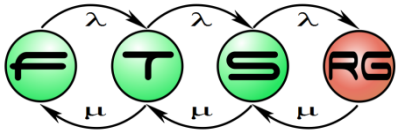
Grafikus szerkesztők fejlesztése

A Graphical Editing Framework

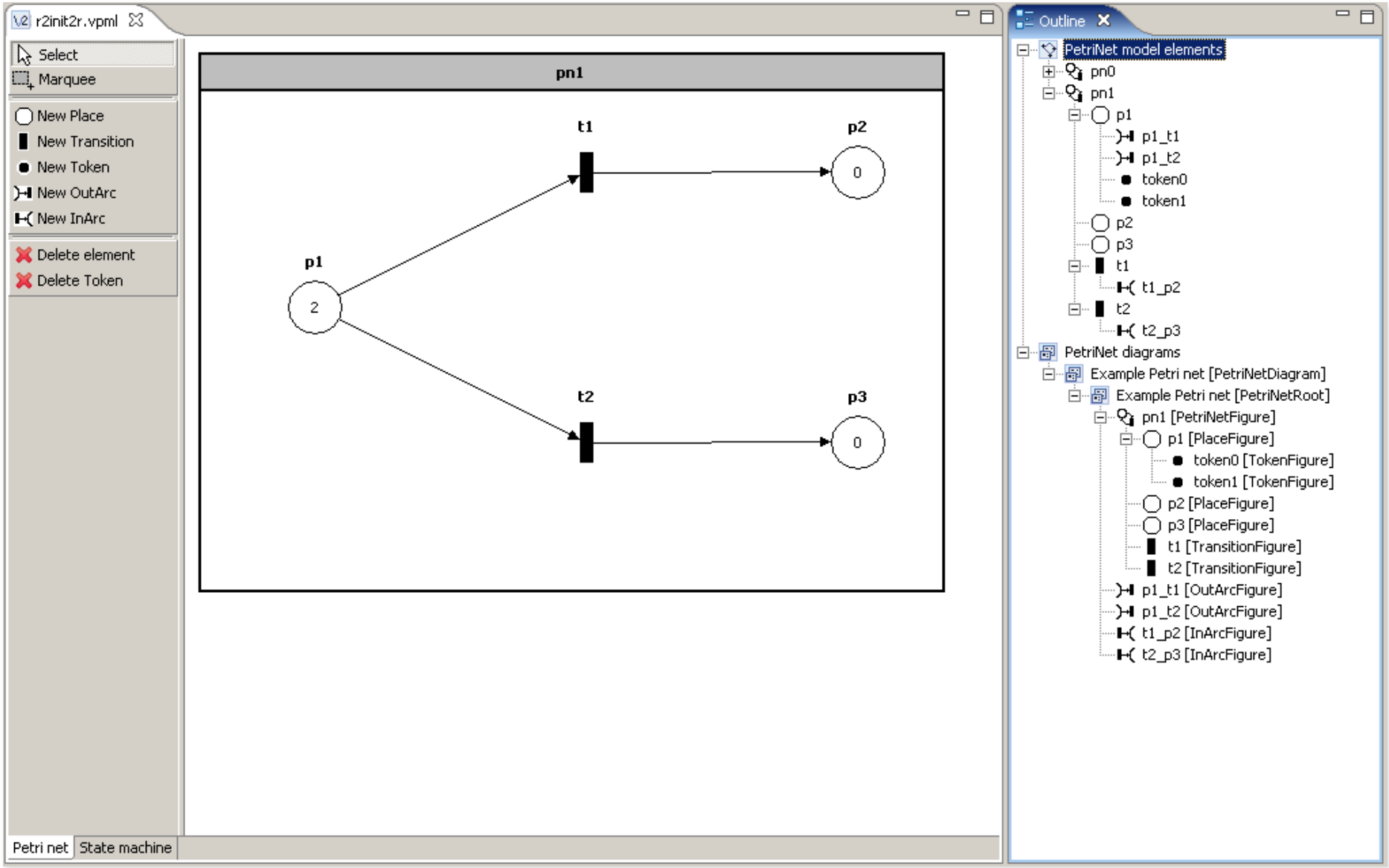


Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

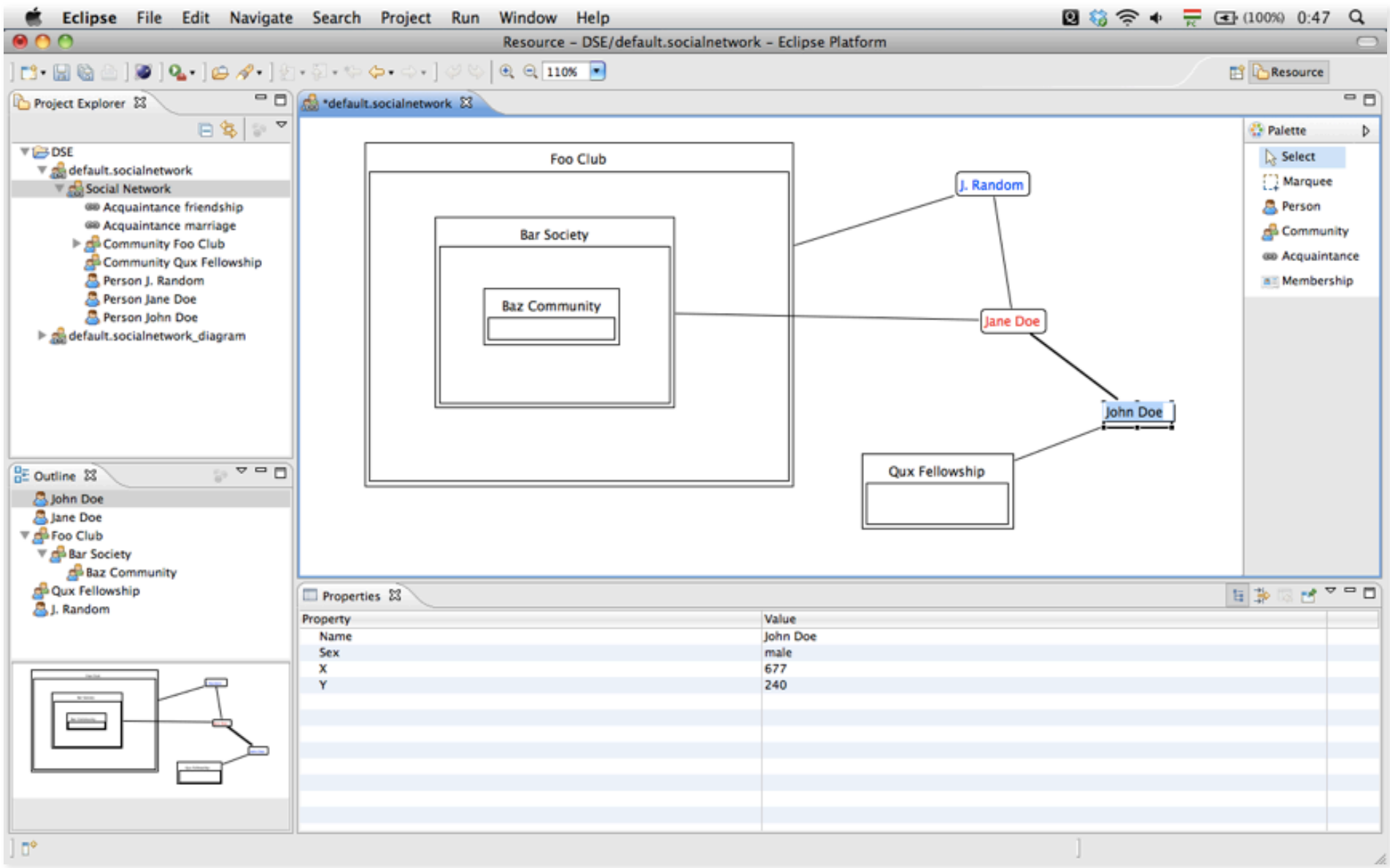
A GEF célja

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

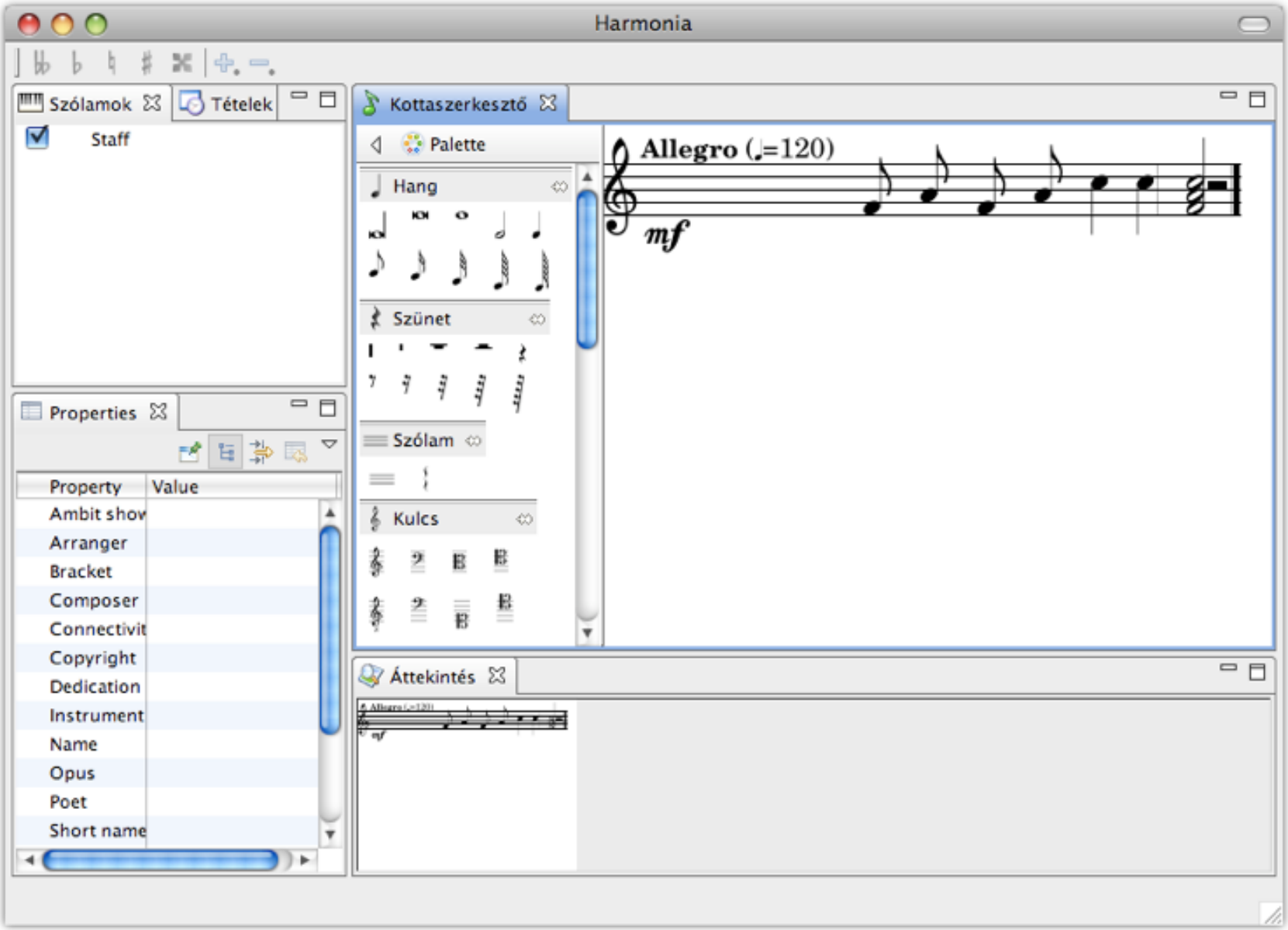
Példa


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]

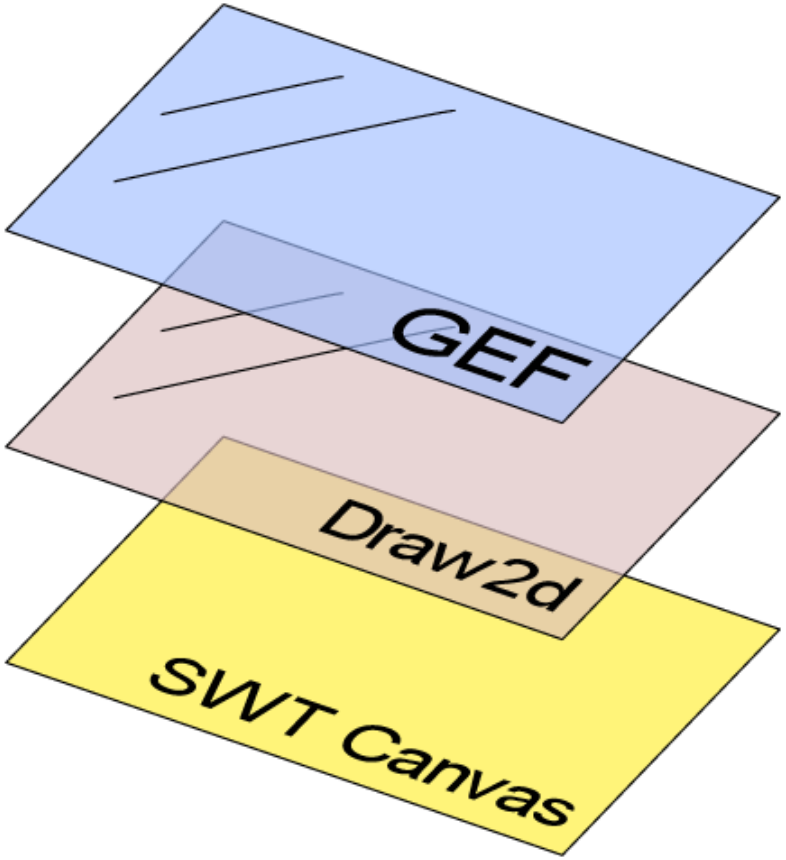
Példa


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


Harmonia
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

A GEF felépítése



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

- Megjelenítés
- Elemek elrendezése
- Nagyítás

- Natív (SWT) réteg

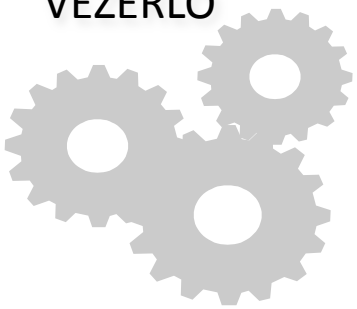
MVC felépítés

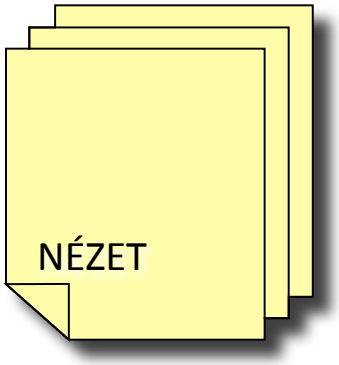
- Model-View-Controller (Modell – Nézet – Vezérlő)
- Adatok tárolása és megjelenítése egymástól elválasztva
- Modell: adatok tárolása
- Nézet: grafikus megjelenítés
- Vezérlő: felhasználói interakció
- Elterjedt: Swing, JFace, MFC, JSF

