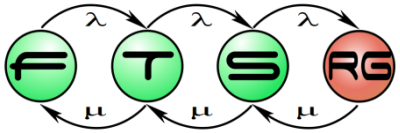
Grafikus szerkesztők fejlesztése

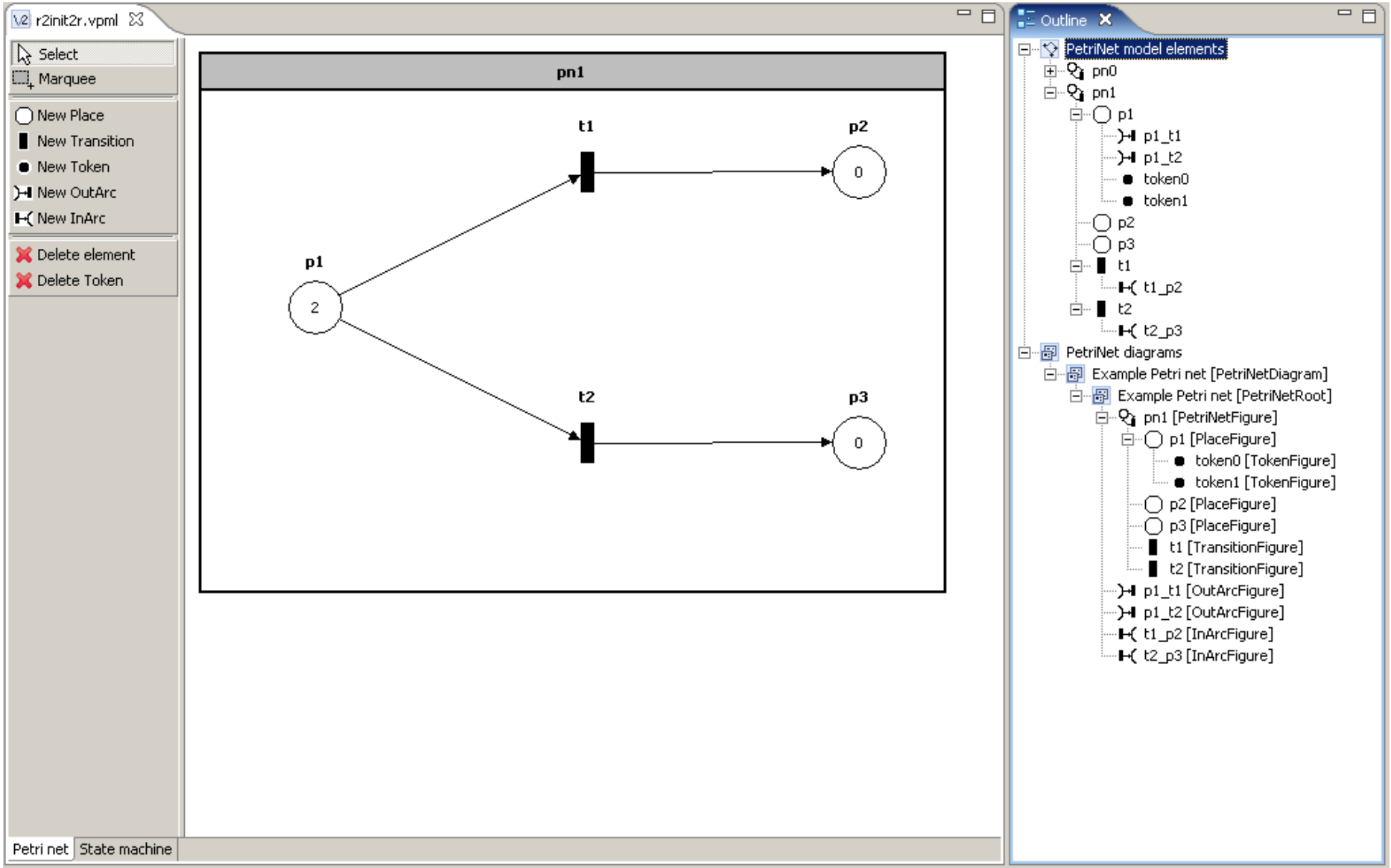
A Graphical Editing Framework



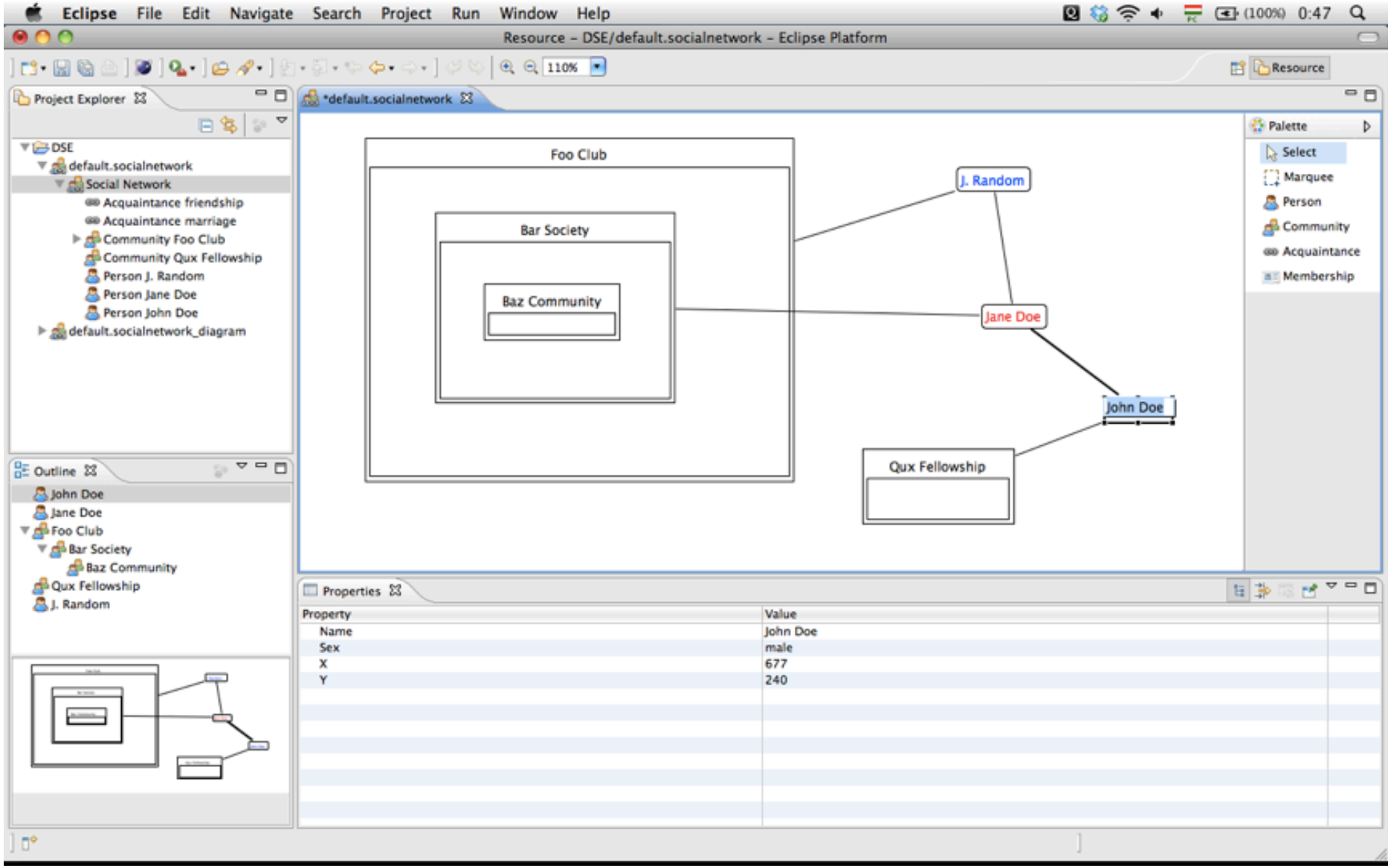
A GEF célja

- Grafikus szerkesztőprogramok
- Integráció az Eclipse környezetbe
- Tetszőleges modell megjelenítése
- Magas absztrakciós szint

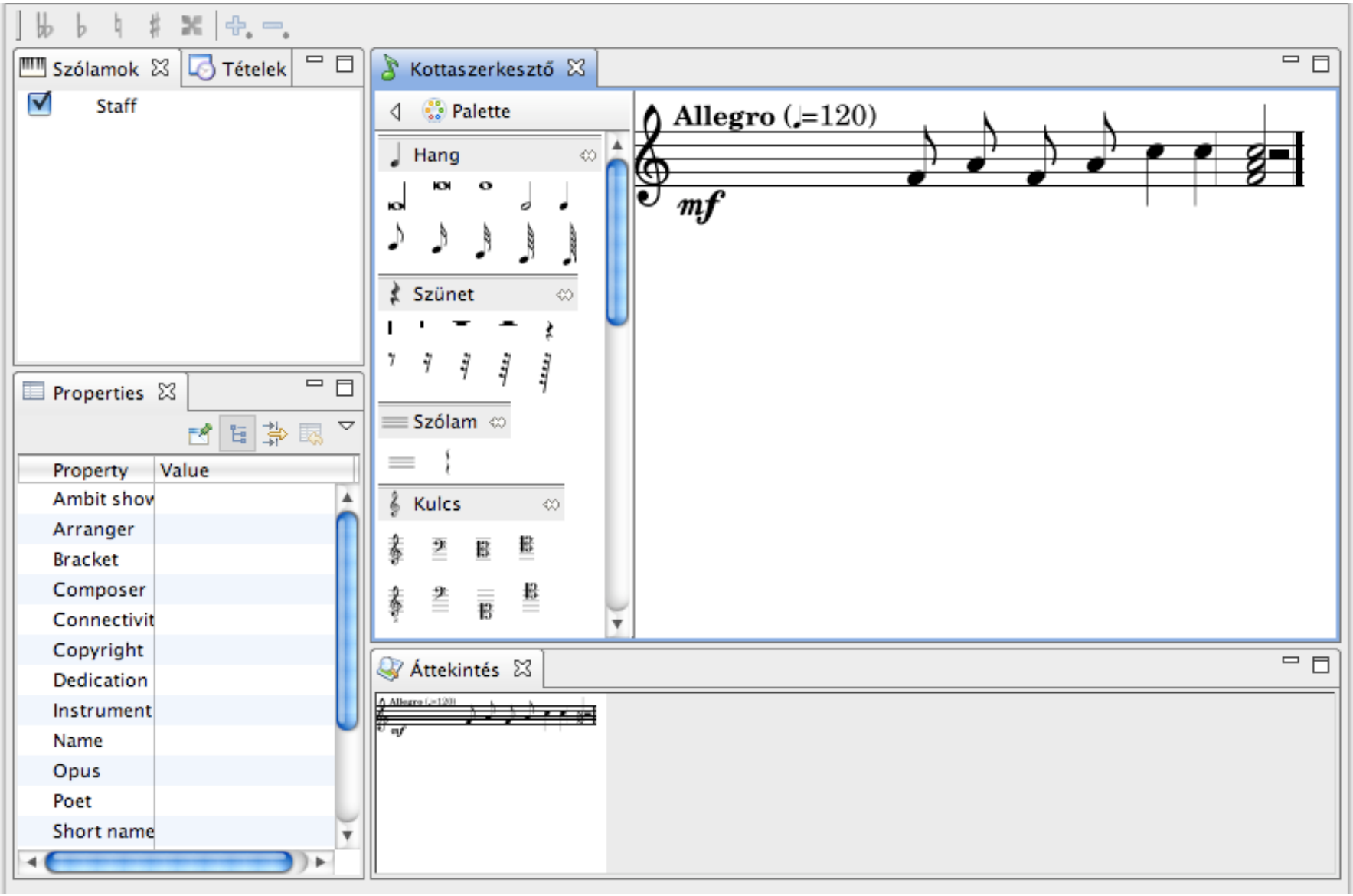
Példa


r2init2r.vpml
Select
Marquee
New Place
New Transition
New Token
New OutArc
New InArc
Delete element
Delete Token
pn1
t1
p2
0
p1
2
t2
p3
0
Petri net
State machine
Outline
PetriNet model elements
pn0
pn1
p1
p1_t1
p1_t2
token0
token1
p2
p3
t1
t1_p2
t2
t2_p3
PetriNet diagrams
Example Petri net [PetriNetDiagram]
Example Petri net [PetriNetRoot]
pn1 [PetriNetFigure]
p1 [PlaceFigure]
token0 [TokenFigure]
token1 [TokenFigure]
p2 [PlaceFigure]
p3 [PlaceFigure]
t1 [TransitionFigure]
t2 [TransitionFigure]
p1_t1 [OutArcFigure]
p1_t2 [OutArcFigure]
t1_p2 [InArcFigure]
t2_p3 [InArcFigure]

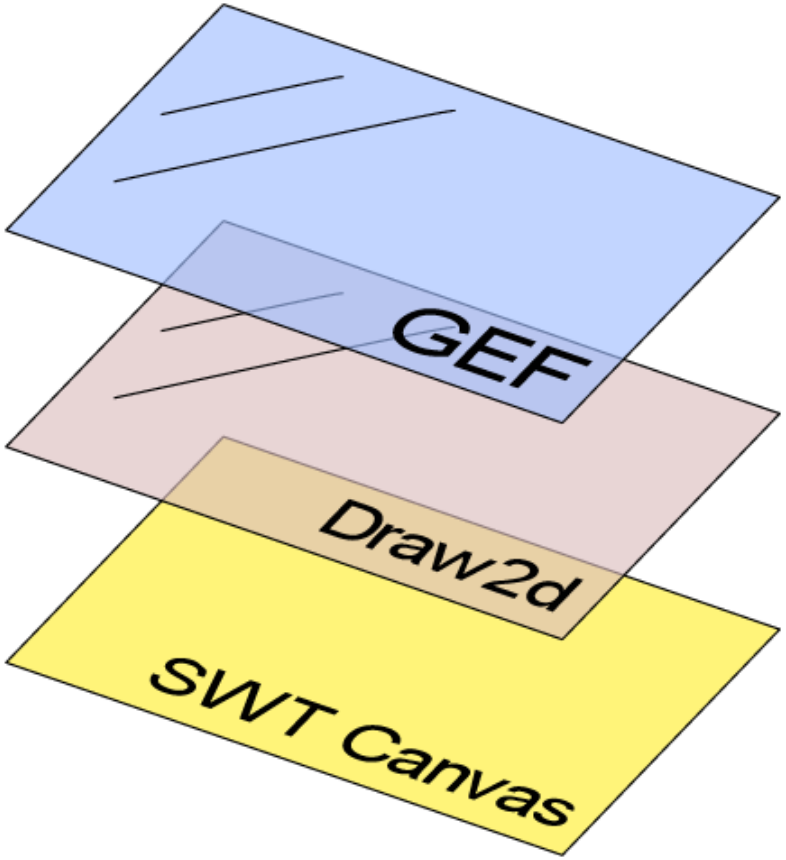
Példa


Eclipse File Edit Navigate Search Project Run Window Help
Resource – DSE/default.socialnetwork – Eclipse Platform
Project Explorer
DSE
default.socialnetwork
Social Network
Acquaintance friendship
Acquaintance marriage
Community Foo Club
Community Qux Fellowship
Person J. Random
Person Jane Doe
Person John Doe
default.socialnetwork_diagram
*default.socialnetwork
Foo Club
Bar Society
Baz Community
J. Random
Jane Doe
John Doe
Qux Fellowship
Palette
Select
Marquee
Person
Community
Acquaintance
Membership
Outline
John Doe
Jane Doe
Foo Club
Bar Society
Baz Community
Qux Fellowship
J. Random
Properties
Property
Value
Name
John Doe
Sex
male
X
677
Y
240

Példa


Szólamok
Tételek
Staff
Kottaszerkesztő
Palette
Hang
Szünet
Szólam
Kulcs
Allegro (♩=120)
mf
Properties
Property
Value
Ambit shov
Arranger
Bracket
Composer
Connectivit
Copyright
Dedication
Instrument
Name
Opus
Poet
Short name
Áttekintés

Megjelenítési GEF környezetben



- Interakció (MVC)
- Modell - nézet leképezés
- Eclipse integráció

- Megjelenítés
- Elemek elrendezése
- Nagyítás

- Natív (SWT) réteg

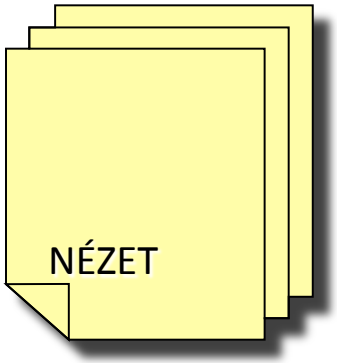
MVC felépítés

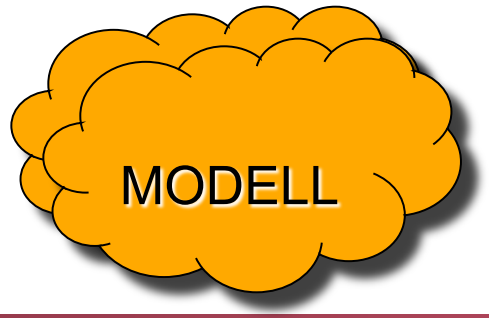
- Model-View-Controller (Modell – Nézet – Vezérlő)
- Adatok tárolása és megjelenítése egymástól elválasztva
  - Modell: adatok tárolása
  - Nézet: grafikus megjelenítés
  - Vezérlő: felhasználói interakció

