Aspect-Oriented Refactoring of Graph Visualization Tool: CHISIO

M. Esat Belviranlı
Celal Çığır
Alptuğ Dilek

26 Dec. 2008
2nd Turkish AOSD Workshop

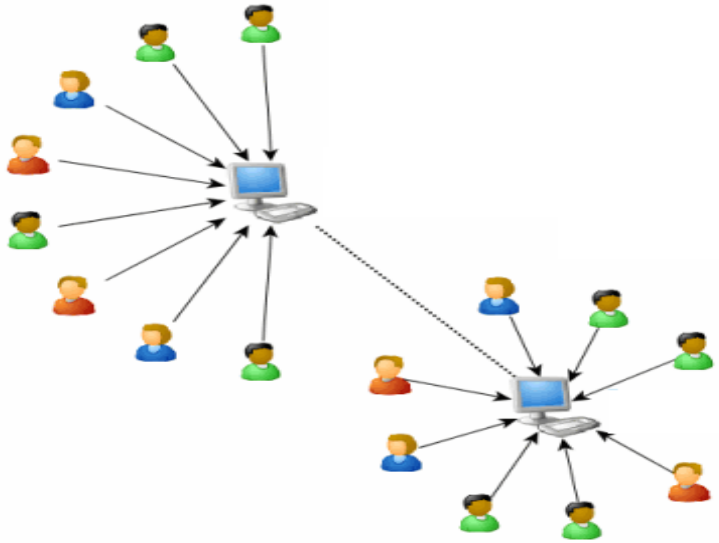
Outline

- Introduction
  - Motivation
  - Background Information
- OO Design Of CHISIO
- Problem Statement
- Aspect-Oriented Programming
- Related Work
- Conclusion

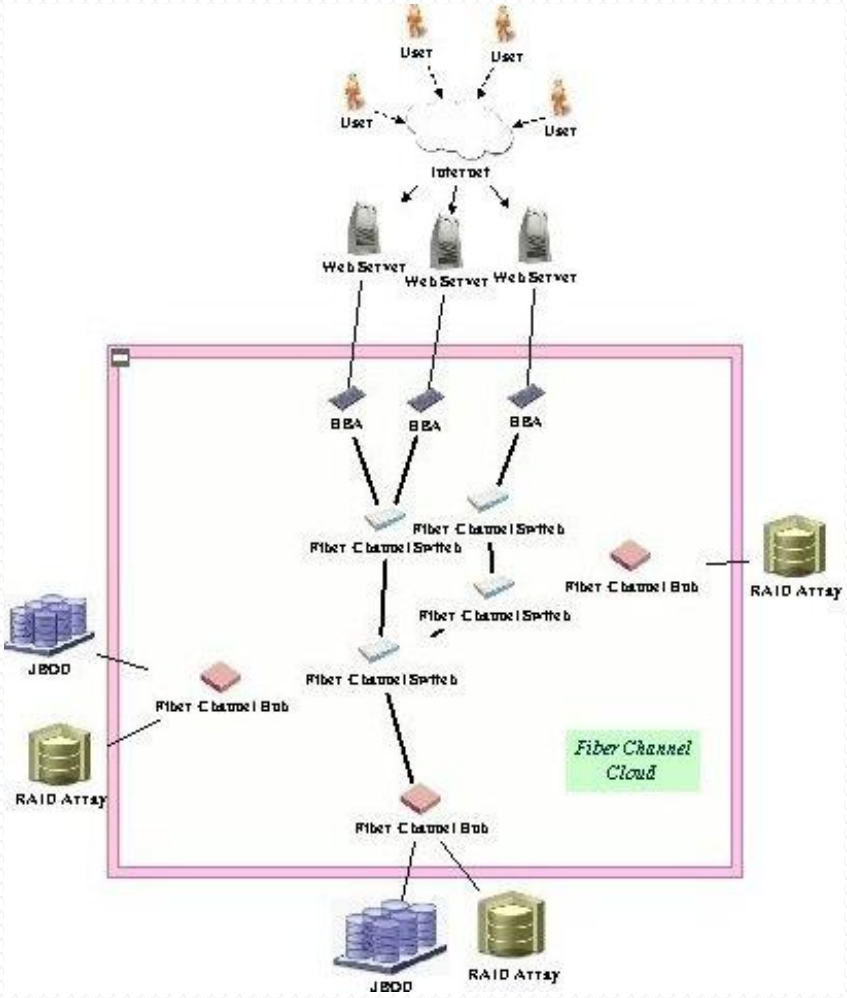
Graphs

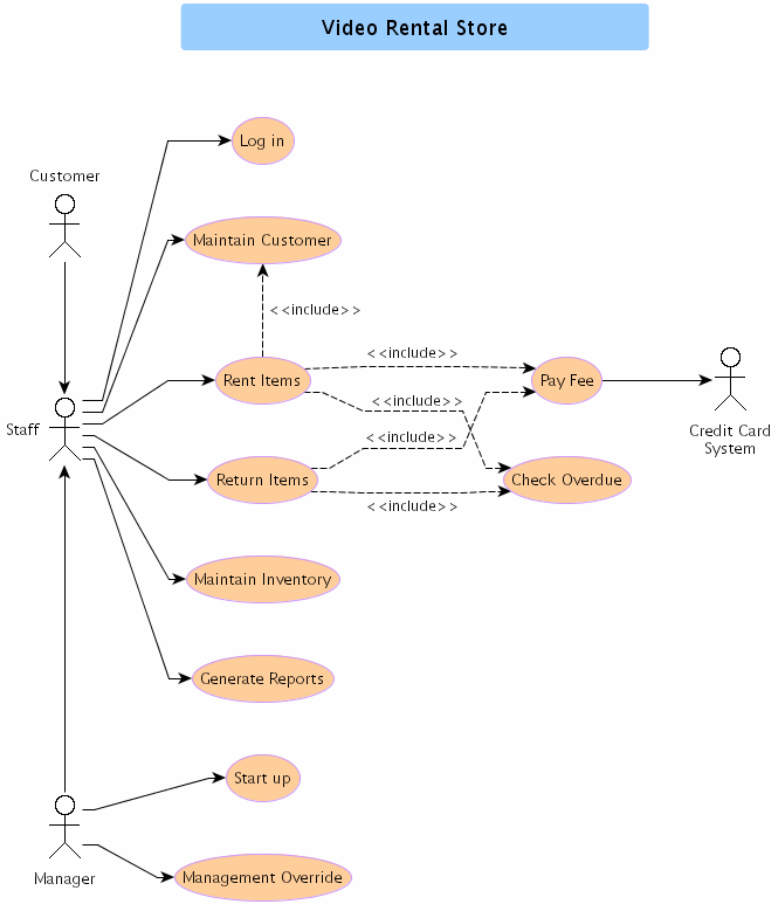
- Graph = nodes + edges
  - An edge has source and target nodes
  - A node has outgoing and incoming edges

- Graphs are excellent models for many real-world applications
  - Web, roads, telephone, internet, cpu architecture, travel, document citations



Graph Examples

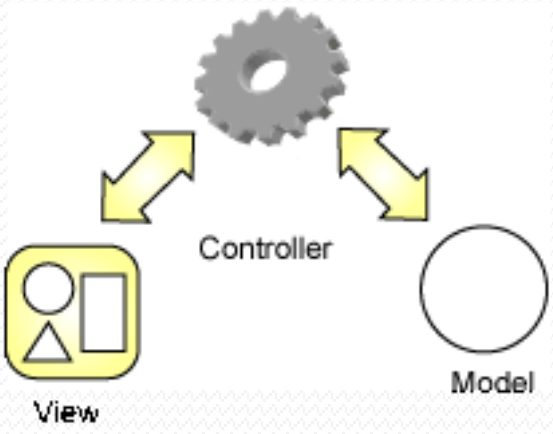




Graph Editor

Used for drawing graphs

- Creation of new graphs
- Visualization of already created graphs
- Layout of graphs
- Editing graphs
  - Changing topology
  - Changing geometry