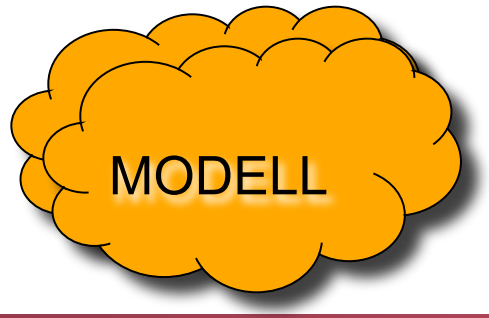
MVC a gyakorlatban

Felhasználó

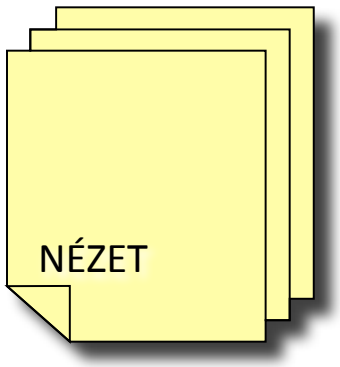
VEZÉRLŐ

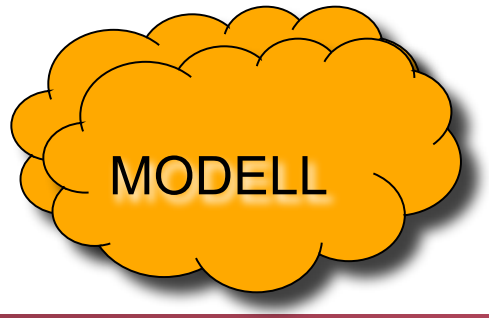




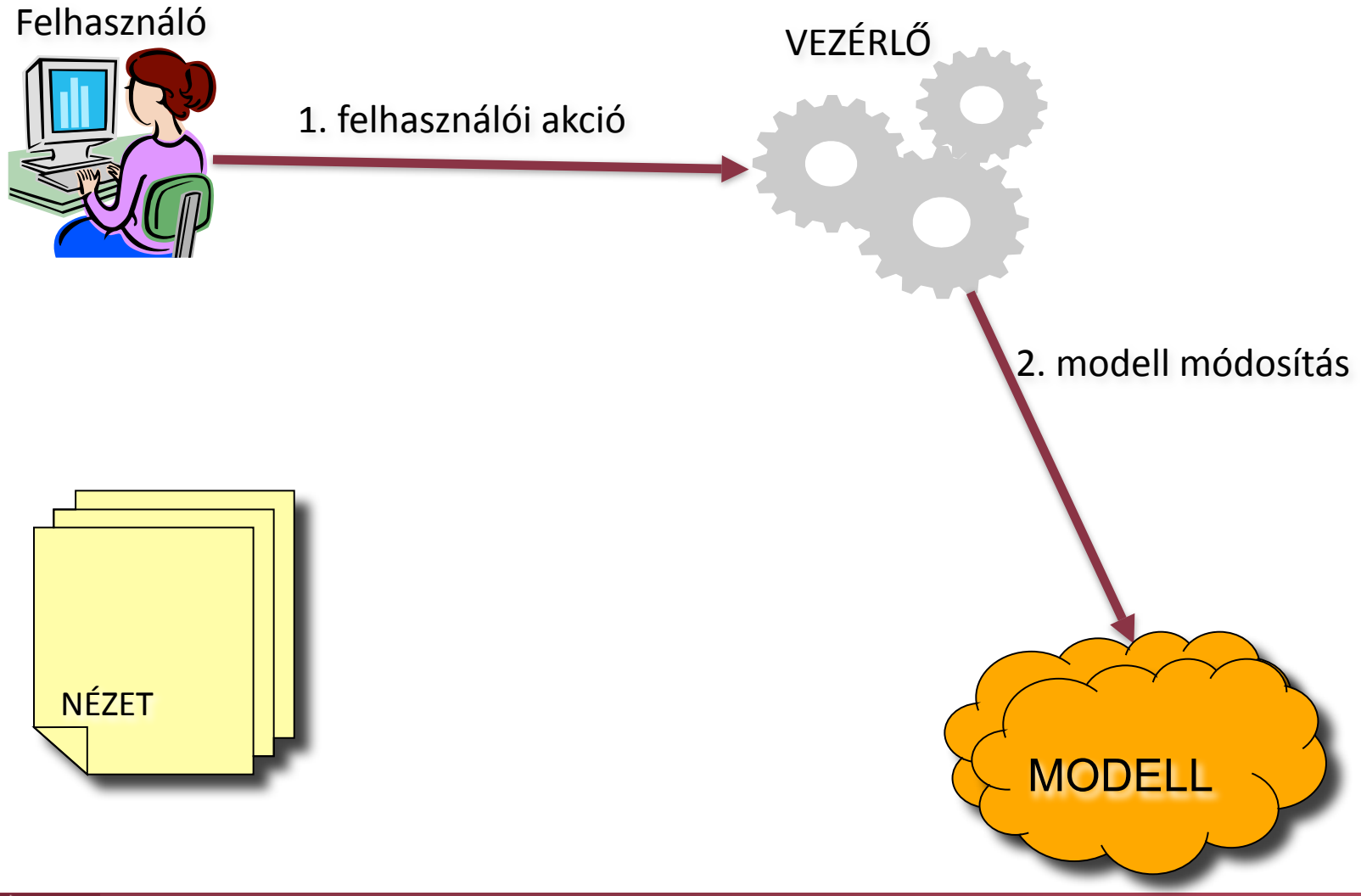


MVC a gyakorlatban

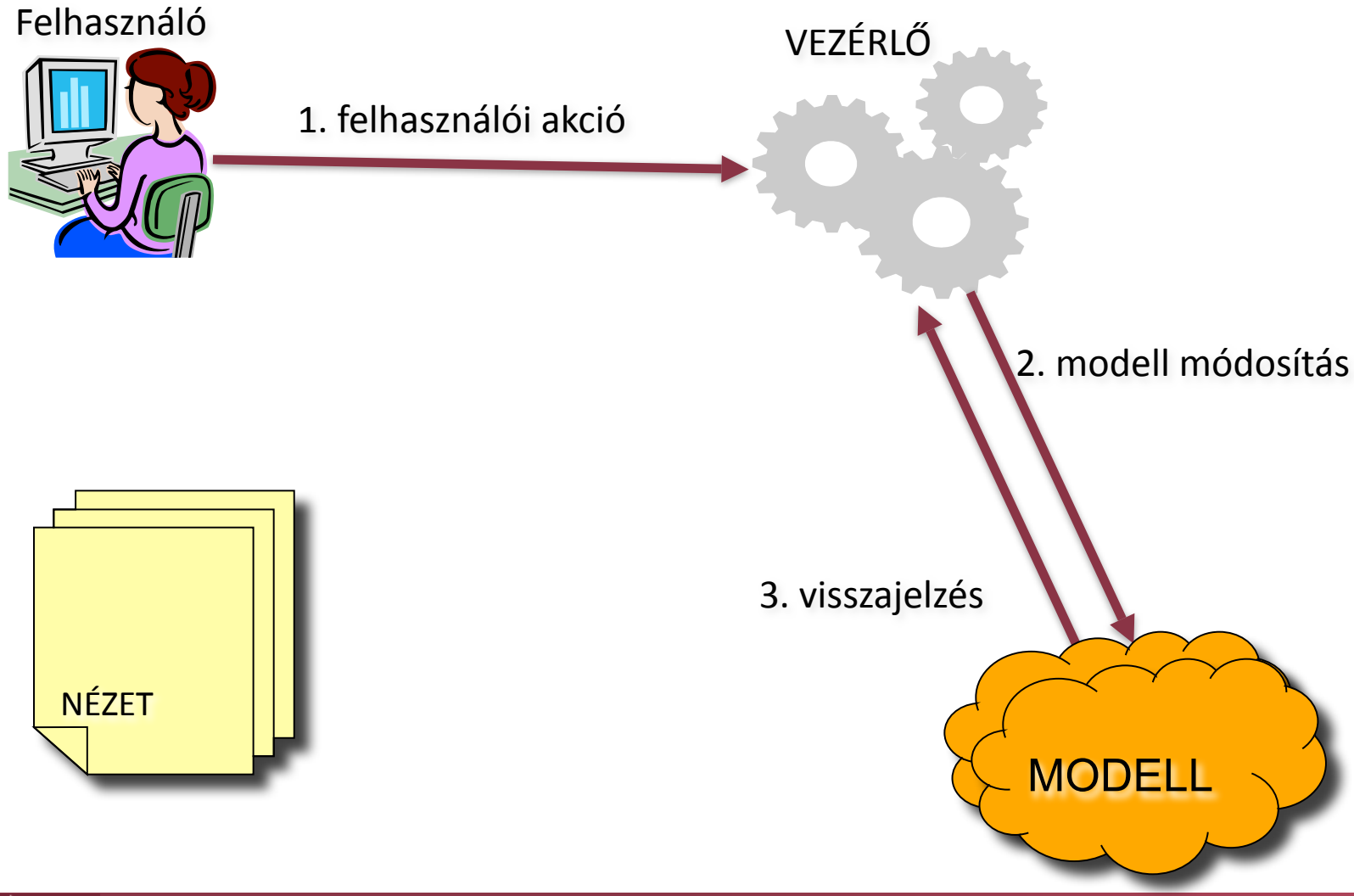




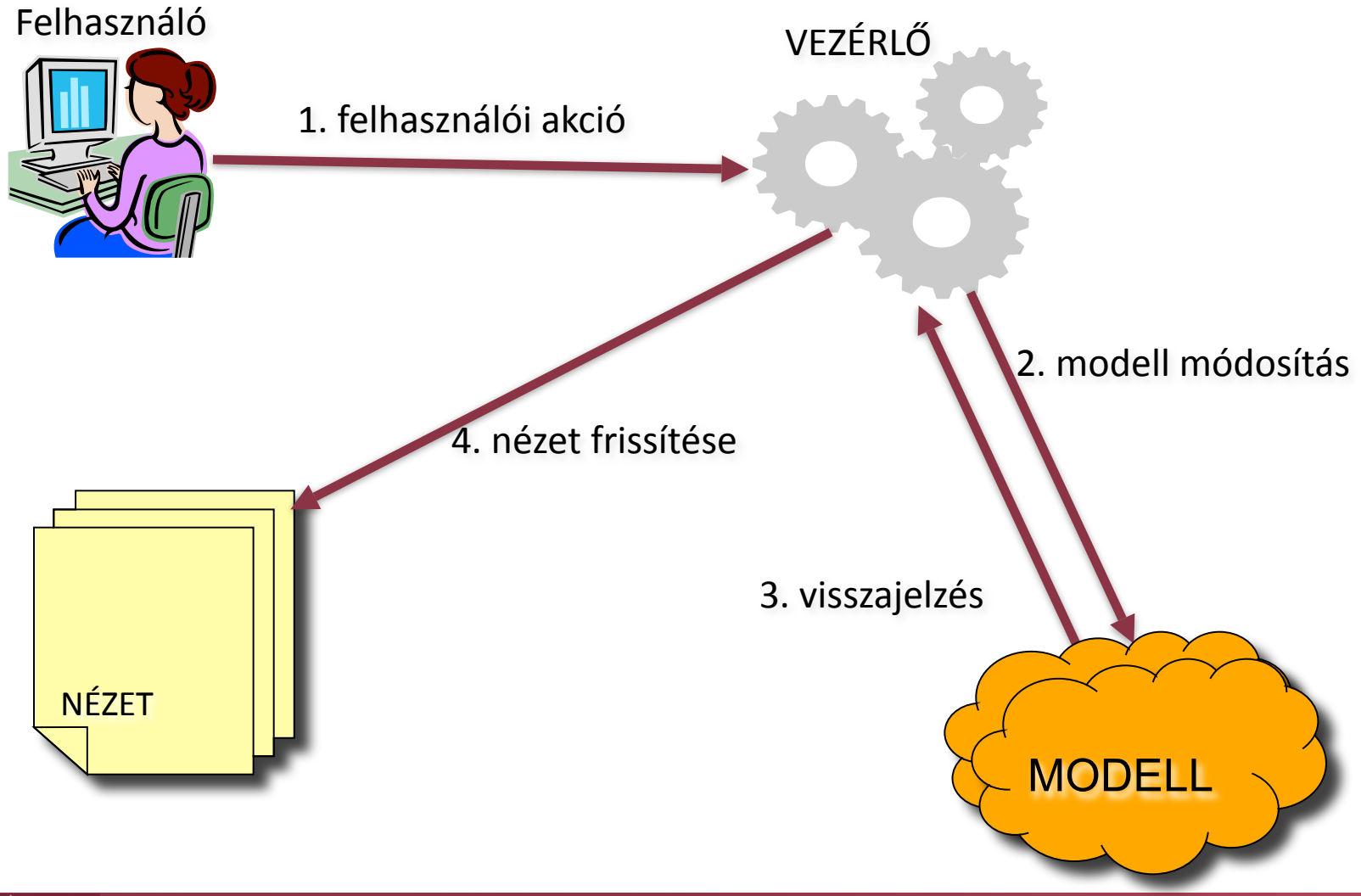
MVC a gyakorlatban



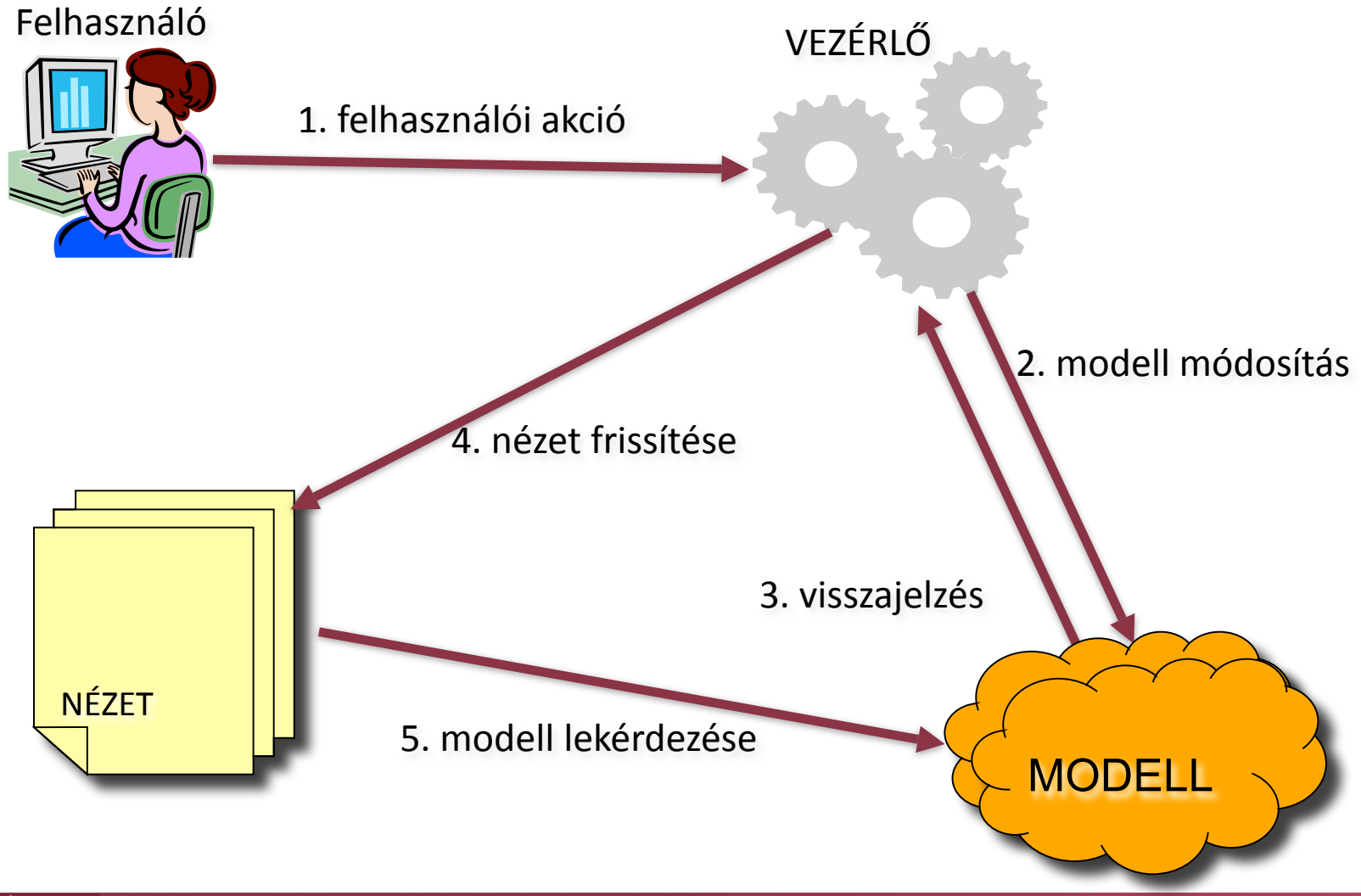
MVC a gyakorlatban



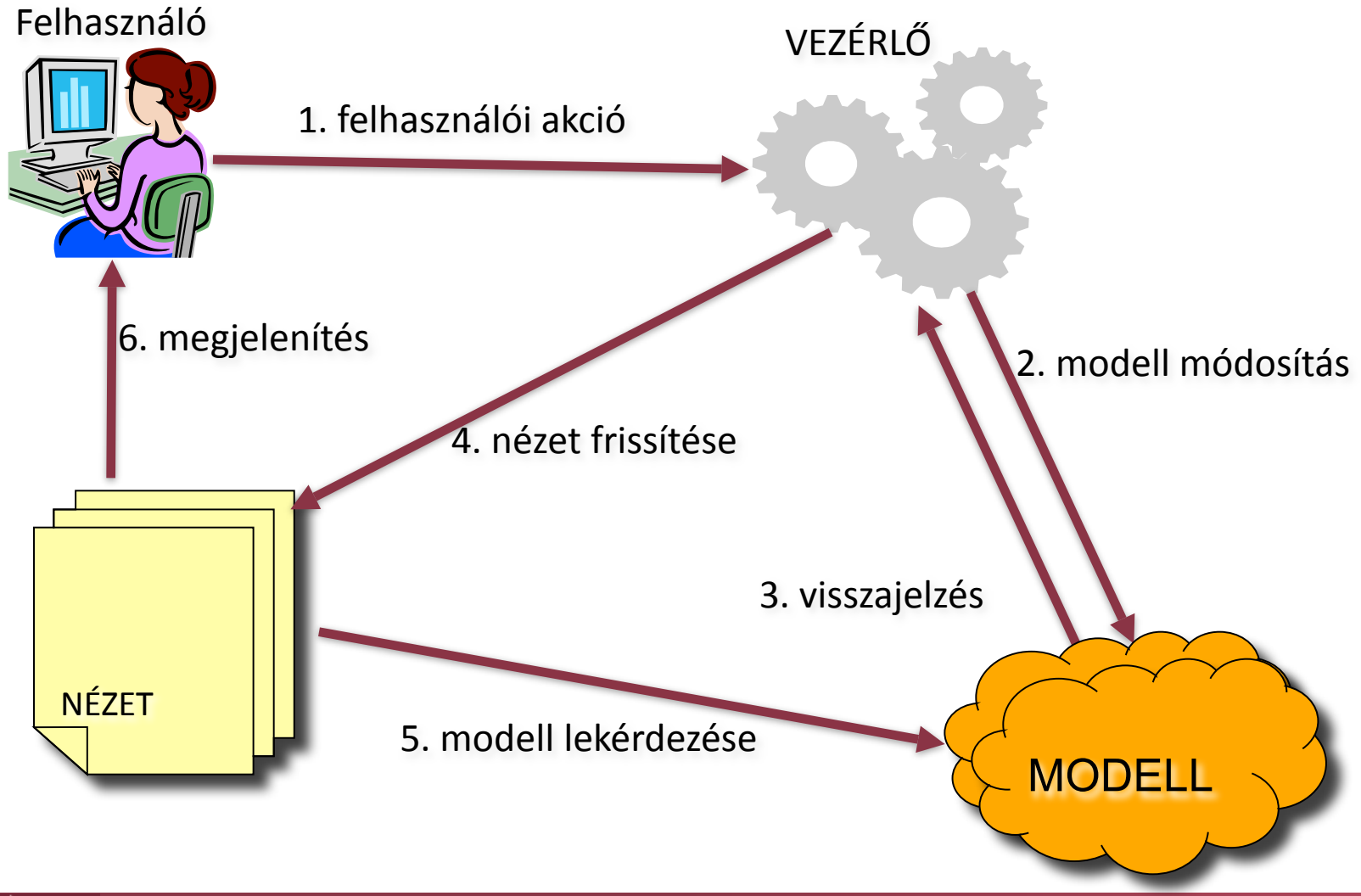
MVC a gyakorlatban



MVC a gyakorlatban



MVC a gyakorlatban



MVC a GEF-ben

Model

MVC a GEF-ben

Model

Értesítés

MVC a GEF-ben: Model

- Modell: tetszőleges
  - Pl. Java osztályok, EMF, adatbázis
  - Célszerű egy gyökérelem
  - Támogatnia kell az értesítéseket
    - Jelentés a vezérlőnek, ha módosítás történt
    - 1 modell <-> több nézet esetén fontos
    - Pl. EMF Notification Framework
  - Célszerű sorosíthatónak lennie
  - Üzleti modell: struktúra, adatok
  - Nézeti modell: megjelenítési információk
    - Pl. pozíció, méret

Egyszerű modell

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

Egyszerű modell

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

Belső modell

Egyszerű modell

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

Nézeti modell

Egyszerű értesítés

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


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

Egyszerű értesítés

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


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

Figyelők kezelése

Egyszerű értesítés

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


public class Notifiable {
   private final List<IModelListener> listeners =
      new ArrayList<IModelListener>();
   public void addListener(IModelL                    ) {
      listeners.add(listener);
    }
   public void removeListener(IMod                    ner) {
      listeners.remove(listener);
    }
   protected void fireListeners() {
      for (IModelListener 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 milyen értékre változott
  - El kell küldeni magát a változást is, lassú
  - Könnyen feldolgozható

GEF workflow

Model

View

GEF workflow

Model

View

Kirajzolás

GEF workflow

Model

View

Kirajzolás

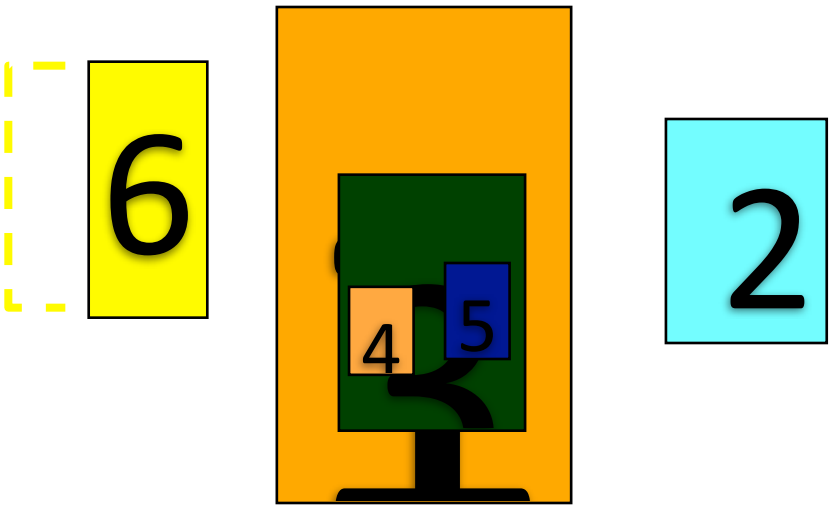
Elrendezés

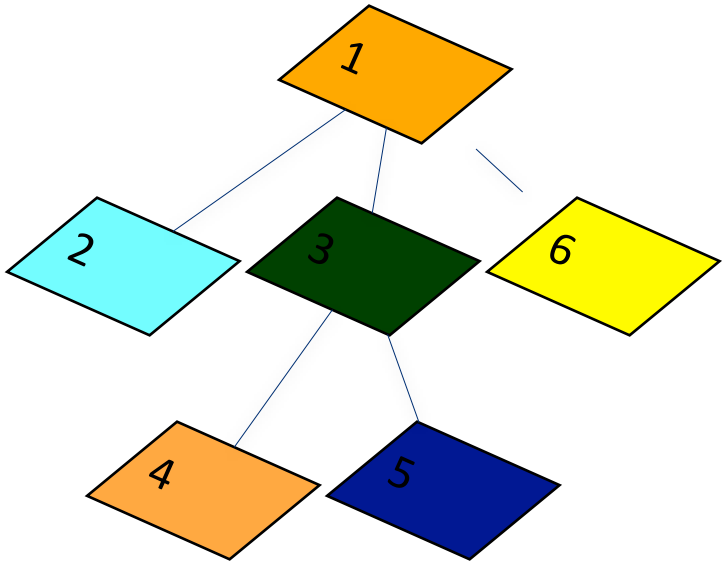
MVC a GEF-ben: View

- Nézet: Draw2D osztályok
  - SWT-re épülő grafikus könyvtár
  - Egyszerű elemek (címke, téglalap, nyíl)
  - Hierarchikus megjelenítés
  - Alap építőelem: Figure
  - GEF nézet = Draw2D Figure példány
  - Bármely SWT alkalmazásban használható
    - Saját üzenetkezelése is van

Draw2D hierarchia

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

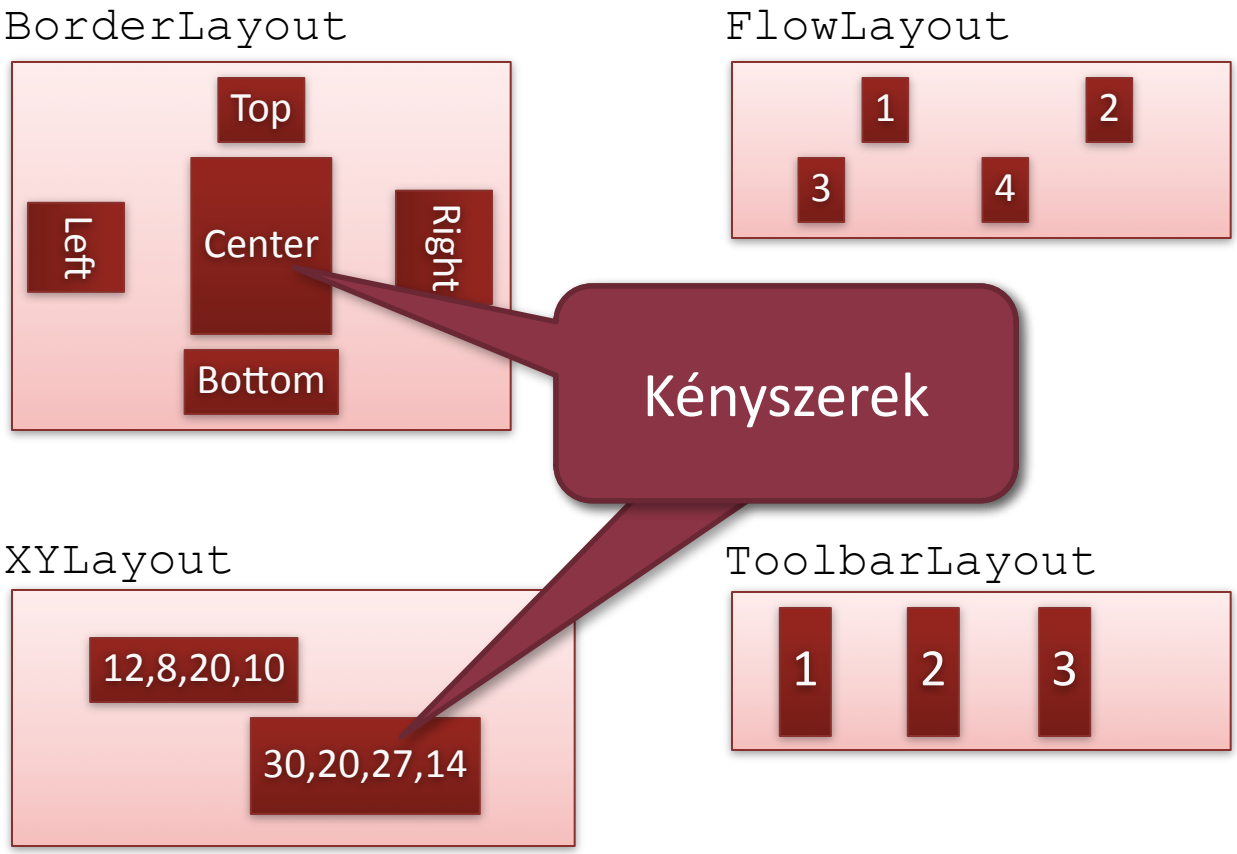




Draw2D LayoutManager

- Gyerekek elrendezése szülőn belül
- Több beépített, lehet saját is
- Constraint: Egy gyerek elhelyezésével kapcsolatos kényszer
  - SWT-nél ez volt a LayoutData
  - 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

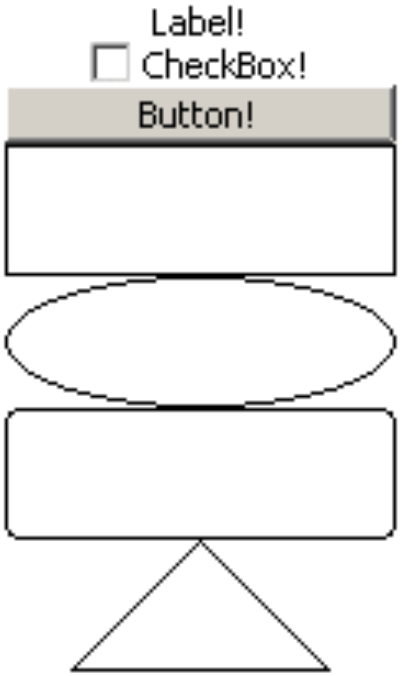
- Egyszerű elemek
  - Label, Button, CheckBox, Image
- Geometriai alakzatok
  - RectangleFigure, Ellipse, Triangle
- Panel: általános konténer elem
- ScrollPane: görgethető tartalmú elem

Draw2D alapelemek

```
public class Example1 extends Figure {
 public Example1() {
   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);
   }
 }
}
```

Draw2D alapelemek

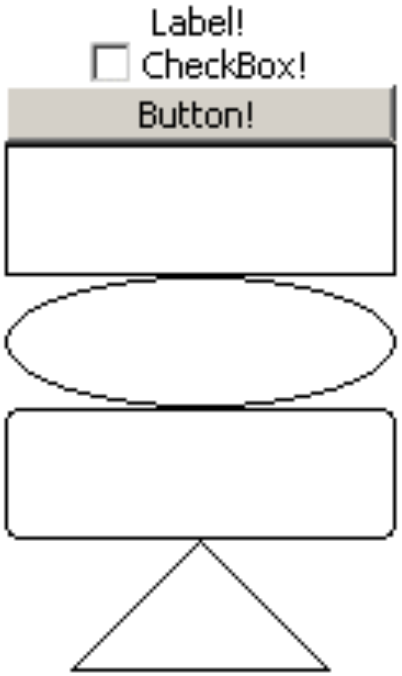
```
public class Example1 extends Figure {
 public Example1() {
   setOpaque(true);
   setBackgroundColor(ColorConstants.white);
   setLayoutManager(new ToolbarLayout());
   add(new Label("Label!"));
   add(new CheckBox("CheckBox!"));
   add(new Button("Button!"));
   add(new RectangleFigure());
   add(new Ellipse());
   add(new RoundedRectangle());
   add(new Triangle());
   for (int i = 3; i <= 6; i++)
    ((Figure) getChildren().get
      (i)).setPreferredSize(-1, 40);
   }
 }
}
```



Draw2D alapelemek

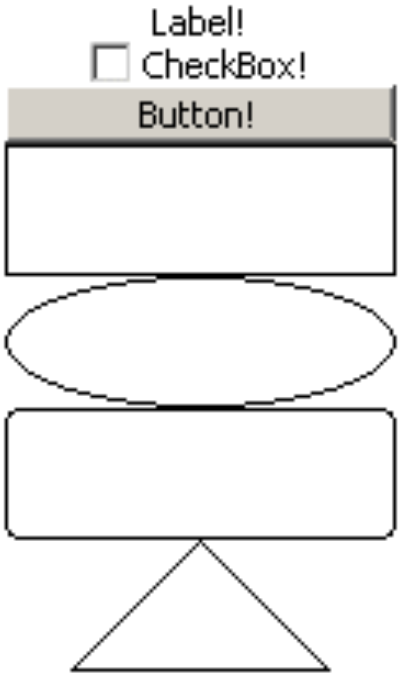
Alapértelmezetten minden átlátszó

```
public class Example1 e
 public Example1() {
   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);
   }
 }
}
```



Draw2D alapelemek

```
public class Example1 extends Figure {
 public Example1() {
   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 figyelembe veheti

Draw2D keretek

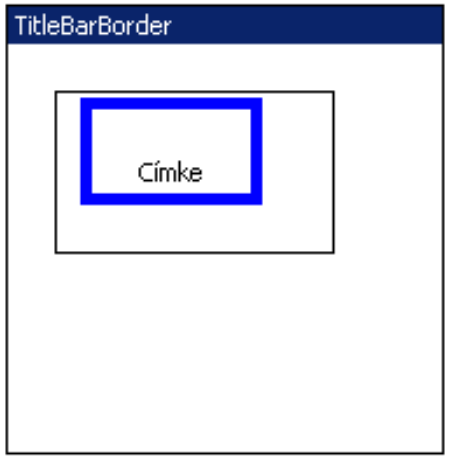
- Minden Figure-höz rendelhető keret
- TitleBarBorder: dialógusablak jelleg
- LineBorder: egyszerű vonal
- MarginBorder: üres hely
- Keretek egymásba ágyazhatók
  - CompoundBorder
- Megosztható Figure-ök között

Draw2D keretek

```
public class Example2 extends Figure {
   public Example2() {
     setOpaque(true);
     setBackgroundColor(ColorConstants.white);
     setLayoutManager(new XYLayout());
     TitleBarBorder border = new TitleBarBorder("TitleBarBorder");
     border.setTextColor(ColorConstants.white);
     setBorder(new CompoundBorder(new LineBorder(1), tb));
     Label label = new Label("Címke");
     label.setBorder(new CompoundBorder(
       new LineBorder(1), new CompoundBorder(
       new MarginBorder(2, 10, 20, 30),
       new LineBorder(ColorConstants.blue, 5))));
     add(label);
     setConstraint(label,
       new Rectangle(20, 20, 120, 70));
   }
}
```

Draw2D keretek

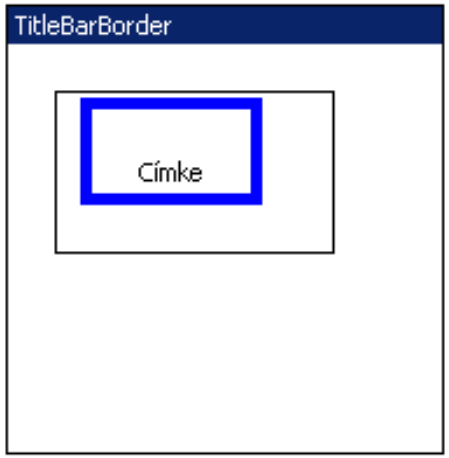
```
public class Example2 extends Figure {
   public Example2() {
     setOpaque(true);
     setBackgroundColor(ColorConstants.white);
     setLayoutManager(new XYLayout());
     TitleBarBorder border = new TitleBarBorder("TitleBarBorder");
     border.setTextColor(ColorConstants.white);
     setBorder(new CompoundBorder(new LineBorder(1), tb));
     Label label = new Label("Címke");
     label.setBorder(new CompoundBorder(
       new LineBorder(1), new CompoundBorder(
       new MarginBorder(2, 10, 20, 30),
       new LineBorder(ColorConstants.blue, 5))));
     add(label);
     setConstraint(label,
       new Rectangle(20, 20, 120, 70));
   }
}
```



Draw2D keretek

