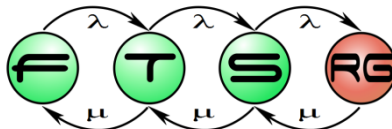


Grafikus szerkesztők

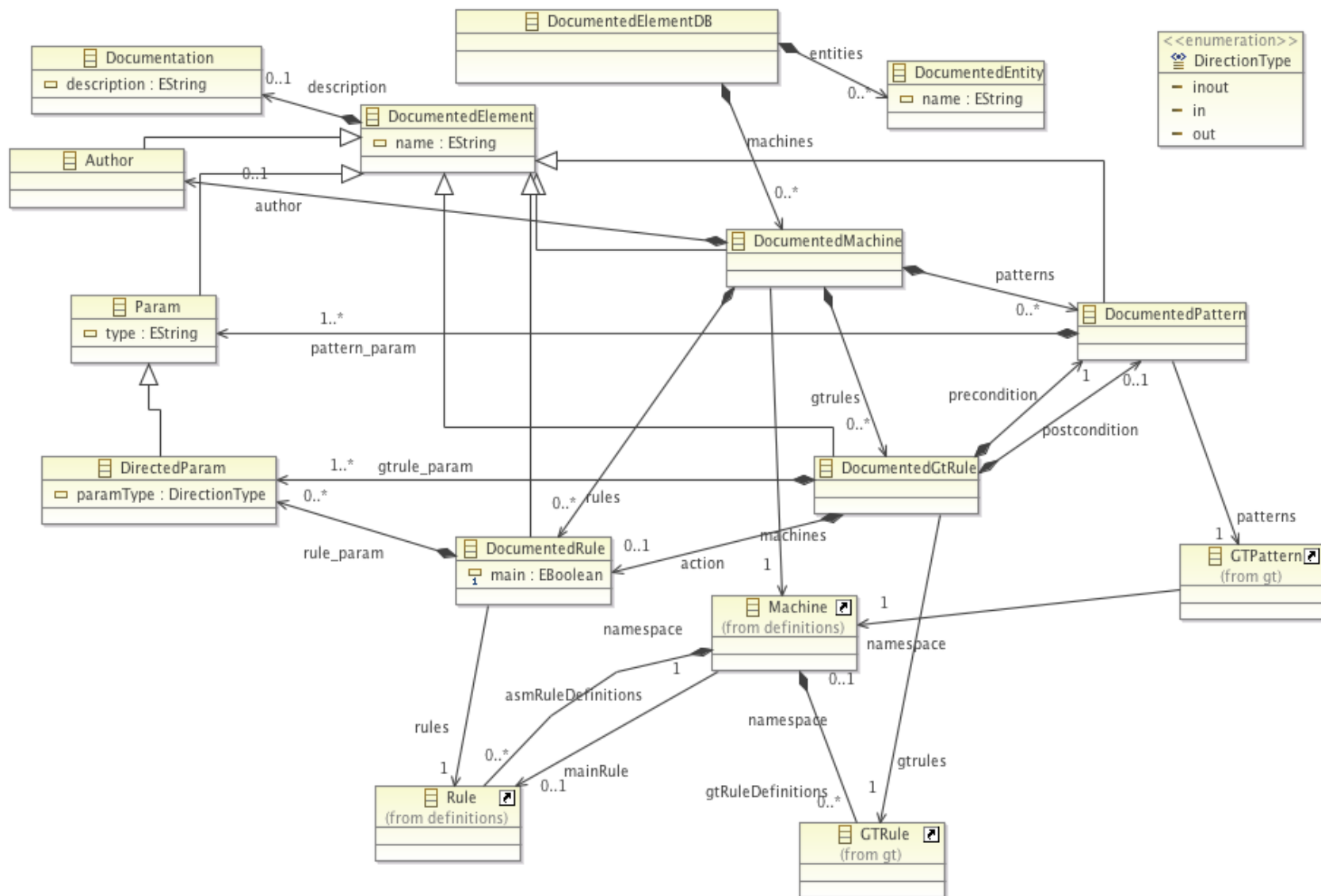
Kiegészítések
Layouting, 3D

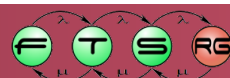


Grafikus szerkesztők

- Modellek megjelenítése
 - (Gráf szerkezet)
 - Tipikusan
 - Lassú szerkesztés
 - Jó áttekinthetőség

Átlátható?





Gráfok elrendezése

Gráf rajzolás

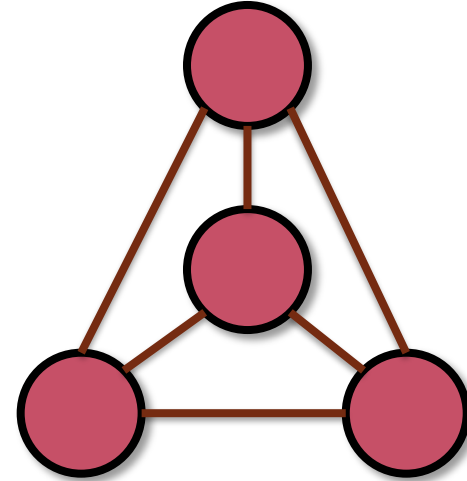
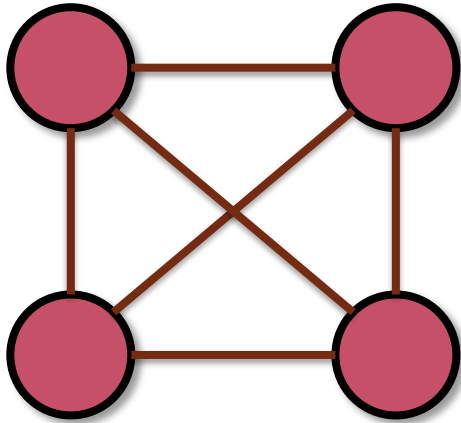
- Matematika

- $G=(V,E)$
- Kiolvasva: csomópontok és élek
- Nincs megjelenítési információ

- Gráf rajzolás

- Rendeljük megjelenítési információt a gráf csomópontokhoz!

