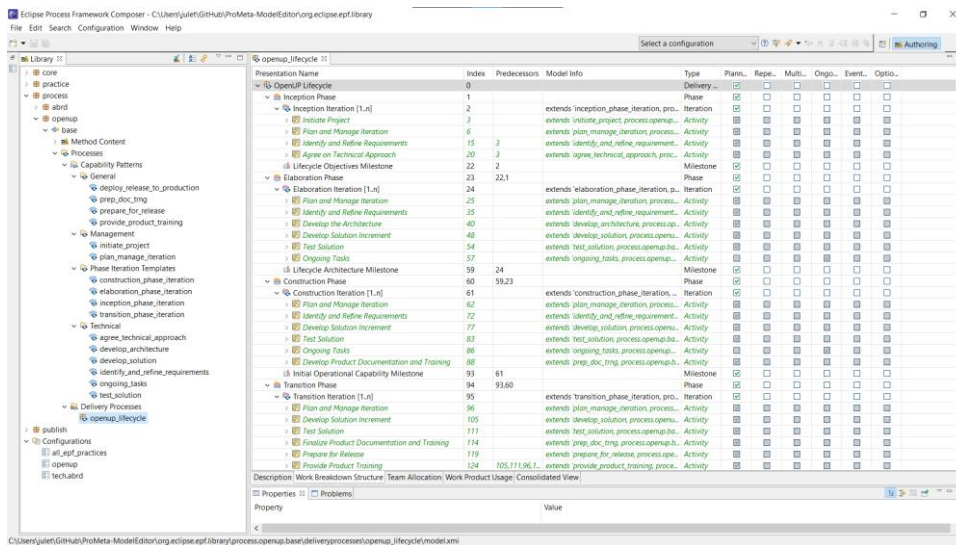


ModelEditor Dokumentazioa

ModelEditor azpisistemaren funtzionamendua dokumentatuko da hobeto uler dadin. Garapenerako Eclipse-ren tresnak erabili dira orokorrean. EPF Composer eta Eclipse IDE izan dira tresna nagusiak baina Visual Studio Code ere erabili da editore moduan. Programazio lengoaiak nagusiak Java eta Xtend izan dira.

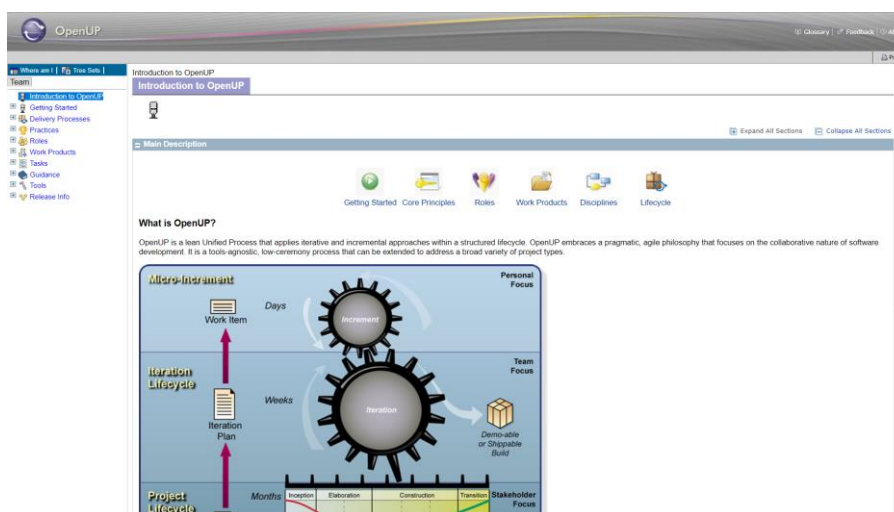
Hasteko, metodologia aukeratu behar dugu eta EPF Composer tresnan kargatu. Metodologiak XMI formatuan egon behar du eta UMA metaereduarekin bat etorri behar du. Metodologia propioa ere defini daiteke tresna horrekin existitzen den metodologia baten aldatetarik eginez. Proiektu honetarako EPF Practices liburutegi osoa erabiltzea erabaki da. Bertan OpenUP eta ABRD metodologiak daude, 1. Irudian ikus datekeen moduan. Metodologia gehiago ere deskargatu daitezke Eclipsen webgunetik, Scrum eta XP adibidez. Etorkizunean RUP metodologia definitzeko aukera ere badago.



The screenshot shows the Eclipse Process Framework Composer interface. The left sidebar displays a tree view of the project structure, including 'core', 'practice', 'process', 'abrd', 'openup', and 'base'. The main area shows the 'OpenUP Lifecycle' model with a table of elements. The table has columns for 'Presentation Name', 'Index', 'Predecessors', 'Model Info', 'Type', 'Planned', 'Repe.', 'Multi.', 'Ongo.', 'Event.', and 'Optio.'. The elements are organized into phases: Inception Phase, Elaboration Phase, Construction Phase, and Transition Phase. Each phase contains several activities and milestones, such as 'Initiate Project', 'Plan and Manage Iteration', 'Identify and Refine Requirements', 'Develop Solution Increment', 'Test Solution', 'Develop Product Documentation and Training', and 'Provide Product Training'.

1. Irudia. OpenUP bizi-zikloa EPF Composer tresnan.

Behin metodologia kargatuta dugula hainbat aukera ditugu. Esan bezala, EPF Composer tresna bera erabili daiteke metodologian aldatetarik egin nahi baditugu. Beste aukera garrantzitsu bat metodologiaren webgunea sortzea da, dokumentazio moduan oso erabilgarria dena. Adibidez, OpenUP metodologiaren webgunea ikus daiteke 2. Irudian.