```
public class Example2 extends Figure {
   public Example2() {
     setOpaque(true);
     setBackgroundColor(ColorConstants.white);
     setLayoutManager(new XYLayout());
     TitleBarBorder border = new TitleBarBorder("TitleBarBorder");
     border.setTextColor(ColorConstants.white);
     setBorder(new CompoundBorder(new LineBorder(1), tb));
     Label label = new Label("Címke");
     label.setBorder(new CompoundBorder(
       new LineBorder(1), new CompoundBorder(
       new MarginBorder(2, 10, 20, 30),
       new LineBorder(ColorConstants.blue, 5))));
     add(label);
     setConstraint(label,
       new Rectangle(20, 20, 120, 70));
   }
}
```

Kényszereket a szülőnél adjuk meg



Draw2D egyéb elemek

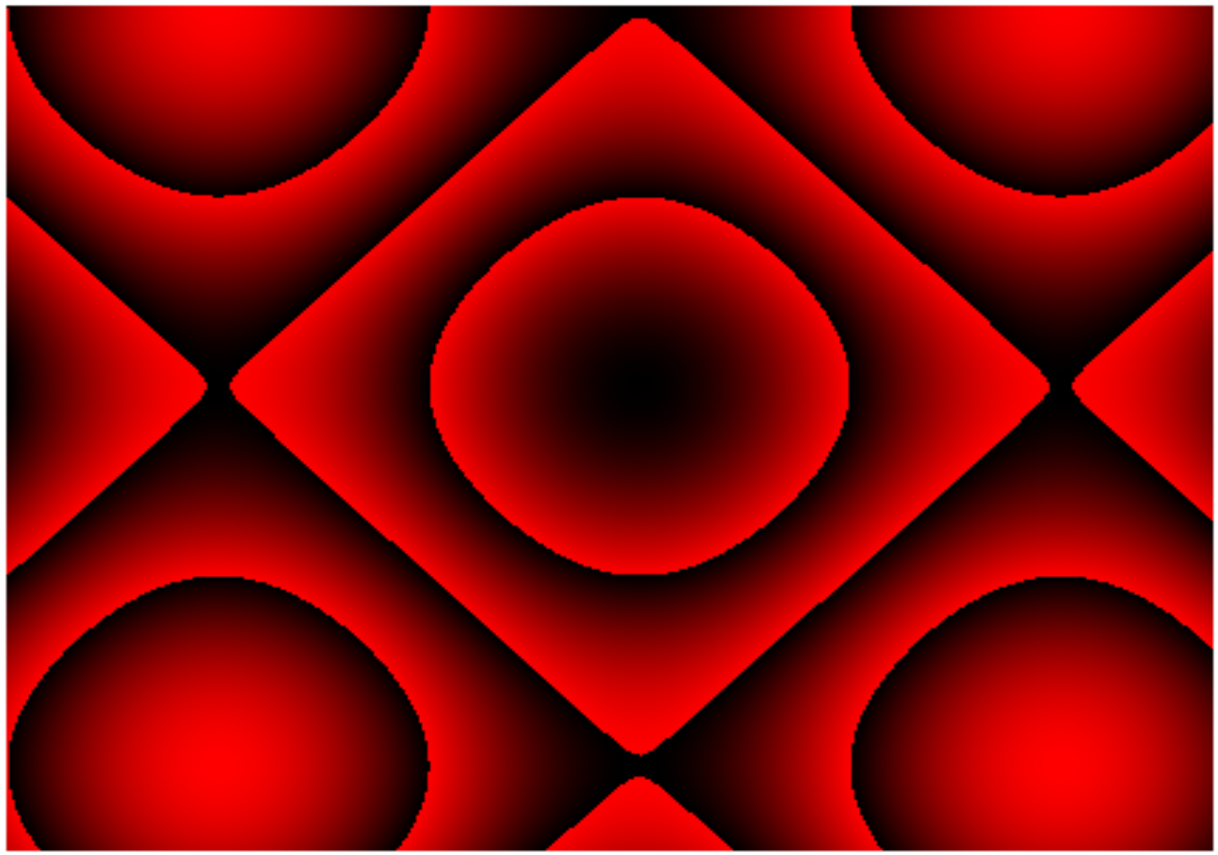
- Nyilak: lásd később
- Saját elemek
  - Beépített elemek kombinációja nem mindig elégséges
  - paintFigure() felülírása kell
  - Tetszőleges SWT rajzoló kód lehet

Draw2D saját elem

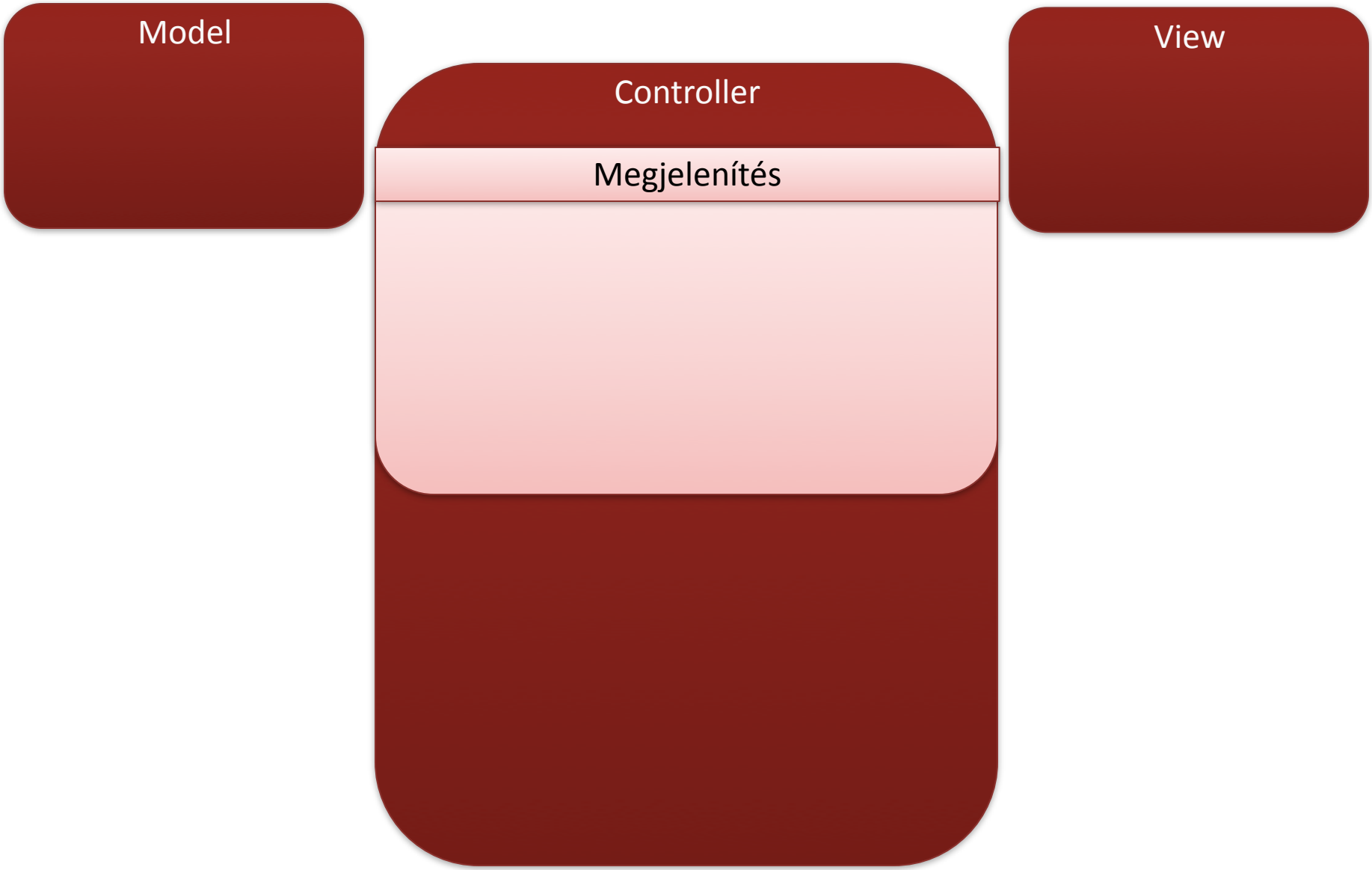
```
public class Example3 extends Figure {
   @Override
   protected void paintFigure(Graphics graphics) {
        Rectangle r = getBounds();
        PaletteData pd = new PaletteData(0xff0000, 0xff00, 0xff);
        pd.redShift = -16; pd.greenShift = -8; pd.blueShift = 0;
        pd.isDirect = true;
        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

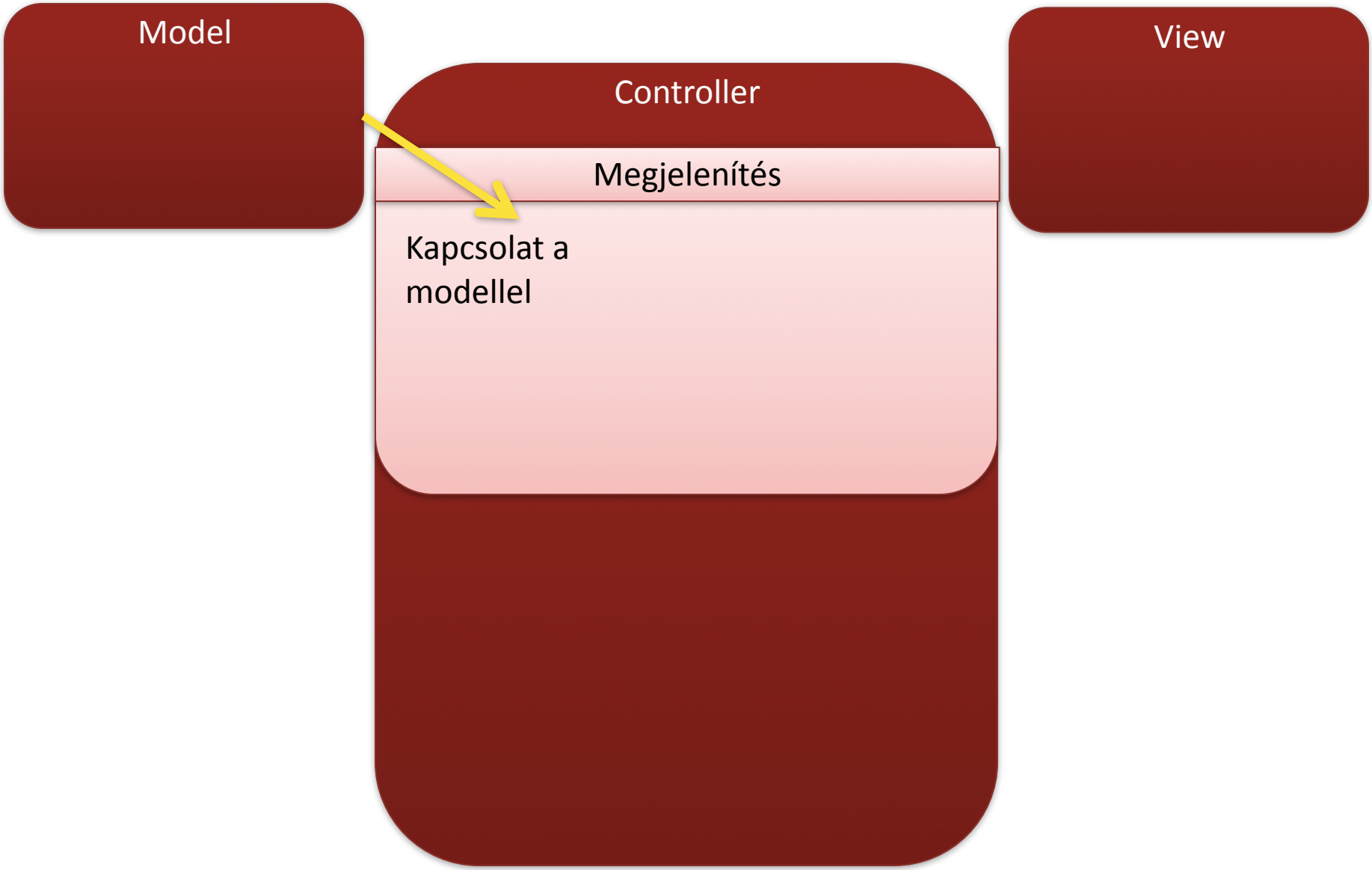
```
public class Example3 extends Figure {
    @Overri
    protect
        Re
        Pa
        pd
        pd
        Im
        fo
                                                    0xff);
                                                t = 0;

                                                , pd);

                                                h) +
                                                  % 256;
        }
        Im
        gr
    }
}
```



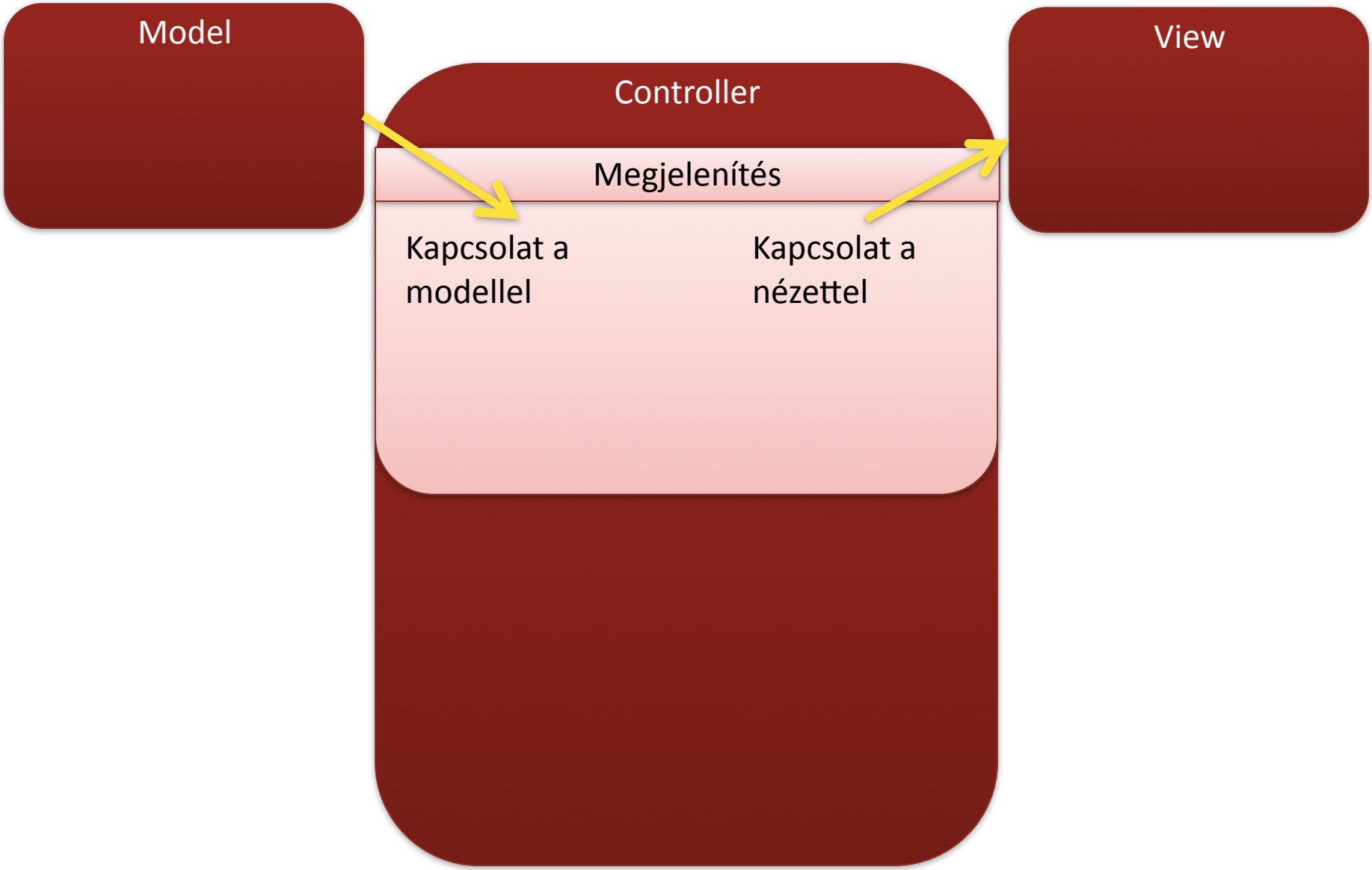
GEF workflow



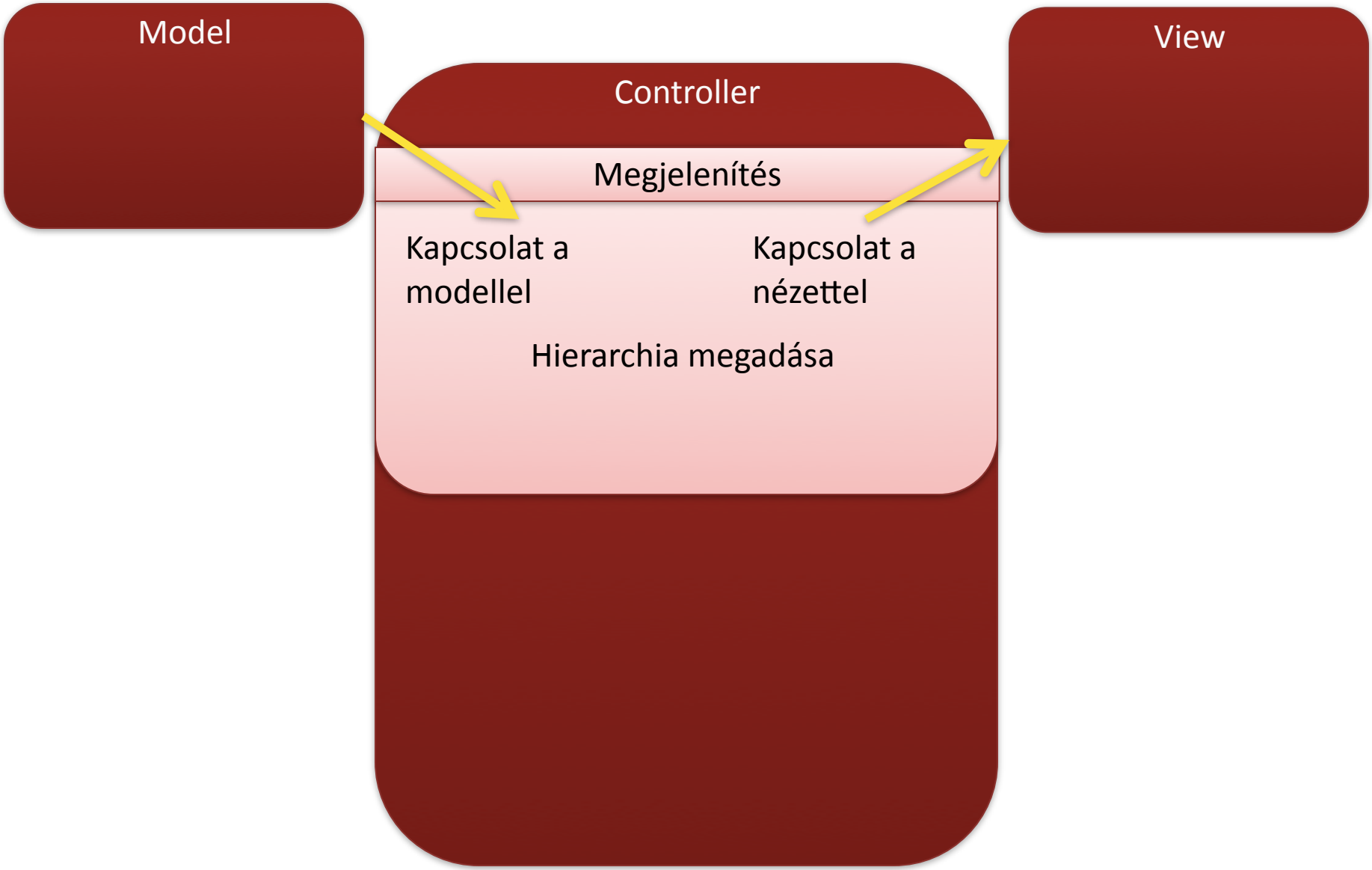
GEF workflow



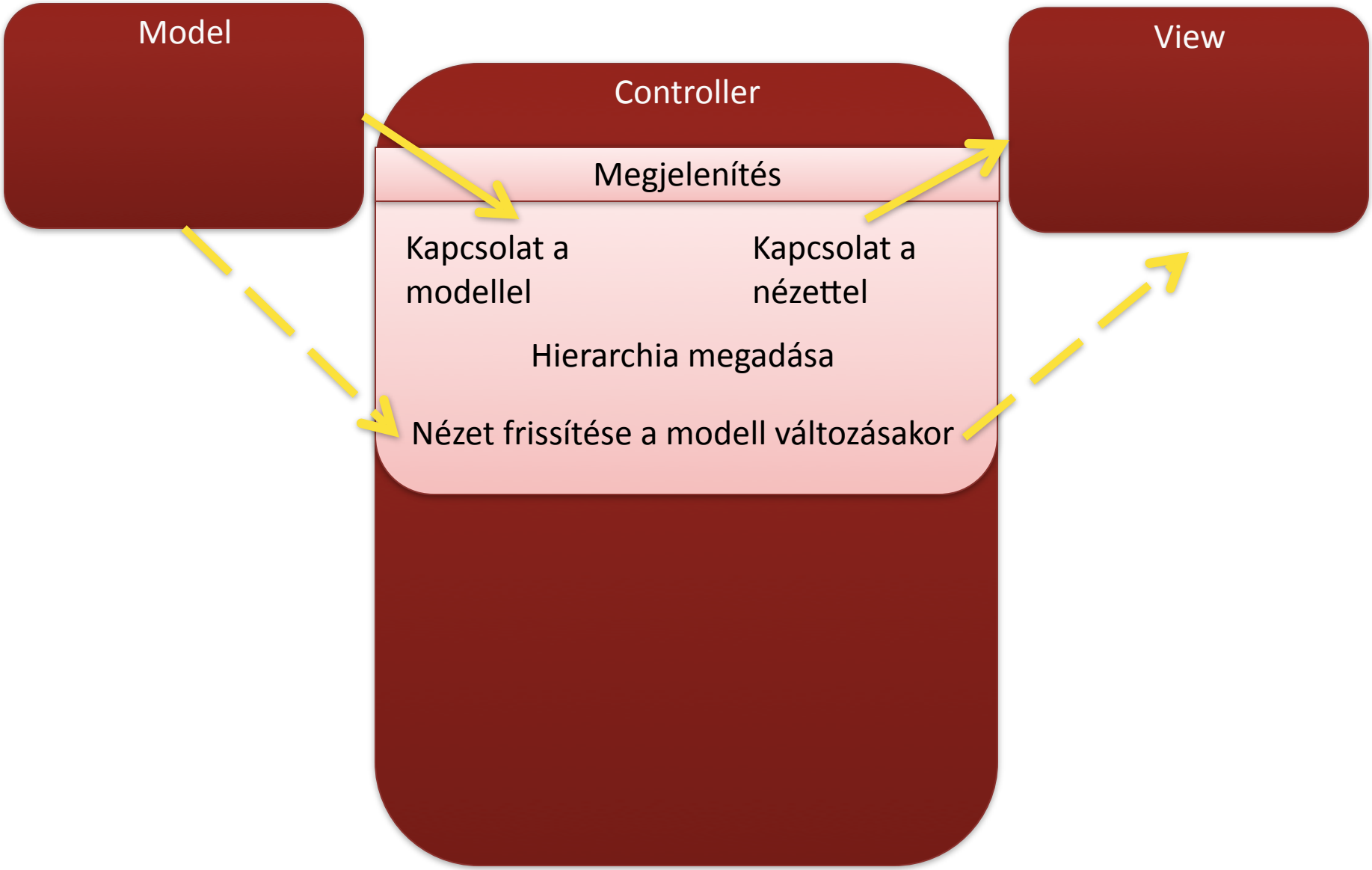
GEF workflow



GEF workflow



GEF workflow

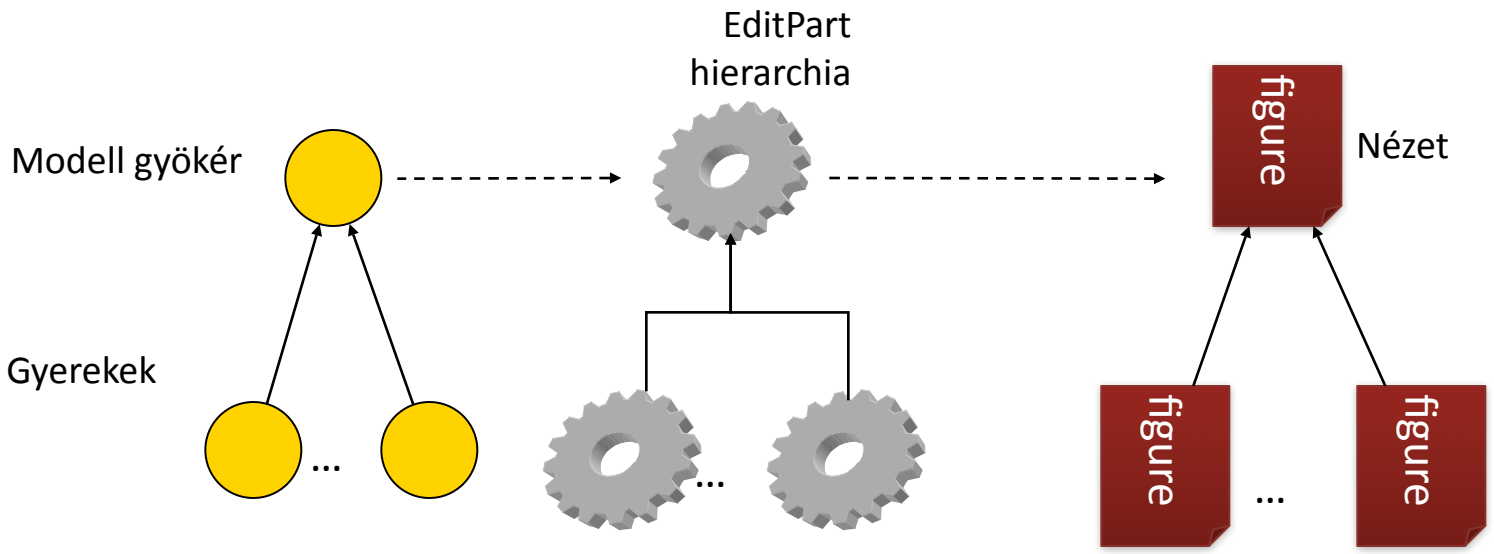


MVC a GEF-ben: Controller

- Vezérlő: EditPart osztályok
  - GEF „lelke”
  - Kapcsolat a modell és a nézet között
  - 1 Figure <-> 1 EditPart
  - 1 modell elem -> több EditPart is lehet
  - 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 SocialNetworkEditPartFactory
    implements EditPartFactory {

  public EditPart createEditPart(EditPart context,
      Object model) {
    EditPart editPart = null;
    if (model instanceof SocialNetwork) {
      editPart = new SocialNetworkEditPart();
    } else if (model instanceof Person) {
      editPart = new PersonEditPart();
    }
    if (editPart != null) {
      editPart.setModel(model);
    }
    return editPart;
  }

}
```

EditPartFactory

- Modell alapján EditPart generálása

```
public class SocialNetworkEditPartFactory
    implements EditPartFactory {

  public EditPart createEditPart(EditPart context,
      Object model) {
    EditPart editPart = null;
    if (model instanceof SocialNetwork) {
      editPart = new SocialNetworkEditPar
    } else if (model instanceof Person) {
      editPart = new PersonEditPart();
    }
    if (editPart != null) {
      editPart.setModel(model);
    }
    return editPart;
  }

}
```

Szülő EditPart

EditPartFactory

- Modell alapján EditPart generálása

```
public class SocialNetworkEditPartFactory
    implements EditPartFactory {

  public EditPart createEditPart(EditPart context,
      Object model) {
    EditPart editPart = null;
    if (model instanceof SocialNetwork) {
      editPart = new SocialNetworkEditPart();
    } else if (model instanceof Person) {
      editPart = new PersonEditPart();
    }
    if (editPart != null) {
      editPart.setModel(model);
    }
    return editPart;
  }

}
```

EditPart tárol egy modell referenciát

Nézet legenerálása

- EditPart feladata
- Sajátunkat célszerű származtatni az AbstractGraphicalEditPart osztályból

```
public class PersonEditPart extends
  AbstractGraphicalEditPart {

    @Override
    protected IFigure createFigure() {
      // Saját Figure létrehozása
      return new PersonFigure();
    }

  }
```

Modell bejárása

- EditPartViewer tartalma
  - 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 SocialNetworkEditPart extends
  AbstractGraphicalEditPart {

    @Override
    protected List<?> getModelChildren() {
      SocialNetwork socialNetwork =
        ((SocialNetwork) getModel());
      return socialNetwork.getEntities();
    }

  }
```

Modell bejárása

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

```
public class SocialNetworkEditPart extends
  AbstractGraphicalEditPart {

    @Override
    protected List<?> getM...en() {
      SocialNetwork socialN...work =
          ((SocialNetwork) getModel());
      return socialNetwork.getEntities();
    }

  }
```