Hogyan rajzoljunk?



- Nincs egységes követelményrendszer
 - Mit akarunk hangsúlyozni?

Gráfrajzoló eszközök

- GraphViz
 - Nem Eclipse-es
 - <http://graphviz.org>
- Zest
 - Akadémiai fejlesztés
 - GEF alprojekt
 - <http://wiki.eclipse.org/index.php/Zest>
- KIELER
 - Akadémiai fejlesztés
 - Eclipse alapú
 - <http://rtsys.informatik.uni-kiel.de/trac/kieler/>

Graphviz

- Parancssoros eszköz
 - Nyílt forrású
 - Szabadon felhasználható
- dot fájlok beolvasása
- Különböző layout algoritmusok

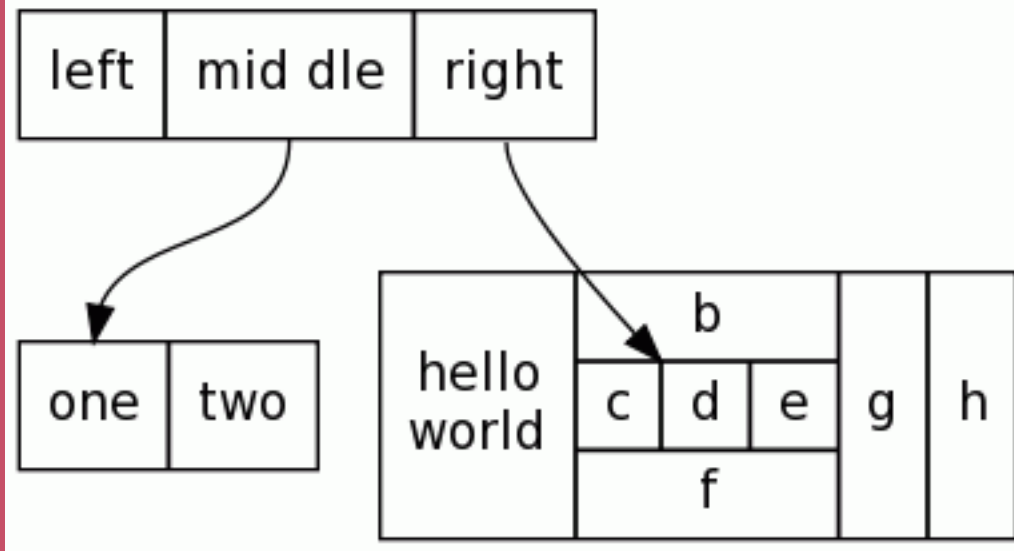
DOT példa

```
digraph structs {  
    node [shape=record];  
    struct1 [label="<f0> left|<f1> mid' dle|  
<f2> right"];  
    struct2 [label="<f0> one|<f1> two"];  
    struct3 [label="hello'nworld |{ b |{c|  
<here> d|e}| f}| g | h"];  
    struct1:f1 -> struct2:f0;  
    struct1:f2 -> struct3:here;  
}
```

DOT példa

```
digraph structs {  
    node [shape=record];
```

```
    struct1["<f2> right"]  
    struct2["<here> d"]  
    struct3["<here> d"]  
    struct1 --> struct3:here;
```



Zest: The Eclipse Visualization Toolkit

- Egyszerű gráfrajzoló könyvtár
 - SWT widget
 - Adat megadása
 - Layout algoritmus megadása

Zest: SWT jellegű felhasználás

- Graph widget
 - addNode
 - addConnection
 - add

Zest példa

```
Graph g = new Graph(shell, SWT.NONE);
```

```
GraphNode n = new GraphNode(g, SWT.NONE, "Paper");
```

```
GraphNode n2 = new GraphNode(g, SWT.NONE, "Rock");
```

```
GraphNode n3 = new GraphNode(g, SWT.NONE, "Scissors");
```

```
new GraphConnection(g, SWT.NONE, n, n2);
```

```
new GraphConnection(g, SWT.NONE, n2, n3);
```

```
new GraphConnection(g, SWT.NONE, n3, n);
```

```
g.setLayoutAlgorithm(new SpringLayoutAlgorithm  
    (LayoutStyles.NO_LAYOUT_NODE_RESIZING), true);
```

Zest példa

```
Graph g = new
```

```
GraphNode n1 =
```

```
GraphNode n2 =
```

```
GraphNode n3 =
```

