A Model Driven Approach for Graph Visualization

CS-587 Model Driven Software Development

Group Members:
Celal ÇIĞIR
Alptuğ DİLEK
Akif Burak TOSUN

Outline

- Objective
- Metamodeling
  - Description of Domain
  - Domain Analysis
  - DSL Grammar
  - Metamodel based on Ecore
  - Metamodel using UML profiling
  - Static semantics & Concrete Syntax
- Model Transformation
  - Model-to-Text transformation
  - Model-to-Model transformation
- Discussion and Lessons Learned
- Conclusion & Future Work

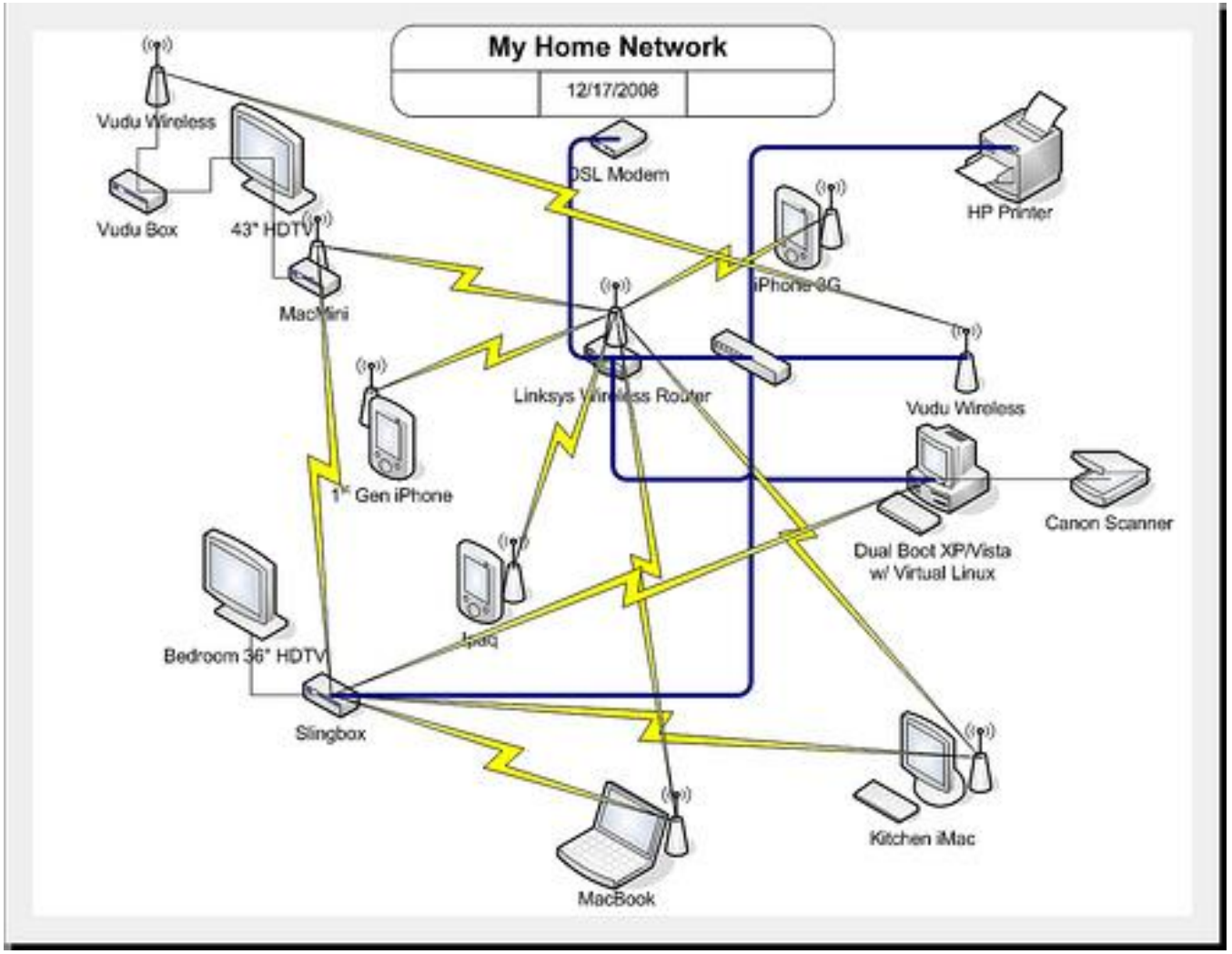
Objective

- Define an extended graph metamodel
  - graph visualization capability
  - corresponding transformations for interoperability between tools (Bridge like role)
- Make the metamodel comprehensive enough
  - support model-driven graph visualization software

