اولین کنفرانس بین المللی و چهارمین کنفرانس ملی آموزش مهندسی
دانشگاه شیراز، ۱۹ تا ۲۱ آبان ۱۳۹۴

ارائه رویکرد نوین جهت مستندسازی خودکار تجربیات تولید نرم افزارهای مهندسی

بی نظیر گنجی[۱]
[۱]عضو هیئت علمی دانشگاه پیام نور، گروه مهندسی کامپیوتر و فناوری اطلاعات، تهران،B_ganji_a@yahoo.com

چکیده - مستندسازی یک از بخش‌های مهم هر پروژه مهندسی، به حساب می‌آید. مستندسازی می‌تواند در مواردی همچون دلایل ایجاد پروژه و شناخت صحیح نیازمندی‌ها، عملیات مربوطه وپیش‌بینی ریسک‌ها، چگونگی کیفیت مورد نظر، نحوه استفاده مجدد از تجربیات یک پروژه و... مفید واقع گردد. بازگوئی ومستندسازی تجربیات، تنها بیان تصمیمات واقدامات انجام شده نیست، بلکه ذکردقیق همه فرآیندهای ذهنی ورفتاری پیش ازتصمیم گیری نیزهست. تنهادراین صورت است که سلسله روابط علت ومعلولی نهفته درفرآیند شکل گیری تجربه، دربرابردیگران قرار می گیرد وارزیابی و بهره برداری ازآن میسر می گردد. درمهندسی نرم افزار نیز همچون سایر علوم مهندسی، باید از مستندسازی استفاده کرد، اما بسیاری از افراد از انجام این مهم، با ذکر بهانه هایی چون افزایش هزینه، کمبود وقت و..سرباز می زنند. در این مقاله رویکرد نوینی جهت مستندسازی خودکارتجربیات مهندسی در تولید نرم افزارها، ارائه شده است. رویکرد مذکور براساس کاربرد روش های مدل رانده می باشد. قابلیت ها و ویژگی های این روش، نشان می دهد که معماری مدل رانده، بلوغ لازم جهت مستند سازی خودکار را دارا است و می تواند جایگاه مناسبی در این زمینه داشته باشد.

کلید واژه- تولید نرم افزار، تولید مدل گرا (MDD)، مستند سازی خودکار، معماری مدل رانده، UML2.0

۱- مقدمه

امروزه اهمیت تولید و بروز رسانی مستندات در بهبود نگهداری و یکپارچه سازی نرم افزارها برای همگان روشن است. نبودن مستندات دقیق و سازمان‌یافته، ممکن است موجب افزایش نرخ خرابی نرم‌افزارها و حتی شکست آنها شود.

مستندسازی[۱] کار ثبت، تولید سند و در واقع ماندگار کردن اطلاعات، قرارها، خواسته ها، طرح ها، روش ها، فرایندها، وقایع و اصولاً ثبت موضوعاتی است که به نحوی مورد توجه است و در ادامه فعالیت ها مورد نیاز خواهد بود[1]. در دایره المعارف ویکی پدیا، مستند سازی به صورت زیر تعریف شده است : هر شیء فراهم کننده ارتباط همچون متن، تصویر، صدا یا ترکیبی از آنها که برای تشریح ویژگی های یک شی، سیستم یا رویه استفاده می شود [2].

در مستندسازی باید:

- به منظور بازیابی سریع و آسان یک موضوع یا نام، همواره یک اصطلاح ثابت و یکدست به کار برده شود.
- مفاهیم مرتبط و نوع ارتباط آنها با هم مشخص شود.
- واژه ها و اصطلاحات دارای دو یا چند معنی، کنترل شوند تا از بروز اختلال معنایی در هنگام بازیابی پیشگیری شود.

در مهندسی نرم افزار نیز پروژه ها، بی نیاز از این مهم نیستند. مستندسازی کامل و صحیح پروژه‌های نرم‌افزاری باعث می‌گردد که کلیه عوامل داخلی و خارجی مؤثر بر نرم‌افزار، مورد تحلیل و بررسی قرار گیرد و مدیر پروژه در حین مراحل تولید نرم‌افزار، از سلامت آن اطمینان حاصل کند. همچنین تهیه مستندات جهت ارتباط بین افراد درگیر در تولید نرم‌افزار از یک طرف، ارتباط بین کاربران از طرف دیگر و نهایتا ارتباط بین دو گروه مزبور، بسیار

مفید است و نگهداری سیستم را تا زمانی طولانی، امکان پذیر می‌سازد[3].