MVC a GEF környezetben

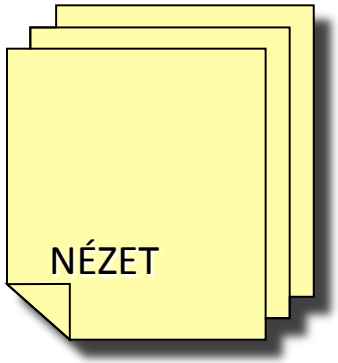
Felhasználó

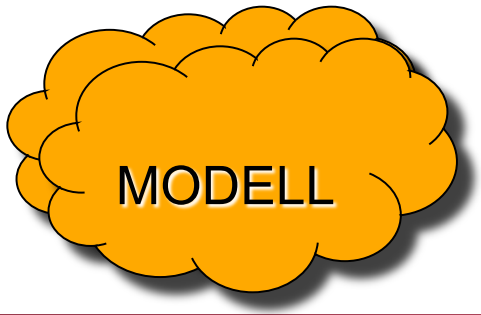
VEZÉRLŐ



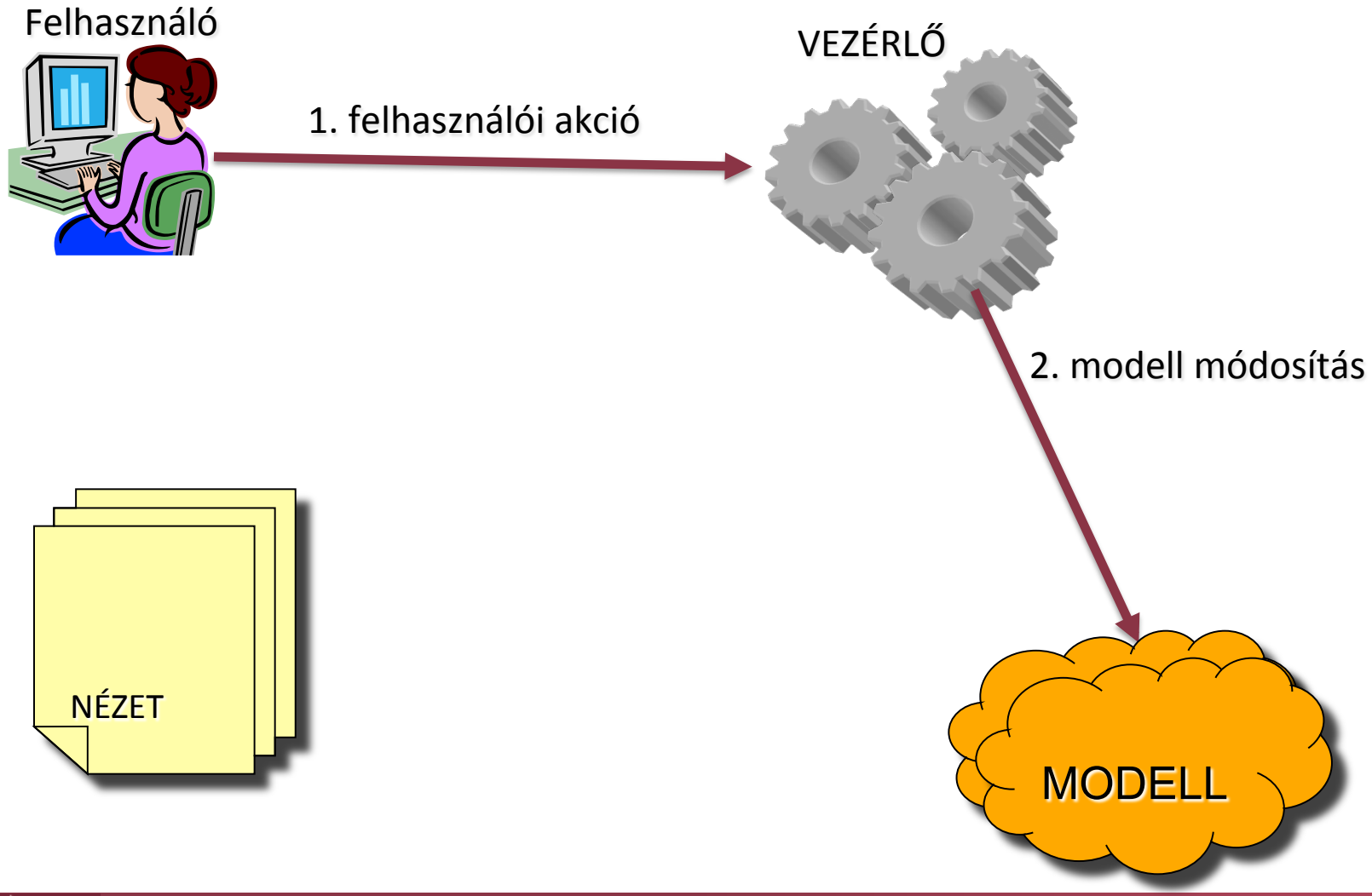


MVC a GEF környezetben

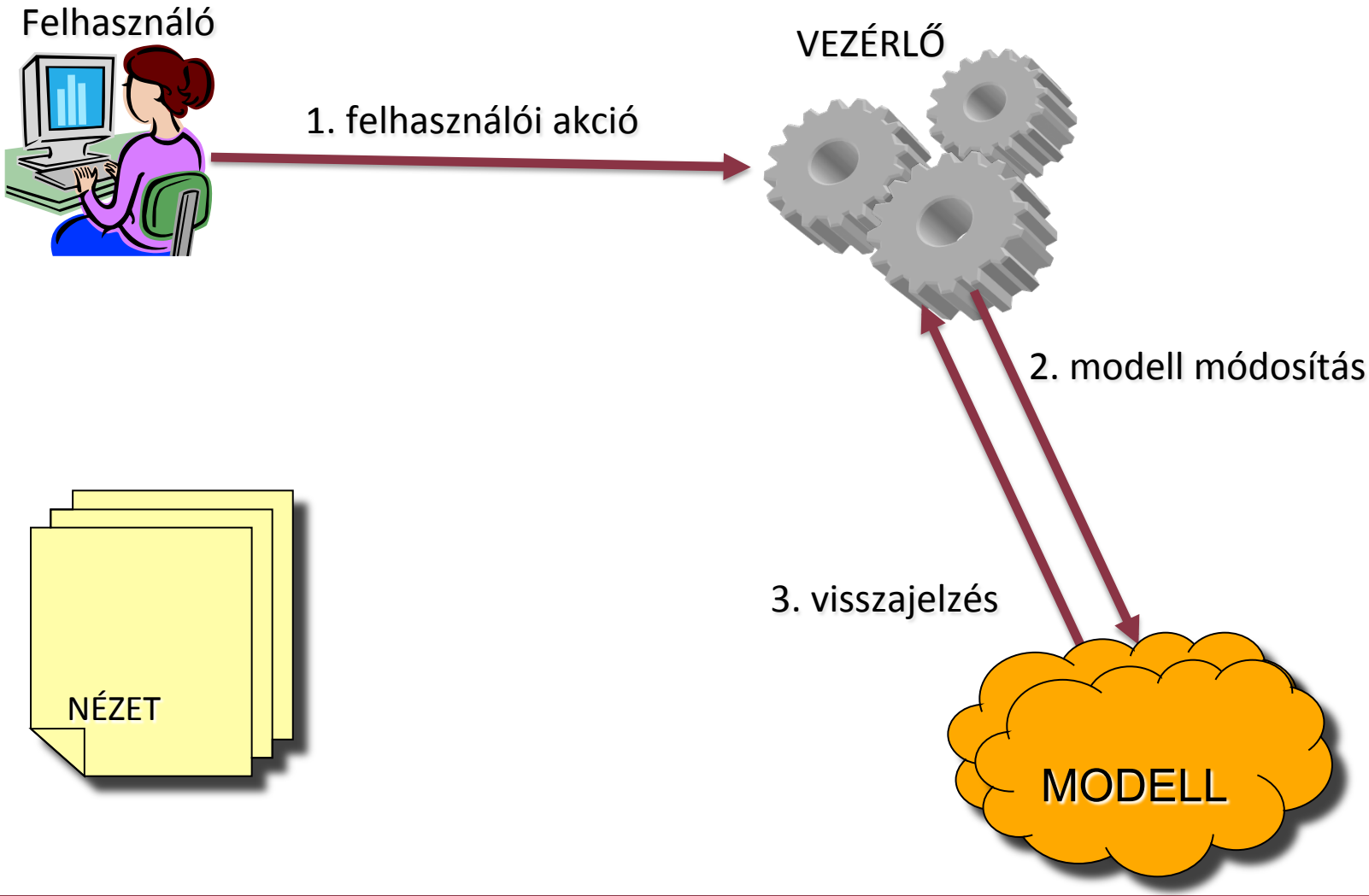




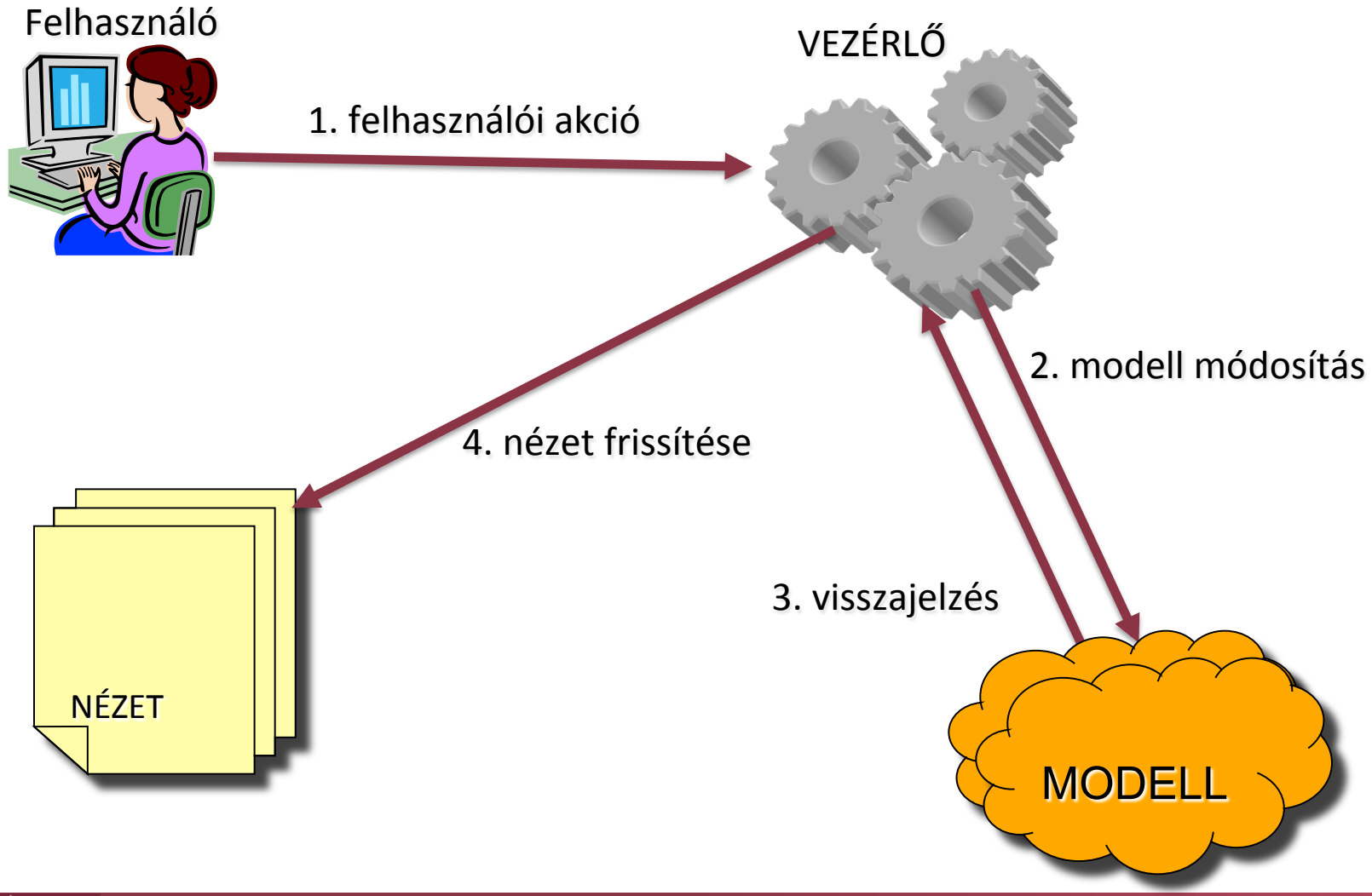
MVC a GEF környezetben



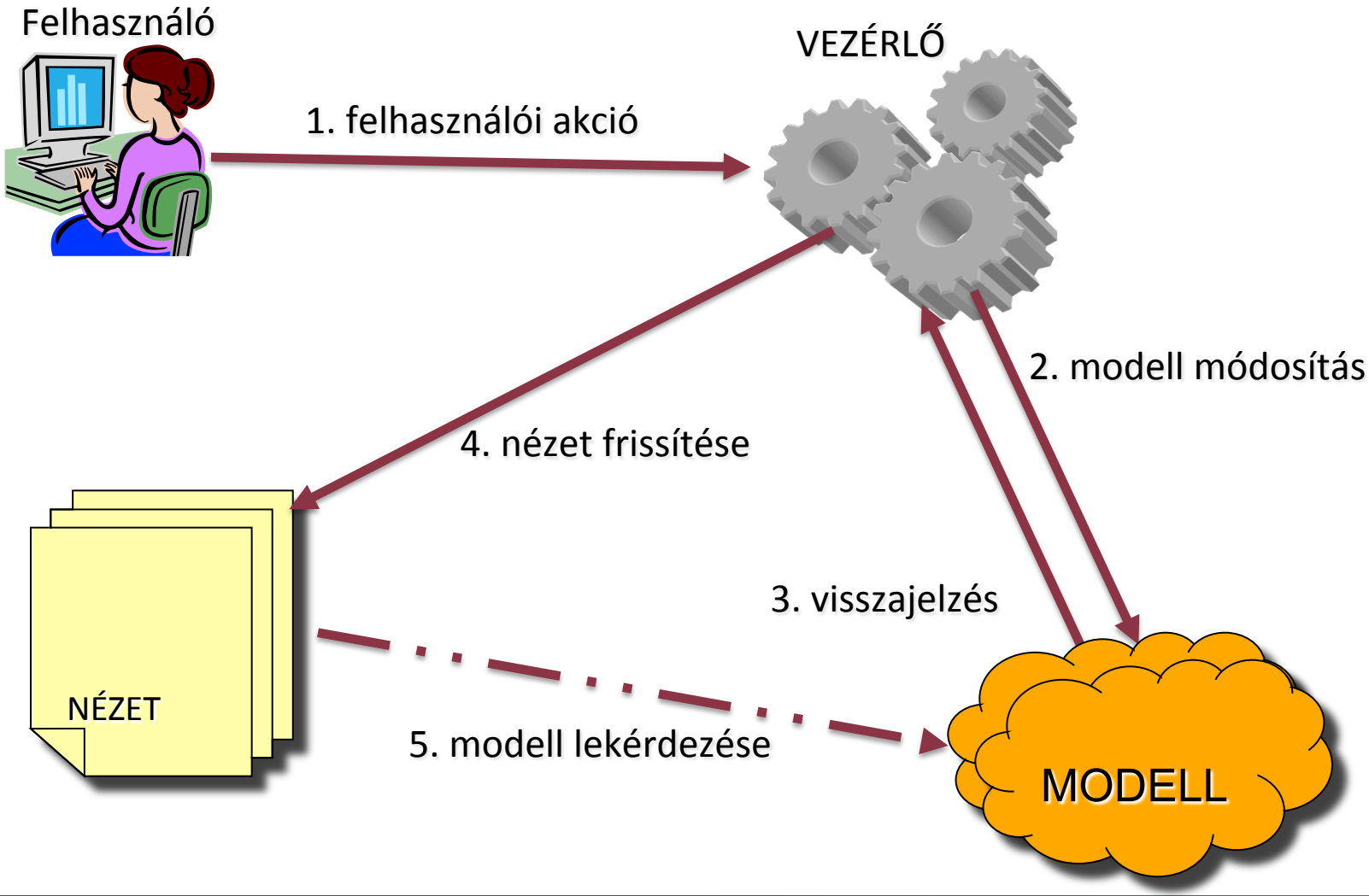
MVC a GEF környezetben



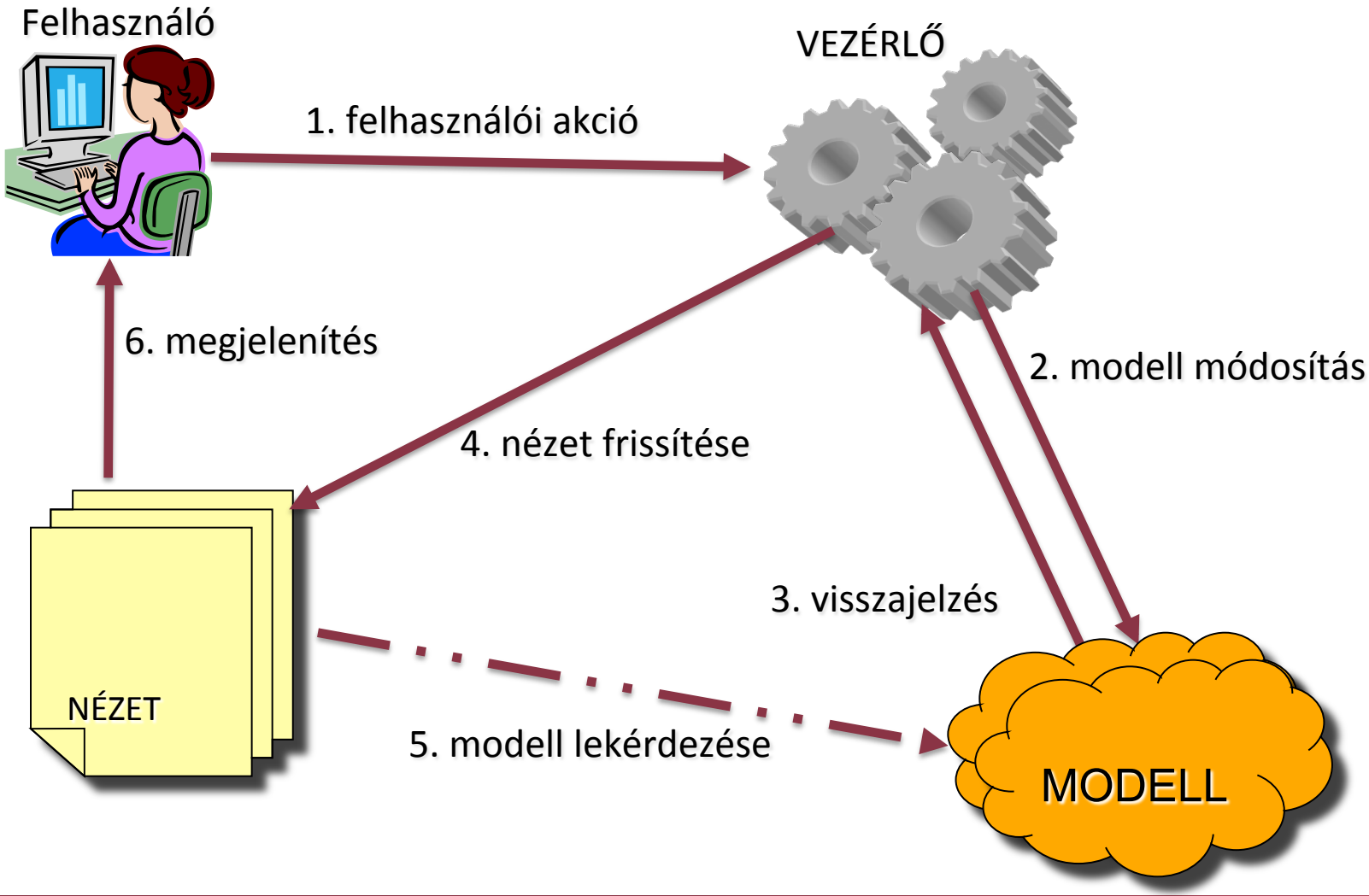
MVC a GEF környezetben



MVC a GEF környezetben



MVC a GEF környezetben



MVC a GEF-ben

Model

MVC a GEF-ben

Model

Értesítés

MVC a GEF-ben: Model

- Modell: tetszőleges
  - Pl. Java osztályok, EMF, adatbázis
  - Hierarchikus felépítés (gyökeres fa)
  - Támogatnia kell az értesítéseket
    - Jelentés a vezérlőnek, ha módosítás történt
    - 1 modell - több nézet esetén kritikus
    - Pl. EMF Notification Framework
  - Üzleti modell
    - Struktúra, adatok
  - Nézeti modell
    - Megjelenítési információk
    - Pl. pozíció, méret

Egyszerű modell

```
public class TestModel {
   private String name;
   private Rectangle bounds;
   public String getName() {
        return name;
      }
   public void setName(String name) {
        this.name = name;
      }
   public Rectangle getBounds() {
        return bounds;
      }
   public void setBounds(Rectangle bounds) {
        this.bounds = bounds;
      }
}
```

Egyszerű modell

```
public class TestModel {
   private String name;
   private Rectangle bounds;
   public String getName() {
        return name;
      }
   public void setName(String name) {
        this.name = name;
      }
   public Rectangle getBounds() {
        return bounds;
      }
   public void setBounds(Rectangle bounds) {
        this.bounds = bounds;
      }
}
```

Üzleti modell

Egyszerű modell

```
public class TestModel {
   private String name;
   private Rectangle bounds;
   public String getName() {
        return name;
      }
   public void setName(String name) {
        this.name = name;
      }
   public Rectangle getBounds() {
        return bounds;
      }
   public void setBounds(Rectangle bounds) {
        this.bounds = bounds;
      }
}
```

Egyszerű modell

```
public class TestModel {
   private String name;
   private Rectangle bounds;
   public String getName() {
        return name;
      }
   public void setName(String name) {
        this.name = name;
      }
   public Rectangle getBounds() {
        return bounds;
      }
   public void setBounds(Rectangle bounds) {
        this.bounds = bounds;
      }
}
```

Nézeti modell

Egyszerű értesítés

```
public interface MyModelListener {
     public void modelChanged();
}


public class TestModel {
  ...
  private final List<MyModelListener> listeners =
       new ArrayList<MyModelListener>();
  public void addListener(MyModelListener listener) {
       listeners.add(listener);
     }
     public void removeListener(MyModelListener listener) {
       listeners.remove(listener);
     }
  protected void notifyListeners() {
       for (MyModelListener listener : listeners) {
         listener.modelChanged();
       }
     }
}
```

Egyszerű értesítés

```
public interface MyModelListener {
     public void modelChanged();
}

public class TestModel {
   ...
   private final List<MyModelListener> listeners =
        new ArrayList<MyModelListener>();
   public void addListener(MyModelListener listener) {
        listeners.add(listener);
     }
     public void removeListener(MyModelListener listener) {
        listeners.remove(listener);
     }
   protected void notifyListeners() {
        for (MyModelListener listener : listeners) {
          listener.modelChanged();
        }
     }
}
```

Figyelők kezelése

Egyszerű értesítés

```
public interface MyModelListener {
     public void modelChanged();
}


public class TestModel {
  ...
  private final List<MyModelListener> listeners =
       new ArrayList<MyModelListener>();
  public void addListener(MyModelListener listener) {
       listeners.add(listener);
     }
     public void removeListener(MyModelListener listener) {
       listeners.remove(listener);
     }
  protected void notifyListeners() {
       for (MyModelListener listener : listeners) {
         listener.modelChanged();
       }
     }
}
```

Egyszerű értesítés

```
public interface MyModelListener {
     public void modelChanged();
}

public class TestModel {
   ...
   private final List<MyModelListener> listeners =
        new ArrayList<MyModelListener>();
   public void addListener(MyModelListener listen
        listeners.add(listener);
     }
     public void removeListener(MyModelListener
        listeners.remove(listener);
     }
   protected void notifyListeners() {
        for (MyModelListener listener : listeners) {
          listener.modelChanged();
        }
     }
}
```

Értesítés küldése

Push vagy Pull értesítés

- Pull: Csak annyit küld, hogy változás történt
  - Gyors, erőforráskímélő
  - Minden változó attribútumot vizsgálni kell
- Push: Pontosan megmondjuk, hogy mi változott (pl. új X pozíció = 172)
  - El kell küldeni magát a változást is, lassú
  - Könnyen feldolgozható

GEF workflow

Model

View

GEF workflow

Model

View

Kirajzolás

GEF workflow

Model

View

Kirajzolás

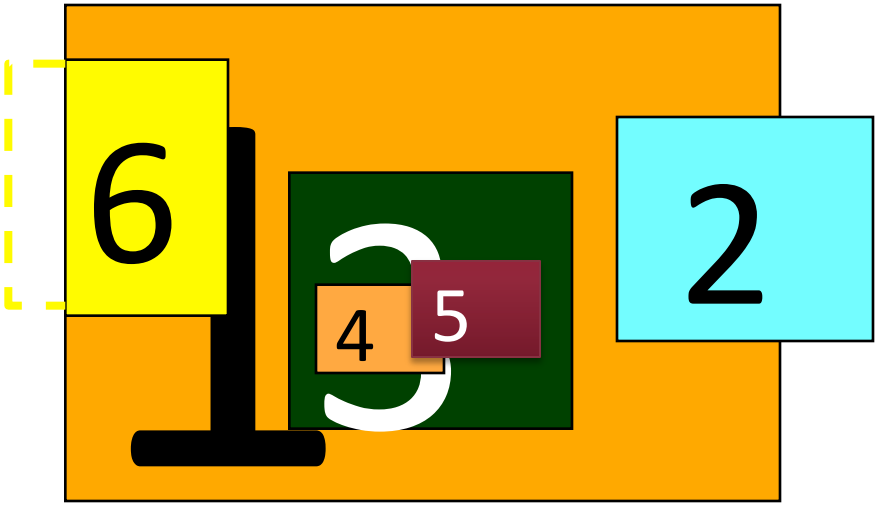
Elrendezés

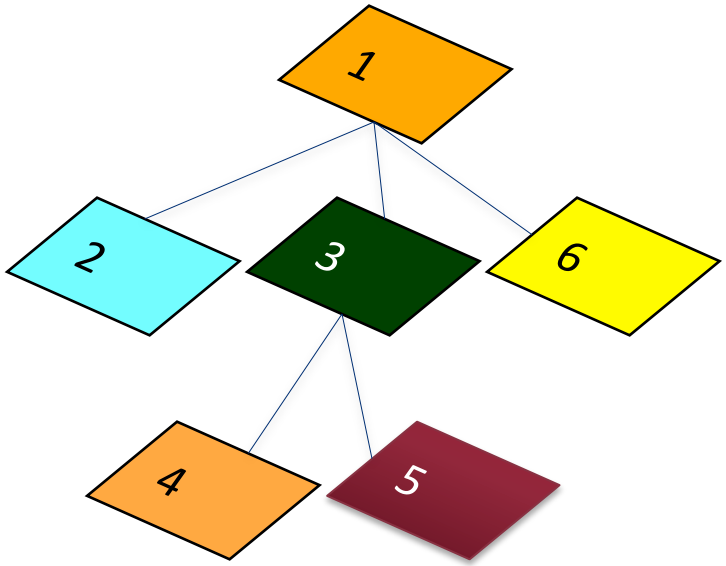
MVC a GEF-ben: View

- Nézet: Draw2D osztályok
  - SWT-re épülő vektorgrafikus könyvtár
  - Egyszerű elemek (címke, téglalap, nyíl)
  - Hierarchikus megjelenítés
  - Alap építőelem: Figure
  - GEF nézet = Draw2D Figure példány

Draw2D hierarchia

- Gyerek pozíciója lehet relatív a szülőhöz képest
- Gyermekek negatív koordináta felé (balra, felfele) levágva
- Pontosan 1 gyökérelem

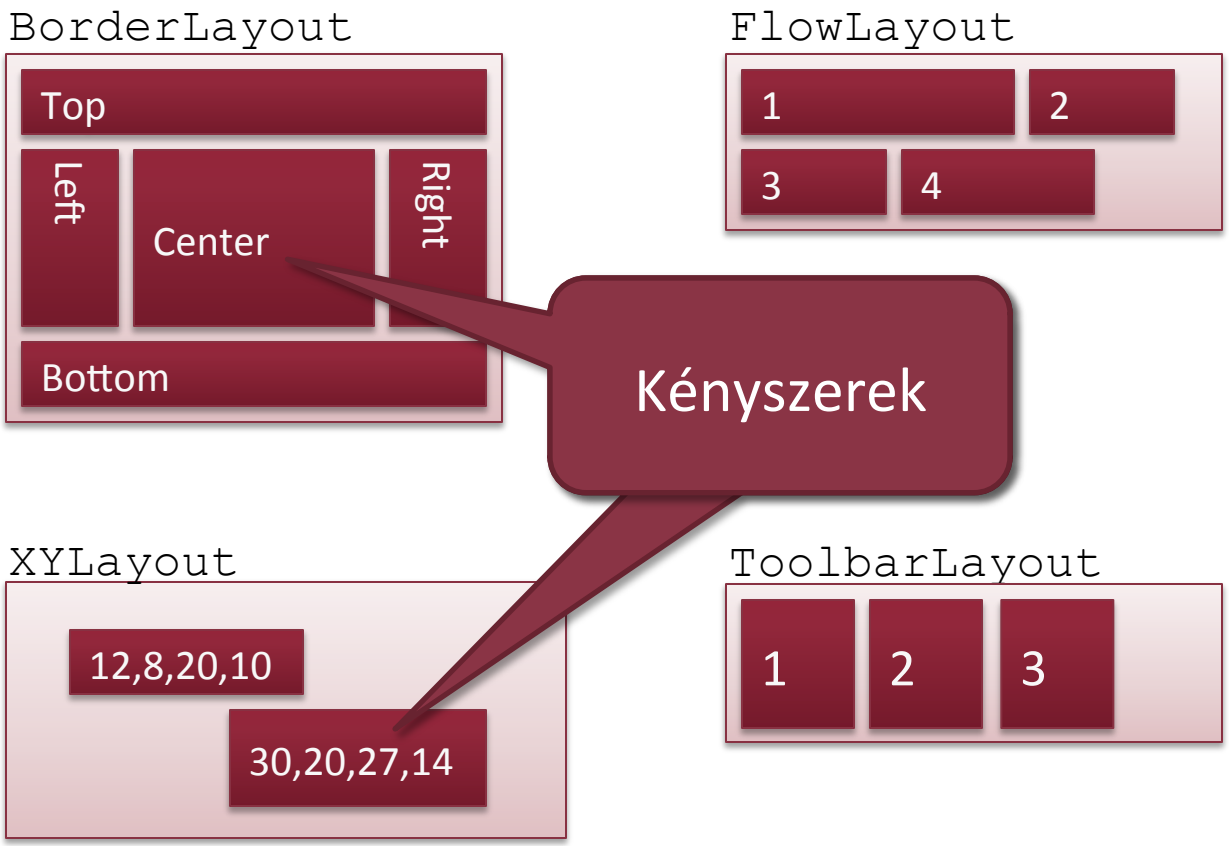




Draw2D LayoutManager

- Gyerekek elrendezése szülőn belül
- Több beépített, lehet saját is
- Constraint: Egy gyerek elhelyezésével kapcsolatos kényszer
  - Szülő pozíció és méret + LayoutManager + Constraint
    - Gyerek pozíció és méret
  - A gyerekhez kell hozzárendelni
  - A szülő LayoutManagere ez alapján végzi el az elrendezést

Draw2D LayoutManagerek



Draw2D alapelemek

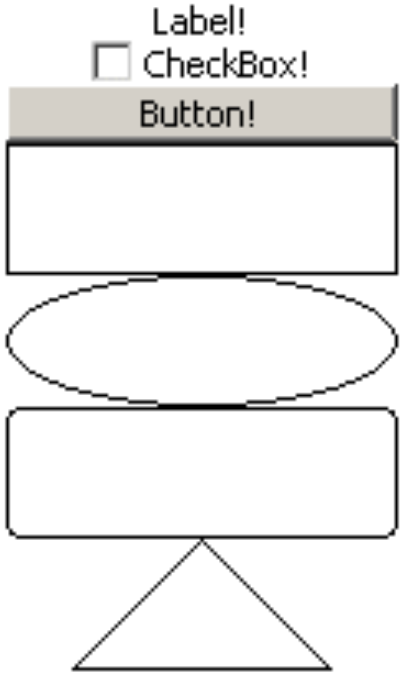
- Egyszerű elemek
  - Label, Button, CheckBox, Image
- Geometriai alakzatok
  - RectangleFigure, Ellipse, Triangle
- Panel: általános konténer elem
- ScrollPane: görgethető tartalmú elem