Modell lekérdezése

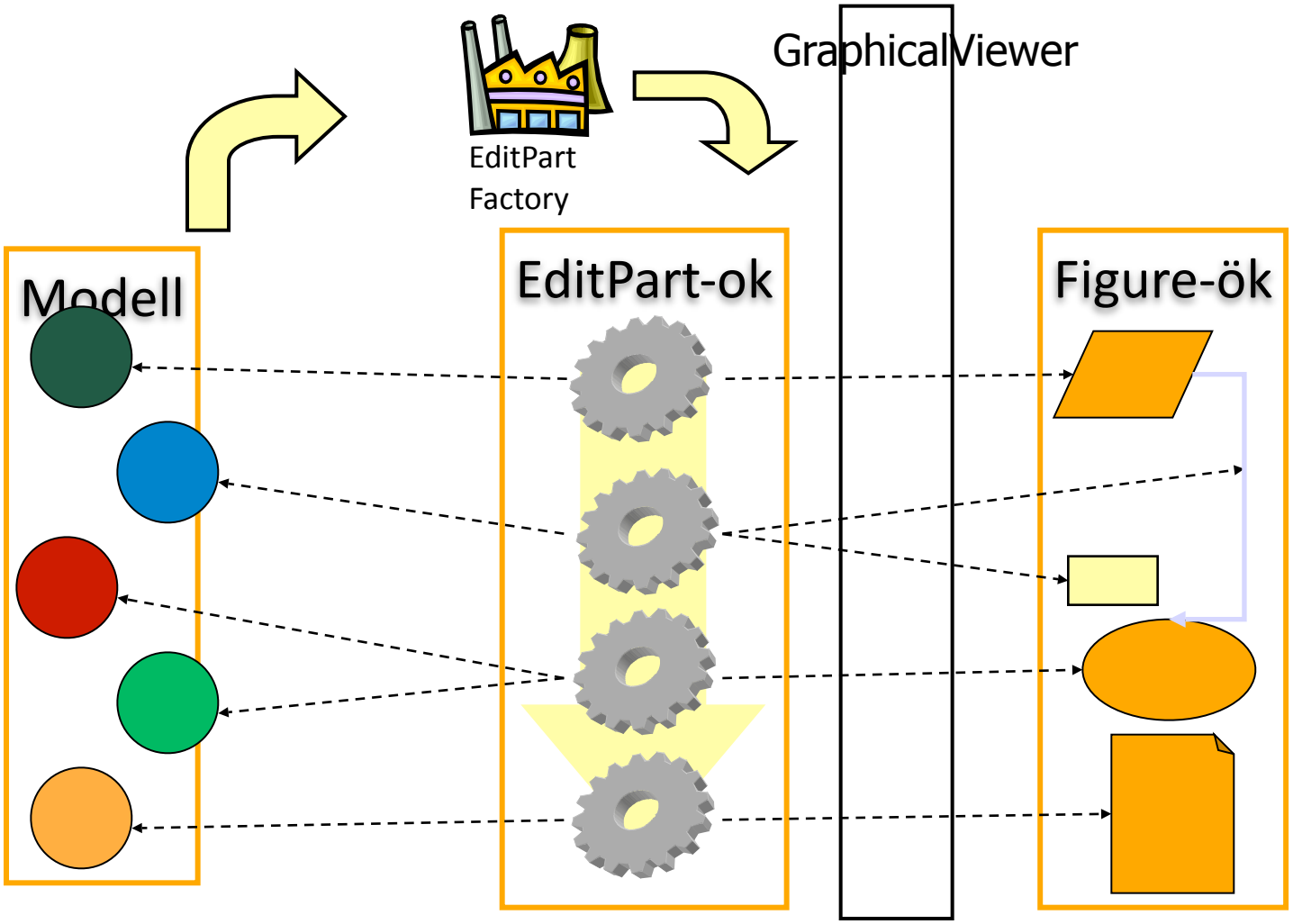
Modell bejárása

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

```
public class SocialNetworkEditPart extends
  AbstractGraphicalEditPart {

    @Override
    protected List<?> getModelChildren() {
      SocialNetwork socialNetwork =
        ((SocialNetwork) getModel());
      return socialNetwork.getEntities();
    }

  }
```

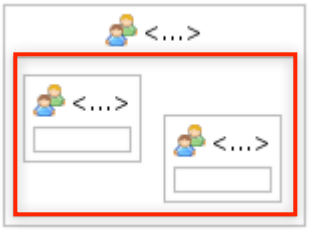
Modellfüggő, nem GEF-specifikus

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 ((CommunityFigure) getFigure())
      .getSubcommunityContainer();
}
```

Nézet frissítése modellváltozáskor

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

EditPart példa

```
public class PersonEditPart extends AbstractGraphicalEditPart implements IModelListener {
  @Override
  protected void refreshVisuals() {
    ((PersonFigure)getFigure()).getLabel().setText(
      ((Person)getModel()).getName());
  }
  @Override
  public void activate() {
    super.activate();
    ((Notifiable)getModel()).addListener(this);
  }
  @Override
  public void deactivate() {
    ((Notifiable)getModel()).removeListener(this);
    super.deactivate();
  }
  @Override
  public void modelChanged() {
    refreshVisuals();
    refreshChildren();
  }
}
```

EditPart példa

```
public class PersonEditPart extends AbstractGraphicalEditPart implements IModelListener {
  @Override
  protected void refreshVisuals() {
    ((PersonFigure)getFigure()).getLabel().setText(
      ((Person)getModel()).getName());
  }
  @Override
  public void activate() {
    super.activate();
    ((Notifiable)getModel()).addListener(this);
  }
  @Override
  public void deactivate() {
    ((Notifiable)getModel()).removeListener(this);
    super.deactivate();
  }
  @Override
  public void modelChanged() {
    refreshVisuals();
    refreshChildren();
  }
}
```

Nézet frissítése

EditPart példa

```
public class PersonEditPart extends AbstractGraphicalEditPart implements IModelListener {
  @Override
  protected void refreshVisuals() {
    ((PersonFigure)getFigure()).getLabel().setText(
      ((Person)getModel()).getName());
  }
  @Override
  public void activate() {
    super.activate();
    ((Notifiable)getModel()).addListener(this);
  }
  @Override
  public void deactivate() {
    ((Notifiable)getModel()).removeListener(this);
    super.deactivate();
  }
  @Override
  public void modelChanged() {
    refreshVisuals();
    refreshChildren();
  }
}
```

Modellfigyelés kezdete

EditPart példa

```
public class PersonEditPart extends AbstractGraphicalEditPart implements IModelListener {
  @Override
  protected void refreshVisuals() {
    ((PersonFigure)getFigure()).getLabel().setText(
      ((Person)getModel()).getName());
  }
  @Override
  public void activate() {
    super.activate();
    ((Notifiable)getModel()).addListener(this);
  }
  @Override
  public void deactivate() {
    ((Notifiable)getModel()).removeListener(this);
    super.deactivate();
  }
  @Override
  public void modelChanged() {
    refreshVisuals();
    refreshChildren();
  }
}
```

Modellfigyelés vége

EditPart példa