مستند سازی پروژه های نرم افزاری، بایستی حداقل شامل موارد زیر باشد[6-4]:

- توصیفی از نیازمندی ها و اهداف درخواست کننده نرم افزار
- امکان سنجی ها و بررسی نیازمندی های نرم افزار
- توصیفی از معماری سیستم که براساس آن تصمیم گیری می شود آیا سیستم پیاده سازی گردد یا خیر؟
- تعیین معماری و چارچوب سیستم، شامل : توصیفی از روش طراحی، مراحل طراحی، عملکرد برنامه های سیستم و داده هایی که بین قسمت های مختلف برنامه ها، رد و بدل می شوند.
-لیست برنامه ها به همراه توضیحات لازم در قسمت هایی که کد برنامه پیچیده است.
-مستندات ارزیابی سیستم که شامل نتایج آزمون های انجام شده برروی سیستم می باشد.
-تعیین بودجه، زمان و نیروی انسانی مورد نیاز برای تولید نرم افزار.
-راهنمای استفاده از سیستم، که به کاربر می گوید چگونه سیستم را نصب نماید و آن را به کار گیرد.
-مستندات نگهداری سیستم که مشکلات شناخته شده نرم افزاری و سخت افزاری را توضیح می دهد و نکاتی را که در مراحل طراحی سیستم جهت توسعه سیستم در نظر گرفته شده است، شرح می دهد.

نکته مهمی که باید در طول حیات نرم افزار در نظر داشت، این است که هنگامی که تغییراتی در سیستم ایجاد می شود تمام مستندات مربوط به آن بایستی بهنگام گردد تا باز هم از طریق سازگاری مستندات با نرم افزار، کارایی لازم دراستفاده ازنرم افزار وجود داشته باشد .گارگ و اسکاچی [7] پیشنهاد کردند به همراه مستندات، از نرم افزاری استفاده شود که روابط بین مستندات در آن ثبت شده باشد و زمانی که تغییراتی در هر یک از مستندات داده می شود، مستندات وابسته دیگر که بایستی بهنگام شوند در آن مشخص گردد.

۱-۲- ابزارهای مستندسازی

برای مستندات یک سیستم نرم افزاری، موارد زیر به کار می رود[8].

مدل ها :

مدل ها، بخش اصلی مستندات توسط طراحان و معماران را تشکیل می دهد. مدل ها خصوصا مدل های تصویری، نمای ساده شده ای از یک واقعیت بوده و یکی از بهترین روش های مستندسازی محسوب می شوند Scott ambler یک مستند نرم افزاری را به صورت زیر تعریف می کند "هر فرآورده غیر از کد منبع که هدفش انتقال اطلاعات به صورت ماندگار باشد " ambler در ادامه یک مدل نرم افزاری، را به صورت زیر تعریف می کند "تجریدی که یک یا چند جنبه از یک مسئله یا راه حلی بالقوه که یک مسئله را توصیف می کند."[9]

این مدل ها در انواع مختلف شامل مدل تحلیل، طراحی، پیاده سازی و استقرار وجود دارد.

متن ها :

شیوه مستند سازی متنی، اگر به تنهایی مورد استفاده قرار گیرد ،ساده ترین روش توصیف جنبه های از سیستم محسوب می شود. در این روش معمولا از قالب های متنی مشخص در توصیف سیستم استفاده می شود. در پروژه های نرم افزاری به دلیل ساده بودن و بالابودن سطح انتزاعی، این روش به کرات استفاده می شود. در عمل این روش، در ترکیب با روش های دیگر مورد استفاده قرار می گیرد. تجربه نشان داده است که استفاده از متن ها در کنار مدل ها لازم بوده و به هنگام انتقال طراحی به طراحان و پیاده سازان، همچنین به هنگام نگهداشت و توسعه سیستم، نقش کلیدی بازی می کنند. **مزایای** این روش :

-نزدیک بودن به زبان طبیعی و بالابودن سطح انتزاعی
- ساده بودن

معایب

- امکان ایجاد ابهام در توصیف جنبه های مختلف سیستم
- عدم امکان خودکار سازی توصیف مستندات

- عدم انتقال بهتر جنبه های مختلف سیستم همچون طراحی آن در صورت پیچیدگی سیستم[9].