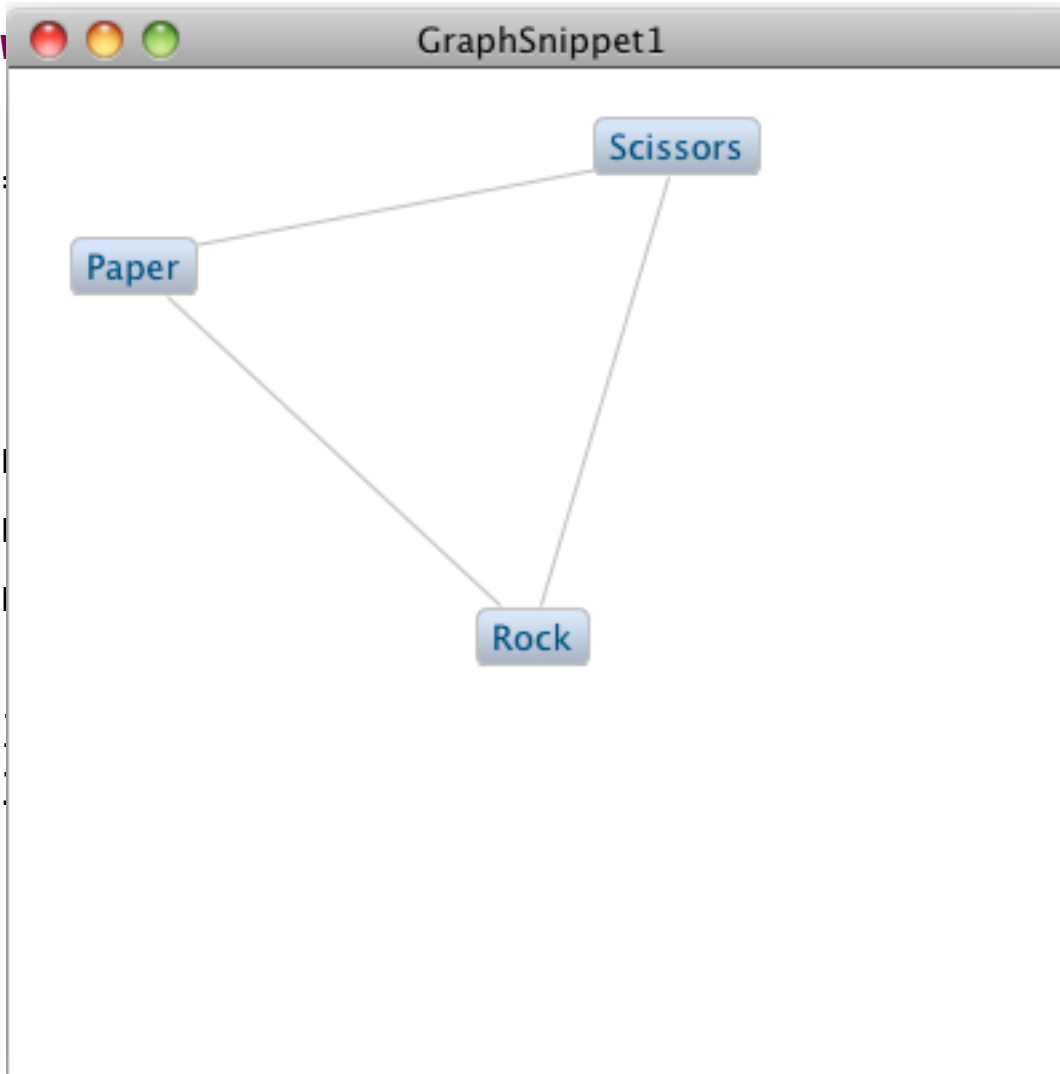
```
new GraphCon
```

```
new GraphCon
```

```
new GraphCon
```

```
g.setLayoutA
```

```
(LayoutSty
```



;