```
public class PersonEditPart extends AbstractGraphicalEditPart implements IModelListener {
  @Override
  protected void refreshVisuals() {
    ((PersonFigure)getFigure()).getLabel().setText(
      ((Person)getModel()).getName());
  }
  @Override
  public void activate() {
    super.activate();
    ((Notifiable)getModel()).addListener(this);
  }
  @Override
  public void deactivate() {
    ((Notifiable)getModel()).removeListener(this);
    super.deactivate();
  }
  @Override
  public void modelChanged() {
    refreshVisuals();
    refreshChildren();
  }
}
```

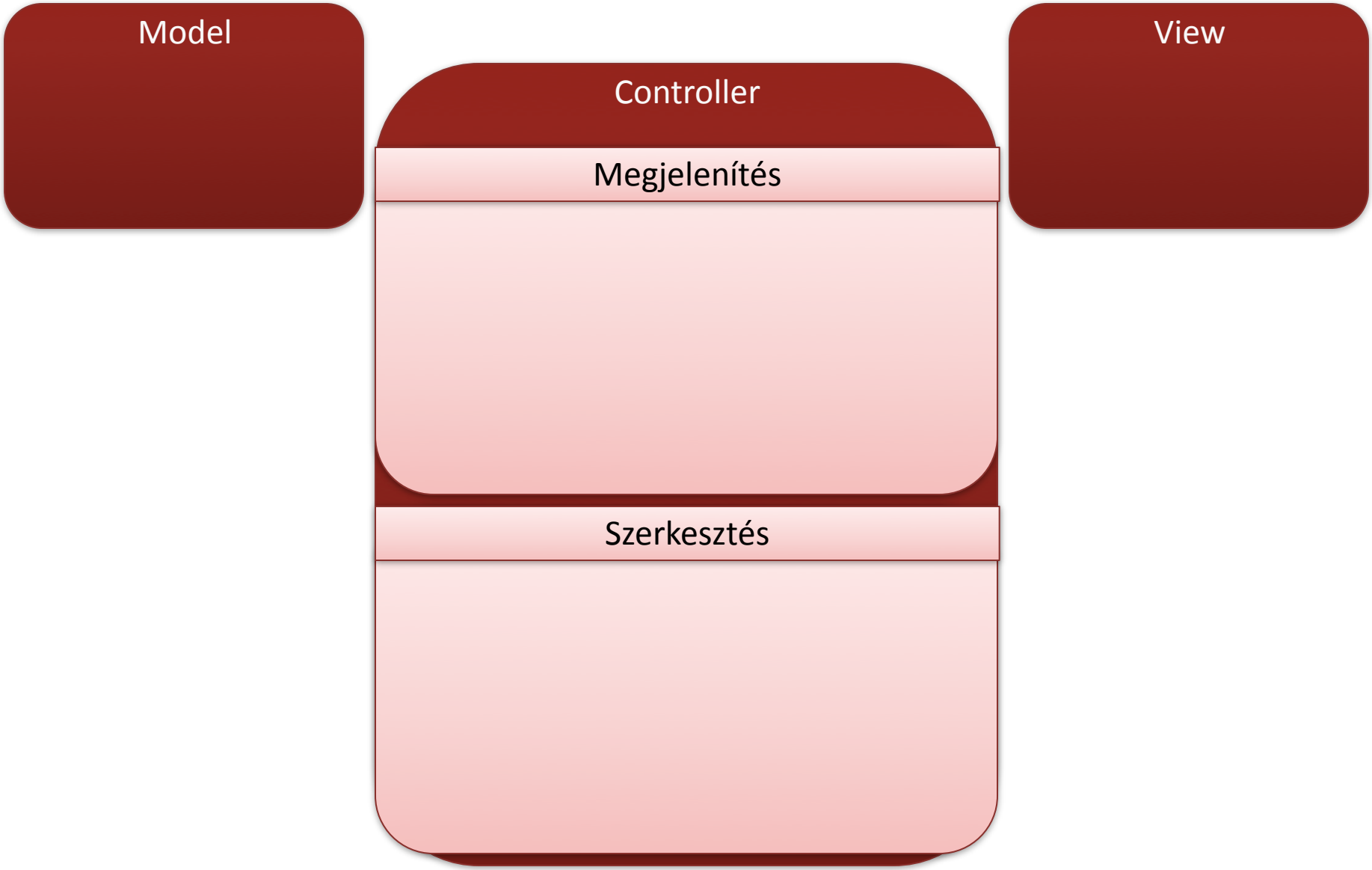
Az értesítő ezt hívja

EditPart példa

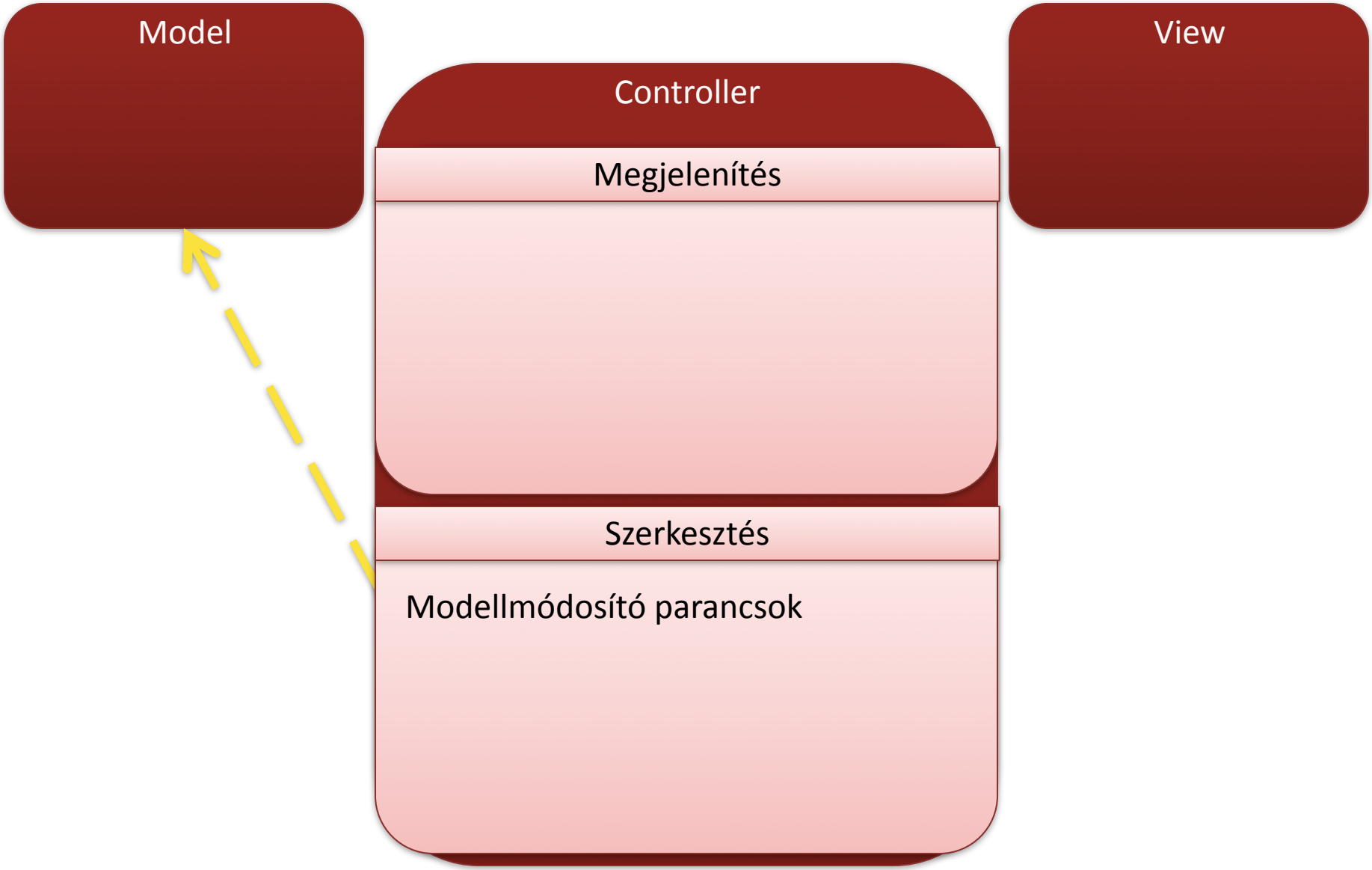
```
public class PersonEditPart extends AbstractGraphicalEditPart implements IModelListener {
  @Override
  protected void refreshVisuals() {
    ((PersonFigure)getFigure()).getLabel().setText(
      ((Person)getModel()).getName());
  }
  @Override
  public void activate() {
    super.activate();
    ((Notifiable)getModel()).addListener(this);
  }
  @Override
  public void deactivate() {
    ((Notifiable)getModel()).removeListener(this);
    super.deactivate();
  }
  @Override
  public void modelChanged() {
    refreshVisuals();
    refreshChildren();
  }
}
```

Gyerekek frissítése

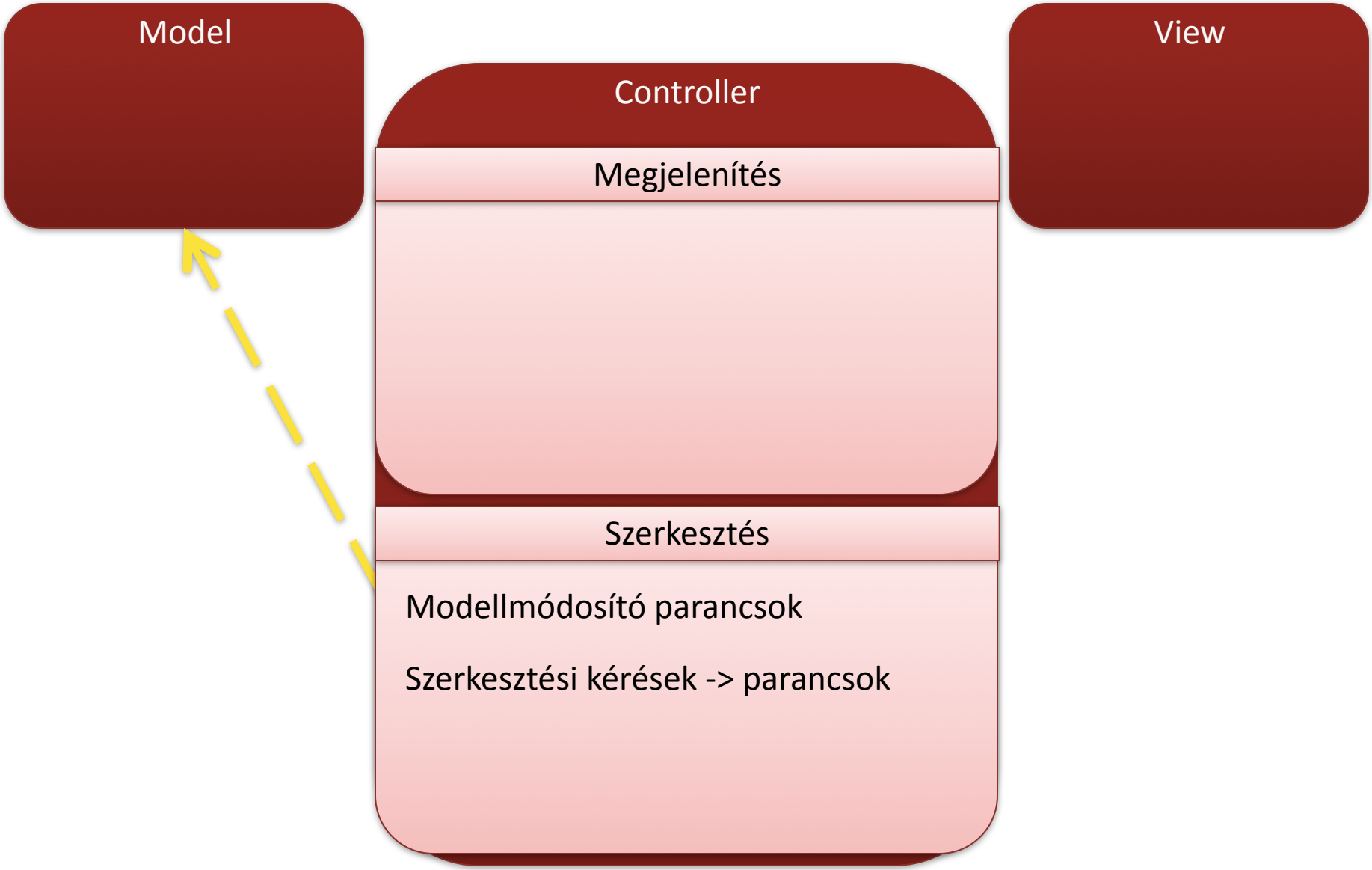
GEF workflow



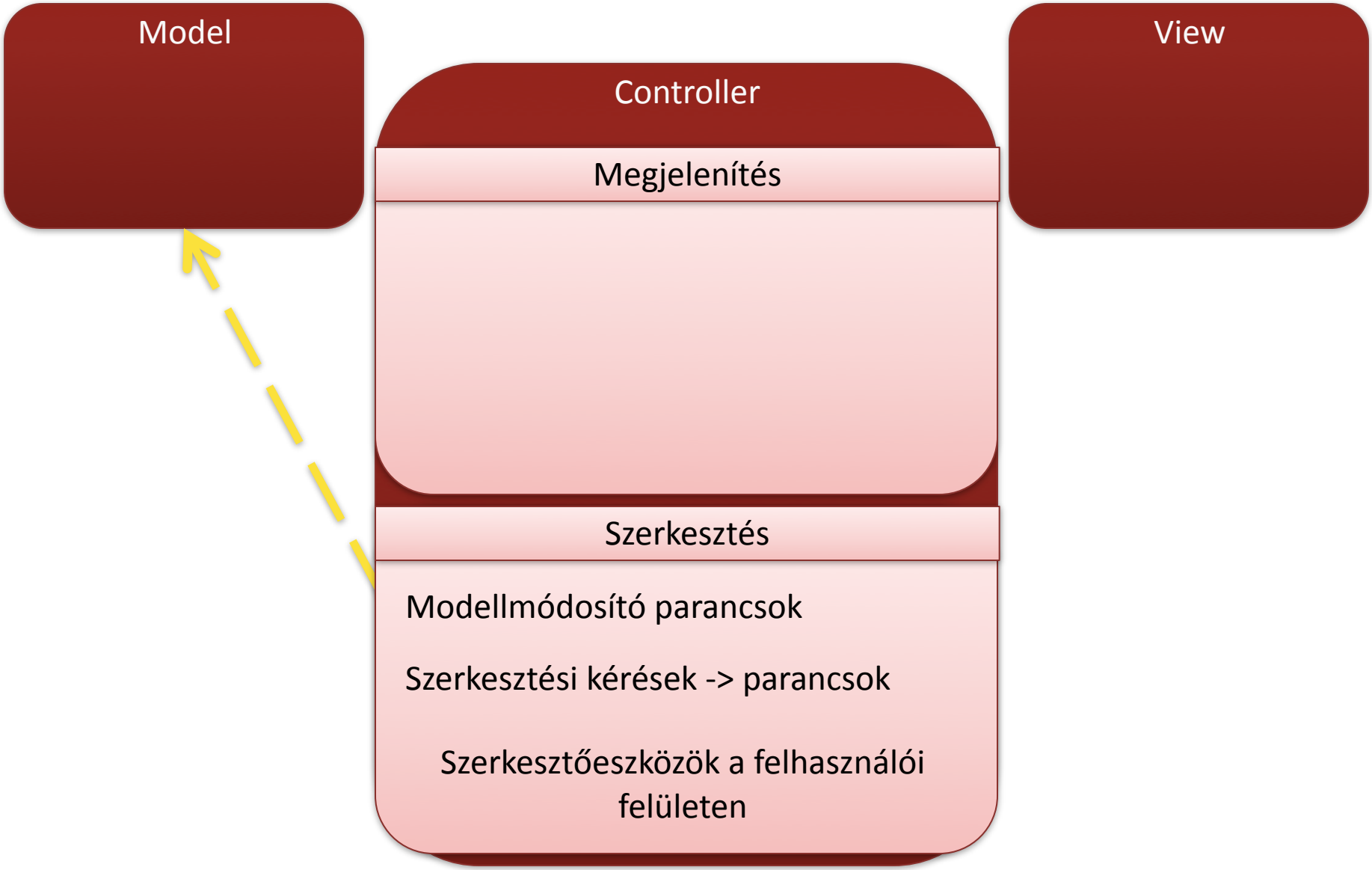
GEF workflow



GEF workflow



GEF workflow



Szerkesztés szereplői

- EditDomain: fogadja az eseményeket az SWT-től, és továbbítja az aktív Toolnak
  - Nem végez feldolgozást, csak összefogja egy modell összes nézetét
- Tool: egy szerkesztési funkciót jelképez
  - Feldolgozza az SWT üzeneteket
  - Létrehoz egy (vagy több) Request-et

Szerkesztés szereplői

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

Szerkesztés szereplői

- 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

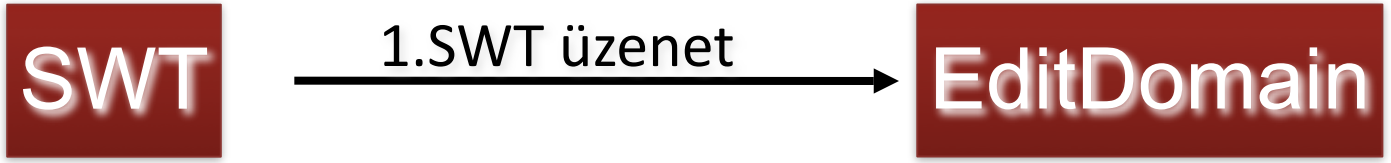
- Command
  - A modell módosítását végzi
  - Visszavonható (ha megírjuk ☺)
- CommandStack
  - Végrehajtott Commandok verme
  - Ez biztosítja az undo/redo lehetőségét
  - EditDomainenként pontosan egy darab
  - Mindig ezen keresztül módosítsunk!

Szerkesztés szereplői

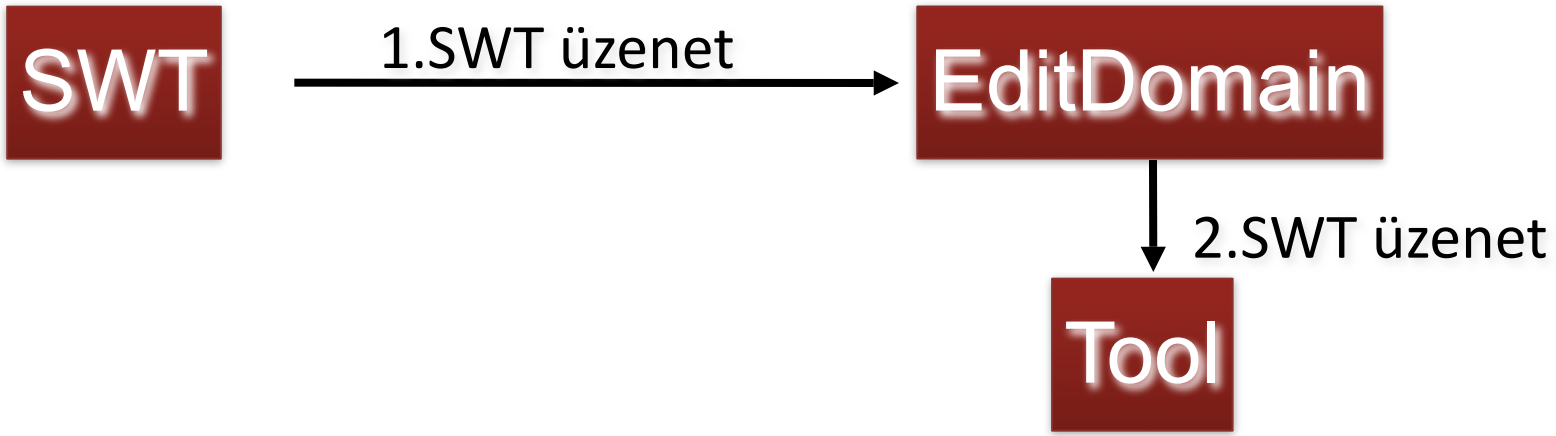
- Action
  - Nem GEF, hanem JFace-specifikus
  - Felhasználói akció belső reprezentációja
    - Előhívás: menü, eszközsor, billentyűparancs
  - Undo/redo: GEF biztosít néhány wrappert, amik lehetővé teszik a CommandStack egyszerű elérését
  - ActionRegistry: actionök listája
    - Több helyen szereplő azonos actionökhöz
- Nincs több (lényeges) szereplő

Szerkesztés folyamata

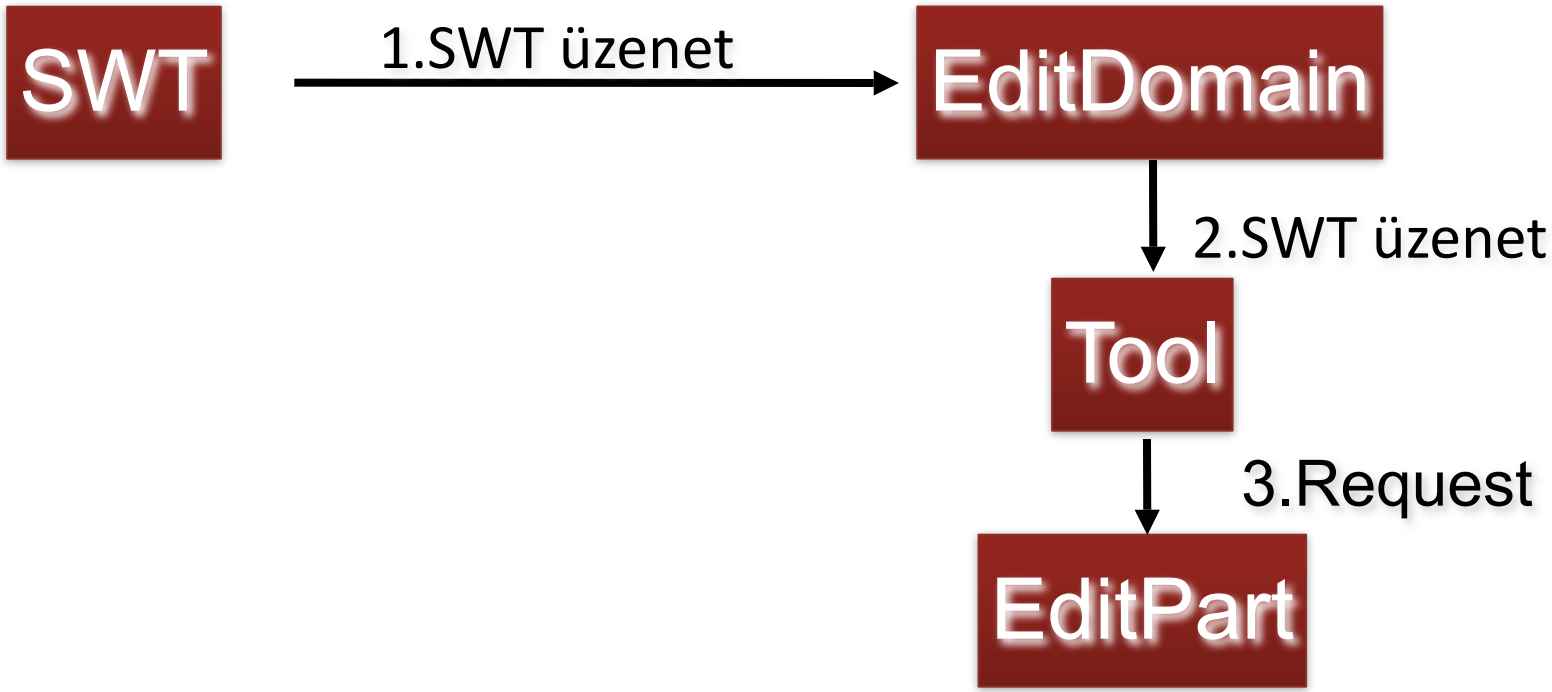
Szerkesztés folyamata



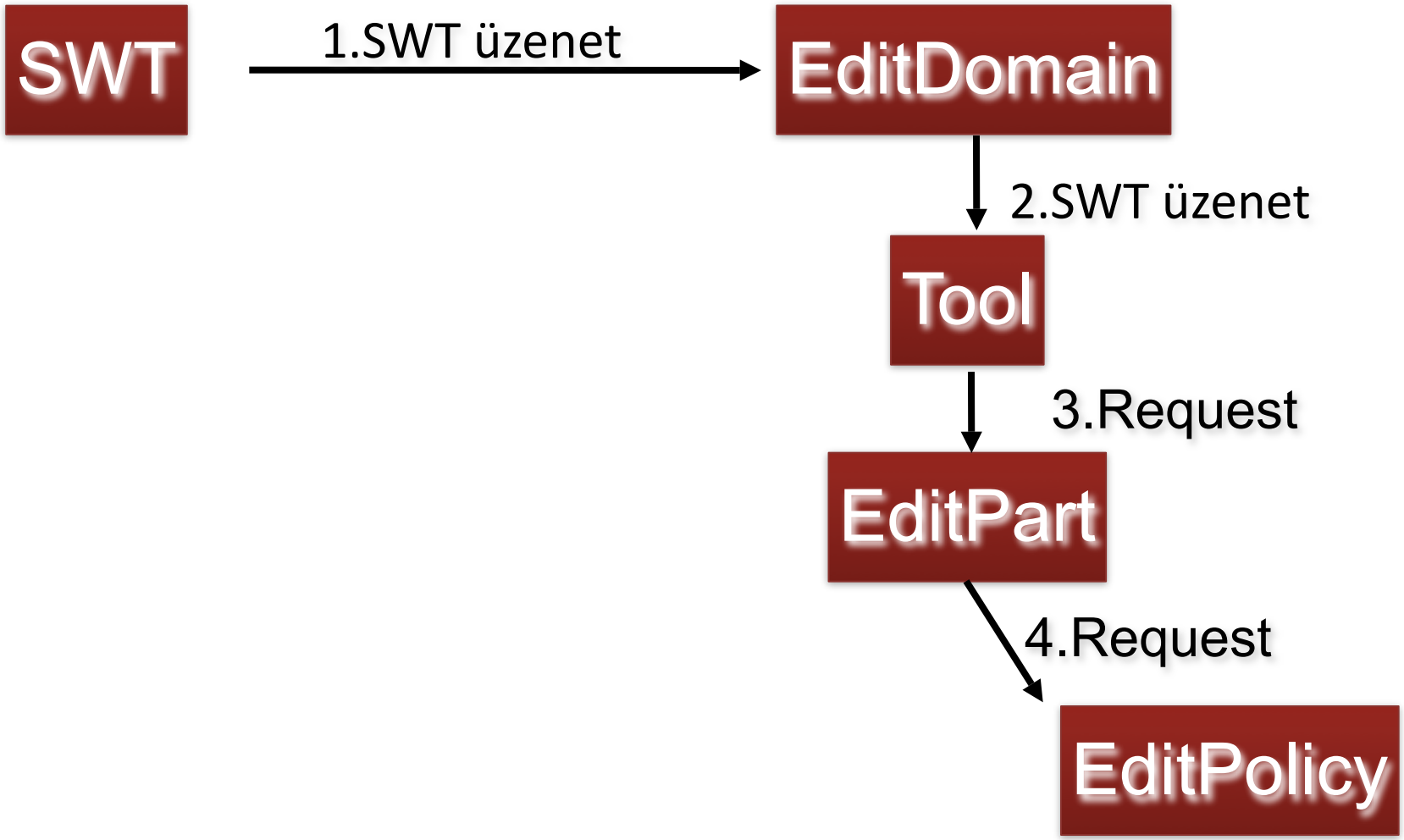
Szerkesztés folyamata



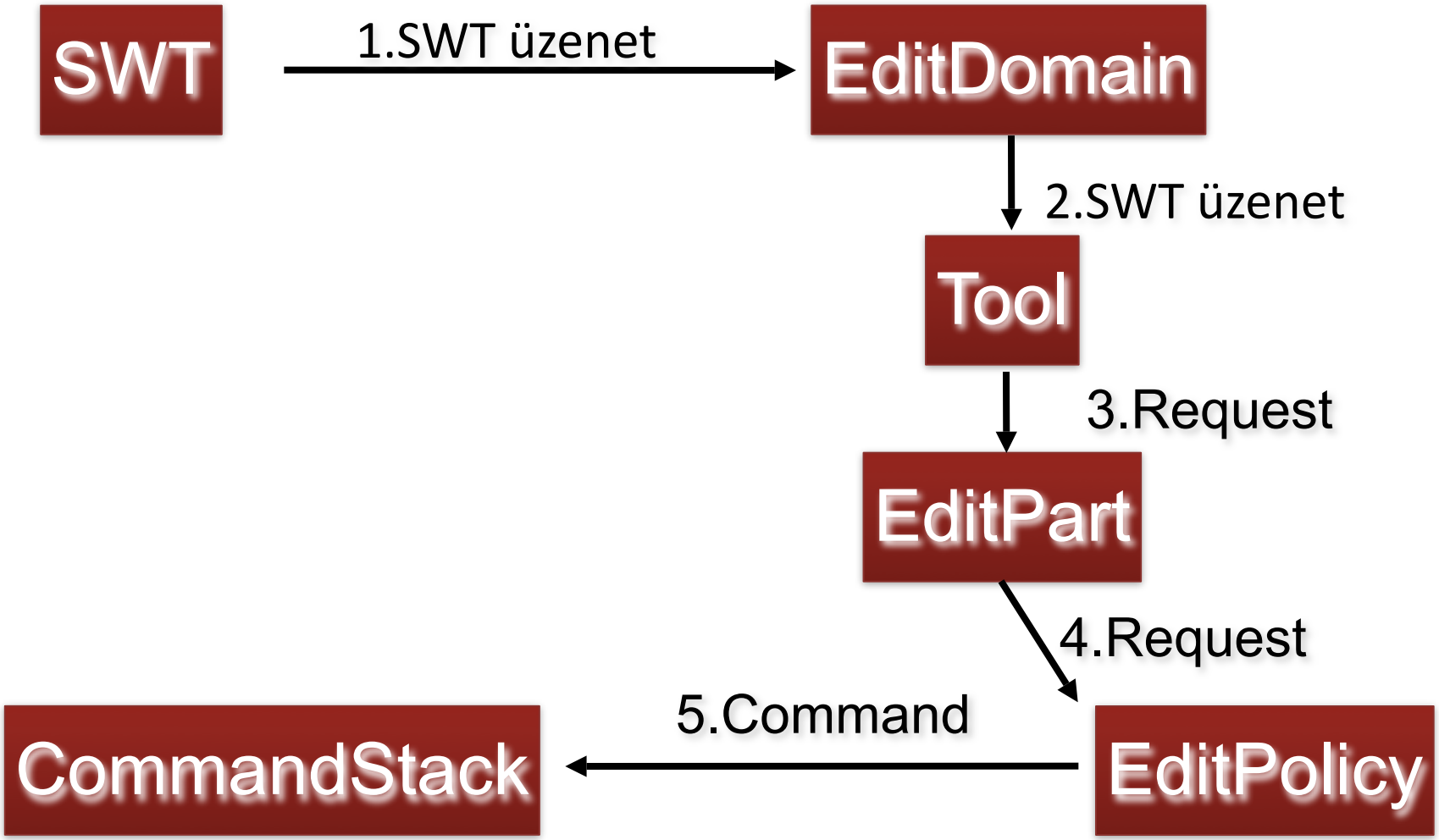
Szerkesztés folyamata



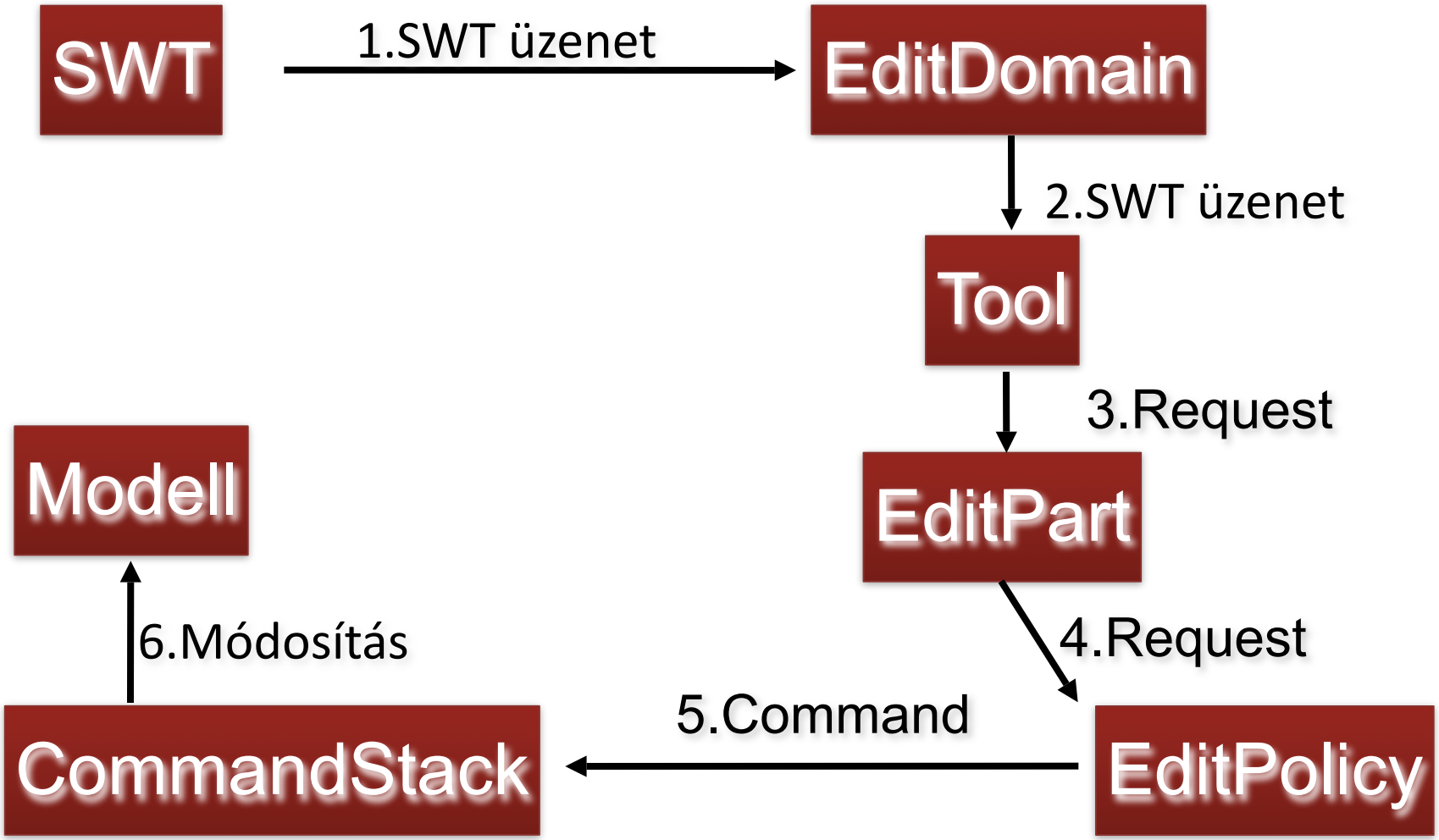
Szerkesztés folyamata



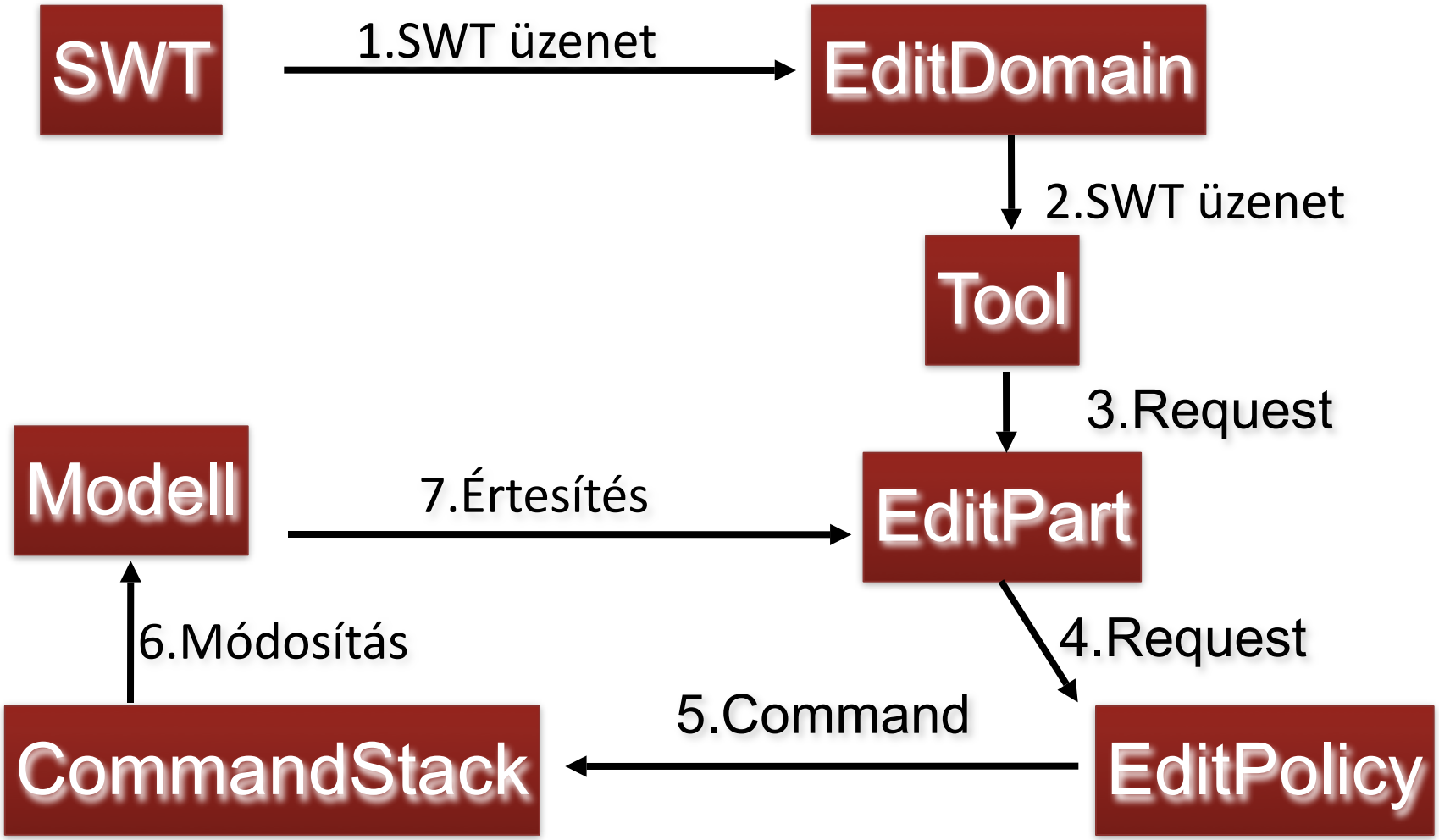
Szerkesztés folyamata



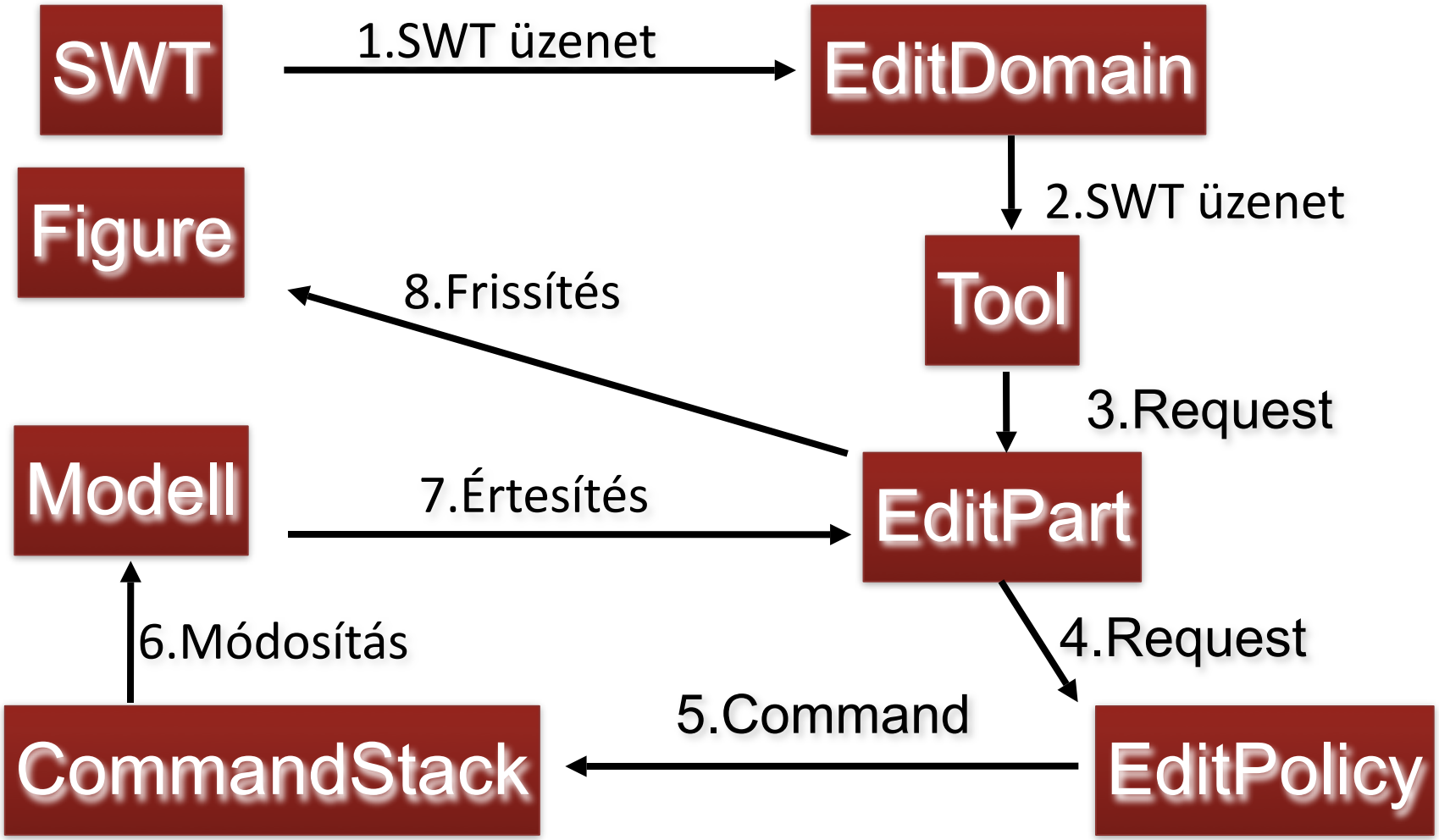
Szerkesztés folyamata



Szerkesztés folyamata



Szerkesztés folyamata



EditDomain

- A szerkesztő állapota
- Aktív eszköz (active tool)
  - Éppen használt szerkesztő funkció
  - Pl. kijelölés, új elem, törlés
- CommandStack
  - Elvégzett módosítások listája
  - Undo / redo támogatáshoz
- Használjuk mindig a DefaultEditDomain-t

Tool

- Beépített:
  - SelectionTool, CreationTool, MarqueeTool
- Saját Tool is készíthető
  - TargetingTool: Ha van egy cél EditPart
  - AbstractTool: teljesen általános
- Aktív tool beállítása
  - EditDomain.setActiveTool()
  - Paletta: lásd később

Request

- ChangeBoundsRequest: átméretezés
- CreationRequest: elem létrehozása
- Minden Requesthez tartozik egy típus azonosító
  - RequestConstants osztályban
  - EditPolicy-k ez alapján azonosítják
  - Példák:
    - REQ_CREATE – létrehozás
    - REQ_DELETE – törlés
    - REQ_MOVE – mozgatás
    - REQ_RESIZE – átméretezés

EditPolicy

- Pontosan egy EditParthoz tartozik
  - getHost() hívással lekérdezhető
  - 1 EditPart -> több EditPolicy lehet
- Szerep azonosítás: string kulcsokkal
  - EditPolicy osztályban konstansok, pl.:
    - COMPONENT_ROLE: alapvető műveletek (pl. törlés)
    - SELECTION_FEEDBACK_ROLE: visszacsatolás kijelölésnél
- Feladatai
  - Request -> Command leképezés
    - Command getCommand(Request)
  - Grafikus visszajelzés a felhasználónak

EditPolicy

- Beépített absztrakt ősosztályok
  - Némi előfeldolgozást végeznek a Requesten
  - ComponentEditPolicy: törlés
  - LayoutEditPolicy: létrehozás, átméretezés
    - Pl. XYLayoutEditPolicy: ha az EditPart nézete XYLayoutot használ
      - Biztosítja a grafikus visszajelzést

EditPolicy példa

```
public class SocialNetworkLayoutEditPolicy extends XYLayoutEditPolicy {


  protected Command createChangeConstraintCommand(
       EditPart child, Object constraint) {
    if ((child.getModel() instanceof ElementModel) &&
        (constraint instanceof Rectangle)) {
      return new ChangeBoundsCommand(
          ((SocialEntity)child.getModel()),
          ((Rectangle)constraint));
    } else {
      return null;
    }
  }


}
```

EditPolicy példa

```
public class SocialNetworkLayoutEditPolicy extends XYLayoutEditPolicy {

  protected Command createChangeConstraintCommand(
       EditPart child, Object constraint) {
    if ((child.getModel() instanceof ElementModel) &&
        (constraint instanceof Rectangle)) {
      return new ChangeBoundsCommand(
          ((SocialEntity)child.getModel()),
          ((Rectangle)constraint));
    } else {
      return null;
    }
  }

}
```

XYLayout szülő

EditPolicy példa

```
public class SocialNetworkLayoutEditPolicy extends XYLayoutEditPolicy {

  protected Command createChangeConstraintCommand(
       EditPart child, Object constraint) {
    if ((child.getModel() instanceof ElementModel) &&
        (constraint instanceof Rectangle)) {
      return new ChangeBoundsCommand(
          ((SocialEntity)child.getModel()),
          ((Rectangle)constraint));
    } else {
      return null;
    }
  }

}
```

Átméretezés

EditPolicy példa

```
public class SocialNetworkLayoutEditPolicy extends XYLayoutEditPolicy {


  protected Command createChangeConstraintCommand(
       EditPart child, Object constraint) {
    if ((child.getModel() instanceof ElementModel) &&
        (constraint instanceof Rectangle)) {
      return new ChangeBoundsCommand(
          ((SocialEntity)child.getModel()),
          ((Rectangle)constraint));
    } else {
      return null;
    }
  }