2. Irudia. OpenUP metodologiaren webgunea.

Metodologia nahi dugun moduan dagoenean hurrengo pausura pasa gaitezke. Aurrerago ere edukiko dugu aldaketak egiteko aukera beharrezkoa bada. Arkitekturan ikus daitekeen moduan, hurrengo pausua XSLT erabiliz ereduari hainbat eraldaketa egitea da. Izan ere, dagoen bezala uzten badugu, ereduak ez dator bat UMA metaereduarekin eta errore pila bat ematen ditu. 3. Irudian agertzen den XSLT fitxategiak 600 lerro baino gehiago dituen arren, nahikoa da Java fitxategi bat exekutatzeko eta bigarren gutxi batzutan eraldatutako .uma fitxategiak lortzen ditugu.

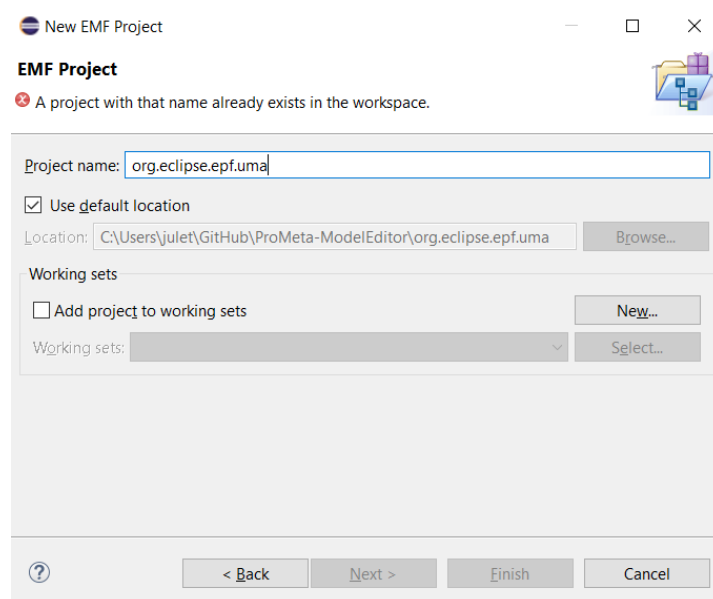
```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:org.eclipse.epf.uma="http://www.eclipse.org/epf/uma/1.0.6/uma.ecore" xmlns:org.eclipse.epf.uma.resourcemanager="http://org.eclipse.epf/uma/resourcemanager.ecore">
  <!--Identity-->
  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
  <!--Root-->
  <xsl:variable name="root" select="'/org.eclipse.epf.openup.uma/src/'"/>
  <!--ResourceManager-->
  <xsl:template match="org.eclipse.epf.uma.resourcemanager:ResourceManager">
    <!--MethodLibrary-->
    <xsl:template match="org.eclipse.epf.uma:MethodLibrary">
      <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
        <xsl:element name="predefinedConfigurations">
          <xsl:attribute name="href">
            <xsl:value-of select="'configurations/all_epf_practices.uma#_QD87wFRHed2CWscN8Mx6Grg'"/>
          </xsl:attribute>
        </xsl:element>
      </xsl:copy>
    </xsl:template>
  </xsl:template>

```

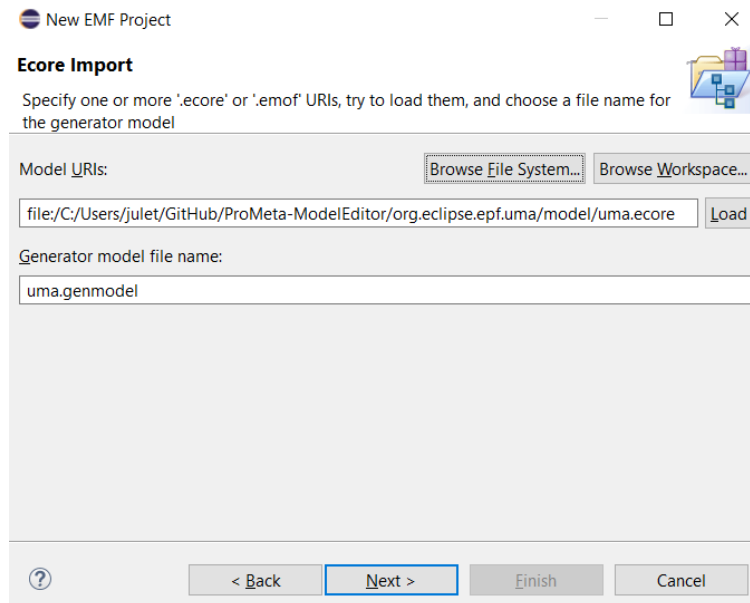
3. Irudia. Eraldaketa egiteko XSLT fitxategia.

Esan bezala, erabiliko dugun metaeredua UMA da, jadanik definituta dagoena. Beraz, lehenik metaeredu hori deskargatu behar da Eclipse-ren webgunetik. Metaereduarekin lan egiteko EMF proiektu bat sortuko dugu, 4. Irudian ikusten den bezala.



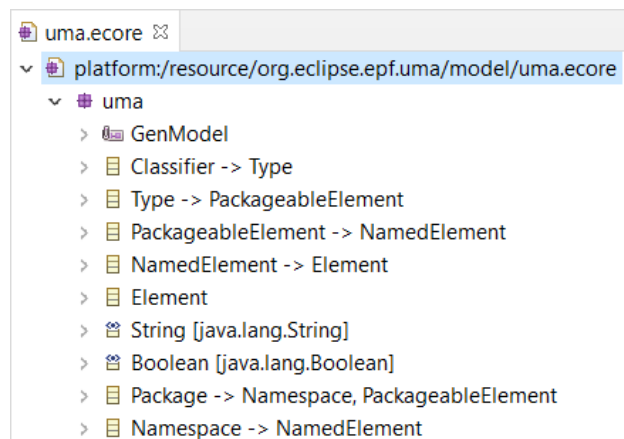
4. Irudia. EMF proiektuaren sorrera Eclipsen.