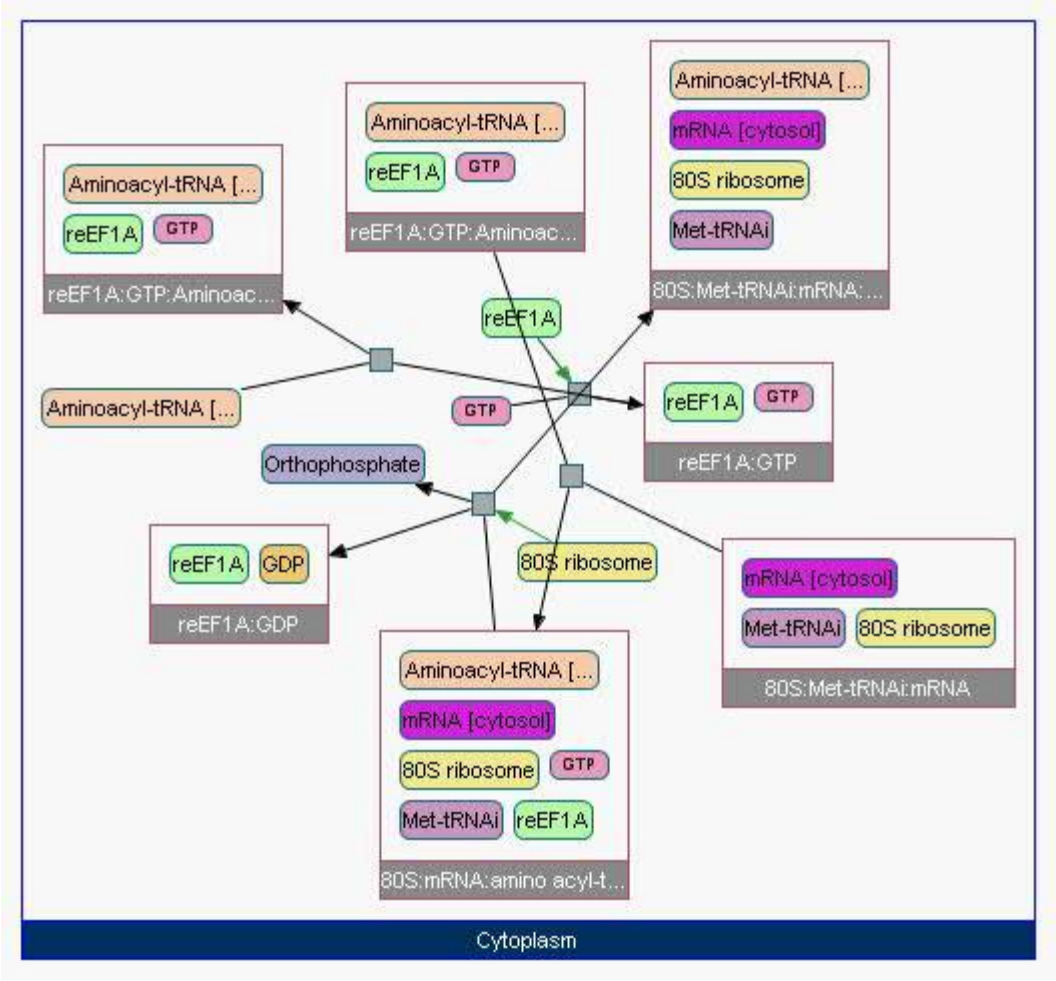
Description of Domain

- Graph Visualization is crucial
- Graphs are useful data structures
  - to capture the relations (represented by edges) between the objects (represented by nodes).
- It is widely used for modeling and simulating the real world problems
  - From network to data flow charts
  - From bioinformatics to computer science etc.