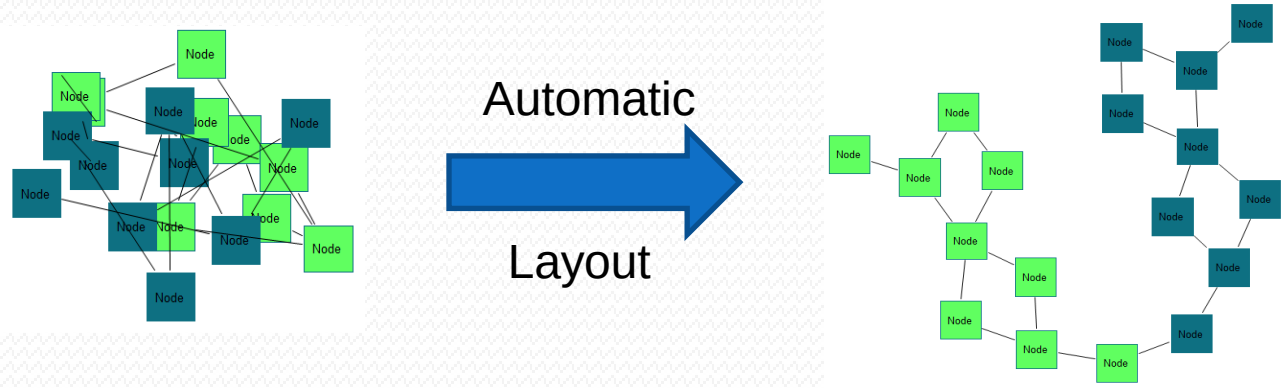


- GEF : Graphical Editing Framework
- Based on a Model-View-Controller (MVC) architecture.

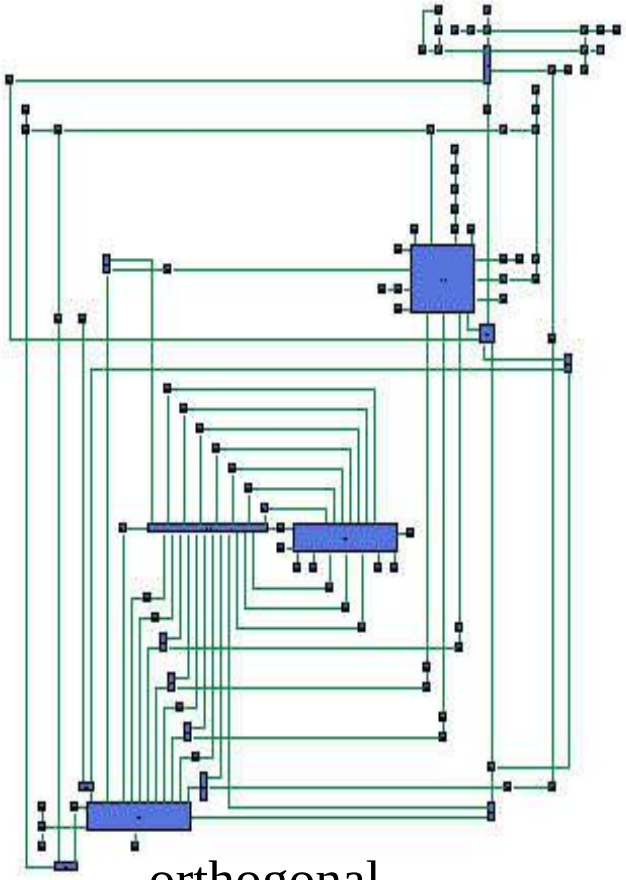
Graph Layout

- Locate nodes and route edges to draw understandable graphs

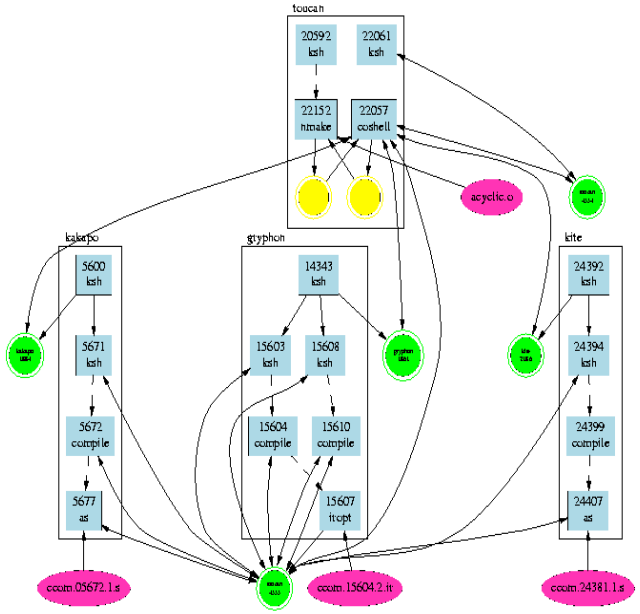
- Automatic Layout
  - Make it without human interaction