Proiektuaren izena aukeratu ondoren inportatzeko *Ecore model* aukeratuko dugu eta 5. Irudian dagoen leihora eramango gaitu. Bertan `uma.ecore` metaeredua aukeratu beharko dugu eta *Next* botoia sakatu. Hurrengo lehioan *Finish* sakatzen badugu proiektua sortuko da.



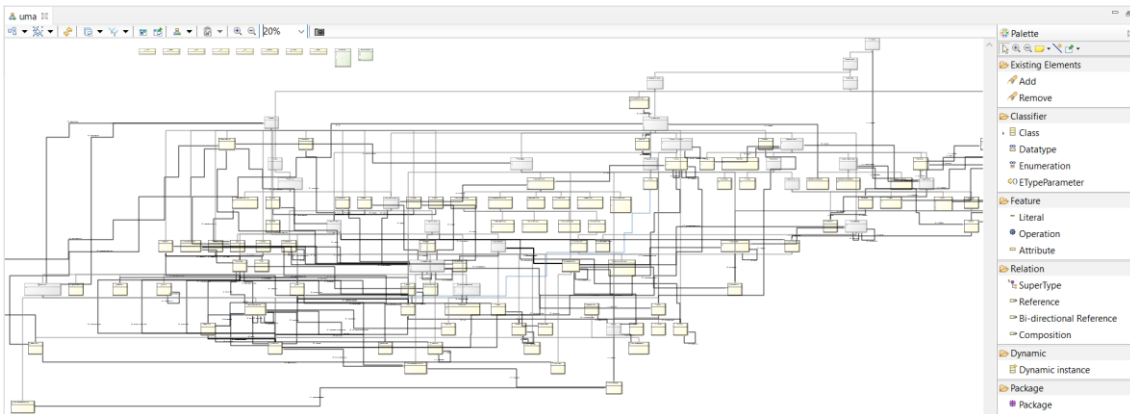
5. Irudia. Uma metaeredua inportatzen Eclipsen.

Proiektu horrek `uma.ecore` eta `uma.genmodel` fitxategiak izango ditu. Fitxategi horiek *Sample Reflective Ecore Model Editor* erabiliz ireki eta editatu daitezke. Adibidez, `uma.ecore` fitxategiaren zati bat ikus daiteke 6. Irudian.



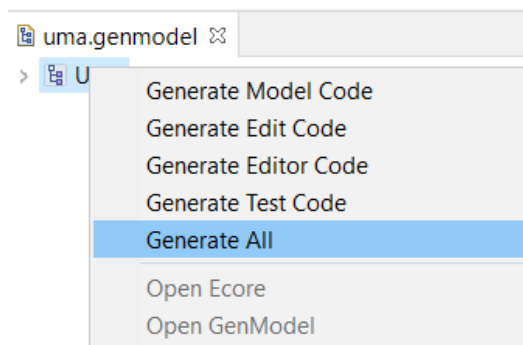
6. Irudia. Uma.ecore fitxategia Eclipsen.

Hurrengo pausua `uma.aird` fitxategia sortzea izango da, metaereduaren diagrama bat sortzeko klase eta atributu guztiekin. Horretarako, `uma.ecore` fitxategian *Initialize Ecore Diagram* aukeratuko dugu eta ondoren *Entities in a Class Diagram*. Diagramaren izena aukeratu eta *Finish* botoia sakatutakoan klase diagrama sortuko da. 7. Irudian ikus daitekeen moduan, metaeredua osoa handia da eta klase pila bat daude. Aldaketaren bat egitea nahiko bagenu eskuineko paleta erabili beharko genuke.



7. Irudia. Uma metaereduaren klase diagrama.

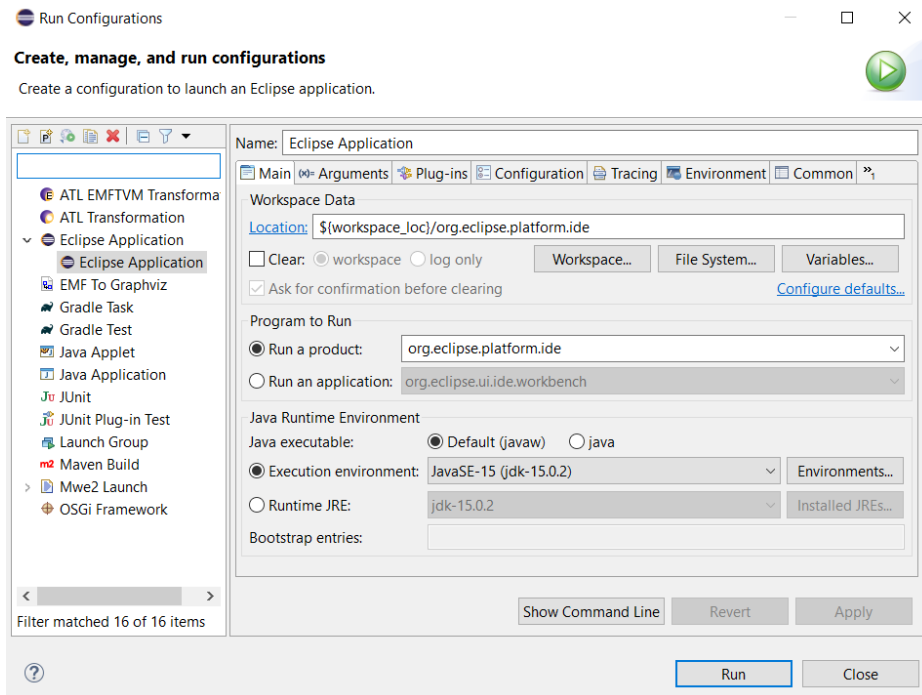
Behin metaereduaren proiektua prest daukagula, editore grafikoa sor dezakegu. Horretarako, `uma.genmodel` fitxategia ireki beharko dugu. Kodea sortzeko hainbat aukera daude, baina sinpleena *Generate All* aukeratzeta da, 8. Irudian ikusten den moduan. Izan ere, lehenengo 3 aukerak beharrezkoak dira editorea sortzeko eta 4. aukera probentzako da.



