌مند انجام شوند. پتانسيلي كه در وراي اين تركيب قدرتمند تجرد و خودكارسازي وجود دارد منجر به ظهور تكنولوژي مدل‌سازي جديدي به همراه روش‌هاي توليد متناظر با آن شده است كه با عنوان توليد مدل‌گر (model-driven development)شناخته مي‌شود.

ويژگي اصلي MDD اين است كه مدل‌ها به فرآورده اصلي طراحي نرم‌افزار بدل گشته‌اند، كه منجر به انتقال تمركز بيشتر از كد برنامه به مدل مي‌شود. با دقيق‌تر شدن مدل‌ها (كه UML 2.0 اين قابليت را فراهم مي‌آورد) سطح خودكارسازي توليد كد از روي مدل افزايش پيدا مي كند و مزاياي MDD بيشتر نمايان مي‌شوند. لازم به ذكر است كه پيش از اين هم در عمل MDD مورد استفاده قرار گرفته است، لكن در حال حاضر به دليل رشد و افزايش قابليت تكنيك‌ها و استانداردها (مانند (UML 2 اين امكان بيشتر

فراهم شده است و MDD بيشتر مورد توجه قرار گرفته است .

اهم موارد جديد در UML 2.0

توسعه هاي جديدي كه در UML 2.0 انجام شده است را مي‌توان در اين پنج دسته عمده گروه‌بندي كرد كه در ادامه به ترتيب اهميت بيان شده‌اند .

افزايش قابل توجه ميزان دقت در تعاريف زبان

كه نتيجه نياز به پشتيباني از خودكارسازي سطح بالاتري است كه براي MDD لازم است. لازمه خود كارسازي رفع ابهام و عدم دقت از مدل‌ها (و در نتيجه از زبان مدل‌سازي) است تا برنامه‌هاي كامپيوتري بتوانند مدل‌ها را تبديل به كد كنند. يكي از اقداماتي كه به منظور كمينه كردن ابهامات و افزايش دقت مدل انجام شده است استفاده از metamodel است. اين مدل خصوصيات هر عنصر مدل سازي UML و ارتباط آن با ساير مفاهيم مدل‌سازي را تعريف مي‌كند. اين metamodel با استفاده از يك زيرمجموعه از عناصر UML - كه بيشتر مفاهيم Class Diagram هستند و اصطلاحا Meta-Object Facility (MOF) ناميده مي‌شوند - تعريف شده است و بوسيله يك مجموعه از محدوديت هاي رسمي كه به زبان ObjectConstraint Language يا OCL نوشته شده است پشتيباني مي‌شود. به طور خلاصه ميزان دقت تعاريف زبان UML 2.0 با روش‌هاي زير بطور قابل توجهي افزايش يافته است :

-يك سازماندهي مجدد عمده در فراساختار metamodel

-توصيف‌هاي معنايي توسعه يافته و دقيق‌تر

-يك چارچوب معنايي پويا و شفاف به منظور پركردن خلاءهايي كه در اين زمينه وجود داشت

بهبود سازماندهي زبان

با پيمانه‌اي (Modular) كردن زبان، علاوه بر اينكه براي كاربران جديد امكان شناخت و استفاده ساده‌تر خواهد بود، همكاري ميان ابزارها را نيز تسهيل مي‌كند. در واقع معماري بهتري براي زبان ايجاد شده است .با توجه به افزايش دقت UML 2.0 ، تعاريف زبان بزرگ تر شده اند و در نتيجه چنانچه از همان

ساختار و معماري UML 1 براي آن استفاده مي‌شد، فهم و استفاده از آن را بسيار مشكل مي‌ساخت .)توضيحات بيشتر در ساير مقالات سايت ميكرو رايانه) به منظور مقابله با مشكل پيچيدگي زبان، UML 2.0 بصورت پيمانه‌اي درآمده است تا امكان استفاده انتخابي از پيمانه‌هاي زبان فراهم شود. شكل كلي اين ساختار در شكل 2 نمايش داده شده است.

همانطور كه مشاهده مي‌شود از يك پايه كه شامل عناصر به اشتراك گذاشته شده استمانند كلاس ها و روابط association ، كه بر روي آن مجموعه‌اي از زير-زبان‌ها يا عناصر زبان وجود دارد. هر كدام از اين زير-زبان‌ها براي مدل‌سازي يك قالب يا جنبه بخصوص مناسب هستند. اين عناصر در Error! Reference source not found.نمايش داده‌شده اند. اين عناصر افقي زبان بطور معمول از يكديگر مستقل هستند و بنابراين شما مي‌توانيد آنها را بصورت مستقل استفاده كنيد (برخلاف UML 1 كه بعنوان مثال activity diagramبطور كامل بر روي state diagram قرار گرفته بود).