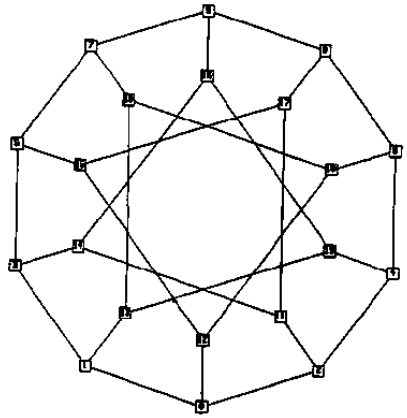
Graph Layout



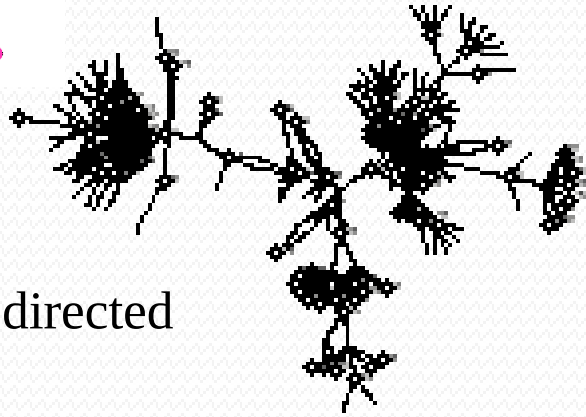
orthogonal



hierarchical



symmetric



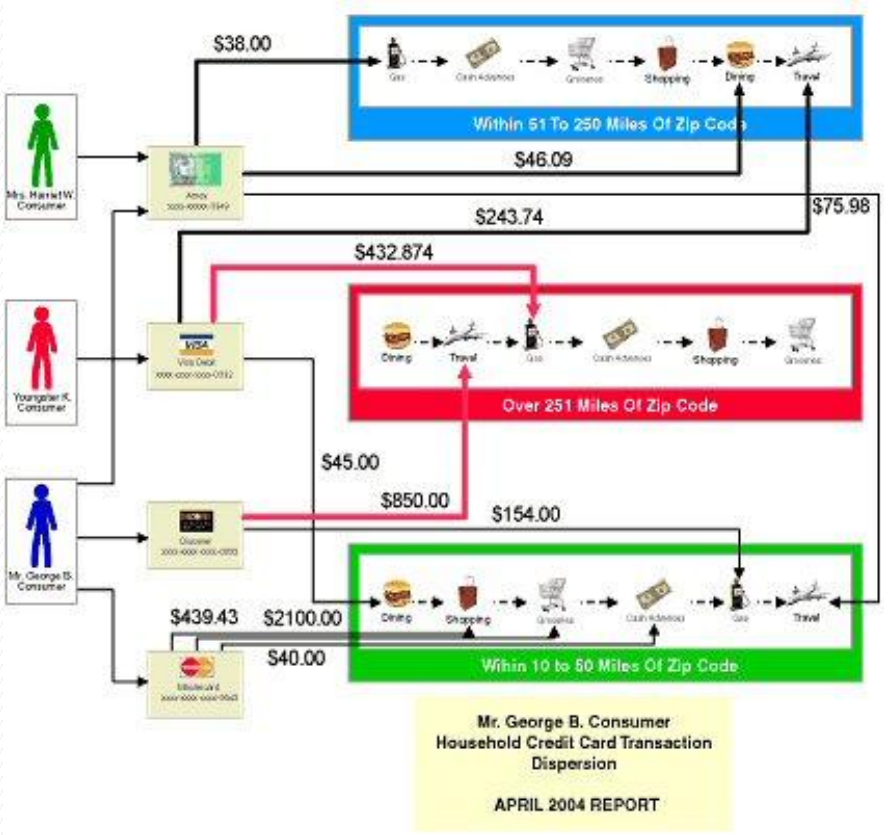
force-directed

Motivation

- There is an obvious gap in non-commercial tools with compound graph support

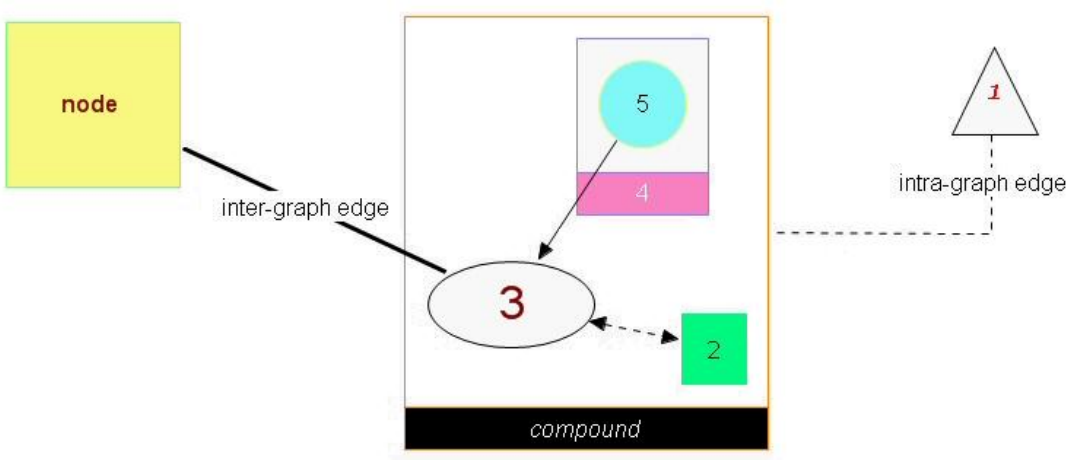
Compound Graph: A graph which can have nested subgraphs

- Design an abstract layout development environment



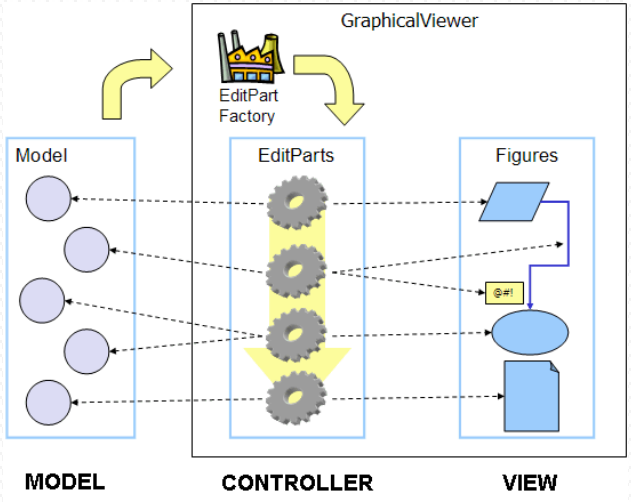
CHISIO as an Editor

- Chisio: A Visual Framework for Compound Graph Layout and Editing
  - A standalone Java application based on GEF
  - Interactive editing ability
  - Automatic layout algorithms
  - Compound support
  - Animation support



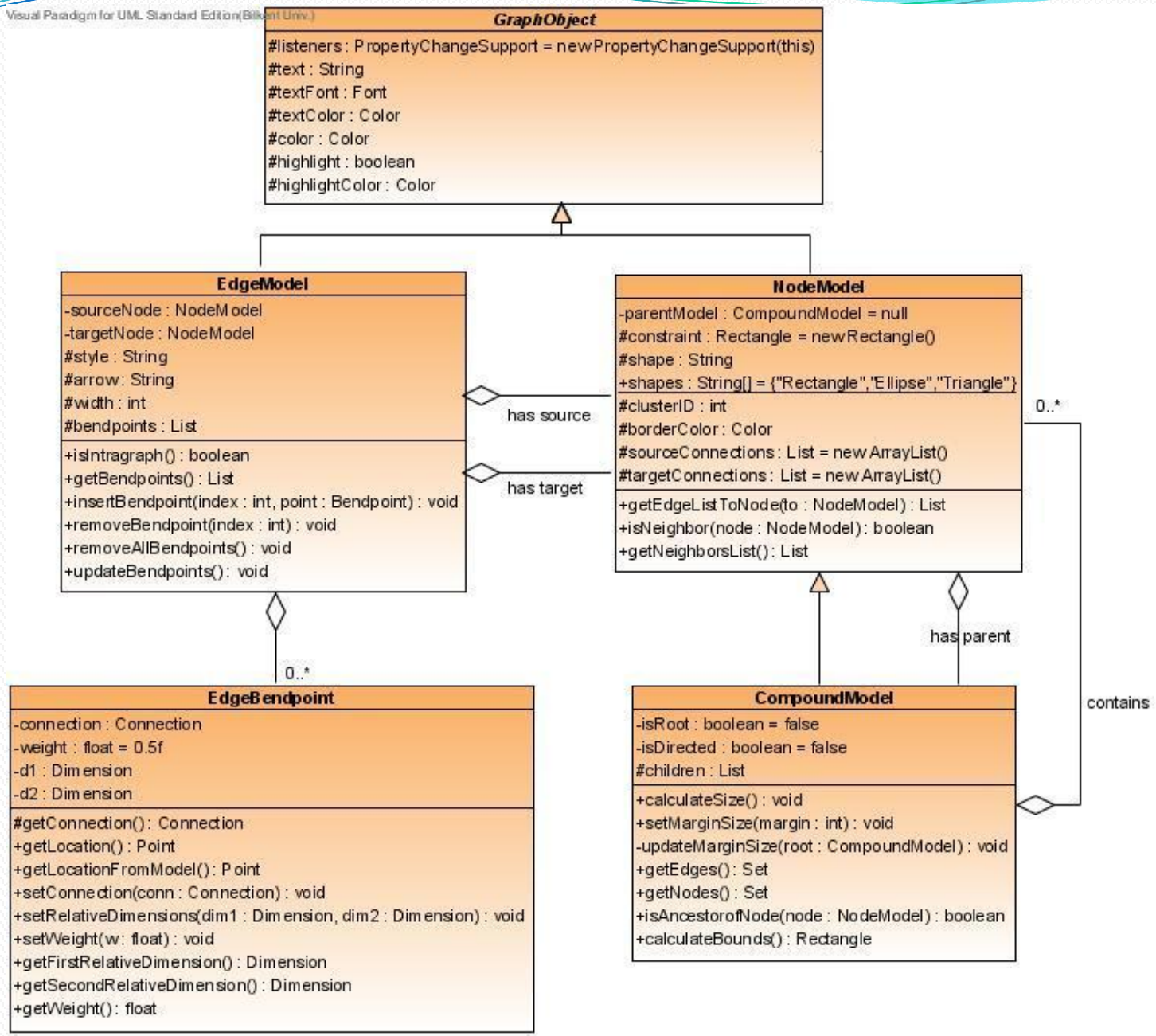
Architecture of CHISIO

- Model-View-Controller (MVC) Architecture
  - CHISIO uses GEF
  - GEF is based on MVC