8. Irudia. Editore grafikoen kodearen sorrera.

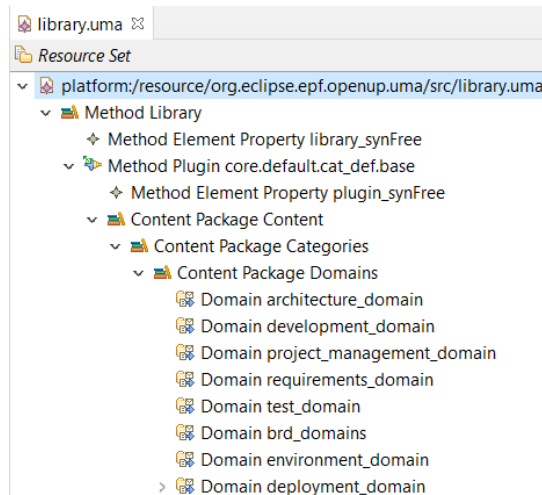
Lehenengo aukerak proiektuan bertan sortuko du ereduaren Java kodea. Beste aukera bakoitzerako Java proiektu berri bat sortuko da automatikoki. Honekin jadanik editore grafikoa daukagu, baina itxura hobea edukitzea nahi badugu ikonoak gehitu beharko ditugu `org.eclipse.uma.edit` proiektuan.

Dena prest daukagunean editorea martxan jar dezakegu *Run As > Eclipse Application* aukeratzuz. Irekitzen dugun lehenengo aldia bada, konfiguratu egin beharko dugu. Adibidez, 9. Irudian agertzen den konfigurazioa aukera daiteke.



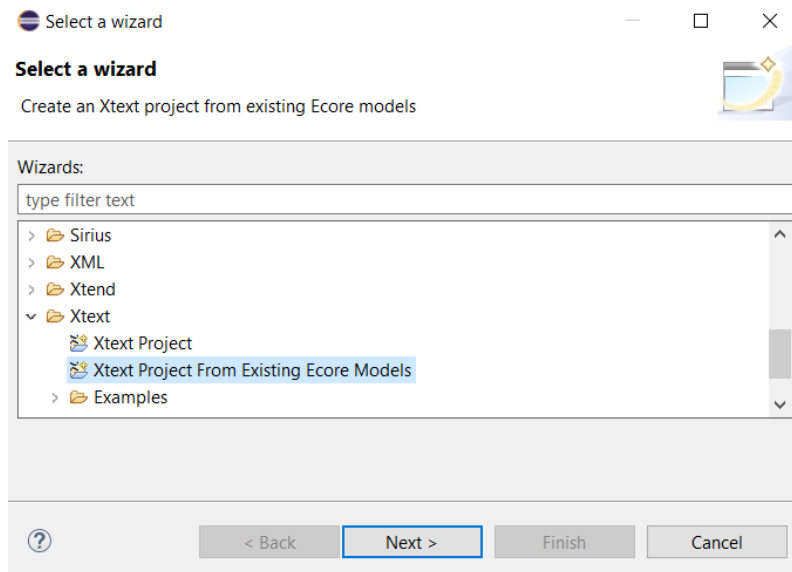
9. Irudia. Editore grafikoa martxan jartzeko konfigurazioa.

Martxan jartzen dugunean Eclipse aplikazio berri bat irekiko da. Bertan proiektu berri bat sortu beharko dugu eta XSLT eraldaketatik lortutako .uma fitxategiak kargatu. Edozein .uma fitxategi irekitzeko aukera izango dugu *Uma Model Editor* erabiliz. Editore honek EPF Composer-en antza du ikonoak eta zuhaitz egitura partekatzen dituelako. Adibidez, `library.uma` fitxategia irekitzen badugu metodologiaren eduki osoa bistaratuko da, 10. Irudian ikus daitekeen bezala.



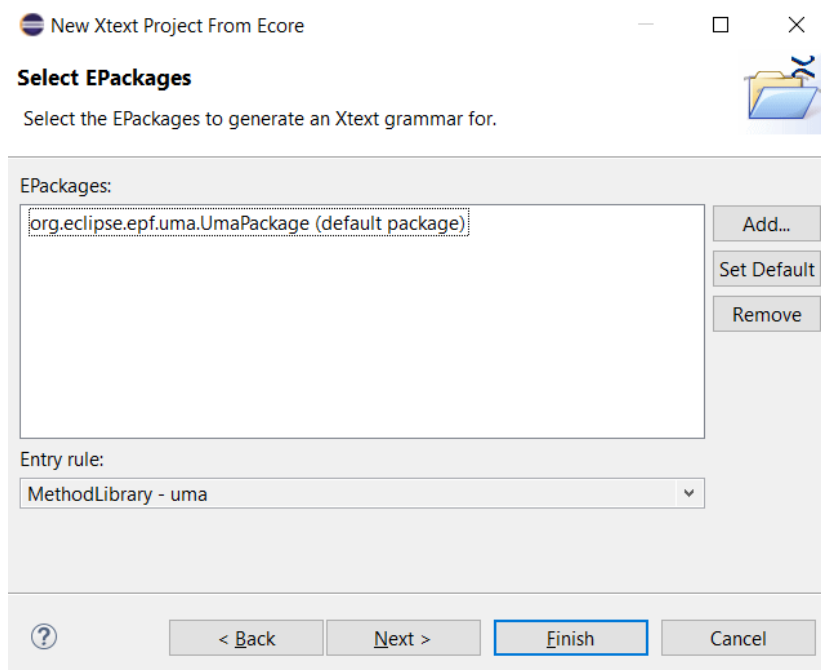
10. Irudia. Editore grafikoarekin library.uma fitxategia bistartzen.