Application of Domain



Application of Domain



Domain Analysis

- Glossary
  - Graph is a kind of data structure that consists of a set of nodes and edges
  - Node is the basic unit of graph used to represent entities
  - Edge is the representation of a relation between the nodes in graph terminology
  - Graph object is the entity that contains common properties of the basic graph units (graph, node, edge)
  - Compound node is a special kind of node which can contain other nodes thanks to having a graph in itself
  - Label is a special unit either text or image associated to a graph object to show some related information

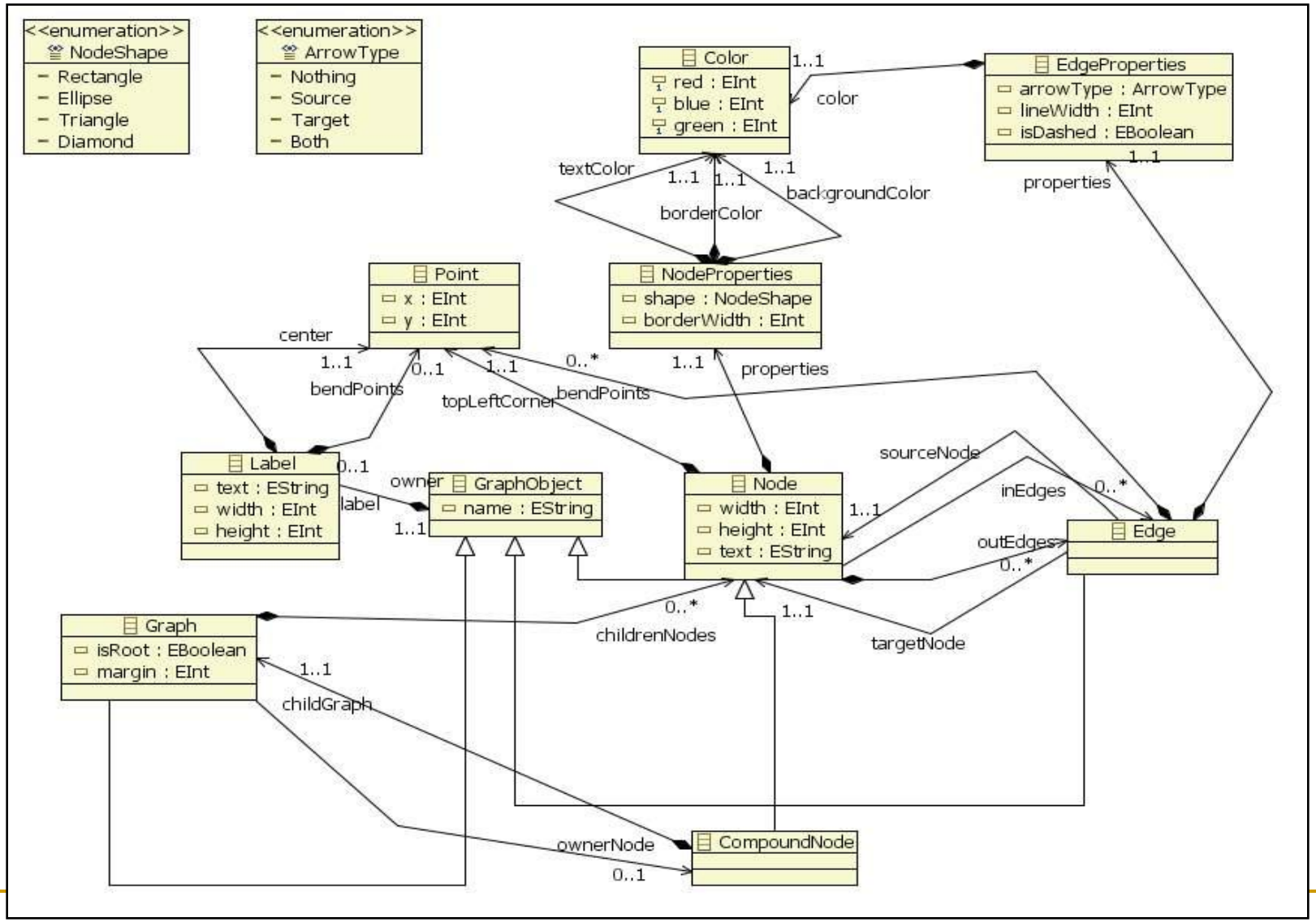
Domain Analysis

- Glossary
  - Node properties is a collection of properties to determine the appearance of a node such as the shape, line width, background and foreground color, etc.
  - Edge properties is a collection of properties to determine the appearance of an edge such as the line width, arrow type, color, etc.
  - Color is simply an RGB representation of 3 color channels
  - Point is the x-y coordinate representation to position various elements in the model

