استاندارد ملی ایران

۲۱۸۳۰

چاپ اول

۱۳۹۶

جمهوری اسلامی ایران

Islamic Republic of Iran

سازمان ملی استاندارد ایران

Iranian National Standards Organization

INSO

21830

1st.Edition

2017

Identical with ISO/IEC 19508: 2014

فناوری اطلاعات –

هسته‌ی ابزار فراشیء (MOF) گروه مدیریت شیء

Information technology — Object Management Group Meta Object Facility (MOF) Core

ICS: 35.040.50

سازمان ملی استاندارد ایران

تهران، ضلع جنوب غربی میدان ونک، خیابان ولیعصر، پلاک ۲۵۹۲

صندوق‌پستی: ۶۱۳۹-۱۴۱۵۵ تهران- ایران

تلفن: ۵-۸۸۸۷۹۴۶۱

دورنگار: ۸۸۸۸۷۰۸۰ و ۸۸۸۸۷۱۰۳

کرج ، شهر صنعتی، میدان استاندارد

صندوق پستی: ۱۶۳-۳۱۵۸۵ کرج - ایران

تلفن: ۸- ۳۲۸۰۶۰۳۱ (۰۲۶)

دورنگار: ۳۲۸۰۸۱۱۴ (۰۲۶)

رایانامه: standard@isiri.org.ir

وبگاه: http://www.isiri.gov.ir

Iranian National Standardization Organization (INSO)

No.1294 Valiasr Ave., South western corner of Vanak Sq., Tehran, Iran

P. O. Box: 14155-6139, Tehran, Iran

Tel: + 98 (21) 88879461-5

Fax: + 98 (21) 88887080, 88887103

Standard Square, Karaj, Iran

P.O. Box: 31585-163, Karaj, Iran

Tel: + 98 (26) 32806031-8

Fax: + 98 (26) 32808114

Email: standard@isiri.org.ir

Website: http://www.isiri .gov.ir

به نام خدا

آشنایی با سازمان ملی استاندارد ایران

سازمان ملی استاندارد ایران به موجب بند یک مادهٔ ۳ قانون اصلاح قوانین و مقررات مؤسسهٔ استاندارد و تحقیقات صنعتی ایران، مصوب بهمن ماه ۱۳۷۱ تنها مرجع رسمی کشور است که وظیفهٔ تعیین، تدوین و نشر استانداردهای ملی (رسمی) ایران را به عهده دارد.

تدوین استاندارد در حوزه‌های مختلف در کمیسیون‌های فنی مرکب از کارشناسان سازمان، صاحب‌نظران مراکز و مؤسسات علمی، پژوهشی، تولیدی و اقتصادی آگاه و مرتبط انجام می‌شود و کوششیء همگام با مصالح ملی و با توجه به شرایط تولیدی، فناوری و تجاری است که از مشارکت آگاهانه و منصفانهٔ صاحبان حق و نفع، شامل تولیدکنندگان، مصرف‌کنندگان، صادرکنندگان و واردکنندگان، مراکز علمی و تخصصی، نهادها، سازمان‌های دولتی و غیردولتی حاصل می‌شود. پیش‌نویس استانداردهای ملی ایران برای نظرخواهی به مراجع ذی‌نفع و اعضای کمیسیون‌های مربوط ارسال می‌شود و پس از دریافت نظرها و پیشنهادها در کمیتهٔ ملی مرتبط با آن رشته طرح و در صورت تصویب، به عنوان استاندارد ملی (رسمی) ایران چاپ و منتشر می‌شود.

پیش‌نویس استانداردهایی که مؤسسات و سازمان‌های علاقه‌مند و ذی‌صلاح نیز با رعایت ضوابط تعیین شده تهیه می‌کنند درکمیته ملی طرح، بررسی و درصورت تصویب، به عنوان استاندارد ملی ایران چاپ و منتشر می‌شود. بدین ترتیب، استانداردهایی ملی تلقی می‌شود که بر اساس مقررات استاندارد ملی ایران شمارهٔ ۵ تدوین و در کمیتهٔ ملی استاندارد مربوط که در سازمان ملی استاندارد ایران تشکیل می‌شود به تصویب رسیده باشد.

سازمان ملی استاندارد ایران از اعضای اصلی سازمان بین‌المللی استاندارد ((ISO[1]، کمیسیون بین‌المللی الکتروتکنیک (IEC)[2] و سازمان بین‌المللی اندازه‌شناسی قانونی ((OIML[3] است و به عنوان تنها رابط[4] کمیسیون کدکس غذایی ((CAC[5] در کشور فعالیت می‌کند. در تدوین استانداردهای ملی ایران ضمن توجه به شرایط کلی و نیازمندی‌های خاص کشور، از آخرین پیشرفت‌های علمی، فنی و صنعتی جهان و استانداردهای بین‌المللی بهره‌گیری می‌شود.

سازمان ملی استاندارد ایران می‌تواند با رعایت موازین پیش‌بینی شده در قانون، برای حمایت از مصرف‌کنندگان، حفظ سلامت و ایمنی فردی و عمومی، حصول اطمینان از کیفیت محصولات و ملاحظات زیست‌محیطی و اقتصادی، اجرای بعضی از استانداردهای ملی ایران را برای محصولات تولیدی داخل کشور و/یا اقلام وارداتی، با تصویب شورای عالی استاندارد، اجباری کند. سازمان می‌تواند به منظور حفظ بازارهای بین‌المللی برای محصولات کشور، اجرای استاندارد کالاهای صادراتی و درجه‌بندی آن را اجباری کند. همچنین برای اطمینان بخشیدن به استفاده‌کنندگان از خدمات سازمان‌ها و مؤسسات فعال در زمینهٔ مشاوره، آموزش، بازرسی، ممیزی و صدورگواهی سامانه‌های مدیریت کیفیت و مدیریت زیست‌محیطی، آزمایشگاه‌ها و مراکز واسنجی (کالیبراسیون) وسایل سنجش، سازمان ملی استاندارد این‌گونه سازمان‌ها و مؤسسات را بر اساس ضوابط نظام تأیید صلاحیت ایران ارزیابی می‌کند و در صورت احراز شرایط لازم، گواهینامهٔ تأیید صلاحیت به آن‌ها اعطا و بر عملیات آن‌ها نظارت می‌کند. ترویج دستگاه بین‌المللی یکاها، واسنجی وسایل سنجش، تعیین عیار فلزات گرانبها و انجام تحقیقات کاربردی برای ارتقای سطح استانداردهای ملی ایران از دیگر وظایف این سازمان است.

1- International Organization for Standardization
2- International Electrotechnical Commission
3- International Organization for Legal Metrology (Organisation Internationale de Metrologie Legals)
4- Contact point
5- Codex Alimentarius Commission

کمیسیون فنی تدوین استاندارد

« فناوری اطلاعات – هسته‌ی ابزار فراشیء (MOF) گروه مدیریت شیء»

رئیس:	سمت و/ یا نمایندگی:
معروف، سینا (کارشناسی مهندسی کامپیوتر، سخت‌افزار)	کارشناس شبکه – کارشناس سازمان فناوری اطلاعات ایران
دبیر:	
فرهاد شیخ احمد، لیلا (کارشناسی‌ارشد مهندسی کامپیوتر - نرم‌افزار)	کارشناس استاندارد
اعضاء: (اسامی به ترتیب حروف الفبا)	
بی‌مانند، هدی (کارشناسی مهندسی کامپیوتر، نرم‌افزار)	کارشناس رایانه و فناوری اطلاعات اداره استاندارد استان ایلام
جمشید عینی، مریم (کارشناسی مهندسی کامپیوتر، نرم‌افزار)	مدیر شبکه و رئیس سیستم شبکه مجتمع قضایی خانواده شماره یک
سجادی، ندا (کارشناسی مهندسی کامپیوتر، نرم‌افزار)	کارشناس ارشد بانک پارسیان
سعیدی، عذرا (کارشناسی‌ارشد مهندسی مخابرات)	کارشناس استاندارد سازمان فناوری اطلاعات ایران
شیرازی میگون، مریم (کارشناسی مهندسی فناوری اطلاعات)	کارشناس پژوهشگاه سازمان ملی استاندارد ایران
قسمتی، سیمین (کارشناسی‌ارشد مهندسی فناوری اطلاعات)	مشاور مرکز اپای دانشگاه تربیت مدرس
ویراستار:	
قسمتی، سیمین (کارشناسی‌ارشد مهندسی فناوری اطلاعات)	مشاور مرکز اپای دانشگاه تربیت مدرس

فهرست مندرجات

عنوان **صفحه**

پیش‌گفتار

استاندارد «فناوری اطلاعات – هسته‌ی ابزار فراشیء (MOF) گروه مدیریت شیء» که پیش‌نویس آن در کمیسیون‌های مربوط بر مبنای پذیرش استانداردهای بین‌المللی به عنوان استاندارد ملی ایران به روش اشاره شده در مورد الف بند ۷ استاندارد ملی شمارهٔ ۵ تهیه و تدوین شده، در پانصد و یازدهمین اجلاسیه کمیتهٔ ملی استاندارد فناوری اطلاعات مورخ ۱۳۹۶/۰۲/۲۰ تصویب شد. اینک این استاندارد به استناد بند یک ماده ۳ قانون اصلاح قوانین و مقررات سازمانی استاندارد و تحقیقات صنعتی ایران، مصوب بهمن ماه ۱۳۷۱، به عنوان استاندارد ملی ایران منتشر می‌شود.

استانداردهای ملی ایران بر اساس استاندارد ملی ایران شمارهٔ ۵ (استانداردهای ملی ایران- ساختار و شیوهٔ نگارش) تدوین می‌شوند. برای حفظ همگامی و هماهنگی با تحولات و پیشرفت‌های ملی و جهانی در زمینهٔ صنایع، علوم و خدمات، استانداردهای ملی ایران در صورت لزوم تجدیدنظر خواهند شد و هر پیشنهادی که برای اصلاح و تکمیل این استانداردها ارائه شود، هنگام تجدیدنظر در کمیسیون‌های مربوط مورد توجه قرار خواهد گرفت. بنابراین، باید همواره از آخرین تجدیدنظر استانداردهای ملی ایران استفاده کرد.

این استاندارد ملی بر مبنای پذیرش استاندارد بین‌المللی زیر از طریق ترجمه تخصصی کامل متن آن به زبان فارسی تهیه و تدوین شده و معادل یکسان استاندارد بین‌المللی مزبور به زبان فارسی است:

ISO/IEC 19508:2014, Information technology — Object Management Group Meta Object Facility (MOF) Core

فناوری اطلاعات – هسته‌ی ابزار فراشیء (MOF) گروه مدیریت شیء

۱ هدف و دامنه کاربرد

هدف از تدوین این استاندارد، تعیین مبنایی برای تعریف فرامدل در خانواده گروه مدیریت شیء (OMG) زبان‌های معماری مدل‌محور (MDA)[1] و مبتنی بر ساده‌سازی قابلیت‌های مدل‌سازی رده[2] UML2 است. این استاندارد علاوه بر فراهم کردن ابزاری برای تعریف فرامدل، همچنین به طور کل برای مدیریت مدل که شامل شناسانه‌ها، یک قابلیت برچسب عام ساده و عملیات بازتابی است، قابلیت‌های هسته را می‌افزاید که به صورت عام تعریف شده است و صرفنظر از فرامدل می‌توان از آن‌ها استفاده کرد.

هسته ابزار فراشیء (MOF 2) توسط سایر مشخصات ابزار فرا شیء گروه مدیریت شیء ساخته می‌شود که شامل موارد زیر است (در این فهرست «مدل مبتنی بر MOF» یعنی هر مدلی که با استفاده از MOF از فرامدل تعریف شده، یک نمونه می‌سازد که شامل خود فرامدل‌ها نیز می‌شود) که شامل موارد زیر است:

- تبادل فراداده زبان نشانه گذاری بسط پذیر (XMI)[3] – برای مبادله مدل‌های مبتنی بر MOF در زبان نشانه گذاری بسط پذیر (XML)[4] [XMI24]
- ابزار MOF 2 و چرخه حیات شیء – برای اتصال و مدیریت مجموعه عناصر مدل مبتنی بر MOF [MOFFOL]
- نسخه‌بندی MOF 2 و چرخه حیات توسعه – برای مدیریت نسخه‌ها و پیکربندی‌های مدل‌های مبتنی بر MOF [MOFVD]
- منظرها[5] و تبدیل‌های مربوط به پرس‌وجوهای MOF- برای تبدیل مدل‌های مبتنی بر MOF [QVT]
- مدل‌های MOF به متن – برای تولید متن از مدل‌های مبتنی بر MOF مانند برنامه‌ها [MOFM2T]
- زبان محدودیت شیء- برای تعیین محدودیت روی مدل‌های مبتنی بر MOF [OCL]

۲ انطباق

دو نقطه پیروی وجود دارد:

- MOF ضروری (EMOF)
- MOF کامل (CMOF)

---

1 - Model-driven architecture
2 - class
3 - eXtensible Markup Language Metadata Interchange
4 - eXtensible Markup Language
5- view

پیاده‌سازی‌های انطباق ممکن است فقط ازEMOF پشتیبانی کنند، برای جزئیات بیشتر به زیربند ۴-۱۲ مراجعه شود، یا ممکن است از CMOF پشتیبانی کنند که شامل EMOF هم می‌شود. برای جزئیات بیشتر به زیربند ۴-۱۴ مراجعه شود.

همه پیاده‌سازی‌های انطباق باید با مدل مستقل از بن‌سازه MOF که در بند ۱۵ مشخص شده انطباق داشته باشند و از نگاشت فناوری مشخص شده در مشخصات مبادله فراداده XML (XMI) [XMI24] پشتیبانی کنند.

۳ مراجع الزامی

در مراجع زیر ضوابطی وجود دارد که در متن این استاندارد به صورت الزامی به آن‌ها ارجاع داده شده است. بدین‌ترتیب، آن ضوابط جزئی از این استاندارد محسوب می‌شوند.

در صورتی که به مرجعی با ذکر تاریخ انتشار ارجاع داده شده باشد، اصلاحیه‌ها و تجدیدنظرهای بعدی آن برای این استاندارد الزام‌آور نیست. در مورد مراجعی که بدون ذکر تاریخ انتشار به آن‌ها ارجاع داده شده است، همواره آخرین تجدیدنظر و اصلاحیه‌های بعدی برای این استاندارد الزام‌آور است.

استفاده از مراجع زیر برای کاربرد این استاندارد الزامی است:

2-1 [UML2Sup] ISO/IEC 19505-2:2012 “Information technology - Object Management Group Unified Modeling Language (OMG UML), Superstructure” (OMG Unified Modeling Language (OMG UML), Superstructure http:// www.omg.org/spec/UML/2.4.1/Superstructure)

نگاشت الزامی MOF به XMI در این مستند مشخص شده است:

2-2 [XMI24] ISO/IEC 19509:2014 “Information technology - Object Management Group XML Metadata Interchange (XMI)” (XML Metadata Interchange - http://www.omg.org/spec/XMI/2.4.2)

محدودیت‌های صوری به شکل OCL بیان شده‌اند که در این مستند مشخص شده است:

2-3 [OCL] ISO/IEC 19507:2012 “Information technology - Object Management Group Object Constraint Language (OCL)” (OMG Object Constraint Language (OCL) - http://www.omg.org/spec/OCL/2.3.1)

این مرجع در پیوست مهاجرت از MOF1 به MOF2 استفاده شده است:

2-4 [MOF1] ISO/IEC 19502:2005 “Meta Object Facility (MOF) Specification Version 1.4.1” (OMG Meta Object Facility (MOF) Specification (Version 1.4) - http://www.omg.org/spec/MOF/1.4)

نگاشت الزامی MOF به XMI در مرجع زیر مشخص شده است:

2-5 [XMI24] ISO/IEC 19509:2014 “Information technology - Object Management Group XML Metadata Interchange (XMI)” (XML Metadata Interchange - http://www.omg.org/spec/XMI/2.4.2)

محدودیت‌های صوری به شکل OCL بیان شده‌اند که در مرجع زیر مشخص شده است:

2-6 [OCL] ISO/IEC 19507:2012 “Information technology - Object Management Group Object Constraint Language (OCL)” (OMG Object Constraint Language (OCL) - http://www.omg.org/spec/OCL/2.3.1)

۴ اصطلاحات و تعاریف

در این استاندارد، اصطلاح و تعریف زیر به کار می‌رود:

۱-۴

تهی

NULL

در این استاندارد ملی از تهی برای نشان دادن نبود مقدار، استفاده شده است. برای مثال یک خاصیت تک‌مقداری که هیچ مقداری نداشته باشد، تهی است و زمانی که عملیاتی تهی بازگرداند، در واقع هیچ مقداری را بازنگردانده است.

۵ نمادها

MOF 2 از زیرمجموعه‌ای از نمادهای مدل‌سازی ساختاری UML 2 استفاده مجدد می‌کند که برای مدل‌سازی رده مورد نیاز هستند. MOF 2 هیچ نماد اضافی را تعریف نمی‌کند.

برای تعاریف نمادها به مشخصات ابرساختاری UML [UML2Sup] مراجعه شود.

۶ اطلاعات افزوده

۱-۶ اطلاعات کلی

فراداده ناسازگار و اغلب اختصاصی در بین سامانه‌های مختلف محدودیت اصلی مبادله داده و یکپارچه‌سازی برنامه‌های کاربردی است. فراداده، داده‌ای در مورد داده است. فراداده‌های ناسازگار، داده‌هایی هستند که توسط ابزار، پایگاه‌های داده، میان‌افزار و غیره برای توصیف ساختار و معنای داده استفاده می‌شوند.

ابزار فراشیء (MOF) یک چارچوب مدیریت فراداده باز و مستقل از بن‌سازه و خدمات فراداده مرتبط ارائه می‌دهد تا امکان توسعه و همکاری متقابل[1] سامانه‌های مبتنی بر فراداده و مدل را فراهم کند. ابزار مدل‌سازی و توسعه، سامانه‌های انبار داده، مخزن‌های فراداده و غیره مثال‌هایی از سامانه‌هایی هستند که از MOF استفاده می‌کنند.

1 -interoperability

MOF به اصول هسته‌ای معماری مدل‌گرای OMG کمک زیادی کرده است. MOF که روی پایه مدل‌سازی ایجاد شده توسط UML ساخته شده است، مفهوم فراداده‌های صوری و مدل‌های مستقل از بن‌سازه (PIM) از فراداده (مانند استانداردهای متعدد OMG مانند UML، خود MOG، CWM،SPEM، جاوا EJB،EDOC، EAI و غیره) و نگاشت از PIMها به بن‌سازه‌های خاص (مثال‌هایی از مدل‌های خاص بن‌سازه و نگاشت: نگاشت MOF به متن در مشخصات MOF به متن[MOFM2T]، نگاشت MOF به XML در مشخصات XML [XMI24]، نگاشت طرح‌واره MOF به XML در تولید XMI در مشخات طرح‌واره XML [XMI24] و مشخصات MOF به جاوا در مشخصات JMI) را عرضه کرده است.

اتخاد مشخصات MOF 1.1 توسط OMG در نوامبر سال ۱۹۹۷ همزمان با اتخاذ UML 1.1 شد. در مارس ۲۰۰۳ OMG هسته با معماری به شدت تغییریافته MOF 2.0 را به کار گرفت و سپس از UML 2.0 استفاده کرد. هماهنگی بین MOF و UML در MOF 2.4 و UML 2.4 با اشتراک فرامدل یکسان برای تعریف UML و MOF با استفاده از محدودیت‌های OCL برای تعریف زیرمجموعه فراداده مربوط برای MOF کامل شد. مشخصات MOF به دست آمده در این مستند ارائه شده و برای ایجاد تمایز با مشخصات MOF 1.4 که هنوز به عنوان مشخصات OMG و استاندارد بین‌المللی ISO/IEC 19505 وجود دارد «MOF2» خوانده می‌شود.

MOF 2 با مجموعه‌ای از مشخصات بازنمایی می‌شود: هسته MOF 2 [MOF2]، نگاشت MOF 2 XMI (در حال حاضر عنوان مبادله فراداده XML گرفته است)[XMI24]، ابزار MOF 2 و چرخه حیات شیء [MOFFOL]، نسخه‌بندی MOF 2 چرخه حیات توسعه [MOFVD]، پرس‌وجو/منظر/تبدیلات MOF 2 [QVT]، مدل MOF به متن [MOFM2T].

۲-۶ ساختار مشخصات MOF 2

MOF 2 از قابلیت‌های مدل‌سازی ساختاری UML 2 بر پایه فرامدل مشترک بین UML 2 و MOF 2 استفاده می‌کند. محدودیت‌های OCL این فرامدل را به MOF2 محدود می‌کنند- زیرمجموعه‌های مربوط در بند EMOF و CMOF تعریف شده‌اند. در پیوست ارجاعی به فایل-های با کد منبع OCL ارائه شده است.

بند ۷ مقدمه‌ای برای فرامدل‌سازی و معماری MOF 2 ارائه می‌دهد. بند ۸، MOF 2 را به عنوان یک زبان فرامدل‌سازی معرفی می‌کند.

هسته MOF 2 فرامدل اشتراکی با قابلیت‌های ویژه MOF 2 را توسعه می‌دهد. این‌ها در بندهای ۹ تا ۱۱ این مستند تعریف شده‌اند. این قابلیت‌های MOF 2 عبارت‌اند از:

- انعکاس: مدلی را توسعه می‌دهد که توانایی توصیف خود را دارد.
- شناسانه‌ها: افزونه‌ای برای شناسایی منحصربه‌فرد اشیاء فراداده بدون تکیه بر داده مدل که ممکن است در معرض تغییر باشد ارائه می‌دهد.

- افزونه: ابزاری ساده برای توسعه عناصر مدل با همتاهای نام-مقدار. بندهای بعدی هر کدام از بسته-هایی را که قابلیت‌های پشتیبانی شده را می‌سازند توصیف می‌کنند.

بسته‌های مختلفی که قابلیت‌های MOF 2 را می‌سازند نمونه‌های CMOF::Package هستند و همه محتوای آن نمونه‌هایی از رده‌های داخل مدل CMOF هستند.

فرامدل، یک مدل است که برای مدل‌سازی خودش استفاده شده است. مدل MOF 2 برای مدل‌سازی خودش و سایر مدل‌ها و فرامدل‌ها استفاده شده است (مانند UML 2 و CWM 2 و غیره). فرامدل برای مدل‌سازی فراداده دلخواه نیز استفاده می‌شود (برای مثال، پیکربندی نرم‌افزار یا فراداده نیازمندی‌ها).

مدل MOF 2 از دو بسته اصلی MOF ضروری (EMOF) و MOF کامل (CMOF) تشکیل شده است. EMOF در بند ۱۲ توضیح داده شده است. EMOF برای تطبیق قابلیت‌های زبان‌های برنامه‌نویسی شی‌گرا و نگاشت‌ها به XMI یا JMI طراحی شده است. MOF کامل (CMOF) تمام قابلیت‌های فرامدل‌سازی MOF 2 را ارائه می‌دهد. برای فراهم کردن این امکان، CMOF انعکاس را توسعه می‌دهد که در بند ۱۳ توضیح داده شده است. خود CMOF در بند ۱۴ توضیح داده شده است.