OO Design of CHISIO

- CHISIO Models
  - Nodes
    - In edges list
    - Out edges list
  - Edges
    - Source node
    - Target node
  - Compound Nodes
    - Children nodes list


Visual Paradigm for UML Standard Edition(Bilkent Univ.)
GraphObject
#listeners : PropertyChangeSupport = new PropertyChangeSupport(this)
#text : String
#textFont : Font
#textColor : Color
#color : Color
#highlight : boolean
#highlightColor : Color
EdgeModel
-sourceNode : NodeModel
-targetNode : NodeModel
#style : String
#arrow: String
#width : int
#bendpoints : List
+isIntragraph() : boolean
+getBendpoints() : List
+insertBendpoint(index : int, point : Bendpoint) : void
+removeBendpoint(index : int) : void
+removeAllBendpoints() : void
+updateBendpoints() : void
NodeModel
-parentModel : CompoundModel = null
#constraint : Rectangle = new Rectangle()
#shape : String
+shapes : String[] = {"Rectangle","Ellipse","Triangle"}
#clusterID : int
#borderColor : Color
#sourceConnections : List = new ArrayList()
#targetConnections : List = new ArrayList()
+getEdgeListToNode(to : NodeModel) : List
+isNeighbor(node : NodeModel) : boolean
+getNeighborsList() : List
has source
has target
0..*
has parent
contains
0..*
EdgeBendpoint
-connection : Connection
-weight : float = 0.5f
-d1 : Dimension
-d2 : Dimension
#getConnection() : Connection
+getLocation() : Point
+getLocationFromModel() : Point
+setConnection(conn : Connection) : void
+setRelativeDimensions(dim1 : Dimension, dim2 : Dimension) : void
+setWeight(w: float) : void
+getFirstRelativeDimension() : Dimension
+getSecondRelativeDimension() : Dimension
+getWeight() : float
CompoundModel
-isRoot : boolean = false
-isDirected : boolean = false
#children : List
+calculateSize() : void
+setMarginSize(margin : int) : void
-updateMarginSize(root : CompoundModel) : void
+getEdges() : Set
+getNodes() : Set
+isAncestorofNode(node : NodeModel) : boolean
+calculateBounds() : Rectangle

OO Design of CHISIO

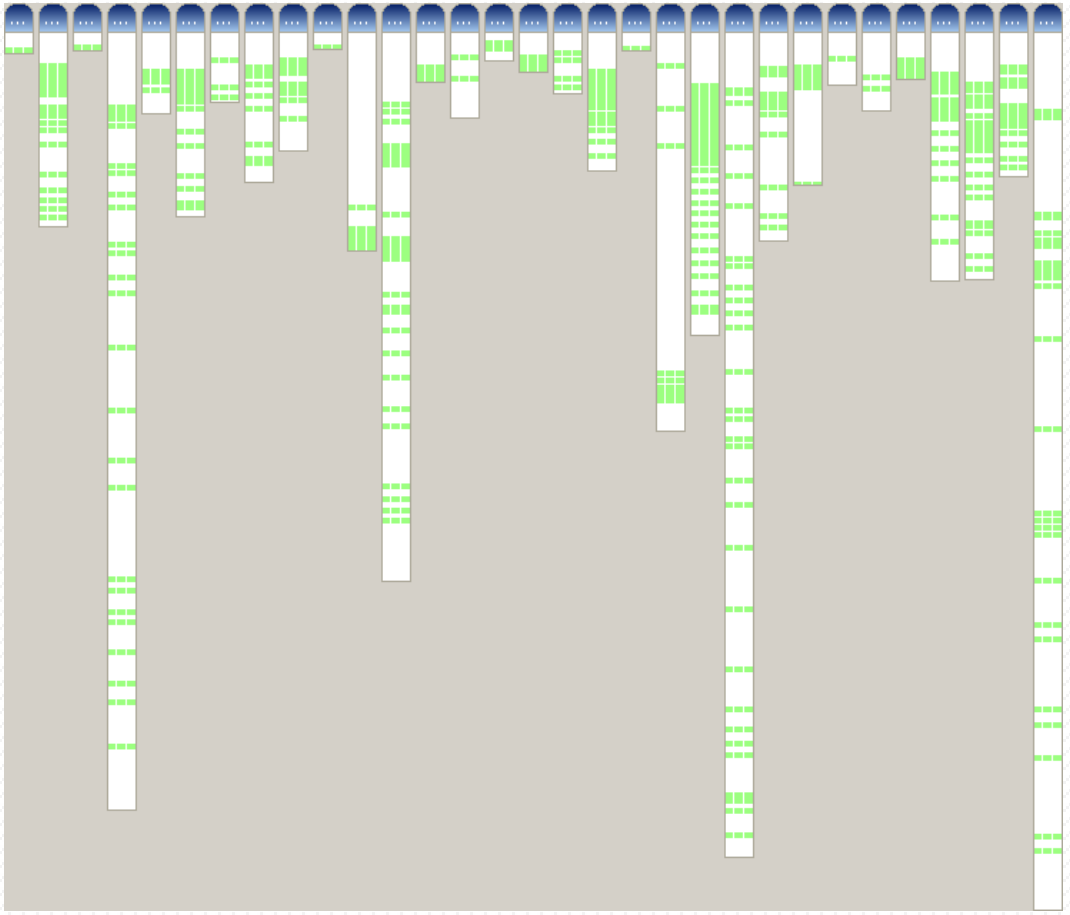
- Applied Design Patterns
  - Composite Pattern
  - Command Pattern
    - Commands are one of the core structures of GEF. All operations on *model* are performed via commands.
  - Singleton Pattern
    - Many dialogs in CHISIO is designed wrt. Singleton.

Crosscutting Concerns

- The notification of controller part of MVC by the model
- Domain specific constraints to scatter among commands, if introduced.
- Layout Profiling feature is inherently cross-cutting
- Action Tracking is similar to Layout Profiling

Problem of CCC

- Difficulty in
  - maintainability
  - adaptability
- Hard to
  - extend
  - understand
  - manage
- Reusability is reduced



The layout profiling aspect scattered among all layout package

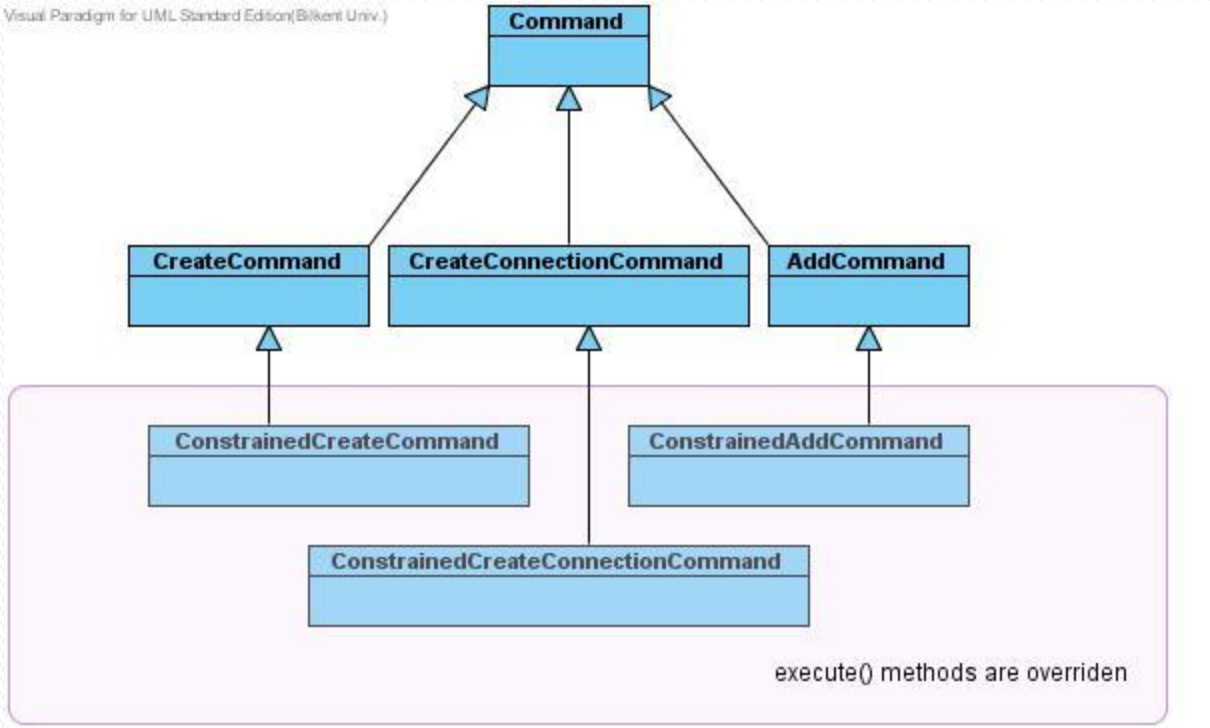
Aspect Identification

- In order to eliminate crosscutting concerns, we have introduced:
  - Production Aspects
    - Command Execution Constraints
    - Model Property Settings
  - Development Aspects
    - Layout Profiling
    - Action Tracking