Kitérő

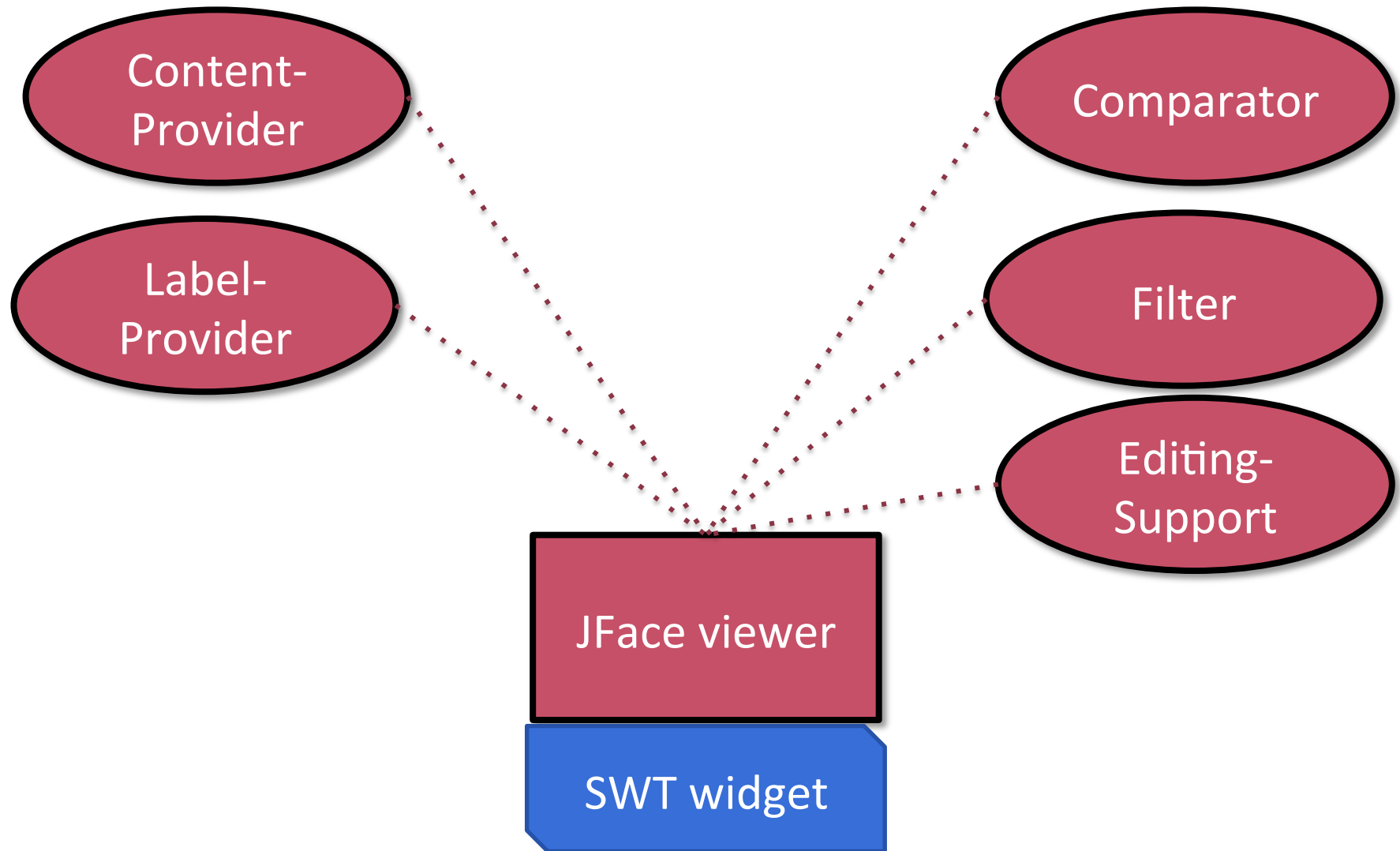
JFace Viewer framework

JFace Viewer framework

- Többféle widget egységes kezelése
 - SWT Table, Tree és List
- MVC minta
 - Modell: ContentProvider, LabelProvider
 - Nézet: Viewer
 - Vezérlő: Listeners

Fontos! Eclipse View és JFace Viewer nem ugyanaz

JFace Viewer - architektúra



Content Provider

- A megjelenítendő elemeket adja meg
- `getElements()`
 - Elemek tömbjét adja vissza
 - Nem kötelező használni
 - Az elemek a `viewer add()` metódussal is hozzáadhatóak
- `inputChanged(Viewer, Object, Object)`
 - Jelzi a providernek, hogy a root objektum megváltozott
 - Ezután a `getElements()` is hívódni fog
 - Ne hívjuk meg közvetlenül, helyette `viewer.setInput(Object)`

Példa: TreeViewer

■ ITreeContentProvider

- A megjelenítendő elemeket adja meg
- `getChildren()`
 - Adott elem gyermekeit listázza
- `hasChildren()`
 - Vannak-e egy csomópontnak gyerekei
 - Ha lassú kiszámolni, legyen igaz
- `getParent()`
 - Szülő visszaadása

Label Provider

- Elemekhez felirat/kép rendelése
 - `getText()`
 - `getImage()`
 - `isLabelProperty()`
 - Érinti-e a feliratot a tulajdonság megváltozása?
 - Alapértelmezett megvalósítás
 - Az elemek `toString()` metódusát használja
 - Képet nem ad vissza

Comparator

- Rendezés definiálása
- Modellelemek összehasonlítása
 - Quick sort

Filter

- Elemi szűrők **listája**
 - Minden modellelemre lefuttatja

Direct Editing

- Szerkesztési támogatás
 - (nem minden widget támogatja)
 - Meg kell adni minden modellelemhez
 - Konverzió modell és megjelenítés között
 - Adat validáció
 - Adatmódosítás

JFace Selection API

- Lekérhető az aktuálisan kijelölt elemek listája
- `getSelection()`
 - `ISelection`
 - Általános jelzése a kijelölésnek
 - Közvetlenül nem használható
 - `IStructuredSelection`
 - Az elemek sorrendje kötött
 - Iterator-al bejárható
 - `ITreeSelection`
 - `IStructuredSelection` leszármazott
 - Hierarchia leírására

JFace API

- Magas szintű API
 - Modellek és GUI widgetek összekötése
 - Elemi funkciók megvalósítandóak
 - Sokféle szolgáltatás adott

Zest JFace API

Zest: JFace API

- GraphViewer osztály
 - JFace viewer megszokott funkcionalitás
 - Content Provider
 - Label Provider
 - Selection API
 - Filter, Ordering
 - ...

Content Provider

- Adat feltöltéshez három forrás támogatott
 - IGraphContentProvider
 - `getElements()`: élek listája
 - `getSource(rel)`, `getTarget(rel)`: kezdő- és végpont
 - IGraphEntityContentProvider
 - `getElements()`: csomópontok listája
 - `getConnectedTo(entity)`: szomszéd csomópontok
 - IGraphEntityRelationshipContentProvider
 - `getElements()`: csomópontok listája
 - `getRelationships(entity, entity)`: élek két csomópont között

Content Provider példa

```
class MyContentProvider implements IGraphEntityContentProvider {
    public Object[] getElements(Object inputElement) {
        return new String[] { "First", "Second", "Third" };
    }
    public Object[] getConnectedTo(Object entity) {
        if (entity.equals("First")) { return new Object[] { "Second" }; }
        if (entity.equals("Second")) { return new Object[] { "Third" }; }
        if (entity.equals("Third")) { return new Object[] { "First" }; }
        return null;
    }
    public double getWeight(Object entity1, Object entity2) {
        return 0;
    }
    public void dispose() {}
    public void inputChanged(Viewer viewer, Object oldInput,
        Object newInput) {}
}
```

Content Provider példa

```
class MyContentProvider implements IGraphEntityContentProvider {  
    public Object[] getElements(Object inputElement) {  
        return new String[] { "First", "Second", "Third" };  
    }  
    public Object[] getConnectedTo(Object entity) {  
        if (entity.equals("First")) { return new Object[] { "Second" }; }  
        if (entity.equals("Second")) { return new Object[] { "Third" }; }  
        if (entity.equals("Third")) { return new Object[] { "First" }; }  
        return null;  
    }  
    public double getWeight(Object entity1, Object entity2) {  
        return 0;  
    }  
    public void dispose() {}  
    public void inputChanged(Viewer viewer, Object oldInput,  
        Object newInput) {}  
}
```

