Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Магистърска програма
„Софтуерни технологии”

Предмет: Обектно-ориентиран анализ и проектиране на софтуерни системи
Зимен семестър, 2020/2021 год.

Тема 6: Eclipse Modeling Framework

Ece

Автор:
Станислав Траилов
фак. номер 25617

декември, 2020
София

Съдържание

Въведение

Eclipse Modeling Framework(EMF) е Eclipse базирана рамка за моделиране и създаване на код за изграждане на инструменти и приложения, базирани на структуриран модел от данни. EMF предоставя инструменти и runtime support (?) за да създаде множество от Java класове, заедно с множество adapter класове (?) , които позволяват преглед и командно базирана редакция на моделите, както и основен редактор. Важно е да се отбележи в допълнение на това, че EMF е успешна среда за моделиране, също така EMF е стабилен стандарт за много други технологии за моделиране.

Характеристики и използване на EMF (Eclipse Modeling Framework).

Eclipse Modeling Framework може да генерира ефективен, логически правилен и лесно променяем имплементационен код. EMF конвертира моделите си до Ecore (EMF metamodel). Поддържа Reflective API, динамични дефиниции на модели и Persistence API.

История

Първата версия на Eclipse Modeling Framework е пусната през Юни месец на далечната 2002 година. Базира се на MOF (Meta Object Facility), което произхожда от OMG (Object Management Group), с две думи – абстрактни езици и среди за разработка за специфициране, конструиране и управление на неутрални мета-модели. С годините EMF еволюира много бързо, благодарение на големия набор от инструменти идващи от самата платформа Eclipse. Днес много Eclipse проекти използват EMF като например WTP, TPTP, DTP и много други. Също така е използван от много известни компании сред които IBM, Bosch, Oracle и други. EMF може и да се похвали с голямо open-source community.

Дефиниции .

Data model – понякога наричан и domain модел, репрезинтира данните с които искаме да работим. Посредством EMF ние можем да си дефинираме нашия domain модел експлицитно.

Мета-модели - в EMF те се състоят главно от 2 части – ecore и genmodel description файлове. Ecore файла съдържа информация за вече дефинираните класове, докато genmodel файла съдържа информация за генерирането на код, например път и информация до файла. Също genmodel файла съдържа параметри за това как да бъде генериран кода.

Модел – в света на EMF , моделът е инстанция на мета-модел.

Ecore модела всъщност е дървовидна структура и представлява root обект, представящ целия модел. Този модел има деца/наследници, които репрезентирант пакети, чиито деца/наследници репрезентират класове, чиито деца/наследници репрезентират атрибути на тези класове.

Основни характеристики

Eclipse Modeling Framework може да се разгледа като съвкупност от Eclipse plugin-и, които могат да се използват за създаване на data model и да генерират код или друг output от този модел. В света на EMF трябва да разграничаваме думите мета-модел и модел. Мета-моделът описва структурата на модела, а модела сам по себе си е конкретна инстанция на мета-модела. EMF позволява ма потребителите си да създадат мета-модел посредством различни подходи – XMI, Java анотации, UML или XML scheme. Също има опция за съхранение на моделите, по подразбиране използваният формат е XML Metadata Interchange.

EMF моделът пък, който съдържа реални данни базирани на структурата на модела, може да бъде използван и да генерира различни видове output, например HTML страници, или да бъде интегриран с външно приложение.

Eclipse Modeling Framework съдържа три фундаментални части:

- **EMF** – CORE EMF средата за разработка, която включва мета модел (Ecore) за описване на модели и runtime support с XMI сериализация по подразбиране, както и много effective reflective API за манипулиране главно на EMF обекти.
- **EMF.Edit** – фреймуърк включващ generic reusable класове за създаване на редактори на EMF модели. Предоставя
  - Content and label provider classes, поддръжка на пропъртита и други класове, които позволяват EMF моделите да бъдат изобразени посредством стандартни desktop (Jface) viewer-и.
  - Команден фреймуърк , включващ множество от generic command implementation classes за създаване на редактори,които поддържат напълно автоматични undo и redo.
- **EMF.Codegen** – EMF генератор на код facility е способно да генерира всичко нужно за построяване на напълно функционален редактор на EMF модел. Включва графичен потребителски интерфейс, от който опциите за генериране могат да се специфицират и генерирането може да се invoke-не. EMF генератора се свързва с JDT (Java development tooling) компонента на Eclipse.