بند ۱۵ یک مدل نمونه انتزاعی از MOF 2 ارائه می‌دهد که به صورت کارآمدی یک مدل مستقل از بن‌سازه برای CMOF ارائه می‌دهد.

پیوست الف پیوندهایی به تعاریف قابل مصرف توسط ماشین از EMOF و CMOF با استفاده از XML ارائه می‌دهد.2 MOF فرامدل مشترکی با UML 2 دارد.

پیوست ب پیوندهایی به فایل‌های منبع OCL به منظور محدودکردن فرامدل UML 2 به EMOF یا CMOF ارائه می‌دهد.

پیوست پ نگاشت الزامی برای مهاجرت از MOF 1.4 به MOF 2 ارائه می‌دهد.

پیوست ت شامل فهرست مراجع نقل قول‌های این مستند است. قالب استفاده شده در این ارجاعات [xyz] است.

پیوست ث شامل اطلاعات قانونی است و پیوست F از نویسندگان MOF 2 تشکر می‌کند.

۷ معماری MOF (آگاهی‌دهنده)

۷-۱ کلیات

این بند معماری MOF را توصیف می‌کند و توضیح می‌دهد چگونه به عنوان پایه مدیریت فراداده مستقل از بن‌سازه برای MDA عمل می‌کند. این بند همچنین تصمیمات معماری اصلی را که روی طراحی MOF 2 تأثیرگذار بودند را به صورت خلاصه بیان می‌کند. در آخر ارتباط MOF 2 با UML 2 و استفاده از MOF برای نمونه‌سازی UML 2 و فرامدل‌های OMG آینده خلاصه شده‌اند. MOF 2 نسلی جدید از MOF است

[MOF2] که با UML 2 هماهنگ شده است [UML2Sup]؛ این جایگزین [MOF1] MOF 1.4 که با UML 1.4[UML1] هماهنگ بود، نیست.

۲-۷ اهداف کلی طراحی MOF 2

مقصد اصلی این نسخه تجدیدنظر شده اصلی MOF، ارائه یک چارچوب فراداده نسل بعد مستقل از بن‌سازه برای MOG است که از یکپارچه‌سازی انجام گرفته در MOF 1.4،XMI 1.2، تولید XMI طرح‌واره‌های XML و JMI 1.0 سود می‌برد. بنیان مدل‌سازی MOF 2 به علت داشتن بینش مشترک به استفاده مجدد از مفاهیم مدل‌سازی هسته‌ای بین UML 2،MOF 2 و سایر فرامدل‌های در حال ظهور OMG تأثیر متقابل زیادی با مشخصات زیرساخت UML 2 دارد. این حقیقت که شرکت‌ها و طراحان یکسان روی هر دو مشخصات کار کرده‌اند و از این اصول طراحی پشتیبانی کرده‌اند، این یکپارچه‌سازی و استفاده مجدد را ممکن کرده است. MOF2 در ادامه سنت MOF از سال ۱۹۹۷ می‌تواند با استفاده از مفاهیم ساده مدل‌سازی رده برای تعریف و تجمیع خانواده‌ای از فرامدل‌ها استفاده شود. همانند MOF1 برای توصیف فرامدل‌های منطبق با MOF از مفهوم مدل‌سازی رده UML استفاده شده است. مسأله برجسته درباره MOF2 این است که ما مفاهیم مدل‌سازی در MOF2 و UML2 را یکپارچه کرده‌ایم و در هر دو مشخصات MOF2 و UML2 از یک کتابخانه زیرساخت UML2 مشترک استفاده کردیم. مزایای اصلی این روش شامل این موارد است:

- قوانین ساده‌تر برای مدل‌سازی فراداده (درک فقط یک زیرمجموعه از مدل‌سازی رده UML بدون هیچ مفاهیم اضافی یا ساختارهای مدل‌سازی).
- نگاشت‌های فناوری مختلف از MOF (مانند JMI،XMI و غیره) حالا به بازه وسیع‌تری از مدل‌های UML مانند پروفایل‌های UML اعمال می‌شوند.
- پشتیبانی وسیع‌تر از ابزار برای فرامدل‌سازی (هر ابزار مدل‌سازی UML را می‌توان ساده‌تر برای مدل‌سازی فراداده استفاده کرد).

در هر مورد می‌توان از MOF2 برای تعریف (بدون نیاز به استفاده مجدد از بسته‌های فراداده خاص) هم فرامدل‌های مبتنی بر شیء و هم مبتنی بر غیرشیء استفاده کرد (همان‌طور که در مورد MOF1 این‌طور بود).

بر پایه تجربه‌های پیاده‌سازهای XMI،MOF و JMI در زمینه فراداده‌های استاندارد صنعتی شناخته‌شده مانند UML و CWM بعضی از اهداف و ملاحظات طراحی مهم عبارت‌اند از:

۱-سادگی استفاده در تعریف و توسعه فرامدل‌ها و مدل‌های زیرساخت نرم‌افزار موجود و جدید. ما به دنبال این بودیم که اطمینان حاصل کنیم تعریف و توسعه فرامدل‌ها و مدل‌های فراداده به سادگی تعریف و توسعه مدل‌های شیء معمول باشند. استفاده از یک هسته مشترک بین UML2، MOF 2 و فرامدل‌های مهم دیگر (CWM، EAI) برای رسیدن به این هدف کلیدی است. انتظار می‌رود

RFPهای آینده برای CWM2، EAI2 و غیره یا از این هسته مشترک استفاده کنند یا تغییراتی را برای ارتقای قابلیت استفاده مجدد، پیشنهاد دهند.

۲-ارتقای بیشتر پیمانه‌ای بودن و قابلیت استفاده مجدد خود مدل MOF. توجه کنید که این نیز یک هدف طراحی مهم برای زیرساخت UML2 بود. از یک جهت ما شروع به کار مدل‌سازی مؤلفه‌گرا کرده‌ایم که خود بسته‌های مدل در چارچوب‌های مدل‌سازی قابل استفاده مجدد هستند. ریشه‌های این کار زمانی آغاز شد که CWM تعریف می‌شد. پیچیدگی مدل‌سازی دامنه مسأله انبارکردن داده استفاده از این روش تقسیم و غلبه را ضروری می‌کرد. پیرایشی که تا اینجا انجام شده است باعث ایجاد مجموعه‌ای پیمانه‌ای‌تر از بسته‌های مناسب برای مدل‌سازی شیء شده است.

۳-استفاده از پیرایش مدل برای ارتقای قابلیت استفاده مجدد مدل‌ها. بعضی از درس‌هایی که یاد گرفته شد تحت تأثیر پیرایش حوزه زبان برنامه‌نویسی در سطح رده بود که در سطح بسیار وسیع‌تری استفاده شده است. همچنین تجربه تیم طراحی CWM و تیم‌های طراحی UML 2 روی کار ما تأثیر می‌گذارند. با وجود اینکه این روش باعث ایجاد تعداد بیشتر بسته‌های با دانه‌بندی مناسب شد، اعتقاد داریم این روش استفاده مجدد و سرعت توسعه فرامدل‌ها و چارچوب‌های مدل‌سازی جدید را ارتقا می‌دهد. یکی از نتایج مستقیم این تلاش استفاده مجدد از زیرمجموعه‌ای از بسته‌های فرامدل با «هسته مشترک» توسط مشخصات MOF 2 و UML 2 است.

۴-تضمین اینکه MOF 2 مستقل از بن‌سازی فناوری است و برای نگاشت از MOF 2 به بن‌سازه‌های فناوری مانند J2EE،.Net، CORBA، خدمت‌های وب و غیره عملی‌تر است.تجربه به‌دست آمده در تعریف مشخصات MOF، XMI و JMI که در حال حاضر بسیاری از نگاشت‌های فناوری را از مدل MOF و به مدل MOF تعریف می‌کند شالوده‌ای محکم برای این تلاش بوده است. اینکه پیاده‌سازی‌های MOF که از نگاشت‌های زبان مختلف استفاده می‌کنند بتوانند با هم همکاری کنند یک هدف طراحی است (برای مثال استفاده از مبادله XML).

۵-تعامد (تفکیک ملاحظات) مدل‌ها و خدمت‌های (خدمات سودمند) اعمال شده به مدل‌ها هدف بسیار مهمی برای MOF 2 است. یکی از تجربه‌های به‌دست آمده در MOF 1 و XMI 1 این بود که طراحی اولیه MOF بیش از حد تحت تأثیر و محدود شده به معناشناسی چرخه حیات CORBA مبتنی بر مخزن‌های فراداده بود. درنتیجه از روشیء با ارتباطات ضعیف‌تر[1] برای مبادله فراداده (و داده) استفاده کرد که محبوبیت XML و XMI این مسأله را نشان می‌دهد. فروشنده‌ها به شکل‌های مختلف از MOF برای تجمیع ابزار توسعه، ابزار انبار داده، ابزار مدیریت برنامه، مخزن‌های متمرکز و توزیعی، درگاه‌های توسعه‌دهنده‌ها و غیره استفاده کردند. مشخص شد که ما برای اینکه در پیاده‌سازی انعطاف‌پذیری فراهم کنیم مجبور بودیم مفاهیم مدل‌سازی را از خدمت‌های فراداده مطلوب مانند مبادله فراداده (استفاده از جریان‌های XML در مقابل استفاده اشیاء Java/CORBA)،بازتاب،

1 - Loosely coupled

همبست[1]، چرخه حیات، نسخه‌بندی، شناسانه، پرس‌وجوها و غیره مجزا کنیم. ما این تعامد مدل‌ها از خدمت‌ها خاصیت بسیار مهم MOF 2 می‌دانیم. به دلیل وجود انتخاب‌های پیاده‌سازی و خدمت‌های متنوع بسیاری از خدمت‌های پیچیده‌تر (همبست، نسخه‌بندی، پرس‌وجو و غیره) در معرض RFPهای OMG بیشتری هستند.

۶-بازتاب مدل‌های MOF 2 با استفاده از خود MOF در مقابل فقط مشخص کردن بازتاب به عنوان مجموعه‌ای از رابط‌های مختص فناوری. این معنی اصلی و واقعی مورد ۵ است که بازتاب به عنوان یک خدمت مستقل مدل‌سازی شود. این روش به روشنی ملاحظات مربوط به بازتاب، مدیریت چرخه حیات و غیره را که در MOF 1 با هم ترکیب شده بودند، از هم مجزا می‌کند.

۷-MOF 2 مفهوم شناسانه را مدل‌سازی می‌کند. نبود این قابلیت در MOF، UML، CWM و غیره پیاده‌سازی قابلیت همکاری فراداده را دشوار کرده بود. نویسندگان می‌دانند که مدل‌سازی شناسانه-های آسان نیست اما می‌خواهند مفید بودن آن را در یک حوزه ساده یعنی شناسانه‌ها برای فراداده نشان دهند. یکی از اهداف طراحی اصلی آسان کردن نگاشت این مدل شناسانه به شناسانه W3C و سازوکارهای ارجاع مانند URI است.

۸-استفاده مجدد از چارچوب‌های مدل‌سازی و بسته‌های مدل در فرالایه‌های مختلف با بسته‌بندی بهتر قابلیت‌های MOF. در تعریف خود MOF، فرامدل‌های مختلف (مانند UML و CWM) و مدل-های کاربر و حتی اشیاء کاربر می‌توان از بعضی از انواع و خدمت‌های رایج استفاده کرد. یک محصول فرعی اصل تعامد این است که بعضی از قابلیت‌های MOF را می‌توان در چند فرالایه به کار برد.

۳-۷ چند فرا لایه؟

در مجموعه استانداردهای OMG که موجب سردرگمی می‌شود انعطاف‌ناپذیری «معماری فرامدل چهار لایه» است که در مشخصات مختلف OMG به آن اشاره می‌شود. توجه کنید که مفاهیم کلیدی مدل‌سازی طبقه-بند و نمونه یا رده و شیء و توانایی هدایت از یک نمونه به فراشیء آن (طبقه‌بند آن) است. این مفهوم اساسی را می‌توان برای مدیریت هر تعداد لایه به کار برد (بعضی اوقات با عنوان فراسطح از آن‌ها یاد می‌شود). رابط-های بازتاب MOF 2 به صورت بازگشتی امکان پیمایش هر تعداد فرالایه را فراهم می‌کنند. دقت داشته باشید که بیشتر سامانه‌ها از مشخصات هسته MOF کوچکی استفاده می‌کنند، v.2 ۹ سطح (معمولاً کمتر از یا مساوی چهار). تعداد لایه‌ها می‌تواند ۲ (سامانه‌های بازتابی عمومی – کلاس/شیء)، ۳ (سامانه‌های پایگاه-داده رابطه‌ای – سامانه‌جدول[2]/جدول/ردیف)، و ۴ (زیرساخت UML2، مشخصات UML 1.4 و MOF 1.4) باشد. MOF 1 و MOF 2 امکان هر تعداد لایه بیشتر یا مساوی ۲ را فراهم می‌کنند. (کمینه تعداد لایه‌ها ۲ است بنابراین می‌توانیم از یک رده به نمونه آن و برعکس حرکت کنیم). MOF 2 و مدل بازتاب آن را می‌توان با ۲ سطح یا هر تعداد سطح که کاربر مشخص کند به کار برد.

---

1 - federation
2 -SysTable

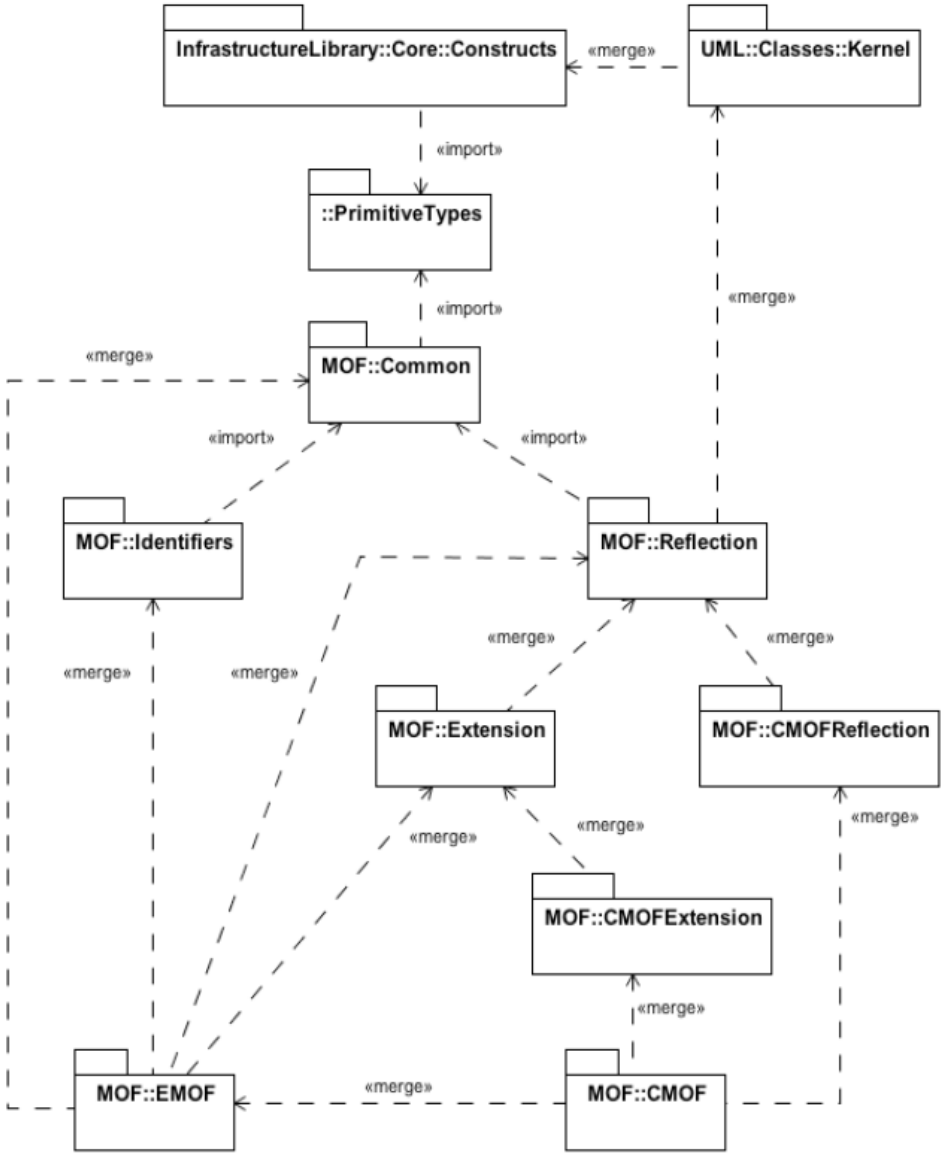
۴-۷ استفاده مجدد از بسته‌های هسته مشترک توسط UML 2 و MOF 2

کتابخانه زیرساخت UML 2 برای راه‌اندازی بقیه UML 2 از بسته‌های با ریزدانگی مناسب استفاده می‌کند. یکی از اهداف طراحی استفاده مجدد از این زیرساخت در تعریف مدل MOF 2 است. در MOF 2 این استفاده مجدد با استفاده از یک سازوکار توسعه‌پذیری CMOF انجام می‌شود – ادغام بسته‌های موجود سازگار با MOF 2. درونبرد کردن[1] بسته‌ها باعث می‌شود عناصر مدل بسته درونبردشده در بسته در حال درونبرد شدن آشکار باشند. ادغام بسته‌ها عناصر مدل در بسته در حال ادغام شدن را با خاصیت‌های جدید بسته‌های ادغام شده توسعه می‌دهد. جزئیات در سند زیرساخت UML 2 در زیربند ادغام بسته پوشش داده شده‌اند. هم مدل UML 2 و هم مدل MOF 2 سازگار با MOF 2 و قابل نمونه‌سازی از MOF 2 طراحی شده‌اند.

مفاهیم مدل‌سازی رده استاندارد (درونبری، زیررده (زیرکلاس)، اضافه کردن رده جدید، ارتباطات و اضافه کردن ارتباطات جدید بین رده‌های موجود) برای قابلیت توسعه MOF 2 استفاده شده‌اند. (این با سازوکار قابلیت توسعه در MOF 1 برابر است). این مفاهیم برای تعریف بسته‌های بیشتر (مانند بازتاب، حدها[2] و شناسانه‌ها) در MOF 2 و هر مدل سازگار با MOF 2 دیگر استفاده شده‌اند.

شکل ۷-۱ نشان می‌دهد که MOF هسته UML را درونبرد می‌کند و سپس هسته را با بسته‌های اضافی توسعه می‌دهد.

1 -import
2 -extent



شکل۱- درونبردهای MOF از هسته UML

۸ صورت‌گرایی[1] زبان

۸-۱ کلیات

این بند فنون استفاده شده برای توصیف MOF را توضیح می‌دهد. MOF هم به صورت متنی و هم به صورت نگاره‌ای توصیف شده است. این استاندارد ملی از ترکیبی از زبان‌ها استفاده می‌کند (زیرمجموعه‌ای از UML، یک زبان محدودیت شی، زبان طبیعی دقیق) تا به صورت دقیق گرامر و معناشناسی‌های انتزاعی MOF را توصیف کند. بر خلاف MOF 1 و UML 1 که از فنون تقریباً متفاوتی استفاده شده بود، مشخصات MOF 2

1 - formalism

از بیشتر صورت‌گرایی‌های زیرساخت UML 2 استفاده می‌کند. به طور خاص EMOF و CMOF هر دو با استفاده از CMOF توصیف شده‌اند که برای توصیف UML 2 نیز استفاده شده است. EMOF به صورت کامل در EMOF با بکارگیری درونبری بسته و ادغام معناشناسی‌های توصیف CMOF آن توصیف شده است. در نتیجه EMOF و CMOF با استفاده از خودشان توصیف شده‌اند و هر کدام از کتابخانه زیرساخت UML 2 مشتق شده یا بخشیء از آن را استفاده کرده‌اند.

۸-۲ مشخصات فرامدل

به بند «صورت‌گرایی زبان» در مستند زیرساخت UML 2 مراجعه شود. مدل CMOF از صورت‌گرایی مشابه استفاده می‌کند و در واقع فقط بسته‌ها را در کتابخانه زیرساخت UML 2 درونبری کرده و ادغام می‌کند. EMOF نسبتاً ساده‌تر است که در زیربند بعدی توضیح داده شده است.

۸-۳ استفاده از بسته‌ها برای بخش‌بندی و توسعه فرامدل‌ها

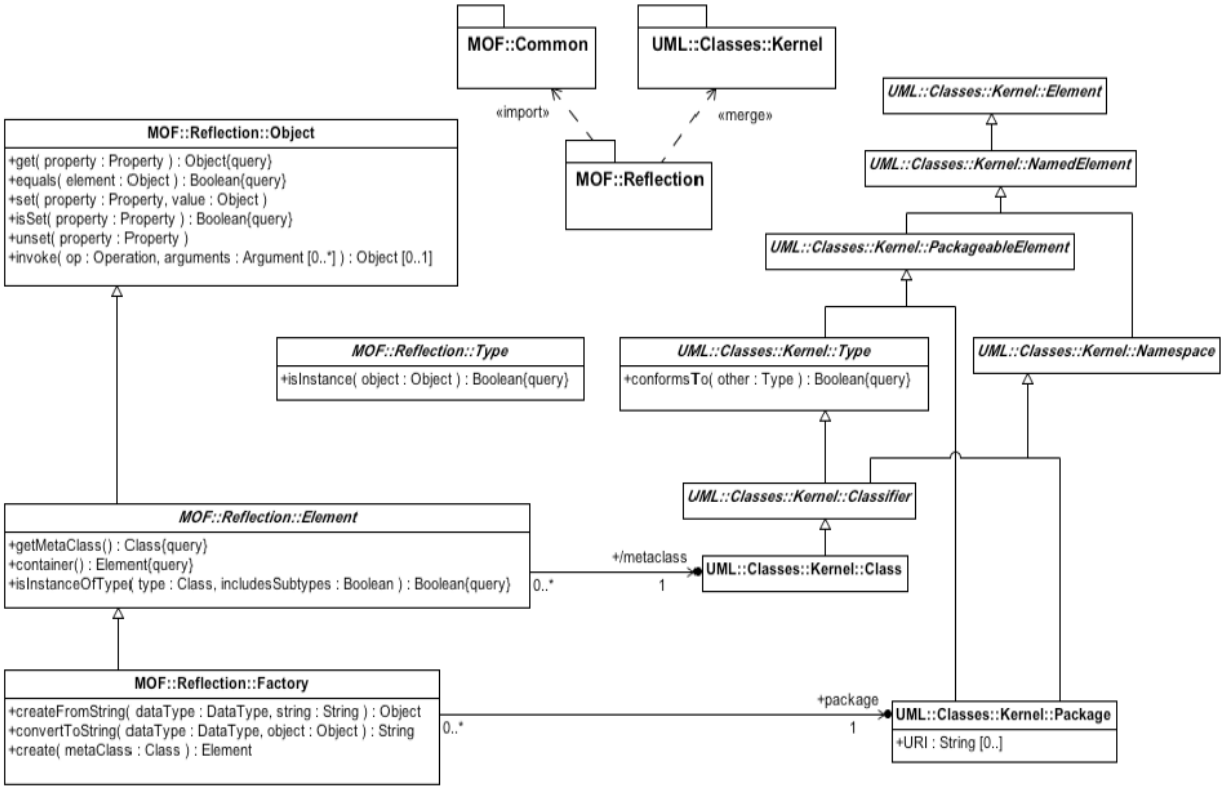
بسته‌های UML2 را می‌توان برای دو منظور استفاده کرد. برای درونبری بسته؛ سازوکاری برای گروهبندی عناصر مدل مرتبط با هم به منظور مدیریت پیچیدگی و آسان کردن استفاده مجدد. ازآنجایی‌که یک بسته یک فضای نام نیز هست عناصر مدل ارجاع‌شده در مرزهای بسته باید با نام کامل بسته خود بیان شوند. درونبری بسته از شدت این محدودیت کم می‌کند: عناصر مدل در بسته درونبری شده به صورت مستقیم در بسته در حال درونبری شدن قابل مشاهده هستند که در آنجا آن‌ها ممکن است در رابطه با سایر عناصر مدل استفاده شوند که با زیرردهایی که خاصیت‌های بیشتری ارائه می‌دهند ویژه شده‌اند. دومین استفاده از بسته-ها آسان کردن ترکیب خاصیت‌های فرامدل‌سازی جدید یا قابل استفاده مجدد برای ساخت زبان‌های مدل‌سازی توسعه‌یافته است. ادغام بسته خاصیت‌های بسته‌ها را با هم ترکیب می‌کند تا قابلیت‌های زبانی یکپارچه جدید تعریف کند. بعد از ادغام بسته رده‌هایی که در بسته در حال ادغام شدن هستند همه خاصیت‌های رده‌های هم‌نام در بسته ادغام‌شده را دارند.