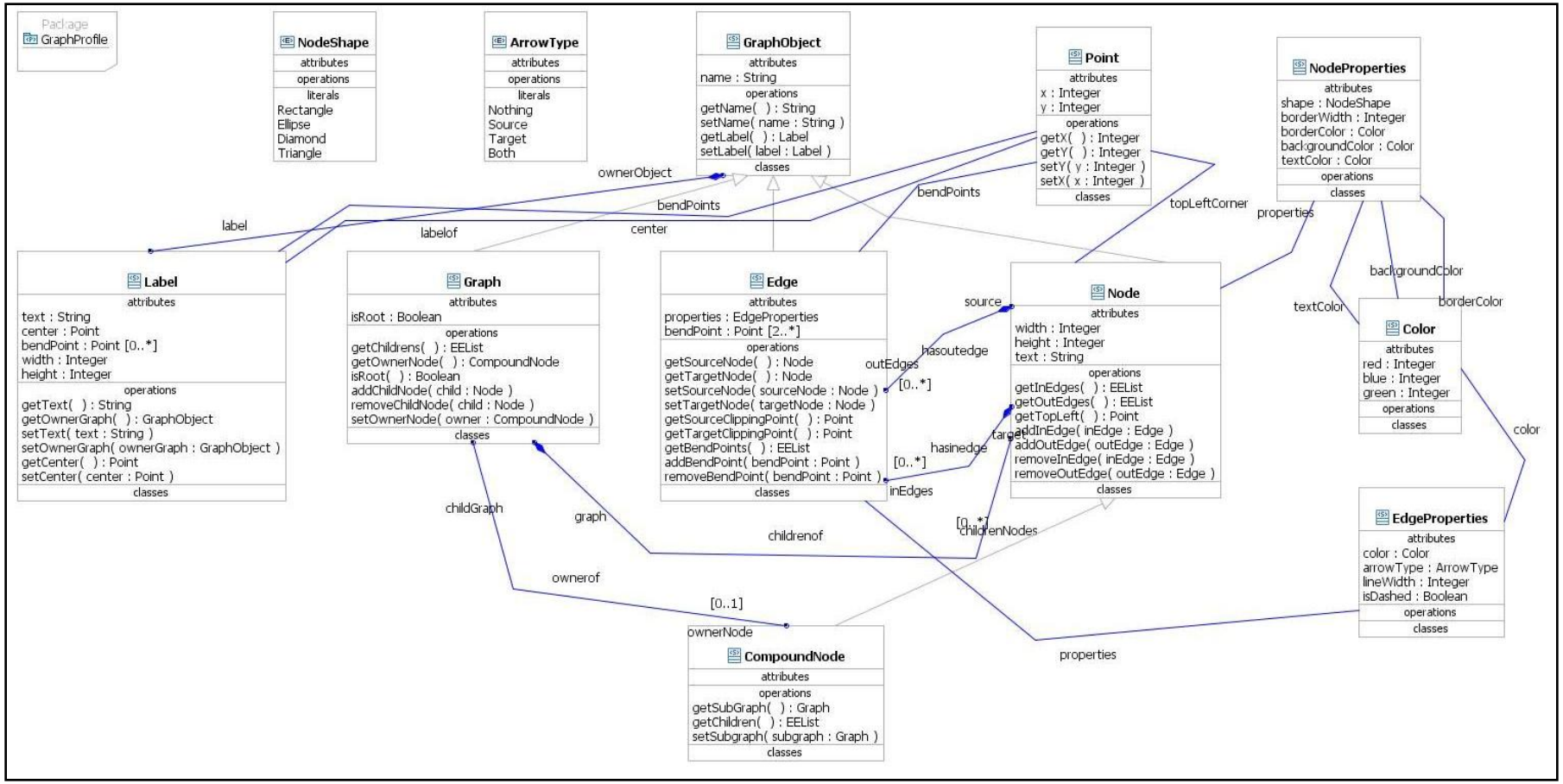
DSL Grammar

- GraphObject:: Label | Graph | Node | Edge
- Label:: PointRef
- PointRef:: Identifier
- Graph:: {Node}* {CompoundNode}?
- Node:: INEDGES OUTEDGES NodePropertiesRef PointRef
- INEDGES:: {Edge}*
- OUTEDGES:: {Edge}*
- NodePropertiesRef:: Identifier
- PointRef:: Identifier
- Edge:: EdgePropertiesRef PointRef {PointRef}+
- EdgePropertiesRef:: Identifier
- CompoundNode:: Graph
- NodeProperties:: FOREGROUND BACKGROUND
- FOREGROUND:: ColorRef
- BACKGROUND:: ColorRef
- ColorRef:: Identifier
- EdgeProperties:: ColorRef
- LabelLink:: LabelRef GraphObjectRef PointRef {PointRef}+
- LabelRef:: Identifier
- GraphObjectRef:: Identifier