Aspect Oriented Programming

- Production Aspects:
  - Command Execution Constraints
    - Not every edge type can be incident to every node type
    - Not every node type can be a child of every compound node type
    - The domain specifications determine node-edge attachment and compound-node containment rules
    - CHISIO commands should contain tangling code if we are to use an OO approach only
    - Hot Deployment feature of JBoss provides dynamic addition/removal of constraints

Prevented Class Hierarchy



Command Interceptor

public Object invoke(Invocation invocation)

- *Cast invocation to method invocation*
- *Assign target object of the method invocation to variable command*
- *If command is related with connection creation, check validity of node-edge relation*
- *Else check validity of node-compound relation*
- *If the validity check fails, show error message*
- *Else, run the command as normal*

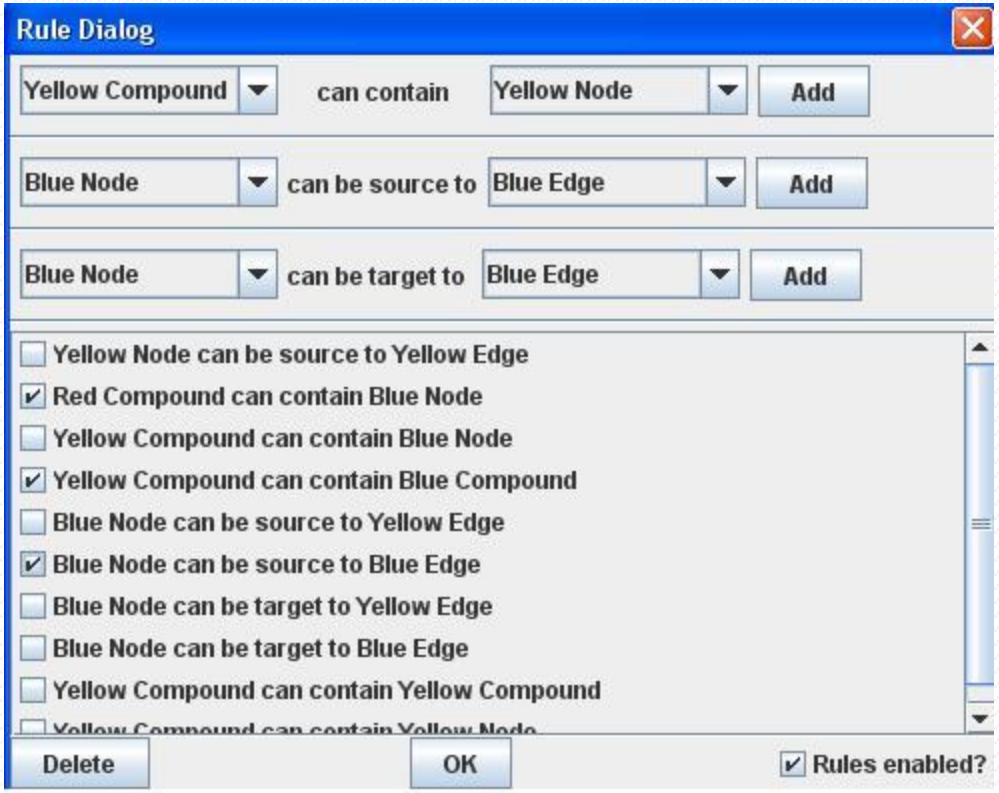
```
if (this.constraintEnabler.isSelected())
{
    AspectManager.instance().addBinding(this.binding);
}
else
{
    AspectManager.instance().removeBinding(this.binding.getName());
}
```

The Dynamic Binding of Advice

```
try {
    this.binding = new AdviceBinding(
            "execution(public void *.ChsCompoundCommand->execute())" +
            " OR execution(public void *.CreateCommand->execute())" +
            " OR execution(public void *." +
                "CreateConnectionCommand->execute())",
            null);
    this.binding.addInterceptor(CommandInterceptor.class);
} catch (ParseException e) {
```

The creation of a new Advice Binding

Rule Dialog


Rule Dialog
Yellow Compound
can contain
Yellow Node
Add
Blue Node
can be source to
Blue Edge
Add
Blue Node
can be target to
Blue Edge
Add
Yellow Node can be source to Yellow Edge
Red Compound can contain Blue Node
Yellow Compound can contain Blue Node
Yellow Compound can contain Blue Compound
Blue Node can be source to Yellow Edge
Blue Node can be source to Blue Edge
Blue Node can be target to Yellow Edge
Blue Node can be target to Blue Edge
Yellow Compound can contain Yellow Compound
Delete
OK
Rules enabled?

Aspect Oriented Programming

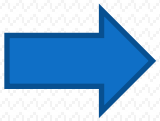
- Production Aspects:
  - Model Property Settings
    - Due to MVC pattern, the model must notify the edit parts (controllers) after certain changes, such as:
      - The setting of the fields within the set methods (such as position of a node, name of an edge, etc.)
      - The addition/removal of a node to/from a compound
      - The addition/removal of an edge to/from in/out edge list of a node
      - The insertion/removal of bend points to/from edge

Automating Notification in set methods

- Most of controller notification takes place in simple set methods
- Each different field needs an identifier string for the controller to discriminate it from others
- Must find a mechanism to capture all such fields
- Request the developers to adapt the following pattern:
  - Field Name: “sample”
  - Set Method Name: “setSample”
  - Identification String Name: “P_SAMPLE”
- Relies on Java Reflection to retrieve notification strings

Model Property Settings Aspect

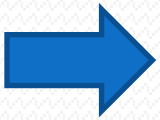
```
pointcut notificationChecker():
    staticinitialization(GraphObject+);
```



```
after():notificationChecker()
```

Creates a mapping btw. The fields and the corresponding notification strings, the first time a GraphObject type class is loaded.

```
pointcut propertySet(GraphObject graphObject, Object object):
    target(graphObject) &&
    set(* GraphObject+.*) &&
    args(object)&&
    withincode(void GraphObject+.set*(*));
```



```
after(GraphObject graphObject, Object object):
  propertySet(graphObject, object)
```

Uses the mapping created by the advice using notificationChecker() to get the notification string corresponding to the field set. Notifies the controllers via using the new value set to the field and the notification string obtained.