Csomópont
objektumok

Content Provider példa

```
class MyContentProvider implements IGraphEntityContentProvider {  
    public Object[] getElements(Object inputElement) {  
        return new String[] { "First", "Second", "Third" };  
    }  
    public Object[] getConnectedTo(Object entity) {  
        if (entity.equals("First")) { return new Object[] { "Second" }; }  
        if (entity.equals("Second")) { return new Object[] { "Third" }; }  
        if (entity.equals("Third")) { return new Object[] { "First" }; }  
        return null;  
    }  
    public double getWeight(Object entity1, Object entity2) {  
        return 0;  
    }  
    public void dispose() {}  
    public void inputChanged(Viewer viewer, Object oldInput,  
        Object newInput) {}  
}
```

Szomszéd
objektumok

Content Provider példa

```
class MyContentProvider implements IGraphEntityContentProvider {  
    public Object[] getElements(Object inputElement) {  
        return new String[] { "First", "Second", "Third" };  
    }  
    public Object[] getConnectedTo(Object entity) {  
        if (entity.equals("First")) { return new Object[] { "Second" }; }  
        if (entity.equals("Second")) { return new Object[] { "Third" }; }  
        if (entity.equals("Third")) { return new Object[] { "First" }; }  
        return null;  
    }  
    public double getWeight(Object entity1, Object entity2) {  
        return 0;  
    }  
    public void dispose() {}  
    public void inputChanged(Viewer viewer, Object oldInput,  
        Object newInput) {}  
}
```

Élsúlyok

Content Provider példa

```
class MyContentProvider implements IGraphEntityContentProvider {  
    public Object[] getElements(Object inputElement) {  
        return new String[] { "First", "Second", "Third" };  
    }  
    public Object[] getConnectedTo(Object entity) {  
        if (entity.equals("First")) { return new Object[] { "Second" }; }  
        if (entity.equals("Second")) { return new Object[] { "Third" }; }  
        if (entity.equals("Third")) { return new Object[] { "First" }; }  
        return null;  
    }  
    public double getWeight(Object entity1, Object entity2) {  
        return 0;  
    }  
    public void dispose() {}  
    public void inputChanged(Viewer viewer, Object oldInput,  
        Object newInput) {}  
}
```

Inputkezelés

Label Provider

- Egyszerű JFace Label provider
 - Objektumokhoz szöveg/kép rendelés
- Kiegészítő lehetőségek
 - IEntityStyleProvider
 - IConnectionStyleProvider
 - IFigureProvider

Label Provider példa

```
class MyLabelProvider extends LabelProvider {  
    final Image image =  
Display.getDefault().getSystemImage(SWT.ICON_WARNING);  
  
    public Image getImage(Object element) {  
        if (element instanceof String) { return image; }  
        return null;  
    }  
  
    public String getText(Object element) {  
        if (element instanceof String) {  
            return element.toString();  
        }  
        return null;  
    }  
}
```

Zest: JFace API példa

```
viewer = new GraphViewer(shell, SWT.NONE);

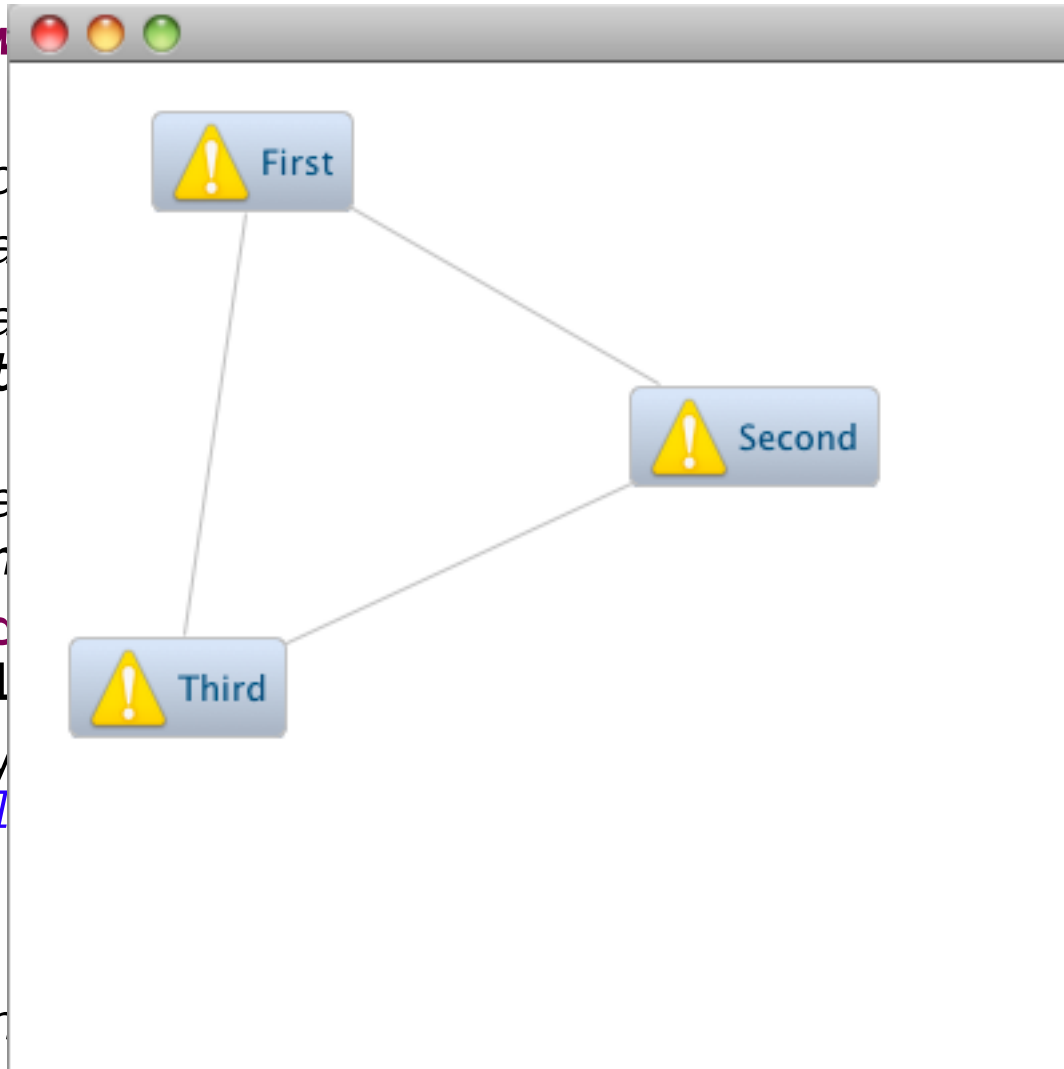
viewer.setContentProvider(new MyContentProvider());
viewer.setLabelProvider(new MyLabelProvider());
viewer.setLayoutAlgorithm(new
SpringLayoutAlgorithm(LayoutStyles.NO_LAYOUT_NODE_RESIZI
NG));

viewer.addSelectionChangedListener(new
ISelectionChangedListener() {
    public void selectionChanged
        (SelectionChangedEvent event) {
        System.out.println
        ("Selection changed: " + (event.getSelection()));
    }
});

viewer.setInput(new Object());
```