}
```

Saját Command

Command példa

```
public class ChangeBoundsCommand extends Command {
  SocialEntity model;
  Rectangle newBounds, oldBounds;
  public MyResizeCommand(SocialEntity model, Rectangle newBounds) {
    this.model = model;
    this.newBounds = newBounds;
  }
  public boolean canExecute() {
    return ((newBounds.width >= MIN_WIDTH) && (newBounds.height >= MIN_HEIGHT));
  }
  public void execute() {
    oldBounds = model.getBounds();
    model.setBounds(newBounds);
  }
  public boolean canUndo() {
    return true;
  }
  public void undo() {
    model.setBounds(oldBounds);
  }
}
```

Command példa

```
public class ChangeBoundsCommand extends Command {
  SocialEntity model;
  Rectangle newBounds, oldBour
  public MyResizeCommand(Socia                    e newBounds) {
    this.model = model;
    this.newBounds = newBounds;
  }
  public boolean canExecute() {
    return ((newBounds.width >= MIN_WIDTH) && (newBounds.height >= MIN_HEIGHT));
  }
  public void execute() {
    oldBounds = model.getBounds();
    model.setBounds(newBounds);
  }
  public boolean canUndo() {
    return true;
  }
  public void undo() {
    model.setBounds(oldBounds);
  }
}
```

Végrehajthatóság feltétele

Command példa

```
public class ChangeBoundsCommand extends Command {
  SocialEntity model;
  Rectangle newBounds, oldBounds;
  public MyResizeCommand(SocialEntity model, Rectangle newBounds) {
    this.model = model;
    this.newBounds = newBounds;
  }
  public boolean canExecute() {
    return ((newBounds.width >= M                nds.height >= MIN_HEIGHT));
  }
  public void execute() {
    oldBounds = model.getBounds();
    model.setBounds(newBounds);
  }
  public boolean canUndo() {
    return true;
  }
  public void undo() {
    model.setBounds(oldBounds);
  }
}
```

Végrehajtás

Command példa

```
public class ChangeBoundsCommand extends Command {
  SocialEntity model;
  Rectangle newBounds, oldBounds;
  public MyResizeCommand(SocialEntity model, Rectangle newBounds) {
    this.model = model;
    this.newBounds = newBounds;
  }
  public boolean canExecute() {
    return ((newBounds.width >= MIN_WIDTH) && (newBounds.height >= MIN_HEIGHT));
  }
  public void execute() {
    oldBounds = model.getBounds();
    model.setBounds(newBounds);
  }
  public boolean canUndo() {
    return true;
  }
  public void undo() {
    model.setBounds(oldBounds);
  }
}
```

Visszavonhatóság feltétele

Command példa

```
public class ChangeBoundsCommand extends Command {
  SocialEntity model;
  Rectangle newBounds, oldBounds;
  public MyResizeCommand(SocialEntity model, Rectangle newBounds) {
    this.model = model;
    this.newBounds = newBounds;
  }
  public boolean canExecute() {
    return ((newBounds.width >= MIN_WIDTH) && (newBounds.height >= MIN_HEIGHT));
  }
  public void execute() {
    oldBounds = model.getBounds();
    model.setBounds(newBounds);
  }
  public boolean canUndo() {
    return true;
  }
  public void undo() {
    model.setBounds(oldBounds);
  }
}
```

Visszavonás

EditPart és EditPolicy-k

- EditPolicy-k telepítése
  - `AbstractEditPart#createEditPolicies()` metódusban
  - `EditPart#installEditPolicy(Object role, EditPolicy editPolicy)` metódus segítségével

```
public class SocialNetworkEditPart
    extends AbstractGraphicalEditPart {

  protected void createEditPolicies() {
    installEditPolicy(EditPolicy.LAYOUT_ROLE,
      new SocialNetworkLayoutEditPolicy());
  }

}
```

EditPart és EditPolicy-k

- EditPolicy-k telepítése
  - `AbstractEditPart#createEditPolicies()` metódusban
  - `EditPart#installEditPolicy(Object role, EditPolicy editPolicy)` metódus segítségével

Policy telepítése

```
public class SocialNetworkEditPart
    extends AbstractGraphicalEditPart {

  protected void createEditPolicies() {
    installEditPolicy(EditPolicy.LAYOUT_ROLE,
      new SocialNetworkLayoutEditPolicy());
  }

}
```

EditPart és EditPolicy-k

- EditPolicy-k telepítése
  - `AbstractEditPart#createEditPolicies()` metódusban
  - `EditPart#installEditPolicy(Object role, EditPolicy editPolicy)` metódus segítségével

```
public class SocialNetworkEditPart
    extends AbstractGraphicalEditPart {

  protected void createEditPolicies() {
    installEditPolicy(EditPolicy.LAYOUT_ROLE,
      new SocialNetworkLayoutEditPolicy());
  }

}
```

Szerep azonosító

Mit kell nekünk megírni?

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

Mit kell nekünk megírni?

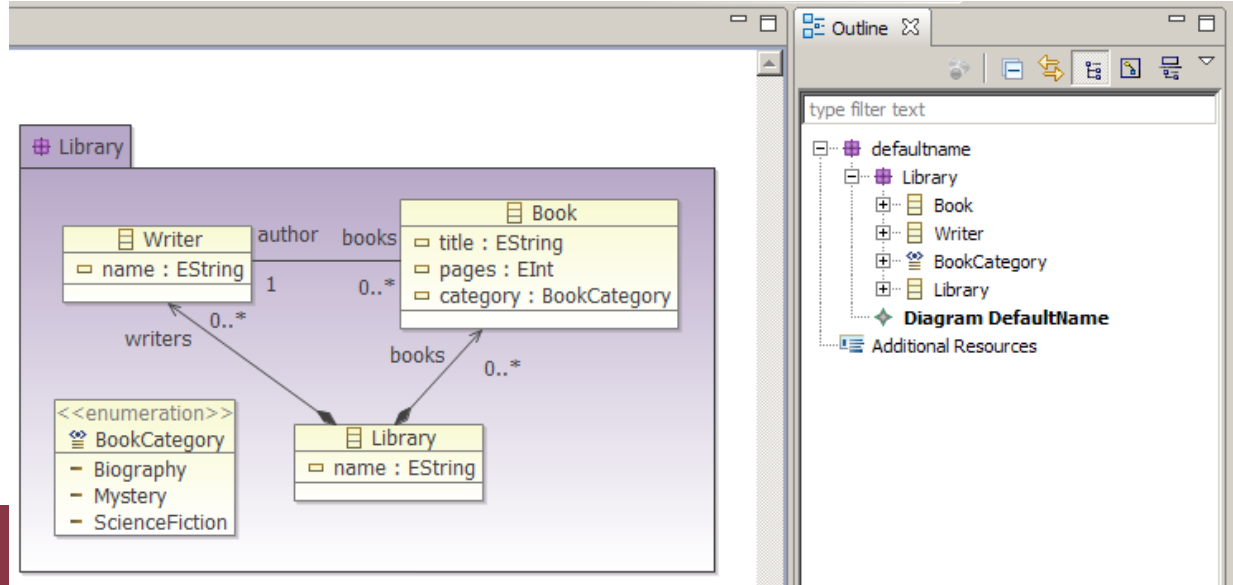
- Modellmódosító Commandok
- Saját EditPolicy-k: Requestek leképezése Commandokká
  - Milyen műveleteket engedünk meg
- EditPart oszályok - szerkesztés
  - EditPolicy-k hozzárendelése
- Editor és tartozékai

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 (lásd labor)
- 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



EditPartViewer

- Három szükséges alkotóelem
  - EditDomain: GEF alkalmazás „állapota”
  - EditPartFactory: modell -> EditPart leképzés
  - Gyökér modellelem

```
public class SocialNetworkGraphicalEditor extends EditorPart {
  public void createPartControl(Composite parent) {
    ScrollingGraphicalViewer viewer = new ScrollingGraphicalViewer();
    viewer.setEditDomain(new DefaultEditDomain(this));
    viewer.setEditPartFactory(new SocialNetworkEditPartFactory());
    viewer.setContents(modelRootElement);
    viewer.createControl(parent);
  }
}
```

GraphicalEditor

- Ősosztály GEF-es Eclipse editorokhoz
  - Létrehoz egy ScrollingGraphicalViewer-t
  - Létrehoz egy pár általános Actiont
    - Undo, redo, törlés, nyomtatás, mentés
    - Nem jeleníti meg őket sehol
  - Vigyázat: nem az API része, bármikor változhat az implementáció!

GraphicalEditor használata

```
public class SocialNetworkGraphicalEditor
     extends GraphicalEditor {
  public SocialNetworkGraphicalEditor() {
    setEditDomain(new DefaultEditDomain(this));
  }
  protected void configureGraphicalViewer() {
    getGraphicalViewer().setEditPartFactory(
      new SocialNetworkEditPartFactory());
  }
  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 SocialNetworkGraphicalEditor
    extends GraphicalEditor {
  public SocialNetworkGraphicalEditor() {
    setEditDomain(new DefaultEditDomain(this));
  }
  protected void configureGraphicalViewer() {
    getGraphicalViewer().setEditPartFactory(
      new SocialNetworkEditPartFactory());
  }
  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 SocialNetworkGraphicalEditor
     extends GraphicalEditor {
  public SocialNetworkGraphicalEditor() {
    setEditDomain(new DefaultEditDomain(this
  }
  protected void configureGraphicalViewer()
    getGraphicalViewer().setEditPartFactory(
      new SocialNetworkEditPartFactory());
  }
  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 SocialNetworkGraphicalEditor
     extends GraphicalEditor {
  public SocialNetworkGraphicalEditor() {
    setEditDomain(new DefaultEditDomain(this));
  }
  protected void configureGraphicalViewer() {
    getGraphicalViewer().setEditPartFactory(
      new SocialNetworkEditPartFactory());
  }
  public void init(IEditorSite site, IEditorInput input)
      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 SocialNetworkGraphicalEditor
     extends GraphicalEditor {
  public SocialNetworkGraphicalEditor() {
    setEditDomain(new DefaultEditDomain(this));
  }
  protected void configureGraphicalViewer() {
    getGraphicalViewer().setEditPartFactory(
      new SocialNetworkEditPartFactory());
  }
  public void init(IEditorSite site, IEditorI
      throws PartInitException {
    super.init(site, input);
    // Modell felépítése az input alapján
  }
  protected void initializeGraphicalViewer() {
    getGraphicalViewer().setContents(modelRoot);
  }
}
```

Modell gyökérelem megadása

Paletta

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

Gyakori ToolEntry-k

- SelectionToolEntry: kijelölés eszköz
- MarqueeToolEntry: csoportos kijelölés
- CreationToolEntry: elem létrehozása
  - Meg kell neki adni az elemeket létrehozó Factory-t
- Minden ToolEntry-hoz tartozik:
  - Név, rövid leírás, kis/nagy ikon
  - Tool osztály, amit példányosít

GraphicalEditorWithPalette

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

GraphicalEditorWithPalette példa

```
public class SocialNetworkGraphicalEditor 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());
    ImageDescriptor icon = Activator.getImageDescriptor("Person.png");
    root.add(new CreationToolEntry(Person.class.getSimpleName(),
        "Create a new person", new SimpleFactory(Person.class),
        icon, icon);
    return root;
  }
}
```

GraphicalEditorWithP… példa

Új paletta

```
public class SocialNetworkGraphicalEditor ex
     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());
    ImageDescriptor icon = Activator.getImageDescriptor("Person.png");
    root.add(new CreationToolEntry(Person.class.getSimpleName(),
        "Create a new person", new SimpleFactory(Person.class),
        icon, icon);
    return root;
  }
}
```

GraphicalEditorWithPalette példa

```
public class SocialNetworkGraphicalEditor 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());
    ImageDescriptor icon = Activator.getImageDescriptor("Person.png");
    root.add(new CreationToolEntry(Person.class.getSimpleName(),
        "Create a new person", new SimpleFactory(Person.class),
        icon, icon);
    return root;
  }
}
```

Új csoport

GraphicalEditorWithPalette példa

```
public class SocialNetworkGraphicalEditor 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());
    ImageDescriptor icon = Activator.getImageDescriptor("person.png");
    root.add(new CreationToolEntry(Person.class.getSimpleName(),
        "Create a new person", new SimpleFactory(Person.class),
        icon, icon);
    return root;
  }
}
```

Elválasztó vonal

GraphicalEditorWithPalette példa

```
public class SocialNetworkGraphicalEditor 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());
    ImageDescriptor icon = Activator.getImageDescriptor("Person.png");
    root.add(new CreationToolEntry(Person.class.getSimpleName(),
        "Create a new person", new SimpleFactory(Person.class),
        icon, icon);
    return root;
  }
}
```

Factory a modellelem létrehozásához

GEF workflow

Model

Értesítés

View

Kirajzolás

Elrendezés

Controller

Megjelenítés

Kapcsolat a modellel

Kapcsolat a nézettel

Hierarchia megadása

Nézet frissítése a modell változásakor

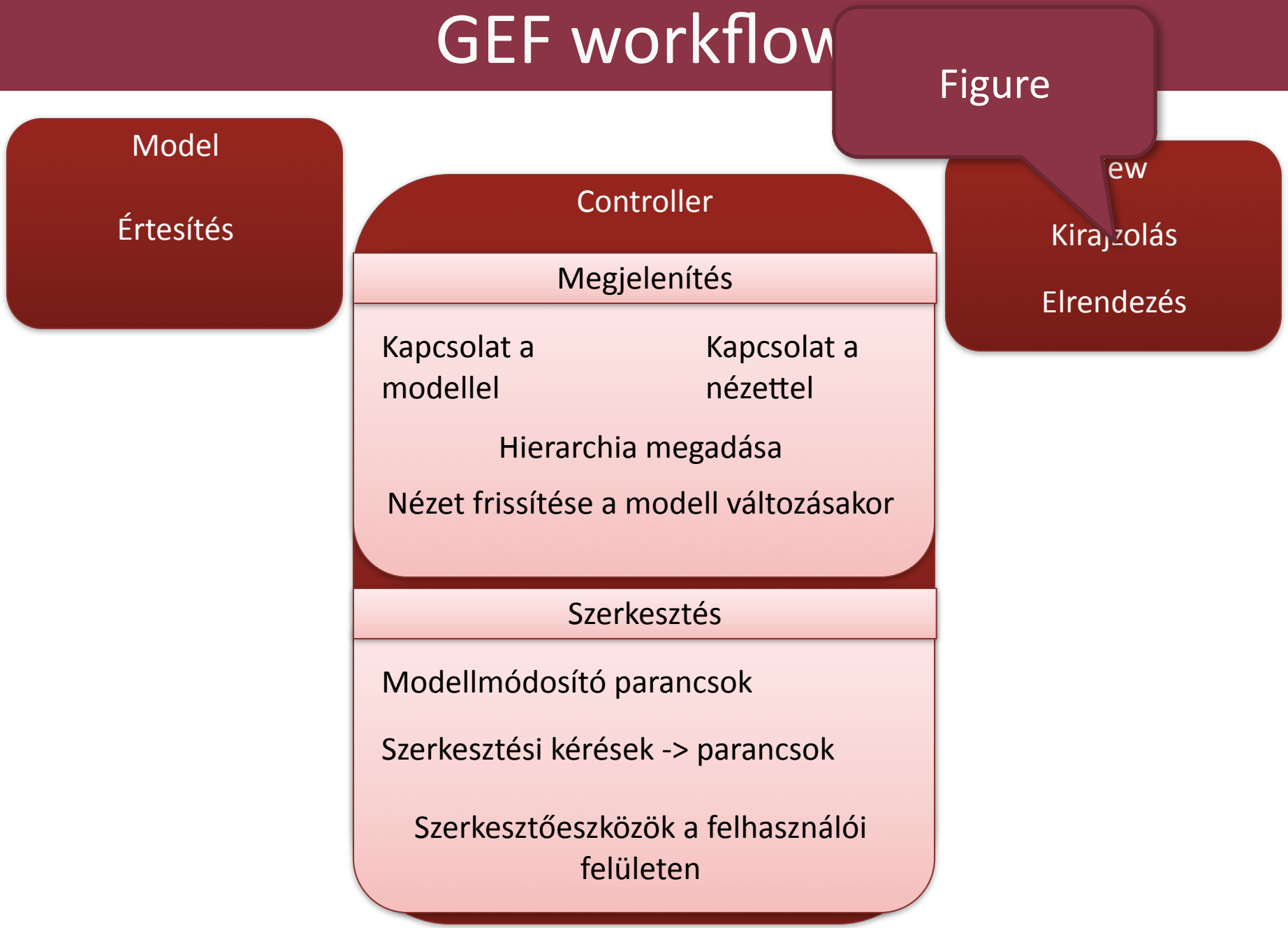
Szerkesztés

Modellmódosító parancsok

Szerkesztési kérések -> parancsok

Szerkesztőeszközök a felhasználói felületen

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

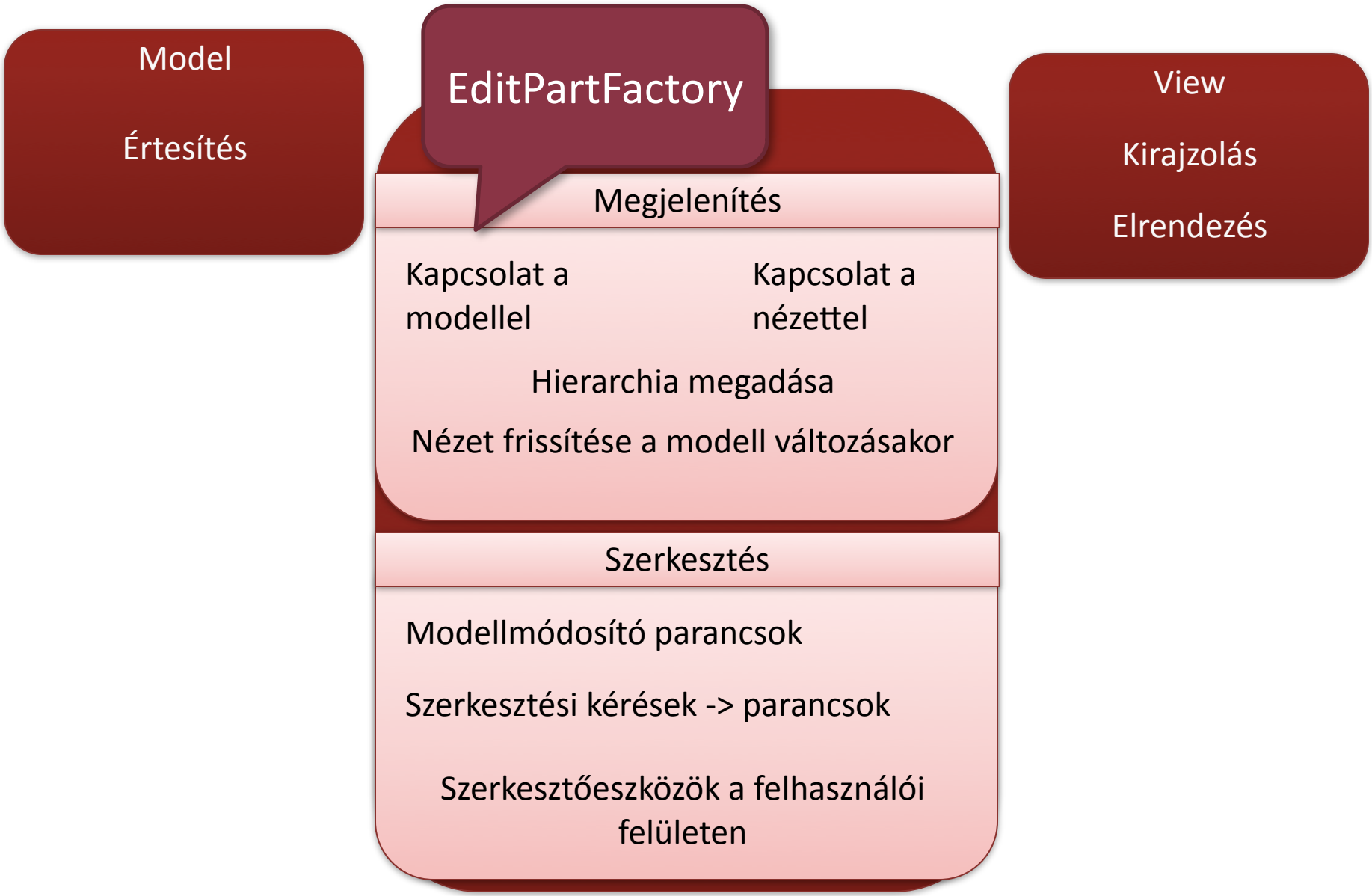
View

Kirajzolás

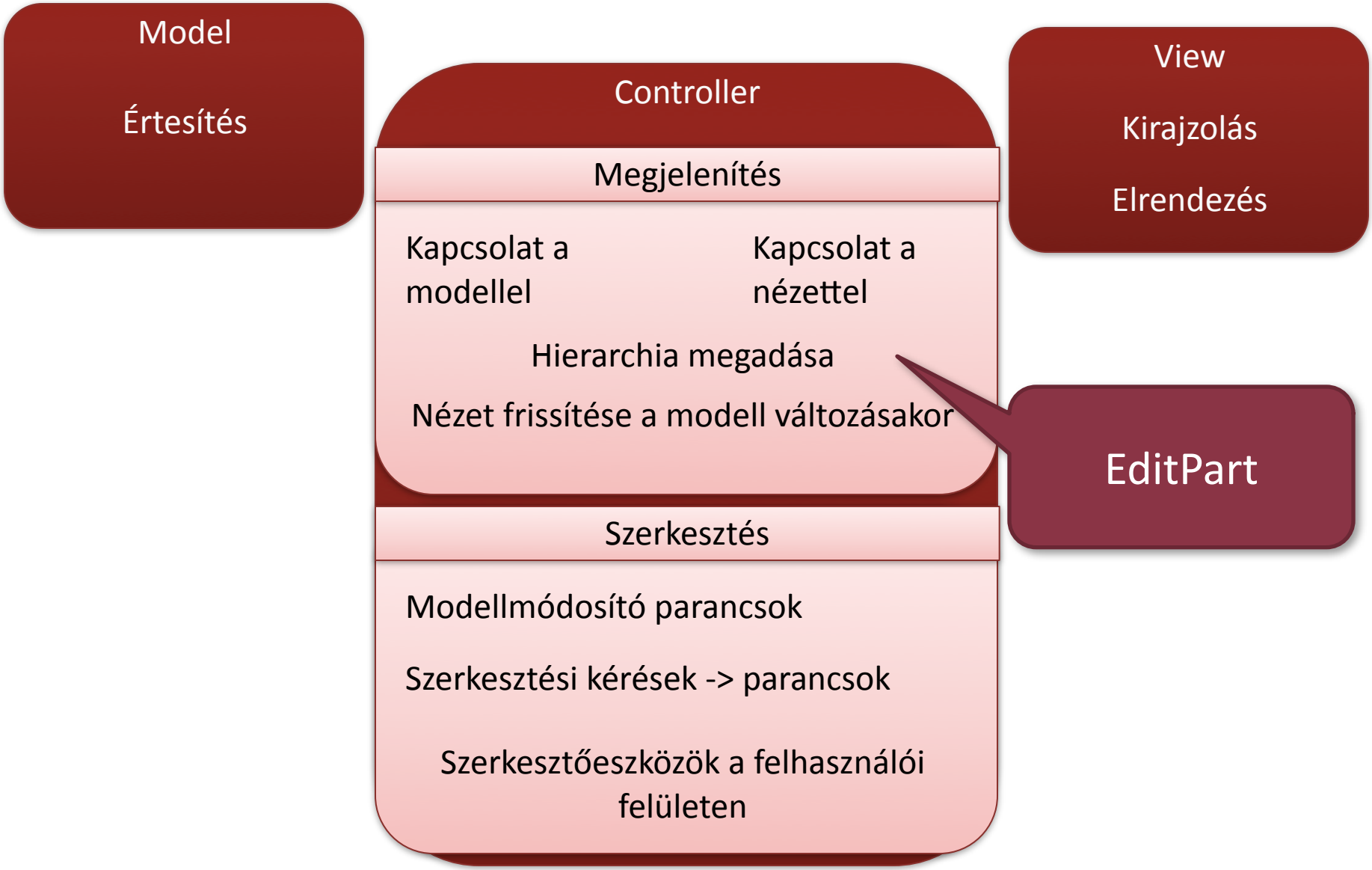
Elrendezés

LayoutManager

GEF workflow



GEF workflow



GEF workflow

Model

Értesítés

View

Kirajzolás

Elrendezés

Controller

Megjelenítés

Kapcsolat a modellel

Kapcsolat a nézettel

Hierarchia megadása

Nézet frissítése a modell változásakor

Command

Szerkesztés

Modellmódosító parancsok

Szerkesztési kérések -> parancsok

Szerkesztőeszközök a felhasználói felületen

GEF workflow

Model

Értesítés

View

Kirajzolás

Elrendezés

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

EditPolicy

GEF workflow

Model

Értesítés

View

Kirajzolás

Elrendezés

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

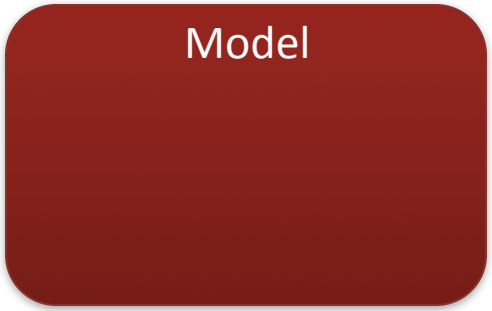
Tool

Összekötők használata a GEF-ben

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)

GEF workflow



GEF workflow

Model

Forrás, cél elérése

Összekötő modell

- Szintén semmi megkötés
- Két lehetőség:
  - Osztály reprezentálja
  - Attribútum reprezentálja
- Tudnia kell a saját forrását és célját

GEF workflow

Model

View

GEF workflow

Model

View

Kirajzolás

GEF workflow

Model

View

Kirajzolás

Elrendezés

Összekötő nézet

- Draw2D PolylineConnection példány
- Figure leszármazott -> lehetnek gyerekei
- GEF-ben nincs összekötő hierarchia
  - Mindegyik a teljes szerkesztőt kitöltheti
- Speciális elemek
  - ConnectionAnchor: végpontok helye
  - ConnectionRouter: összekötő alakja
  - RotatableDecoration: végpontok „dísze”

ConnectionAnchor

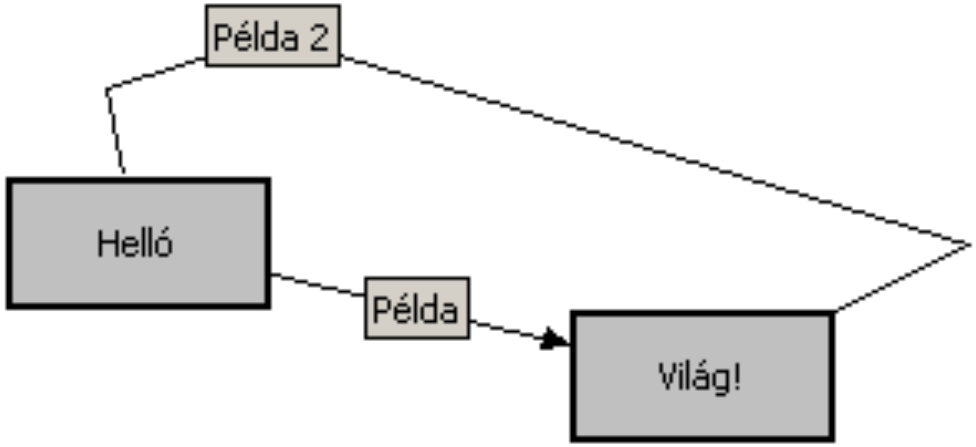
- Összekötők végpontjai
- Két beépített megvalósítás:
  - ChopboxAnchor: téglalap Figure-höz
  - EllipseAnchor: ellipszis Figure-höz
- Egyéb esetben kell sajátot írni

ConnectionRouter

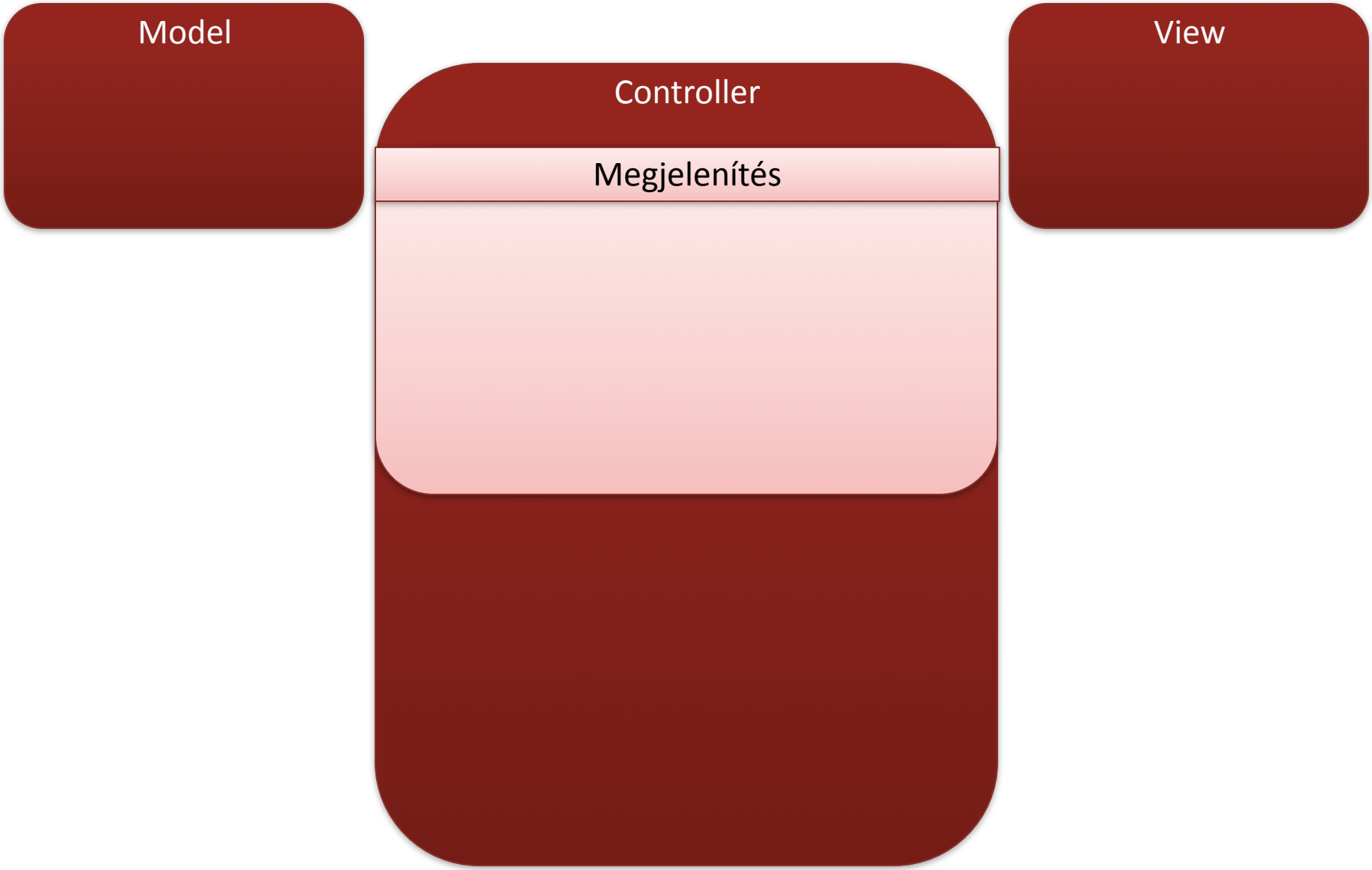
- A két Anchor közötti közbenső pontokat számolja ki
- ≈ LayoutManager, itt is lehet Constraint
- Típusai
  - NullRouter: egyenes vonal
  - BendpointConnectionRouter: kézi töréspontok
  - ManhattanConnectionRouter: automatikus derékszögű töréspontok
  - ShortestPathConnectionRouter: összekötő kikerüli az akadályokat
- Görbe összekötők támogatása
  - GEF-ben nincs beépítve, saját osztállyal kell megoldani

Összekötő nézet példa

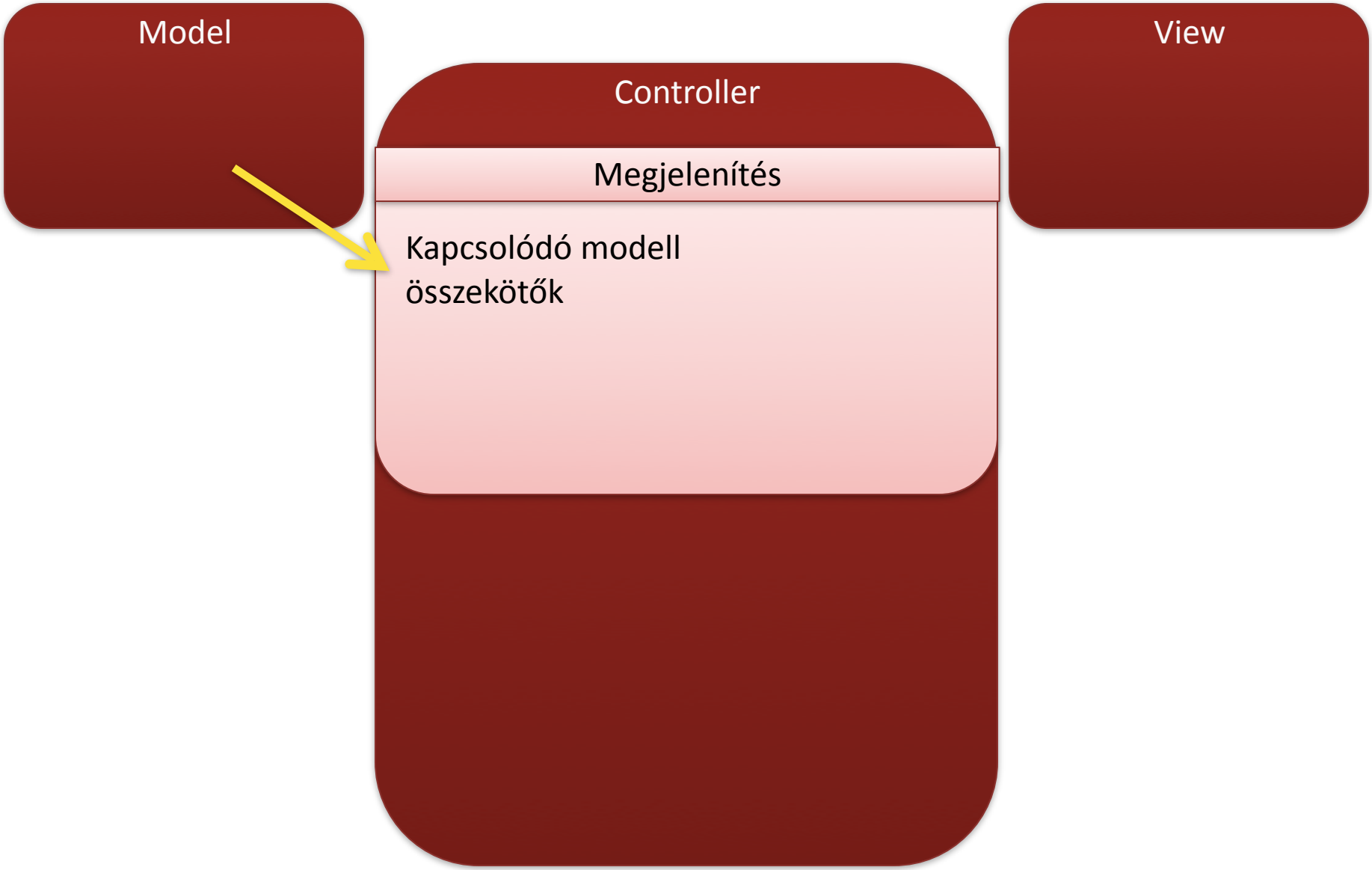
```
public class TestConnectionView extends PolylineConnection {
   private Label label;