Editore grafikoa bukatuta dagoenez, testu editorearekin has gaitezke. Testu editorea sortzeko Xtext tresna erabiliko dugu. Helburua DSL bat sortzea denez, gramatika bat definitu beharko dugu. Metaeredua erabil dezakegu gramatika automatikoki sortzeko, 11. Irudian agertzen den *Xtext Project From Existing Ecore Models* aukera erabilita.



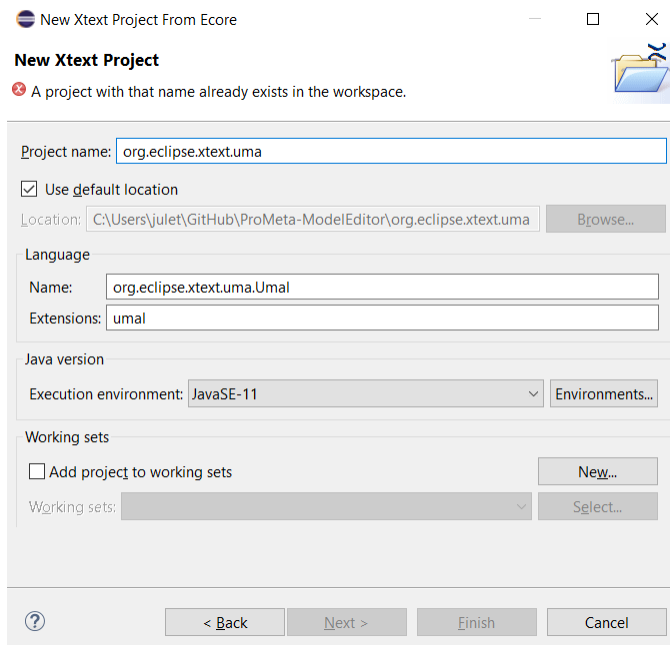
11. Irudia. Xtext proiektua sortzen Eclipsen.

Hurrengo lehioan *Add* botoia sakatu eta `uma.genmodel` fitxategia aukeratu dugu, 12. Irudian ikus daitekeen bezala. Beheko aukeretan metaereduaren erroa aukeratu behar da, kasu honetan *MethodLibrary* dena.








12. Irudia. Uma metaeredutik Xtext proiektua sortzen.

Hurrengo leihoan proiektuaren izena, lengoariaren izena eta fitxategien luzapena aukeratu beharko ditugu, 13. Irudian ikusten den moduan.



13. Irudia. Xtext proiektuaren eta lengoaiaren izenak aukeratzeko.

Amaitutakoan 14. Irudian agertzen diren 5 Xtext proiektuak sortuko dira. Lehenengoa proiektu nagusia da, gramatika eta fitxategi garrantzitsuenak izango dituen. Beste bi proiektuek editorearen kodea dute eta probarako bi proiektu ere badaude.

- >  org.eclipse.xtext.uma
- >  org.eclipse.xtext.uma.ide
- >  org.eclipse.xtext.uma.tests
- >  org.eclipse.xtext.uma.ui
- >  org.eclipse.xtext.uma.ui.tests

14. Irudia. Testu editorearen proiektuak Eclipsen.

Gramatika lehenengo proiektuan dago eta Umal.xtext izena du. Metaeredua oso handia denez, sortutako gramatikak 2600 lerro inguru ditu eta errore batzuekin sortzen da. Adibidez, 15. Irudian gramatikaren erroaren zatia ikus daiteke.

```

1 // automatically generated by Xtext
2 grammar org.eclipse.xtext.uma.Umal with org.eclipse.xtext.common.Terminals
3
4 import "http://www.eclipse.org/epf/uma/1.0.6/uma.ecore"
5 import "http://www.eclipse.org/emf/2002/Ecore" as.ecore
6
7 MethodLibrary returns MethodLibrary:
8   'MethodLibrary'
9   guid=String0
10   '{'
11     ('name' name=String0)?
12     ('presentationName' presentationName=String0)?
13     ('briefDescription' briefDescription=String0)?
14     ('suppressed' suppressed=Boolean)
15     ('orderingGuide' orderingGuide=String0)?
16     ('authors' authors=String0)?
17     ('changeDate' changeDate=Date)?
18     ('changeDescription' changeDescription=String0)?
19     ('version' version=String0)?
20     ('kind' (' kind=[Kind] ( "," kind=[Kind])* ' ' )? )?
21     ('copyrightStatement' copyrightStatement=[SupportingMaterial])?
22     ('ownedRules' (' ownedRules+=Constraint ( "," ownedRules+=Constraint)* ' ' )? )?
23     ('methodElementProperty' (' methodElementProperty+=MethodElementProperty ( "," methodElementProperty+=MethodElementProperty)* ' ' )? )?
24     ('methodPlugins' (' methodPlugins+=MethodPlugin ( "," methodPlugins+=MethodPlugin)* ' ' )? )?
25     ('predefinedConfigurations' (' predefinedConfigurations+=MethodConfiguration ( "," predefinedConfigurations+=MethodConfiguration)* ' ' )? )?
26   '}'

```

15. Irudia. Xtext proiektuko lengoaiaren gramatika.

Behin gramatika konpondutakoan `GenerateUmal.mwe2` fitxategia *Rus As > MWE2 Workflow* aukeratu beharko dugu editorearen kodea sortzeko. Gainerako proiektuetan editorearen Java eta Xtend kodea sortuko da.