نمونه های اولیه [۲]: در کنار مدل ها و متن ها، از عناصری به نام نمونه های اولیه نیز در مستندسازی بخش هایی از طراحی، استفاده می شود. طراحان واسط کاربری از این نمونه ها عمدتا برای مستند سازی واسط کاربری سیستم استفاده می کنند.در عین حال طراحان دیگر سیستم در صورت نیاز ممکن است از آنها جهت به تصویر کشیدن یک سناریو یا موارد دیگری که از طریق مدل سازی قابل ارائه نیستند، استفاده نمایند[9].

۲- طرح مسئله:

با وجود تمام مزایایی که برای مستند سازی بیان شد، عملیاتی شدن آن، یکی از نقاط ضعف در فرآیند توسعه نرم افزار است و همواره به آن به عنوان یک کار جنبی نگاه می شود. بسیاری از توسعه دهندگان احساس می کنند که کار اصلی آنها تولید کد است. نوشتن مستندات در طی فرآیند توسعه زمان می برد و منجر به کند شدن فرآیند می گردد. مستندات، وظیفه اصلی توسعه دهنده راپشتیبانی نمی کند و حضور آن، تنها وظیفه آنهایی را که بعدها می آیند، پشتیبانی می کند. بنابراین نوشتن مستندات طوری احساس می شود که گویی کاری است که تنها برای نمایش، ایجاد می شود نه برای کار اصلی. غیر از مدیر پروژه که نوشتن مستندات را اجباری می کند، هیچ انگیزه ای برای نوشتن مستندات وجود ندارد. به همین دلیل است که مستندات با کیفیت کافی ایجاد نمی شود و همین نکته باعث مشکلاتی در هنگام نگهداری و یکپارچه سازی نرم افزار می شود. از طرفی بروز رسانی مستندات نیز کار دشواری است و اگر جدی گرفته نشود، می تواند باعث ایجاد تناقض و سردرگمی توسعه دهنده شود . فشار کارفرمایان به پیمانکاران برای کاهش هزینه تولید نرم افزار، عموماً منجر به حذف و یا کم رنگ شدن، بخش مستندات سیستم های نرم افزاری شده است[10].

بنابراین در این مقاله برآنیم که یک روش کم هزینه، برای ایجاد و بروز رسانی مدل ها که نقش اساسی در تهیه مستندات سیستم خصوصا در مستندات مرحله طراحی نرم افزار، به عهده دارند، پیشنهاد دهیم.

۳- معماری مدل رانده[۳]

گروه مدیریت شیء (OMG[۴]) [11] در سال ۱۹۸۹ شکل گرفت . این گروه از ائتلاف چند سازمان ایجاد شده است و هدف آن ایجاد استانداردها و تشویق بکار گیری فن آوری شیءگرایی است. طی دهه ۹۰, OMG بر اساس دستورالعمل های مهندسی نرم افزار، مجموعه استانداردهایی را ایجاد کرد که درمجموع به آنها معماری مدیریت شیء (OMA[۵]) گفته می شود . معماری مدل رانده، نتیجه تلاشی است که این گروه برای ایجاد یک فناوری جدید انجام داده است. این معماری یک روش ساده و فراگیر است که استاندارد های مورد نیاز جهت ساخت، یکپارچه سازی ونگهداری دارایی های نرم افزاری را تعیین می کند. هدف MDAفراهم کردن یک روش مدل سازی معماری سازمان است که تحلیل گران و توسعه دهندگان بتوانند از آن برای توصیف یک کار یا یک دارایی نرم افزاری استفاده کنند[۱۲] .

می توان گفت MDA یادآور زمانی است که UML, استاندارد طراحی و ساخت نرم افزار بود. معماری مدل رانده از مدل ها به عنوان اهرمی قدرتمند برای توسعهٔ نرم افزار استفاده می کند. معماری مدل رانده چارچوبی را برای پردازش و شرح مدل ها تعریف می کند. ابزارهای MDA مدل های کسب وکار را به برنامه های کاربردی کامل، استقرارپذیر و قابل اجرا با کمترین تصمیمات فنی تبدیل می کنند. از جمله مزایای معماری مدل رانده، می توان به موارد زیر اشاره کرد:

ساده کردن تغییرات نیازها، قابلیت حمل، به هم پیوستن پلت فرم ها، قابل تبدیل بودن درخواست ها، مستقل بودن پلت فرم ها، حذف کد نویسی دستی رفتار یک مدل، طراحی بهتر و دقیق تر با تمرکز بر روی مدل ها، انقلابی در افزایش سطح تجرید، جدا سازی قواعد کاری از محیط پیاده سازی [15-13].