   public TestConnectionView() {
        label = new Label();
        label.setOpaque(true);
        label.setBorder(new LineBorder());
        add(label, new ConnectionLocator(this,
                 ConnectionLocator.MIDDLE));

        PolygonDecoration decoration = new PolygonDecoration();
        decoration.setTemplate(PolygonDecoration.TRIANGLE_TIP);
        setTargetDecoration(decoration);

        setConnectionRouter(new BendpointConnectionRouter());
   }

   public setBendpoints(List<Bendpoint> bendpoints) {
        setRoutingConstraint(bendpoints);
   }

}
```



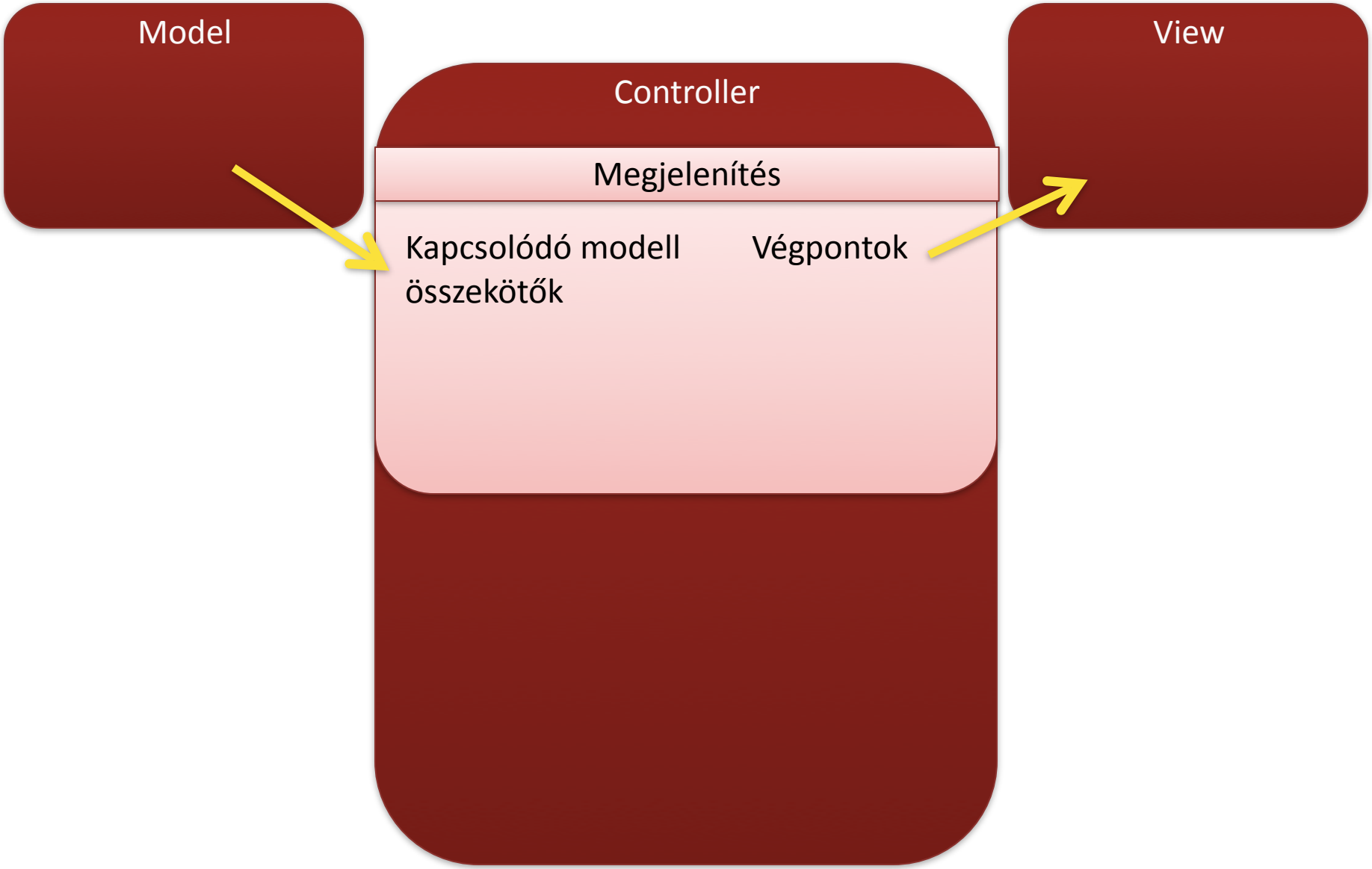
GEF workflow



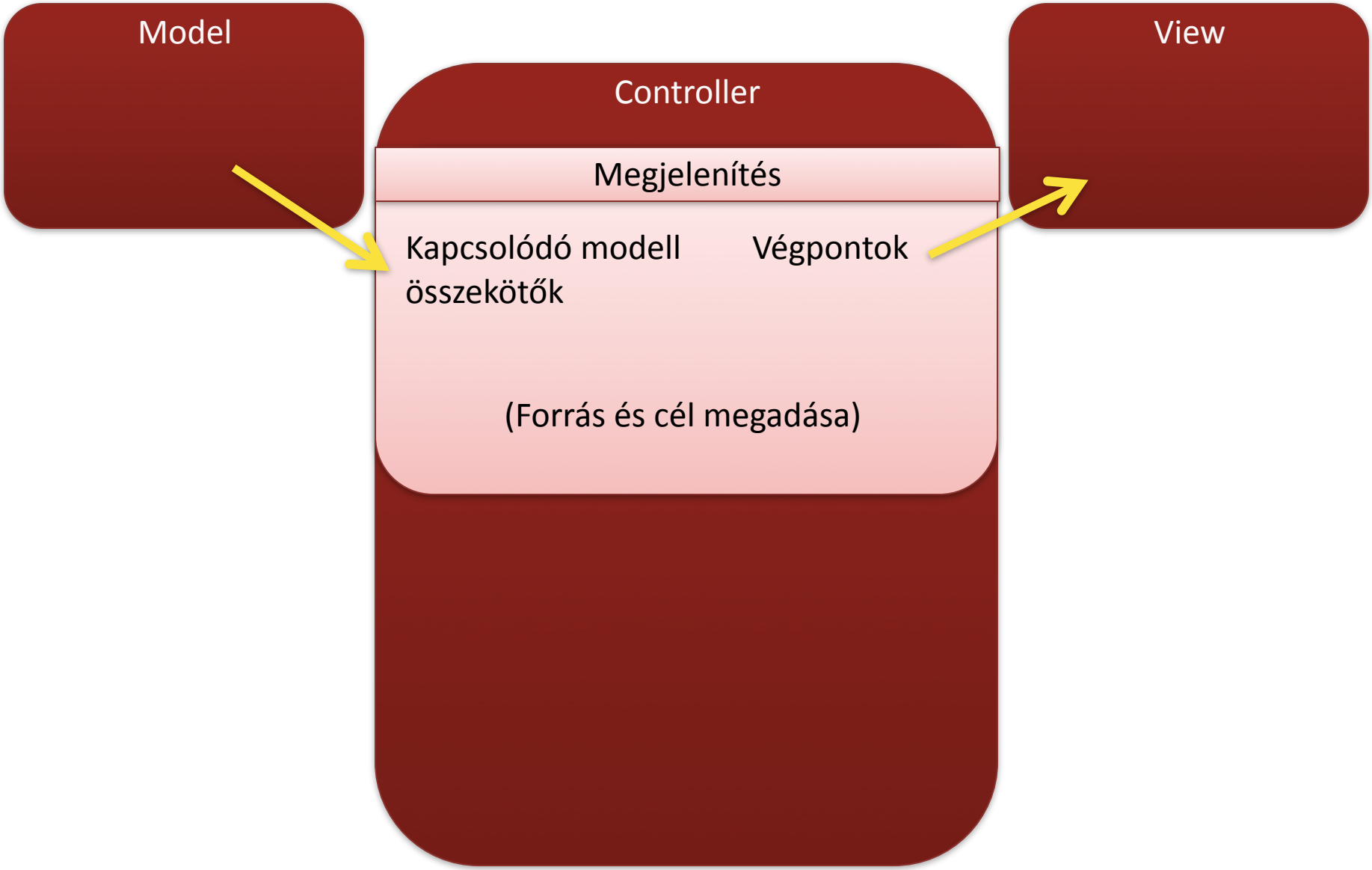
GEF workflow



GEF workflow



GEF workflow



Kapcsolat a modellel

- Forrás/cél objektumnak mindenképpen tudnia kell az összekötőkről
- Navigálás az összekötőkhöz a forrás/cél AbstractGraphicalEditPartban
- getModel(Source|Target)Connections()