Editorea exekutatzeko modua editore grafikoaren berdina da eta konfigurazio bera aukeratzen badugu, bi editoreak batera erabil ahal izango ditugu. Eredu osoa erabili ordez, adibide simple batekin hobeto ikus ditzakegu bi editoreak 16. Irudian.



16. Irudia. Editore grafikoaren ezkerrean eta testu editorea eskuinean.

Hala ere, oraindik aspektu garrantzitsu bat falta da, editoreak ez baitira elkarren artean sinkronizatzen. Helburua da bietako bat aldatzen badugu automatikoki segundu gutxi batzutan bestearekin sinkronizatzea. Horrela, nahiz eta bi fitxategi desberdin izan, edozein momentutan alda daiteke editorez. Hau garrantzitsua da editore bakoitzak bere abantailak dituelako. Editore grafikoak ereduak modu bisualean ikustea ahalbidetzen du eta testu editoreak, berriz, aldaketa errepikakorrak azkarrago egitea.

Beraz, bi itzultzaile behar ditugu, uma-tik umal-era itzultzen duena eta umal-etik uma-ra itzultzen duena. Bigarren itzulketa nahiko sinplea da, kode sorreraren antzekoa baita. Izan ere, Xtend-ek itzulketa hau errazten du serializatzaileak definituz Xtend programazio lengoia erabiliz. Lehenengoa zailagoa da, Xtend-ek uma fitxategien lengoia detektatu behar baitu, eta umal lengoariaren antzeko tratamendua eman. Horretarako, Xtend-ek umal-erako automatikoki sortu dituen fitxategietako batzuk eskuz sortu behar dira uma-rako.

Bukatzeko, aldaketak dauden bakoitzean metodologiaren informazio garrantzitsuena datu-base batean gordetzeko balio duen SQL fitxategia eguneratzen da. Datu-basearen tamainaren ideia bat egiteko, sortutako fitxategiak 9400 lerro 1100 INSERT inguru dauzka. Argi dago, beraz, prozesu hau automatizatzearen erabilgarritasuna, informazio hori guztia eskuz gehitzeko denbora pila bat beharko genukeelako. 17. Irudian sortutako kodearen lehenengo 3 INSERT aginduak ikus daitezke.



17. Irudia. Datu-basearen datuak dituen SQL fitxategia.

Editatzea bukatutakoan, fitxategi hori phpMyAdmin-etik inportatuko dugu, datu-basean metodologiaren informazio guztia gordetzeko. 18. Irudian metodologiaren datu-baseko taulak agertzen dira phpMyAdmin-en.