Работата с EMF е много прагматична – създава се / дефинира се модел в Ecore формат, който представлява подмножество на UML клас диаграми. Вече от този Ecore модел може да бъде генериран Java код.

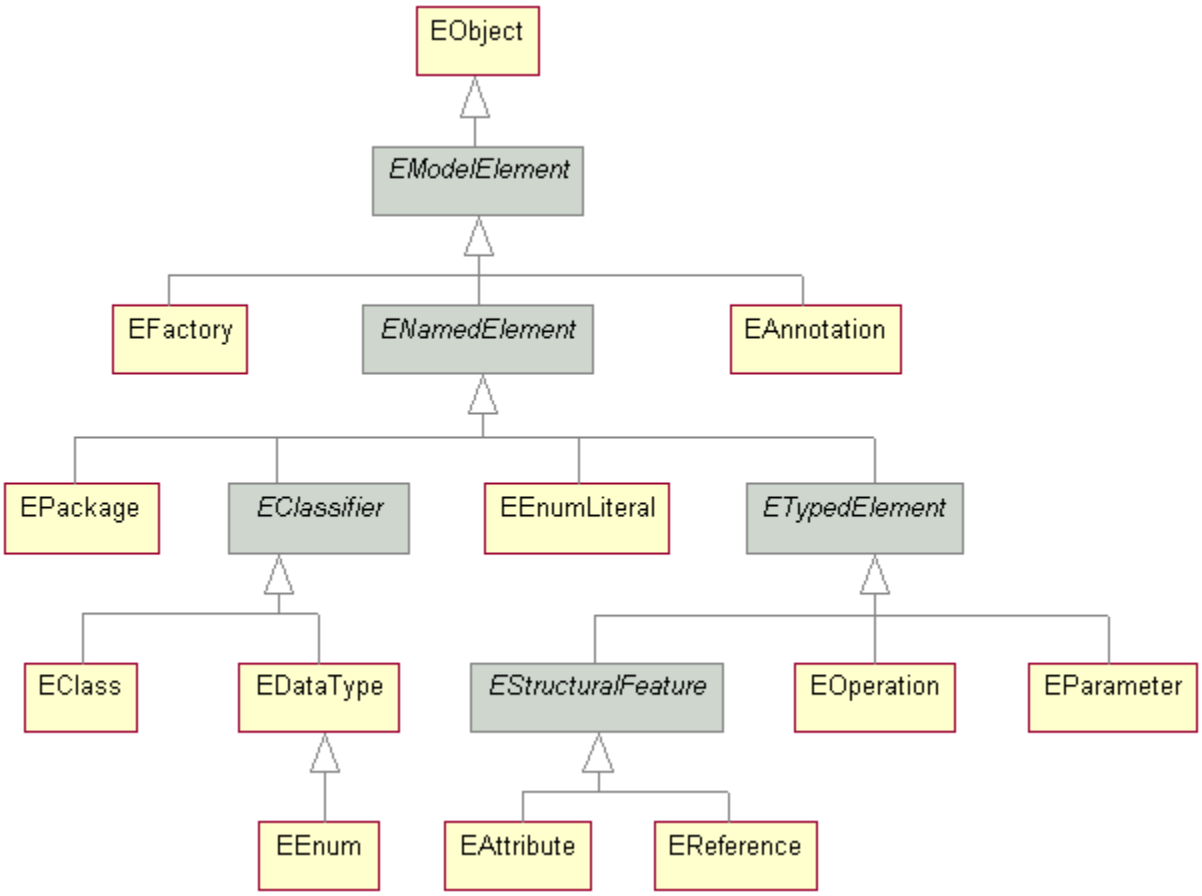
Предоставя support за Java interfaces, UML и XML Schema. EMF конвертира моделите към Ecore (EMF metamodel).