۳-۱- مفاهیم پایه ای

در این بخش مفاهیم پایه ای و اولیه معماری مدل رانده، توضیح داده شده است.

۱-۱-۱- مدل ران

در روش مدل ران، مدل ها وظیفه هدایت و راهبری جریان فهم، طراحی، ساخت، استقرار، بهره برداری، نگهداشت و اصلاح سیستم را بر عهده دارند.

۱-۱-۲- مدل مستقل از محاسبه (CIM) [۶]

یک مدل مستقل از محاسبه، دیدی از سیستم بر پایه دیدگاه مستقل از محاسبه است. یک CIM مدلی از سیستم است که نشان می دهد سیستم در محیط چگونه کار می کند.

۱-۱-۳- مدل مستقل از سکو (PIM) [۷]

یک مدل مستقل از سکو دیدی از سیستم بر پایه دیدگاه مستقل از سکو است. این مدل به توصیف یک سیستم می پردازد، اما جزئیات مربوط به استفاده از سکو را نشان نمی دهد. یک PIM می تواند شامل مشخصهٔ دیدگاه ها، اطلاعات و محاسبات سازمان باشد.

مدل خاص سکو(PSM) [۸]

یک مدل خاص سکو دیدی از سیستم بر پایه دیدگاه خاص سکو است. یک PSM مشخصه های داخل PIM را با جزئیات مربوط به چگونگی استفاده سیستم بر روی یک سکو خاص ترکیب می کند. امروزه این مدل معمولاًبه شکل کتابچه راهنمای نرم افزار و سخت افزار و یا گاهی حتی تنها در فکر معمار وجود دارد.

۱-۱-۴- نگاشت [۹]

نگاشت برای تبدیل یک PIM به PSM برای یک سکو خاص استفاده می شود. در این فرآیند، مدل سکو ماهیت نگاشت را تعیین می کند.

۴- چرخه حیات معماری مدل رانه

در گام اول عناصری ازPIM برای مشخص کردن چگونگی نگاشت مورد استفاده در تبدیل PIM به PSM نشانه گذاری می شود. گام بعدی این است که PIM نشانه گذاری شده را گرفته و آن را به یک PSM تبدیل کنیم . این کار می تواند هم به صورت دستی و هم به شکل خودکار انجام شود. ورودی به این فرآیند، PIM نشانه گذاری شده و یک نگاشت است. برخی از ابزارها ممکن است یک PIM را مستقیماً به یک کد قابل استقرار تبدیل کنند (بدون آنکه ابتدا یک PSM تولید کنند). با این حال، بهتر است چنین ابزارهایی PSM را نیز تولید کنند تا فهم و اشکال زدایی کد آسان تر انجام شود. نتایج فرآیند تغییر شکلِ یک PIM عبارتند از :"یک PSM " و"یک سابقه تغییر شکل". سابقهٔ تغییر شکل شامل یک نگاشت از عناصر PIM به عناصر متناظر در PSMاست که نشان می دهد چه بخش هایی از نگاشت برای هر قسمت از فرآیند تغییر شکل به کار رفته اند. این سابقه می تواند در اختیار شخصی که بر روی PIM یا PSM کار می کند، قرار گیرد. ابزارهای مدل سازی MDA که سابقه تغییر شکل را نگهداری می کنند، می توانندر هنگام تغییرات، مدل هایPIM و PSM را با یکدیگر هماهنگ سازند[12].

۴-۱- استانداردها و فرامدل های معماری مدل رانه

گروه مدیریت شیء تعدادی از استانداردها و فرامدل های مختلف را به خدمت گرفته است تا با همکاری آنها معماری مدل رانده شکل گیرد. در این قسمت برخی از مهم ترین فرامدل ها که در راهکار پیشنهادی استفاده می شوند، معرفی شده است .