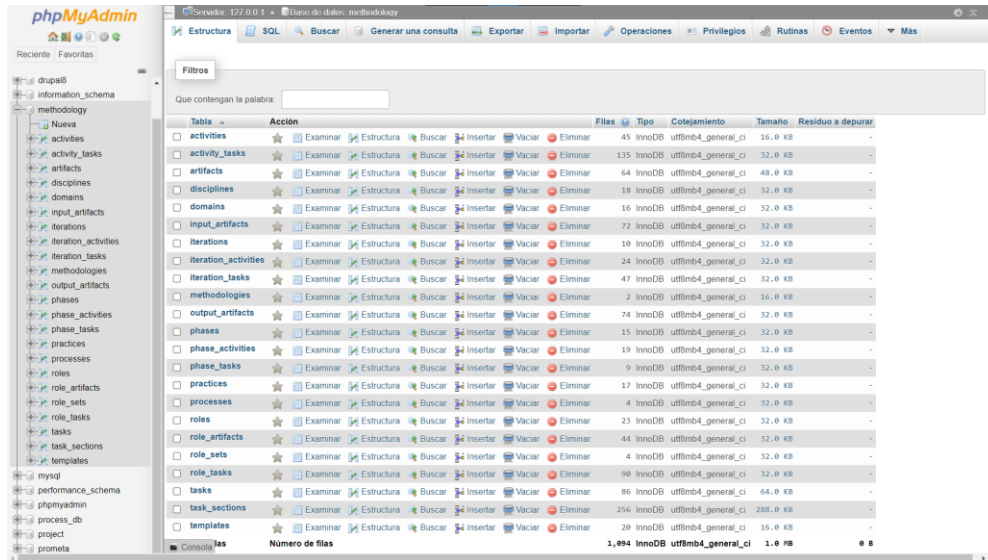


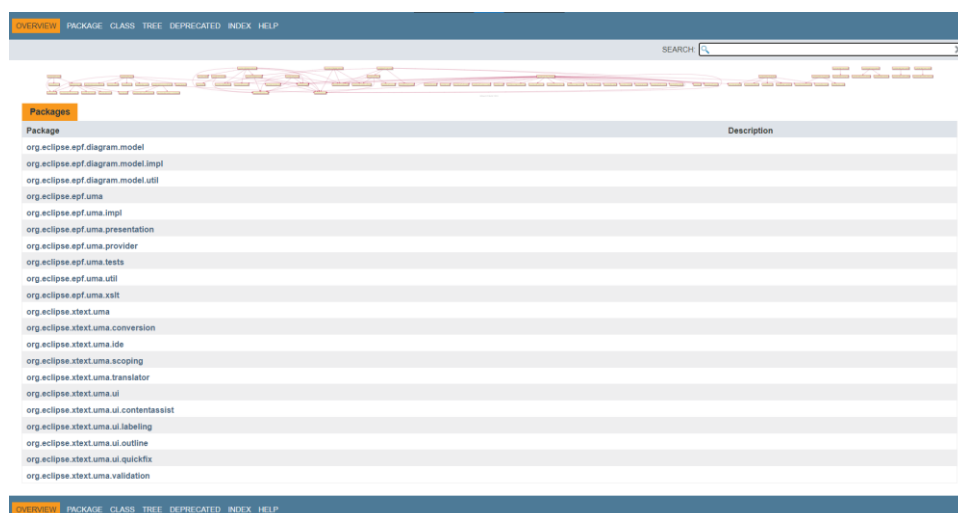
Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
activities	Examinar Estructura Buscar Insertar Vaciar Eliminar	45	InnoDB	utf8mb4_general_ci	16.0 KB	-
activity_tasks	Examinar Estructura Buscar Insertar Vaciar Eliminar	135	InnoDB	utf8mb4_general_ci	32.0 KB	-
artifacts	Examinar Estructura Buscar Insertar Vaciar Eliminar	64	InnoDB	utf8mb4_general_ci	48.0 KB	-
disciplines	Examinar Estructura Buscar Insertar Vaciar Eliminar	18	InnoDB	utf8mb4_general_ci	32.0 KB	-
domains	Examinar Estructura Buscar Insertar Vaciar Eliminar	16	InnoDB	utf8mb4_general_ci	32.0 KB	-
input_artifacts	Examinar Estructura Buscar Insertar Vaciar Eliminar	72	InnoDB	utf8mb4_general_ci	32.0 KB	-
iterations	Examinar Estructura Buscar Insertar Vaciar Eliminar	10	InnoDB	utf8mb4_general_ci	32.0 KB	-
iteration_activities	Examinar Estructura Buscar Insertar Vaciar Eliminar	24	InnoDB	utf8mb4_general_ci	32.0 KB	-
iteration_tasks	Examinar Estructura Buscar Insertar Vaciar Eliminar	47	InnoDB	utf8mb4_general_ci	32.0 KB	-
methodologies	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	16.0 KB	-
output_artifacts	Examinar Estructura Buscar Insertar Vaciar Eliminar	74	InnoDB	utf8mb4_general_ci	32.0 KB	-
phases	Examinar Estructura Buscar Insertar Vaciar Eliminar	15	InnoDB	utf8mb4_general_ci	32.0 KB	-
phase_activities	Examinar Estructura Buscar Insertar Vaciar Eliminar	19	InnoDB	utf8mb4_general_ci	32.0 KB	-
phase_tasks	Examinar Estructura Buscar Insertar Vaciar Eliminar	9	InnoDB	utf8mb4_general_ci	32.0 KB	-
practices	Examinar Estructura Buscar Insertar Vaciar Eliminar	17	InnoDB	utf8mb4_general_ci	32.0 KB	-
processes	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	32.0 KB	-
roles	Examinar Estructura Buscar Insertar Vaciar Eliminar	23	InnoDB	utf8mb4_general_ci	32.0 KB	-
role_artifacts	Examinar Estructura Buscar Insertar Vaciar Eliminar	44	InnoDB	utf8mb4_general_ci	32.0 KB	-
role_sets	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	32.0 KB	-
role_tasks	Examinar Estructura Buscar Insertar Vaciar Eliminar	90	InnoDB	utf8mb4_general_ci	32.0 KB	-
tasks	Examinar Estructura Buscar Insertar Vaciar Eliminar	86	InnoDB	utf8mb4_general_ci	64.0 KB	-
task_sections	Examinar Estructura Buscar Insertar Vaciar Eliminar	254	InnoDB	utf8mb4_general_ci	288.0 KB	-
templates	Examinar Estructura Buscar Insertar Vaciar Eliminar	20	InnoDB	utf8mb4_general_ci	16.0 KB	-
roles	Examinar Estructura Buscar Insertar Vaciar Eliminar	1,094	InnoDB	utf8mb4_general_ci	1.0 MB	0 B

18. Irudia. Metodologiaren datu-basea phpMyAdmin-en.

Kodea gordetzeko eta bertsio kontrolerako Git eta GitHub tresnak erabili dira. Dokumentazio webgunearen hosting-erako GitHub Pages erabili da. Java proiektuen dokumentazioa sortzeko modu ohikoa Javadoc erabiltzea da. Kasu honetan, plugin bat gehitu da, dokumentazioarekin batera diagramak automatikoki sortzeko.

UMLDoclet tresnak Javadoc metadatuak erabiltzen ditu PlantUML diagramak automatikoki sortzeko eta HTML dokumentazioan txertatzeko. Diagramak irudi klikagarri gisa txertatzen dira eta pakete eta klaseko dokumentazioa estekatzen dira eskuragarri dagoenean. Pakete dependentzia, pakete eta klase diagramak sortzen ditu elementu guztietarako.