Създаване на модел

EMF ни позволява да създаваме модели посредством Ecore обекти, които позволяват да дефнинираме обекти, връзки, properties и операции на обектите. Те са следните :

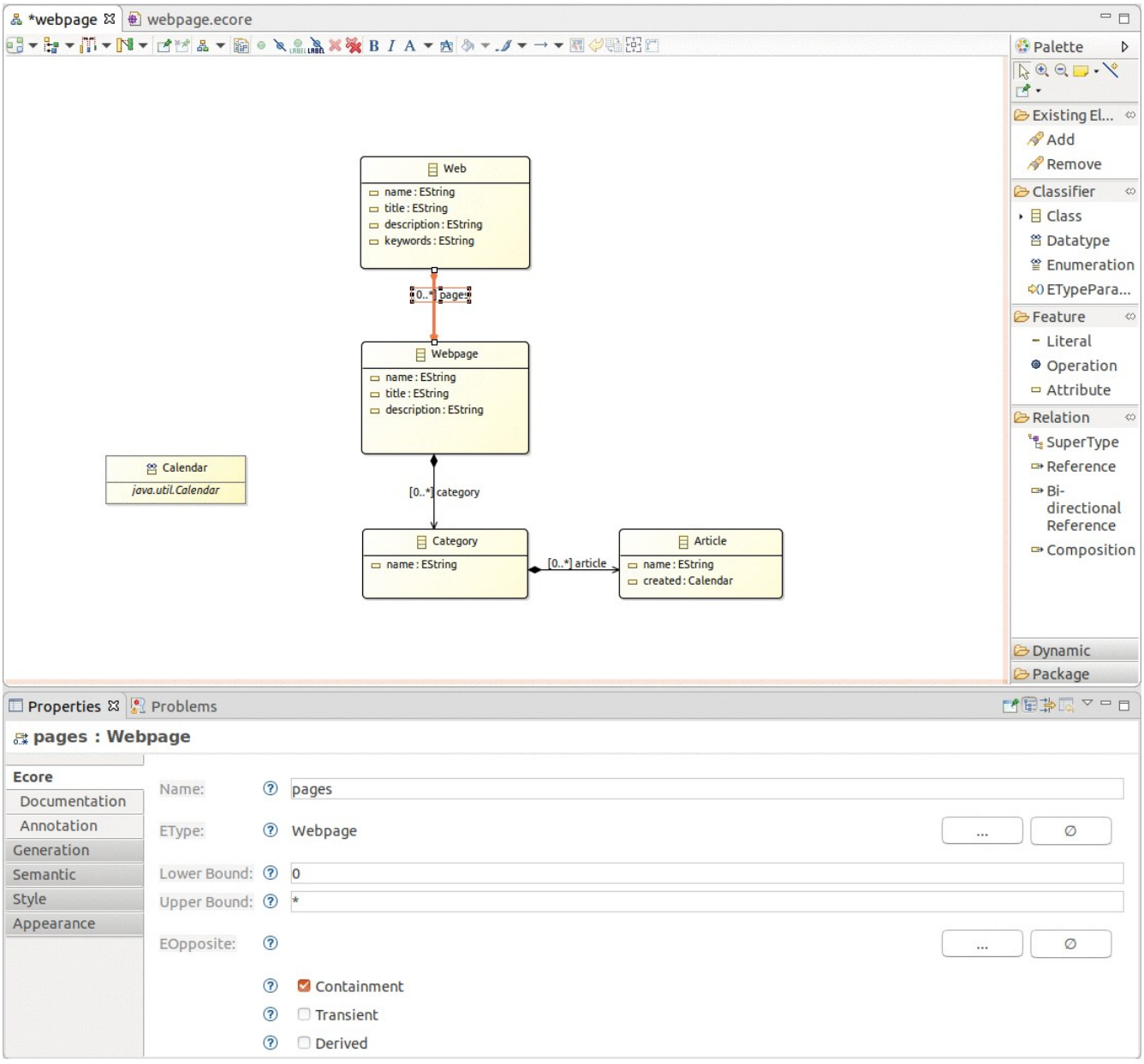
- Eclass – репрезентира клас, с 0 или повече атрибути и 0 или повече връзки.
- Eattribute – представлява атрибут, който съдържа име и тип.