Zest: JFace API példa

```
viewer = new  
  
viewer.setCo  
viewer.setLa  
viewer.setLa  
SpringLayout  
NG));  
viewer.addSe  
ISelectionCh  
  
public  
    (Sel  
        Sy  
        ("Sel  
    }  
  
});  
viewer.setIn
```



```
er());
```

```
);
```

```
NODE_RESIZI
```

```
ection())));
```

Elrendezések

- Gráfelrendezések megadása
 - Rendező algoritmus (layout algorithm)
 - Sokféle beépítve
 - Grid
 - Radial, Tree
 - Spring
 - Bővíthető
 - Meglevő algoritmus paraméterezésével
 - Több algoritmus egymás után futtatásával
 - Teljesen saját algoritmus írásával

Zest és GEF/GMF

- Hogy kapcsolódik a Zest a GEF-hez?
 - Közös függőség: Draw2D
 - Rajzolás: IFigure
 - GEF/GMF IFigure elrendezhető Zest-tel
 - IFigure elemek csomagolása
 - Zest elrendező algoritmusok kimenete
 - Gráf helyett tetszőleges IFigure használhatóak
 - LabelProvider segítségével megadható

Változatok

- Zest 1.x
 - Korai változat
- Zest 2.0
 - Jobb layout támogatás
 - DOT integráció
 - Import, export
 - Xtext alapú editor
 - Nem binárisan kompatibilis az 1.x változatokkal
 - Egyszerű módosítások
 - Bugfixek
 - DE: Fejlesztés alatt

KIELER projekt

- Akadémiai projekt
- Automatikus elrendezés
 - GMF
 - Graphiti
 - UML
- Osztálydiagram jellegű struktúrákhoz jó