ابزار فراشیء (MOF[۱۰])

ابزار فراشیء یک چارچوب برای تعریف، دستکاری و یکپارچگی فراداده ها و داده ها است. این کار با یک روش مستقل از سکو انجام می شود. مشخصهٔ MOF اجازه می دهد مدل ها از یک برنامه کاربردی خارج و به یک برنامه دیگر وارد شوند، از طریق شبکه های مختلف منتقل شوند، در یک انباره ذخیره و بازیابی شوند و به قالب های مختلف تبدیل شوند. تمام مدل ها و فرامدل

های مورد استفاده در MDA بر اساس ابزار فراشیء تعریف می شوند و اصطلاحا پیرو MOF هستند[13].

زبان مدل سازی یکپارچه (UML[11])

UMLیک زبان مدل سازی گرافیکی است که برای تصویرسازی، مستندسازی و محدودسازی فرآورده های سیستم های شیءگرا مورد استفاده قرار می گیرد که در ادامه توضیح بیشتر داده شده است.

زبان محدودیت شیء(OCL[12])

ابزارکمکی برایUML است. این استاندارد اجازه می دهد محدودیت ها ومنطق فراروی استفاده از مدل ها مشخص شود OCLیک زبان رسمی است که عباراتی را در مدل های UML توصیف می کند.

نمایه های UML

به کمک نمایه ها می توان زبان UML را برای یک ناحیه خاص از محاسبات (مثلاً محاسبات توزیع شده) و یا یک سکو خاص (مثل EJB یا CORBA) سفارشی کرد . در معماری مدل رانده ، مدل های PIM و PSM به کمک نمایه ها تعریف می شوند. [16]

۱-۱-۵- زبان مدل سازی یکنواخت(UML)

UML اولین بار توسط شرکت Rational ارائه شد و پس از آن از طرف بسیاری از شرکت های کامپیوتری و مجامع صنعتی و نرم افزاری دنیا مورد حمایت قرار گرفت؛ به طوریکه تنها پس از یک سال، توسط گروه Object Management Group، به عنوان زبان مدل سازی استاندارد پذیرفته شد. UML توانایی ها و خصوصیات بارز فراوانی دارد که می‌تواند به طور گسترده‌ای در تولید نرم‌افزار استفاده گردد.[19]

ویژگی های UML

UML یک زبان مدل سازی است، اما چیزی فراتر از چند نماد گرافیکی است. به طوریکه در ورای این نمادها، یک سمنتیک (معناشناسی) قوی وجود دارد، به طوریکه یک تولیدکننده می‌تواند مدل هایی تولید کند که تولیدکننده های دیگر و یا حتی یک ماشین آن را بخواند و بفهمد. بنابراین یکی دیگر از نقش های مهم UML تسهیل ارتباط بین اعضای پروژه و یا بین تولیدکنندگان مختلف می باشد. یکی دیگر از ویژگی های مهم UML این است که مستقل از متدولوژی یا فرایند تولید نرم افزار می باشد. از دیگر ویژگی های آن می توان به پشتیبانی از مفاهیم سطح بالای شی گرایی مثل Collaboration، Framework، Pattern و Component اشاره کرد. همچنین UML با استفاده از یک سری مکانیزم های گسترش پذیر امکان می دهد که بتوان زبان های مدل سازی جدیدتری (با گسترش مفاهیم پایه ای موجود) ایجاد کرد.

ابزارهای مدل سازی با سفارشی سازی UML از طریق نقاط گسترش آن، فاصله بین مستندات طراحی و کد را به نحو خوبی کاهش داده، ریسک، هزینه و زمان تولید نرم افزار را کاهش می دهند.[20]

۵- راهکار:

می دانیم که در سیستم های نرم افزاری، مدل ها اصلی ترین نقش را برای مستندسازی، ایفا می کنند. در عمل به هنگام توسعه سیستم های نرم افزاری، نیاز به ابزارهای مدل سازی است که مارا در توسعه ونگهداشت این مدل ها یاری نموده وتا حد امکان قابلیت های خودکار سازی را در ایجاد آنها، ارائه نمایند. به کارگیری مدل ها در توسعه سیستم ومستند سازی آن مزایای زیادی به همراه دارد که مهم ترین آن ها عبارتند از [5] :