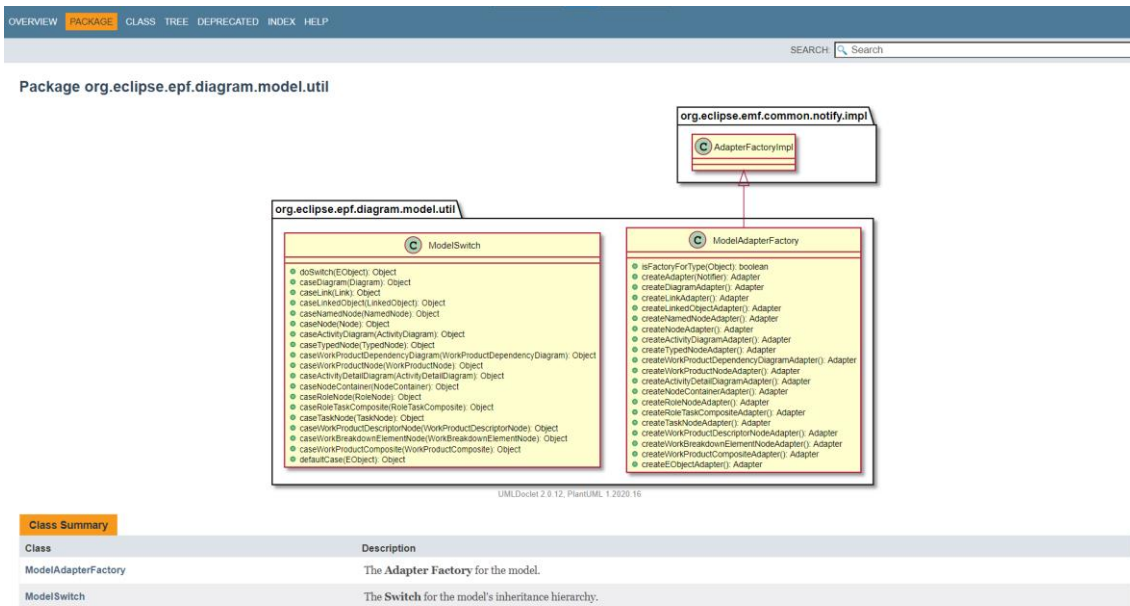
Dokumentazio webguneak Javadoc webgune baten itxura du, baina hainbat lekutan diagramak daude. 19. Irudian ModelEditor azpisistemako dokumentazio webgunearen hasierako orria agertzen da. Bertan paketeak zerrendatzen dira eta goiko aldean paketeen arteko dependentzia diagrama ikus daiteke.



Package	Description
org.eclipse.epf.diagram.model	
org.eclipse.epf.diagram.model.impl	
org.eclipse.epf.diagram.model.util	
org.eclipse.epf.uma	
org.eclipse.epf.uma.impl	
org.eclipse.epf.uma.presentation	
org.eclipse.epf.uma.provider	
org.eclipse.epf.uma.tests	
org.eclipse.epf.uma.util	
org.eclipse.epf.uma.xslt	
org.eclipse.xtext.uma	
org.eclipse.xtext.uma.conversion	
org.eclipse.xtext.uma.ide	
org.eclipse.xtext.uma.scoping	
org.eclipse.xtext.uma.translator	
org.eclipse.xtext.uma.ui	
org.eclipse.xtext.uma.ui.contentassist	
org.eclipse.xtext.uma.ui.labeling	
org.eclipse.xtext.uma.ui.outline	
org.eclipse.xtext.uma.ui.quickfix	
org.eclipse.xtext.uma.validation	

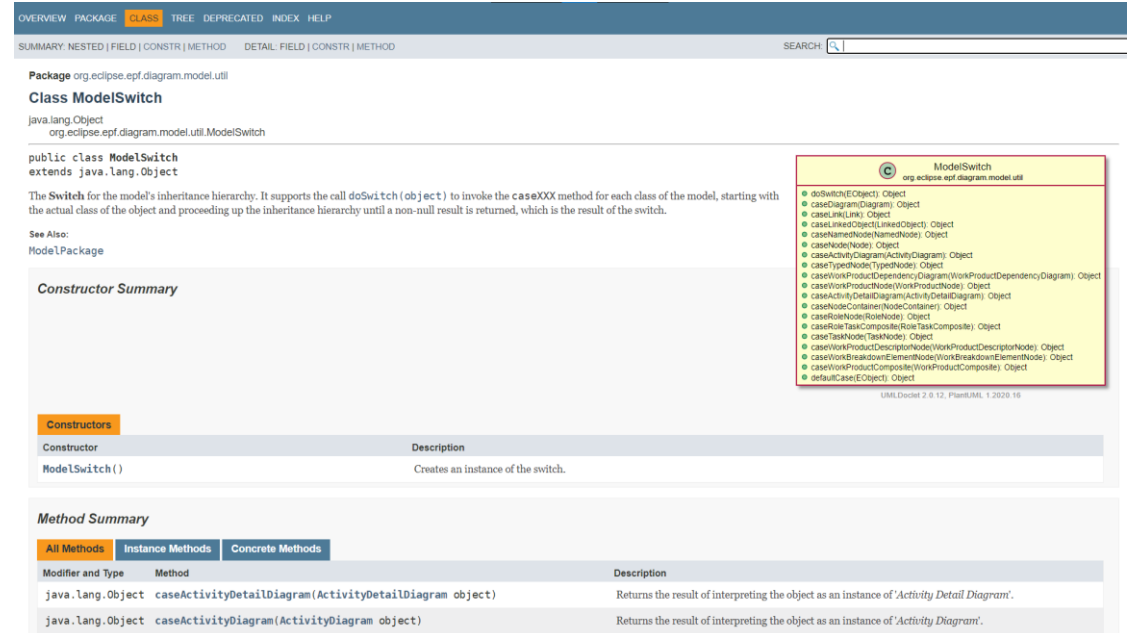
19. Irudia. ModelEditor-en paketeak eta paketeen arteko dependentziak.

Pakete bat aukeratzeko badugu, diagrama eta klaseak bistaratu dira. 20. Irudian adibide bat ikus daiteke.



20. Irudia. Pakete diagrama adibidea.

Klase bat aukeratzeko badugu, bere diagrama eta metodoak bistaratu dira. 21. Irudian ikus daiteke klase diagrama baten adibidea.



21. Irudia. Klase diagrama adibidea.