- Ereference – репрезентира асоциация/връзка между два класа. Има флагове, които индикират дали представя ограничение и към кой клас сочи връзката.
- EdataType – тип на атрибут, например int, float или java.util.Date.



Фиг.1 Ecore йерархия

Посредством тези характеристики на EMF и по-специално Ecore могат да се създават модели като дадения пример долу.



Фиг.2 Примерен модел

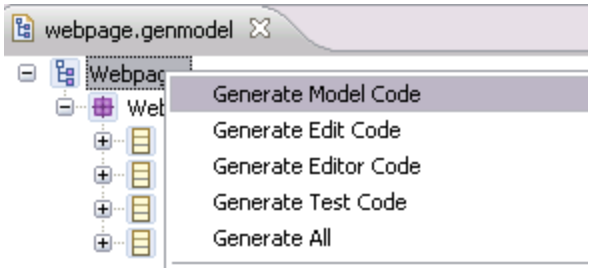
Генериране на код от Модел

Информацията съхранявана в EMF моделите, може да бъде използвана в последствие за достигане на желан резултат. Типичен use-case е използване на EMF за дефиниция на domain модел за вашето приложение и чрез него да генерирате съответсващи Java имплементационни класове от този модел. EMF поддържа и безопасно разширение(на ръка) на генерирания код. Тук трябва да отбележим,. Че ако се промени модела, то наново ще трябва да бъде генериран код, за да е съвместим с модела.

Генерираният код е базиран на .genmodel файлове и неговото генериране се извършва след като натинсте десен бутон на мишката върху тях и изберете Generate Model Code. Генерираният код ще се състои от следните 3 пакета :

- model.<project_name> - пакет съдържащ интерфейсите и factory за създаване на Java класове.

- Model.<project_name>.impl - конретна имплементация на интерфейсите, дефинирани в модела
- model.<project_name>.util – AdapterFactory-то