ارتباط بهتر بین توسعه دهندگان سیستم

ابهام کمتر و دقت بیشتر در مستند سازی مطالب

ایجاد خطاهای کمتر در مستند سازی

نمایش کامل تر ، جزیی تر و انتزاعی تر سیستم

کسب بهتر دانش از دیگران

کنترل و کاهش پیچیدگی سیستم

در یک مدل از هر نوع از فرآورده‌های فیزیکی (به عنوان مثال، یک اتومبیل، ساختمان، پل و موارد دیگر)، نیاز به حذف برخی جزئیات و مجردسازی است که به صورت 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 کاملا دقیق و قاعده‌مند انجام شوند. پتانسیلی که در ورای این ترکیب قدرتمند تجرد و خودکارسازی وجود دارد، منجر به ارائه راهکار نوینی که در این مقاله با عنوان تولید مدل‌گرا (model-driven development) پیشنهاد شده است. ویژگی اصلی MDD این است که مدل‌ها به فرآورده اصلی طراحی نرم‌افزار بدل گشته‌اند، که این امر منجر به انتقال تمرکز بیشتر، از کد برنامه به مدل می‌شود. با دقیق‌تر شدن مدل‌ها (که UML 2.0 این قابلیت را فراهم می‌آورد) سطح خودکارسازی تولید کد از روی مدل، افزایش پیدا می کند و مزایای MDD بیشتر نمایان می‌شوند. هر چند، مستندات همه مراحل تولید نرم افزار را نمی توان با این روش تهیه کرد، ولی در تهیه مستندات مربوط به طراحی نرم افزار بسیار کارآمد خواهد بود. در این روش، از UML2. x که نگارش جدیدی از استاندارد UML است، استفاده می شود. نمایه های UML به همراه زبان محدودیت شیء (OCL) که وظیفه توصیف عبارات درمدل های UML را برعهده دارد، مستند سازی دراین روش را کامل می کنند. لازمه خودکارسازی، رفع ابهام و بی دقتی مدل‌ها (و در نتیجه از زبان مدل‌سازی) است تا برنامه‌های کامپیوتری بتوانند مدل‌ها را تبدیل به کد کنند. یکی از اقداماتی که به منظور کمینه کردن ابهامات و افزایش دقت مدل انجام شده است، استفاده از metamodel ها است. این مدل خصوصیات هر عنصر مدل سازی UML و ارتباط آن با سایر مفاهیم مدل‌سازی را تعریف می‌کند. این metamodel ها با استفاده از یک زیرمجموعه از عناصر UML - که بیشتر مفاهیم Class Diagram هستند و اصطلاحا MOF نامیده می‌شوند و در قسمت های قبل، به عنوان عناصر معماری مدل رانده معرفی شده اند ،تعریف می شوند و بوسیله یک مجموعه از محدودیت های رسمی که به زبان OCL نوشته شده است، پشتیبانی می‌شود. در این روش، بهبود قابل توجه در توانایی مدل کردن سیستم‌های نرم‌افزاری بزرگ با افزایش قابلیت‌های جدید انعطاف‌پذیرِ سلسله‌مراتبی، ایجاد شده است تا از مدل‌سازی نرم‌افزار در سطوح دلخواه پیچیدگی، پشتیبانی کند. در این روش، می توان عناصر مدل‌سازی مربوط به یک گونه بخصوص را با یکدیگر ترکیب کرد 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 برنامه نویسی procedure ها استفاده می شود و آنها می‌توانند تا چندین سطح در داخل یکدیگر فراخوانی شوند (Nested Procedure Call).

۶- بحث و ارزیابی

قابلیت های روش MDD و کاربرد آن، در مستندسازی طراحی سیستم های نرم افزاری از چند جنبه مختلف قابل بررسی است از جمله :

حمایت از کدنویسی خودکار

در روش MDD،کدی توسط برنامه نویس ایجاد نمی شود و همه کدها، به صورت خودکار از روی مدل ها و فشردن یک دکمه از نرم افزار مربوطه تولید می شوند. بنابراین روش ارائه شده، تا حد زیادی در گیری افراد جهت برنامه نویسی را کم می کند. در این روش تولید مدل ها و مستندات جزء وظیفه اصلی محسوب می شود و تولید کد تا حد امکان به طور خودکار انجام می شود. ابزار و نرم افزارهای مورد استفاده این توانایی را دارند که مدل های مختلف را به همراه مستندات آنها، به خوبی نگهداری و یکپارچه سازند که نتیجه آن وجود اطلاعات کافی در زمان نگهداری، توسعه و یکپارچه سازی نرم افزارها است که دلیل اصلی ایجاد این قابلیت استفاده از زبان UML است.