Draw2D egyéb elemek

- Nyilak
- Keretek
- Saját elemek
  - Beépített elemek kombinációja nem mindig elégséges
  - paintFigure() felülírása kell
  - Tetszőleges SWT rajzoló kód lehet

Alapelemek - példa

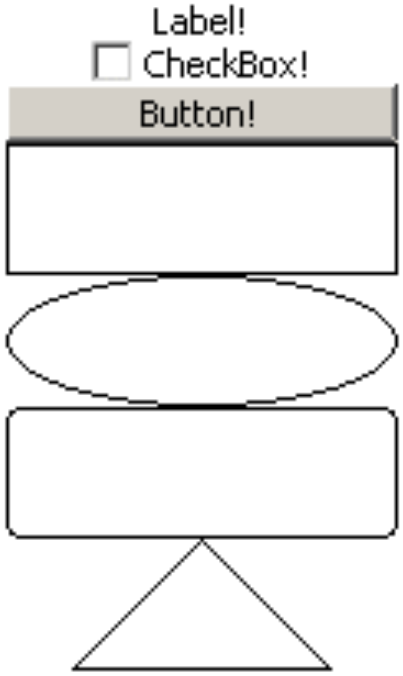
```
public class Pelda1 extends Figure {
 public Pelda1() {
    setOpaque(true);
    setBackgroundColor(ColorConstants.white);
    setLayoutManager(new ToolbarLayout());
    add(new Label("Label!"));
    add(new CheckBox("CheckBox!"));
    add(new Button("Button!"));
    add(new RectangleFigure());
    add(new Ellipse());
    add(new RoundedRectangle());
    add(new Triangle());
    for (int i = 3; i <= 6; i++)
     ((Figure) getChildren().get
       (i)).setPreferredSize(-1, 40);
    }
 }
}
```



Alapelemek - példa

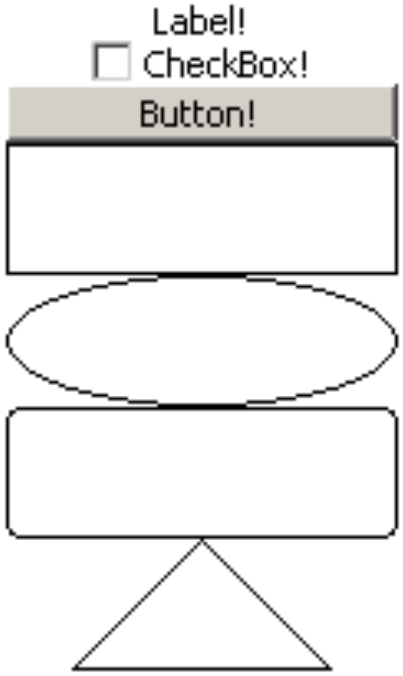
```
public class Pelda1 ext
 public Pelda1() {
   setOpaque(true);
   setBackgroundColor(ColorConstants.white);
   setLayoutManager(new ToolbarLayout());
   add(new Label("Label!"));
   add(new CheckBox("CheckBox!"));
   add(new Button("Button!"));
   add(new RectangleFigure());
   add(new Ellipse());
   add(new RoundedRectangle());
   add(new Triangle());
   for (int i = 3; i <= 6; i++)
    ((Figure) getChildren().get
      (i)).setPreferredSize(-1, 40);
   }
 }
}
```

Alapértelmezetten minden átlátszó



Alapelemek - példa

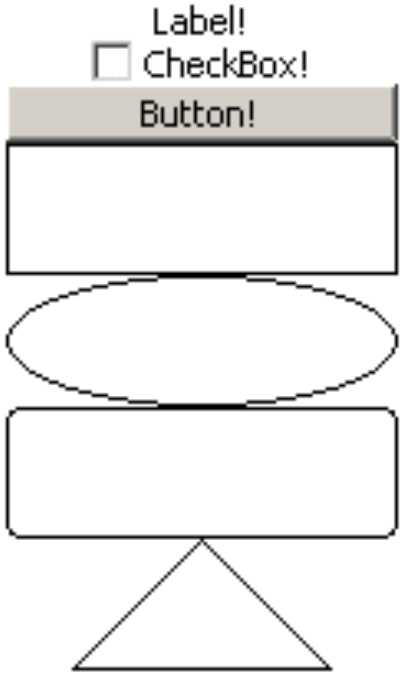
```
public class Pelda1 extends Figure {
 public Pelda1() {
    setOpaque(true);
    setBackgroundColor(ColorConstants.white);
    setLayoutManager(new ToolbarLayout());
    add(new Label("Label!"));
    add(new CheckBox("CheckBox!"));
    add(new Button("Button!"));
    add(new RectangleFigure());
    add(new Ellipse());
    add(new RoundedRectangle());
    add(new Triangle());
    for (int i = 3; i <= 6; i++)
     ((Figure) getChildren().get
       (i)).setPreferredSize(-1, 40);
    }
 }
}
```



Alapelemek - példa

```
public class Pelda1 extends Figure {
 public Pelda1() {
    setOpaque(true);
    setBackgroundColor(ColorConstants.white);
    setLayoutManager(new ToolbarLayout());
    add(new Label("Label!"));
    add(new CheckBox("CheckBox!"));
    add(new Button("Button!"));
    add(new RectangleFigure());
    add(new Ellipse());
    add(new RoundedRectangle());
    add(new Triangle());
    for (int i = 3; i <= 6; i++)
     ((Figure) getChildren().get
       (i)).setPreferredSize(-1, 40);
    }
 }
}
```

Preferált méret:
LayoutManager figyelheti

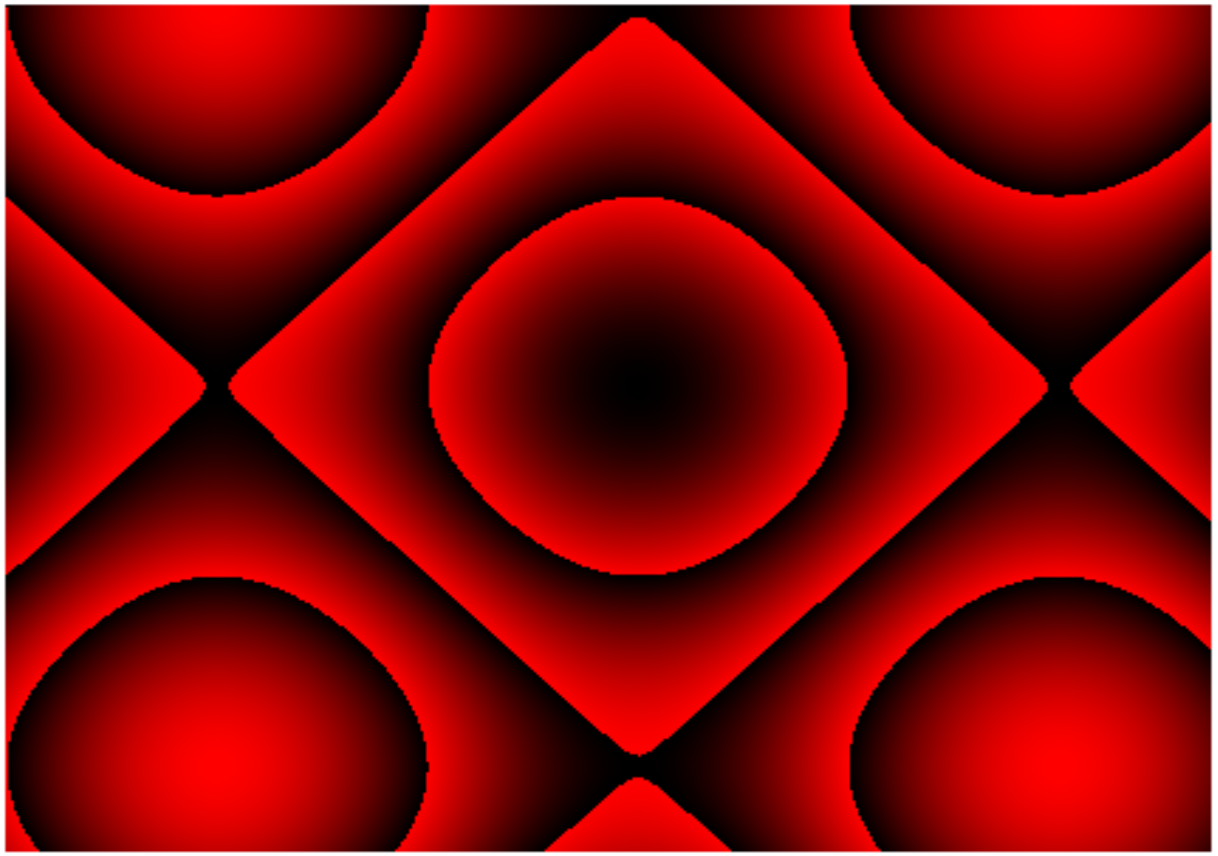


Draw2D saját elem

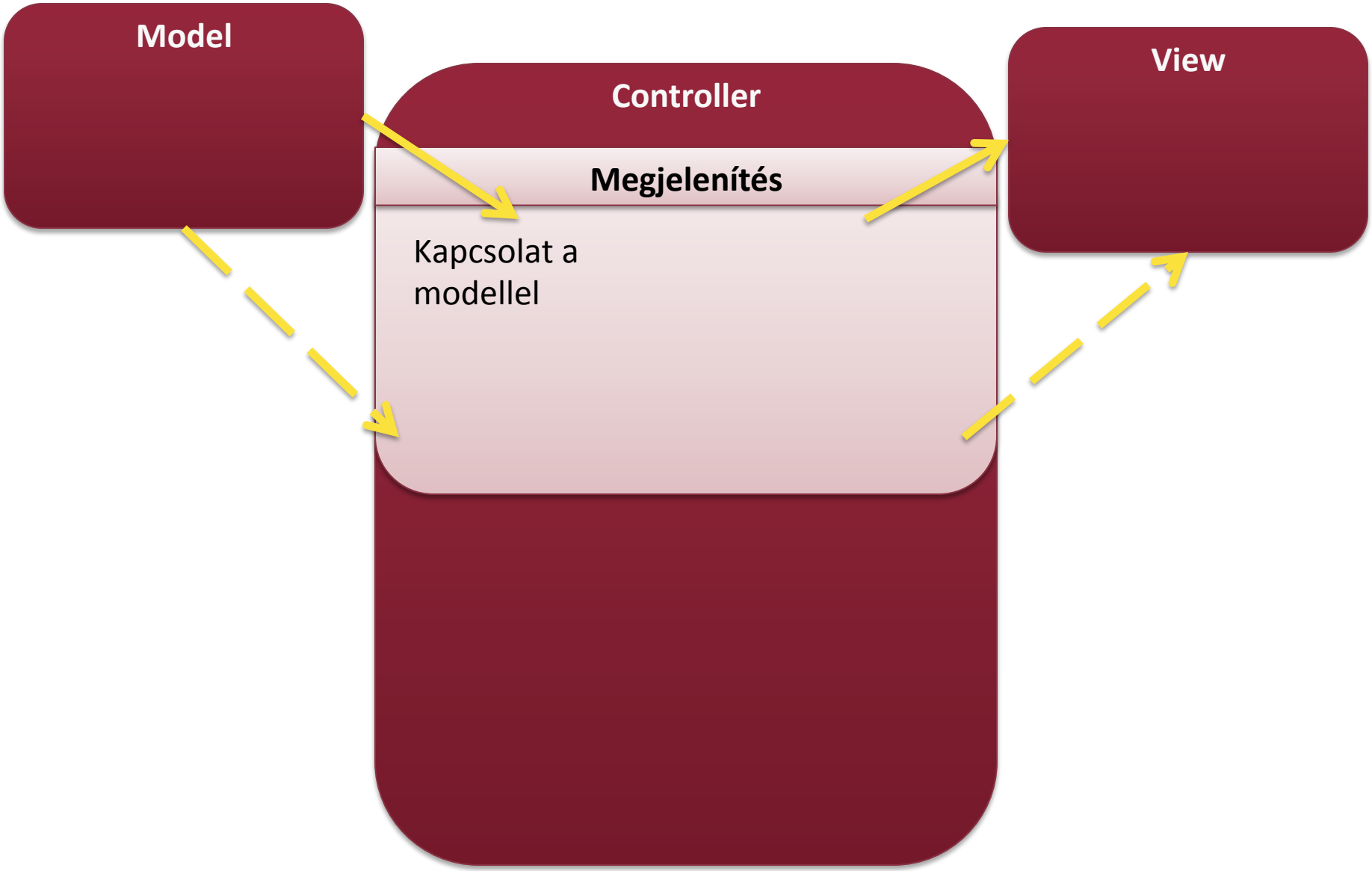
```
public class Example3 extends Figure {
   @Override
   protected void paintFigure(Graphics graphics) {
        Rectangle r = getBounds();
        PaletteData pd = new PaletteData(0xff0000, 0xff00, 0xff);
        pd.redShift = -16; pd.greenShift = -8; pd.blueShift = 0;
        pd.isDirect = true;
        ImageData id = new ImageData(r.width, r.height, 24, pd);
        for (int u = 0; u < r.width; u++) {
            for (int v = 0; v < r.height; v++) {
               int rc = ((int) ((Math.sin(u * 9.0 / r.width) +
                   Math.cos(v * 7.0 / r.height)) * 256.0)) % 256;
               id.setPixel(u, v, rc << 16);
            }
        }
        Image img = new Image(Display.getCurrent(), id);
        graphics.drawImage(img, r.getTopLeft());
   }
}
```

Draw2D saját elem

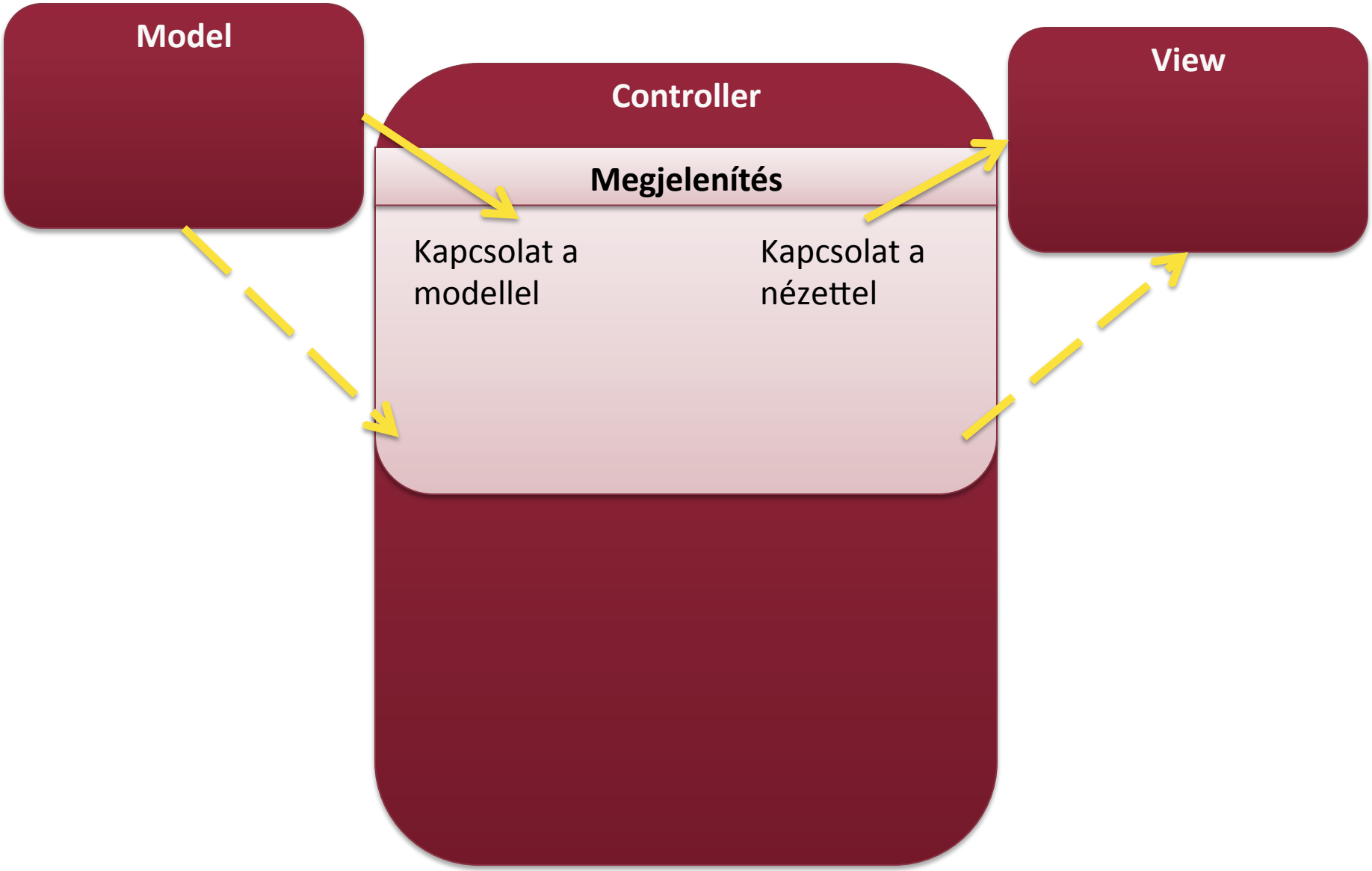
```
public class
   @Override
   protected
        Rectan
        Palett
        pd.red
        pd.isD
        ImageD
        for (i
            fo
                                                    ff);
                                                    = 0;
                                                    od);
                                                    +
                                                    256;
            }
        }
        Image
        graphi
   }
}
```



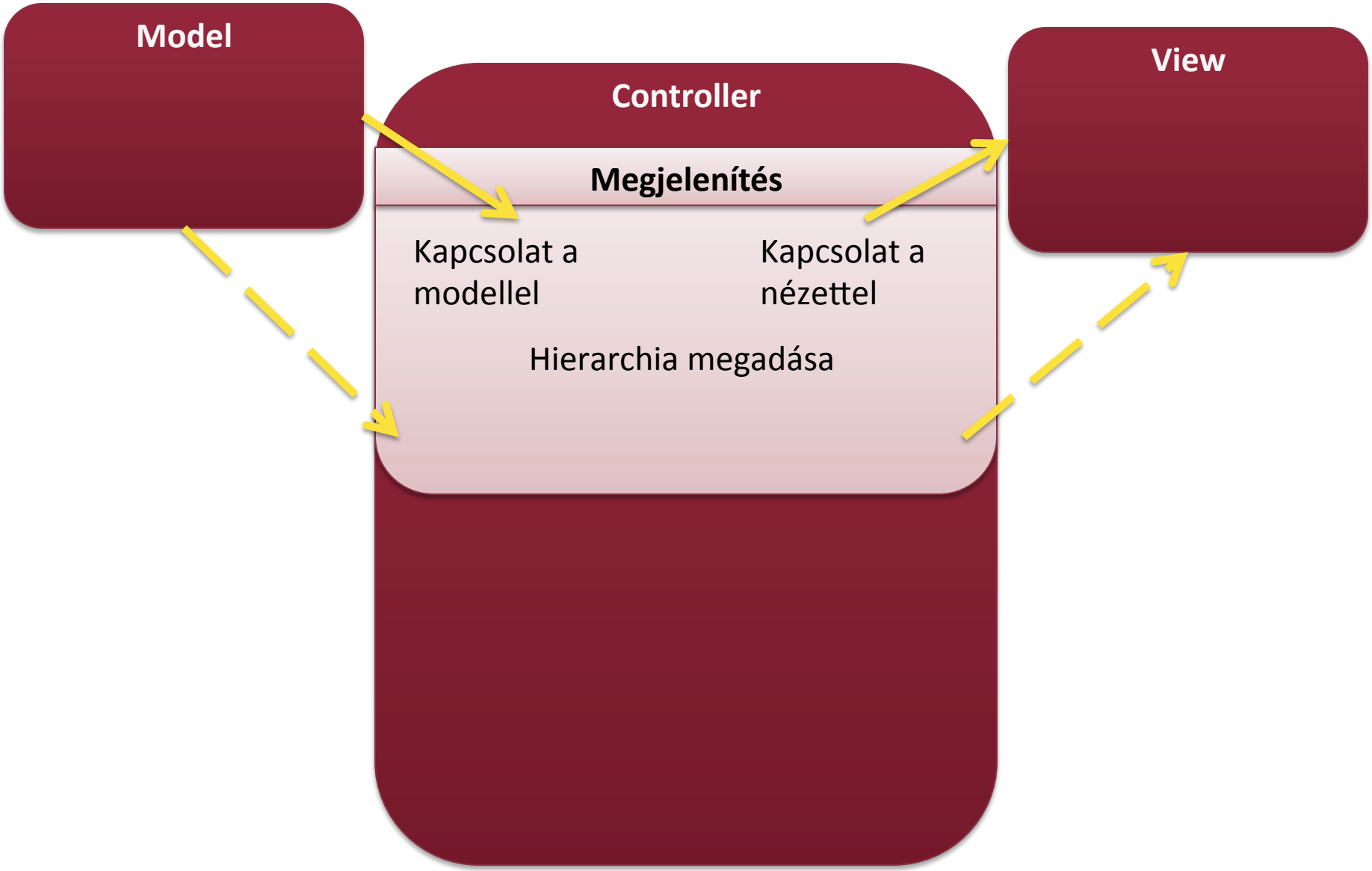
GEF workflow



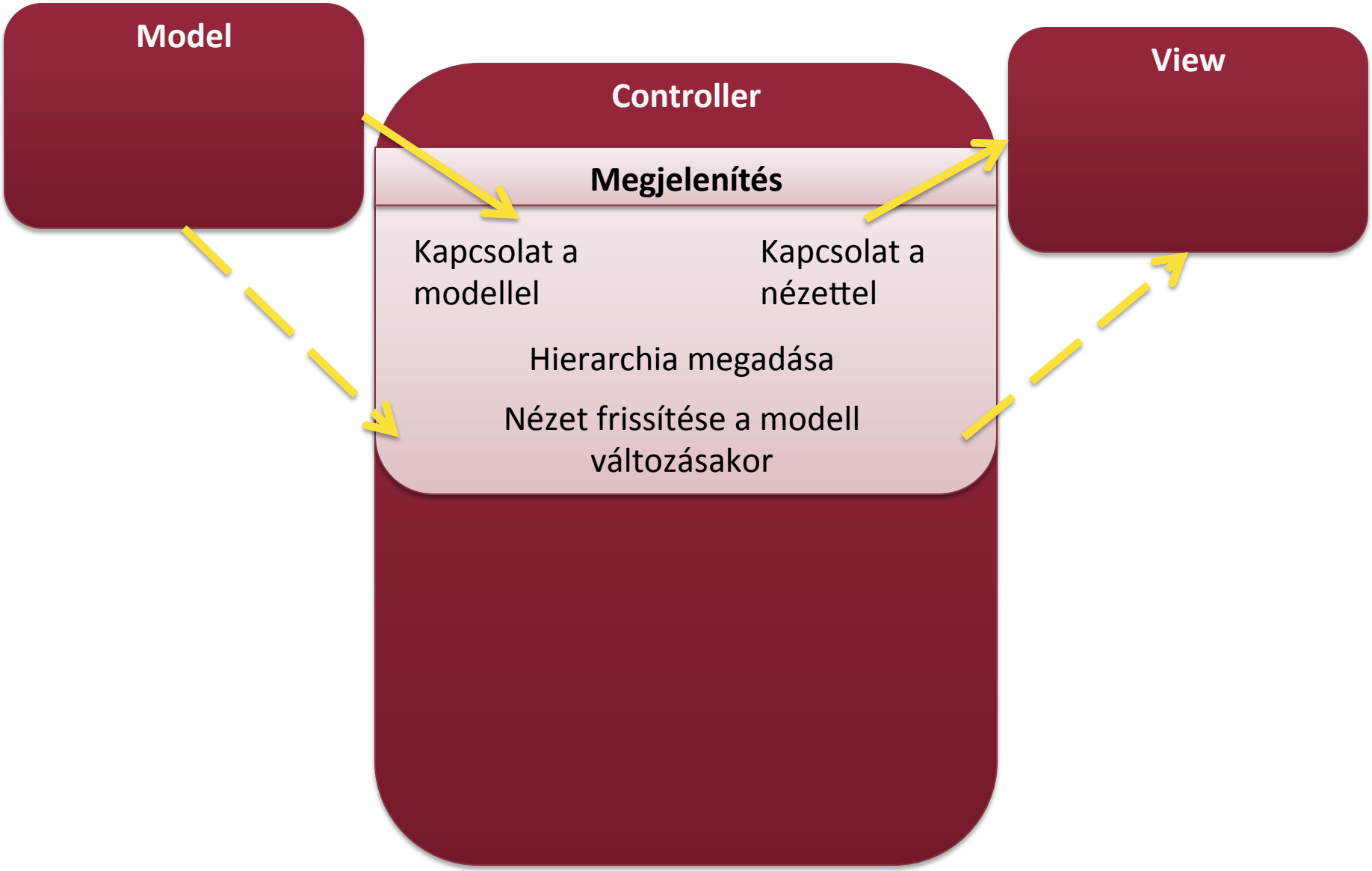
GEF workflow



GEF workflow



GEF workflow

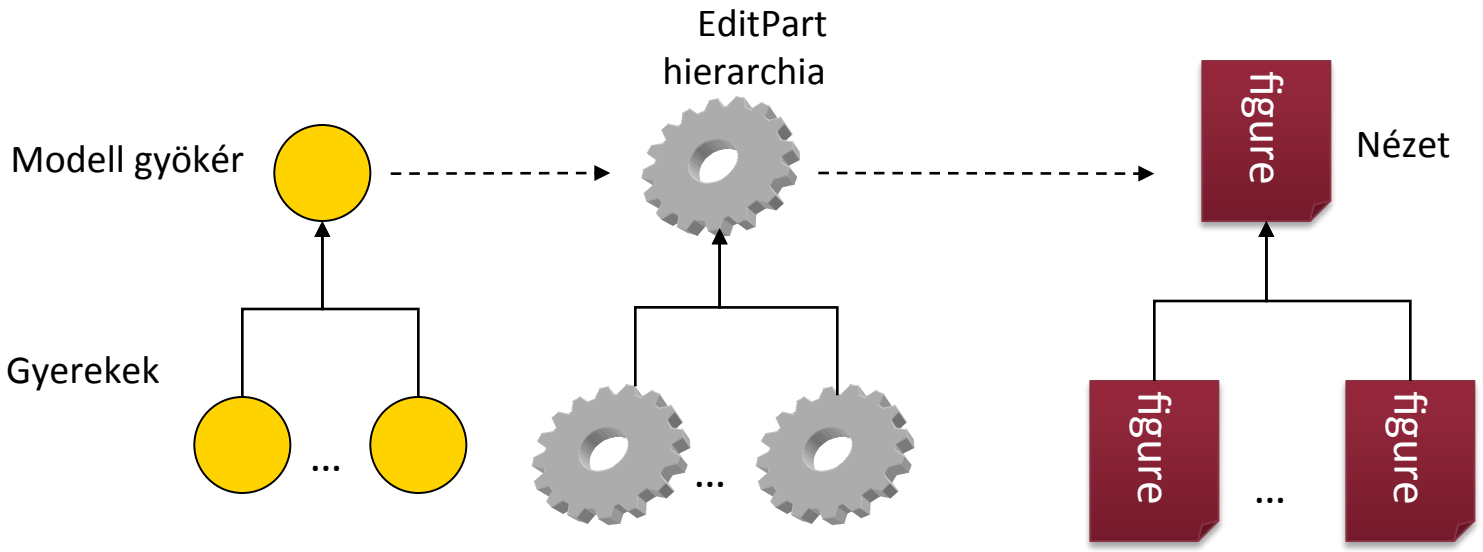


MVC a GEF-ben: Controller

- Vezérlő: EditPart osztályok
  - GEF „lelke”
  - Kapcsolat a modell és a nézet között
  - 1 Figure <-> 1 EditPart
  - 1 modell elem -> több EditPart is lehet
    - Több Figure
  - Nézet frissítése modell alapján
  - Felhasználói akciók kezelése
  - Modell módosítása ezek alapján