```
protected List getModelSourceConnections() {
  return ((Person)getModel()).getSourceAcquaintances();
}
protected List getModelTargetConnections() {
  return ((Person)getModel()).getTargetAcquaintances();
}
```

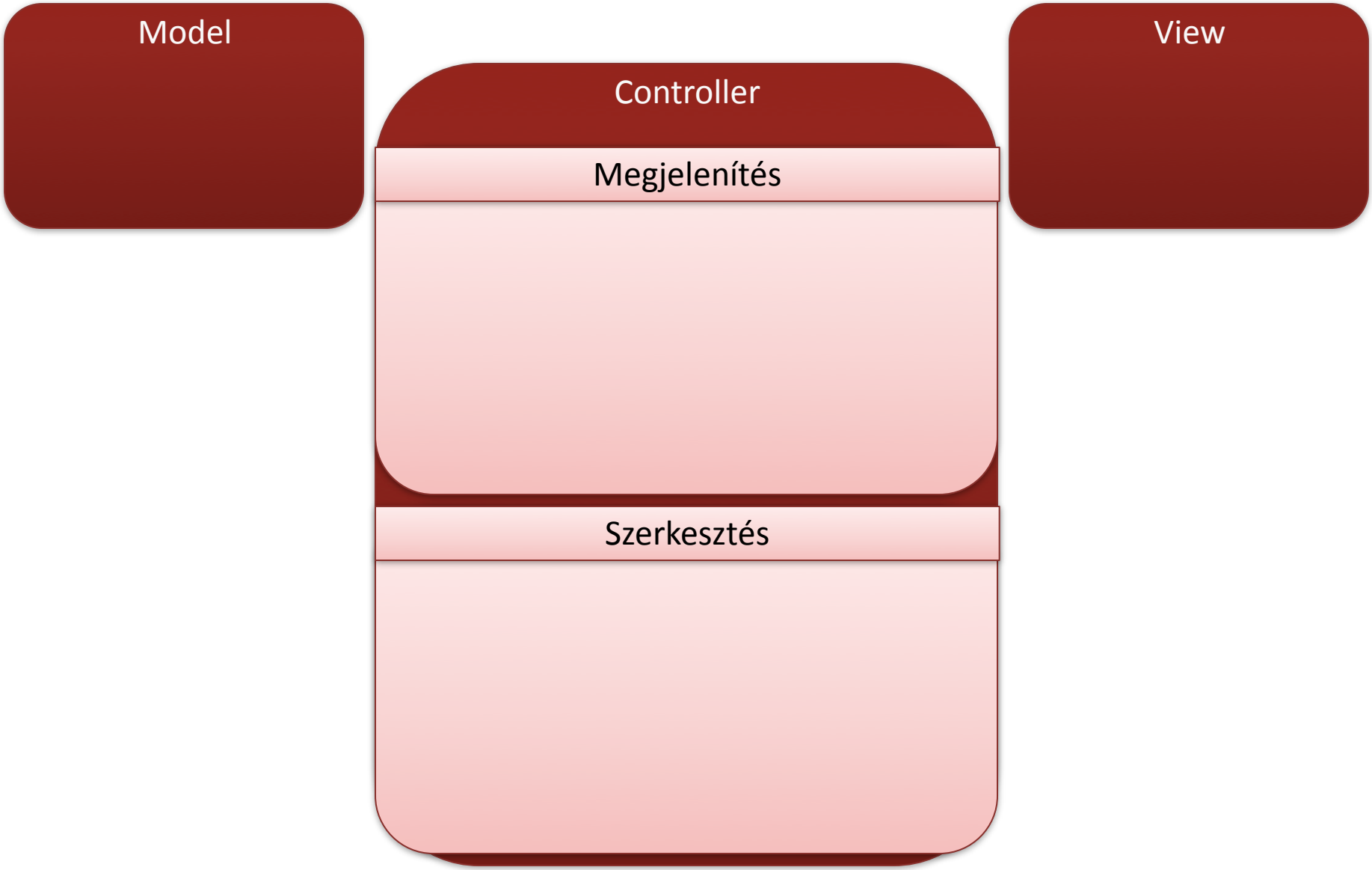
Kapcsolat a nézettel - NodeEditPart

- ConnectionAnchorok visszaadása
- Forrás/cél EditPartok biztosítják
  - NodeEditPart interfészben deklarált metódusok
    - get(Source|Target)ConnectionAnchor()

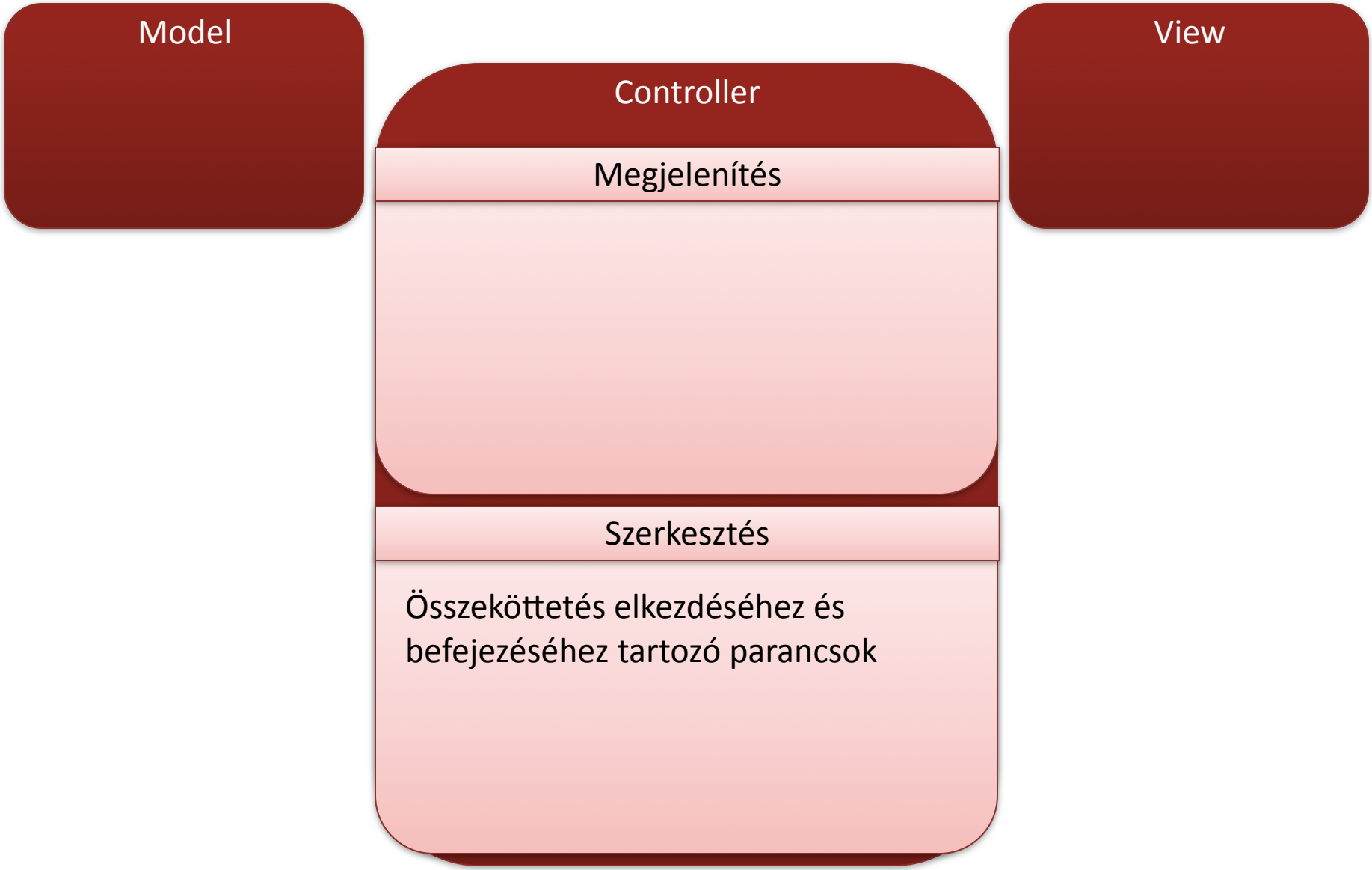
Összekötő EditPart

- Ősosztály: AbstractConnectionEditPart
  - Mindent tud, amit a többi EditPart
- Nyíl nézet létrehozása: createFigure()
  - Mindenképpen egy PolylineConnection kell
- Forrás és cél EditPartok lekérdezhetők (opcionális)
  - getSource() és getTarget() függvények
  - Létrehozás közben nem feltétlenül elérhető!
- Szerepelnie kell az EditPartFactoryban is

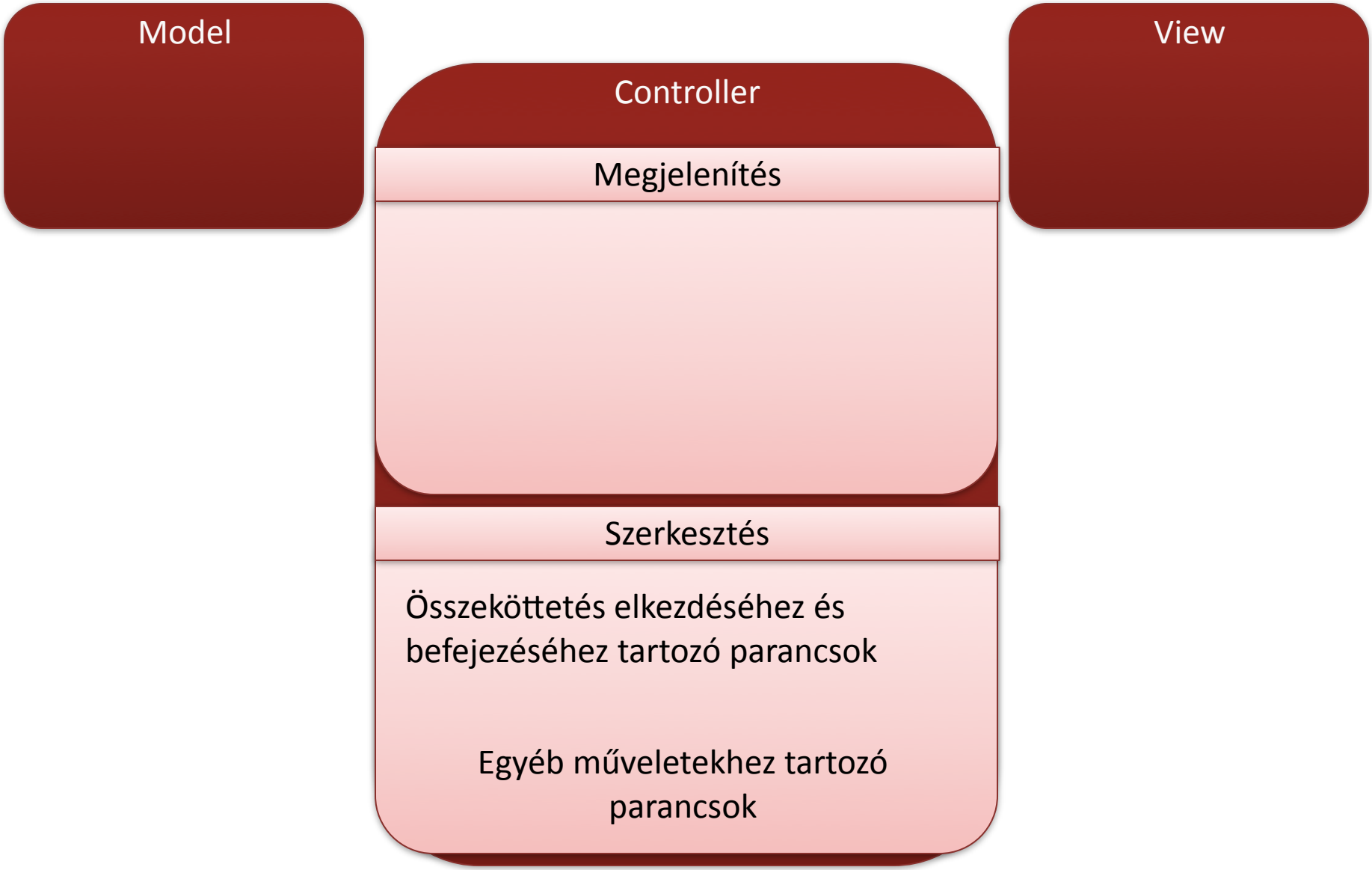
GEF workflow



GEF workflow



GEF workflow



EditPolicy: Összekötő létrehozása

- GraphicalNodeEditPolicy
  - Nem az összekötő EditParthoz, hanem a forrás/cél elemek EditPartjához tartozik!
  - Role: GRAPHICAL_NODE_ROLE
- Két lépcsős létrehozás
  - 1. getConnectionCreateCommand()
    - Request -> köztes Command (ez nem hajtódik végre)
  - 2. getConnectionCompleteCommand()
    - Request + köztes Command -> végső Command
      - Csak ez hajtódik végre!

Nyíl EditPolicy-k

- Speciális nyíl EditPolicy-k
  - ConnectionEndpointEditPolicy: kijelölés
    - Role: CONNECTION_ENDPOINTS_ROLE
  - ConnectionEditPolicy: törlés
    - Role: CONNECTION_ROLE
  - BendpointEditPolicy: töréspontok módosítása
    - Role: CONNECTION_BENDPOINTS_ROLE
    - Csak akkor, ha BendpointConnectionRouter van

Mit kell nekünk megírni?

- Összekötő modell kód, értesítéssel
  - Összekötő végpontok modelljében el kell tudni érni az onnan induló/oda érkező összekötőket!
- Opcionális: Összekötő nézet osztályok
- Forrás/cél EditPart osztályok
  - Kapcsolódó kimenő/bejövő összekötők modellbeli reprezentációja
    - getModel(Source|Target)Connections()
  - Végpontok
    - get(Source|Target)ConnectionAnchor()
- Opcionális: Összekötő EditPart osztályok
  - Forrás/cél visszaadása
    - getSource(), getTarget()

Mit kell nekünk megírni?

- Összekötő módosító Commandok
- Összekötő létrehozó EditPolicy
  - Forrás/cél EditParthoz tartozik
  - GraphicalNodeEditPolicy
  - Két lépcsős
- Egyéb EditPolicy-k
  - Törlés, töréspontok módosítása stb.

GEF workflow

Model

Forrás, cél elérése

Controller

Megjelenítés

Kapcsolódó modell összekötők

Végpontok

(Forrás és cél megadása)

Szerkesztés

Összeköttetés elkezdéséhez és befejezéséhez tartozó parancsok

Egyéb műveletekhez tartozó parancsok

View

Kirajzolás

Elrendezés

GEF workflow

Model

Forrás, cél elérése

PolylineConnection
PolylineDecoration

View

Kirajzolás

Elrendezés

Megjelenítés

Kapcsolódó modell összekötők

Végpontok

(Forrás és cél megadása)

Szerkesztés

Összeköttetés elkezdéséhez és befejezéséhez tartozó parancsok

Egyéb műveletekhez tartozó parancsok

GEF workflow

Model

Forrás, cél elérése

Controller

Megjelenítés

Kapcsolódó modell összekötők

Végpontok

(Forrás és cél megadása)

Szerkesztés

Összeköttetés elkezdéséhez és befejezéséhez tartozó parancsok

Egyéb műveletekhez tartozó parancsok

View

Kirajzolás

Elrendezés

ConnectionAnchor
ConnectionRouter

GEF workflow

Model

Forrás, cél elérése

View

Kirajzolás

Elrendezés

GraphicalEditPart

Controller

Megjelenítés

Kapcsolódó modell összekötők

Végpontok

(Forrás és cél megadása)

Szerkesztés

Összeköttetés elkezdéséhez és befejezéséhez tartozó parancsok

Egyéb műveletekhez tartozó parancsok

GEF workflow

Model

Forrás, cél elérése

Controller

Megjelenítés

Kapcsolódó modell összekötők

Végpontok

(Forrás és cél megadása)

Szerkesztés

Összeköttetés elkezdéséhez és befejezéséhez tartozó parancsok

Egyéb műveletekhez tartozó parancsok

View

Kirajzolás

Elrendezés

NodeEditPart

GEF workflow

Model

Forrás, cél elérése

Controller

Megjelenítés

Kapcsolódó modell összekötők

Végpontok

(Forrás és cél megadása)

Szerkesztés

Összeköttetés elkezdéséhez és befejezéséhez tartozó parancsok

Egyéb műveletekhez tartozó parancsok

View

Kirajzolás

Elrendezés

ConnectionEditPart

GEF workflow

Model

Forrás, cél elérése

Controller

Megjelenítés

Kapcsolódó modell összekötők

Végpontok

(Forrás és cél megadása)

Szerkesztés

Összeköttetés elkezdéséhez és befejezéséhez tartozó parancsok

Egyéb műveletekhez tartozó parancsok

View

Kirajzolás

Elrendezés

GraphicalNode EditPolicy

GEF workflow

Model

Forrás, cél elérése

Controller

Megjelenítés

Kapcsolódó modell összekötők

Végpontok

(Forrás és cél megadása)

Szerkesztés

Összeköttetés elkezdéséhez és befejezéséhez tartozó parancsok

Egyéb műveletekhez tartozó parancsok

Connection EditPolicy

View

Kirajzolás

Elrendezés

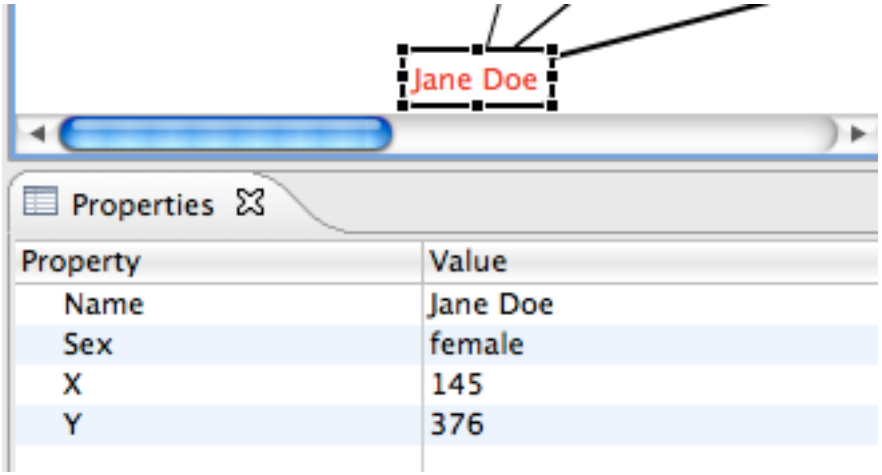
Haladó GEF fejlesztés

További lehetőségek

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

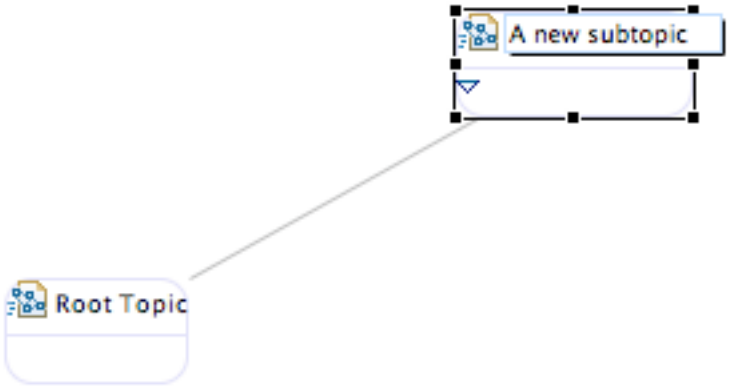
Tulajdonságok

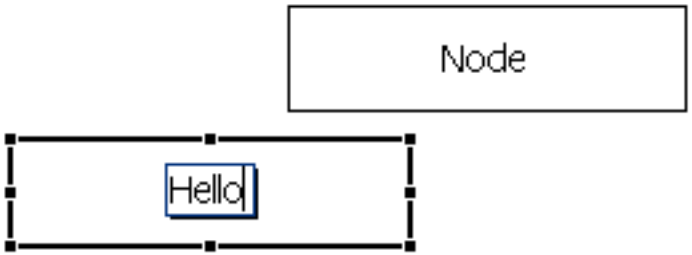
- Kijelölt edit parthoz tartozó modellelem tulajdonságainak szerkesztése az Eclipse Properties nézetében



Direct editing

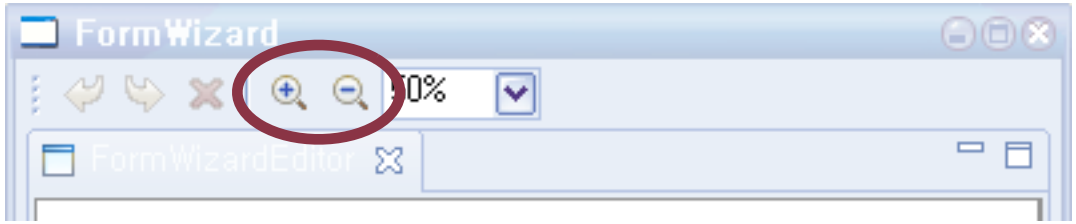
- Cél: gyors szerkesztés közvetlenül a rajzterületen





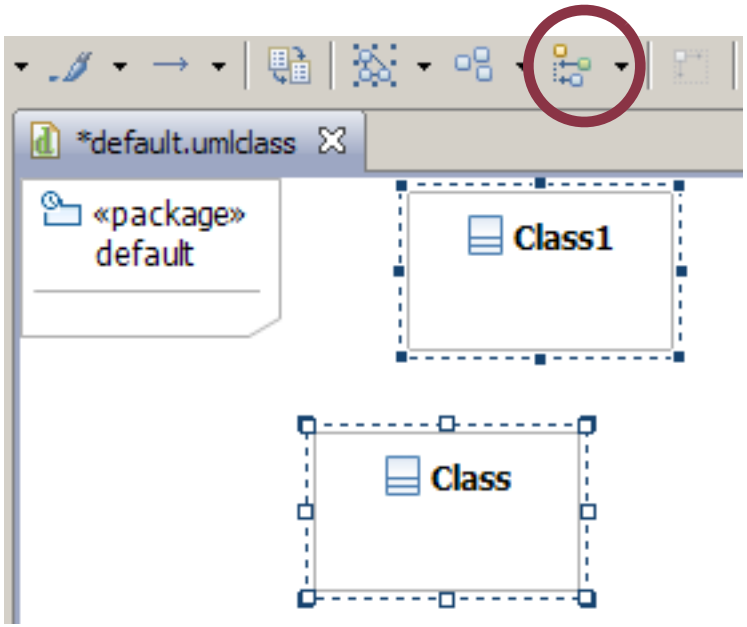
Grafikus szerkesztő nagyítása

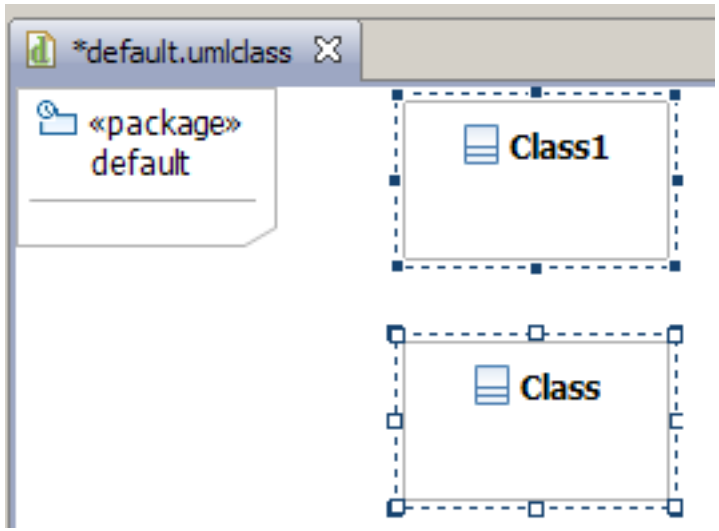
- Cél: diagram vektoros nagyítása (kicsinyítése)



Igazítás

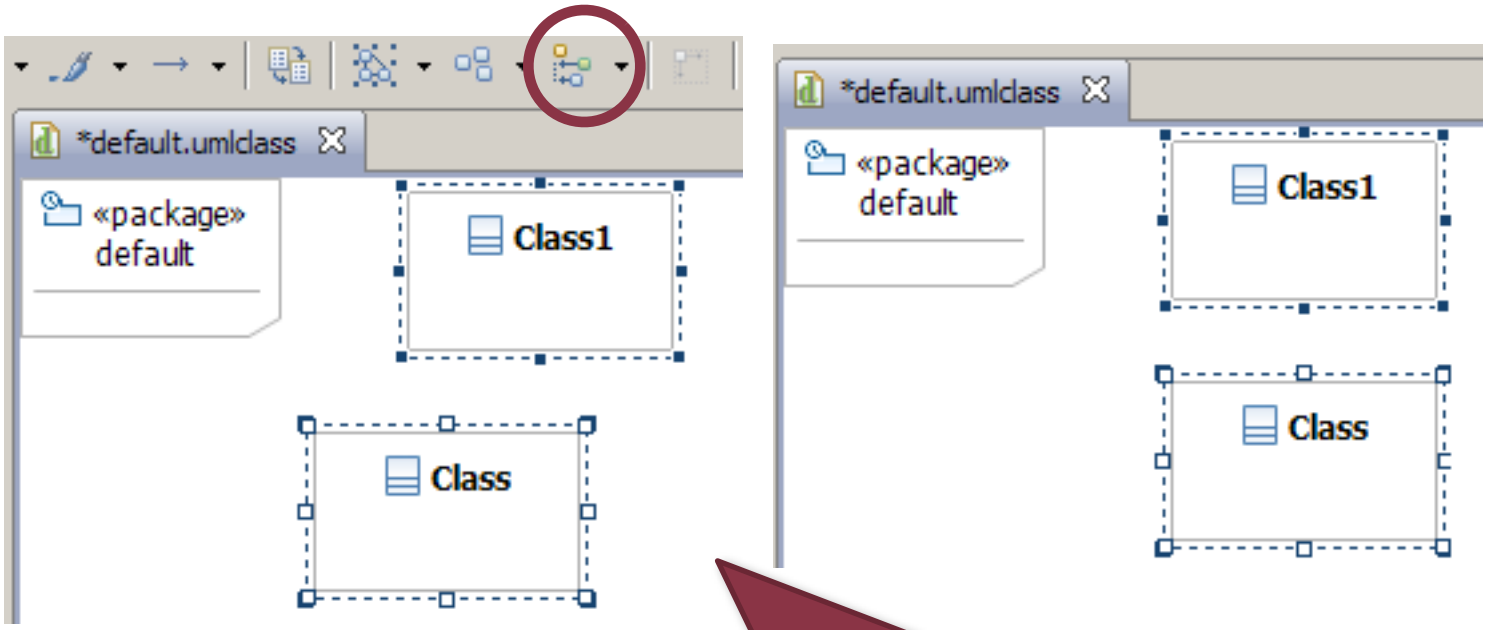
- Cél: alakzatok egymáshoz igazítása





Igazítás

- Cél: alakzatok egymáshoz igazítása

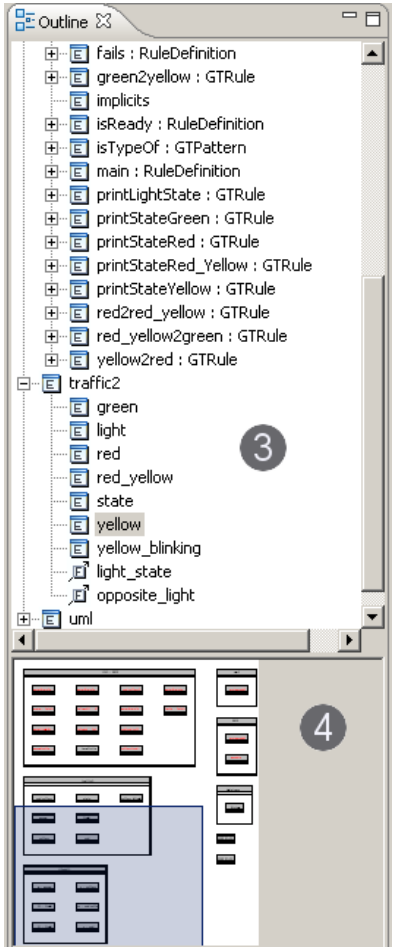


Használat:

1. Alakzatok kijelölés
2. Igazító gomb megnyomása

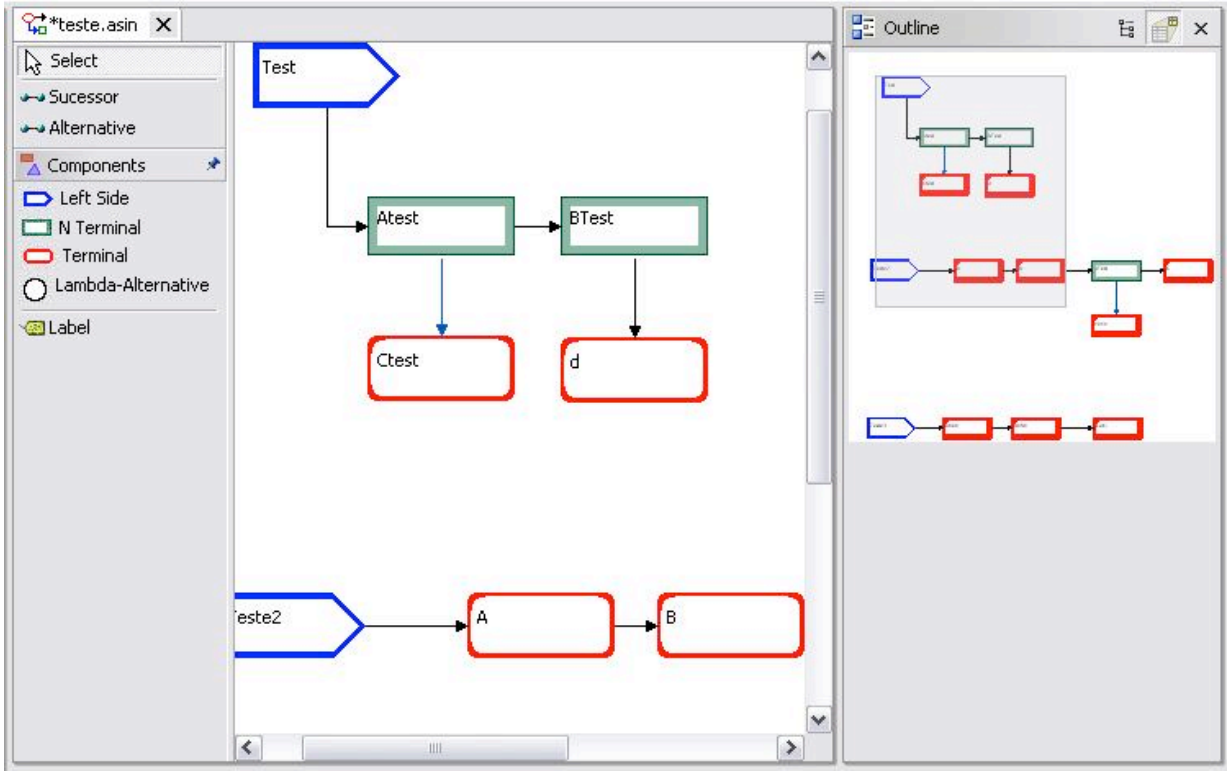
Fa modellnézet, kicsinyített vázlat

- GEF tartalmaz egy saját TreeViewer implementációt
  - TreeEditPart-okat jelenít meg
  - Fa nézetű szerkesztőhöz is felhasználható
  - Kiválasztás szinkronizálható a grafikus szerkesztőhöz
- Outline nézet alja: scrollozható thumbnail vázlat



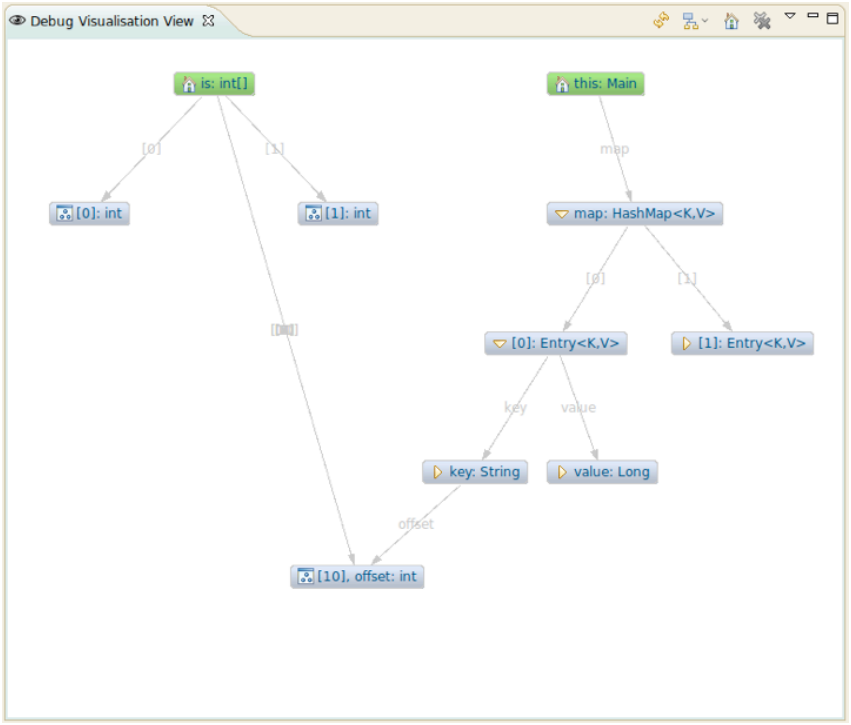
Fa modellnézet, kicsinyített vázlat

- A kész Outline nézet
  - Zoom-ot követi
  - “Lusta” renderelés



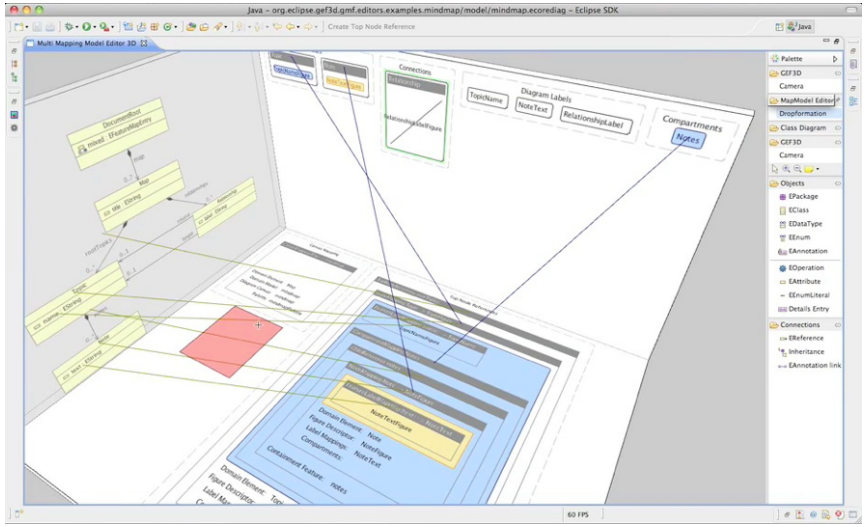
Kitekintés

- Zest
  - gráfvizualizáció
- GEF3D
  - a GEF térbeli kiterjesztése
- GMF
  - generikus helyett generatív megoldás a GEF és EMF összekötésére



Kitekintés

- Zest
  - gráfvizualizáció
- GEF3D
  - a GEF térbeli kiterjesztése
- GMF
  - generikus helyett generatív megoldás a GEF és EMF összekötésére



Kitekintés

- Zest
  - gráfvizualizáció
- GEF3D
  - a GEF térbeli kiterjesztése
- GMF
  - generikus helyett generatív megoldás a GEF és EMF összekötésére