۹ بازتاب

۹-۱ کلیات

فراشیءها امکان استفاده از اشیاء را بدون دانش قبلی از خاصیت‌های خاص اشیاء فراهم می‌کنند. در زمینه MOF رده یک شی (یعنی فراشیء آن) طبیعت شیء را نشان می‌دهد (نوع آن، خاصیت‌های آن). بسته بازتاب امکان این پیدا کردن و دستکاری فراشیءها و فراداده را فراهم می‌کند.

شکل ۹-۱ معماری بازتاب MOF را نشان می‌دهد. رده‌های سمت چپ نمودار توسط MOF::Reflection تولید شده یا توسعه داده شده‌اند و سمت راست نمودار رده‌های مشترک با ابرساختاری UML را نشان می‌دهد. این ارتباط در نمودار بسته تعبیه‌شده نیز نشان داده شده است.



شکل ۲- بسته بازتاب

۲-۹ عنصر

هر عنصر یک رده دارد که خاصیت‌ها و عملیات آن را توضیح می‌دهد. عنصر یک نمونه از این رده است. عنصر Basic::Element را ادغام کرده و توسعه می‌دهد. همه عناصرمدل که Reflection::Element را اختصاصی می‌کنند قابلیت‌های بازتابی را به ارث می‌برند. به طور خاص این شامل همه عناصر مدل زیرساخت UML2 است.

همه شرط‌های زیربند ۱۲-۵ نیز بخشیء از این رفتار بازتابی هستند.

اگر هر عملگر بازتابی تلاش کند بازداری چرخه‌ای ایجاد کند خطای IllegalArgumentException رخ خواهد داد.

خاصیت‌ها

/metaclass: Class

رده‌ای را بازمی‌گرداند که این عنصر را توصیف می‌کند. این یک خاصیت اشتقاقی است که برای راحتی و سازگاری ارائه شده است.

عملیات

getMetaClass () : Class

رده‌ای را بازمی‌گرداند که این عنصر را توصیف می‌کند.

Container (): Element

دربردارنده‌های پدر این عنصر را در صورت وجود بازمی‌گرداند. اگر هیچ عنصر دربردارنده‌ای وجود نداشته باشد تهی بازمی‌گرداند.

محدودیت‌ها

هیچ محدودیت اضافی وجود ندارد.

معناشناسی

عنصر رده ابر رده همه رده‌های تعریف شده در MOF است و ابر رده ضمنی همه فرا رده‌های تعریف شده با استفاده از MOF است: نیازی نیست این ارتباط ابر رده با عنصر به صورت صریح در فرامدل‌های سازگار با MOF مدل‌سازی شود و اگر به این شکل ضمنی عنصر در فهرست ابر رده نیست.

با ساختن خاصیت‌ها با نوع عنصر می‌توان در هر مدل سازگار با MOF به عناصر ارجاع داد که شبیه به xsd:any در طرح‌واره‌های XML است.

هر عنصر می‌تواند به منظور به‌دست آوردن رده‌ای که توصیفی بازتابی از آن عنصر ارائه می‌دهد به فرا رده خود دسترسی داشته باشد. به دلیل اینکه هم MOF و هم نمونه‌های MOF ریشه در رده عنصر دارند، MOF از هر تعداد فرالایه پشتیبانی کند که در بند ۷ «معماری MOF» توضیح داده شد.

در ادامه برهم‌کنش بین مقادیر پیش‌فرض، تهی،isSet و unSet توضیح داده می‌شود.

خاصیت‌های تک‌مقداری:

اگر خاصیت تک‌مقداری یک پیش‌فرض داشته باشد:

- وقتی عنصری ساخته می‌شود مقدار پیش‌فرض در آن قرار می‌گیرد. isSet = false
- اگر مقدار آن خاصیت بعدها به صورت صریح تنظیم شود، isSet =true مگر اینکه به مقدار پیش‌فرض (در صورت وجود) تنظیم شود که isSet=false.
- اگر خاصیت unSet باشد آنگاه مقدار خاصیت به پیش‌فرض باز می‌گردد و isSet=false.

اگر یک خاصیت تک‌مقداری پیش‌فرض نداشته باشد:

- در هنگام ایجاد مقدار آن تهی است. isSet=false
- اگر مقدار خاصیت بعدها تنظیم شود حتی به تهی، isSet=true.
- اگر خاصیت unSet باشد آنگاه مقدار خاصیت به تهی بازمی‌گردد و isSet=false.

خاصیت‌های چندمقداری:

- وقتی عنصر ایجاد می‌شود یک فهرست خالی است (isSet=false).
- اگر فهرست به هر شکلی تغییر داده شود (به جز unSet)، isSet=true.
- اگر فهرست unSet باشد، پاک شده است و تبدیل به یک فهرست خالی می‌شود (isSet=false).

پیاده‌سازی isSet به پیاده‌ساز بستگی دارد. برای مقادیر پیش‌فرض به منظور دستیابی به فراداده در زمان اجرا نیازی به پیادی‌سازی‌ها نیست. کافیست که در رده پیاده‌سازی یک ثابت برای پیش‌فرض تولید شود.

پایه منطقی

عنصر در بازتاب بسته معرفی شده است به گونه‌ای که می‌توان آن را Core::Basic ترکیب کرد تا EMOF تولید شود که می‌توان آن را با CMOF ادغام کرد تا قابلیت بازتابی را برای MOF و همه نمونه‌های MOF فراهم کرد.

۹-۳ کارخانه[1]

عنصر را می‌توان از یک کارخانه ساخت. یک کارخانه یک نمونه از رده کارخانه MOF است. یک کارخانه نمونه‌های نوع‌های یک بسته را می‌سازد.

خاصیت‌ها

- Package:Package[1] بسته‌ای را که این کارخانهی برای آن است بازمی‌گرداند.

عملیات

createFromString (dataType: DataTape,string: String): Object

یک شیء می‌سازد که از مقدار رشته ایجاد شده است. اگر امکان ساخت وجود نداشته باشد تهی بازمی-گرداند.قالب رشته توسط SimpleTypeطرح XML متناظر با آن نوع داده‌ای تعریف شده است.

- استثنا: اگر نوع داده‌ای تهی باشد NullPointerException.
- استثنا: اگر نوع داده‌ای عضو بسته بازگشتی توسط () getPackage نباشد IllegalArgumentException.

convertToString (datatype: DataType, object: Object): String

یک بازنمایی رشته‌ای از شیء ایجاد می‌کند. اگر امکان ایجاد وجود نداشته باشد تهی بازمی‌گرداند. قالب رشته توسط SimpleTypeطرح XML متناظر با آن نوع داده‌ای تعریف شده است.

- استثنا: اگر نوع داده‌ای عضو بسته بازگشتی توسط () getPackage نباشد یا شیء نمونه معتبری از آن نوع داده‌ای نباشد IllegalArgumentException.

1 -factory

Create (metaClass: Class): Element

عنصری می‌سازد که نمونه‌ای از فرا رده است. metaClass == Object::metaClass و metaClass.isInstance (object) == true.

همه خاصیت‌های عنصر unset در نظر گرفته می‌شوند. مقادیر یکسان هستند گویی object.unset (property) برای همه خاصیت‌ها فراخوانی شده است.

اگر امکان ساخت وجود نداشته باشد تهی بازمی‌گرداند. رده‌هایی که abstract = true همیشه تهی بازمی-گردانند.

metaClass ساخته شده عناصر برابر با metaClass است.

- استثنا: اگر رده تهی باشد NullPointerException.
- اگر نوع داده‌ای عضو بسته بازگشتی توسط () getPackage نباشد IllegalArgumentException.

محدودیت‌ها

شرایط زیر در مورد metaClass: Class و همه خاصیت‌های آن باید قبل از اینکه بتوان از آن یک نمونه ساخت باید ارضا شوند. اگر این نیازمندی‌ها ارضا برآورده نشوند () create ایجاد خطا می‌کند.

[1] فراشیء باید تنظیم شود.

[2] نام باید کمینه یک حرف داشته باشد.

[3] نوع خاصیت باید مشخص شود.

[4] Property: 0 <= LowerBound <= UpperBound required.

[5] Property: 1 <= UpperBound required.

[6] خاصیت‌ها فقط خواندنی در EMOF اختیاری هستند.

[7] خاصیت‌های نوع رده نمی‌توانند پیش‌فرض داشته باشند.

[8] خاصیت‌های چندمقداری نمی‌توانند پیش‌فرض داشته باشند.

[9] Property: Container end must not have upperBound >1, a property can only be contained in one container.

[10] Property: Only one end may be composite

[11] انتهاهای مرکب دو سویه باید به هم ارجاع دهند.

[12] مقدار پیش‌فرض خاصیت باید نوع داده‌ای منطبق با نوع‌داده‌ای خاصیت داشته باشد.

موردهای ۳ تا ۱۲ به همه خاصیت‌های رده اعمال می‌شوند.

این شرایط به همه ابر رده‌های رده‌ای که در حال ایجاد است اعمال می‌شوند.

تغییرات از MOF 1.4

بدون تغییر.

۹-۴ شیء

بازتاب شیء را به عنوان یک ابرنوع عنصر معرفی می‌کند تا بتواند نوعی را داشته باشد که نمایشگر هم عناصر و هم مقادیر داده باشد. شیء نمایشگر هر مقداری است و همان java.lang.Object در جاوا است.

۹-۴-۱ عملیات

Equals (object: Object): Boolean

برابر بودن شیء را با این نمونه شیء مشخص می‌کند. برای نمونه‌های رده اگر شیء و این نمونه شیء به یک شیء ارجاع داشته باشند درست را بازمی‌گرداند. برای نمونه‌های نوع داده‌ای اگر شیء مقدار مشابهی با نمونه شیء داشته باشد درست بازمی‌گرداند.

get (property: Property) : Object

مقدار خاصیت داده شده را می‌گیرد. اگر خاصیت حد بالای تعدد[1] ۱ داشته باشد () get مقدار خاصیت را بازمی‌گرداند. اگر خاصیت حد بالای تعدد بزرگتر از ۱ داشته باشد () get یک مجموعه بازتابی شامل مقادیر خاصیت را بازمی‌گرداند. اگر مقداری وجود نداشته باشد مجموعه بازتابی نتیجه خالی است.

- استثنا: اگر خاصیت عضو رده از () class نباشد IllegalArgumentException.

set (property: Property, object: Object)

اگرحد بالایی خاصیت برابر با ۱ باشد () set به صورت تجزیه‌ناپذیر مقدار خاصیت را به پارامتر شیء به‌روزرسانی می‌کند. اگر خاصیت حد بالایی تعدد بزرگتر از ۱ داشته باشد شیء باید یک نوع از مجموعه بازتابی باشد. این رفتار همانند عملیات زیر هستند که به صورت تجزیه‌ناپذیر انجام شده‌اند:

```
ReflectiveSequence list = element.get (property);
List.clear ();
List.addAll ( (ReflectiveSequence) object);
```

مقدار بازگشتی وجود ندارد.

- استثنا: اگر خاصیت عضوی از رده () getMetaClass نباشد IllegalArgumentException.
- استثنا: اگر نوع خاصیت (element) isInstance نادرست برگرداند و خاصیت حد بالایی برابر با ۱ داشته باشد ClassCastException.

1 - multiplicity

- استثنا: اگر عنصر مجموعه بازتابی نباشد و خاصیت حد بالای بیشتر از یک داشته باشد ClassCastException.
- استثنا: اگر عنصر تهی باشد، خاصیت از نوع رده باشد و حدبالایی بیشتر از یک باشد IllegalArgumentException.

isSet (property: Property): Boolean

اگر خاصیت حد بالایی ۱ داشته باشد () isSet درست بازمی‌گرداند اگر مقدار خاصیت متفاوت از مقدار پیش‌فرض خاصیت باشد. اگر خاصیت حد بالایی بیشتر از ۱ داشته باشد () isSet درست بازمی‌گرداند اگر تعداد اشیاء فهرست بیشتر از ۰ باشد.

- استثنا: اگر خاصیت عضو رده () getMetaClass نباشد IllegalArgumentException .

Unset (property: Property)

اگر خاصیت حد بالایی ۱ داشته باشد () unset به صورت تجزیه‌ناپذیر مقدار خاصیت را به مقدار پیش‌فرض برای خاصیت‌های نوع داده‌ای و تهی برای خاصیت‌های نوع رده تنظیم می‌کند. اگر خاصیت حد بالای بیشتر از ۱ داشته باشد () unset مقادیر مجموعه بازتابی خاصیت را پاک می‌کند. این رفتار همانند این است که این عملیات به صورت تجزیه‌ناپذیری اجرا شوند:

ReflectiveCollection list = object.get (property);
List.clear ();

مقدار بازگشتی وجود ندارد.

بعد از فراخوانی () unset ، object.isSet (property) == false

- استثنا: اگر خاصیت عضو رده () getMetaClass نباشد IllegalArgumentException .

۱۰ شناسانه‌ها

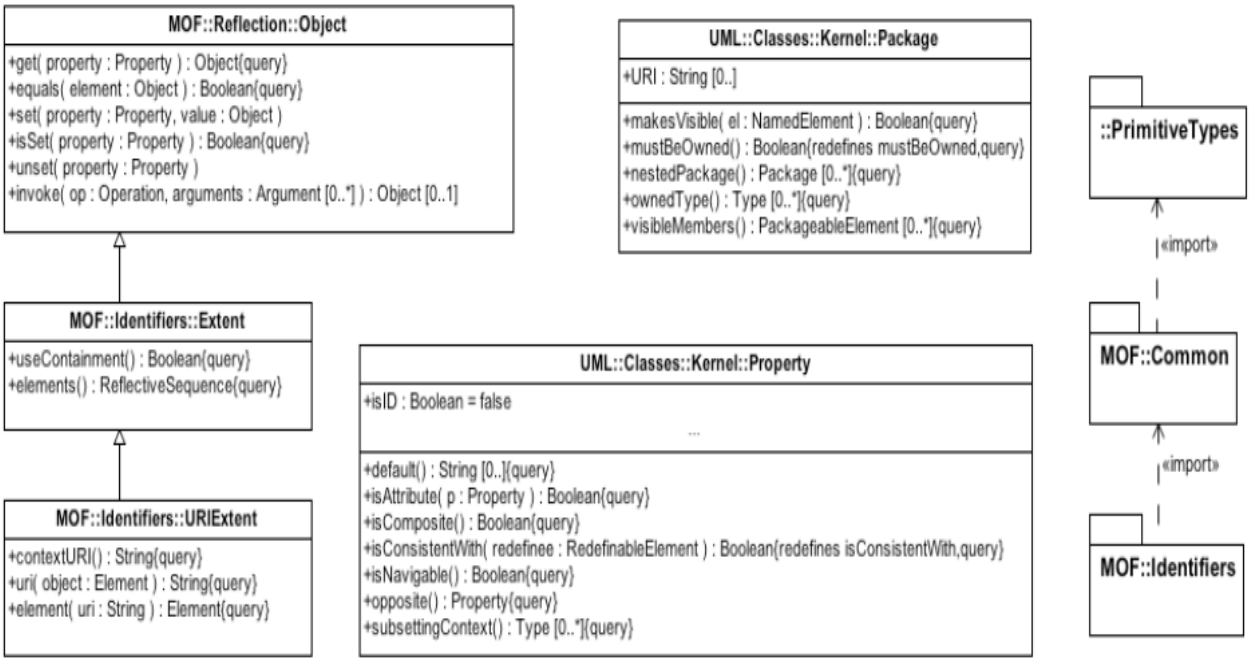
۱-۱۰ کلیات

هر عنصر یک شناسانه در زمینه یک حد دارد که به روشنی آن را از سایر عناصر متمایز می‌کند.

استفاده‌های عملی برای شناسانه‌های شیء وجود دارد. شناسانه‌ها می‌توانند سریال‌سازی ارجاع‌ها به اشیاء خارجی را برای مبادله ساده کنند. آن‌ها می‌توانند به هماهنگ ساختن به‌روزرسانی‌های داده که در آن‌ها تکرار وجود دارد کمک کنند و شناسایی روشنی از اشیاء در ارتباط مانند رابط‌های کاربری ارائه می‌دهند. شناسانه‌ها در جایی که پیاده‌سازی‌ها ممکن است چند شیء پیاده‌سازی داشته باشند که باید یک شیء در نظر گرفته شوند امکان مقایسه برای شناسایی را فراهم می‌کنند. شناسانه‌ها با ارائه یک شناسانه تغییرناپذیر که می‌توان از آن برای همبسته‌کردن عناصر مدل در سراسر تبدیلات مدل در جایی که هم مدل‌های منبع و هم هدف در معرض تغییر هستند استفاده کرد توسعه مدل‌گرا را نیز آسان می‌کنند. تطبیق مدل به مدل به

ابزاری برای تعیین چگونگی نگاشت عناصر مدل که به داده کاربر (مانند نام) وابسته نباشد که در معرض تغییر باشد نیاز دارد.

شکل ۱۰-۱ معماری شناسانه‌های MOF را نشان می‌دهد. رده‌های سمت چپ نمودار توسط MOF::Identifiers معرفی شده و از MOF::Object مشتق شده‌اند. این‌ها قابلیت‌های MOF هستند نه عناصر مدل. سمت راست نمودار بسته و خاصیت، صفت‌های URI و isID آن‌ها که در MOF ایجاد شده‌اند و از این نسخه به بعد با ابرساختاری UML مشترک هستند را نشان می‌دهد. نمودار بسته در سمت راست موقعیت شناسانه‌های بسته در پشته بسته‌ای را نشان می‌دهد.



شکل۳ - بسته شناسانه‌ها

۱۰-۲ گستره

یک حد زمینه‌ای است که می‌توان در آن یک عنصر را در میان مجموعه‌ای از عناصر شناسایی کرد. هر عنصر می‌توان عضو صفر یا بیشتر حد باشد. حد یک عنصر نیست بلکه بخشیء از قابلیت MOF است.

خاصیت‌ها

خاصیت‌های دیگری ندارد.

عملیات

useContainment (): Boolean

وقتی درست باشد به شکل بازگشتی همه عناصر اعضای () elements را شامل می‌شود.

Elements (): ReflectiveSequence

یک دنباله بازتابی از عناصری که به صورت مستقیم توسط این حد ارجاع می‌شوند بازمی‌گرداند. اگر exclusive () درست باشد این عناصر باید container () تهی داشته باشند. عملگر Extent.elements () یک عملگر بازتابی است و ارجاعی بین حد و عنصر نیست.

محدودیت‌ها

محدودیت بیشتری وجود ندارد.

معناشناسی‌ها

وقتی عنصر ساخته می‌شود به هیچ حدی تخصیص داده نمی‌شود.

پایه منطقی

حدها زمینه‌ای فراهم می‌کنند که در آن عناصر MOF را می‌توان مستقل از هر مقدار در عنصر شناسایی کرد.

۳-۱۰ URIExtent

یک حد که شناسانه‌های URI فراهم می‌کند. یک URIExtent می‌تواند یک URI داشته باشد که ممکن است در تعیین شناسانه‌ها برای عناصر شناسایی شده در حد استفاده شود. پیاده‌سازی‌ها ممکن است از مقادیر خاصیت‌های با isID==true در تعیین شناسانه یک عنصر استفاده کنند..

خاصیت‌ها

هیچ خاصیت اضافی ندارد.

عملیات

contextURI (): String

شناسانه‌ای برای حد مشخص می‌کند که یک زمینه URI برای شناسایی عناصر در زمینه ایجاد می‌کند. از یک URI تخصیص داده شود حد یک شناسانه دارد. URI در IETF RFC-2396 تعریف شده و در آدرس http://www.ietf.org/rfc/rfc2396.txt در دسترس است.

Uri (element: Element): String

URI عنصر داده شده در حد را بازمی‌گرداند. اگر عنصر در حد نباشد تهی بازمی‌گرداند.

Element (uri: String): Element

عنصر شناسایی شده توسط URI در حد را بازمی‌گرداند. اگر با URI داده شده هیچ عنصری در حد نباشد تهی بازمی‌گرداند. دقت داشته باشید که این عنصر لزوماً شامل خاصیت متناظر با URI نیست. URI عنصر را در زمینه حد شناسایی می‌کند. همان عنصر ممکن است شناسانه متفاوتی در یک حد دیگر داشته باشد.

محدودیت‌ها

هیچ محدودیت اضافی وجود ندارد.