Kezdeti nézet felépítése

- Modell alapján EditPartok létrehozása
  - EditPartFactory
- Nézet Figure-ök példányosítása
  - GraphicalEditPart.createFigure()



EditPartFactory

- Modell alapján EditPart generálása

```
public class TestGEFEditPartFactory implements
  EditPartFactory {
  public EditPart createEditPart(EditPart context,
  Object model) {
      EditPart ep = null;

      if (model instanceof ElementModel)
        ep = new ElementEditPart();
      else if (model instanceof ParentModel)
        ep = new ParentEditPart();

      if (ep != null)
        ep.setModel(model);
      return ep;
    }
  }
```

EditPartFactory

- Modell alapján EditPart generálása

```
public class TestGEFEditPartFactory implements
  EditPartFactory {
  public EditPart createEditPart(EditPart context,
  Object model) {
      EditPart ep = null;

      if (model instanceof ElementModel)
        ep = new ElementEditPart();
      else if (model instanceof ParentModel)
        ep = new ParentEditPart();

      if (ep != null)
        ep.setModel(model);
      return ep;
    }
  }
```

Szülő EditPart

EditPartFactory

- Modell alapján EditPart generálása

```
public class TestGEFEditPartFactory implements
  EditPartFactory {
  public EditPart createEditPart(EditPart context,
  Object model) {
      EditPart ep = null;

      if (model instanceof ElementModel)
        ep = new ElementEditPart();
      else if (model instanceof ParentModel)
        ep = new ParentEditPart();

      if (ep != null)
        ep.setModel(model);
      return ep;
    }
  }
```

EditPartFactory

- Modell alapján EditPart generálása

```
public class TestGEFEditPartFactory implements
  EditPartFactory {
  public EditPart createEditPart(EditPart context,
  Object model) {
      EditPart ep = null;

      if (model instanceof ElementModel)
        ep = new ElementEditPart();
      else if (model instanceof ParentModel)
        ep = new ParentEditPart();

      if (ep != null)
        ep.setModel(model);
      return ep;
    }
  }
```

EditPart tárol egy modell referenciát

Nézet legenerálása

- EditPart feladata
- Sajátunkat célszerű származtatni az AbstractGraphicalEditPart osztályból

```
public class ElementEditPart extends
  AbstractGraphicalEditPart {
    ...
    @Override
    protected IFigure createFigure() {
      // Saját Figure létrehozása
      ElementFigure fig = new
  ElementFigure();
      return fig;
    }
    ...
  }
```

Modell bejárása

- GEF modell felépítése
  - EditPartFactory
  - Modell gyökér eleme
- Hogy jutunk el a modell többi részéhez?
  - EditParton keresztül
  - Mindenki megmondja a saját gyerekeit
  - Rekurzívan bejárható az egész modell
    - Ne legyen benne tartalmazás kör

Modell bejárása

- EditPart.getModelChildren()
  - Az EditParthoz tartozó modellelem gyerekeit kell visszaadni listaként
    - Lista sorrendje számít -> nézetek takarása

```
public class TestParentEditPart extends
  AbstractGraphicalEditPart {
    ...
    @Override
    protected List getModelChildren() {
      // Saját modell lekérdezése
      ParentModel pm = ((ParentModel)
  getModel());
      return pm.getChildren();
    }
    ...
  }
```

Modellfüggő, nem GEF-specifikus

Modell bejárása

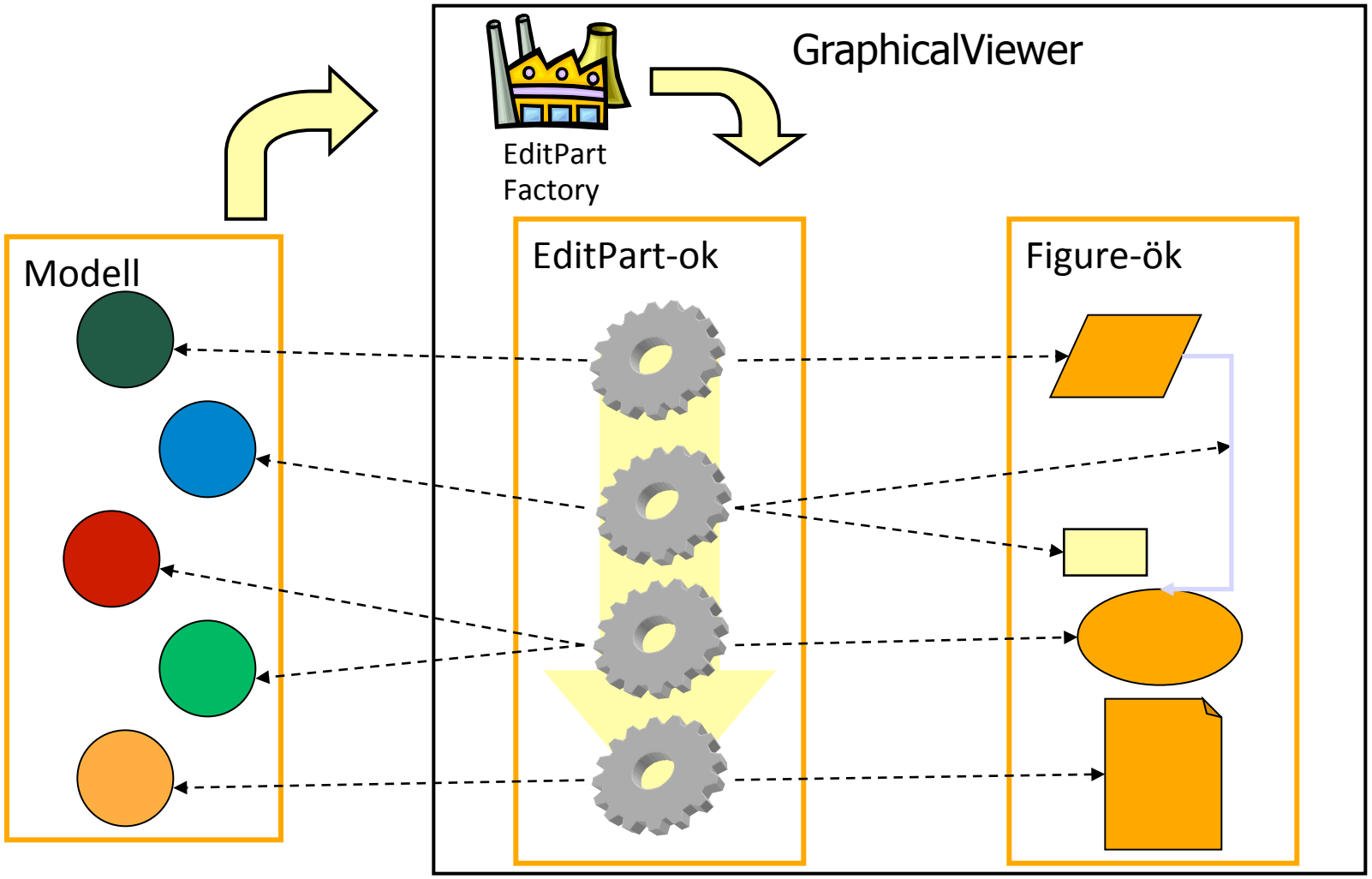
- EditPart.getModelChildren()
  - Az EditParthoz tartozó modellelem gyerekeit kell visszaadni listaként
    - Lista sorrendje számít -> nézetek takarása

```
public class TestParentEditPart extends
  AbstractGraphicalEditPart {
    ...
    @Override
    protected List getMode                    () {
      // Saját modell lekérdezése
      ParentModel pm = ((ParentModel)
  getModel());
      return pm.getChildren();
    }
    ...
  }
```

Modell lekérdezése

Modellfüggő, nem GEF-specifikus

Nézet felépítés - összefoglalás



ContentPane

- ContentPane: a gyerekekhez tartozó nézeteket tartalmazó nézet
- Felülírjuk, ha egy összetett Figure-nek csak egy része tartalmazza a gyerek-Figure-öket
- EditPartban írhatjuk felül (alapesetben maga a Figure)

- ...
- ```
  @Override
  public IFigure getContentPane() {
      return ((MyFigure)
  getFigure()).getPlaceOfChildren();
  }
  ...
  ```
  ```

Nézet frissítése modellváltozáskor

- Modell figyelése
  - activate(), deactivate()
- Nézet frissítése
  - refreshVisuals(): nem strukturális módosítás
  - refreshChildren(): gyerekek listája változik

EditPart példa

```
public class ParentEditPart extends AbstractGraphicalEditPart
      implements MyModelListener {

 protected void refreshVisuals() {

  ((ParentView) getFigure()).setLabel(

   ((ParentModel) getModel()).getName());

 }

 public void activate() {
   super.activate();
   ((ParentModel) getModel()).addListener(this);

 }

  public void deactivate() {
   ((ParentModel) getModel()).removeListener(this);
   super.deactivate();

 }

  public void modelChanged() {

   refreshVisuals();

   refreshChildren();

  }

}
```

EditPart példa

```
public class ParentEditPart extends AbstractGraphicalEditPart
      implements MyModelListener {

 protected void refreshVisuals() {

  ((ParentView) getFigure()).setLabel(

   ((ParentModel) getModel()).getName());

 }

 public void activate() {
   super.activate();
   ((ParentModel) getModel()).addListener(this);

 }

  public void deactivate() {
   ((ParentModel) getModel()).removeListener(this);
   super.deactivate();

 }

  public void modelChanged() {

   refreshVisuals();

   refreshChildren();

  }

}
```

Nézet frissítése

EditPart példa

```
public class ParentEditPart extends AbstractGraphicalEditPart
      implements MyModelListener {

 protected void refreshVisuals() {

  ((ParentView) getFigure()).setLabel(

   ((ParentModel) getModel()).getName());

 }

 public void activate() {
   super.activate();
   ((ParentModel) getModel()).addListener(this);

 }

  public void deactivate() {
   ((ParentModel) getModel()).removeListener(this);
   super.deactivate();

 }

  public void modelChanged() {

   refreshVisuals();

   refreshChildren();

  }

}
```

Modellfigyelés kezdete

EditPart példa

```
public class ParentEditPart extends AbstractGraphicalEditPart
      implements MyModelListener {

 protected void refreshVisuals() {

  ((ParentView) getFigure()).setLabel(

   ((ParentModel) getModel()).getName());

 }

 public void activate() {
   super.activate();
   ((ParentModel) getModel()).addListener(this);

 }

  public void deactivate() {
   ((ParentModel) getModel()).removeListener(this);
   super.deactivate();

 }

  public void modelChanged() {

   refreshVisuals();

   refreshChildren();

  }

}
```

Modellfigyelés vége

EditPart példa

```
public class ParentEditPart extends AbstractGraphicalEditPart
      implements MyModelListener {

 protected void refreshVisuals() {

  ((ParentView) getFigure()).setLabel(

   ((ParentModel) getModel()).getName());

 }

 public void activate() {
   super.activate();
   ((ParentModel) getModel()).addListener(this);

 }

  public void deactivate() {
   ((ParentModel) getModel()).removeListener(this);
   super.deactivate();

 }

  public void modelChanged() {

   refreshVisuals();

   refreshChildren();

  }

}
```

Modell ezt hívja

EditPart példa

```
public class ParentEditPart extends AbstractGraphicalEditPart
      implements MyModelListener {

 protected void refreshVisuals() {

  ((ParentView) getFigure()).setLabel(

   ((ParentModel) getModel()).getName());

 }

 public void activate() {
   super.activate();
   ((ParentModel) getModel()).addListener(this);

 }

  public void deactivate() {
   ((ParentModel) getModel()).removeListener(this);
   super.deactivate();

 }

  public void modelChanged() {

   refreshVisuals();

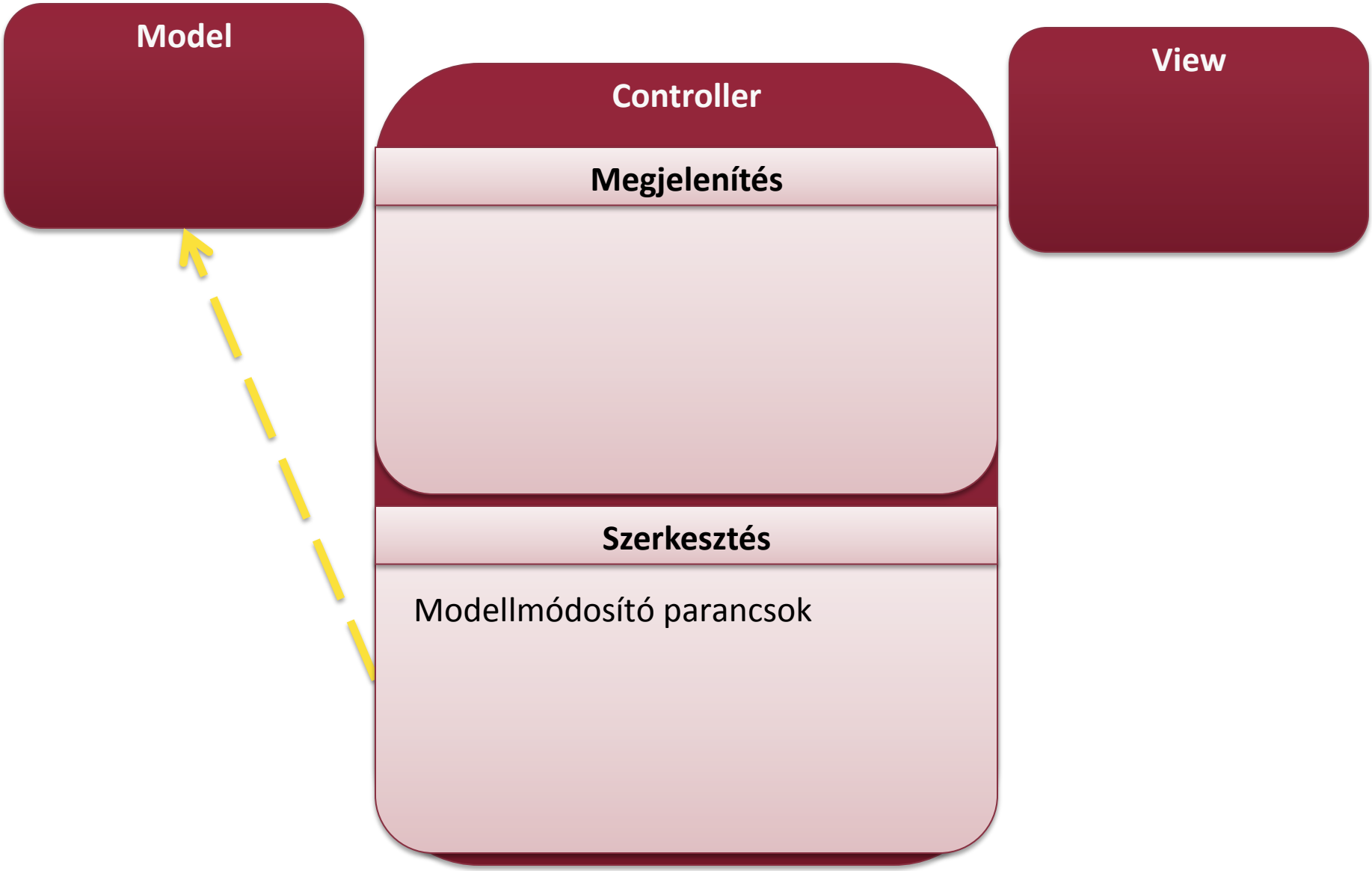
   refreshChildren();