Model Property Settings Aspect

```
pointcut childNodeAdditionRemoval(CompoundModel compoundModel, Object child):
    target(compoundModel) &&
    args(child) &&
    execution(void CompoundModel.*Child(Object));

pointcut edgeAdditionRemoval(NodeModel nodeModel):
    target(nodeModel) &&
    (execution(void NodeModel.add*Connection(EdgeModel)) ||
        execution(void NodeModel.remove*Connection(EdgeModel)));

pointcut bendPointAdditionRemoval(EdgeModel edgeModel):
    target(edgeModel) &&
    execution(void EdgeModel.*Bendpoint*(..));
```

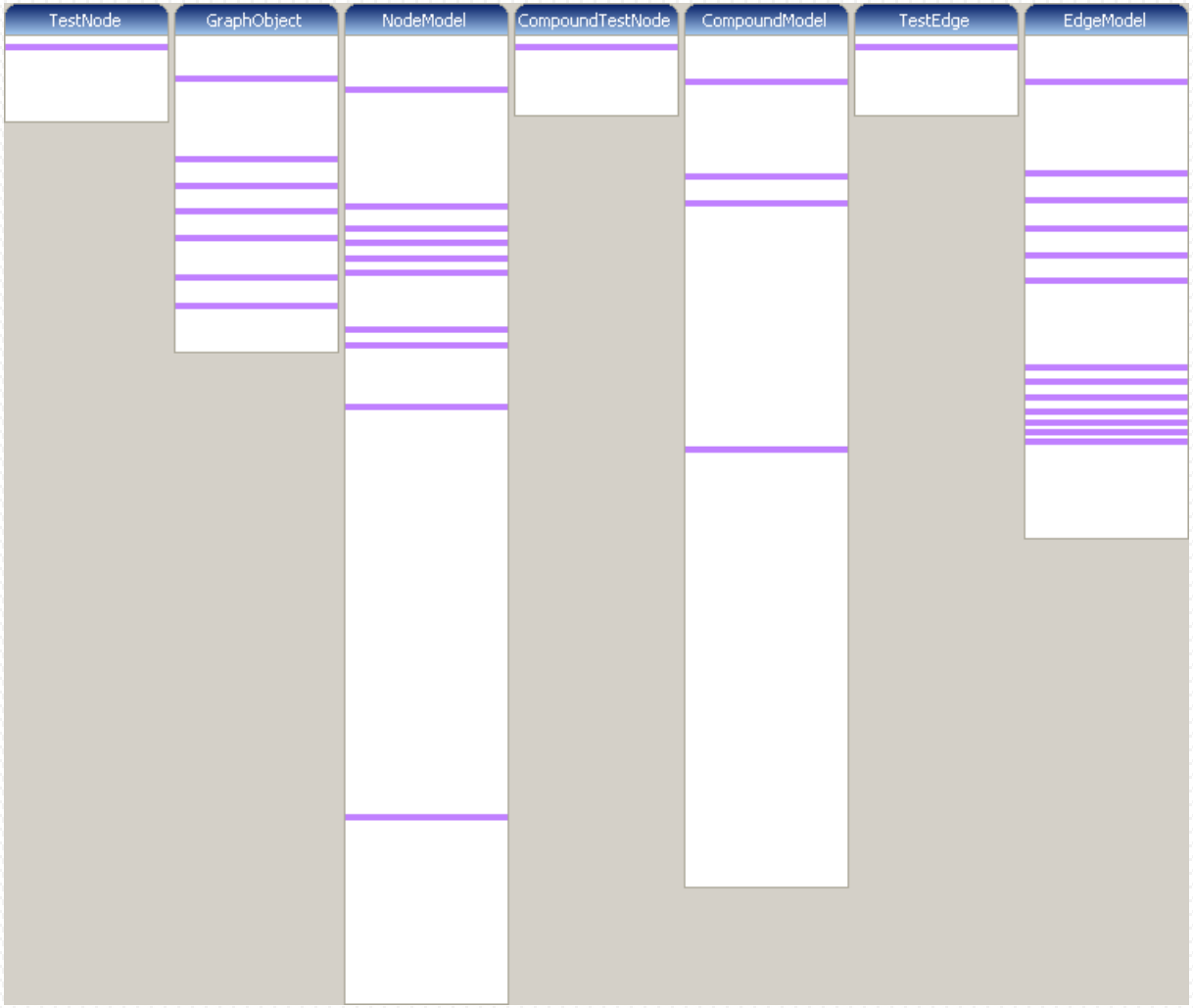
```
after(CompoundModel compoundModel, Object child):childNodeAdditionRemoval(
        compoundModel, child)

after(NodeModel nodeModel):edgeAdditionRemoval(nodeModel)

after(EdgeModel edgeModel):bendPointAdditionRemoval(edgeModel)
```

Specific point-cuts written for different GraphObject classes. The advices handle different situations where the controllers are needed to be handled.

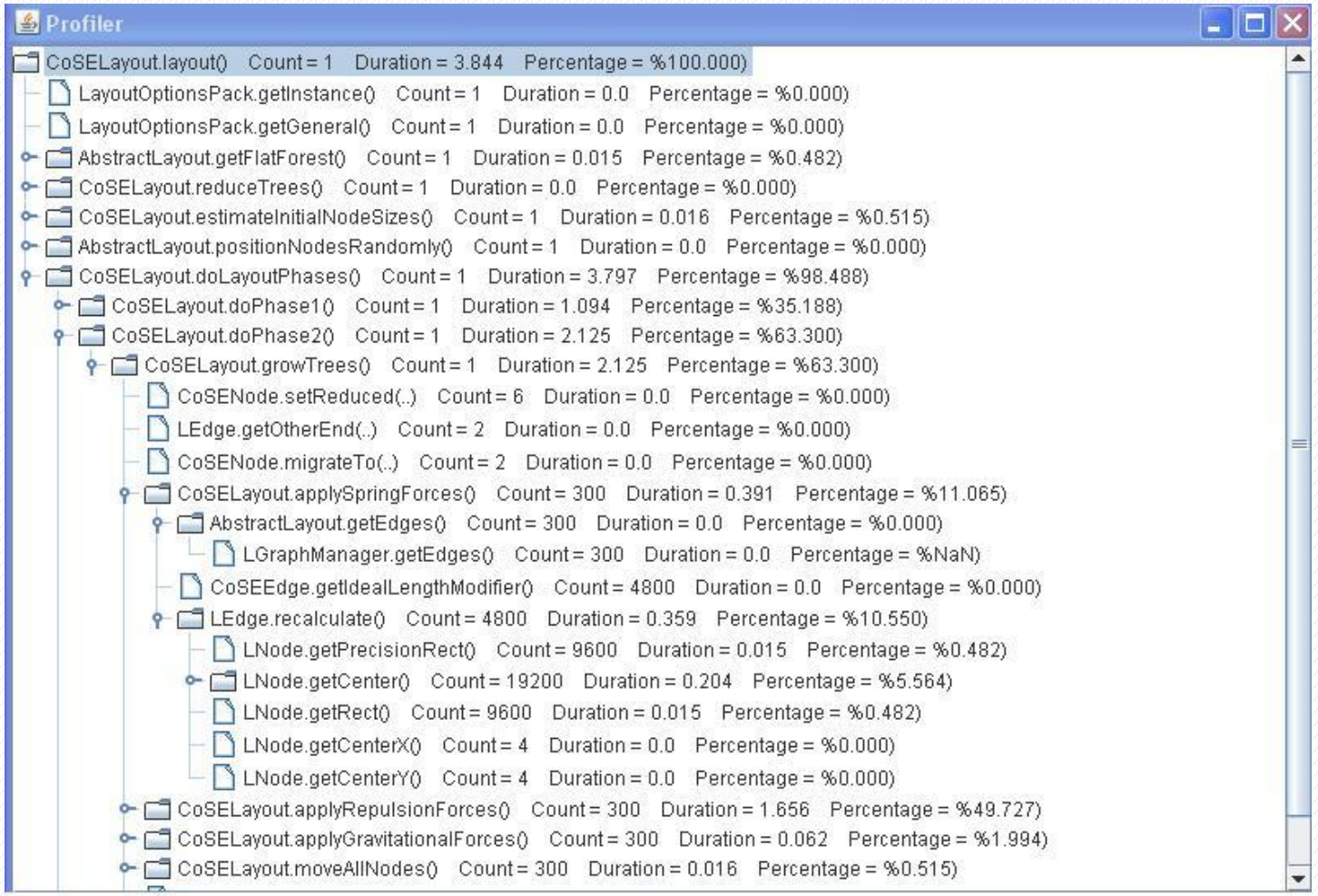
Model Property Settings Aspect



Aspect Oriented Programming

- Development Aspects:
  - Layout Profiling
    - Is a must functionality to test and debug layout algorithms
    - Determination of total count and time of individual methods
    - Detection of hot spots
    - Tree based representation eases understandability

Layout Profiling Console



Layout Profiling Aspect

```
pointcut runLayoutMethod():
    execution(* org.gvt.layout.AbstractLayout.runLayout(..));
```

```
before(): runLayoutMethod()
```

The profiling dialog, displaying a tree showing call hierarchy, is launched. Starts timer to refresh the tree.

```
after(): runLayoutMethod()
```

Stops auto-refresh of the tree inside the profiling dialog.