علاوه‌بر اين عناصر افقي زبان بصورت سلسله‌مراتبي در سه سطح سازمان‌دهي شده‌اند كه سطح بالاتر قابليت‌هاي بيشتري نسبت به سطح پايين‌تر دارد. به اين ترتيب زبان از يك بعد ديگر نيز پيمانه‌اي است و شما را قادر مي‌سازد كه حتي در يك واحد زبان هم يك زيرمجموعة بخصوص را انتخاب كنيد. اين ساختار معماري زبان شما را قادر مي سازد كه تنها يك زير مجموعه از UML را بياموزيد و بكار ببريد كه مناسب كار شما است و به مرور زمان و كسب تجارب بيشتر مي‌توانيد با عناصر قدرتمندتر زبان آشنا شويد و در هنگام نياز از آنها استفاده كنيد .

بهبود قابل توجه در توانايي براي مدل كردن سيستم‌هاي نرم‌افزاري بزرگ

برخي از نرم افزارهاي كاربردي مدرن تجميع برنامه‌هاي كاربردي مستقل را در قالب سيستمي از سيستم‌ها نمايان مي‌سازند و اين روندي است ادامه دار كه منجر به پيچيده‌تر شدن سيستم‌ها خواهد شد. براي پشتيباني از چنين روندهايي، قابليت‌هاي جديد انعطاف‌پذير سلسله‌مراتبي به زبان اضافه شده است تا از مدل‌سازي نرم‌افزار (توضيحات بيشتر در ساير مقالات سايت ميكرورايانه) در سطوح دلخواه

پيچيدگي پشتيباني كند. بطور كلي تعداد قابليت هاي جديد اضافه شده به UML 2.0 كم است تا از بزرگ شدن زياد زبان جلوگيري شود و بخش عمده‌اي از اين قابليت‌هاي جديد مدل‌سازي، گسترشي بر استفاده از قابليت‌هاي جديد است كه به شما امكان مدل‌سازي سيستم‌هاي نرم‌افزاري بزرگ را مي‌دهد. علاوه بر اين، اين گسترش‌ها با استفاده از يك راهكار پايه‌اي يكسان بدست آمده‌اند : استفاده بازگشتي از مجموعه يكساني از مفاهيم در سطوح مختلف تجرد. بدين معني كه شما مي‌توانيد عناصر مدل‌سازي مربوط به يك گونه بخصوص را با يكديگر تركيب كنيد تا واحدهايي ايجاد شود كه بعنوان بلوك‌هاي سازنده در سطح بعدي تجرد مورد استفاده قرار گيرند. اين وضعيت قابل مقايسه با روشي است كه در برنامه نويسي procedure ها مي‌توانند تا چندين سطح در داخل يكديگر فراخواني شوند (NestedProcedure Call) . بطور مشخص، قابليت‌هاي مدل‌سازي زير بدين روش توسعه يافته‌اند :

-ساختارهاي پيچيده (Complex Structures)

-فعاليت‌ها (Activities)

-تعامل‌ها (Interactions)

-ماشين‌هاي حالت (State machines)

سه تاي اول از ليست بالا ۹۰ %قابليتهاي جديد UML 2.0 را شكل مي‌دهند .

ساختارهاي پيچيده(Complex Structures)

به اين منظور عناصر پايه اي ساختاري كه part ناميده مي‌شوند ممكن است يك يا چند درگاه (port) داشته باشند كه با استفاده از كانال‌هاي ارتباطي كه اتصال‌دهنده (connector) ناميده مي‌شوند (همانگونه كه درشكل ۳ نمايش داده شده است) به يكديگر متصل شده‌اند. اين ساختار تجمعي مي تواند در داخل عناصر سطح بالاتر كپسوله شود كه port هاي خود را دارند و مي‌توانند با عناصر سطح‌بالاتري متصل شوند و به همين ترتيب عناصر سطح بالاتري مي‌توانند ساخته شوند .