Metamodel based on ECore



Metamodel using UML profiling



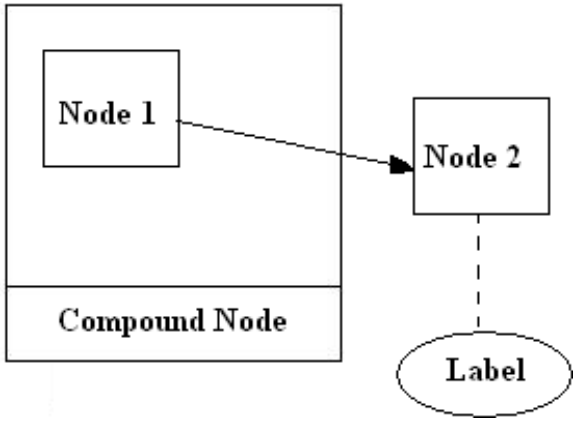
Static Semantics

```
(1) context Graph ERROR "Duplicate node names
detected!" :
        isRoot ?
nodesUnderGraph().forAll(n|n.withSameName(
                nodesUnderGraph())) : true;

(2) context Graph ERROR "The root graph must
not have an owner " +
        " compound node!":
        isRoot ? ownerNode == null : true;

(3) context Node ERROR "Zero length node name
not allowed!" :
        name.length > 0;

(4) context Node ERROR "Width and Height should
be greater than zero "
        +name :
        width > 0 && height > 0;

(5) context Node ERROR "Node Properties should
not be null!":
        this.properties != null;

(6) context NodeProperties ERROR
"BackgroundColor in NodeProperties should not
be null!":
        this.backgroundColor != null;

(7) context NodeProperties ERROR "BorderColor
in NodeProperties should not be null!":
        this.borderColor != null;

(8) context NodeProperties ERROR "TextColor in
NodeProperties should not be null!":
        this.textColor != null;
```

```
(9) context Color ERROR "Red value of Color
should be between 0 an 255!":
        this.red >= 0 && this.red <=255;

(10) context Color ERROR "Blue value of Color
should be between 0 an 255!":
        this.blue >= 0 && this.blue <=255;

(11) context Color ERROR "Green value of Color
should be between 0 an 255!":
        this.green >= 0 && this.green <=255;

(12) context CompoundNode ERROR "No graph defined
for "+name :
        childGraph != null;

(13) context CompoundNode ERROR "Child graph's
owner does not match "
        + " this compound node " + name + "!":
        childGraph.ownerNode == this;

(14) context Edge ERROR "Source and target node
should not be null!":
        sourceNode != null && targetNode != null;

(15) context Edge ERROR "Edge Properties should
not be null!":
        this.properties != null;

(16) context Edge ERROR "Source and target node
of an edge cannot" +
        " be same":
        sourceNode != targetNode;
```

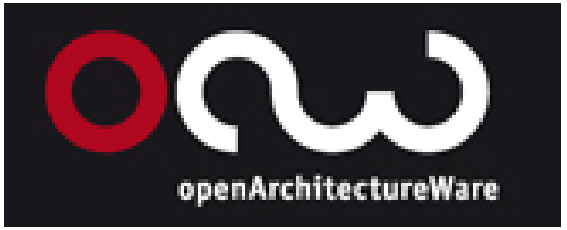
Concrete Syntax

- Interesting approach
- Actual concrete syntax defined in M1 Level
- Sample One:

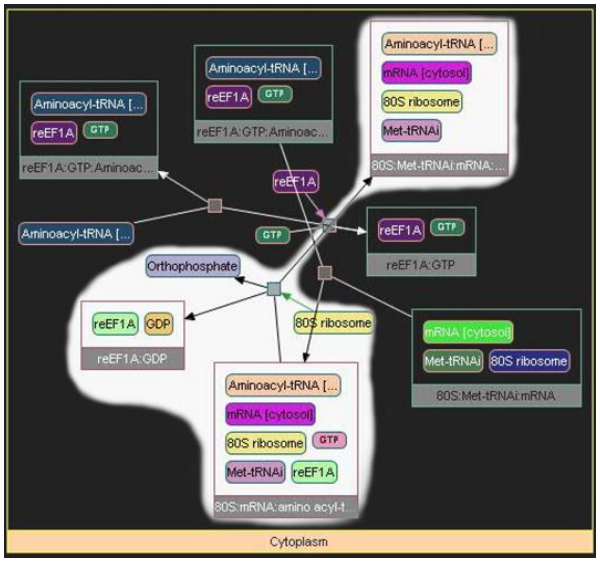


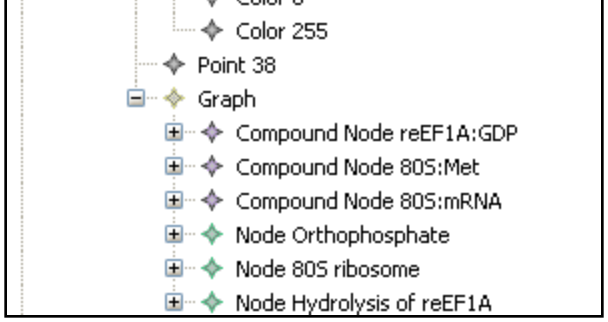
Model-to-Text Transformation

- Target Text Format : GraphML
- Objective :
  - validate the correctness of the metamodel
  - test the expressiveness power
- CHISIO for visualization
- openArchitectureWare:
  - Check, Xtend, Xpand
  - WorkFlow



M2T Example



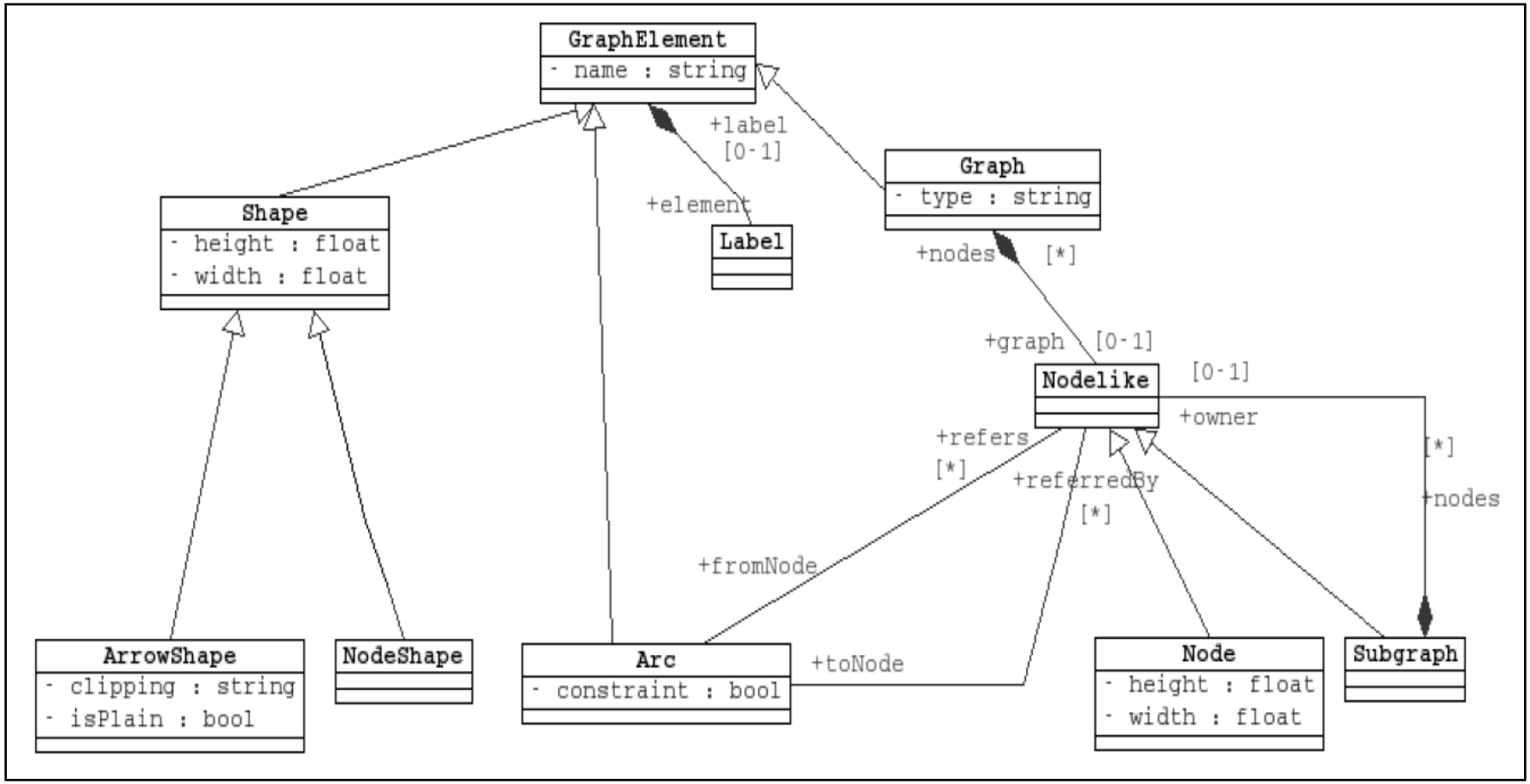



```
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
  <graph  id="graphroot">
    <data key="margin">0</data>
      <node id="Cytoplasm">
        <data key="text">Cytoplasm</data>
        <data key="shape">Rectangle</data>
          <graph  id="graphCytoplasm">
            <data key="margin">10</data>
              <node id="reEF1A:GDP">
              .....
              .....
  </graph>
</graphml>
```