  }

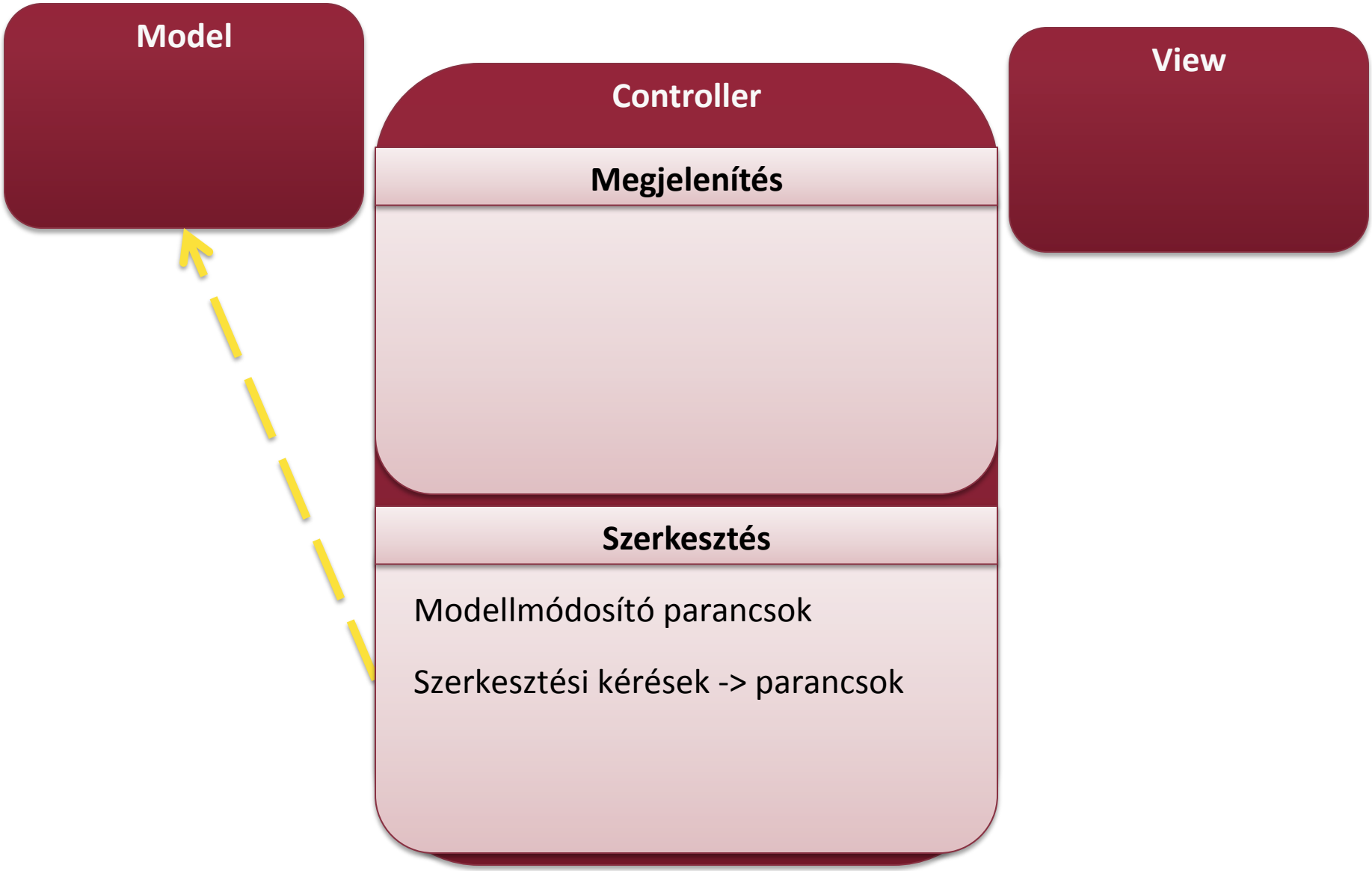
}
```

Gyerekek frissítése

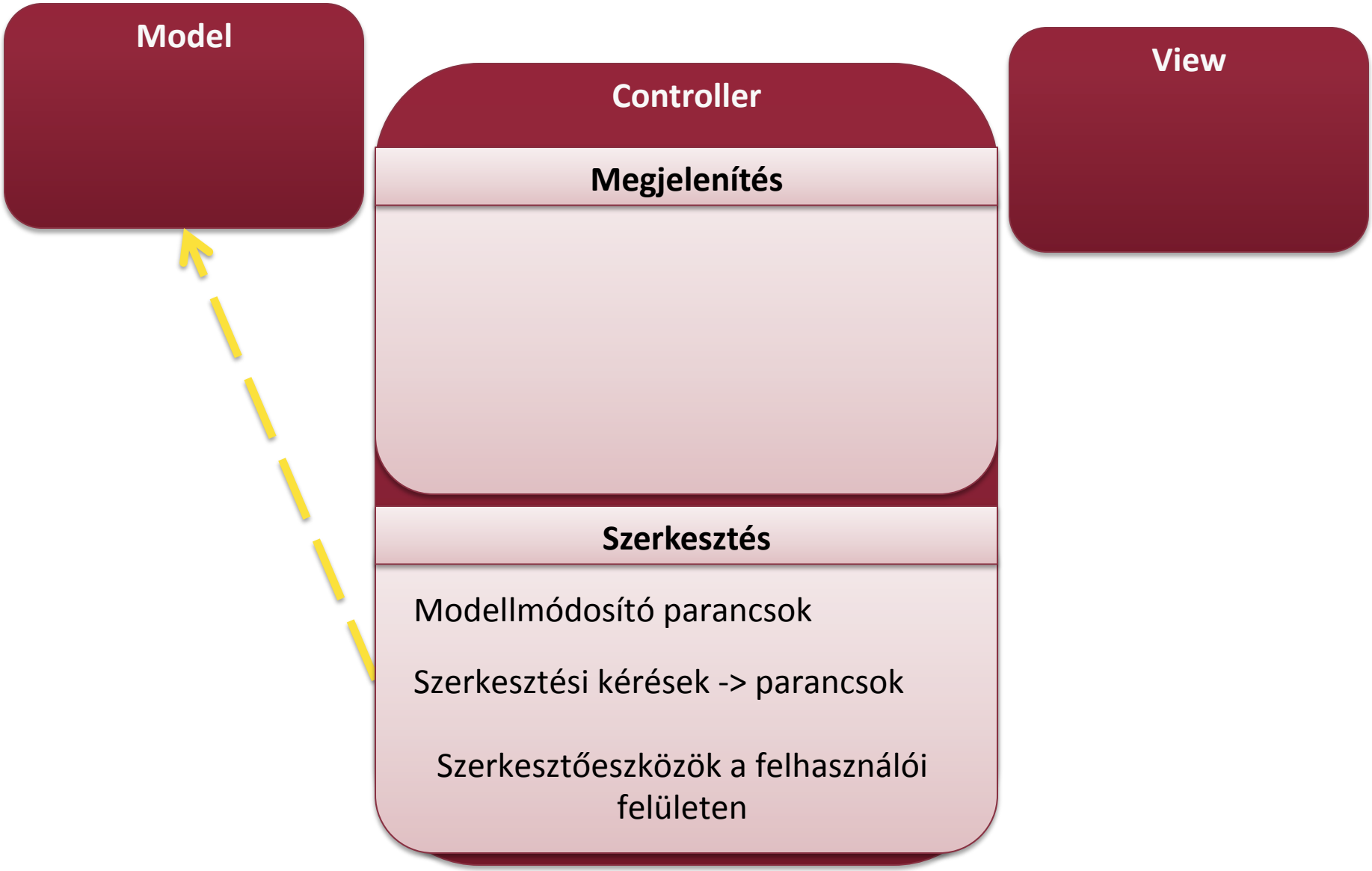
GEF workflow



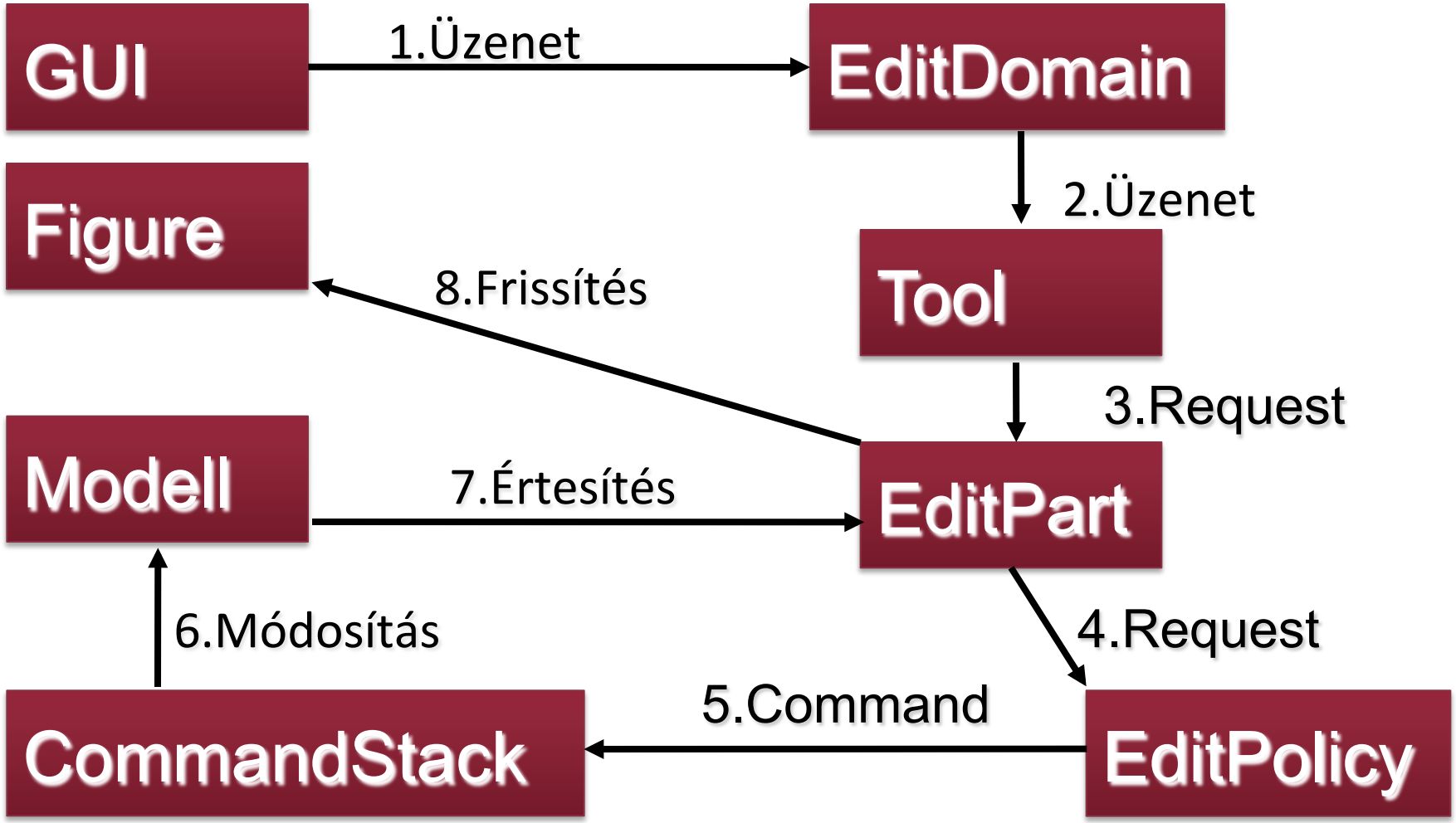
GEF workflow



GEF workflow



Szerkesztés folyamata



Szerkesztés szereplői I.

- EditDomain: fogadja a GUI eseményeket, és továbbítja az aktív Toolnak
  - Nem végez feldolgozást
- Tool: egy szerkesztési funkciót jelképez
  - Feldolgozza az GUI üzeneteket
  - Létrehoz egy (vagy több) Request-et
  - Pl. SelectionTool, CreationTool, MarqueeTool
  - Saját is készíthető

Szerkesztés szereplői II.

- Request
  - GEF-szintű esemény
  - Pl. CreateRequest, DeleteRequest, ChangeBoundsRequest
  - Továbbítódik a cél EditParthoz
- EditPolicy
  - EditParthoz tartozó „szerkesztési szabály”
  - Request -> Command leképzés
  - 1 EditPart -> több EditPolicy lehet

Szerkesztés szereplői III.

- EditPart
  - A saját EditPolicyjai segítségével átalakítja a bejövő Requestet egy Commandá
  - Észleli a modell változását az értesítési mechanizmuson keresztül
  - Modellváltozás esetén frissíti a nézetet, illetve a struktúrát

Szerkesztés szereplői IV.

- Command
  - A modell módosítását végzi
  - Visszavonható (ha megírjuk ☺)
- CommandStack
  - Végrehajtott Commandok verme
  - Ez biztosítja az undo/redo lehetőségét
  - EditDomainenként pontosan egy darab
  - Mindig ezen keresztül módosítsunk!

Szerkesztés szereplői V.

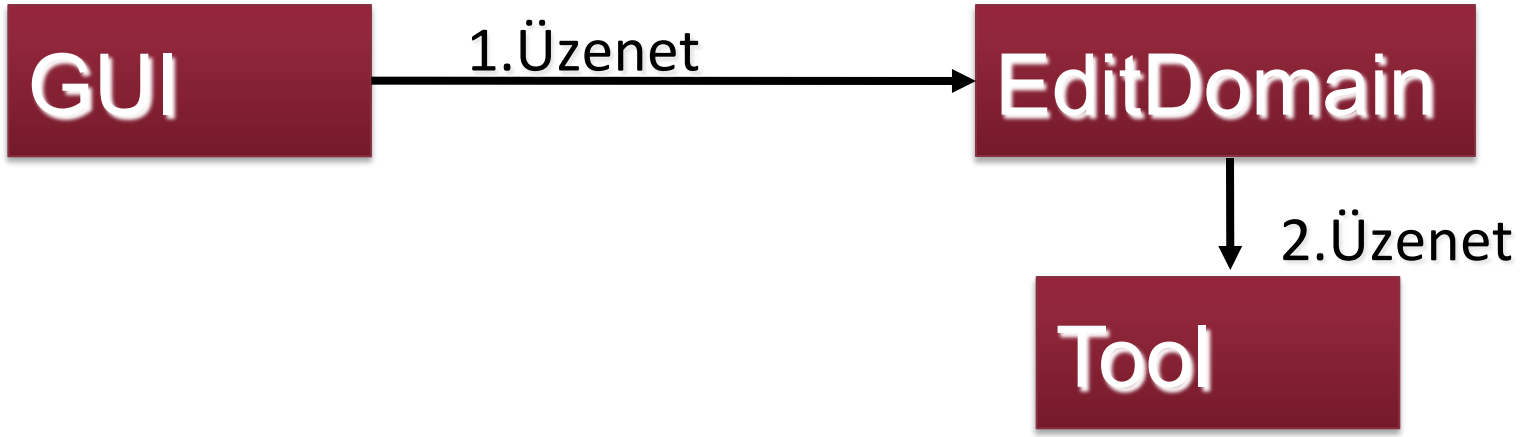
- Action
  - Nem GEF-specifikus (JFace)
  - Nem „grafikus” felhasználói akció
    - Menüelemek, billentyűlenyomások, toolbar elemek
  - GEF biztosít néhány wrappert, amik lehetővé teszik a CommandStack egyszerű elérését
  - ActionRegistry: actionök listája
    - Több helyen szereplő azonos actionökhöz
- Nincs több (lényeges) szereplő

Szerkesztés folyamata

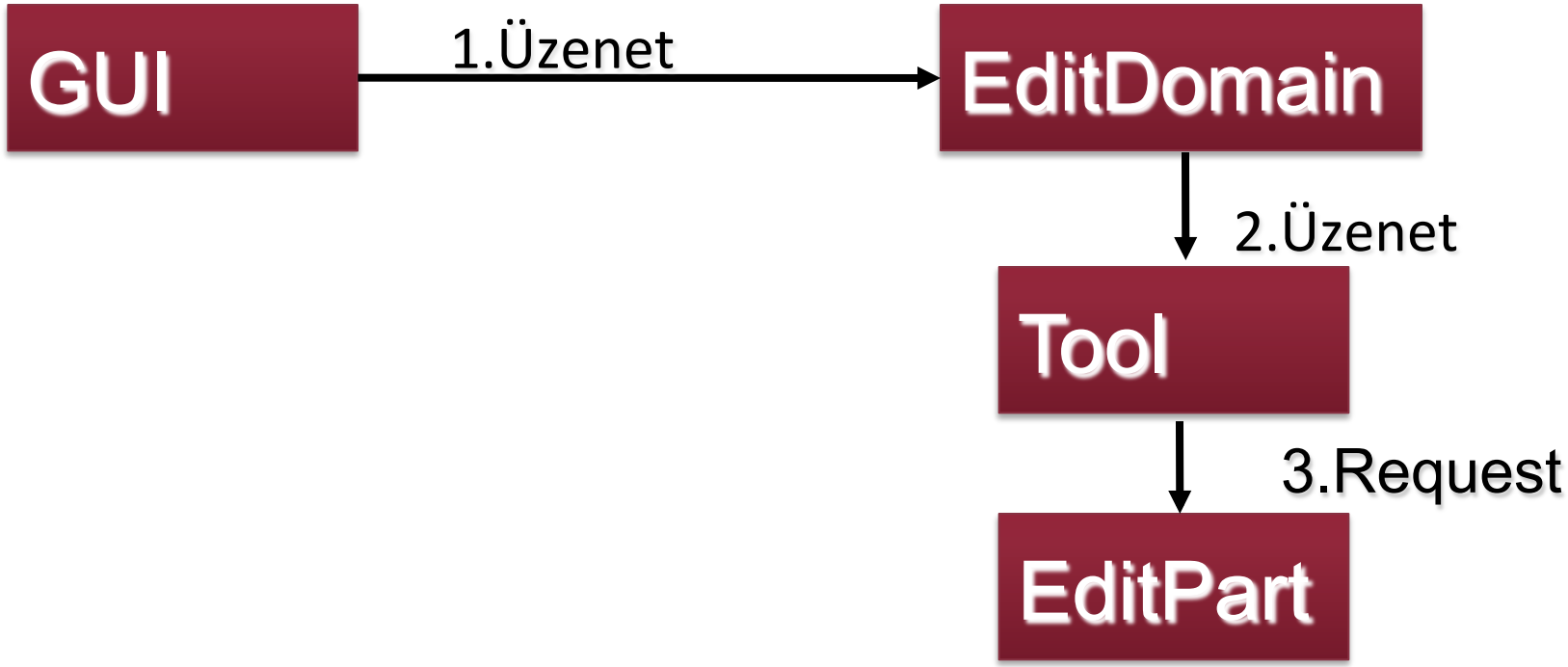
GUI

Szerkesztés folyamata

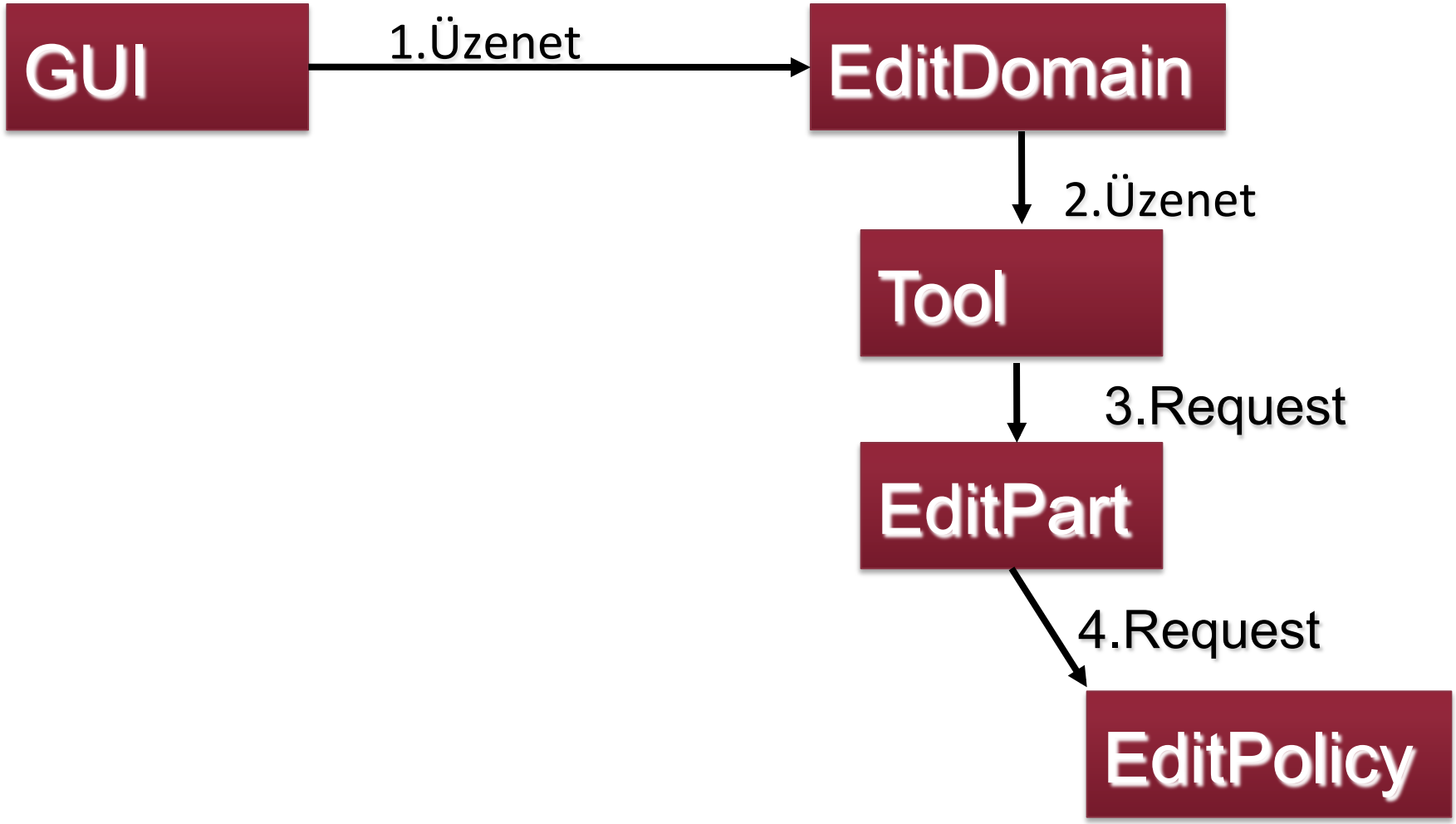
Szerkesztés folyamata



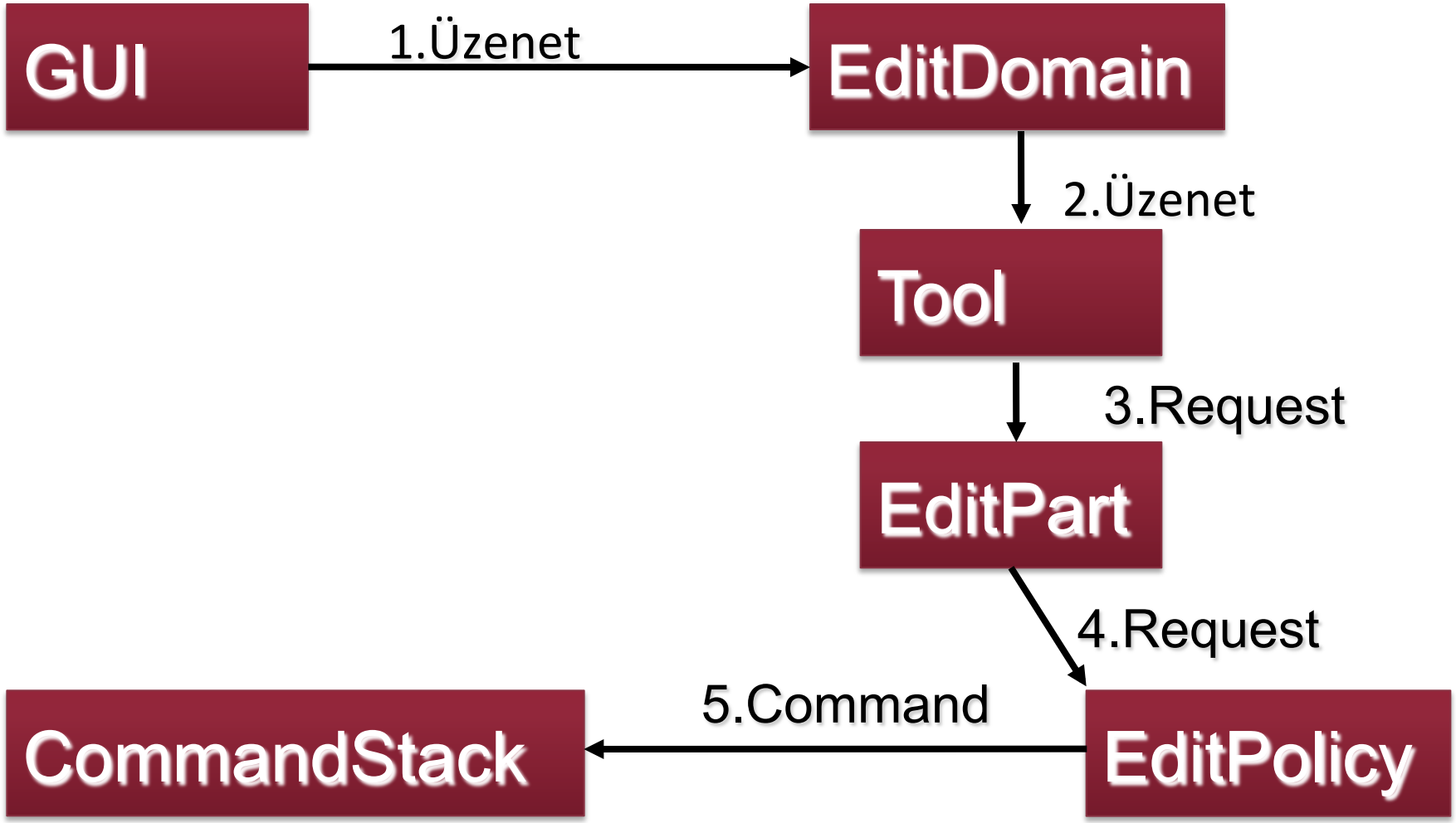
Szerkesztés folyamata



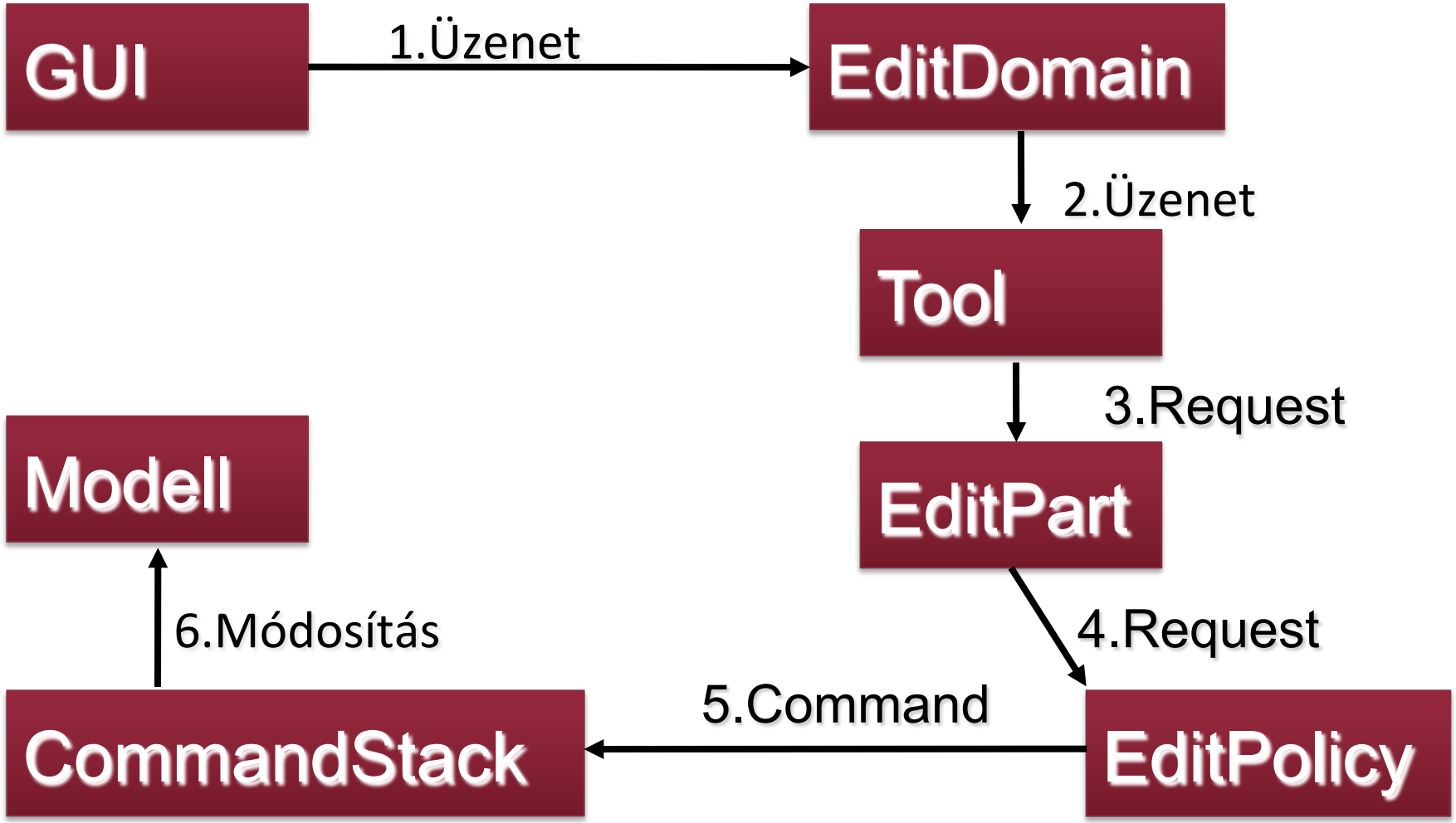
Szerkesztés folyamata



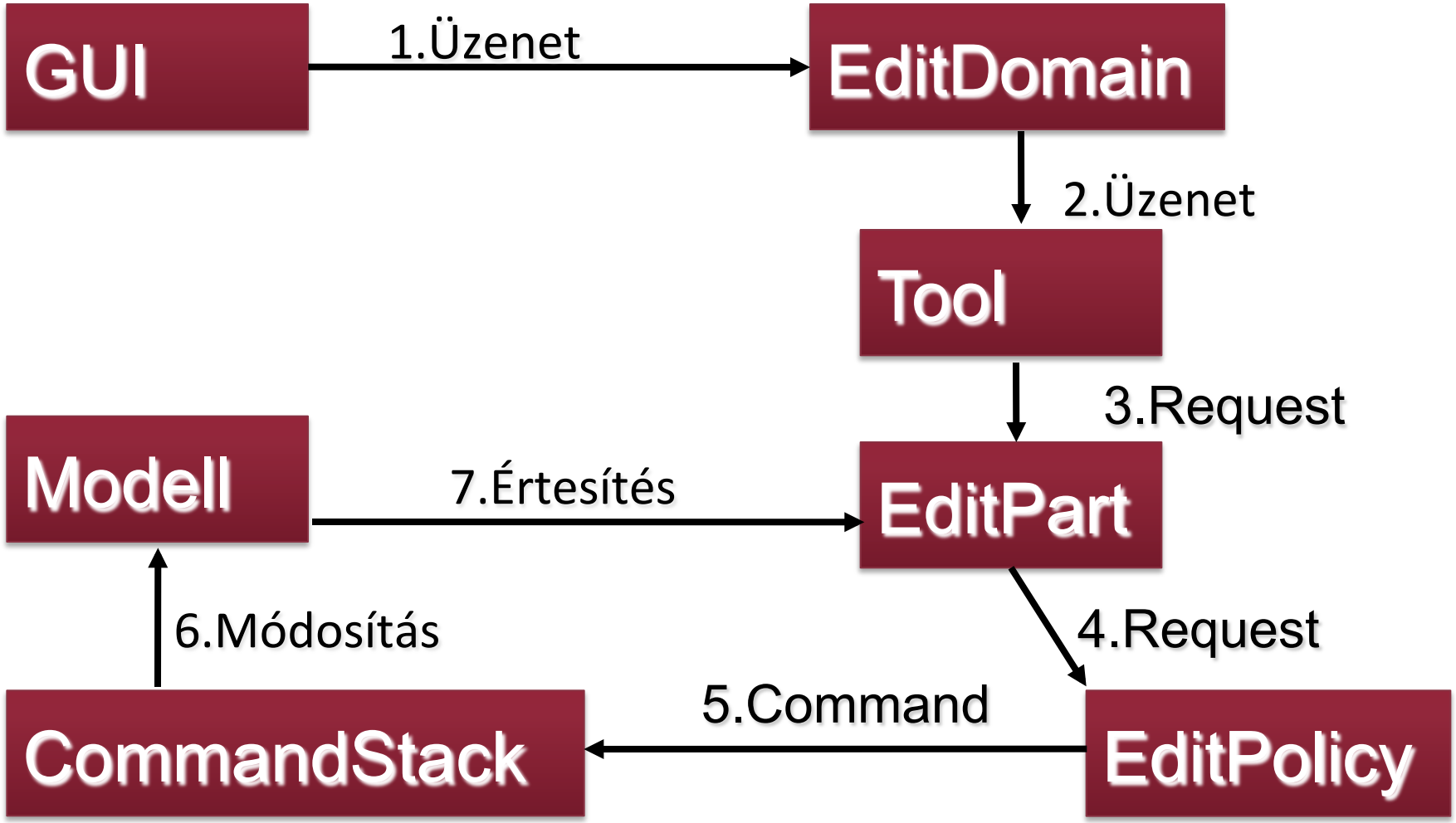
Szerkesztés folyamata



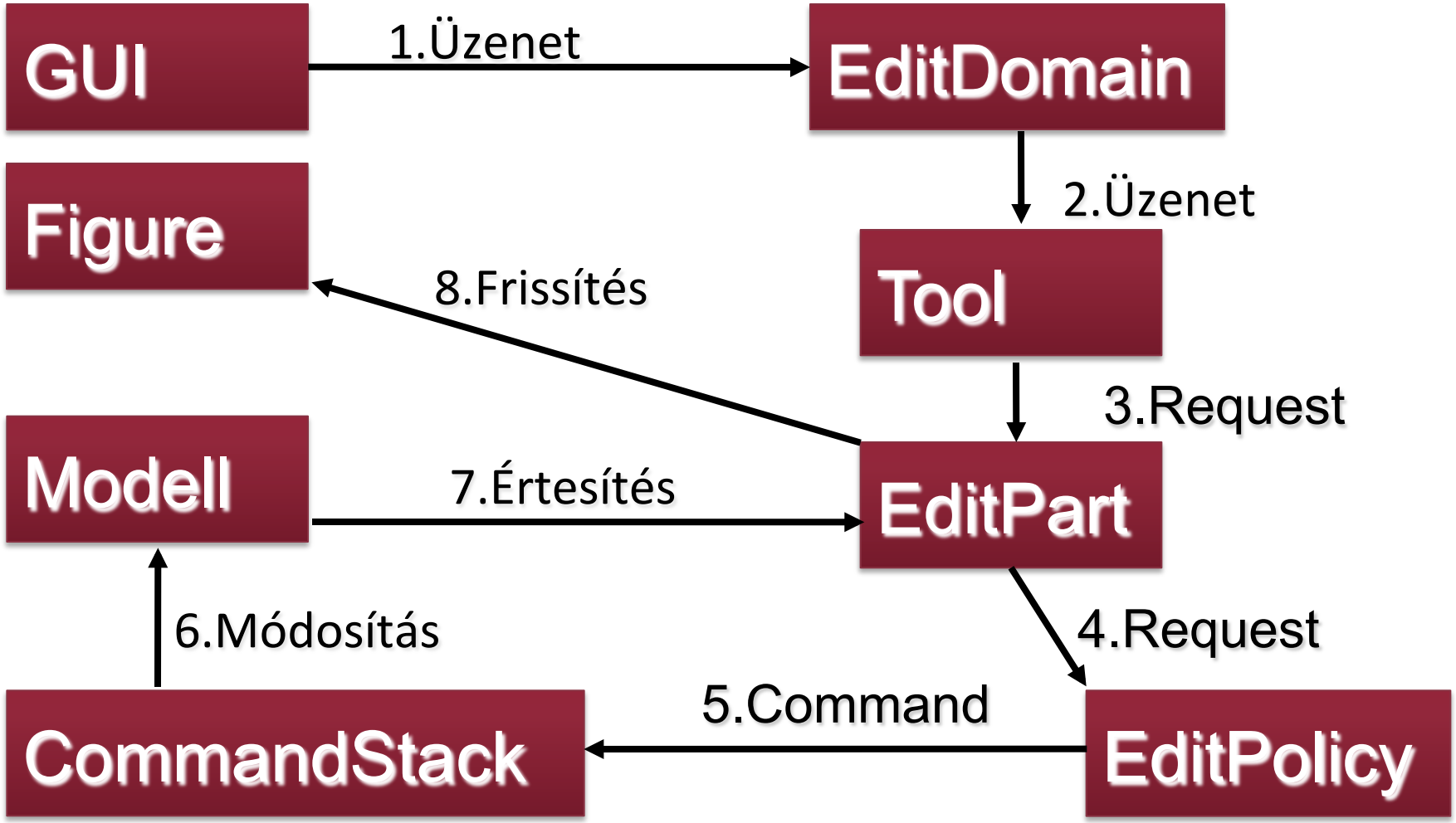
Szerkesztés folyamata



Szerkesztés folyamata



Szerkesztés folyamata



Mit kell nekünk megírni?

- Modell kód, értesítéssel
  - Generáltatható EMF segítségével
- Nézet osztályok
- EditPart osztályok 1.
  - Modell megjelenítés
    - createFigure(), refreshVisuals()
  - Modell változás figyelés
    - activate(), deactivate()
- EditPartFactory (modell -> EditPart)

Mit kell nekünk megírni?

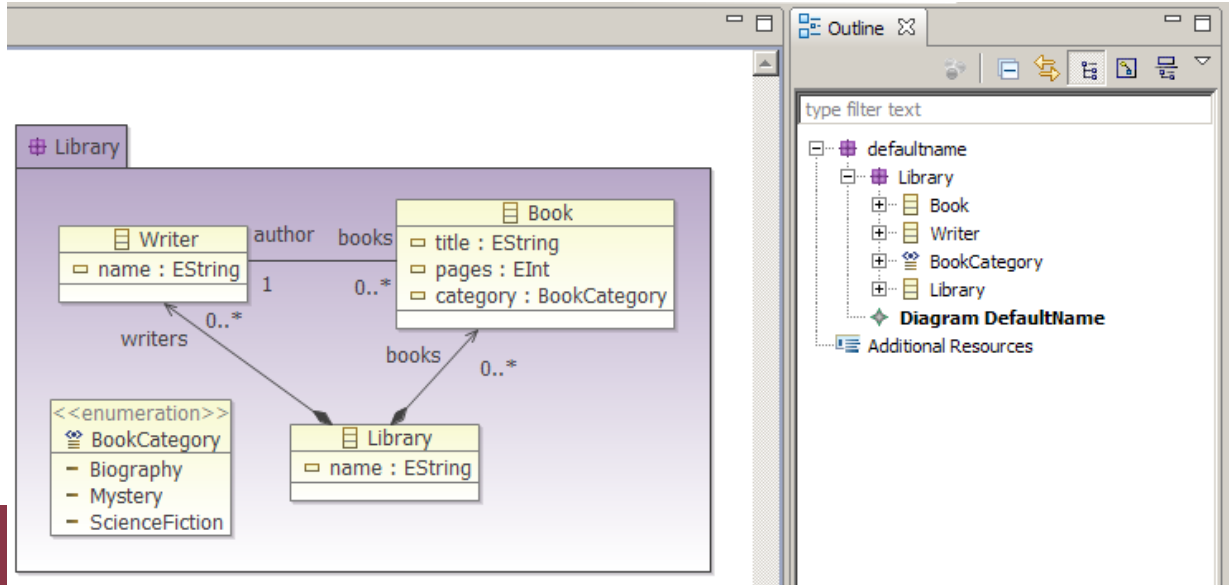
- Modell módosító Commandok
- Saját EditPolicy-k, amik a Commandokat használják
  - Milyen műveleteket engedünk meg
- EditPart oszályok 2.
  - EditPolicy-k hozzárendelése
- Editor és tartozékai
  - EditPartViewer, Palette

Editor készítése

- Feladatai
  - Létrehoz egy EditPartViewert
  - Kezeli a nem grafikus műveleteket
    - Actionök (undo/redo is ezek közé tartozik)
  - Létrehozza a menü és toolbar bejegyzéseket
    - ActionBarContributor
- Megoldás
  - Saját EditorPart, ezeket mi írjuk meg
  - GraphicalEditor használata
    - Egyszerű, prototípushoz jó

EditPartViewer

- Egy EditPart hierarchia megjelenítéséért felelős
- Elvben hasonló, mint a JFace viewerek
- Fa- vagy grafikus nézet
  - TreeViewer: tipikusan Outline nézethez
  - GraphicalViewer: grafikus nézet
    - ScrollingGraphicalViewer: javasolt megvalósítás



GraphicalEditor

- Ősosztály GEF-es Eclipse editorokhoz
  - Létrehoz egy ScrollingGraphicalViewer-t
  - Létrehoz néhány általános Actiont
    - Undo, redo, törlés, nyomtatás, mentés
    - Nem jeleníti meg őket sehol

GraphicalEditor használata