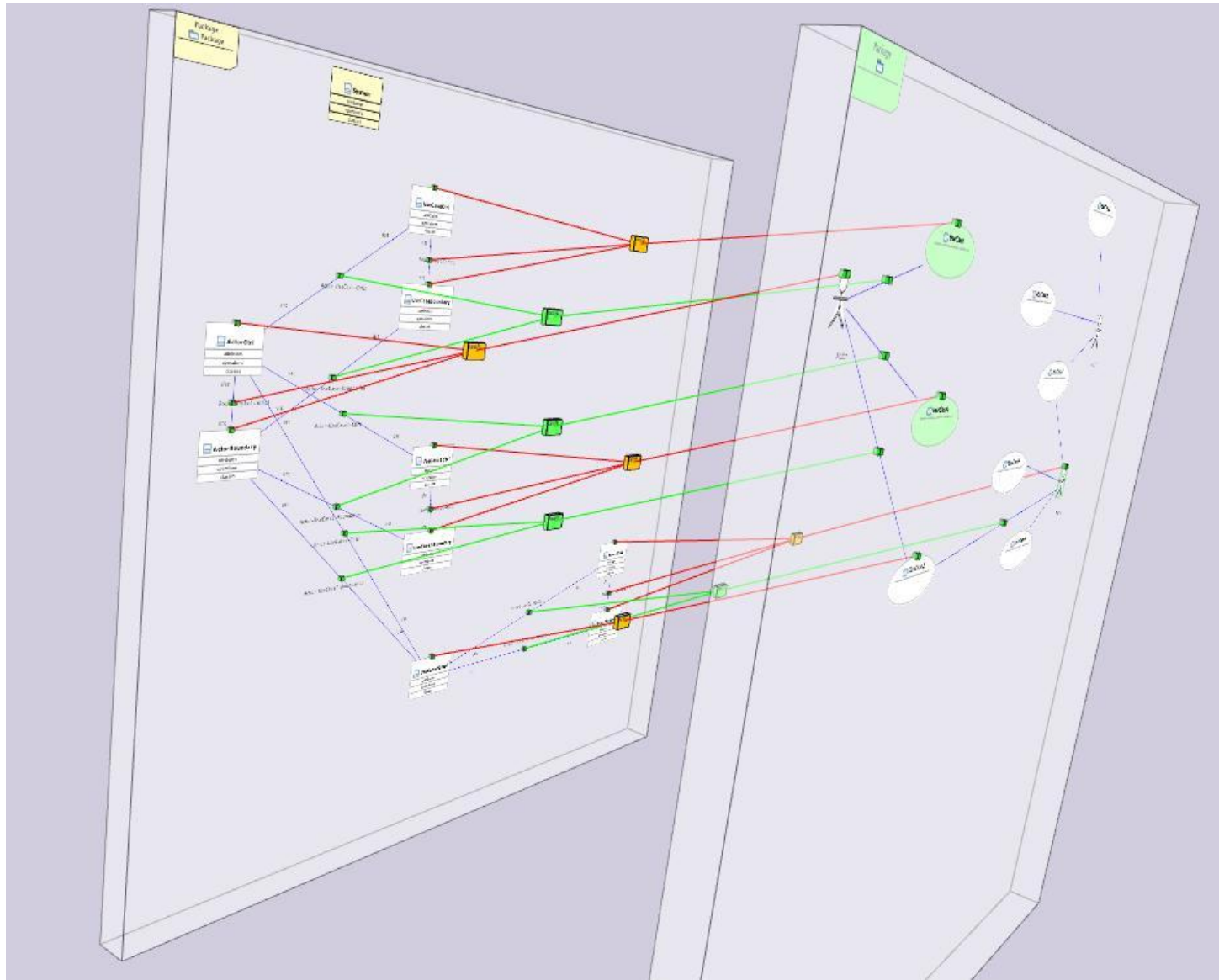
További eszközök

- IBM ILOG JViews Graph Layout for Eclipse
 - <http://www-01.ibm.com/software/integration/visualization/jviews/graph-layout-eclipse/>
- yEd Graph Editor
 - Külön alkalmazás
 - Algoritmusok felhasználhatóak Eclipse környezetben is

GEF3D

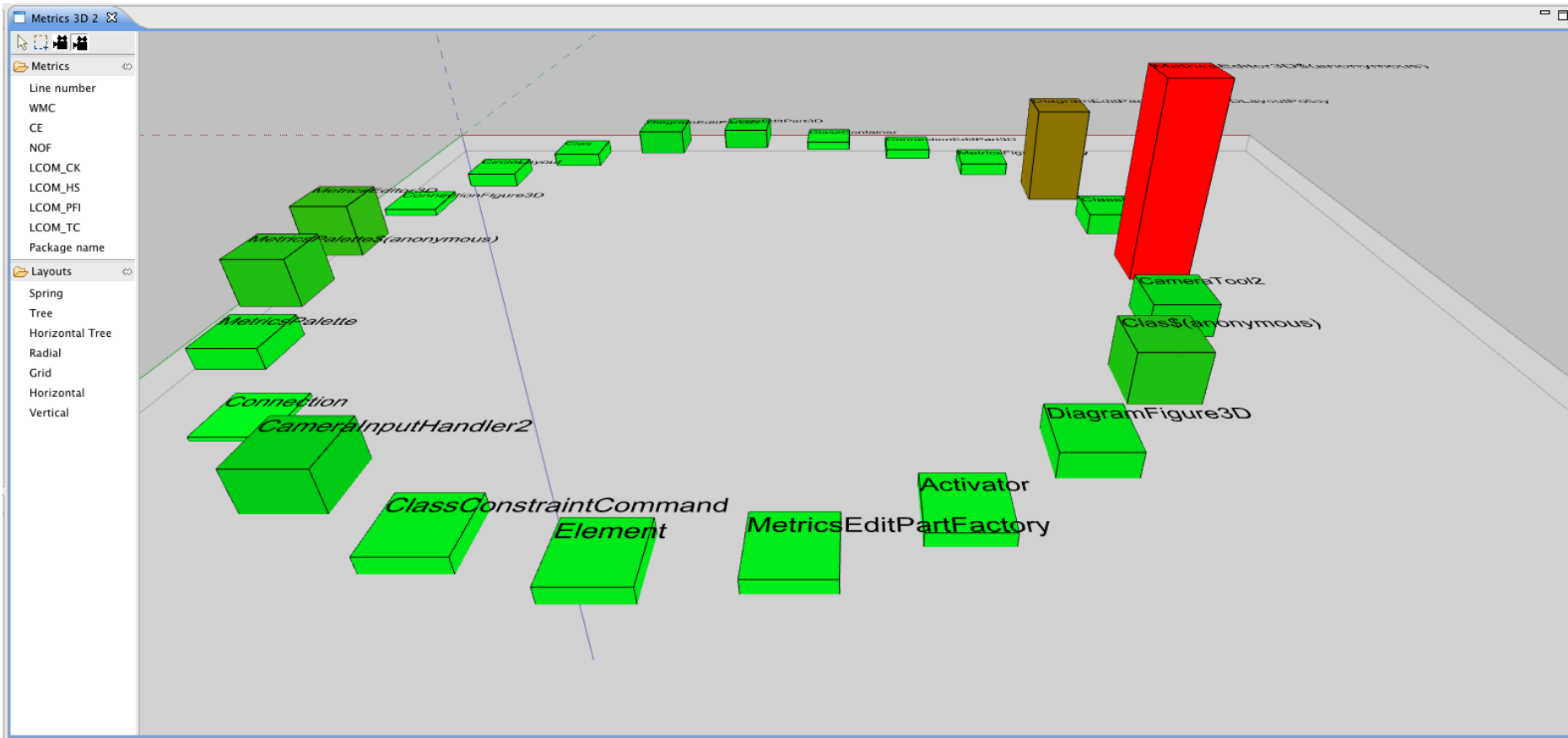
- Cél
 - Teljes, 3 dimenziós szerkesztési lehetőségek
 - Létező GEF-es editorok újrahasznosítása
 - 2,5D: meglevő editor egy sík
- GEF3D
 - Akadémiai projekt

