2. Gyakorlat: GMF

Simon Balázs, BME IIT, 2013.

1 Feladat

Készítsünk grafikus GMF editort a context diagramok-hoz (CD) és data-flow diagramok-hoz (DFD)!

Lépések:

1. Olvassuk el a use-case GMF editor elkészítésére vonatkozó GMF tutorial-t (2. fejezet)!
2. Készítsük el a 3. fejezet útmutatói alapján a CD és DFD diagramokhoz tartozó grafikus editort!

2 GMF Tutorial

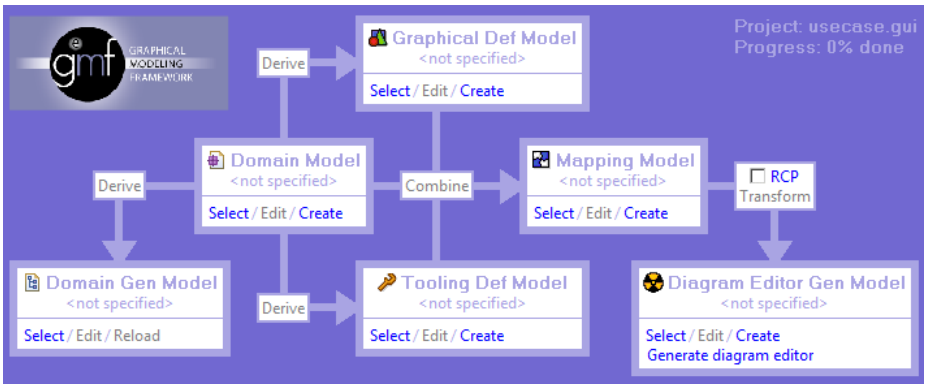
2.1 Feladat

Készítsünk grafikus modellező nyelvet a use-case EMF modellhez!

2.2 Megoldás

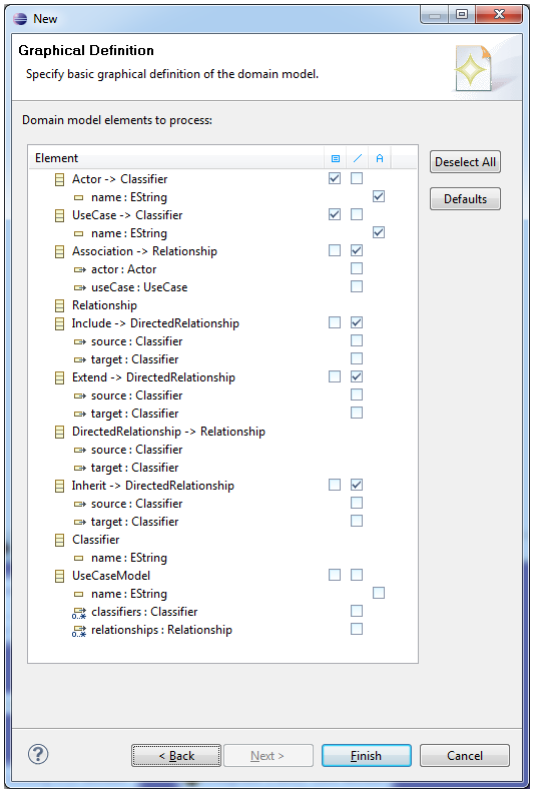
(Megjegyzés: egy másik példán keresztül az Eclipse-ben lévő GMF cheat sheet is bemutatja a GMF használatát. A cheat sheet előhozásához válasszuk a **Help > Cheat sheets…** menüpontot, azon belül pedig a **Graphical Modeling Framework > GMF Tutorial** elemet. Ez a cheat sheet a jelen use-case tutorial-hoz nem szükséges.)

1. **File > New > Project…**
   1.1. **Graphical Modeling Framework > Graphical editor project**
   1.2. **Project name:** usecase.gui
   1.3. **Next**
   1.4. Pipáljuk ki a **Show dashboard view for the created project** opciót
   1.5. **Finish**
2. A következő dashboard jelenik meg:



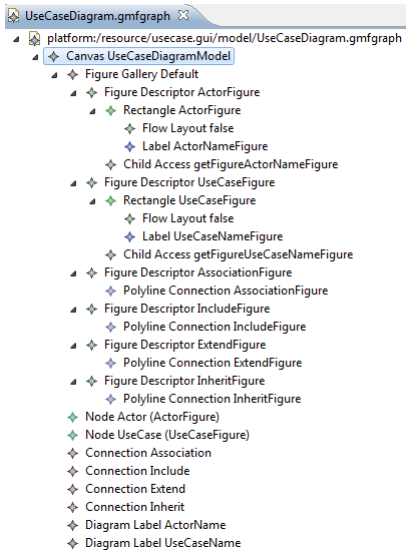
3. A **usecase.gui** projekten jobbgombbal kattintva hozzuk elő a **Properties** ablakot, és állítsuk be a **UseCaseDiagram** projektet, mint függőséget a **Java Build Path** opciónál a **Projects** fülön!
4. A **Domain Model** és a **Domain Gen Model** már készen van, ezeket az **EMF Tutorial**-ban már elkészítettük. Ezeket csak hozzá kell rendelni az aktuális projekthez:
   4.1. A **GMF dashboard**-on a **Domain model** alatt válasszuk a **Select** linket és adjuk meg a **UseCaseDiagram.ecore**-t!
   4.2. A **Domain Gen Model** alatt is válasszuk a **Select** linket és adjuk meg a **UseCaseDiagram.genmodel**-t!
5. A következő lépés annak megadása, hogy az egyes EMF objektumok milyen alakzatként jelennek majd meg az ábrán.
6. Kattintsunk a **Domain Model** és a **Graphical Def Model** közötti **Derive** linkre!
   6.1. **Parent folder:** usecase.gui/model
   6.2. **File name:** UseCaseDiagram.gmfgraph
   6.3. **Next**
   6.4. **Diagram element:** UseCaseModel
   6.5. **Next**

6.6. Ellenőrizzük le, hogy a wizard a megfelelő típusú grafikus elemeket választotta-e:



6.7. **Finish**

7. A grafikus leírás a következő fahierarchiaként épül fel:



8. Kezdjük az asszociációkkal. Ezek mindegyike polyline-ként van definiálva. A következő módosításokra van szükség:
   8.1. **AssociationFigure:** ezen nem kell változtatni
   8.2. **IncludeFigure:** a cél oldalra nyilat kell tenni
      8.2.1. Kattintsunk a **Polyline Connection IncludeFigure**-ra jobb gombbal, és adjunk hozzá egy új **Polyline Decoration** gyereket! A **Polyline Decoration** alapértelmezettként két vonalból álló nyílhegyet jelent.
      8.2.2. Állítsuk be a **Polyline Decoration Name** tulajdonságát **IncludeOpenArrow**-ra a **Properties** ablakban!

8.2.3. Állítsuk be a **Polyline Connection IncludeFigure Target Decoration** tulajdonságát **Polyline Decoration IncludeOpenArrow**-ra!

8.3. **ExtendFigure:** a cél oldalra nyilat kell tenni

8.3.1. Kattintsunk a **Polyline Connection ExtendFigure**-ra jobb gombbal, és adjunk hozzá egy új **Polyline Decoration** gyereket!

8.3.2. Állítsuk be a **Polyline Decoration Name** tulajdonságát **ExtendOpenArrow**-ra a **Properties** ablakban!

8.3.3. Állítsuk be a **Polyline Connection ExtendFigure Target Decoration** tulajdonságát **Polyline Decoration ExtendOpenArrow**-ra!

8.4. **InheritFigure:** a cél oldalra háromszög fejű nyilat kell tenni

8.4.1. Kattintsunk a **Polyline Connection InheritFigure**-ra jobb gombbal, és adjunk hozzá egy új **Polygon Decoration** gyereket! A **Polygon Declaration** alapértelmezettként kitöltött háromszög fejű nyilat jelent.

8.4.2. Állítsuk be a **Polygon Decoration Name** tulajdonságát **InheritArrow**-ra, a **Filled** tulajdonságát pedig **false**-ra!

8.4.3. Állítsuk be a **Polyline Connection InheritFigure Target Decoration** tulajdonságát **Polygon Decoration InheritArrow**-ra!

9. Ezzel az asszociáció típusú elemek elkészültek. Következzenek a classifier-ek!

9.1. **UseCaseFigure**: alapértelmezettként téglalap van hozzárendelve, de nekünk ellipszis kell

9.1.1. Kattintsunk jobb gombbal a **Rectangle UseCaseFigure** elemen, és a **Delete** menüponttal töröljük!

9.1.2. Kattintsunk jobb gombbal a **Figure Descriptor UseCaseFigure** elemen, és adjunk hozzá egy **Ellipse** gyereket! Állítsuk be az **Ellipse Name** tulajdonságát **UseCaseFigure**-re!