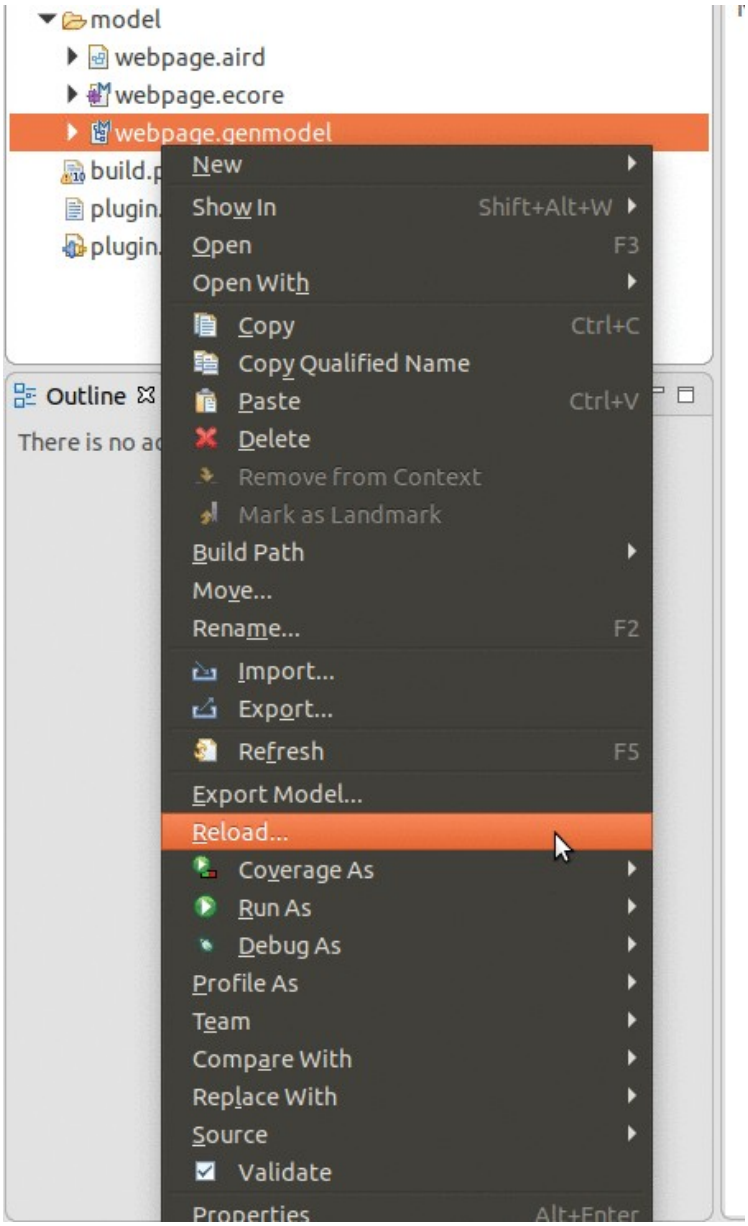
Фиг.3 Генериране на код

Централния фактори обект има методи за създаване не всички дефинирани обекти посредством createObjectName() методи. За всеки атрибут, генерираните интерфейс и имплементация съдържат getter методи (ако е позволено от дефиницията на модела) и setter методи. Всеки setter метод има и генерирана нотификация към т.нар. observers на модела.Това означава, че другите обекти могат да се „нагодят“ към модела и да реагира на промени в модела. Това можем да си го представим като съобщение, когато някой от setter-ите на модела бъде извикан.

Всеки генериран интерфейс наследява Eobject интерфейса. Eobject е базата на всеки EMF клас и е EMF еквивалента на java.lang.Object класа в Java. Eobject и неговия съответен имплементационен клас EobjectImpl предоставят lightweight base клас , който позволява генерираните интерфейси и класове да участват в EMF notification and persistence frameworks ?

Всеки генериран метод пък е анотиран/тагнат с @generated. Ако желаете ръчно да променяте метода или да накарате EMF да overwrite не метода по време на следваща генерация на код, трябва да премахнете тази анотация/таг.

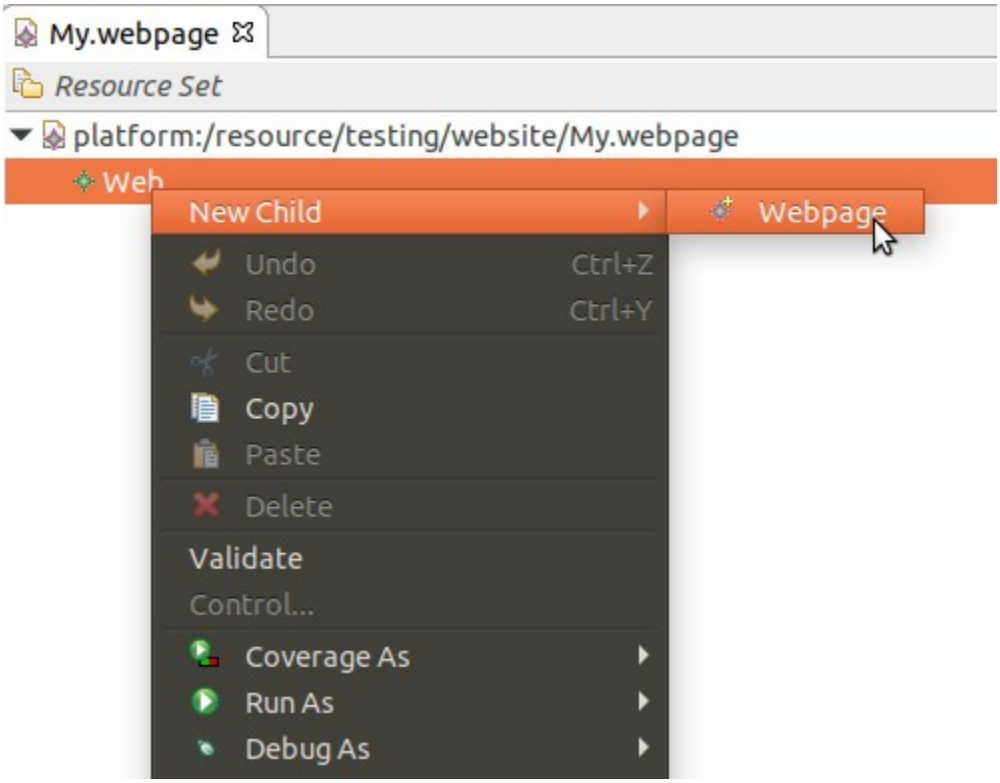
Ако промените вашия .ecore model, може да update-нете .генмодел просто като използвате функцията Reload.