معناشناسی

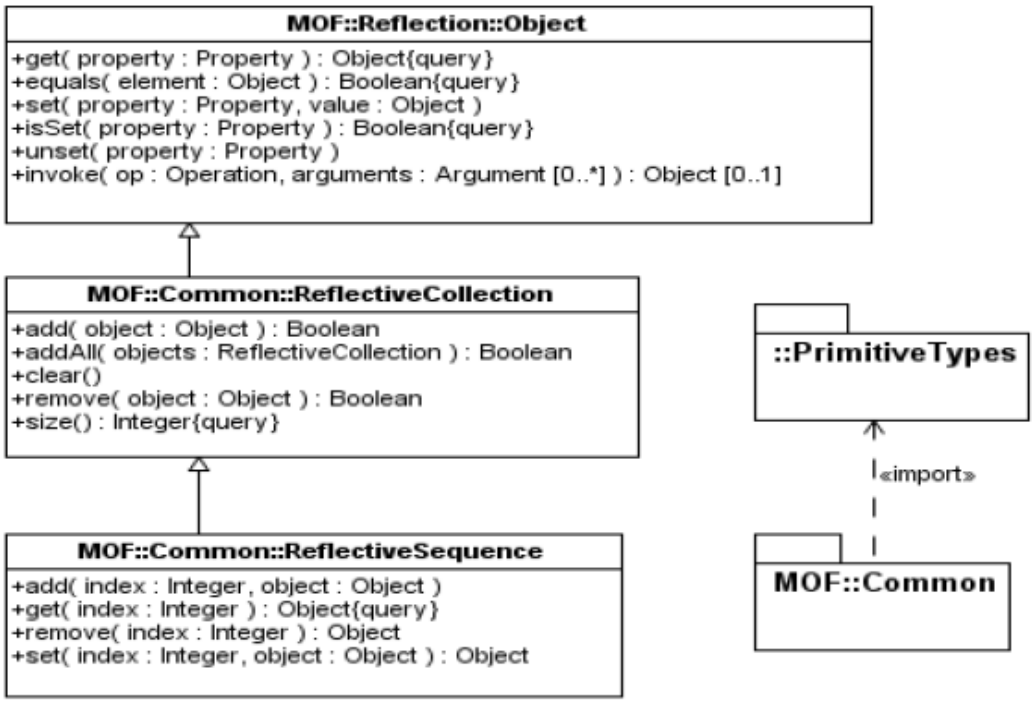
URI می‌تواند مقدار خاصیت‌هایی که به عنوان شناسانه علامتگذاری شده‌اند را ترکیب کند (isID==true).

پایه منطقی

URIها شناسانه‌های دارای استاندارد غیررسمی هستند. آن‌ها برای شناسایی عناصر MOF و هدایت پیوندها بین آن‌ها مفید هستند.

۴-۱۰ MOF::Common

بسته MOF::Common شامل خاصیت‌های داخلی MOF برای مدیریت هستارهای چندمقداری است. این خاصیت آنگونه که در شکل ۱۰-۲ نشان داده شده است در سرتاسر MOF استفاده شده‌اند اما عناصر مدل نیستند. نمودار بسته شکل ۱۰-۲ استفاده از بسته انواع اولیه مشترک بین UML و MOF را نشان می‌دهد.



شکل ۴- بسته مشترک

۵-۱۰ مجموعه بازتابی

مجموعه بازتابی یک رده بازتابی برای دستیابی به خاصیت‌هایی است که بیشتر از یک مقدار ممکن دارند. این مجموعه در بسته MOF::Common تعریف شده تا قابلیت استفاده در بسیاری از قابلیت‌های MOF را آسان کند.

برای خاصیت‌های ترتیبی باید دنباله بازتابی برگردانده شود (به ادامه متن مراجعه شود).

اصلاحات انجام شده در مجموعه بازتابی مقادیر شیء برای آن خاصیت را به صورت تجزیه‌ناپذیر به‌روزرسانی می‌کند.

- استثنا: اگر نوع خاصیت isInstance (Element) نادرست باشد، ClassCastException برگرداند.

Add (object: Object): Boolean

شیء را به انتهای مجموعه اضافه می‌کند. اگر شیء اضافه شده باشد درست بازمی‌گرداند.

addAll (elements: ReflectiveSequence):Boolean

اشیاء را به انتهای مجموعه اضافه می‌کند. اگر هر کدام از عناصر اضافه شده باشند درست بازمی‌گرداند.

Clear ()

همه اشیاء را از مجموعه حذف می‌کند.

Remove (object: Object): Object

شیء مشخص شده را از مجموعه حذف می‌کند. اگر شیء حذف شده باشد درست بازمی‌گرداند.

Size (): Integer

تعداد اشیاء در مجموعه را بازمی‌گرداند.

۱۰-۶ دنباله بازتابی

دنباله بازتابی زیررده (زیرکلاس)ی از مجموعه بازتابی است که برای دسترسی به خاصیت‌های ترتیبی با بیشتر از یک مقدار ممکن استفاده می‌شود. اصلاحات انجام شده در دنباله بازتابی مقادیر عناصر را برای آن خاصیت به صورت تجزیه‌ناپذیری به‌روزرسانی می‌کند.

- استثنا: اگر عنصری که می‌خواهد به مجموعه اضافه شود تکراری باشد و Property.isUnique ()==true آنگاه IllegalArgumentException رخ می‌دهد.
- استثنا: اگر از اندیسی خارج از بازه ۰ تا اندازه دنباله استفاده شود IndexOutOfBoundsException رخ می‌دهد.
- استثنا: اگر عنصری که می‌خواهد به فهرست اضافه شود تکراری باشد و Property.isUnique ()==true آنگاه IllegalArgumentException رخ می‌دهد.

add (index: Integer, object: Object)

شیء را به اندیس مشخص شده در دنباله اضافه کرده و سایر اشیاء را جابه‌جا می‌کند.

get (index: Integer): Object

شیء موجود در اندیس داده شده در دنباله را بازمی‌گرداند.

remove (index: Integer): Object

شیء موجود در اندیس مشخص شده را از دنباله حذف می‌کند. شیء حذف شده را بازمی‌گرداند.

Set (index: Integer, object: Object): Object

شیء موجود در اندیس مشخص شده را با شیء جدید جایگزین می‌کند. شیء حذف شده را بازمی‌گرداند. رفتار عملیات تعریف شده در مجموعه بازتابی هنگامی که برای دنباله بازتابی استفاده شوند به این شکل است:

Add (object: Object): Boolean

شیء را به انتهای دنباله اضافه می‌کند. اگر شیء اضافه شده باشد درست بازمی‌گرداند.

addAll (objects: ReflectiveCollection): Boolean

تمام شیءهای پارامتر مجموعه را به انتهای دنباله هدف اضافه می‌کند:

- به همان ترتیب این کار را انجام می‌دهد ولی اگر پارامتر ترتیب نداشته باشد ترتیب تصادفی است.
- اگر هدف یک دنباله منحصربه‌فرد است آنگاه فقط درصورتی که در حال حاضر وجود نداشته نباشند، این شامل اشیایی می‌شود که از پارامتر اضافه شده‌اند که این باعث می‌شود تکراری‌های پارامتر در صورتی که منحصربه‌فرد نباشد حذف شوند.

اگر همه اشیاء اضافه شوند درست بازمی‌گرداند.

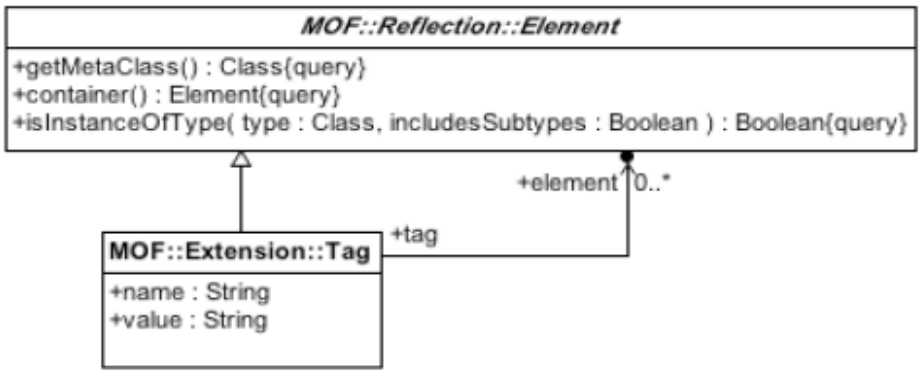
Remove (object: Object): Boolean

اولین نمونه از شیء مشخص شده را از دنباله حذف می‌کند.

۱۱ بسط

۱-۱۱ کلیات

مدل‌های MOF امکان تعریف عناصر فرامدل مانند رده‌هایی که خاصیت‌ها و عملیاتی دارند را فراهم می‌کنند. اما بعضی اوقات حاشیه‌نویسی پویای عناصر مدل با اطلاعات اضافی (شاید پیش‌بینی نشده) ضروری است. این اطلاعات می‌تواند شامل اطلاعات از دست رفته در مدل یا داده مورد نیاز یک ابزار خاص باشند. قابلیت بسط MOF سازوکار ساده‌ای برای ارتباط دادن یک مجموعه نام-مقدار با عناصر مدل ارائه می‌دهد تا به این نیاز رسیدگی شود. این در شکل ۱۱-۱ نشان داده شده است.



شکل ۵ - بسته بسط

۲-۱۱ برچسب

یک برچسب اطلاعاتی است که می‌توان آن را با هر تعداد از عناصر مدل مرتبط کرد. یک عنصر مدل می‌تواند با برچسب‌های زیادی مرتبط شود و همان برچسب می‌تواند با عناصر مدل زیادی مرتبط شود.

خاصیت‌ها

- نام: رشته – نام استفاده شده برای متمایز کردن برچسب‌ها مرتبط با یک عنصر مدل.
- مقدار: رشته – مقدار برچسب. MOF هیچ معنایی به این مقادیر نمی‌دهد.
- عناصر: Element[0..*] – عناصری که برچسب به آن‌ها اعمال شده است.
- مالک: Element[0..1] – عنصری که مالک برچسب است (به منظور مدیریت).

عملیات

هیچ عملگر اضافی وجود ندارد.

محدودیت‌ها

هیچ محدودیت اضافی وجود ندارد.

معناشناسی

یک برچسب نشانگر یک مقدار دارای نام است که می‌تواند با صفر یا چند عنصر مدل مرتبط شود. یک عنصر مدل نمی‌تواند بیشتر از یک برچسب با همان نام داشته باشد. شیوه قرار دادن عنصر مدل توسط MOF مشخص نشده است.

ممکن است مالک برچسب عنصر دیگری باشد. این می‌تواند یک بسته باشد که برچسب‌ها به صورت خارجی به مدل اعمال شده‌اند یا یکی از عناصری که برچسب به آن‌ها اعمال شده است.

پایه منطقی

همتاهای نام-مقدار رشته‌ای ساده برای مدل‌های MOF بسط‌پذیری فراهم می‌کنند که بازه وسیعی از نیازمندی‌ها را پوشش می‌دهند. استفاده از آن‌ها برای این است که نیاز به تعریف مجدد فرامدل‌ها به منظور ارائه افزونه‌های پویا و ساده کاهش پیدا کند.

۱۲ مدل MOF ضروری (EMOF)

۱-۱۲ کلیات

این بند MOF ضروری را توضیح می‌دهد که زیرمجموعه‌ای از MOF است که بسیار با امکانات OOPL و XML متناظر است. ارزش MOF ضروری در این است که چارچوب ساده‌ای برای نگاشت مدل‌های MOF به پیاده‌سازی‌های مانند JMI و XMI برای فرامدل‌های ساده فراهم می‌کند. هدف اصلی EMOF این است که بتوان با استفاده از مفاهیم ساده فرامدل‌های ساده را تعریف کرد درحالیکه از بسط‌ها (با سازوکار بسط رده معمول در MOF) برای فرامدل‌سازی پیچیده‌تر با استفاده از CMOF پشتیبانی می‌کند. هم EMOF و هم CMOF (در بند بعدی توضیح داده شده است) از کتابخانه زیرساخت UML2 استفاده می‌کنند. انگیزه این هدف برداشتن مانع برای ورود به توسعه ابزار مدل‌گرا و تجمیع ابزار است.

مدل EMOF از مدل‌های رده محدودشده UML 2 استفاده می‌کند و قابلیت‌های زبانی بیشتری که در این استاندارد ملی تعریف شده‌اند را نیز شامل می‌شود. همان‌طور که در شکل ۱۲-۱ نشان داده شده است مدل EMOF بسته‌های قابلیت بازتاب،شناسانه‌ها و بسط را با هم ادغام می‌کند تا خدمت‌هایی برای کشف، دستکاری، شناسایی و توسعه فراداده ارائه دهد. بسته MOF::Common نیز برای فراهم کردن خاصیت‌های داخلی MOF ادغام شده است.

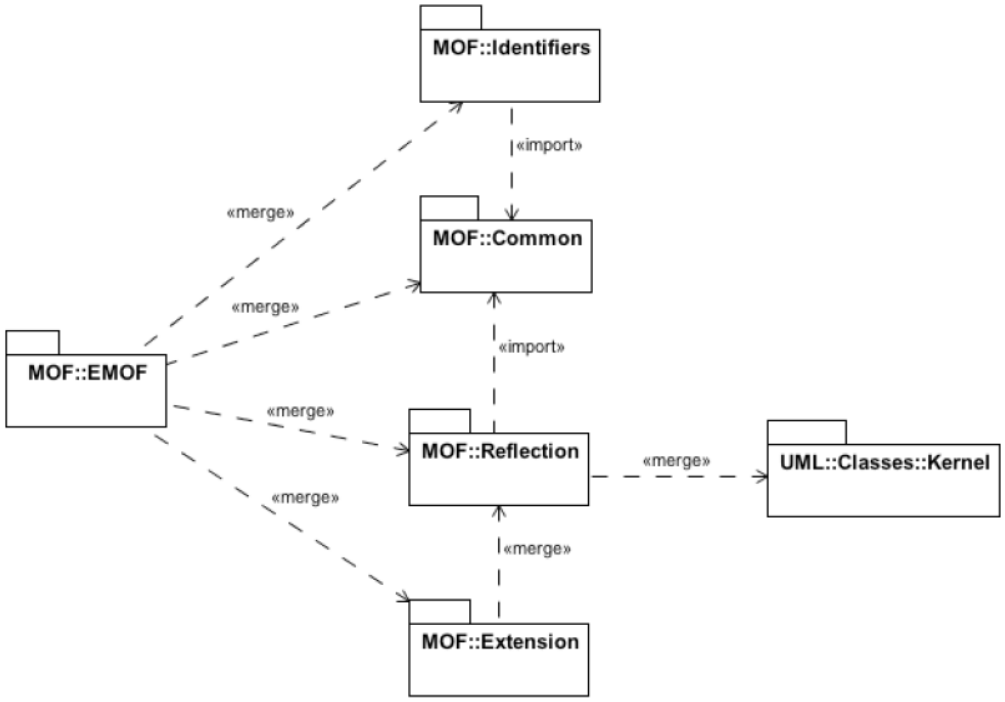
EMOF همانند همه فرامدل‌های در خانواده MOF 2 و UML 2 به عنوان یک مدل UML توصیف شده است. اما پشتیبانی کامل EMOF نیاز دارد تا به طور کامل در خودش تعیین شود، تمام ادغام‌ها و بازتعریف-های بسته که در مدل UML تعیین شده‌اند باید حذف شوند. این بند مدل UML از EMOF و مدل EMOF کامل و ادغام شده را ارائه می‌دهد. این منجر به ایجاد یک مدل کامل و مستقل از EMOF می‌شود که هیچ وابستگی به بسته‌های دیگر یا قابلیت‌های فرامدل‌سازی که توسط خود EMOF پشتیبانی نشده‌اند ندارد.

یادآوری- معناشناسی‌های انتزاعی که در «معناشناسی‌های انتزاعی CMOF» مشخص‌شده‌اند برای EMOF اختیاری هستند.

ارتباط بین EMOF و InfrastructureLibrary::Core::Basic نیاز به توضیح بیشتر دارد. EMOF پایه را با قابلیت‌های MOF و بسط‌هایی از خودش که در زیر توصیف شده‌اند را ادغام می‌کند. به صورت ایده‌آل EMOF باید فقط پایه را با استفاده از زیررده‌هایی که خاصیت‌ها و عملیات اضافی ارائه می‌دهند بسط می‌داد. سپس می‌شد به صورت صوری EMOF را در EMOF مشخص کرد بدون اینکه نیازی به ادغام بسته باشد.

اما این کافی نیست زیرا بازتاب باید شیء را در سلسله‌مراتب رده به عنوان یک ابر رده Basic::Element معرفی کند که نیاز به ادغام دارد. در نتیجه این ادغام EMOF یک مدل مجزا است که پایه را ادغام می‌کند اما از آن ارث‌بری نمی‌کند.

EMOF با استفاده از PackageMerge به صورت مستقیم با فایل‌های Basic XMI سازگار است. تعریف EMOF با استفاده از ادغام بسته همچنین تضمین می‌کند که EMOF با هر تغییر در پایه به‌روزرسانی می‌شود. دلیل مشخص کردن مدل کامل و ادغام شده EMOF در این بند ارائه یک فرامدل است که می‌تواند برای راه‌اندازی ابزار فرامدل که در EMOF ریشه دارند استفاده شود بدون اینکه به یک پیاده‌سازی از CMOF و معناشناسی‌های ادغام بسته نیاز باشد.



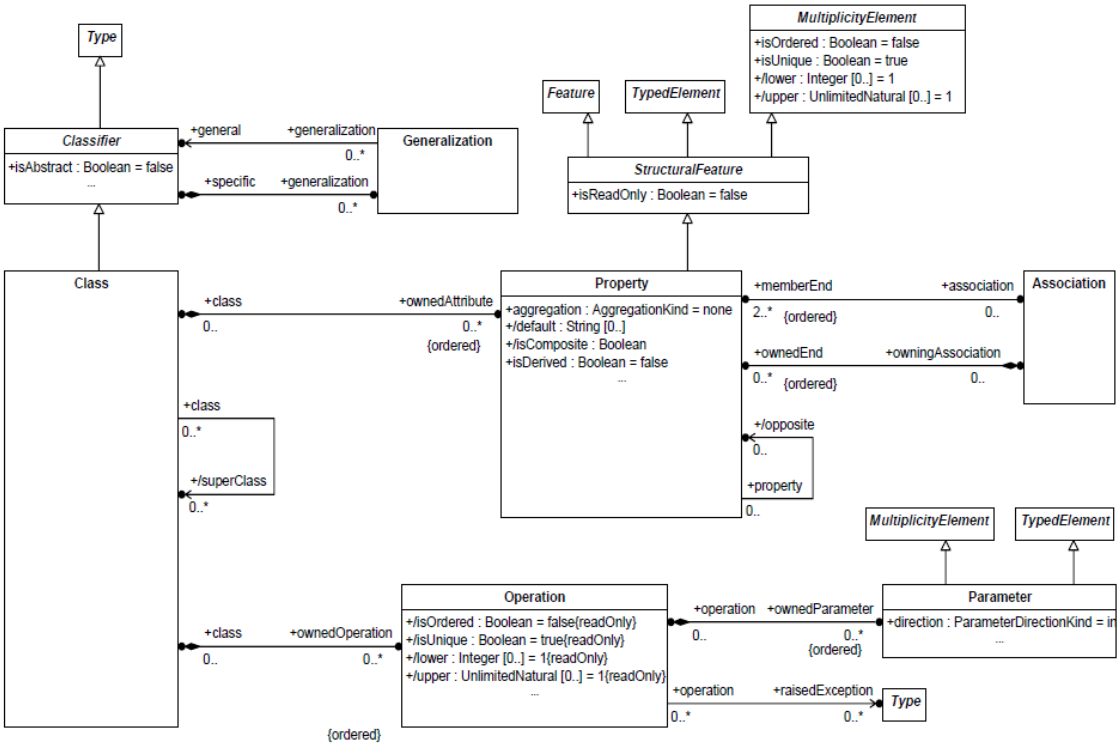
شکل ۶- نگاه کلی به مدل EMOF

EMOF از کمینه عناصر مورد نیاز برای مدل‌سازی سامانه‌های شی‌گرا استفاده می‌کند. EMOF از بسته هسته UML 2 به همان شکلی که هست برای ساختار فرامدل بدون هیچ بسطی استفاده می‌کند، البته یک سری محدودیت‌ها را اعمال می‌کند.

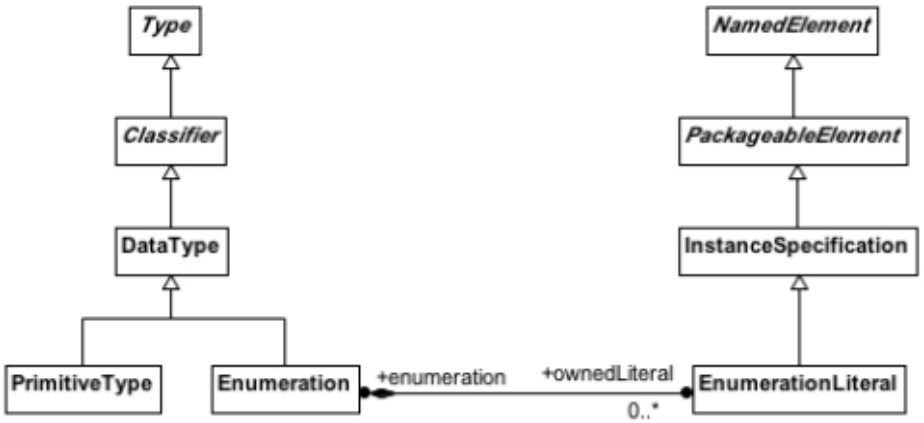
۲-۱۲ مدل ادغام شده EMOF

این زیربند مدل هم‌ارز EMOF را ارائه می‌دهد که با ادغام قابلیت‌های MOF با UML و حذف رده‌ها و خاصیت‌هایی که توسط محدودیت‌های EMOF مستثنی‌شده یا نیاز است خالی باشند به‌دست آمده است. بنابراین بخش‌هایی از UML است که می‌توانند برای مشخص‌کردن فرامدل‌های EMOF استفاده شوند. این مدل بعد از اعمال معناشناسی‌های ادغام بسته به طور کال در خود EMOF تعیین شده است. توضیحات عناصر مدل همان است که در UML است و اینجا تکرار نمی‌شود.

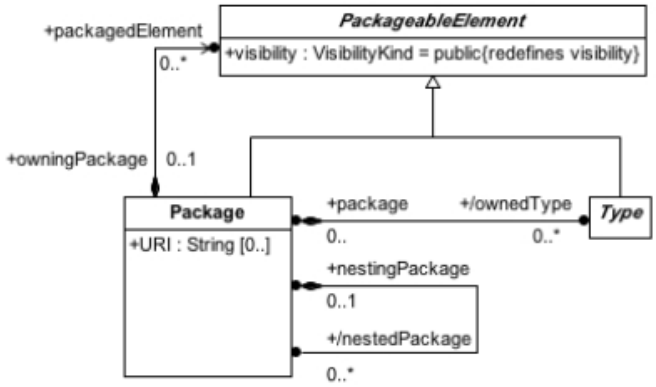
نتایج ادغام قابلیت‌های توصیف شده در زیربند بعدی در بعضی از نمودارها نیز نشان داده شده‌اند.