9.1.3. Kattintsunk jobb gombbal az **Ellipse**-en és adjunk hozzá egy **Flow Layout** gyereket!

9.1.4. Kattintsunk jobb gombbal az **Ellipse**-en és adjunk hozzá egy **Label** gyereket! A **Label Name** tulajdonságát állítsuk **UseCaseNameFigure**-re, **Text** tulajdonságát pedig **<…>**-re!

9.1.5. A **ChildAccess Accessor** tulajdonságát állítsuk **getFigureUseCaseNameFigure**-ra, a **Figure** tulajdonságot pedig **Label UseCaseNameFigure**-ra!

9.2. **Actor:** sajnos pálcikaember nincs a beépített figurák között, így saját figurát kell készíteni

9.2.1. Töröljük a **Figure Descriptor ActorFigure Rectangle ActorFigure** gyerekét!

9.2.2. Adjunk hozzá a **Figure Descriptor ActorFigure**-hoz egy **Custom Figure** gyereket, állítsuk be a **Name** property értékét **ActorFigure**-re, a **Qualified Class Name**-et pedig **usecase.gui.shapes.ActorShape**-re!

9.2.3. Kattintsunk jobb gombbal az **Custom Figure ActorFigure**-on és adjunk hozzá egy **Flow Layout** gyereket!

9.2.4. Kattintsunk jobb gombbal az **Custom Figure ActorFigure**-on és adjunk hozzá egy **Label** gyereket! A **Label Name** tulajdonságát állítsuk **ActorNameFigure**-re, **Text** tulajdonságát pedig **<…>**-re!

9.2.5. A **ChildAccess Accessor** tulajdonságát állítsuk **getFigureActorNameFigure**-ra, a **Figure** tulajdonságot pedig **Label ActorNameFigure**-ra!

9.2.6. Adjuk hozzá a projekthez az alábbi külső **jar**-okat, mint referenciát, amelyek az eclipse plugins könyvtárában találhatók valamilyen verziószámmal ellátva:

9.2.6.1. org.eclipse.draw2d.jar

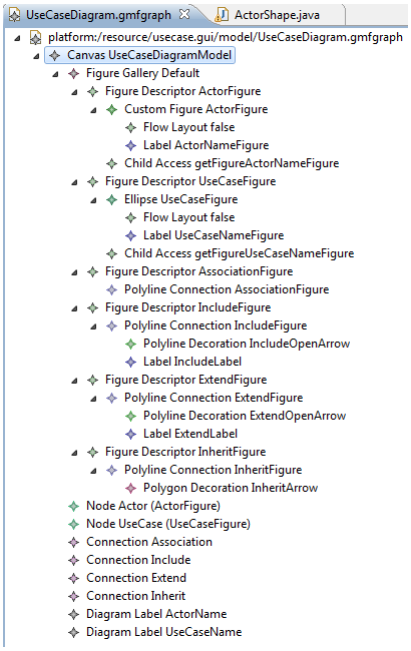
9.2.6.2. org.eclipse.swt.jar

9.2.6.3. org.eclipse.swt.win32.win32.x86.jar

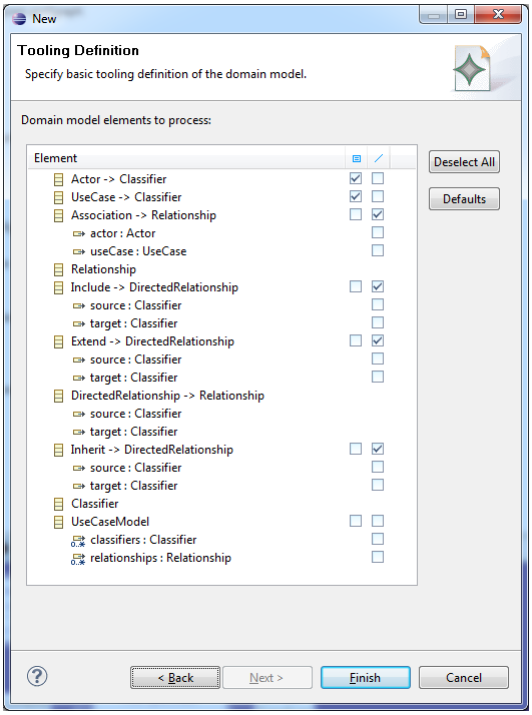
9.2.7. Készísünk a projekt **src** könyvtárában egy **ActorShape** nevű osztályt a **usecase.gui.shapes** csomagban:

```
package usecase.gui.shapes;

import org.eclipse.draw2d.Graphics;
import org.eclipse.draw2d.Shape;
import org.eclipse.draw2d.geometry.Rectangle;

public class ActorShape extends Shape
{
    @Override
    protected void fillShape(Graphics g) {
        Rectangle r = getBounds();
        int centerX = r.x + r.width / 2;
        int centerY = r.y + r.height / 2;

        // Head:
        Rectangle head = new Rectangle((int)(centerX-r.width*0.2), r.y,
                                       (int)(r.width*0.4), (int)(r.height*0.4));
        g.fillOval(head);
    }

    @Override
    protected void outlineShape(Graphics g) {
        Rectangle r = getBounds();
        int centerX = r.x + r.width / 2;
        int centerY = r.y + r.height / 2;

        // Head:
        Rectangle head = new Rectangle((int)(centerX-r.width*0.2), r.y,
                                       (int)(r.width*0.4), (int)(r.height*0.4));
        g.drawOval(head);
        // Body:
        g.drawLine(centerX, (int)(r.y+r.height*0.4), centerX, (int)(r.y+r.height*0.7));
        // Legs:
        g.drawLine(centerX, (int)(r.y+r.height*0.7), r.x, r.y+r.height);
        g.drawLine(centerX, (int)(r.y+r.height*0.7), r.x+r.width, r.y+r.height);
        // Arms:
        g.drawLine(centerX, (int)(r.y+r.height*0.5), r.x, (int)(r.y+r.height*0.4));
        g.drawLine(centerX, (int)(r.y+r.height*0.5), r.x+r.width, (int)(r.y+r.height*0.4));
    }
}
```