فعاليت‌ها (Avtivities)

Activityها در UML براي مدل‌كردن انواع مختلفي از جريان مورد استفاده قرار مي‌گيرد : جريان signal يا data، و نيز جريان هاي algorithmic يا .procedural در UML 1 يك محدوديت عمده براي Activity ها وجود داشت و آن هم اين بود كه آنها بر پايه ماشين حالت قرار گرفته بودند و بنابراين در حوزه معنايي

ماشين‌هاي حالت محدود شده بودند. در UML 2.0 پاية ماشين حالت با يك چارچوب معنايي عمومي ديگر كه تمام اين محدوديت‌ها را حذف كرده است جايگزين شده است. (در واقع، پايه معنايي با استفاده از petri net coloredهاي عمومي‌شده جايگزين شده است). علاوه بر اين از برخي استانداردهاي صنعتي براي مدلسازي فرآيندهاي كسب و كار مانند BPEL4WS الهام گرفته است .

تعاملها (Interactions)

تعاملهاي ميان اشياء در UML 1 يا با استفاده از collaboration diagram و يا با استفاده از sequence diagramنمايش داده مي‌شدند. لكن متأسفانه دو قابليت اساسي جا افتاده بودند :

1-امكان استفاده مجدد از توالي‌ها كه ممكن بود در متن دنباله‌هاي ديگر تكرار شوند. به عنوان مثال، يك دنباله كه هويت‌شناسي كاربر را انجام مي‌دهد ممكن است در چندين جاي مختلف يك برنامه كاربردي رخ دهد. بدون امكان بسته‌بندي اين دنباله‌هاي تكرار شونده در عناصر جداگانه، لازم بود كه آنها را چندين مرتبه بيان كنيد كه علاوه بر افزودن سربار اضافي، نگهداري مدل را نيز مشكل‌تر مي‌كرد .

2-امكانات كافي براي مدل‌كردن جريان‌هاي پيچيده مختلف كه در بازنمايي تعاملات سيستم‌هاي پيچيده رايج هستند. اين قابليت‌ها شامل تكراركردن يك زيردنباله، مسيرهاي اجرايي آلترناتيو، اجراهاي همروند و مستقل از ترتيب، حلقه، شرط و موارد مشابه مي‌شود .

مهمترين نوآوري در اين زمينه معرفي تعامل‌ها (Interaction) بصورت يك واحد مدلسازي جداگانه نام گذاري شده است كه امكان پارامتري كردن آنها نيز وجود دارد و بنابراين مي‌توان هر سطح پيچيدگي دلخواهي از تعامل‌هاي ميان اشياء را در يك نمودار تعامل مدل كرد .

همانگونه كه در شكل ٤ نشان داده شده است شما مي‌توانيد اين تعاملات بسته‌بندي شده را در تعاملات سطح بالاتر بصورت بازگشتي فراخواني كنيد .

ماشين هاي حالت (State Machine)

مهمترين قابليتي كه ماشين هاي حالت در UML 2.0 اضافه شده است كاملا مشابه موارد قبلي است. ايده اصلي اين است كه شما مي‌توانيد يك ماشين حالت پيچيده را كاملا بصورت پيمانه‌اي مدل كنيد كه داراي نقاط مشخصي براي ورود و خروج است. به اين ترتيب شما مي‌توانيد تجزيه داخلي يك ماشين

حالت را بوسيله يك مجموعه از ماشين‌هاي حالت جداگانه و قابل استفاده انجام دهيد. به اين ترتيب توصيف يك الگوي رفتاري مشترك در چند حوزه مختلف به سادگي انجام مي‌پذيرد .

بهبود پشتيباني براي سفارشي سازي براي يك حوزه بخصوص

تجارب عملي استفاده از UML ارزش مكانيزم‌هاي توسعه (Extension) آن را نمايان ساخته است. در UML 1.xفقط از مكانيزم هاي توسعه stereotype و profile استفاده مي شد، لكن در UML 2.0 مكانيزم توسعه جديد metamodel اضافه شده است كه امكانات توسعه سطح بالاتري را فراهم مي‌كند .

تقويت، تطابق با اصول، روشني و وضوح بيشتر براي مفاهيم مختلف مدل‌سازي

زبان ساده‌تر و سازگارتر شده است. اقدامات جديد شامل تقويت و تثبيت مفاهيم و – در بعضي موارد – حذف مفاهيم تكراري، پالايش چندين تعريف و افزودن توضيحات متني و مثال بوده است