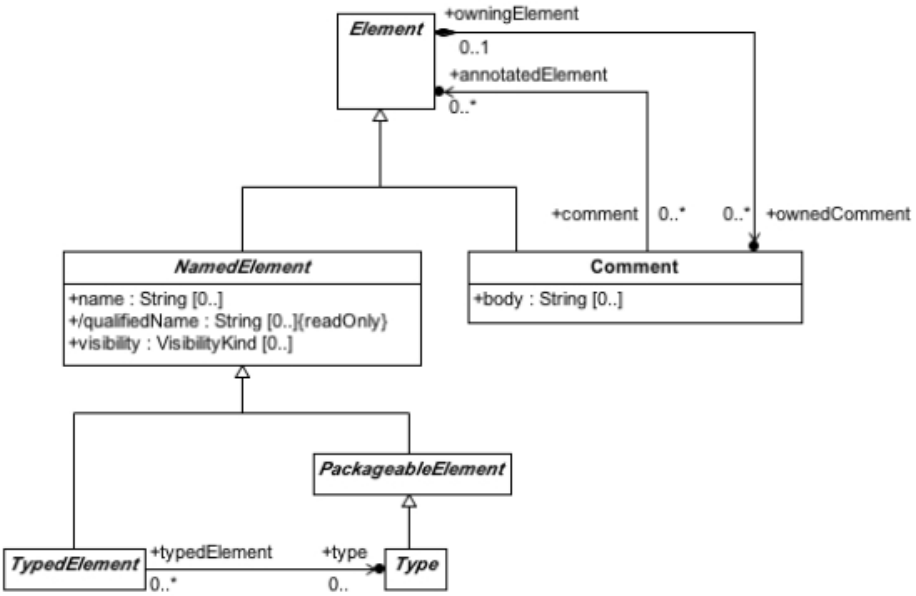
شکل ۷- رده‌های EMOF



شکل ۸- انواع داده‌ای EMOF



شکل ۹ - بسته EMOF



شکل ۱۰ - انواع EMOF

۳-۱۲ عناصر ادغام شده از MOF

مدل EMOF بسته‌های زیر را از MOF ادغام می‌کند. برای نمودارهای قابلیت‌های EMOF به بندهای قابلیت‌ها مراجعه شود (۹ تا ۱۱).

- شناسانه‌ها
- بازتاب
- انواع اصلی
- بسط‌ها

۴-۱۲ محدودیت‌های EMOF

این محدودیت‌ها یک بازنمایی صوری در OCL اجرایی دارند که در پیوست B ارجاع داده شده است.

[۱] نوع Operation::raisedException بیشتر به Class محدود است تا Type.

[۲] از لحاظ نمادی حذف پیکان‌ها به گونه‌ای که ارتباطات دوسویه قابل تمایز از ارتباطات بدون جهت نباشند ممکن نیست.

[۳] برای همه NamedElements نیاز به نام وجود دارد به استثنای ValueSpecifications.

[۴] Core::Basic و EMOF از میدان دیدها پشتیبانی نمی‌کنند. همه میدان دیدهای خاصیت‌ها باید هر جا که ممکن است به صورت صریح به عمومی تنظیم شوند یعنی برای همه NamedElements، ElementImports و PackageImports. علاوه بر این امکان استفاده از نام مستعار برای هیچ ElementImport وجود ندارد.

[۵] تعاریف دودویی، صحیح و رشته با این تعاریف پیاده‌سازی سازگار هستند:

o Boolean: http://www.w3.org/TR/xmlschema-2/#boolean o Integer: http://www.w3.org/TR/xmlschema-2/#integer o String: http:// www.w3.org/TR/xmlschema-2/#string [XSD-D]

[۶] تمام معناشناسی‌های مشخص شده در بند ۱۵ «معناشناسی‌های انتزاعی CMOF» برای EMOF اختیاری هستند.

[۷] Property.isID فقط می‌تواند برای یک خاصیت از یک رده درست باشد.

[۸] یک فرامدل EMOF به استفاده از این فرا رده‌های به‌هم‌چسبیده از هسته UML محدود هستند:

- Association
- Class
- Comment
- DataType
- Enumeration
- EnumerationLiteral
- Generalizaion
- InstanceValue
- LiteralNull
- LiteralReal
- LiteralString
- LiteralUnlimitedNatural
- Operation
- Package
- Parameter
- PrimitiveType
- Property

[۹] این خاصیت‌ها باید خالی باشند:

- Association::navagableOwnedEnd
- Class::nestedClassifier
- Classifier:: / عمومی برای نمونه‌های Datatype
- Operation::bodyCondition
- Operation::postCondition
- Operation::precondition
- Operation::redefinedOperation
- Parameter::defaultValue
- Property::qualifier
- Property::redefinedProperty
- Property::subsettedProperty

[۱۰] این خاصیت‌ها باید نادرست باشند:

- Association::isDerived
- Classifier::isFinalSpecialization
- Feature::isStatic
- Property::isDerivedUnion
- RedefinableElement::isLeaf

[۱۱] Generalization::isSubstitutable باید درست باشد.

[۱۲] یک ارتباط دقیقاً دو memberEnd دارد، ممکن است هیچ وقت یک navigableOwnedEnd نداشته باشد(آن‌ها همیشه برای رده هستند) و ممکن است بیشینه یک ownedEnd داشته باشد.

[۱۳] یک عملگر بیشینه می‌توان یک پارامتر داشته باشد که جهت آن «return;» باشد و یک عملگر نمی‌تواند برای هر محدودیت هیچ ParameterSet داشته باشد[8].

[۱۴] Comments فقط نمونه‌های NamedElement را حاشیه‌نویسی می‌کنند.

[۱۵] فقط یک صفت عضو یک رده دارای isId=true است.

[۱۶] Property::aggregation باید یا none یا composite باشد.

[۱۷] Enumerationها ممکن است خاصیت یا عملگر نداشته باشند.

[۱۸] BehavioralFeature باید دنباله‌ای باشد.

[۱۹] کلاس نباید فعال باشد.

[۲۰] یک EnumerationLiteral نباید یک ValueSpecification داشته باشد.

[۲۱] یک پارامتر عملگر نباید هیچ اثر، استثنا یا مشخصه‌های جریانی نداشته باشد.

[۲۲] نمی‌توان یک TypedElement را با یک ارتباط نوع داد.

[۲۳] یک TypedElement باید نوعی متفاوت از یک LiteralSpecification یا یک OpaqueExpression داشته باشد.

[۲۴] یک TypedElement که نوعی پارامتر یا خاصیت از نوع رده است نمی‌تواند یک مقدار پیش‌فرض داشته باشد.

[۲۵] برای یک TypedElement که نوعی پارامتر یا خاصیت نوع Enumeration است مقدار پیش‌فرض(اگر وجود داشته باشد) باید از نوع InstanceValue باشد.

[۲۶] برای یک TypedElement که نوعی پارامتر یا خاصیت نوع اولیه است مقدار پیش‌فرض(اگر وجود داشته باشد) باید از نوع LiteralSpecification باشد.

[۲۷] یک خاصیت مرکب با تعدد الزامی که زیرمجموعه دارد نمی‌تواند زیرمجموعه یک خاصیت مرکب دیگر با تعدد الزامی باشد.

[۲۸] در یک خاصیت از نوع یک DataType باید aggregation = none باشد.

[۲۹] خاصیت که مالک آن یک DataType است فقط می‌تواند از نوع یک DataType باشد.

[۳۰] خاصیت انتهاعضو هر رابطه باید از نوع یک رده باشد.

[۳۱] یک خاصیت یا پارامتر چندمقداری نمی‌تواند یک مقدار پیش‌فرض داشته باشد.

[۳۲] مقادیر lowerValue و upperValue از MultiplicityElement باید به ترتیب یک نوع LiteralInterger و LiteralUnlimitedNatural باشند.

۵-۱۲ تعاریف EMOF و دستورالعمل‌های استفاده برای مدل‌های UML

وقتی بسته EMOF برای مدیریت فراداده استفاده می‌شود این قوانین استفاده اعمال می‌شوند.

بسته

- اگرچه EMOF بسته و بسته‌های تودرتو تعریف می‌کنید همیشه با ارجاع مستقیم شیء به عناصر مدل ارجاع می‌دهد. EMOF هیچ گاه از نام‌های عناصر استفاده نمی‌کند. هیچ عملگری برای دسترسی به هیچ‌چیز توسط NamedElement::name وجود ندارد. نمونه‌های مدل‌های EMOF در صورت نیاز معناشناسی‌های فضای نام بیشتری برای بسته‌های تودرتو فراهم می‌کنند.

خاصیت‌ها

- همه خاصیت‌ها به صورت تجزیه‌ناپذیر ویرایش می‌شوند.
- وقتی یک مقدار به‌روزرسانی می‌شود دیگر به مقدار قدیمی ارجاع نمی‌شود.

- خاصیت‌های مشتق‌شده هنگامی که به آن‌ها مراجعه شود یا منبع مشتق‌شده آن‌ها آن‌طور که توسط پیاده‌سازی مشخص شده تغییر کند به‌روزرسانی می‌شوند. آن‌ها به‌طور خاص با استفاده از () set نیز می‌توانند به‌روزرسانی شوند اگر قابل به‌روزرسانی باشند.

Type == DataType

- زمانی که یک شیء ایجاد شده یا زمانی که خاصیت بازنشانی شده است مقدار خاصیت پیش‌فرض است.
- خاصیت‌های با حد بالای فراونی بیشتر از ۱ فهرست‌های خالی دارند تا نشانگر این باشد که هیچ مقداری تنظیم نشده است. اگر Property.isUnique==true باشد مقادیر فهرست منحصر به فرد هستند.
- خاصیت‌های «شناسانه» خاصیت‌هایی هستند که property.idID==true است.

Type==Class

- خاصیت‌های با حد بالایی تعدد ۱ مقدار تهی دارند تا نشانگر این باشد که به هیچ شیء اشاره نمی‌شود.
- خاصیت‌های با حد بالای فراونی بیشتر از ۱ فهرست‌های خالی دارند تا نشانگر این باشد که به هیچ شیء اشاره نمی‌شود. تهی مقدار معتبری در فهرست نیست.
- EMOF از نام خاصیت‌ها استفاده نمی‌کند و دسترسی از طریق آرگومان خاصیت رابط‌های بازتابی است. مهم نیست نام خاصیت‌ها چیست، این نام‌ها در EMOF هیچ وقت استفاده نمی‌شوند. داشتن نام‌های مشابه هیچ معنای خاصی ندارد. برای عملیات نیز همینطور است، از نام استفاده نمی‌شود و تداخل نام، باطل‌سازی یا معناشناسی‌ها بازتعریف وجود ندارد. EMOF تابع فراخوانی به عنوان بخشیء از رابط بازتابی ندارد بنابراین هیچ معناشناسی برای فراخوانی یک عملگر EMOF وجود ندارد. نام‌ها و انواع پارامترها هیچ وقت مقایسه نمی‌شوند و محدودیتی در مورد آنچه آن‌ها می‌توانند به تنهایی با به صورت ترکیبی داشته باشند وجود ندارد. سایر نمونه‌های فرامدل‌های EMOF یا نگاشت‌های زبانی مانند JMI2 ممکن است معناشناسی‌های اضافی داشته باشند یا محدودیت‌هایی در عمل وجود داشته باشد که نیاز به تعاریف خاص‌تر از معنای ارث‌بری داشته باشند.

Property::isComposite==true

- یک شیء فقط می‌تواند یک دربردارنده داشته باشد.
- خاصیت‌های دربردارنده همیشه حد بالایی ۱ دارند.
- فقط یک خاصیت دربردارنده ممکن است غیرتهی باشد.
- بازداری چرخه‌ای غیرمعتبر است.
- اگر یک شیء یک دربردارنده داشته باشد و یک دربردارنده جدید قرار است تنظیم شود شیء از دربردارنده قدیمی حذف شده و سپس دربردارنده جدید تنظیم می‌شود.
- اضافه کردن یک دربردانده خاصیت‌های دربردارنده و بازداری را به ترتیب روی شیء محیط و محاط به‌روزرسانی می‌کند.
- مقدار جدید به این خاصیت اضافه می‌شود.

Property::isComposite==false دوسویه،

- ابتدا شیء از انتهای طرف مقابل خاصیت حذف می‌شود.
- اگر خاصیت طرف مقابل مقدار جدید حد بالایی ۱ داشته باشد مقدار قدیمی حذف می‌شود.
- این شیء به خاصیت طرف مقابل مقدار جدید اضافه می‌شود.
- مقدار جدید به این خاصیت اضافه می‌شود.

شیء

- هر چیزی که MOF به آن‌ها دسترسی دارد یک شیء است.
- یک شئ که یک عنصر نباشد ممکن است نمونه‌ای از یک DataType باشد.

۶-۱۲ برچسب‌های از قبل تعریف شده

این زیربند یک برچسب از قبل تعریف شده را توضیح می‌دهد که نام آن « org.omg.emof.oppositeRoleName» است که می‌توان آن را به نمونه‌های خاصیت درون نمونه‌های مدل EMOF اعمال کرد.

محدودیت‌ها

Context Tag inv:

برچسب از پیش تعریف شده فقط می‌تواند به نمونه‌هایی از خاصیت اعمال شود که خاصیت «مقابل» آن‌ها خالی است

org.omg.emof.oppositeRoleName بدون نام تلویحاً به این معنی است که

element.oclIsKindOf (Property) و element.oclAsType (Property).opposite->isEmpty ().

معناشناسی‌ها

اگر یک نمونه از یک برچسب نام org.omg.emof.oppositeRoleName داشته باشد آنگاه مقدار آن یک نام نقش مشخص می‌کند که عبارت‌ها می‌توانند از آن استفاده کنند تا در جهت مقابل خاصیت حرکت کنند مانند عبارات OCL و QVT.

اگر عبارتی از یک نام نقش مشخص شده استفاده کند که از برچسب با نام org.omg.emof.oppositeRoleName استفاده می‌کند و بیشتر از یک خاصیت برچسبی با این نام نقش دارند آنگاه به زبان بیان بستگی دارد که تعیین کند این یک خطا است یا حرکت معکوس در همه این خاصیت‌ها را نشان می‌دهد. زبان بیان نباید در صورت ابهام خاصیت را به صورت تصادفی انتخاب کند.

پایه منطقی

استفاده از این برچسب نسبت به استفاده از خاصیت «مقابل» خاصیت سبک‌تر است. استفاده از خاصیت «مقابل» در تمام مواردی که فقط توانایی عبارت‌ها برای حرکت در مسیر مقابل مورد نیاز است این پیامدهای منفی را دارد:

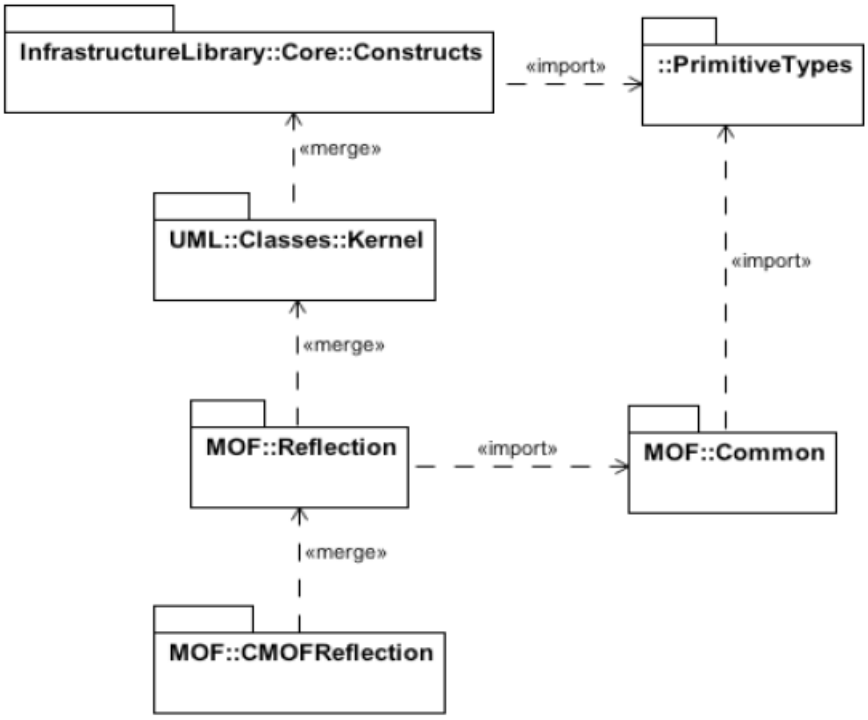
- این منجر به همبستگی قوی‌تر میان رده‌ها می‌شود.
- این به سه شیوه به بار زمان اجرایی که نمونه‌های مدل روی زیرساختی که آن‌ها را مدیریت می‌کند می‌گذارند اضافه می‌کند:۱) افزایش ردپای کلی به دلیل اینکه خاصیت مقابل به قرارداد رده‌ای که مالک خاصیت اضافی معرفی شده به عنوان مقابل خاصیت اصلی است بسیار اضافه می‌کند ۲) نیاز به تخصیص فضا برای نمونه‌های خاصیت اضافی و ۳) نیاز به نگهداری ثبت ارجاعی در میان نمونه‌های خاصیت اصلی و نمونه‌های خاصیت اضافی.

مشخص‌سازی نحو به‌هم‌چسبیده که از عبارت‌ها برای پیمایش از طریق org.omg.emof.oppositeRoleName در زبان‌هایی مانند OCL و QVT استفاده می‌کنند، خارج از محدوده هسته MOF است.

۱۳ بازتاب CMOF

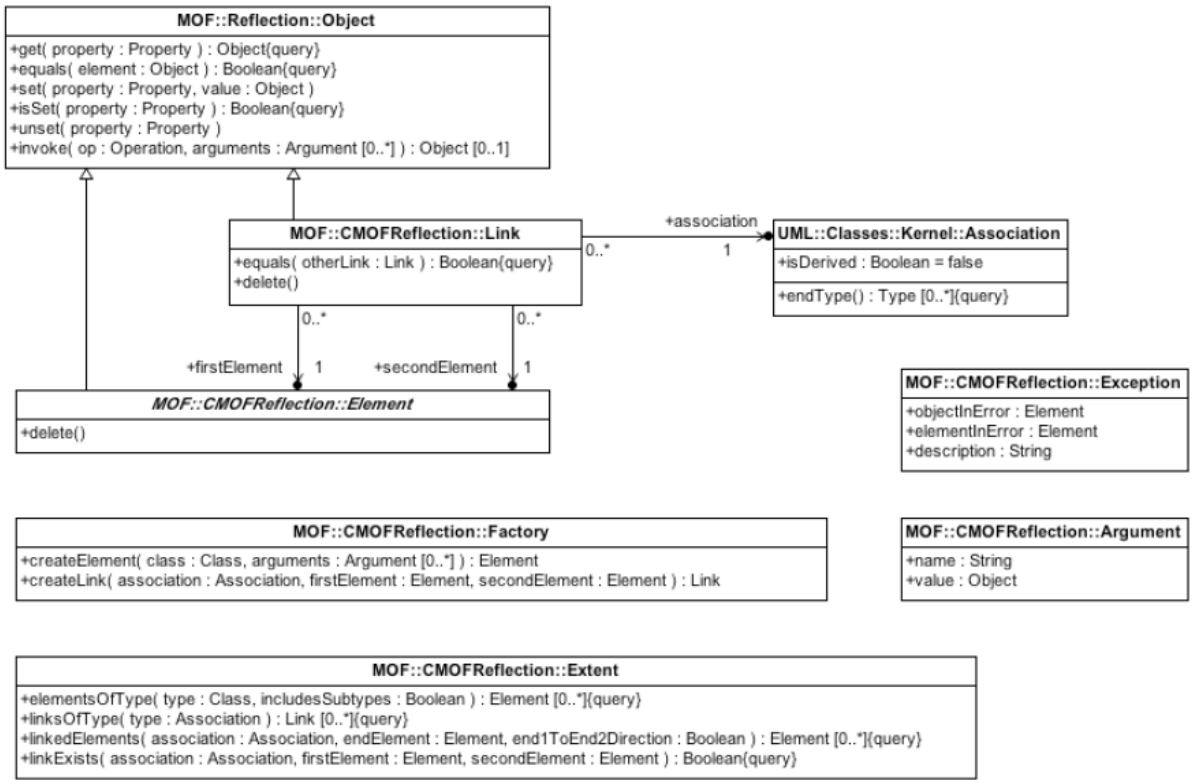
۱-۱۳ کلیات

CMOF::Reflection قابلیت‌های توسعه‌داده شده را بر روی بسته MOF::Reflection ارائه می‌کند. نمودار بسته در شکل ۱۳-۱، چگونگی توسعه داده شدن MOF::Reflection توسط CMOF:Reflection را نشان می‌دهد.



شکل ۱۱- بازتاب CMOF

CMOF:Reflection عملیات افزوده‌ای را در شیء، حد و رده‌های کارخانه موجود ادغام می‌کند و رده (کلاس) پیوند و datatype آرگومان را می‌افزاید. این افزوده‌ها توسط CMOF::Reflcetion در شکل ۱۳-۲ نشان داده شده است.



شکل ۱۲- بسته بازتاب CMOF

۱۳-۲ پیوند

این یک رده (کلاس) جدیدی است که نمونه‌ای از رابطه را به روشی بازنمایی کرده که عنصر نمونه‌ای از یک رده (کلاس) را بازنمایی می‌کند.

خاصیت‌ها

association: Association

این رابطه‌ای است که در آن پیوند یک نمونه است.

firstElement: Element

این عنصری است که با اولین انتهای رابطه، ارتباط دارد.

secondElement: Element

این عنصری است که با دومین انتهای رابطه، ارتباط دارد

عملیات

equals(otherLink:Link): Boolean

True بازمی‌گرداند اگر otherlink دارای رابطه، firstElement، secondElement همگی برابر با همان‌ها در این پیوند داشته باشد.

delete()

این پیوند را حذف می‌کند. این امر ممکن است عناصر مشابه مرتبط شده با پیوندهای دیگر برای این رابطه را نگه دارد.

محدودیت‌ها

firstElement باید مطابق با نوع اولین memeberEnd از رابطه انجمنی باشد.

SecondElement باید مطابق با نوع دومین memeberEnd از رابطه انجمنی باشد.

مجموعه پیوندها به طور کل نباید محدودیت‌های تعدد ارتباط عضو انتهایی رابطه انجمنی را بشکنند.

معناشناسی‌ها

وقتی یک پیوند ایجاد می‌شود، در هیچ بسطی درج نمی‌شود.

پایه منطقی

تا زمانی که MOF 2 امکان پیوند عناصر مشابه را بیش از یکبار در همان رابطه می‌دهد

(if isUnique=false for the association ends)

پیوند نیاز به مقدار تاپل[1] دارد گرچه به عنوان heavyweight اولین رده (کلاس) نیست.

۳-۱۳ آرگومان

این datatype جدیدی است که برای بازنمایی آرگومان‌های نام‌گذاری‌شده برای عملیات بازتابی باز-بی‌انتها به کار می‌رود. آرگومان باز-بی‌انتها است و امکان تأمین عناصر و مقادیر داده‌ها را می‌دهد.

خاصیت‌ها

Name: string – نام آرگومان

Value: Object - مقدار آرگومان

۱- یک سطر از یک رابطه را یک تاپل (tuple) می نامند. هر تاپل در جدول نمایانگر یک نمونه از یک هستار است که رکورد هم گفته می شود. تاپل ها ممکن است روی یکی از صفات خاصه به طور مرتب ذخیره شوند. ولی به طور کلی لازم نیست مرتب باشند.

محدودیت‌ها

آرگومان نوع داده‌ای است که از عملیات بازتابی باز پشتیبانی می‌کند. نوع داده دارای هیچ معناشناسی از خود نوع داده نیست. محدودیت‌ها به محتوای جایی که آرگومان آن را تأمین کرده است، وابسته است.

معناشناسی‌ها

هیچ

پایه منطقی

از آنجا که MOF 2 امکان می‌دهد پارامترها و خاصیت‌ها عملیات دارای پیش‌فرض باشد، نیاز به شناسایی صحیحی از مقادیر تأمین شده است.