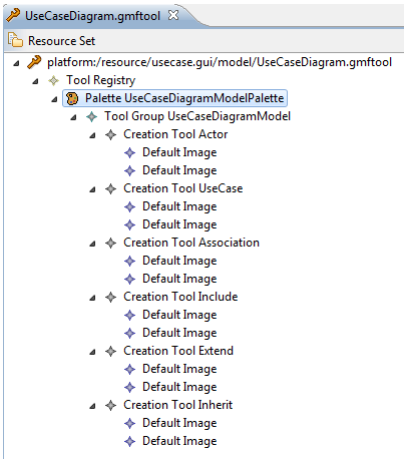
10. Ezzel minden grafikus objektumot sikerült definiálnunk. A fa így néz ki:



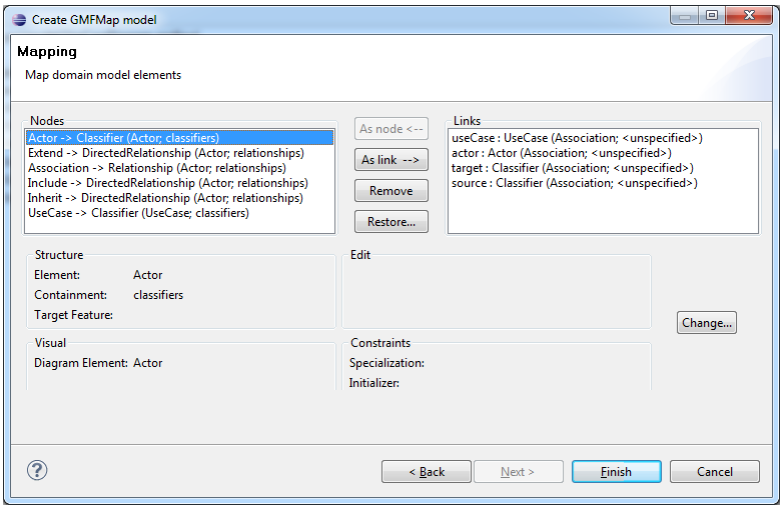
11. A következő lépés, hogy megadjuk azt, hogy a rajzoló eszköztáron milyen ikonok jelennek meg.
12. Kattintsunk a **GMF Dashboard**-on a **Domain Model** és a **Tooling Def Model** közötti Derive linkre!
    12.1. Parent folder: usecase.gui/model
    12.2. File name: UseCaseDiagram.gmftool
    12.3. Next
    12.4. Diagram element: UseCaseModel
    12.5. Next
13. Ellenőrizzük le, hogy a wizard a megfelelő típusú elemeket választotta-e:



14. A **UseCaseDiagram.gmftool** fában a következő elemek láthatók:

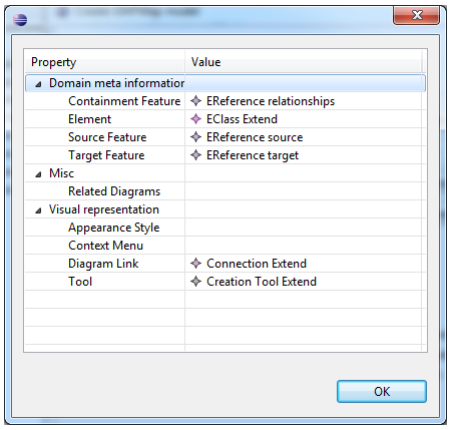


15. Hagyjuk meg az alapértelmezett beállításokat! Amennyiben kívánjuk, a **Default Image**-eket kell lecserélni más képekre.
16. A következő lépés az eddig elkészült dolgok összerendelése egymással. Kattintsunk a **GMF Dashboard**-on a **Mapping Model**-től közvetlenül balra lévő **Combine** linkre!
    16.1. **Parent folder**: usecase.gui/model
    16.2. **File name**: UseCaseDiagram.gmfmap
    16.3. **Next**
    16.4. **Select Domain Model, Class**: UseCaseModel
    16.5. **Next**
    16.6. **Next**
    16.7. Ebben a lépésben a következő ablak jelenik meg:

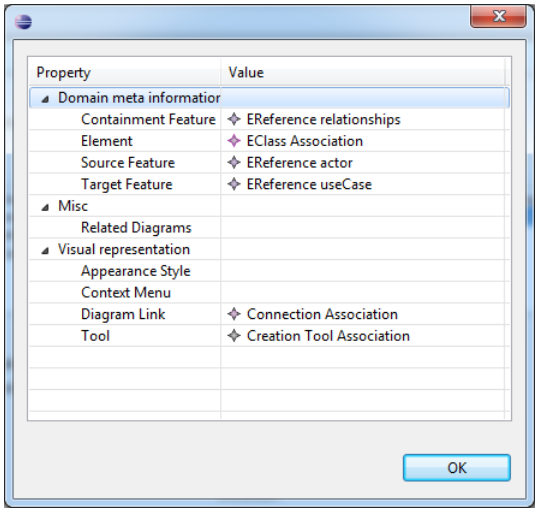


    16.8. A wizard most elég sok dolgot félreértelmezett.
    16.9. A **Remove** gombbal töröljük ki a **Links** listában lévő elemeket!

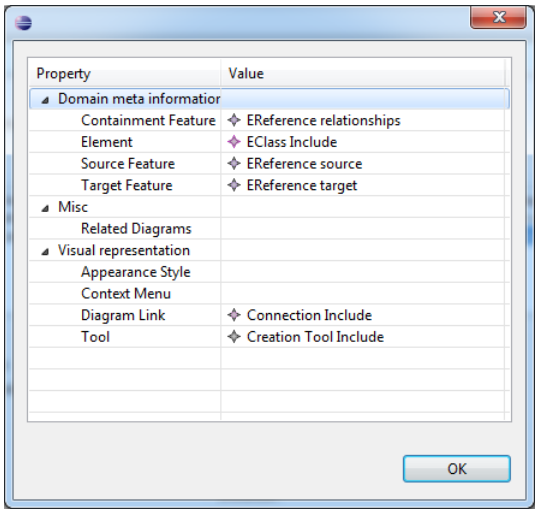
16.10. Tegyük át az **Extend**-et a linkek közé, majd a **Change…**-re kattintva állítsuk be a következőket:



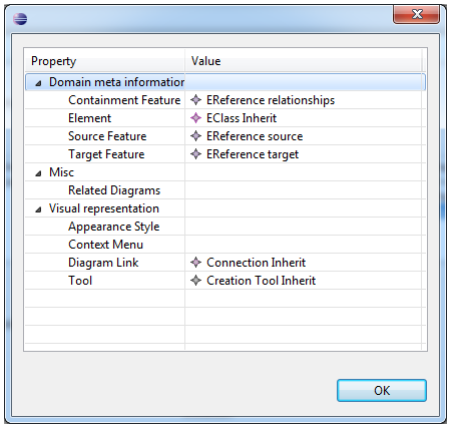
16.11. Hasonlóan járjunk el az Association-nel:



16.12. Hasonlóan járjunk el az Include-dal:

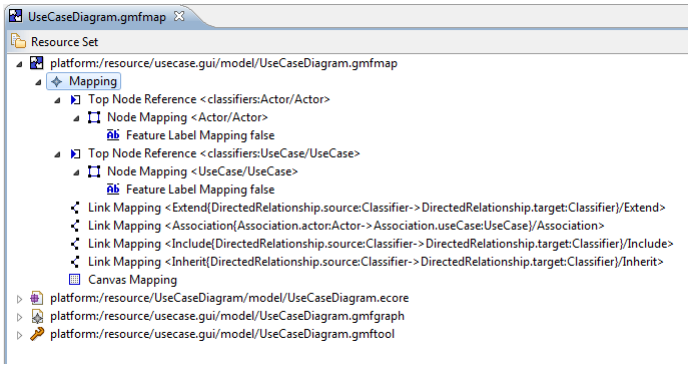


16.13. Hasonlóan járjunk el az Inherit-tel:

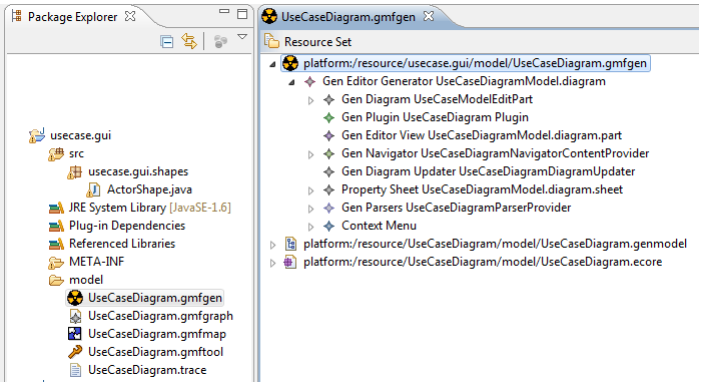


16.14. **Finish**

17. Ezzel elkészült a leképzés is:

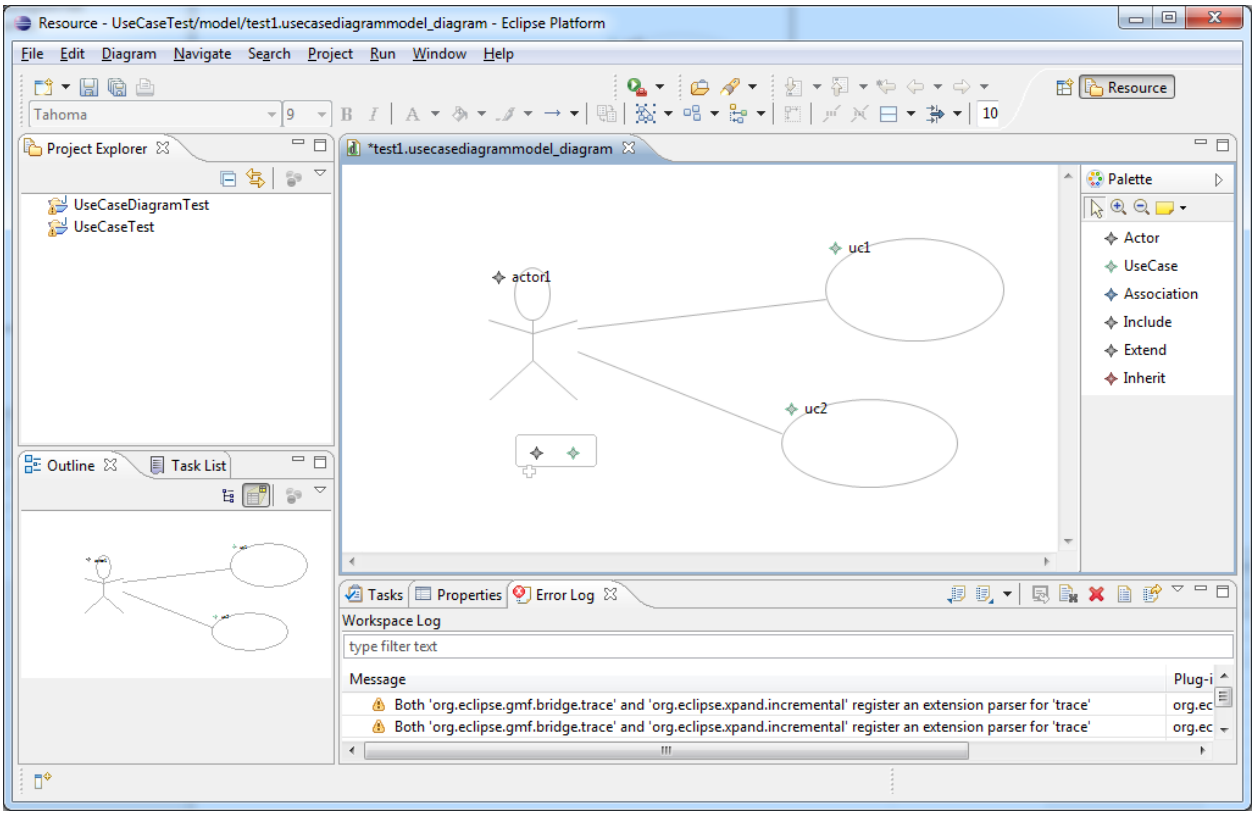


18. Vizsgáljuk meg, hogy a többi elemnek is jók-e a tulajdonságai, vagyis hogy az **Actor**-hoz a **Creation Tool Actor**, a **UseCase**-hez pedig a **Creation Tool UseCase** van-e beállítva a **Node Mapping Tool** tulajdonságaként a **Properties** ablakban. Ha nem, akkor javítsuk ezeket is.
19. Már csak a grafikus editort kell legenerálni. Kattintsunk a **GMF Dashboard**-on a **Transform** linkre a **Mapping Model** és a **Diagram Editor Gen Model** között!
20. Nyissuk meg az elkészült **UseCaseDiagram.gmfgen**-t:



21. Végül kattintsunk a **GMF Dashboard**-on a **Diagram Editor Gen Model** alatt a **Generate diagram editor** linken!
22. Ekkor elkészül a **UseCaseDiagram.diagram** nevű projekt, de egy kis hiba miatt nem fordul. A hiba javításához másoljuk át a **usegace.gui.shapes** csomagot az **ActorShape** osztállyal együtt ebbe a projektbe!
23. Kattintsunk jobbgombbal a **UseCaseDiagram.diagram** nevű projekten, majd válasszuk a **Run As > Eclipse Application** menüpontot!

24. A megjelenő új Eclipse-ben készítsünk egy üres EMF projektet! A projekt neve legyen **UseCaseTest**!
25. A **model** könyvtáron kattintsunk jobb gombbal, majd válasszuk a **New > Other…** menüpontot!