Фиг.4. Reload след генериране

2.3.3 Генериране на EMF Editor plug-ins

След като сме генерирали код от модела, EMF предоставя възможността да бъде генериран напълно функционален Eclipse editor за всеки модел. По подразбиране той се състои от два плъгина – edit плъгин който включва адаптери, които предоставят структуриран изглед и командно-базиран плъгин(editor plugin), който предоставя функционалност за редактиране на обектите от модела. Така имаме възможност да създаваме инстанции на вече създадени модели, който съдържат различни данни.



Фиг.5 Генериране на Editor на модел

2.3.4 Наследяване на модели

EMF позволява разширяване на съществуващи модели посредством наследяване. Например, може да бъде дефиниран базов модел (base model) и негово разширение базирано на него. Така може и да реализираме депълнителна йерархия в нашия модел, което разбира се ще се отрази и на генерираният код. Например, ако имаме модел base.ecore, който съдържа клас BaseClass , може да наследим този модел като създадем нов extended.ecore и укажем кой модел искаме да наследим. След това, ако искаме да наследим вече създаденият BaseClass , EMF ни предоставя тази възможност като посочим ESuper type докато създаваме нов клас – ExtendedClass. След като генерираме код от нашия новосъздаден модел, ще видим, че ExtendedClass наследява BaseClass.

Предимства и недостатъци при използване на EMF

Предимства

- Лесен за инсталация и употреба
- Безплатен и open-source
- Голям набор от потребители (community) включващ активни форуми

Недостатъци

- Въпреки, че EMF предоставя функционалност за създаване на модели, все пак EMF не е UML и не поддържа цялата функционалност на UML.
- Проблеми при опити за използване на мета-модела извън Eclipse.

- Голям брой и размер зависимости, които EMF довлачва. Това може да се окаже проблем, ако искаме да вградим модела в runtime среда.
- Несъвместимост с runtime среди, които използват multi-class лоудъри.
- EMF не предоставя thread-safe достъп до моделите.
- Large memory footprint
- Генерира само Java код

Сравнителен анализ

Въпреки че Visual Paradigm е инструмент с коренно различна идея, избрах него за сравнителен анализ, тъй като поддържа голяма част от функционалностите на EMF, включително моделиране и генериране на код.

Генерираният код от EMF може да бъде само и единствено Java код, докато при Visual Paradigm например може да е на много езици като например Java, C#, C++, Python, PHP и много други. Също така Visual Paradigm има много по-широка интеграция със среди за разработка като Eclipse, NetBeans IDE, Intellij IDEA, Visual Studio, Android Studio.
Трябва да отбележим, че целта на EMF далеч не е тази на Visual Paradigm , а е един вид подмножество на целите на VP. Можем спокойно да кажем, че EMF опростява моделирането , но с цената на не малко наброй функционалности.

Сравнение на EMF и Visual Paradigm

Критерии	**EMF**	**Visual Paradigm**
IDE Integrations	Eclipse	Eclipse, IntelliJ, AndroidStudio, Visual Studio, NetBeans
Code generator language	Java	Java, C++, C#, Python, PHP и др.
Поддръжка	Средна	Силна
Последна версия	Май 2019	Юли 2020
Общност	Голяма	Голяма
Безплатен	Да	Да

Заключение

EMF е лесен за употреба, с прагматичен подход към мета-моделирането, зряла и доказана среда за моделиране, подпомагана с голяма сила от open-source общността. EMF не е UML и инструмент за моделиране. И все пак налага не малка доза ограничения породени не толкова от самия него, но повече от Eclipse платформата. Може би това е инструментът, който ни трябва, когато ясно и точно знаем какво искаме да постигнем.

Използвани литературни източници

1. Eclipse Modeling – EMF official documents
   https://www.eclipse.org/modeling/emf/docs/
2. EMF Tutorial
   https://www.vogella.com/tutorials/EclipseEMF/article.html