۴-۱۳ شیء

بازتاب CMOF عملیات افزوده‌ای را اضافه می‌کند.

invoke(op:Operation, arguments : Argument[0..*]) : Object[0..*]

عملیات ایجاد شده را بر روی شیء را با ارسال آرگومان‌های ایجاد شده و بازگشت نتیجه، فراخوانی می‌کند. بیش از یک مقدار برای نتیجه را تولید می‌کند، نتیجه عملگر فراخوانی‌شده، نوعی از ReflectiveCollection که دربردارنده مقادیر نتیجه تولیده شده است.

عملیات باید بر روی رده (کلاس) شیء تعریف شود و آرگومان‌ها باید به پارامترهای عملگر ارجاع داده شود. اگر آرگومانی برای پارامتر ایجاد نشده باشد، مقدار آن در صورت وجود همان پیش‌فرض است.

پایه منطقی

معادل قابلیت‌های MOF 1.4 را اضافه می‌کند.

۵-۱۳ عنصر

بازتاب CMOF عملیات افزوده زیر را اضافه می‌کند.

عملیات

delete()

عنصر را حذف می‌کند.

isInstanceOfType(type: Class, includeSubtypes: Boolean): Boolean

True بازمی‌گرداند اگر این عنصر نمونه‌ای از رده (کلاس) مشخص شده باشد یا اگر includeSubtypes، True باشد، هر زیررده (زیرکلاسی) از آن هم True باشد.

پایه منطقی

معادل قابلیت‌های MOF 1.4 را اضافه می‌کند.

۱۳-۶ کارخانه

بازتاب CMOF دو عملگر افزوده را اضافه می‌کند.

عملیات

createElement(class:Class, arguments : Argument[0..*]) : Element

خلاف عملگر ساده ()create ، کارخانه به آرگومان‌ها امکان ارائه برای استفاده به عنوان مقادیر اولیه خاصیت‌ها می‌دهد.

آرگومان‌ها باید به خاصیت‌ها Datatype رده (کلاس) ارجاع دهد. اگر آرگومانی برای یک خاصیت به صورت مقدار پیش‌فرض ایجاد نشود، در صورت وجود به کار خواهد رفت.

createLink(association : Association, firstElement : Object, secondElement : Object) : Link

پیوندی از عناصر ایجاد شده ایجاد می‌شود که نمونه‌ای از ارتباط ایجاد شده است. Firstelement مرتبط با پایان اولین (خاصیت‌ها شامل پایان‌های رابطه که مرتب شده‌اند) است و باید مطابق با نوع آن باشد و مربوط به secondElement باشد.

پایه منطقی

معادل قابلیت‌های MOF 1.4 را اضافه می‌کند.

۱۳-۷ بسط

بازتاب CMOF چهار عملگر افزوده را اضافه می‌کند.

عملیات

elementsOfType(type : Class, includeSubtypes : Boolean) : Element[0..*]

آن عناصری را که نمونه‌هایی از رده (کلاس) ایجاد شده در بسط است را بازمی‌گرداند. اگر includeSubtypes، True باشد، نمونه‌های از زیررده (زیرکلاس) آن‌ها همچنین بازمی‌گردد.

linksOfType(type : Association) : Link[0..*]

آن پیوندهایی را که نمونه‌هایی از رابطه ایجاد شده است، بازمی‌گرداند.

linkedElements(association : Association, endElement : Element, end1ToEnd2Direction : Boolean) : Element[0..*]

رابطه ایجاد شده را از عنصر ایجاد شده نشان می‌دهد. جهت نمایش با پارامتر emd1ToEnd2Direction معین می‌شود. اگر True، باشد، عنصر ایجاد شده به عنوان پایان اولین رابطه عمل می‌شود.

linkExists(association : Association, firstElement : Element, secondElement : Element): Boolean

True بازمی‌گرداند اگر دست‌کم یک پیوند برای ارتباط بین عناصر ایجاد شده در انتهای موردنظرشان وجود داشته باشد.

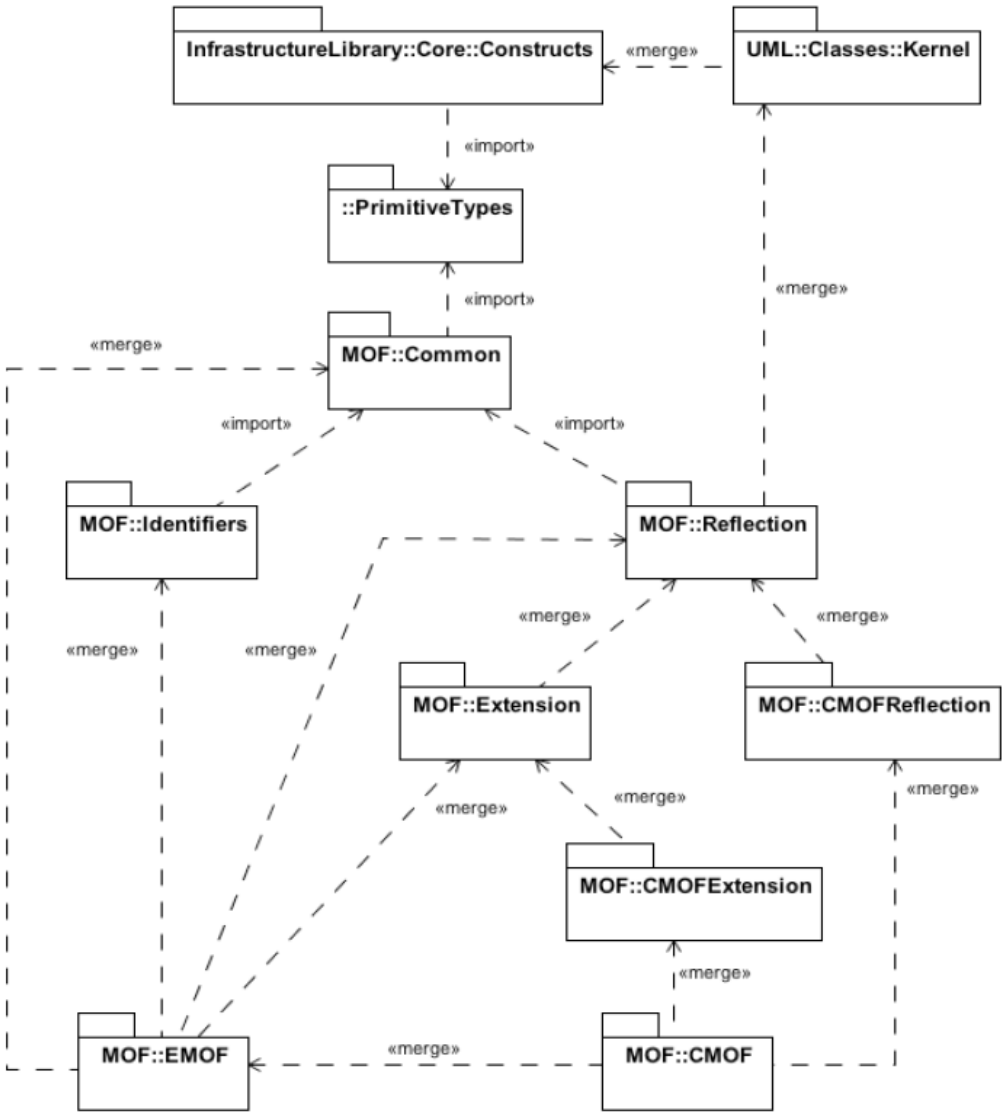
پایه منطقی

معادل قابلیت‌های MOF 1.4 را اضافه می‌کند.

۱۴ مدل کامل MOF (CMOF)

۱-۱۴ کلیات

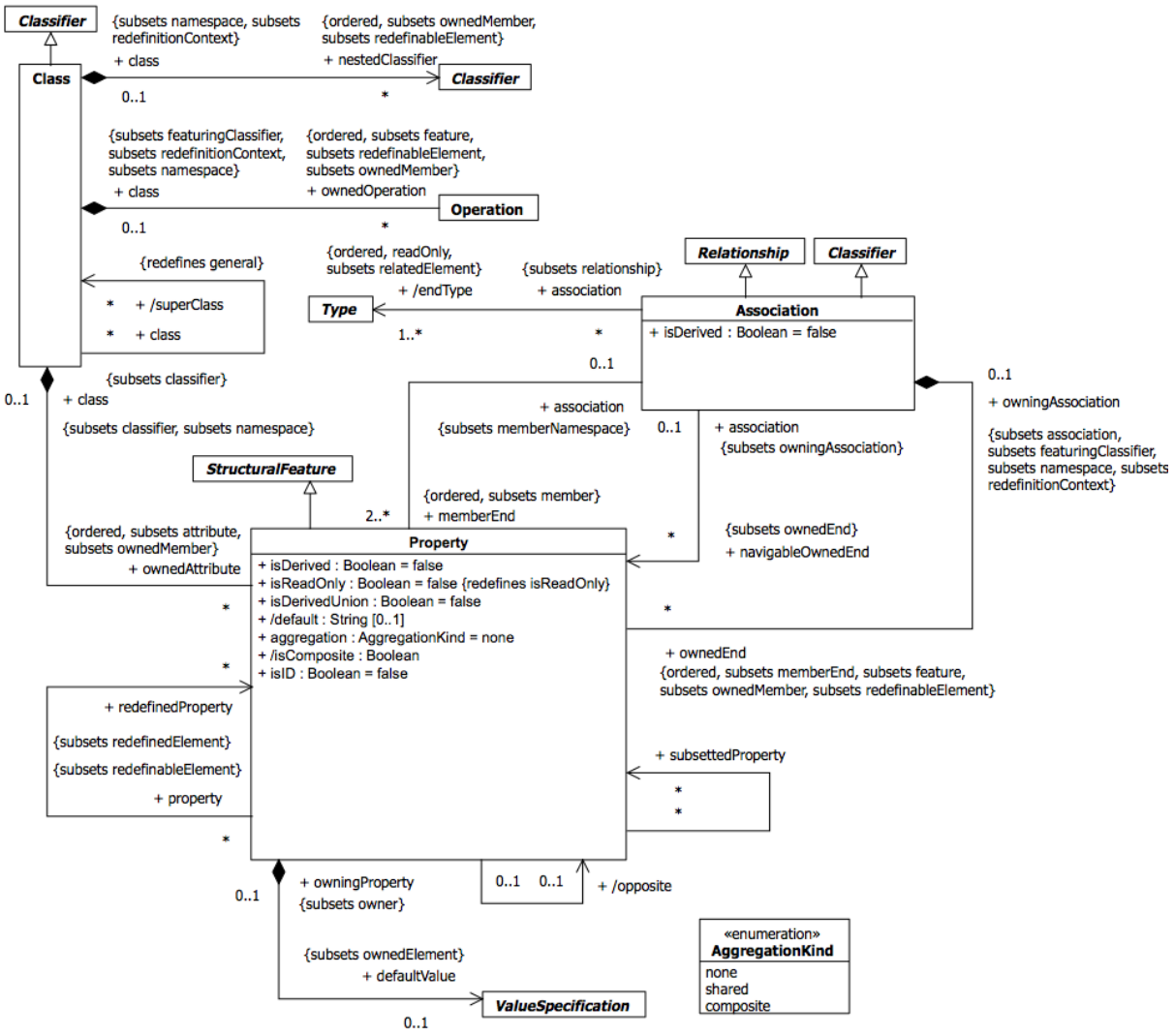
مدل CMOF فرامدلی است که برای مشخص ساختن فرامدل‌های دیگر از قبیل UML 2 به کار می‌رود. این مدل ازEMOF و core::Construct از UML 2 ساخته شده است. بسته مدل هیچ رده‌ای (کلاسی) از خودش تعریف نمی‌کند. به جای آن، بسته‌ها را بسطهایشان که قابلیت‌های فرامدل‌سازی مبنا را تعریف می‌کنند باهم ادغام می‌کند. ساختار بسته کامل شامل CMOF، در شکل ۱۴-۱ نشان داده شده است.



شکل ۱۳- بسته CMOF

۲-۱۴ عناصر به‌کاررفته از UML 2

شکل ۲-۱۴ برخی از کلاس‌ها (رده‌ها) و ارتباطات به‌هم‌چسبیده اصلی UML 2 را نشان می‌دهد که به منظور مشخص ساختن فرامدل‌های CMOF به کار می‌رود. عناصر مهم دیگری وجود دارد که از UML به کار می‌رود ولی این عناصر ساختار مدل‌سازی رده (کلاس) را ارائه می‌دهد. برای جزییات بیشتر به مشخصات UML مراجعه شود.



شکل ۱۴ – رده‌های (کلاس‌های) به‌هم‌چسبیده اصلی از بسته هسته UML 2

۳-۱۴ عناصر درون‌برد شده از MOF

مدلCMOF بسته‌های EMOF را از MOF که شامل بسته‌های قابلیت‌های MOF است را ادغام می‌کند.

- شناسانه‌ها
- بازتاب
- بسط

۴-۱۴ محدودیت‌های CMOF

این زیربند محدودیت‌های متعلق به بسته CMOF را به تفصیل بیان می‌کند که در فرامدل‌ها با پیاده‌سازیCMOF پردازش می‌شود. این محدودیت‌ها جانشین محدودیت‌های EMOF از زیربند ۱۲-۴

می‌شود؛ یعنی توصیه می‌شود اعتبارسنجی فرامدل نسبت به تمام محدودیت‌های تعریف شده در این بند با صرف‌نظر کردن از تمام تعاریف محدودیت‌ها از زیربند ۱۲-۴ انجام شود.

فرامدل CMOF، بسته‌های دیگر MOF و خود UML منطبق با این‌ها است.

این محدودیت‌ها دارای بازنمایی صوری در OCL اجرایی است همان‌طور که در پیوست ب بدان ارجاع شده است.

[۱] تعدد Association::memberEnd به جای *..2 محدود به 2 است.(یعنی، از ارتباطات n-ary پشتیبانی نمی‌شود) ؛ خلاف EMOF، ارتباطات CMOF می‌تواند دارای پایان‌های ارتباطی جهت‌دار باشد.

[۲] نوع Operation::raisedException به جای نوع محدود به رده (کلاس) است.

[۳] به منظور پشتیبانی از پیاده‌سازی‌های محدود شده از رده (کلاس) اعداد صحیح، هر نمونه از اعداد صحیح به صورت مقدار یک صفت از یک عنصر در بسط اعداد صحیحی رخ دهد که می‌توانند با استفاده از قالب کامل دو ۳۲ بیت بازنمایی شوند، به بیان دیگر هر عدد صحیح در بسط -2_{31} تا1- -2_{31}

[۴] به منظور پشتیبانی از پیاده‌سازی‌های محدود شده رده (کلاس) رشته، هر نمونه از رشته به عنوان مقدار صفت از یک عنصر رخ می‌دهد که طول آن از ۶۵۵۳۵ کاراکتر بیشتر نخواهد شد.

[۵] از لحاظ نمادی حذف پیکان‌ها به گونه‌ای که ارتباطات دوسویه قابل تمایز از ارتباطات بدون جهت نباشند ممکن نیست.

[۶] برای همه NamedElements نیاز به نام وجود دارد به استثنای ValueSpecifications.

[۷] CMOF از میدان دیدها پشتیبانی نمی‌کنند. همه میدان دیدهای خاصیت‌ها باید هر جا که ممکن است به صورت صریح به عمومی تنظیم شوند یعنی برای همه NamedElements، ElementImports و PackageImports. علاوه بر این امکان استفاده از نام مستعار برای هیچ ElementImport وجود ندارد.

[۸] Enumeration دارای صفت یا خاصیت نیست.

[۹] Property.isID می‌تواند تنها برای یک خاصیت از یک رده (کلاس) درست باشد.

[۱۰] فرامدل CMOFمحدود به استفاده از فراردهای (فراکلاس‌های) به‌هم‌چسبیده زیراز هسته UML است:

- Association
- Class
- Comment
- DataType
- ElementImport
- Enumeration
- EnumerationLiteral
- Genralization
- InstanceValue
- LiteralBoolean

- LiteralInteger
- LiteralNull
- LiteralReal
- LiteralString
- LiteralUnlimitedNatural
- OpaqueEcpression
- Operation
- Package
- PackageImport
- PackageMerge
- Parameter
- PrimitveType
- Property

[۱۱] این خاصیت‌ها باید خالی باشند:

- Class::nestedClassifer
- Property::qualifier

[۱۲] مقدار Feature::isStatic باید غلط باشد.

[۱۳] یک خاصیت یا پارامتر چندمقداری نمی‌تواند یک مقدار پیش‌فرض داشته باشد. مقدار پیش‌فرض از خاصیت با نوع پارامتر با PrimitiveType باید نوعی از LiteralSpecification باشد. مقدار پیش‌فرض از خاصیت یا نوع پارامتر با Enumeration باید نوعی از InstanceValue باشد. یک خاصیت یا نوع پارامتر با یک رده (کلاس) نمی‌توانند دارای یک مقدار پیش‌فرض باشند.

[۱۴] مقادیر MultiplicityElement::lowerValue و uperValue باید به ترتیب نوعی از LiteralInteger و LiteralUnlimited باشد

[۱۵] Generalization::isSubstitutable باید درست باشد.

[۱۶] تنها یک عضو صفت از رده (کلاس) می‌تواند دارای IsId=true باشد. عضوهای دیگر(به طور مثال، آن-هایی که به ارث‌برده‌اند) باید مجدد تعریف شوند: یا غیرقابل‌دسترس شوند یا مجدد تعریف شوند تا IsId=false شود.

[۱۷] Property::aggregation باید یا ’none‘ شود یا ’compostie‘.

[۱۸] BehavioralFeature باید ترتیبی شود.

[۱۹] رده (کلاس) نباید active باشد.

[۲۰] EnumerationLiteral نباید دارای ValueSpecification باشد.

[۲۱] پارامتر عملیات نباید دارای هیچ‌گونه خصیصه‌های تأثیری، استثناء،یا جریانی باشد.

[۲۲] TypedElement نمی‌تواند توسط رابطه نوع‌گذاری شود.

[۲۳] TypedElement در عوض LiteralSpecification یا یک OpaqueExpressiom باید دارای نوع باشد.

[۲۴] TypedElement نوعی پارامتر یا نوعی خاصیت با class است، که نمی‌تواند دارای پیش‌فرض باشد.

[۲۵] برای TypedElement گه نوعی پارامتر یا نوعی خاصیت با Enumeration است، در صورت وجود، مقدار پیش‌فرض باید نوعی InstanceValue باشد.

[۲۶] برای TypedElement گه نوعی پارامتر یا نوعی خاصیت با PrimitiveType است، در صورت وجود، مقدار پیش‌فرض باید نوعی InstanceValue باشد.

[۲۷] یک خاصیت مرکب با تعدد الزامی که زیرمجموعه دارد نمی‌تواند زیرمجموعه یک خاصیت مرکب دیگر با تعدد الزامی باشد.

[۲۸] نوع خاصیت با نوع DataType باید دارای aggregation=none باشد.

[۲۹] مالکیت خاصیت با DatType می‌تواند تنها نوعی توسط DataType باشد.

[۳۰] هر خاصیت عضو انتهایی ارتباط باید نوعی توسط یک رده (کلاس) باشد.

[۳۱] محدودیت باید دست‌کم یک عنصر را تحمیل کند و باید از طریق OpaqueExpression مشخص شود.

[۳۲] بدنه OpaqueExpression باید تهی نباشد.

۱۴-۵ بسط‌ها برای قابلیت‌های CMOF

این زیربند بسط‌هایی را برای قابلیت‌های MOF2 به تفصیل توضیح می‌دهد.

۱۴-۵-۱ بازتاب

۱- CMOF کارخانه را به منظور مشخص ساختن قالب آرگومان رشته از Factory::createFromString و نتیجه Factory::convertToString همان‌طور که در «نگاشت تسهیل فراشیء 2.0 XMI(MOF)» تعریف شده، بسط می‌دهد تا از مقادیر پیش‌فرض برای انواع داده ساخت‌یافته پشتیبانی کند.

۱۴-۵-۲ بسط

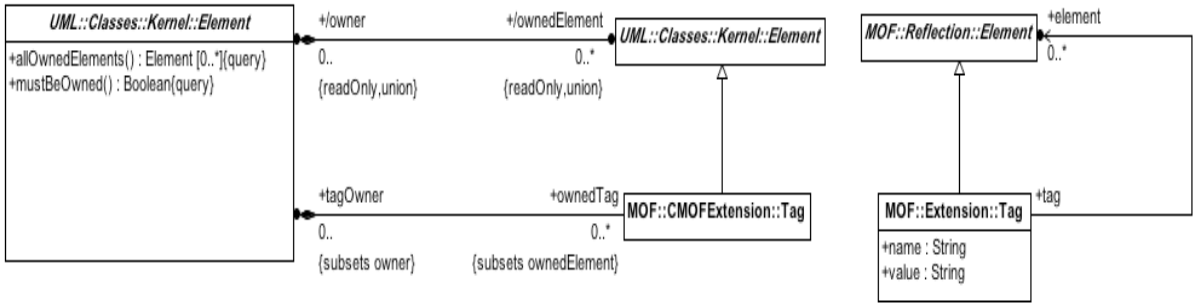
۲- CMOF بسط بسته را با یک ارتباط بین عنصر با نقش tagOwner و برچسب با ownedTag پایان‌های ارتباطی جهت‌دار بسط می‌دهد.

Associations:

tagOwner : Element[0..1] {subsets Element::owner}

یادآوری- اگرچه پایان یک ownedTag متعلق به ارتباط است، باید جهت‌دار باشد تا از بازیابی برچسب‌هایی که شیء دارد با استفاده از UML ReadlinkAction پشتیبانی کند. (به زیربند ۱۱-۳-۳۳ UML2.4 مراجعه شود). به اضافه، OCL قابلیت

بازیابی برچسب‌هایی را که یک عنصر e، دارد را با عبارت e.owmedTag فراهم می‌کند. (به زیر بند ۷-۵-۳ OCL2.2مراجعه شود.) بدین معنی که ثابت‌های زیر برگرفته از شکل ۱۵ است.



شکل ۱۵ – بسته بسط CMOF

۱۵ معناشناسی‌های انتزاعی CMOF

۱-۱۵ کلیات

این بند، معناشناسی‌های CMOF را با توصیف قابلیت‌های کارکردی سامانه مدل‌سازی‌شده و چگونگی ارتباط آنها با عناصر این مدل توصیف می‌کند. این قابلیت‌ها هرگونه نگاشت به فناوری پیاده‌سازی مستقل است که توصیف‌های آنها انتزاعی است.

به یاد داشته باشید که تمام این قابلیت‌ها با انواع، تعدد، محدودیت‌ها، قابلیت مشاهده، قابلیت تنظیم، غیره محدود شده که توسط این مدل به آن تحمیل می‌شود و احتمالاً در بعد توسط ملاحظات دیگر از قبیل امنیت کنترل دسترسی، ثبات تاریخچه نسخه‌های داده‌ها و غیره محدود می‌شود. بنابراین استفاده از قابلیت‌ها می‌تواند در وضعیت‌های خاص رد شود. نتایج خرابی در یک مورد استثناء به دست می‌آید (به بسته استثناها مراجعه شود).