Layout Profiling Aspect

```
pointcut allLayoutMethodCalls():
    execution(* org.gvt.layout.*.*(..)) &&
    !runLayoutMethod() &&
    cflow(execution(* org.gvt.layout.AbstractLayout+.layout(..)));
```

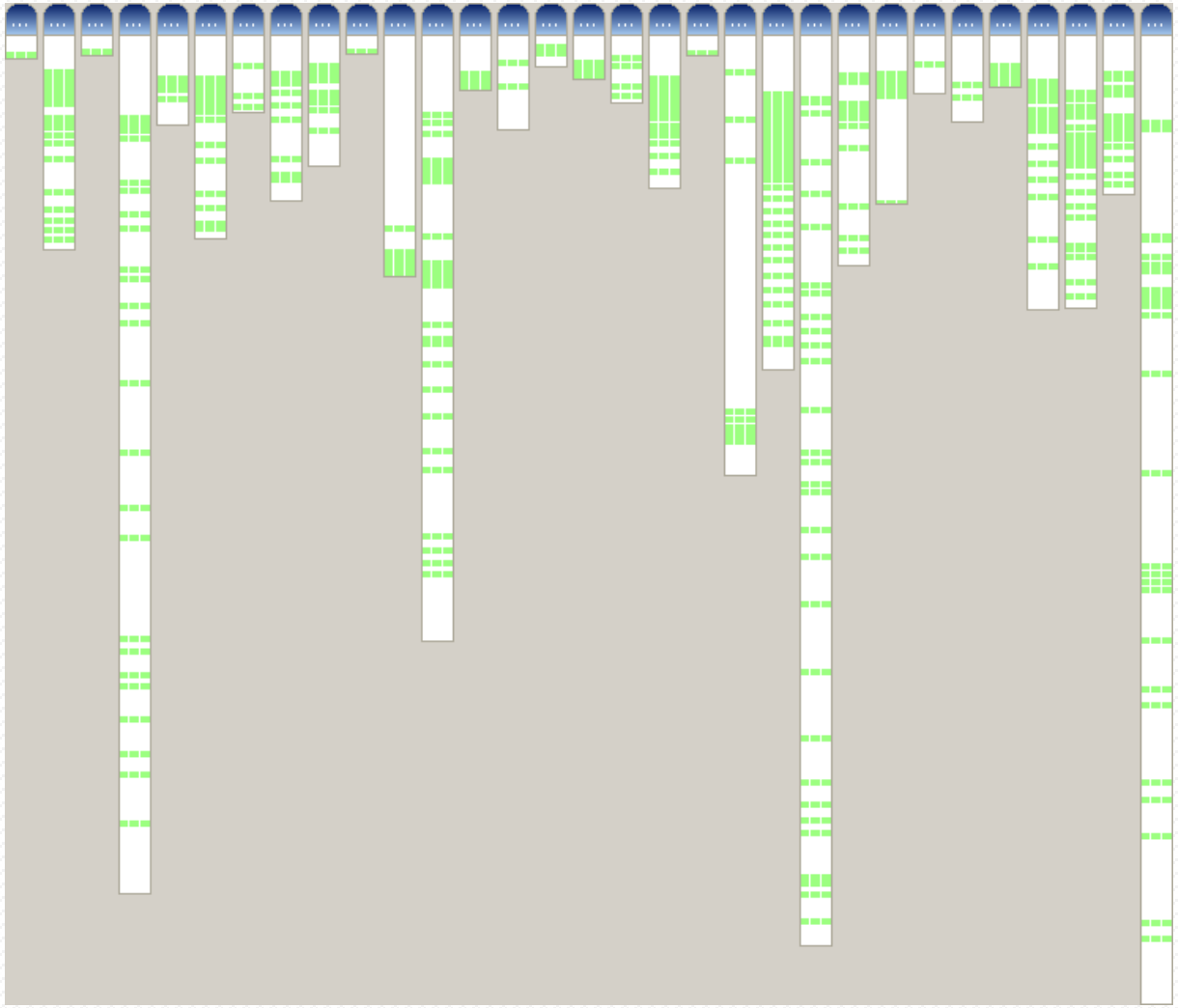
```
before(): allLayoutMethodCalls()
```

A tree node for the method to be executed is created (if not there already). The start time for the method is stored. The tree node along with other information is put into stack for further use.

```
after(): allLayoutMethodCalls()
```

The tree node and the other information put into stack is popped. The elapsed time is added to the total count of the method.

Layout Profiling Aspect



Aspect Oriented Programming

- Development Aspects:
  - Action Tracking
    - The underlying MVC structure of GEF is hard to learn & trace
    - Actions trigger introduction of EditPolicies, EditParts, Commands
    - The tracking mechanism reveals how the system works in a neat way
    - Reduces adaptation time to the project for the newcomers

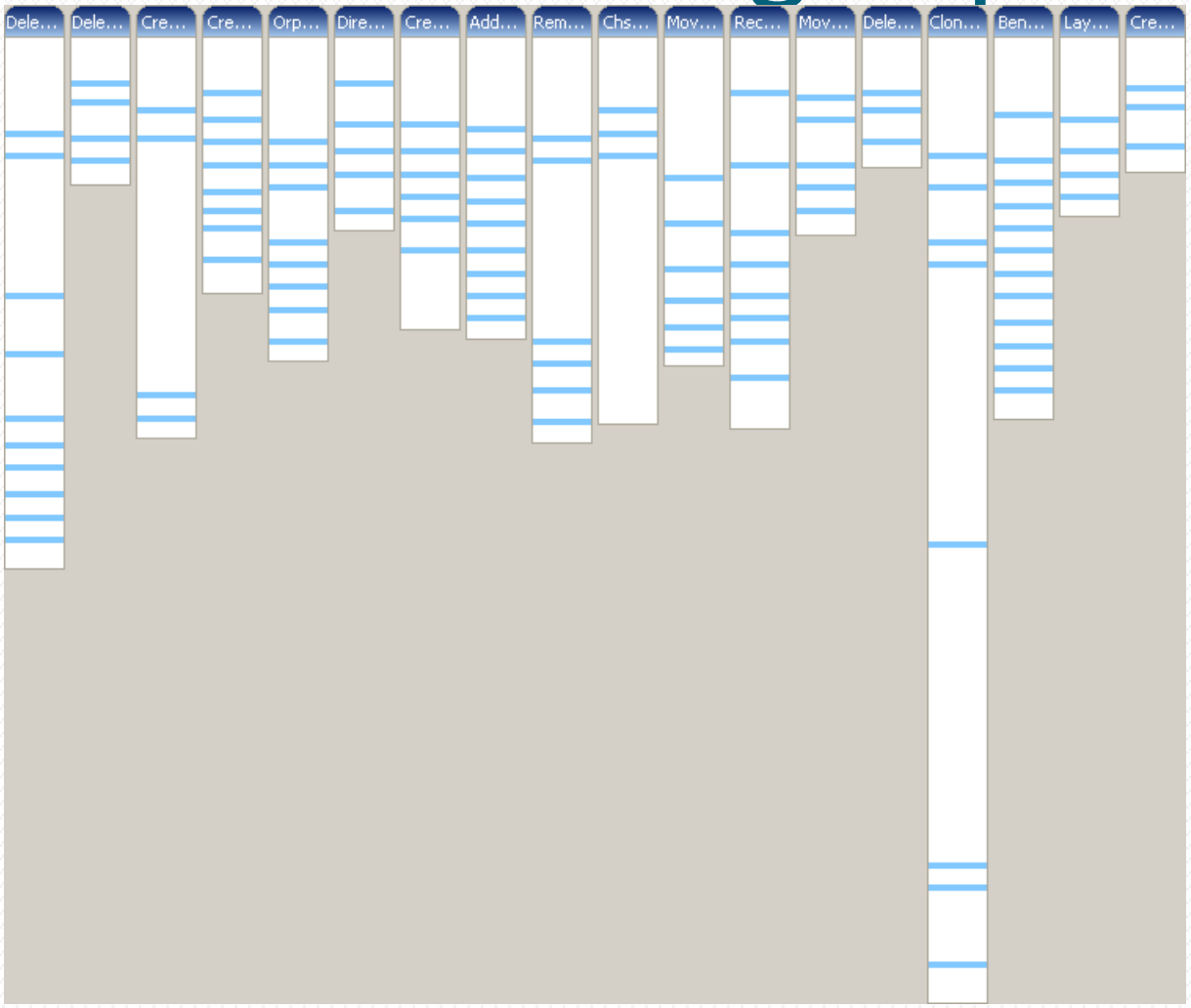
Action Tracking Aspect

```
pointcut editPolicyMethods() :
   execution (* org.eclipse.gef.EditPolicy+.*(..) );
pointcut commandMethods() :
   execution (* org.eclipse.gef.commands.Command+.*(..) );
pointcut commandConstructionMethods() :
    execution (org.eclipse.gef.commands.Command+.new(..));
```

```
before() : editPolicyMethods()
before() : commandMethods()
before() : commandConstructionMethods()
```