```
public class TestGEFEditor extends GraphicalEditor {
   public TestGEFEditor() {
       setEditDomain(new DefaultEditDomain(this));
   }

   protected void configureGraphicalViewer() {
       getGraphicalViewer().setEditPartFactory(
              new TestGEFEditPartFactory());
   }

   public void init(IEditorSite site, IEditorInput input)
                     throws PartInitException {
       super.init(site, input);
       // Modell felépítése az input alapján
   }

   protected void initializeGraphicalViewer() {
       getGraphicalViewer().setContents(modelRoot);
   }
       ...
   }
```

GraphicalEditor használata

EditDomain a konstruktorban

```
public class TestGEFEditor extends GraphicalEditor {
   public TestGEFEditor() {
       setEditDomain(new DefaultEditDomain(this));
   }

   protected void configureGraphicalViewer() {
       getGraphicalViewer().setEditPartFactory(
              new TestGEFEditPartFactory());
   }

   public void init(IEditorSite site, IEditorInput input)
                     throws PartInitException {
       super.init(site, input);
       // Modell felépítése az input alapján
   }

   protected void initializeGraphicalViewer() {
       getGraphicalViewer().setContents(modelRoot);
   }
       ...
   }
```

GraphicalEditor használata

```
public class TestGEFEditor extends GraphicalEditor {
   public TestGEFEditor() {
       setEditDomain(new DefaultEditDomain(th
   }

   protected void configureGraphicalViewer()
       getGraphicalViewer().setEditPartFactory(
             new TestGEFEditPartFactory());
   }

   public void init(IEditorSite site, IEditorInput input)
                     throws PartInitException {
       super.init(site, input);
       // Modell felépítése az input alapján
   }

   protected void initializeGraphicalViewer() {
       getGraphicalViewer().setContents(modelRoot);
   }
       ...
   }
```

EditPartFactory megadása

GraphicalEditor használata

```
public class TestGEFEditor extends GraphicalEditor {
   public TestGEFEditor() {
       setEditDomain(new DefaultEditDomain(this));
   }

   protected void configureGraphicalViewer() {
       getGraphicalViewer().setEditPartFactory
             new TestGEFEditPartFactory());
   }

   public void init(IEditorSite site, IEditor
                    throws PartInitException {
       super.init(site, input);
       // Modell felépítése az input alapján
   }

   protected void initializeGraphicalViewer() {
       getGraphicalViewer().setContents(modelRoot);
   }
       ...
   }
```

Megnyitott fájl feldolgozása (Eclipse editor)

GraphicalEditor használata

```
public class TestGEFEditor extends GraphicalEditor {
   public TestGEFEditor() {
       setEditDomain(new DefaultEditDomain(this));
   }

   protected void configureGraphicalViewer() {
       getGraphicalViewer().setEditPartFactory(
              new TestGEFEditPartFactory());
   }

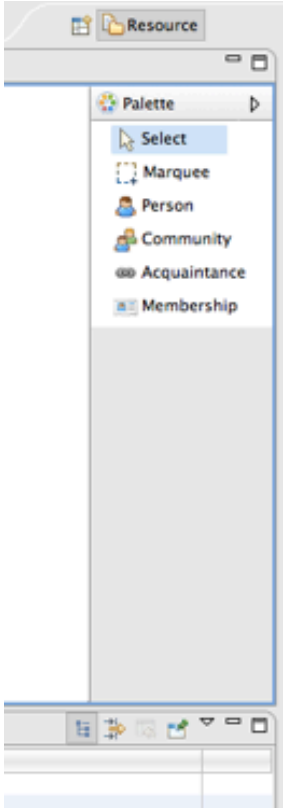
   public void init(IEditorSite site, IEditorInput input)
                     throws PartInitException {
       super.init(site, input);
       // Modell felépítése az input alapján
   }

   protected void initializeGraphicalViewer() {
       getGraphicalViewer().setContents(modelRoot);
   }
       ...
   }
```

Modell gyökérelem megadása

Eszköztár (Palette)

- Aktív eszköz váltása
- Eszközök grafikus megjelenítése
- Belül ez is egy külön GEF GraphicalViewer
- PaletteRoot: eszköztár gyökere
- PaletteEntry: eszköztár bejegyzés
  - PaletteContainer: eszközök csoportja
  - ToolEntry: egy konkrét eszköz



Gyakori ToolEntry-k

- SelectionToolEntry: kijelölés eszköz
- MarqueeToolEntry: csoportos kijelölés
- CreationToolEntry: elem létrehozása
  - Factory osztály, ez lekérdezhető a Requesten kereszül az EditPolicyban -> azonosítás
- Minden ToolEntry-hoz tartozik
  - Név, rövid leírás, kis/nagy ikon
  - Tool osztály, amit példányosít

GraphicalEditorWithPalette

- Olyan GraphicalEditor, ami létrehozza saját magának a palettát
- Palettához csak egy függvényt kell megírnunk
  - getPaletteRoot()

GraphicalEditorWithPalette példa

```
public class TestGEFEditor extends GraphicalEditorWithPalette {
   protected PaletteRoot getPaletteRoot() {
        PaletteRoot root = new PaletteRoot();
        PaletteGroup selectionToolGroup = new
                PaletteGroup("Selection");
        ToolEntry tool = new SelectionToolEntry();
        selectionToolGroup.add(tool);
        root.setDefaultEntry(tool);
        tool = new MarqueeToolEntry();
        selectionToolGroup.add(tool);
        root.add(selectionToolGroup);
        root.add(new PaletteSeparator());
        root.add(new CreationToolEntry("New Place",
               "Creates a new Petri net place",
               new SimpleFactory(PetriPlace.class),
               MyPlugin.getImageDescriptor("place.png"),
               MyPlugin.getImageDescriptor("place.png"));
           ...
           return root;
   }
   ...
}
```

GraphicalEditorWithPale

Új eszköztár

```
public class TestGEFEditor extends Graphica                te {
   protected PaletteRoot getPaletteRoot() {
        PaletteRoot root = new PaletteRoot();
        PaletteGroup selectionToolGroup = new
                PaletteGroup("Selection");
        ToolEntry tool = new SelectionToolEntry();
        selectionToolGroup.add(tool);
        root.setDefaultEntry(tool);
        tool = new MarqueeToolEntry();
        selectionToolGroup.add(tool);
        root.add(selectionToolGroup);
        root.add(new PaletteSeparator());
        root.add(new CreationToolEntry("New Place",
                "Creates a new Petri net place",
                new SimpleFactory(PetriPlace.class),
                MyPlugin.getImageDescriptor("place.png"),
                MyPlugin.getImageDescriptor("place.png"));
            ...
            return root;
   }
   ...
}
```

GraphicalEditorWithPalette példa

```
public class TestGEFEditor extends GraphicalEditorWithPalette {
   protected PaletteRoot getPaletteRoot() {
        PaletteRoot root = new PaletteRoot();
        PaletteGroup selectionToolGroup = new
                PaletteGroup("Selection");
        ToolEntry tool = new SelectionToolEntry();
        selectionToolGroup.add(tool);
        root.setDefaultEntry(tool);
        tool = new MarqueeToolEntry();
        selectionToolGroup.add(tool);
        root.add(selectionToolGroup);
        root.add(new PaletteSeparator());
        root.add(new CreationToolEntry("New Place",
                "Creates a new Petri net place",
                new SimpleFactory(PetriPlace.class),
                MyPlugin.getImageDescriptor("place.png"),
                MyPlugin.getImageDescriptor("place.png"));
            ...
            return root;
   }
   ...
}
```

Új csoport

GraphicalEditorWithPalette példa

```
public class TestGEFEditor extends GraphicalEditorWithPalette {
   protected PaletteRoot getPaletteRoot() {
        PaletteRoot root = new PaletteRoot();
        PaletteGroup selectionToolGroup = new
                PaletteGroup("Selection");
        ToolEntry tool = new SelectionToolEntry();
        selectionToolGroup.add(tool);
        root.setDefaultEntry(tool);
        tool = new MarqueeToolEntry();
        selectionToolGroup.add(tool);
        root.add(selectionToolGroup);
        root.add(new PaletteSeparator());
        root.add(new CreationToolEntry("New Place",
               "Creates a new Petri net place",
               new SimpleFactory(PetriPlace.class),
               MyPlugin.getImageDescriptor("place.png"),
               MyPlugin.getImageDescriptor("place.png"));
           ...
           return root;
   }
   ...
}
```

GraphicalEditorWithPalette példa

```
public class TestGEFEditor extends GraphicalEditorWithPalette {
   protected PaletteRoot getPaletteRoot() {
        PaletteRoot root = new PaletteRoot();
        PaletteGroup selectionToolGroup = new
                PaletteGroup("Selection");
        ToolEntry tool = new SelectionToolEntry();
        selectionToolGroup.add(tool);
        root.setDefaultEntry(tool);
        tool = new MarqueeToolEntry();
        selectionToolGroup.add(tool);
        root.add(selectionToolGroup);
        root.add(new PaletteSeparator());
        root.add(new CreationToolEntry("New Place",
               "Creates a new Petri net place",
               new SimpleFactory(PetriPlace.class),
               MyPlugin.getImageDescriptor("place.png"),
               MyPlugin.getImageDescriptor("place.png"));
           ...
           return root;
   }
   ...
}
```

Elválasztó vonal

GraphicalEditorWithPalette példa

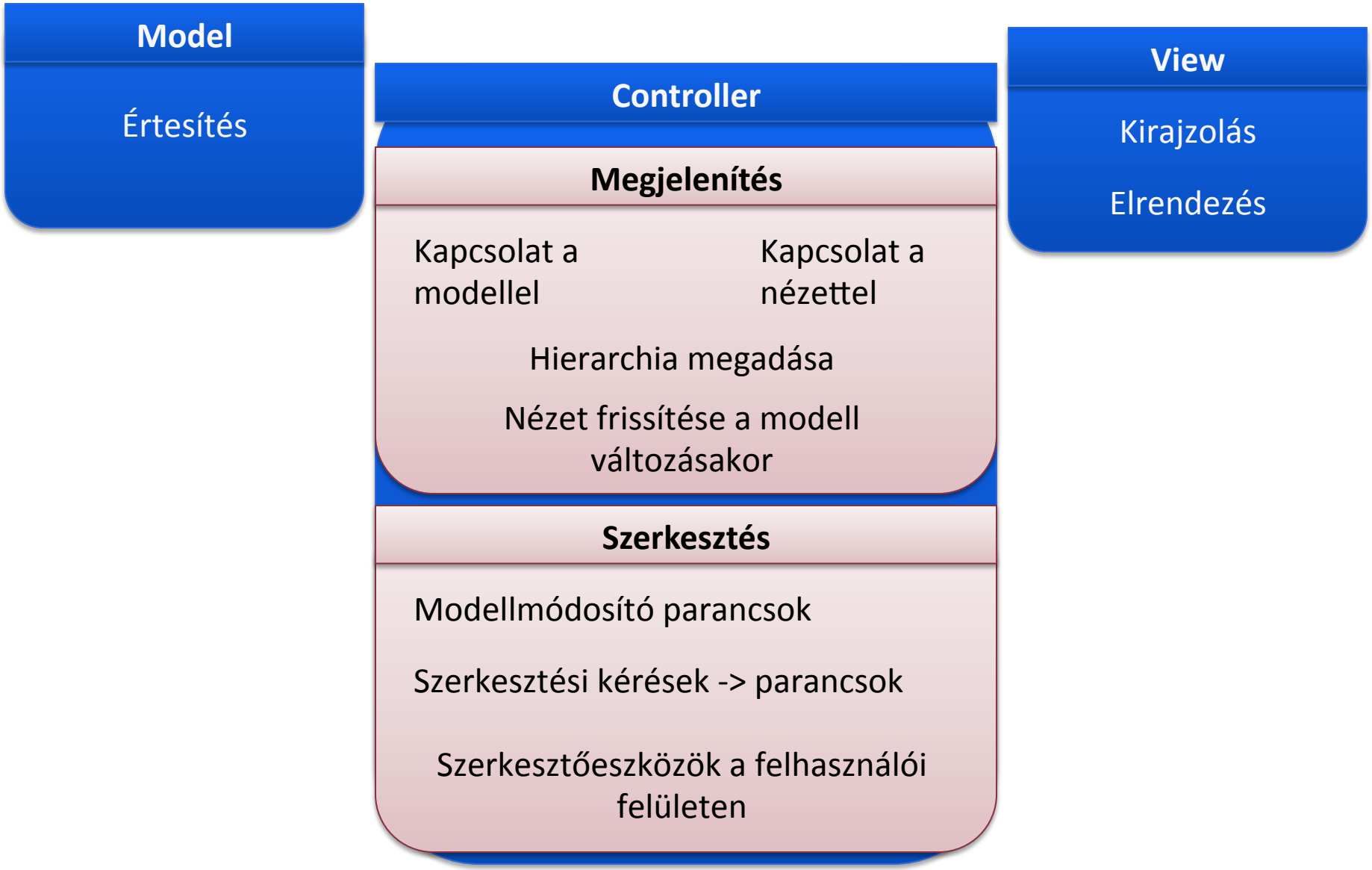
```
public class TestGEFEditor extends GraphicalEditorWithPalette {
   protected PaletteRoot getPaletteRoot() {
        PaletteRoot root = new PaletteRoot();
        PaletteGroup selectionToolGroup = new
                PaletteGroup("Selection");
        ToolEntry tool = new SelectionToolEntry();
        selectionToolGroup.add(tool);
        root.setDefaultEntry(tool);
        tool = new MarqueeToolEntry();
        selectionToolGroup.add(tool);
        root.add(selectionToolGroup);
        root.add(new PaletteSeparator());
        root.add(new CreationToolEntry("New Place",
                "Creates a new Petri net place",
                new SimpleFactory(PetriPlace.class),
                MyPlugin.getImageDescriptor("place.png"),
                MyPlugin.getImageDescriptor("place.p  "));
            ...
            return root;
   }
   ...
}
```

Factory a tool létrehozáshoz

GraphicalEditorWithPalette példa