تسهیل ساختارهای پیچیده[۱۳]

به این منظور، عناصر پایه ای ساختاری که part نامیده می‌شوند، ممکن است یک یا چند درگاه داشته باشند که با استفاده از کانال‌های ارتباطی که اتصال‌دهنده نامیده می‌شوند، به یکدیگر متصل شده‌اند. در روش MDD، این ساختار تجمعی می تواند در داخل عناصر سطح بالاتر کپسوله شود که درگاه های خود را دارند و می‌توانند با عناصر سطح بالاتری متصل شوند و به همین ترتیب، عناصر سطح بالاتری می‌توانند ساخته شوند. در

مستند سازی سیستم های بزرگ و پیچیده، ابتدا مستند سازی را از عناصر سطح پایین شروع کرده و سپس با استفاده از این ویژگی، به عناصر سطح بالاتر مرتبط می شوند. این امر مستندسازی را ساده تر و سرعت انجام کار را بالا می برد.

فعالیت‌ها [۱۴]

فعالیت ها در UML برای مدل‌کردن انواع مختلفی از جریان ها مورد استفاده قرار می‌گیرد : جریان signal یا data، و نیز جریان های algorithmic یا procedural.

در روش هایی که از زبان UML1 استفاده می کنند، یک محدودیت عمده برای فعالیت 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 روش پیشنهادی که از UML 2.0 استفاده می کند، پایهٔ ماشین حالت با یک چارچوب معنایی عمومی دیگر که تمام این محدودیت‌ها را حذف کرده است، جایگزین شده است.

تعامل ها [۱۵]

تعامل های میان اشیاء در روش هایی که از زبان UML1 استفاده می کنند، با استفاده از collaboration diagram و یا با استفاده از sequence diagram نمایش داده می‌شوند. اما متأسفانه دو قابلیت اساسی جا افتاده بودند :

۱- امکان استفاده مجدد از توالی‌ها که ممکن بود در متن دنباله‌های دیگر تکرار شوند. به عنوان 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 کرد که علاوه بر افزودن سربار اضافی، نگهداری مدل را نیز مشکل‌تر می‌کند.

۲- امکانات کافی برای مدل‌کردن جریان‌های پیچیده مختلف که در بازنمایی تعاملات سیستم‌های پیچیده رایج هستند. این قابلیت‌ها شامل تکرارکردن یک زیردنباله، مسیرهای اجرایی آلترناتیو، اجراهای همروند و مستقل از ترتیب، حلقه، شرط و موارد مشابه می‌شود. مهمترین نوآوری در این زمینه معرفی تعامل‌ها بصورت یک واحد مدل‌سازی جداگانه 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. در مستند سازی سیستم های نرم افزاری پیچیده، وقتی مستندات قسمتی از برنامه آماده شد و این تکه برنامه در جای دیگر(حتی با داده های متفاوت) تکرار شود، با توجه به پارامتری بودن مدل ها، نیازی به مستند سازی مجدد نیست و فقط داده های جدید در مدل جایگذاری می شوند و با توجه به اینکه مدل ها به طور خودکار و بدون نیاز به برنامه نویسی کاربر و توسط سیستم مدل رانده ،امکان تبدیل به کد را دارند و به طور ضمنی کل مستندات را به همراه دارند. این امر موجب افزایش سرعت و کاهش هزینه خواهد شد.

ماشین های حالت [۱۶]

ایده اصلی در روش پیشنهادی این است که می توان یک ماشین حالت پیچیده را کاملا بصورت پیمانه‌ای مدل کرد که دارای نقاط مشخصی برای ورود و خروج است. به این ترتیب می توان تجزیه داخلی یک ماشین حالت را بوسیله یک مجموعه از ماشین‌های حالت جداگانه و قابل استفاده انجام داد. به این ترتیب توصیف یک الگوی رفتاری مشترک در چند حوزه مختلف، به سادگی انجام می‌پذیرد. در بحث مستندسازی این قابلیت باعث سادگی و کاهش هزینه می شود. خصوصا در بروزرسانی مستندات چرا که هر گونه تغییر در سیستم، منجر به تغییر در پیمانه ها و مدل ها بدون درگیر شدن در جزئیات می شود و این امر به سهولت بروزشدن مستندات کمک می کند.