All the advices simply update the action tracking console.

Action Tracking Aspect



Action Tracking Console

Action Tracker

```
Command org.gvt.editpolicy.ChsContainerEditPolicy.getCreateCommand(CreateRequest)
Command org.gvt.editpolicy.ChsXYLayoutEditPolicy.getCreateCommand(CreateRequest)
org.gvt.command.CreateCommand(CompoundModel, NodeModel)
void org.gvt.command.CreateCommand.setConstraint(Rectangle)
boolean org.gvt.command.CreateCommand.canExecute()
Command org.gvt.editpolicy.ChsContainerEditPolicy.getCreateCommand(CreateRequest)
Command org.gvt.editpolicy.ChsXYLayoutEditPolicy.getCreateCommand(CreateRequest)
org.gvt.command.CreateCommand(CompoundModel, NodeModel)
void org.gvt.command.CreateCommand.setConstraint(Rectangle)
boolean org.gvt.command.CreateCommand.canExecute()
Command org.gvt.editpolicy.ChsContainerEditPolicy.getCreateCommand(CreateRequest)
Command org.gvt.editpolicy.ChsXYLayoutEditPolicy.getCreateCommand(CreateRequest)
org.gvt.command.CreateCommand(CompoundModel, NodeModel)
void org.gvt.command.CreateCommand.setConstraint(Rectangle)
boolean org.gvt.command.CreateCommand.canExecute()
Command org.gvt.editpolicy.ChsContainerEditPolicy.getCreateCommand(CreateRequest)
Command org.gvt.editpolicy.ChsXYLayoutEditPolicy.getCreateCommand(CreateRequest)
org.gvt.command.CreateCommand(CompoundModel, NodeModel)
void org.gvt.command.CreateCommand.setConstraint(Rectangle)
boolean org.gvt.command.CreateCommand.canExecute()
Command org.gvt.editpolicy.ChsContainerEditPolicy.getCreateCommand(CreateRequest)
Command org.gvt.editpolicy.ChsXYLayoutEditPolicy.getCreateCommand(CreateRequest)
org.gvt.command.CreateCommand(CompoundModel, NodeModel)
void org.gvt.command.CreateCommand.setConstraint(Rectangle)
boolean org.gvt.command.CreateCommand.canExecute()
Command org.gvt.editpolicy.ChsContainerEditPolicy.getCreateCommand(CreateRequest)
```

Related Work

- Commercial Graph Visualization Tools
  - TSV, JViews Diagrammer, yFiles, etc.
- Non-commercial Graph Visualization Tools
  - Prefuse, Graphlet, GDToolkit, etc.
- Not much work on an Aspect Oriented General Graph Visualization and Layout Tool
- Aspect Oriented UML Drawing Tool in First Turkish Aspect-Oriented Software Development Workshop in year 2003

Related Work

	Move/Resize/ Create/Delete	Zoom/ Scale	Compound Support	Highlighting	Save/ Load	Clustering/ Grouping	Object Inspector
TSV	+	+	+	+	+	+	+
JViews	+	+	+	+	+	+	+
yEd	+	+	+	-	+	+	+
aiSee	+	+	+	-	+	-	+
Graphviz	+	+	-	-	+	-	+
Prefuse	+	+	+	-	+	-	-
Gravisto	+	+	-	-	+	-	+
JGraphEd	+	+	-	-	+	-	+
VGJ	+	+	-	-	+	+	+
GDToolkit	+	+	-	-	+	-	+
uDraw	+	+	-	-	+	-	+
VCG	-	+	-	-	+	-	-
CHISIO	+	+	+	+	+	+	+

Comparison Table

Discussion & Conclusion

- CHISIO is refactored wrt. AOP paradigms
- Existing cross-cutting concerns are avoided as much as possible
- New features, which are inherently cross-cutting, such as Layout Profiling and Action Tracking are integrated to the system
- The difficulties in improving an already designed and implemented system are experienced
- A design from scratch with AOP in mind should have been easier.
- Further identification of cross-cutting concerns should be performed as future work

REFERENCES

- B. Tekinerdogan, CS586 Course Slides, Bilkent University, Ankara, Turkey+
- Graph Layout Toolkit and Graph Editor Toolkit User's Guide and Reference Manual. Tom Sawyer Software, Oakland, CA, USA, 1992-2002. http://www.tomsawyer.com.
- yFiles User's Guide. yWorks GmbH, D-72076 Tbingen, Germany, 2002. http://www.yworks.com.
- JViews User's Guide. ILOG SA, 94253 Gentilly Cedex, France, 2002. http://www.ilog.com.
- uDraw(Graph), 2005. http://www.informatik.uni-bremen.de/uDrawGraph.
- Graphviz - Graph Visualization Software. http://www.graphviz.org.
- VGJ, Visualizing Graphs with Java. Auburn University, Auburn, Alabama. http://www.infosun.fmi.uni-passau.de/Graphlet
- E. Gamma, R. Helm, R. Johnson and J. Vlissides.Design Patterns Elements of Reusable Object Oriented Software. Addison-Wesley, 1995..
- Graphlet, A toolkit for graph editors and graph algorithms. University of Passau, Passau, Germany. http://www.infosun.fmi.uni-passau.de/Graphlet
- Y. Dedeoğlu, H. Sözer, M. Tekkalmaz, A. Uluçınar. *Aspect-Oriented UML Class Diagram Drawing Tool*,1st Turkish AOSD Workshop Preceedings, TAOSD 2003.
- The GraphML File Format http://graphml.graphdrawing.org/
- The Prefuse Visualization Toolkit. http://prefuse.org/
- The AspectJ(TM) Programming Guide. http://www.eclipse.org/aspectj/doc/released/progguide/
- JBoss AOP http://www.jboss.org/jbossaop/
- Eclipse Graphical Editing Framework (GEF) http://www.eclipse.org/gef/
- Composite Pattern http://en.wikipedia.org/wiki/Composite_pattern
- Command Design Pattern http://www.jboss.org/jbossaop/

Thanks for listening…

- You can use CHISIO in your graph drawings by downloading it from CHISIO website
  - http://www.cs.bilkent.edu.tr/~ivis/chisio.html
- Any questions?