```
public class TestGEFEditor extends GraphicalEditorWithPalette {
   protected PaletteRoot getPaletteRoot() {
        PaletteRoot root = new PaletteRoot();
        PaletteGroup selectionToolGroup = new
                PaletteGroup("Selection");
        ToolEntry tool = new SelectionToolEntry();
        selectionToolGroup.add(tool);
        root.setDefaultEntry(tool);
        tool = new MarqueeToolEntry();
        selectionToolGroup.add(tool);
        root.add(selectionToolGroup);
        root.add(new PaletteSeparator());
        root.add(new CreationToolEntry("New Place",
               "Creates a new Petri net place",
               new SimpleFactory(PetriPlace.class),
               MyPlugin.getImageDescriptor("place.png"),
               MyPlugin.getImageDescriptor("place.png"));
           ...
           return root;
   }
   ...
}
```

GEF workflow



GEF workflow

Model

Értesítés

Controller

Megjelenítés

Kapcsolat a modellel

Kapcsolat a nézettel

Hierarchia megadása

Nézet frissítése a modell változásakor

Szerkesztés

Modellmódosító parancsok

Szerkesztési kérések -> parancsok

Szerkesztőeszközök a felhasználói felületen

View

Kirajzolás

Elrendezés

EditPartFactory

GEF w

Figure

Model

Értesítés

Controller

Megjelenítés

Kapcsolat a modellel

Kapcsolat a nézettel

Hierarchia megadása

Nézet frissítése a modell változásakor

Szerkesztés

Modellmódosító parancsok

Szerkesztési kérések -> parancsok

Szerkesztőeszközök a felhasználói felületen

View

Kirajzolás

Elrendezés

EditPartFactory

GEF w

Figure

Model

Értesítés

Controller

Megjelenítés

Kapcsolat a modellel

Kapcsolat a nézettel

Hierarchia megadása

Nézet frissítése a modell változásakor

Szerkesztés

Modellmódosító parancsok

Szerkesztési kérések -> parancsok

Szerkesztőeszközök a felhasználói felületen

View

Kirajzolás

Elrendezés

EditPartFactory

LayoutManager

GEF w

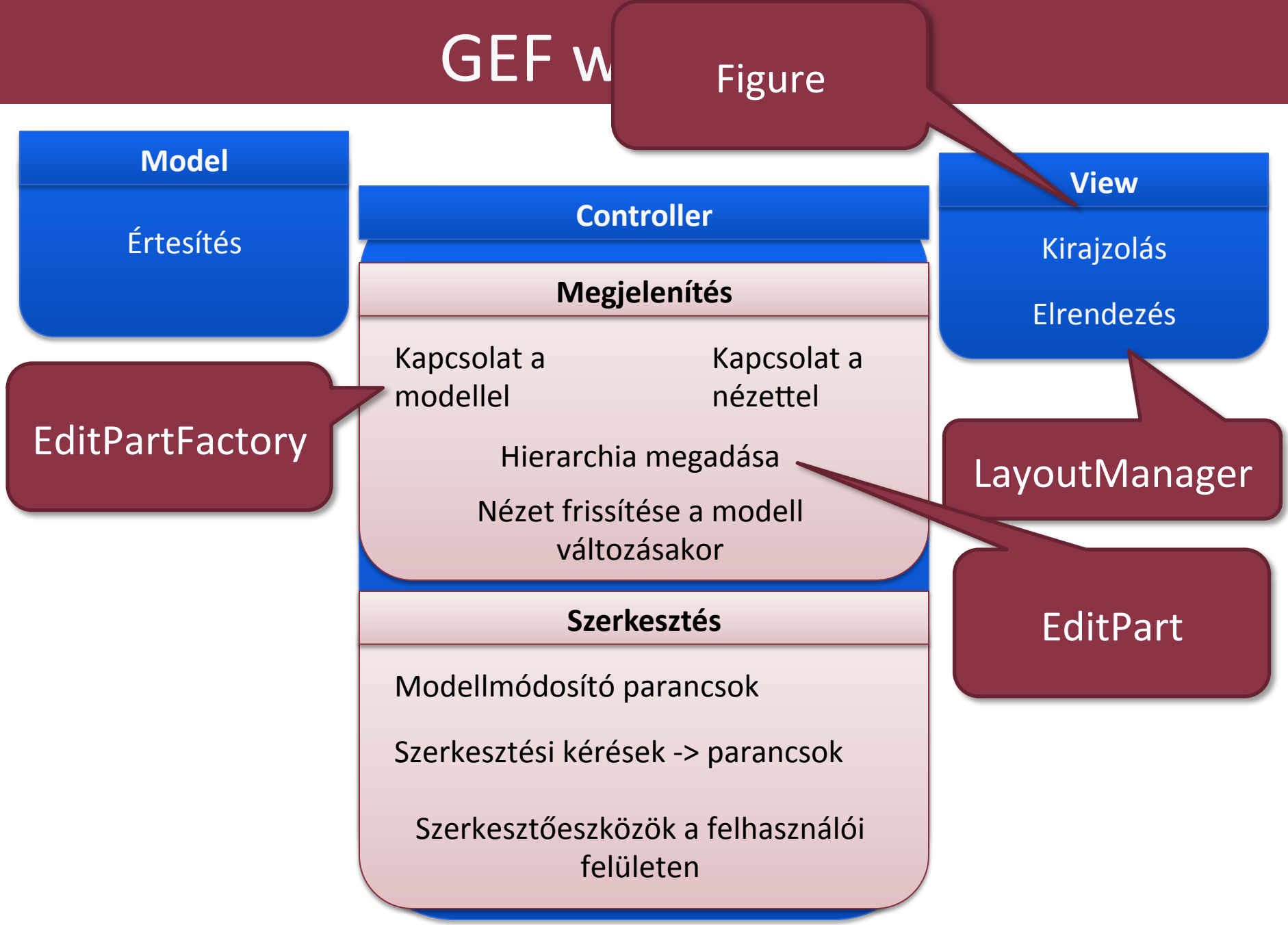


Figure
Model
Értesítés
Controller
Megjelenítés
Kapcsolat a modellel
Kapcsolat a nézettel
Hierarchia megadása
Nézet frissítése a modell változásakor
Szerkesztés
Modellmódosító parancsok
Szerkesztési kérések -> parancsok
Szerkesztőeszközök a felhasználói felületen
View
Kirajzolás
Elrendezés
EditPartFactory
LayoutManager
EditPart

GEF w

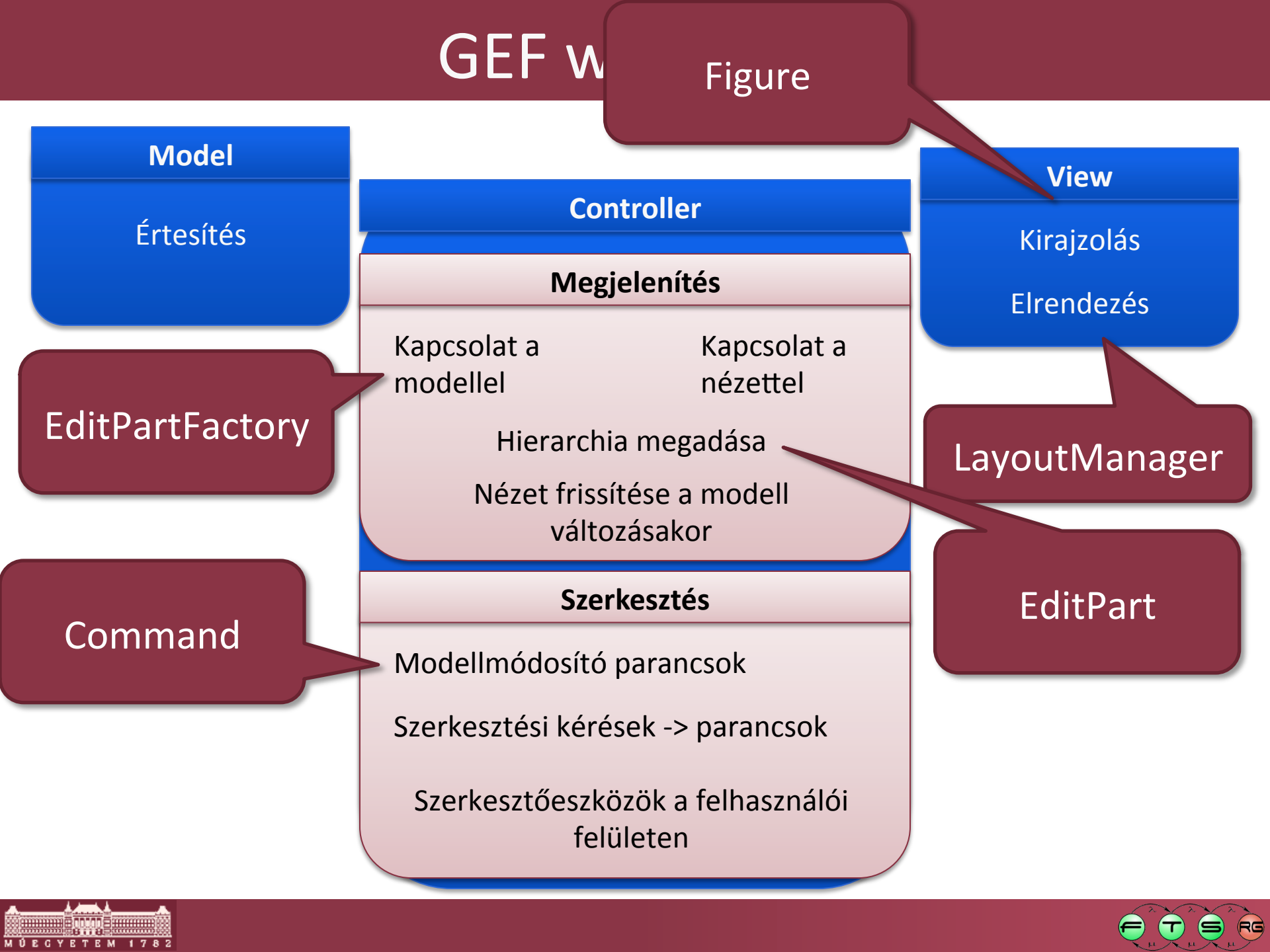


Figure
Model
Értesítés
Controller
Megjelenítés
Kapcsolat a modellel
Kapcsolat a nézettel
Hierarchia megadása
Nézet frissítése a modell változásakor
Szerkesztés
Modellmódosító parancsok
Szerkesztési kérések -> parancsok
Szerkesztőeszközök a felhasználói felületen
View
Kirajzolás
Elrendezés
EditPartFactory
LayoutManager
Command
EditPart

GEF w

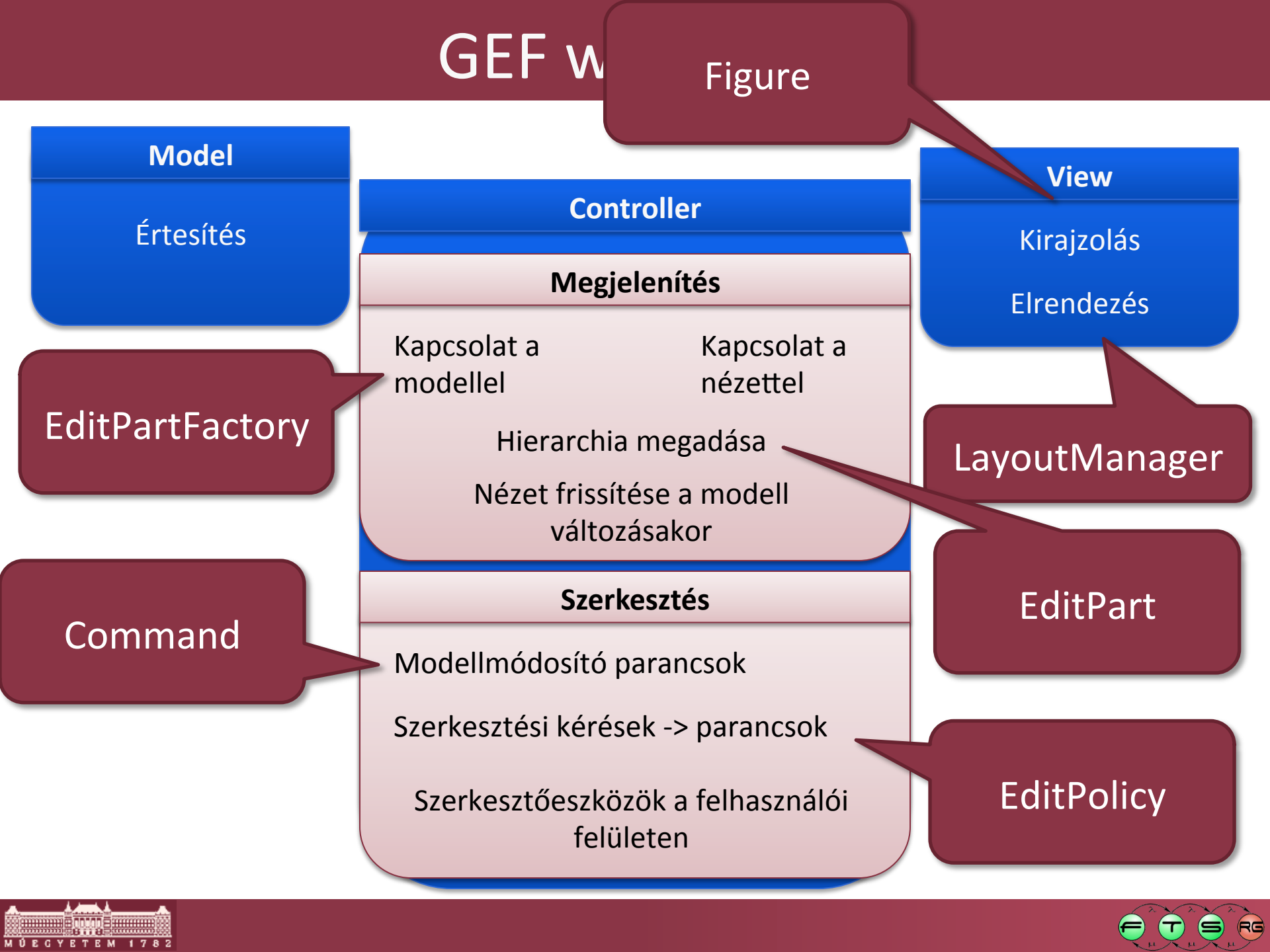


Figure
Model
Értesítés
Controller
Megjelenítés
Kapcsolat a modellel
Kapcsolat a nézettel
Hierarchia megadása
Nézet frissítése a modell változásakor
Szerkesztés
Modellmódosító parancsok
Szerkesztési kérések -> parancsok
Szerkesztőeszközök a felhasználói felületen
View
Kirajzolás
Elrendezés
EditPartFactory
LayoutManager
EditPart
Command
EditPolicy

GEF w

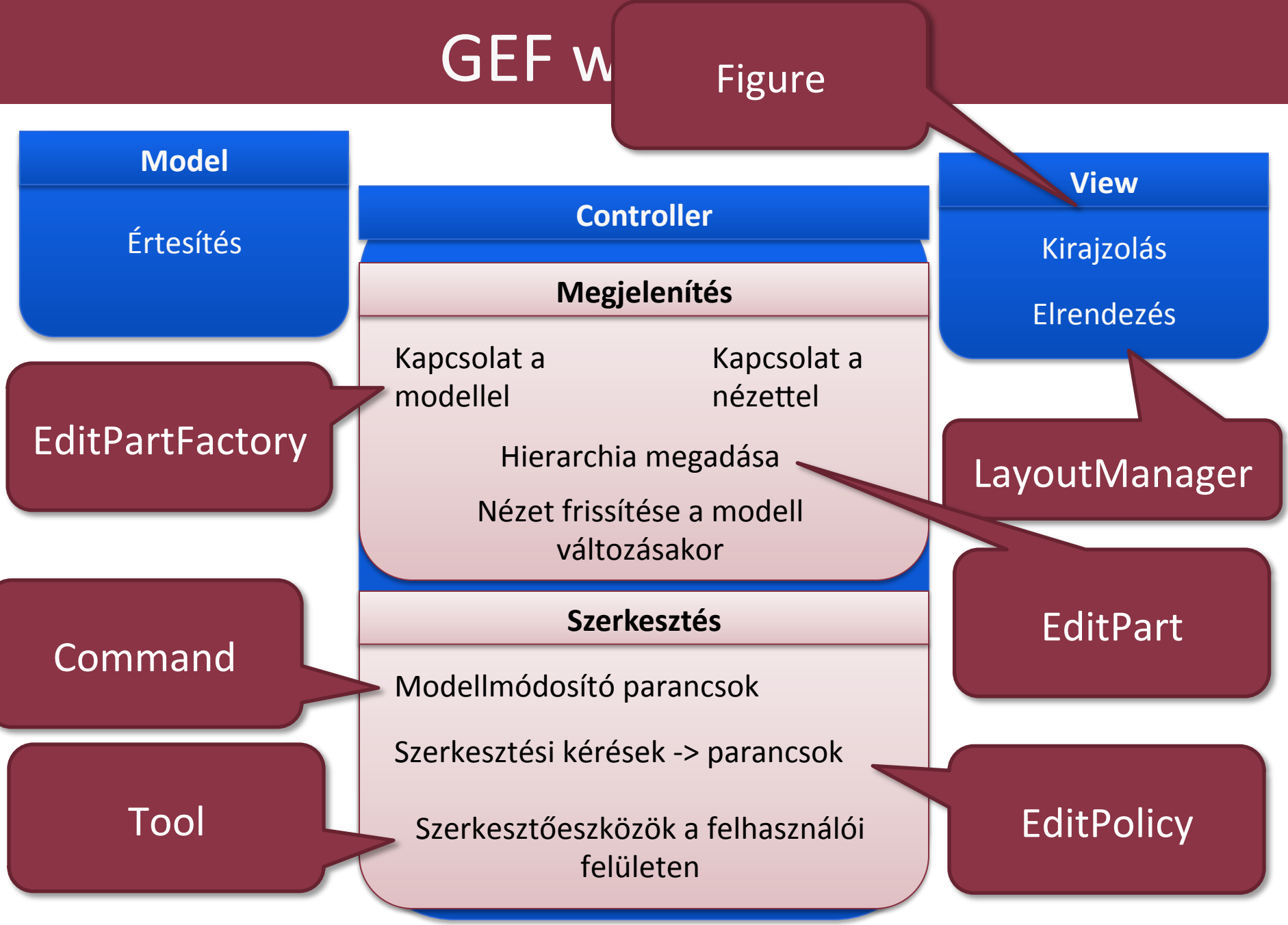


Figure
Model
Értesítés
Controller
Megjelenítés
Kapcsolat a modellel
Kapcsolat a nézettel
Hierarchia megadása
Nézet frissítése a modell változásakor
Szerkesztés
Modellmódosító parancsok
Szerkesztési kérések -> parancsok
Szerkesztőeszközök a felhasználói felületen
View
Kirajzolás
Elrendezés
EditPartFactory
LayoutManager
EditPart
Command
EditPolicy
Tool

Nyilak (összekötők)

- Hasonlóak a normál objektumokhoz
  - DE: fontos különbségek
- Megjelenítés külön (felsőbb) rétegben
- Van saját EditPart
  - AbstractConnectionEditPartból származik
  - Saját EditPolicy-k, Requestek, stb.
- Irányítottak (modell szinten)

- *Most nem részletezzük*
  - *Feltöltött fóliákban teljesség kedvéért*

További lehetőségek

- Modell tulajdonságok szerkesztése az Eclipse *Properties* nézetében
- Szövegek (címkék) szerkesztése közvetlenül a rajzon (direct editing)
- Nagyítási lehetőség
- Igazítás
- Különálló fa és áttekintő modellnézet

- *Most nem részletezzük*
  - *Feltöltött fóliákban teljesség kedvéért*

GEF – További források

- Create an Eclipse-based application using the Graphical Editing Framework
  - IBM Developerworks: http://www.ibm.com/developerworks/library/os-eclipse-gef11/
- Vainolo tutorials:
  - http://www.vainolo.com/tutorials/
- GEF wiki:
  - http://wiki.eclipse.org/GEF/Articles%2C_Tutorials%2C_Slides

Graphiti

Mi a gond a GEF használatával?

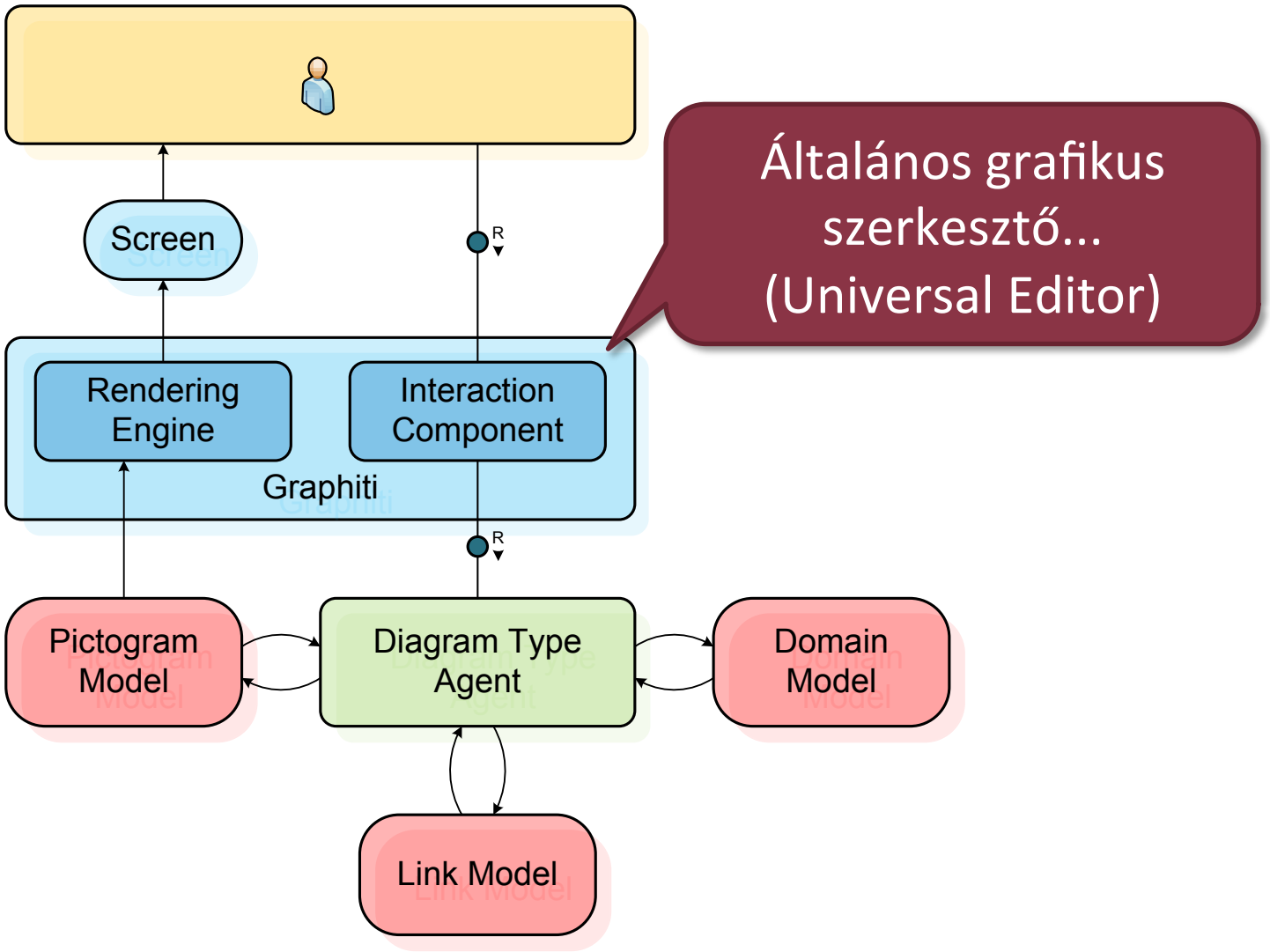
- Bonyolult
- Sok kézzel írt kód

- Egyszerű GEF editor
  - Kétféle csomópont, közöttük élek
  - 3400 sor kód (modell nélkül)
  - 16 osztály, 150 metódus…
  - Nem lehetne egyszerűbben?

Egyszerűsítés

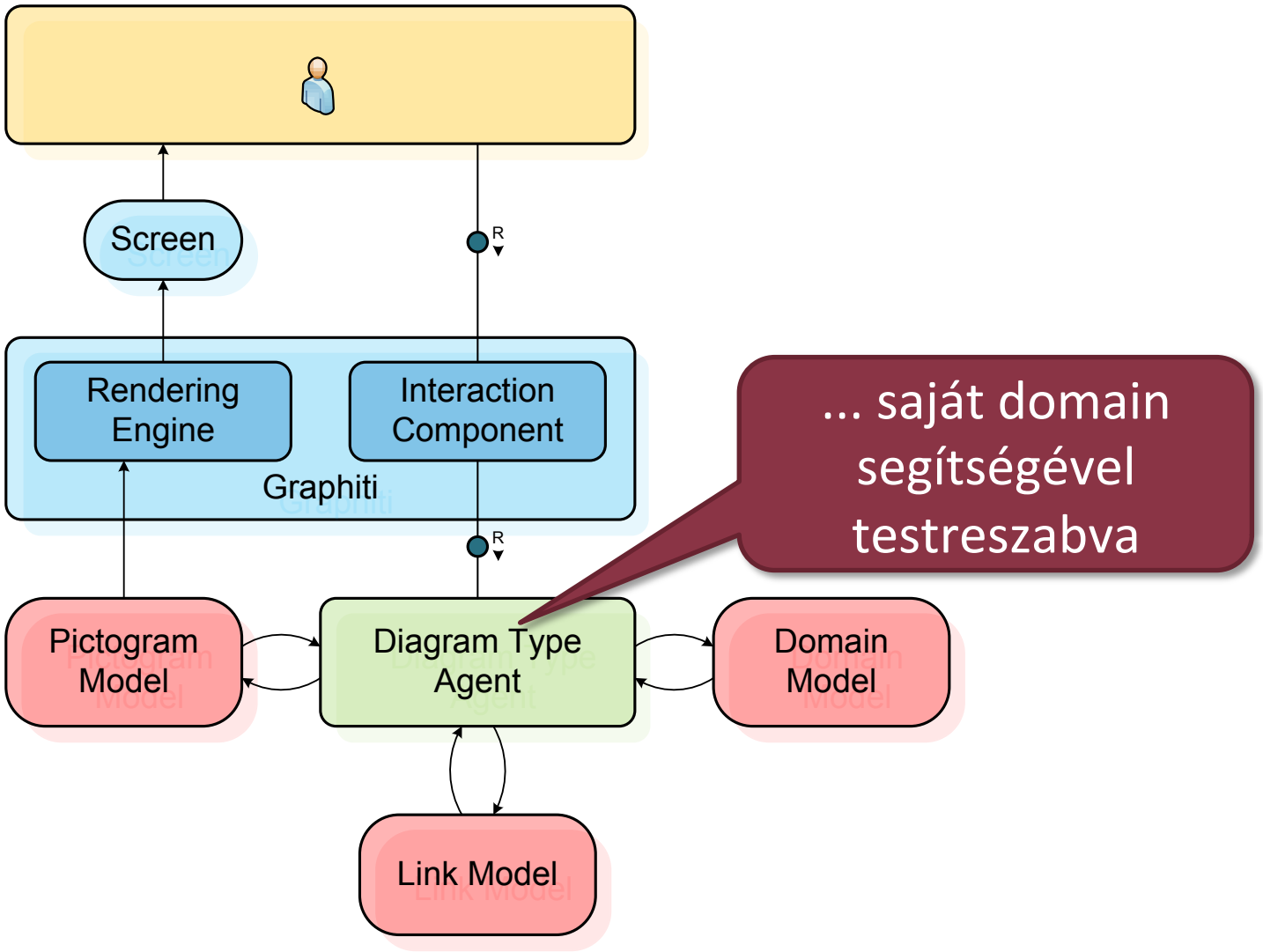
- EMF modellek
  - Egyformán kezelés
  - Mentés/megnyitás egyszerű
  - Fájlok közötti hivatkozások kezelhetőek
- Egyszerűbb interakció
  - Csak magas szintű funkciók implementációja

Graphiti architektúra



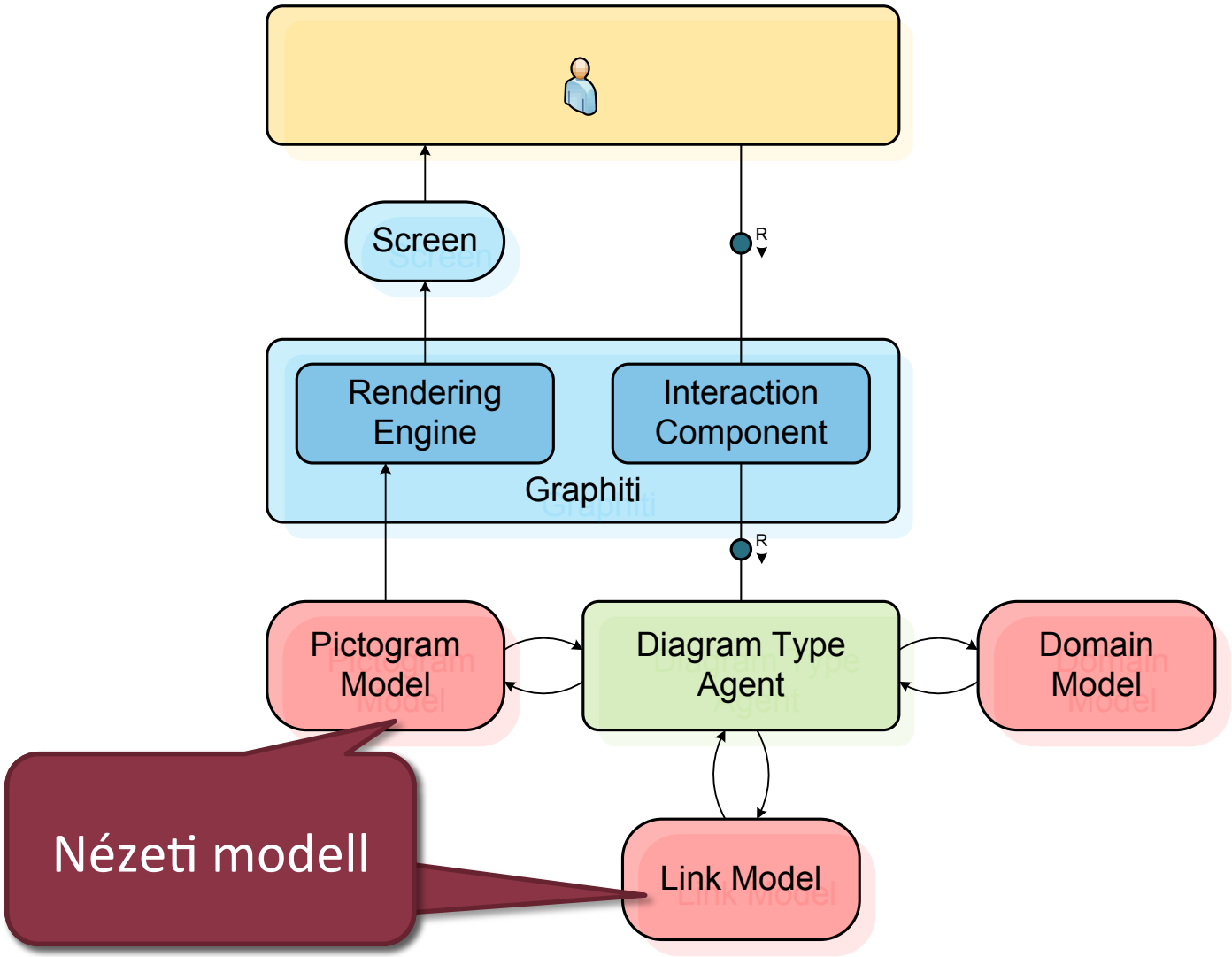
Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

Graphiti architektúra



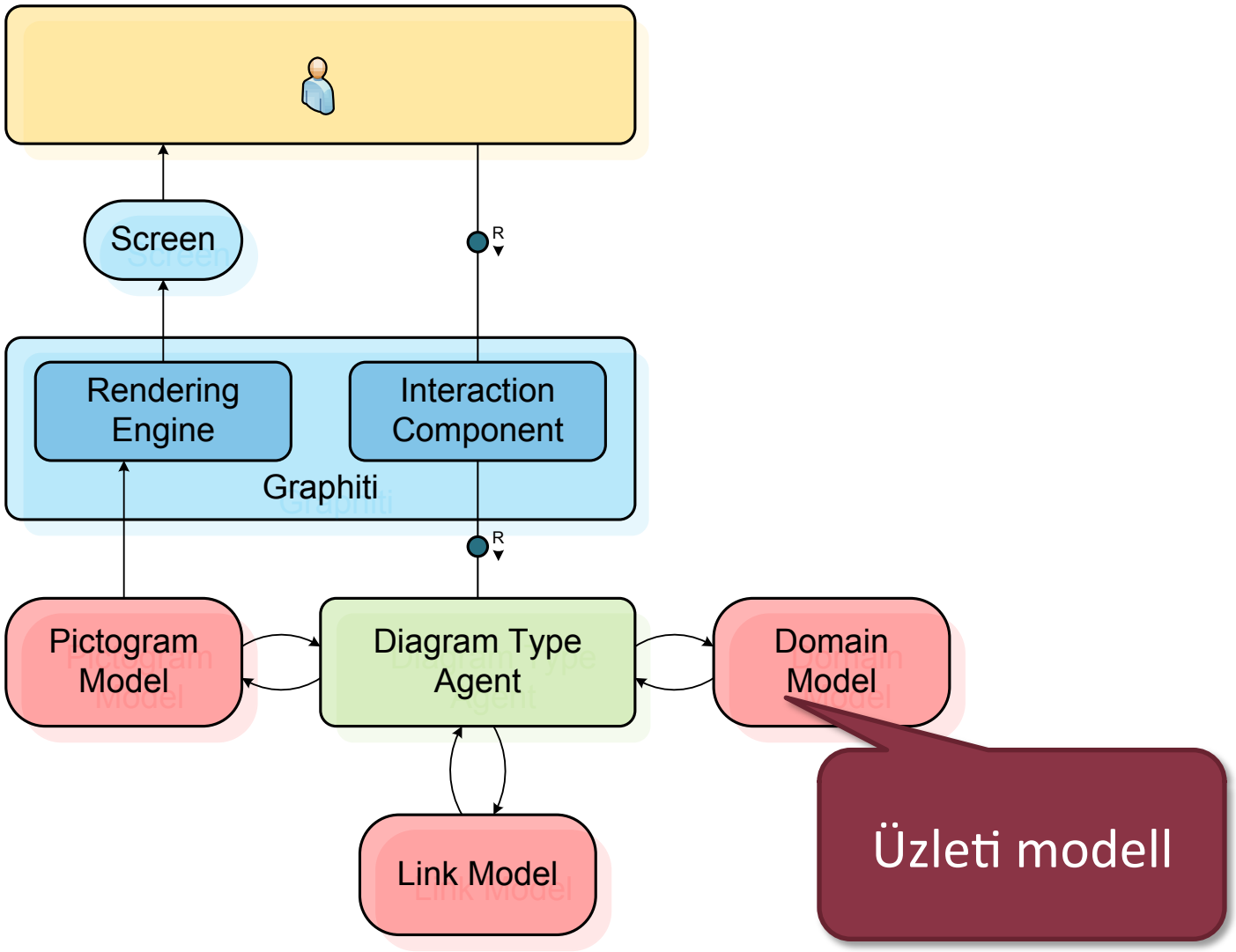
Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

Graphiti architektúra



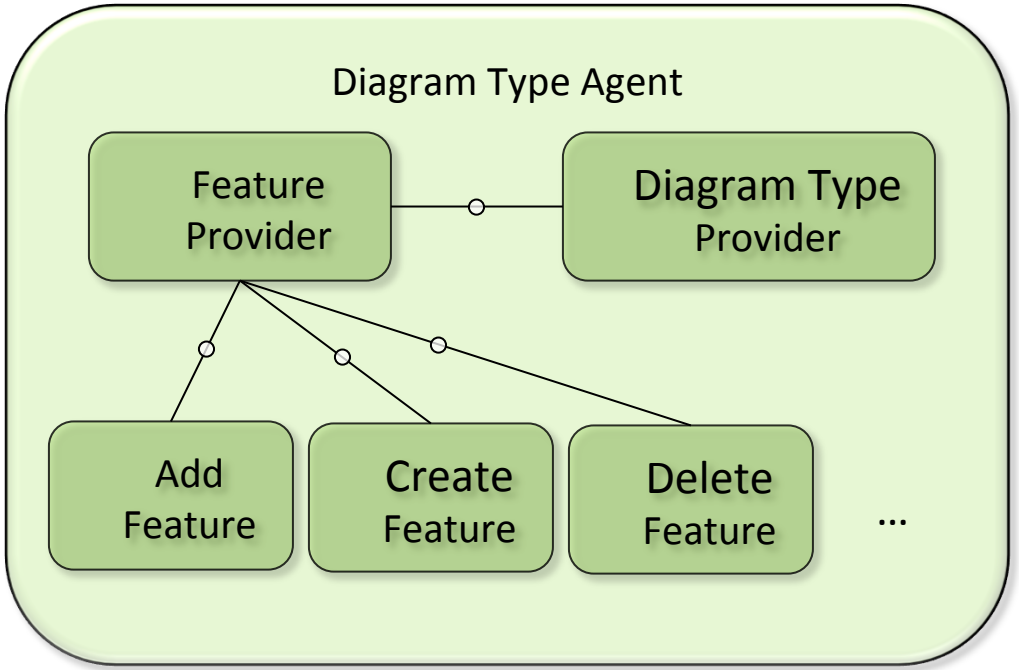
Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

Graphiti architektúra



Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

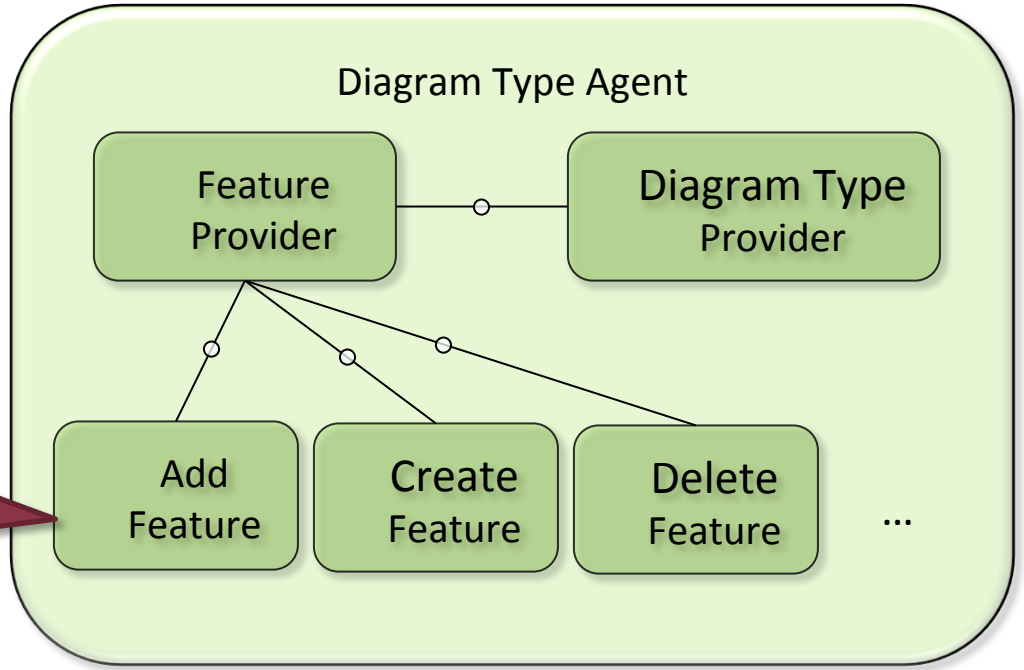
Graphiti architektúra 2. – Diagram Type Agent



Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

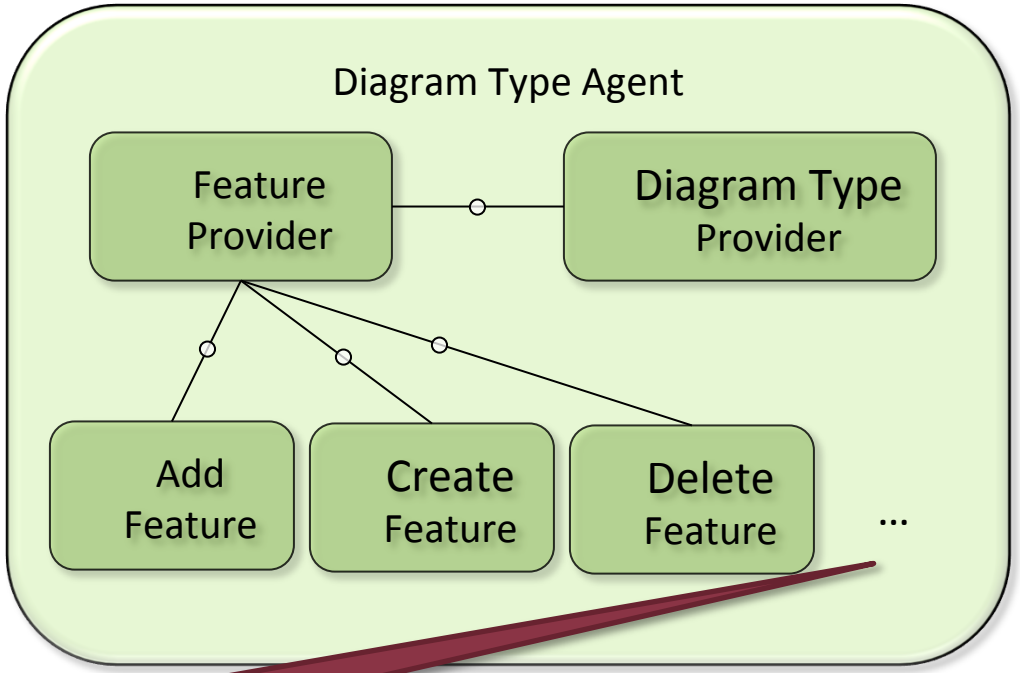
Graphiti architektúra 2. – Diagram Type Agent

Funkció (Feature): szerkesztési művelet az üzleti modellen



Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

Graphiti architektúra 2. – Diagram Type Agent



Leggyakoribb felhasználói szolgáltatások elérhetőek

Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

Milyen fogalmakkal dolgozunk?

Domain

Links

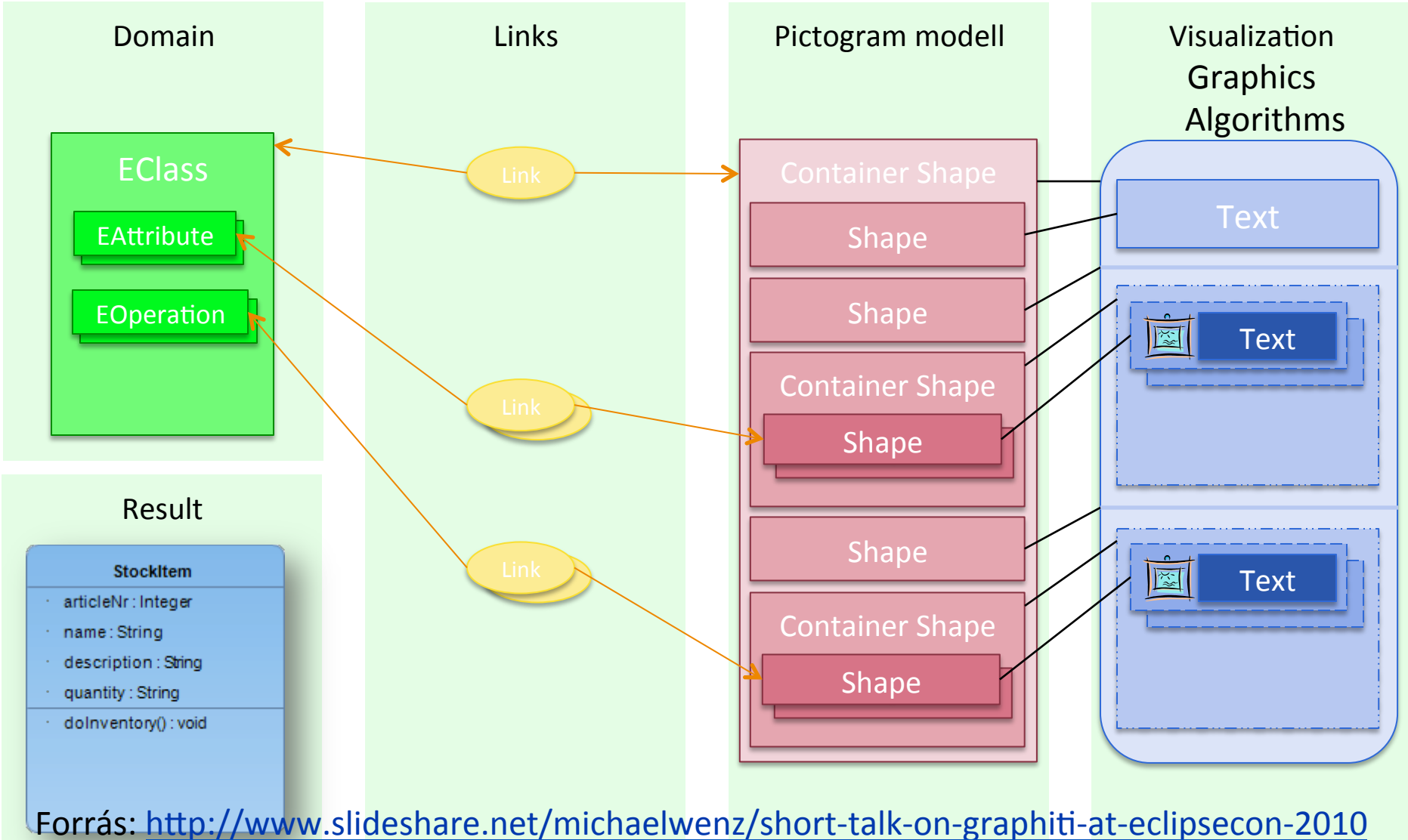
Pictogram modell

Visualization
Graphics
Algorithms

Result

Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

Milyen fogalmakkal dolgozunk?



Forrás: http://www.slideshare.net/michaelwenz/short-talk-on-graphiti-at-eclipsecon-2010

Pictogram és link model

- Pictogram metamodell
  - Megjelenő objektumok
  - EMF modell
  - Metamodell elérhető: http://eclipse.org/graphiti/images/pictograms.pdf
- Link
  - Kapcsolat Pictogram és saját modellek között
  - Generikus (nincs példánymodellhez kötve)

Tool építés lépései

1. Diagram Type Provider implementáció
2. Diagram Type Provider regisztráció
   - Kiterjesztési pont
3. Feature Provider implementáció
4. Feature implementáció

Tool építés lépései

1. **Diagram Type Provider implementáció**
2. Diagram Type Provider regisztráció
   - Kiterjesztési pont
3. Feature Provider implementáció
4. Feature implementáció

DiagramTypeProvider