۷- نتیجه گیری

کار مستندسازی به دلیل نیاز به بررسی دقیق منابع و ثبت تصمیم های اتخاذ شده، فعالیت وقت گیر و پرهزینه ای است .اما باید توجه داشت که هزینه و وقت صرف شده جهت مستندسازی در زمان بازیابی اطلاعات جبران می شود، چون بازیابی در نظام هایی

که از شیوه های مستندسازی استفاده می کنند، دقیق تر و سریع تر است. در روشMDD با استفاده از دو دسته مدل، مدل های مستقل از سکو و مدل های وابسته به سکو و تبدل کننده های مناسب بین این مدل ها و زبان UML2، می توان از دامنه مسئله به کد مسئله دست یابد. بدین ترتیب مستندات طراحی فقط شامل مدل ها بوده و تمامی مراحل کار به صورت خودکار صورت گرفته ،مشکل مستندسازی دستی و مسائل مشابه وجود ندارد و مستندات سیستم های وسیع و پیچیده در کمترین زمان و هزینه، آماده خواهد شد.

۸- مراجع

[3]بهشتی زهرا، بررسی روش های مستندسازی و ارائه روش مناسب مستندسازی برای سیستم های اطلاعاتی در ایران، پایا ن نامه کارشناسی ارشد، دانشکده فنی و مهندسی، دانشگاه آزاد اسلامی نجف آباد، ۱۳۷۸

[8]مرآت‌نیا احمد (ترجمه)، استانداردهای مهندسی نرم‌افزار، تألیف مازاک.، چاپ اول ۱۳۸۹.

[10] مقایسه متدولوژیهای ایجاد و توسعه سیستم های اطلاعاتی، انتشارات انیستیتو ایزایران ،۱۳۹۰.

[۱۶]استاد زاده، س ش،یک روش مبتنی بر معماری مدل را نه برای مدل سازی یکپارچه سلول‌های چارچوب زکمن، دانشکده فنی و مهندسی، گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد علوم و تحقیقات، پایان نامه کارشناسی ارشد،1385.

[17]استاد زاده، س ش، شمس، ف، نقش معماری مدل رانه در یکپارچه سازی سیستم‌های فوق وسیع، پنجمین کنفرانس بین‌المللی مدیریت فناوری اطلاعات و ارتباطات، ص ص ۱-1387،5.

[19]فتح اللهی، ع، بررسیUML از نظر قابلیت پوشش به چارچوب زکمن،دانشکده مهندسی برق و کامپیوتر، دانشگاه شهیدبهشتی، پایان‌نامه کارشناسی ارشد،1383.

[1]Sommervil, Ian, Software Engineering, 4th ed.,2012 .

[2]https://fa.wikipedia.org.last update 2015.

[4]Green, Steve, Information system design, first edition, Chapman and Hall,2005.

[5]Phona, Vir, A Standard for Software Documentation, ANSI/ANS, 10-13, 2011.

[6]Pressman Rger S., Software Engineering A Practitioner's Approach, seven Edition, 2010.

[7] Garg P.K. and Scacchi W., A hypertext system to maintain software life-cycle documents, IEEE Software, 2001, 7(3), 90-98.

[9] ambler ,Scott, Agile/Lean Documentation: Strategies for Agile Software Development access in: http://www.agilemodeling.com/essays/agileDocumentation.htm#sthash.aioG6itO.dpuf,last update 2015

[15] Frankel, David S. , Model Driven Architecture Applying MDA to EnterpriseComputing, OMG Press, John Wiley & Sons, 2003.

[18] Zachman, John A.,The Zachman Framework:A Primer for Enterprise Engineeringand Manufacturing, Zachman International,2009.

[20] http://www.uml.org.last update 2015

[1] Documentation

[22] prototypes
[3] Model Driven Architecture
[4] Object managment group
[5] Object management architecture
[6] Computation Independent Model
[7] Platform Independent Model
[8] Platform Specific Model
[9] Mapping
[10] Meta object facility
[11] Unified modeling language
[12] Object constraint language
[13] Structures Complex
[14] Avtivities
[15] Interactions
[16]Machine State