2,5D editor – Kapcsolatok megjelenítése



Forrás: <http://jaxenter.com/gef-goes-3d-10076.html>

3D editor



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

GEF workflow

Model

Értesítés

EditPartFactory

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

Model

Értesítés

EditPartFactory

Controller**Megjelenítés**Kapcsolat a
modellelKapcsolat a
nézettel

Hierarchia megadása

Nézet frissítése a modell
változásakor**Szerkesztés**

Modellmódosító parancsok

Szerkesztési kérések -> parancsok

Szerkesztőeszközök a felhasználói
felületen**View**

Kirajzolás

Elrendezés

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

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

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

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

EditPolicy

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

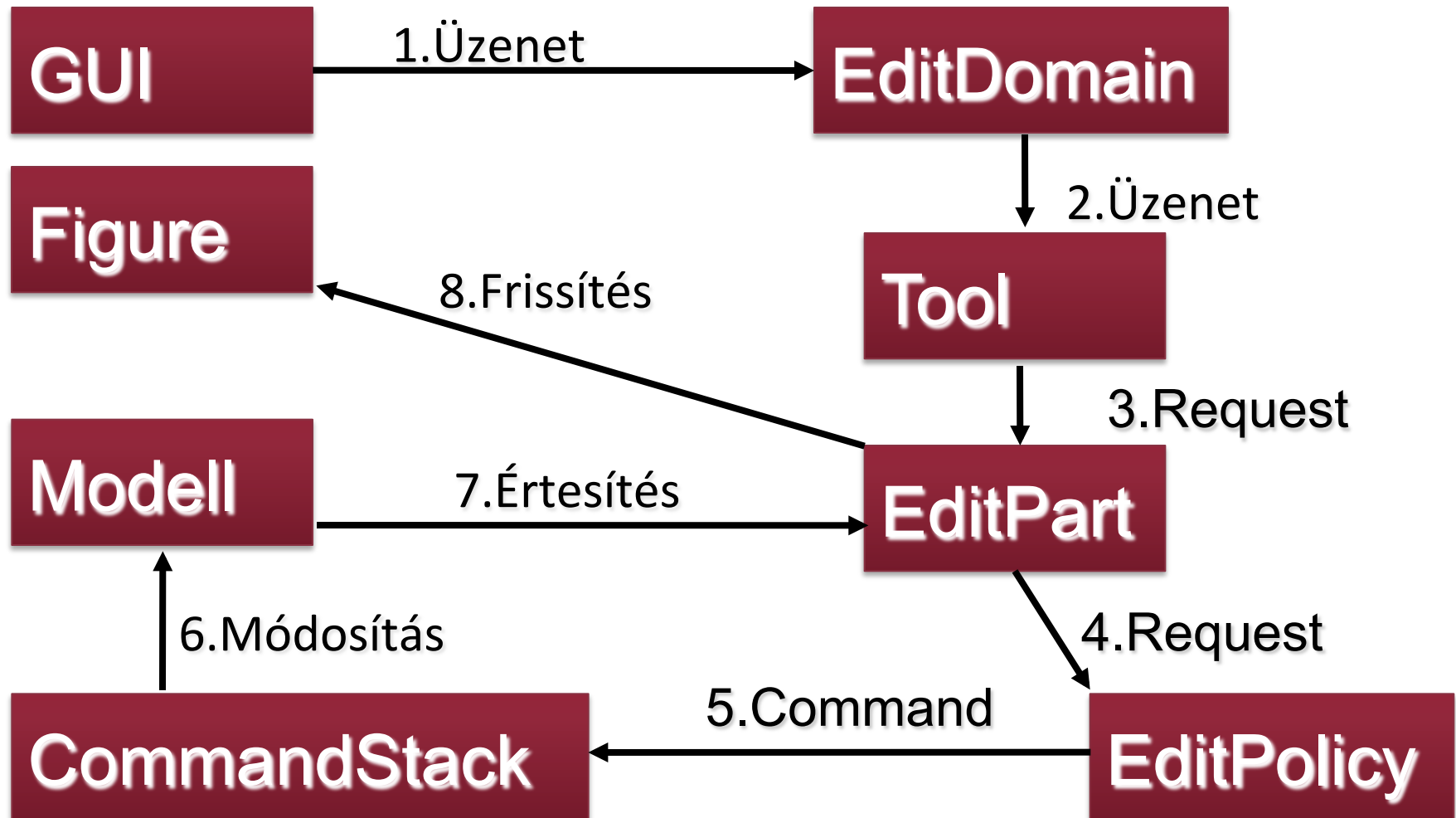
Command

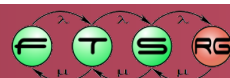
EditPart

Tool

EditPolicy

Szerkesztés folyamata





2,5D editor

- Létező GEF editor esetén kell
 - GraphEditor3D
 - 2D editor alosztálya
 - Kamera eszköz hozzáadása
 - EditPartFactory módosítás
 - GraphEditPart3D
 - 2D edit part módosítás 3D IFigure számára
 - GraphFigure3D
 - Sík kijelölése, amelyre a 2D elemek mozgathatóak

Igazi 3D editor

■ GEF örökség

○ Modell

- Nem kell változtatni*

○ Nézet

- IFigure3D interfész térbeli objektumokhoz

○ Vezérlő

- Logikailag hasonló a GEF megoldásához
- További parancsok és eszközök (pl. kamera)

Problémák

- Dokumentáció nincs
 - GEF nem egyszerű alap
- [@anderiasch](#) (Twitter)
- "Now some Code. If you're not familiar reading Code, you can spend the time figuring out what you're doing on a dev conference." [#gddde](#)

Javasolt buborékok (egyet válassz:)

Szöveg

Szöveg