```
public class SocialDiagramTypeProvider
      extends AbstractDiagramTypeProvider
      implements IDiagramTypeProvider {

      public SocialDiagramTypeProvider() {
      setFeatureProvider(new
SocialNetworkFeatureProvider(this));
      }
}
```

DiagramTypeProvider

```
public class SocialDiagramTypeProvider
      extends AbstractDiagramTypeProvider
      implements IDiagramTypeProvider {

      public SocialDiagramTypeProvider() {
      setFeatureProvider(new
SocialNetworkFeatureProvider(this));
      }
}
```

Feature Provider regisztráció

Tool építés lépései

1. Diagram Type Provider implementáció
2. **Diagram Type Provider regisztráció**
   - Kiterjesztési pont
3. Feature Provider implementáció
4. Feature implementáció

Feature Provider regisztráció

- Két kiterjesztési pont
  - Típusdefiníció
    - Egyszerű leírás
    - org.eclipse.graphiti.ui.diagramTypes
  - Implementáció
    - Csatolás provider és típusdefiníció között
    - org.eclipse.graphiti.ui.diagramTypeProviders

Tool építés lépései

1. Diagram Type Provider implementáció
2. Diagram Type Provider regisztráció
   - Kiterjesztési pont
3. **Feature Provider implementáció**
4. Feature implementáció

Feature Provider

- Használjuk az *AbstractFeatureProvider* ősosztályt
- Megfelelő metódusba funciók regisztrálása
  - Ha nincs, akkor null visszatérési érték
- Többféle funció
  - Create: modellobjektum létrehozása
  - Add: hozzáadás a diagramhoz
  - Copy, Paste
  - Update
  - DirectEditing

Tool építés lépései

1. Diagram Type Provider implementáció
2. Diagram Type Provider regisztráció
   - Kiterjesztési pont
3. Feature Provider implementáció
4. **Funció implementáció**

Funkció implementáció

- Absztrakt megvalósítások
  - Pl. AbstractAddShapeFeature
- Típusfüggő implementáció
  - Pl. canAdd és add metódusok

Összegzés

- Magasabb szintű könyvtár GEF felett
  - EMF modellek
  - Univerzális szerkesztő
- Kevés kódolás
- Egységes kinézet
- De hiányzó funciók