    25.1. A megjelenő ablakban válasszuk ki az **Examples** alatt a **UseCaseDiagram Diagram**-ot!
    25.2. **Next**
    25.3. **File name**: test1.usecasediagrammodel_diagram
    25.4. **Next**
    25.5. **File name**: test1.usecasediagrammodel
    25.6. **Finish**
26. Ha minden jól működik, akkor tudunk use-case diagramot rajzolni:



27. Egy dolog hiányzik még: az include és extend kapcsolatokra az **<<include>>** és **<<extend>>** felirat. Ezek hozzáadásához nyissuk meg ismét a **UseCaseDiagram.gmfgraph** szerkesztőjét, majd hajtsuk végre a következő lépéseket:
    27.1. A **Canvas UseCaseDiagramModel / Figure Gallery Default / Figure Descriptor IncludeFigure / Polyline Connection IncludeFigure**-re jobb gombbal kattintva adjunk hozzá egy új **Label**-t, majd ennek a tulajdonságait a következőképpen adjuk meg:
        27.1.1. **Name:** IncludeTextFigure
        27.1.2. **Text:** <<include>>
    27.2. A **Canvas UseCaseDiagramModel / Figure Gallery Default / Figure Descriptor IncludeFigure**-re jobb gombbal kattintva adjunk hozzá egy új **Child Access**-t, tulajdonságait pedig így állítsuk be:
        27.2.1. **Figure:** Label IncludeTextFigure
        27.2.2. **Accessor:** getFigureIncludeTextFigure
    27.3. Kattintsunk jobb gombbal a **Canvas**-en, és a **New Child** menüpont alatt adjunk hozzá egy új **Labels Diagram Label** elemet! Ennek a tulajdonságait a következőképpen adjuk meg:

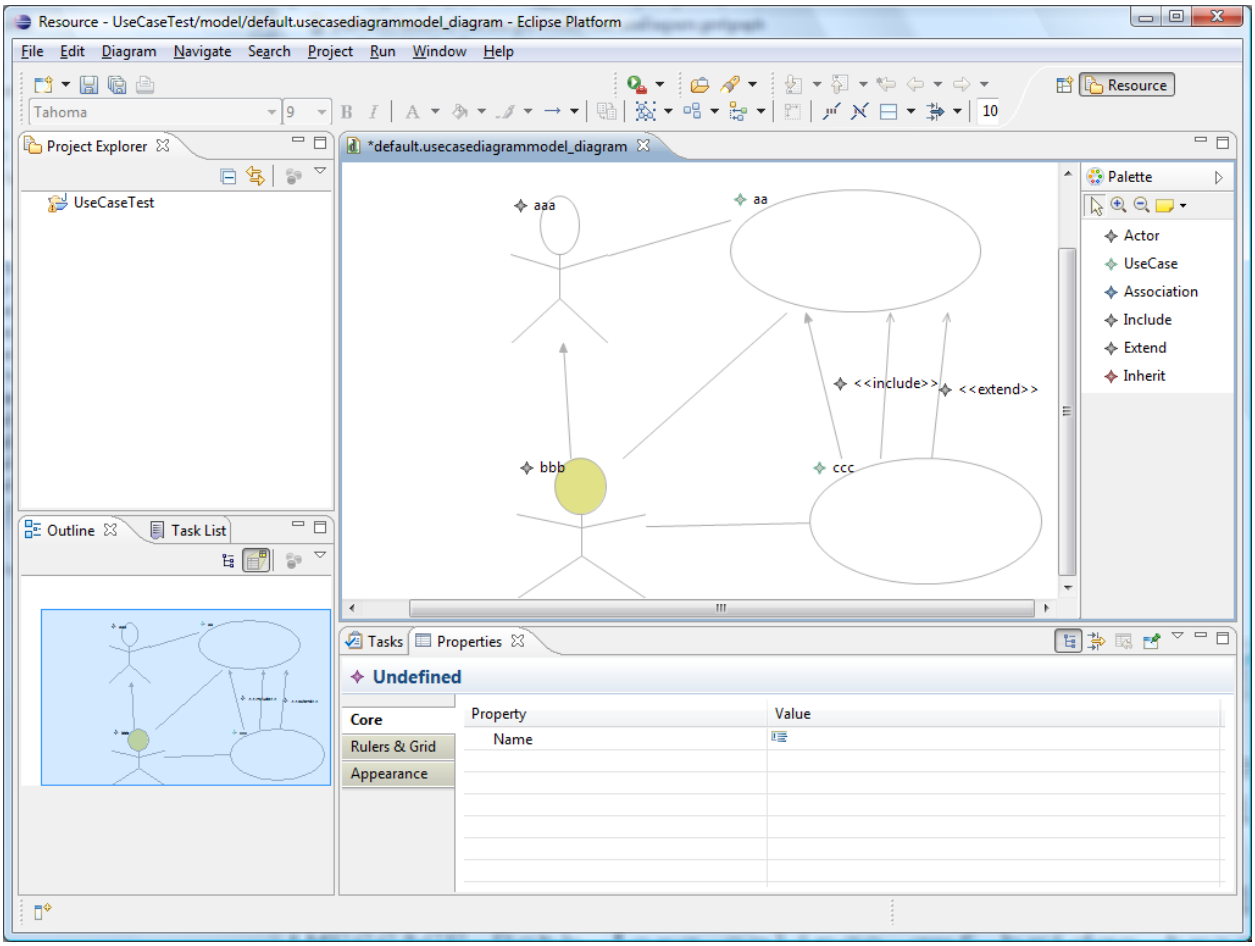
27.3.1. **Name:** IncludeText

27.3.2. **Figure:** Figure Descriptor IncludeFigure

27.3.3. **Accessor:** Child Access getFigureIncludeTextFigure

27.4. A fenti lépésekhez hasonló módon járjunk el az **extend** relációval is!

28. Nyissuk meg a **UseCaseDiagram.gmfmap**-et, és a **Mapping / Link Mapping <Include…/Include>**-hoz jobb gombbal adjunk hozzá egy **Label Mapping**-et, **Read Only** tulajdonságát állítsuk **true**-ra, a **Diagram Label** tulajdonságát pedig **Diagram Label IncludeText**-re! Hasonlóan járjunk el az **extend** relációval is! (Ha a címke szövege nem statikus, akkor **Feature Label Mapping**-et válasszunk! Ennek a **Features to display** tulajdonságában meg lehet adni, hogy az objektumnak mely attribútumát kösse hozzá a label szövegéhez.)
29. Nyissuk meg újra a **GMF Dashboard**-ot (figyeljünk arra, hogy a **Project Explorer**-ben a **usecase.gui** legyen a kiválasztott projekt), majd kattintsunk újra a **Mapping Model** és a Di**agram Editor Gen Model** közötti **Transform** linkre, végül pedig a **Diagram Editor Gen Model** dobozában lévő **Generate diagram editor** linkre!
30. Ezután előáll a **UseCaseDiagram.diagram** projekt egy frissebb változata, amelyben a nyilakon megjelennek a kívánt címkék:



3 CD és DFD grafikus szerkesztőjének elkészítése

Készítsünk grafikus modellező nyelvet a CD és DFD szerkesztéséhez!

A diagramon elegendő azokat a data-flow-kat ábrázolni, amelyeknek mindkét vége process-hez, store-hoz vagy terminator-hoz kapcsolódik. Azokat a data-flow-kat nem kell megjeleníteni, amelyeknek valamelyik vége szabad.