۲-۱۵ رویکرد

MOF چارچوب مستقل از بستر مدیریت فراداده‌ها است که شامل ایجاد، دستکاری، یافتن، تغییر، از بین بردن اشیا و ارتباطات بین آن اشیا همان‌طور که توسط فرامدل‌ها از قبل تعیین شده می‌شود. این زیربند قابلیت‌های هسته که MOF ایجاد می‌کند و معناشناسی‌ها و رفتارهای آن اقدامات را توصیف می‌کند . این قابلیت‌ها ممکن است در مشخصات بعدی در مجموعه MOF2 بسط داده شده یا تصحیح شود. این زیربند اجبار نمی‌کند که تمام پیاده‌سازی‌های MOF نیاز به پشتیبانی از تمام این قابلیت‌ها باشد: برای تعریف بیشتر، معنای قابلیت‌ها در هنگامی‌که ایجاد می‌شود، چیست. نقاط انطباق به صورت مجزا توصیف می‌شود. برای مثال، این زیربند معناشناسی‌های :Reflection تعریف می‌کند. این محدودیت بر روی آن دست پیاده‌سازی‌هایی است که بازتاب را ایجاد می‌کند ولی نیاز به ایجاد در تمامی پیاده‌سازی‌ها نیست.

برای تعریف خوب، رفتار خوب و قابل درک این قابلیت‌ها، برخی از این قابلیت‌ها با توجه به مفهوم کمرنگی از منطق «بسط» از عناصر مدل که برخی مفاهیم انتزاعی مکانی و محتوا را ایجاد می‌کند توصیف می‌شود. این امر به طور گسترده‌تری به عنوان قسمتی از ابزار MOF 2 و چرخه حیات شیء تعریف خواهد شد.

هدف کلی تعریف این قابلیت‌ها، ایجاد تعریفی بن‌سازه منفرد است که می‌اتوان آن را به عنوان مبنای انقیاد زبانی به منظور دست‌یابی به برخی سطوح سازگاری و همکاری متقابل به کار برد. همچنین تصمیمات فرامدل‌سازی بر اساس معناشناسی‌ها و محدودیت‌ها را میسر می‌کند که باید تعریف شوند.: افزایش سطح تعریف معناشناسی.

این زیربند MOF را از مشخصات سامانه فرافرامدل به مشخصات سامانه مدل‌سازی شده تبدیل می‌کند. (مدل مستقل از بن‌سازه). این امر نیاز به معرفی تفصیلی و محدودیت‌های بیشتری نسبت به آنچه اکنون در زیرساخت UML2 دارد که مبتنی بر مبنای آن است. به طور خاص برای رسیدن به این امر لازم بوده تا مدل نمونه‌های UML2 بسط داده شود.

اگرچه این رویکرد برحسب مدل نمونه‌ها توصیف شده (به طور مثال، شکاف‌ها)، مهم است تا تاکید شود این رویکرد مدل مشخصات است و رویکرد پیاده‌سازی را تعیین نمی‌کند. رده‌های نمونه‌ها از قبیل شکاف‌ها در عملیات مشخص شده ظاهر نمی‌شوند: توصیه می‌شود اگر اینها با استفاده از شکاف‌ها پیاده‌سازی شدند، به پیاده‌سازی‌ها عمل شود ولی به طور کل با استفاده از سازوکارهای به مراتب کارا تری پیاده‌سازی خواهند شد.

مشخصات برحسب واسط بازتابی است ولی مقصود آن این است تا واسط‌های مشخص تولیدشده‌ای برای فرامدل‌های مشخص شده باشد.

۳-۱۵ مدل نمونه‌های MOF

اصول

این دامنه معناشناسی‌ها فقط نمودار رده‌ها را از ساخت‌ها پوشش می‌دهد.

به طور کلی خاستگاه این رویکرد، ممانعت از افزونگی و این است که تنها درجایی که نیاز است شکاف‌ها وجود داشته باشند (به طور مثال، نه برای صفات مشتق شده).خواستگاه این رویکرد ساده‌سازی مشخصات رفتار به‌روزرسانی (که دراینجا کوششی برای آن نمی‌شود) و به طور بالقوه مشخصات ترتیب XMI است.

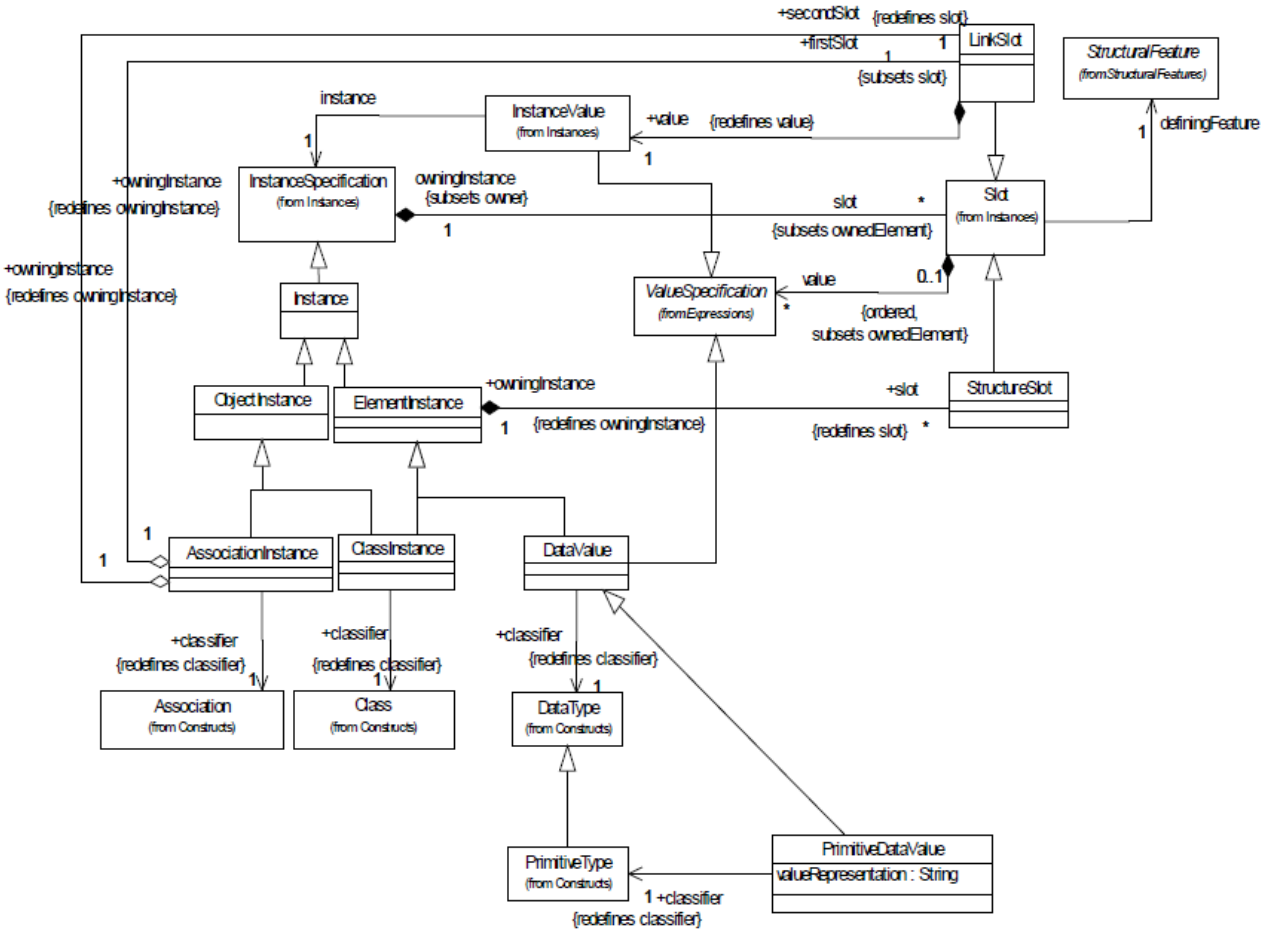
در اینجا استثناء، نمونه‌های ارتباطی است که هم به عنوان AssociationInstances و هم از طریق شکاف‌ها بر روی اشیاء پیوند داده شده نمایش داده می‌شوند (تنها برای پایان ارتباط جهت‌دار). در تئوری مقادیر پایان جهت‌دار می‌تواند از طریق پرس‌وجو بر روی AssociationInstances به دست آید ولی این امر برای موارد زیر نیست:

- برای ساده‌سازی توضیح
- برای حفظ این باور که این خاصیت‌ها، صفات واقعی هستند

- برای ایجاد سازگاری بیشتر با مبنا

DataValues هم به عنوان Instances (تا زمانی که دارای شکاف هستند) و هم به عنوان ValueSpecification عمل می‌کنند.: تا زمانی که همواره مطرح شود datavalues به طور مستقیم در شکاف ذخیره شده تا این که به آن ارجاع داده شوند (که نیاز به اندکی مرتب‌سازی هویت‌ها است)

شکل ۱۶ مدل دامنه معناشناسی را برای ساخت‌ها نشان می‌دهد. این شکل نحو انتزاعی برای نمونه‌ها است.



شکل ۱۶- مدل دامنه معناشناسی برای ساخت‌ها

در زیر محدودیت‌هایی بر رده‌های معرفی شده نشان داده می‌شود.

ObjectInstance (applies to both ClassInstances and AssociationInstances)

۱-دقیقاً یک شکاف برای هر StructuralFeature ذخیره شده وجود دارد و نه بیشتر. storedStructuralFeature شامل شکاف‌های ارثبری شده هستند ولی موارد زیر را مستثنی می‌شوند:

- خاصیت‌های مشتق شده (از جمله الحاق‌های مشتق شده)
- خاصیت‌هایی که مجدد تعریف شده باشند (انجا خاصیتی وجود دارد که عضو جداکننده است که redefinedProperty برابر با این خاصیت است).

۲-جداکننده انتزاعی نیست

۳-نمونه از محدودیت‌ها پیروی می‌کند که طبقه‌بندی کننده آن محتوا است.

ClassInstance

۱-یک نمونه از طریق بیشینه یک ترکیب تعلق می‌یابد.

۲-بیشینه یک شکاف تعلق‌یافته وجود دارد که شکاف مربوط به خاصیتی که خلاف خاصـیت isComposite است ممکن است دارای یک مقدار باشد.

۳-ترکیب‌ها دوره‌ای نیستند.

StructureSlot

۱-تعداد مقادیر مربوط با تعدد (بالا و پایین) از definingFeature آن

۲-اگر ویژگی isUnique باشد، تعدا دو مقدار مساوی است.

۳-شکاف‌ها برای خاصیت‌های مرکب مسدود می‌شوند. برای تمام مقادیر در شکاف، اشیاء ارجاع داده شـده از طریق خاصیت مرکب ارجاع به عقب می‌شوند؛ علاوه بر این هیچ شکافی از خاصیت مرکب در اشیاء دیگر بـه مالک این شکاف ارجاع داده نمی‌شود.

۴-مقادیر شکاف‌ها زیرمجموعه‌ای از آن دسته هستند که برای هر شکاف خود زیرمجموعه هستند.

LinkSlot

۱-جایی که ویژگی، پایان جهت‌دار است، شکاف ClassInstance با شکاف پیوند سازگار است.

۲-تعداد پیوندها با تعدد و یگانگی پایان‌ها سازگار است.

PrimitiveDataValue

۱-اگر جداکننده شمارشی باشد، valueRepresentation نام معتبر EnumerationLiteral است.

Further constraints on Abstract Syntax

در زیر محدودیت‌های جدیدی امده که توصیه می‌شود معرفی شوند.

Datatype

برای تمام خاصیت‌ها، isReadOnly ،true است. IsComposite، false است، isDerivedUnion، false است.

Datatype ممکن است در ارتباطات شرکت نکند

PrimitiveType

برای تمام خاصیت‌ها، isDerived، true است.

Enumeration

برای تمام خاصیت‌ها، isDerived، true است.

Property

اگر یکی از زوج مرکب‌ها، isUnique باشد، دیگری نیز باشد باشد. منتها یکی از redefinedProperty وsubsettedProperty باید تنظیم شود.

Association

یک رابطه انجمنی اگر خاصیت‌های آن مشتق شود، مشتق می‌شود.

۴-۱۵ ملاحظاتی بر روی مدل‌سازی نمونه MOF

این زیربند قابیت‌های بازتابی را برحسب مدل نمونه بالا، مدل‌سازی می‌کند: بنابراین هر امضاء بازتابی، ترجمه/ مدل‌سازی می‌شود همان‌طور که عملگر هم‌ارزی برروی مدل نمونه‌ها در بالا ترجمه/ مدل‌سازی شده است.

پیاده‌سازی برای خاصیت‌ها و عملیات مشتق شده، دارای ابهام است: تابع ساخت *extInvoke* برای فراخوانی کد درنظرگرفته شده در پیاده‌سازی به کار می‌رود. هیچ شکافی برای خاصیت‌های مشتق شده تخصیص داده نمی‌شود.

تابع اضافی *extend()* برای بازنمایی بسط و بسط‌های حاضر یک شیء به کار می‌روند. مقدار آن مستقل از محتوا است. به علاوه انتظار می‌رود کارخانه با دست‌کم یک بسط همبسته شود.(این امر به خوبی در دامنه RFP ابزار MOF2 وجود دارد)

هیچ تمایزی بین شکاف‌ها بر مبنای تعدد ساخته نمی‌شود: فرض بر این است که شکاف می‌تواند مجموعه‌ای را نگهداری کند؛ همچنین مجموعه می‌تواند نمونه معتبری از Datavalue باشد. در میان انقیادهای زبان، انتظار می‌رود برای خاصیت چند مقداری (upper bound > 1) که مجموعه خالی به جای تهی بازگشت داده می‌شود: برای ساده‌سازی مشخصات این امر اینجا صورت نمی‌گیرد.

عملیات کمکی/ساده OCL به کار می‌رود.

برای روشن‌تر شدن، مقدار قابل تشخیص *'null'* در اینجا برای مقادیر خاصیت جهت نشان دادن این که آنها تهی هستند، به کار می‌رود.

۵-۱۵ قابلیت‌های شیء

Object::getType(): Type به صورت **ObjectInstance::getType(): Type** مدل‌سازی می‌شود.

post: result = self.classifier

Object::container(): Object به صورت **Instance::container(): ClassInstance** مدل‌سازی می‌شود.

post: result = self.get(self.owningProperty())

Object::get(Property p): Element

به صورت **ObjectInstance::get(Property p): ElementInstance** مدل‌سازی می‌شود.

--اگر یک پایان ارتباط خارجی وجود داشته باشد، پیوند را پیمایش می‌کند در غیر این صورت به شکاف دسترسی پیدا می‌کند یا مقدار را اشتقاق می‌یابد.

post: (p.namespace.isOclType(Association) and result = navigate(p)) or
self.propertySlot(p) <> null and (
(self.propertySlot(p).value <> null and result = self.propertySlot(p).value) or
result = p.default) or
(p.isDerivedUnion and result = unionedProperties(p)->union(s| s = self.get(s)) or
(p.isDerived and result = self.extInvoke('get', p))

Object::set(Property p, Element v) به صورت

ObjectInstance::set(Property p, ElementInstance v) مدل‌سازی می‌شود.

pre: not(p.isReadOnly)
post: internalSet(p, v)

Object::isSet(Property p): Boolean به صورت ObjectInstance::isSet(Property p): Boolean مدل‌سازی می‌شود.

post: result = (self.propertySlot(p).value = null)

Object::unset(Property p) به صورت **ObjectInstance::unset(Property p)** مدل‌سازی می‌شود.

pre: not(p.isReadOnly)

--به پیش‌فرض خاصیت تنظیم شود. این امر اگر پیش‌فرض تهی نباشد (null)، اثر مطلوب خواهد داشت.

post: internalUnset(p)

Object::delete() به صورت **ObjectInstance::delete()** مدل‌سازی می‌شود.

--تمام اشیاء مرکب و شکاف‌ها حذف شوند.

post: (self.allProperties->select(p| isComposite(p), delete(self.get(p))) and
self.allSlottableProperties->forAll(p| destroyed(self.propertySlot(p))) and
extent().removeObject(self)
not(extent().objects() includes self) and
destroyed(self)

Object::invoke(Operation op, Set{Tuple{Parameter p, ValueSpecification v}} args):Element

به صورت

ClassInstance::invoke(Operation op, Set{Tuple{Parameter p, ValueSpecification v}} args):Element

مدل‌سازی شود.

--اطمینان حاصل شود که تمام پارامترها برای این عملیات در نظر گرفته شده و تمام مقادیر از نوع درست هستند و تمام پارامترهای الزامی در نظر گرفته شده است.

pre: args->forAll(Tuple{p, v}| op.parameter includes p and conformsTo(p.type, v)) and
op.parameter->select(p| p.lower > 1, args includes Tuple{p, x})

Object::isInstanceOfType(type: Class, includeSubclasses: Boolean): Boolean به صورت

ClassInstance::isInstanceOfType(type: Class, includeSubclasses: Boolean): Boolean
مدل‌سازی شود.

post: result = (self.classifier = type or
includeSubclasses and self.classifier.allParents() includes type)

۱۵-۶ قابلیت‌های پیوند

Link:equals(otherLink:Link): Boolean به صورت

AssociationInstance:equals(otherLink:AssociationInstance): Boolean مدل‌سازی می‌شود.

post: result = (self.association = otherLink.association and
self.firstSlot.value = otherLink.firstSlot.value and
self.secondSlot.value = otherLink.secondSlot.value)

Link:delete() به صورت **AssociationInstance:delete()** مدل‌سازی می‌شود.

post: destroyed(self.firstSlot) and destroyed(self.secondSlot) and
extent().removeObject(self) not(extent().objects() includes self) and
destroyed(self)

۱۵-۷ قابلیت‌های کارخانه

موارد زیر بر روی رده (کلاس) کارخانه تعریف می‌شود:

Factory::createObject(Type t, Set{Tuple{Property p, ValueSpecification v}} args): Object

--شیء و شکاف‌ها را برای خاصیت ایجاد می‌کند (از جمله آنهایی که به ارث برده شده‌اند) که مشتق نشده‌اند و مجدد تعریف نشده با زیرمجموعه دیگری نباشند. مقادیر در نظر گرفته شده یا مقادیر پیش‌فرض در صورت وجود تخصیص داده می‌شود.

pre:

--تمام آرگومان‌ها، خاصیت‌های معتبر از نوع درست هستند و مقادیر برای تمام خاصیت‌های الزامی بدون پیش‌فرض در نظر گرفته می‌شود.

not(isAbstract(t)) and

args->forAll(Tuple{p, v}| t.allProperties includes p and conformsTo(p.type, v)) and

op.parameter->select(p| p.lower > 1 and p.default = null and args includes Tuple{p, x})

--شکاف‌ها ایجاد می‌شوند و مقادیر از آرگومان‌ها یا پیش‌فرض‌ها تنظیم می‌شود.

post: oclIsNew(result) and

extent().addObject(result)

extent().objects includes result and

result.classifier = c and

t.allSlottableProperties->forAll(a | exists(s:StructureSlot | oclIsNew(s) and

s.definingFeature = a and

s.owningInstance = result) and

t.allProperties->forAll(p| (exists(v | args includes [p,v] and internalSet(p,v))) or

(self.internalUnset(p))

--همچنین نیاز به فراهم‌سازی برای تنظیم خاصیت‌ها با استفاده از محدودیت (؟) است.

Factory::createLink(association : Association, firstObject : Object, secondObject : Object) : Link

به صورت

Factory::createLink(association : Association, firstObject : Object, secondObject : Object) : AssociationInstance

مدل‌سازی می‌شود.

--همچنین نیاز به پیوند و تخصیص اشیاء مورد نظر است.

Pre

--وارسی می‌شود که آیا اشیاء نمونه‌های معتبری از نوع درست هستند.

not(association.isAbstract) and conformsTo(association.memberEnd[0].type, firstObject) and conformsTo(associa tion.memberEnd[1].type, secondObject)

post: oclIsNew(result) and

extent().addObject(result) and

extent().linksOfType(association) includes result and

result.classifier = association and

oclIsNew(s1) and s1.definingFeature = association.memberEnd[0] and s1.value = firstObject and

oclIsNew(s2) and s2.definingFeature = association.memberEnd[1] and s2.value = secondObject

به صورت **Factory::createFromString(dataType: DataType, string: String) : Element**

Factory::createFromString(dataType: DataType, string: String): DataValue مدل‌سازی می‌شود.

--انتشار: نیاز به مجموعه قوانین نحو لغوی دارد.

pre: self.package.member includes dataType

Factory::convertToString(dataType: DataType, element: Element) : String به صورت

Factory::createFromString(dataType: DataType, element: DataValue): String مدل‌سازی می‌شود.

--پیامد: نیاز به مجموعه قوانین نحو لغوی دارد.

pre: self.package.member includes dataType

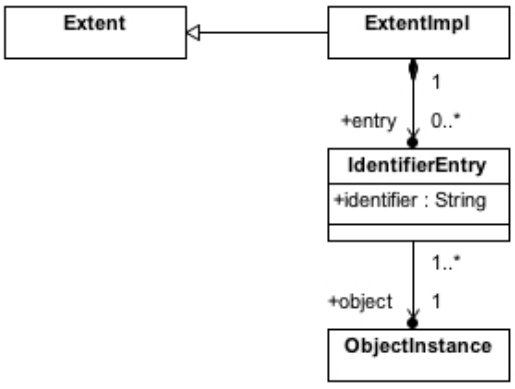
post: createFromString(dataType, result) = element

--توصیه می‌شود رشته تولید شده به مقدار مشابه تجزیه شود.

۸-۱۵ قابلیت‌های بسط

این زیربند، قابلیت‌های کمینه‌ای را برای هسته MOF توصیف می‌کند. در این زیربند به پی‌آمدها از قبیل ایجاد بسط اشاره نمی‌شود که در دامنه ابزار MOF2 و RFP چرخه حیات شیء وجود دارد. هیچیک از کاربردهای صفات 'exclusive' و 'useContainment' را پوشش نمی‌دهد. فرض بر این است که اشیاء به طور مستقیم در یک بسط گنجانده می‌شود.

مانند مدل انتزاعی رفتار، بسط‌ها با استفاده از مدل نمونه زیر پیاده‌سازی می‌شود.



شکل ۱۷- مدل نمونه برای بسط

Extent::objects(): Object به صورت **ExtentImpl:objects(): ObjectInstance** مدل‌سازی می‌شود.

post: result = entry->object

Extent::objectsOfType(type: Class, includeSubtypes: Boolean): Object به صورت

ExtentImpl:objectsOfType(type:Class, includeSubtypes: Boolean): ObjectInstance
مدل‌سازی می‌شود.

post:

result = self.entry->object->select(o| o.classifier = type or

(includeSubtypes and o.classifier.allParents includes type)

Extent::linksOfType(type: Association): Link به صورت

linksOfType(type: Association): AssociationInstance مدل‌سازی می‌شود.

post:

result = self.entry->object->select(o| o.classifier = type)

Extent::linkedObjects(association : Association, endObject : Object, end1ToEnd2Direction : Boolean) : Object

به صورت

ExtentImpl:linkedObjects(association : Association, endObject : Object, end1ToEnd2Direction : Boolean) : ObjectInstance

مدل‌سازی می‌شود.

post:

result = self.entry->object->select(o, r| o.classifier = association and

((end1ToEnd2Direction and o.firstObject = endObject and r = o.secondObject) or

o.secondObject = endObject and r = o.firstObject))

Extent::linkExists(association : Association, firstObject : Object, secondObject : Object) : Boolean

به صورت

Extent::linkExists(association : Association, firstObject : Object, secondObject : Object) : Boolean

مدل‌سازی می‌شود.

result = self.entry->object->exists(o| o.classifier = association and

o.firstSlot.value = firstObject and

o.secondSlot.value = secondObject)

Extent::identifier(o: Object): String به صورت

ExtentImpl::identifier(o: ObjectInstance): String مدل‌سازی می‌شود.

post:

result = self.entry->object->select(eo| eo = o).identifier

Extent::object(id: String): Object به صورت

ExtentImpl::object(id: String): ObjectInstance مدل‌سازی می‌شود.

post:

result = self.entry->select(i| i = id).object

۹-۱۵ عملیات افزوده

[۱]

این بخش نیاز به شکاف را برای تمام خاصیت‌های رده (کلاس) (از جمله ارث‌بری شده‌ها) را به استثنا خاصیت‌های مشتق شده و مجدد تعریف شده تعیین می‌کند.

Class::allSlottableProperties(): Set(Property);

result = self.allProperties()->select(not is Derived)

[۲]

تمام خاصیت‌هایی که تعریف مجدد نشده‌اند (از جمله ارث‌بری شده‌ها) از رده (کلاس).

Class::allProperties(): Set(Property);

result = member->select(oclIsKindOf(Property).oclAsType(Property)

[۳]

تمام خاصیت‌هایی که به طور مستقیم یا غیر مستقیم مجدد توسط خاصیت تعریف شده‌اند.

Property::allRedefinedProperties() : Set(Property)

result = self.redefinedProperty->union(self.redefinedProperty.allRedefinedProperties())

[۴]

این بخش شکاف متناظر با خاصیت درنظر گرفته شده را بازمی‌گرداند. برای خاصیت‌هایی که مجدد تعریف شده‌اند، شکاف متناظر با همان خاصیت مجدد تعریف شده است.

به یاد داشته باشید خاصیت‌های مشتق شده تنها وقتی دارای شکاف هستند که توسط یک خاصیت غیرمشتق شده مجدد تعریف شوند و نتیجه ممکن است تهی باشد.

result = self.slot->any(definingFeature =

p.applicableDefinition(self.classifier))

[۵]

این بخش خاصیتی را بازمی‌گرداند که شکاف دارای داده‌هایی برای خاصیت درخواست داده شده در یک نمونه از رده (کلاس) خواهد بود

```
Property::applicableDefinition(Class c): Property
applicableDefinition = c.allSlottableProperties().any(p |
p=self or p.allRedefinedProperties()->includes(self))
```

[۶]

این بخش خاصیت منفردی را با یک شکاف بازمی‌گرداند که مالک فعلی شیء را مبتنی بر نمونه فعلی نمایش می‌دهد.

```
Object::owningProperty(): Property modeled as ClassInstance::owningProperty(): Property
result = self.classifier.allSlottableProperties()->any(p |
p.opposite <> null and p.opposite.isComposite and self.get(p)<> null)
```

[۷]

تمام خاصیت‌های یک شیء که مجدد تعریف نشده‌اند.

```
result = self.classifier.allProperties()
```

[۸]

این بخش خاصیت‌هایی را که زیرمجموعه‌ای از اجتماع مشتق شده است را بازمی‌گرداند.

```
Object::unionedProperties(p: Property): Set(Property)
pre: p.isDerivedUnion
post:
result = self.allProperties->select(sp| sp.subsettedProperty includes p)
```

[۹]

این بخش تمام مجدودیت‌های جداکننده را بازمی‌گرداند.

```
CMOF::Classifier::allConstraints(): Set(Constraint)
post:
result = extent().objectsOfType(Constraint)->select(c | c.context = self)
```

[۱۰]

این بخش مقدار خاصیت را خواه فقط خواندنی تنظیم می‌کند. (برای مقداردهی اولیه به کار می‌رود.)

```
ObjectInstance::internalSet(Property p, ElementInstance v)
post: (self.propertySlot(p) <> null and self.propertySlot(p).value = v) or
(p.namespace (self.p.namespace.isOclType(Association) and
not (self.allParents() includes p.namespace) and -- allow access to own assoc ends setLink(p, v)) or
(p.isDerived and result = self.extInvoke('set', p, v))
```

[۱۱]

این بخش مقدار خاصیت را بدون در نظر گرفتن اینکه فقط خواندنی باشد یا خیر برمی‌دارد. (برای مقداردهی اولیه به کار می‌رود.)

ObjectInstance::internalUnset(Property p)
post: (self.propertySlot(p) <> null and self.propertySlot(p).value = null) or
(p.isDerived and result = self.extInvoke('unset', p, v))

[۱۲]

این بخش یک شیء را به یک بسط می‌افزاید – تنها برای ایجاد.

ExtentImpl::addObject(ObjectInstance o, String suppliedId [0..1]): String
pre: not(self.entry.identifier includes suppliedId)
post: oclIsNew(e) and oclType(e) = IdentifierEntry and
e.object = o and
self.entry includes e
self.entry->select(ex | ex.identifier = e.identifier)->size() = 1 -- the new id is unique and
(suppliedId <> null implies e.identifier = suppliedId)

[۱۳]

این بخش یک شیء را از یک بسط خارج می‌کند. (تنها برای از بین بردن).

ExtentImpl::removeObject(ObjectInstance o)
pre: self.objects includes o
post: let e = self@pre.entry->select(ex|ex.object = o) and
destroyed(e) and
not(self.entry includes e)

[۱۴]

این بخش پایان ارتباط را از یک شیء پیمایش می‌کند.

ObjectInstance::navigate(Property p): Set(ObjectInstance)
pre: p.namespace.isOclType(Association)
post:

--پیوندهای مرتبط را با پرس‌وجوی بسط پیداکن.

--پی‌آمد: نیاز به سروکار داشته با زیرمجموعه‌ها/ اجتماع‌ها دارد.

let values= extent().objectsOfType(p.namespace)->select(link| link.get(p.opposite) = self)->get(p) and

(p.isUnique implies result = oclAsSet(values))

and not(p.isUnique implies result = values)

[۱۵]

پایان ارتباط را تنظیم می‌کند.

ObjectInstance::setLink(Property p, Element v)

-- اگر خاصیت چندمقداری باشد سپس به عناصر جداگانه تقسیم می‌شود و پیوندها را ایجاد می‌کند.

--در هریک از موارد حذف پیوندهای موجود

pre: p.namespace.isOclType(Association)

post: let oldValues= extent@pre().objectsOfType(p.namespace)->select(link|
link.get(p.opposite) = self)->get(p)

and

values->forAll(v| v.delete()) and

(p.upper = 1 implies self.createLink(p, v)) and

(p.upper > 1 implies v->forAll(o| createLink(p, o))

[۱۶]

پیوند جداگانه‌ای را ایجاد می‌کند.

ObjectInstance::createLink(Property p, ObjectInstance v)

--از ایجاد معمولی شیء استفاده شود.

post: factory.create(p.namespace, Set{ Tuple{p, v}, Tuple{p.opposite, self}})

پیوست الف

(آگاهی‌دهنده)

XMI برای هسته MOF 2

URLها در این پیوست، مراجع الزامی برای پرونده‌های قابل‌خواندنیِ ماشین، همبسته با این استاندارد هستند.

Package::URI برای EMOF:

http://www.omg.org/spec/MOF/20110701/emof.xmi

و برای CMOF:

http://www.omg.org/spec/MOF/20110701/cmof.xmi

هسته MOF2 به طور مستقیم از ابرساختار L3، UML2 برای بازنمایی فرامدل استفاده مجدد می‌کند. بنابراین اگرچه EMOF و CMOF دارای بسته URI خود هستند، آنها فضای‌نامی یا پیشوند خود را تعریف نمی‌کنند. (که بسته URI را از طریق برچسب org.omg.xmi.nsURI باطل می‌کند.

MOF 2.4 از UML 2.4 استفاده می‌کند و بنابراین:

uri فضای‌نام:

http://www.omg.org/spec/UML/20110701

پیشوند فضای‌نام:

Uml

بسته بسط MOF برای برچسب‌های MOF، استثنا موارد بالا است که توسط UML پوشش داده نمی‌شود و دارای فضای‌نامی به صورت زیر است:

uri فضای‌نام:

http://www.omg.org/spec/MOF/20110701

پیشوند فضای‌نام:

Mofext

یادآوری- برای انتقال این مقاصد، تبدیل در هردوجهت فرامدل‌های MOF 2 EMOF و CMOF و همچنین فرامدل‌های انطباق که در UML 2.4 بازنمایی‌شده بدون هیچ کاستی امکان‌پذیر است.

- برچسب contentType به ”any“ برای عنصر تنظیم می‌شود.

پیوست ب

(آگاهی‌دهنده)

محدودیت‌های فرامدل در OCL

محدودیت‌ها به منظور اعتباربخشی اینکه آیا فرامدل معتبری همراه با پرونده‌ها همان‌طور که در زیر اشاره شده است، ایجاد شده یا خیر، باید در مدل UML به کار رود. آنها تحت‌الشاع محیط OCL هستند.

محدودیت‌ها برای EMOF:

http://www.omg.org/spec/MOF/20110701/CMOFConstraints.ocl

محدودیت‌ها برای CMOF:

http://www,omg.org.spec/MOF/20110701/EMOFConstraints.ocl

پیوست پ

(آگاهی‌دهنده)

مهاجرت از MOF 1.4

پ-۱ کلیات

هنگامی که MOF 2 تولید واقعی از MOF(MOF 2) باشد، جایگزین نسخه MOF 1.4 نشده و آن را لغو نمی‌کند[MOF 1] . برای همین یک مهاجرت و انتقال الزامی برای فرامدل‌های MOF 1.4 به فرامدل‌های MOF 2 CMOF توسط این پیوست ایجاد شده است. فرامدل‌های MOF 1.4 را می‌توان مبتنی بر نگاشت مستقیم به فرامدل‌های MOF 2ترجمه کرد که این کار می‌تواند کلاً همان‌طور که در زیر اشاره شده است، خودکار صورت بگیرد. به یاد داشته باشد صفاتی که دارای نگاشت صریح مستقیمی هستند در زیر فهرست نشده‌اند (به طور مثال، ModelElement::name از MOF 1.4 به NamedElement::name از MOF 2).

این پیوست مهاجرت و انتقال فرامدل‌های MOF 1.4 را به MOF 2 کامل نشان می‌دهد. فرامدل‌های MOF 1.4 را می‌توان مبتنی بر نگاشت مستقیم به مدل‌های MOF 2 ترجمه کرد که این کار می‌تواند کلاً همان‌طور که در زیر اشاره شده است، خودکار صورت بگیرد. به یاد داشته باشد صفاتی که دارای نگاشت صریح مستقیمی هستند در زیر فهرست نشده‌اند (به طور مثال، ModelElement::name از MOF 1.4 به NamedElement::name از MOF 2).

پ-۲ مهاجرت و انتقال فرامدل

یک فرامدل معتبر MOF 1.4 را می‌توان مبتنی بر نگاشت مستقیم به مدل‌های MOF 2 ترجمه کرد که این کار می‌تواند خودکار صورت بگیرد.

MOF 1.4	نگاشت MOF2
ModelElement::annotation	نمونه جدید از رده (کلاس) Comments::Comment که به عنصر متناظر از طریق صفت annotatedElement پیوند داده شده و صفت Comment::body به مقدار annotation و صفت Comment::usage به "documentation" تنظیم می‌شود.
ModelElement::container	NamedElement::namespace (به یاد داشته باشید که این انتزاعی است که تخصیص مناسبی باید برای آن به کار رود)
ModelElement::constraints	رابطه بین محدودیت و عنصر محدود شده تنها توسط محدودیت به عنصر محدود شده قابل‌پیمایش است و نه برعکس. برای محدودسازی یک عنصر، نیاز به افزودن عنصر به صفت Constraint::contexthx از محدودیت مفروض دارد.
Namespace::contents	Namespace::ownedMember (به یاد داشته باشید که این انتزاعی است که تخصیص مناسبی باید برای آن به

MOF 1.4	نگاشت MOF2
	کار رود)
GeneralizableElement::isLeaf	پشتیبانی نمی‌شود. می‌توان از این صفت بدون از دست دادن هرگونه اطلاعاتی صرفنظر شود همان‌طور که صفت تحمیل می‌کند مدل و نه اشیاء مدل‌سازی شود.
ModelElement::container	NamedElement::namespace (به یاد داشته باشید که این انتزاعی است که تخصیص مناسبی باید برای آن به کار رود)
ModelElement::constraints	رابطه بین محدودیت و عنصر محدود شده تنها توسط محدودیت به عنصر محدود شده قابل‌پیمایش است و نه برعکس. برای محدودسازی یک عنصر، نیاز به افزودن عنصر به صفت Constraint::contexthx از محدودیت مفروض دارد.
Namespace::contents	Namespace::ownedMember (به یاد داشته باشید که این انتزاعی است که تخصیص مناسبی باید برای آن به کار رود)
GeneralizableElement::isLeaf	پشتیبانی نمی‌شود. می‌توان از این صفت بدون از دست دادن هرگونه اطلاعاتی صرفنظر شود همان‌طور که صفت تحمیل می‌کند مدل و نه اشیاء مدل‌سازی شود.
GeneralizableElement::isRoot	پشتیبانی نمی‌شود. می‌توان از این صفت بدون از دست دادن هرگونه اطلاعاتی صرفنظر شود همان‌طور که صفت تحمیل می‌کند مدل و نه اشیاء مدل‌سازی شود.
GeneralizableElement::supertypes	Classifier::general
Class::isSingleton	از رده (کلاس) MOF 1.4 و isSingleton=true دیگر به طور مستقیم پشتیبانی نمی‌کند. می‌تواند توسط درج یک محدودیت بر رده (کلاس) زیر شبیه‌سازی شود. self.metaobject.allInstances.size=1
CollectionType	به طور مستقیم پشتیبانی نمی‌شود. می‌تواند توسط یک نمونه از رده (کلاس) DataType با هیچ صفتی ('value' نامیده می‌شود) از همان نوع و تعدد مانند CollectionType جایگزین شود.
EnumerationType	به Enumeration نگاشت می‌شود که رشته‌ها در مقدار صفت EnumerationType::labels از MOF1.4 به نمونه رده (کلاس) EnumerationLiteral با رشته برچسب مانند نام و همان ترتیب نگاشت می‌شود EnumerationLiteral به یک enumeration از طریق صفت Enumeration::ownedLiteral پیوند داده می‌شود.
AliasType	پشتیبانی نمی‌شود. به یک زیرنوع از نوع داده به‌هم چسبیده نگاشت می‌شود که متناظر با نوعی که AliasType بدان اشاره می‌کند، است. هر محدودیتی از AliasType را به زیرنوع می‌چسباند.
StructureType	به DataType نگاشت می‌شود.
StructureField	به Property که برای DataType است که متناظر با نوعی است که AliasType بدان اشاره می‌کند، نگاشت می‌شود. هر محدودیتی از AliasType را به زیرنوع می‌چسباند.
Feature::scope	پشتیبانی نمی‌شود- تمام ویژگی‌ها در MOF 2 نمونه درسطح هستند.
StructuralFeature::isChangeable	به Property::isReadOnly نگاشت می‌شود که دارای معنای وارون است (یعنی

MOF 1.4	نگاشت MOF2
	isReadOnly=isChangeable نباشد)
Reference	افزونگی مانند یک عنصر مجزا از AssociationEnd ارجاع داده شده (که در حال حاضر خصوصیت است). حقیقت این است که AssociationEnd دارای ارجاعی باشد بدین معنا که توصیه می‌شود خصوصیت جدید متناظر با AssociationEnd برای رده (کلاس) باشد نه برای رابطه.
Reference::referencedEnd	خصوصیت بازنمایی‌کننده خود رابطه. افزونگی مانند بالا است.
Reference::exposedEnd	Property::opposite افزونگی مانند بالا است.
Operation::exceptions	Operation::raisedException
Exception	یک استثناء می‌تواند هرگونه زیررده (زیرکلاس) مورددلخواه باد. طبق پیش‌فرض توصیه می‌شود، رده‌ای (کلاسی) با همان نام به عنوان استثناء باشد.
AssociationEnd	Property همبسته با یک ارتباط از طریق صفت memberEnd . برای ایجاد Property (یعنی AsoociationEnd) که در رده (کلاس) نمایش داده شده است (مشابه با استفاده از مرجع در MOF 1.4)، نیاز به این دارد که رده (کلاس) آن را دارا باشد (زمانی‌که همچنان memberEnd از رابطه باشد).
AssociationEnd::isNavigable	همان‌طور که در MOF 1.4 تعریف شده، پشتیبانی نمی‌شود. در MOF 2 تمام پایان‌ها برحسب پیمایش MOF 1.4 قابل پیمایش هستند MOF 2 .قابلیت پیمایش را برحسب قابلیت پیمایش به طور مستقیم از یک نوع تا پایان تعریف می‌کند- دارا بودن مرجع در MOF 1.4 آنالوگ است.
AssociationEnd::aggregation	Property::isComposite (مرکب به true نگاشت می‌شود، هیچ نگاشتی به false نمی‌شود، shared که در حال مشخص شدن در MOF 1.4 است به false نگاشت می‌شود.
AssociationEnd::isChangeable	به نگاشت StructuralFeature::isChangeabke مراجعه شود.
Import	رده‌های (کلاس‌های) PackageImport، PackageMerge یا ElementImport . اگر تمام عناصر درونبردشده بسته‌ها باشند، Import به PackageImport در مورد isClustered=false و به PackageMergeدر مورد isClustered=trueنگاشت می‌شود.۰(این امر ممکن است نیاز داشته باشد تا از الحاق با RFP ابزاMOF 2 دوری جست.)
Import::importedNamespace	PackageImport::importedPackage,PackageMerge::mergedPackage , ElementImport::importedElement به یاد داشته باشید که در MOF 2، یک درونبرد نام‌ها را در داخل بسته‌های درونبردشده ایجاد می‌کند که قابل مشاهده بدون هیچ شرطی است.
Tag	Extension::Tag

پ-۳ مهاجرت و انتقال API

این زیربند APIهای هم‌ارز بین API بازتابی MOF 1.4 ، JMI API و API هسته MOF 2 را به طور خلاصه آورده است. نام‌های رده (کلاس) به صورت پررنگ و مورب به کار رفته است.

یادآوری- بیشتر اختلافات برای MOF 2 در جایی است که API قبلی دارای عملیات خاصی (مرتبط با پیشکارهای[1] رده در انقیاد زبانی داشته‌اند یا عملیات ساده‌ای بودند.

MOF 1.4	JMI	MOF 2
RefBaseObject	RefBaseObject	Object
refMofId	refMofId	Extent::identifier
refMetaObject	refMetaObject	getMetaclass
refImmediatePackage	refImmediatePackage	
refOutermostPackage	refOutermostPackage	
refItself	Java Object.equals	Element::equals
refVerifyConstraints	refVerifyConstraints	
refDelete	RefObject::refDelete	delete
RefObject	RefFeatured	
refValue	refGetValue	get
refSetValue	refSetValue	set
refUnsetValue	Set value to null	unset
refAddValueBefore	Use of live collections	
refAddValueAt	ditto	
refModifyValue	ditto	
refModifyValueAt	ditto	
refRemoveValue	ditto	
refRemoveValueAt	ditto	
refInvokeOperation	refInvokeOperation	invoke
Test for value = null	Test for value = null	isSet
	RefObject	
refIsInstanceOf	refIsInstanceOf	isInstanceOfType
	refClass (gets proxy)	
refImmediateComposite	refImmediateComposite	container
refOutermostComposite	refOutermostComposite	
RefBaseObject::refDelete	refDelete	delete
RefClass	RefClass	
refCreateInstance	refCreateInstance	Factory::create
refAllObjects (includeSubtypes=true)	refAllOfType	Extent::elementsOfType (includeSubtypes=true)
refAllObjects (includeSubtypes=false)	refAllOfClass	Extent::objectsOfType (includeSubtypes=false)
	refCreateStruct	Factory::createFromString
	refGetEnum	Factory::createFromString
refAssociation	refAssociation	
refAllLinks	refAllLinks	Extent::linksOfType
refLinkExists	refLinkExists	Extent::linkExists
refQuery	refQuery	Extent::linkedElements

1 - proxy

MOF 1.4	JMI	MOF 2
refAddLink	refAddLink	Factory::create
refAddLinkBefore	Use live collections	
refModifyLink	ditto	
refRemoveLink	refRemoveLink	delete
	refAssociationLink (datatype – tuple)	
RefPackage	RefPackage	Extent
refMofId (inherited)	refMofId (inherited)	
refMetaObject(inherited)	refMetaObject(inherited)	
refImmediatePackage(inherited	refImmediatePackage(inherited)	
refOutermostPackage(inherited)	refOutermostPackage(inherited)	
refItself(inherited)	Java Object.equals	
refClassRef	refClass	
refAssociation	refAssociation	
refPackage	refPackage	
	refAllPackages	
	refAllAssociations	
	refCreateStruct	
	refGetEnum	
RefBaseObject::refDelete	refDelete	
		useContainment (attribute)
		elements
		identifier
		element

کتاب‌نامه

[1] [CORBA] Common Object Request Broker Architecture (CORBA) Specification http:// www.omg.org/spec/CORBA/3.3/ ISO/IEC 19500:2012 Information technology - Object Management Group Common Object Request Broker Architecture (CORBA)

[2] [MOFFOL] MOF Facility Object Lifecycle (MOFFOL) http://www.omg.org/spec/MOFFOL/2.0/

[3] [MOFVD] Meta Object Facility (MOF) Versioning and Development Lifecycle Specification http:// www.omg.org/spec/MOFVD/2.0

[4] [MOFM2T] MOF Model to Text Transformation Language http://www.omg.org/spec/MOFM2T/1.0/

[5] [QVT] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification http:// www.omg.org/spec/QVT/1.1

[6] [SMOF] MOF Support for Semantic Structures http://www.omg.org/spec/SMOF/1.0/

[7] [UML1] OMG Unified Modeling Language Specification http://www.omg.org/spec/UML/1.4 ISO/IEC 19501:2005 Unified Modeling Language Specification

[8] [UML2Inf] OMG Unified Modeling Language (OMG UML), Infrastructure http://www.omg.org/spec/ UML/2.4.1/Infrastructure/ ISO/IEC 19505-1:2012 Information technology - Object Management Group Unified Modeling Language (OMG UML), Infrastructure

[9] [XMI2] XML Metadata Interchange http://www.omg.org/spec/XMI/2.0 ISO/IEC 19503:2005 XML Metadata Interchange Specification

[10] [XSD-D] W3C Recommendation XML Schema Part 2: Datatypes Second Edition http:// www.w3.org/TR/2004/REC-xmlschema-2-20041028