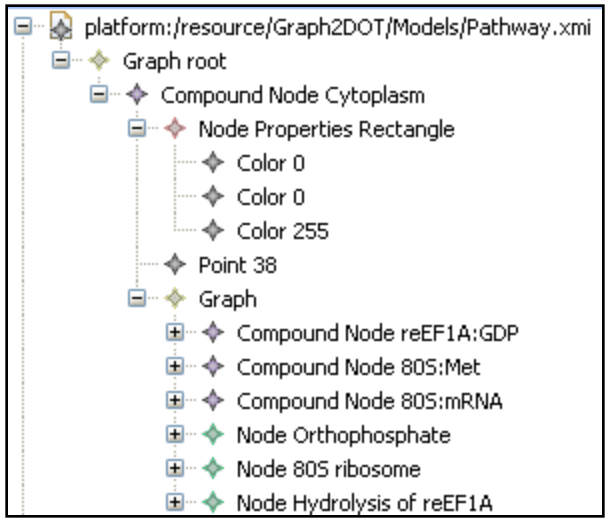
Model-to-Model Transformation

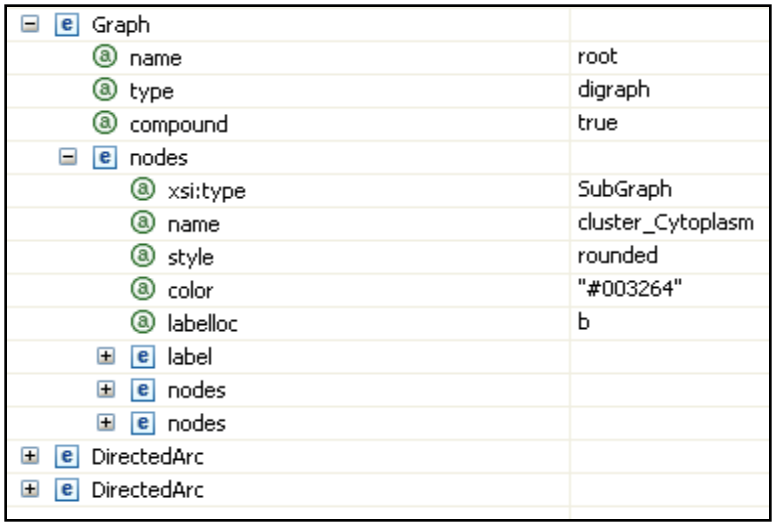
- Target Metamodel : DOT
  - DOTtoText : Another M2T Transformation
- Objective :
  - Interoperability btw. Graph visualization tools
    - GraphViz & CHISIO
- ATL (Atlas Transformation Language)

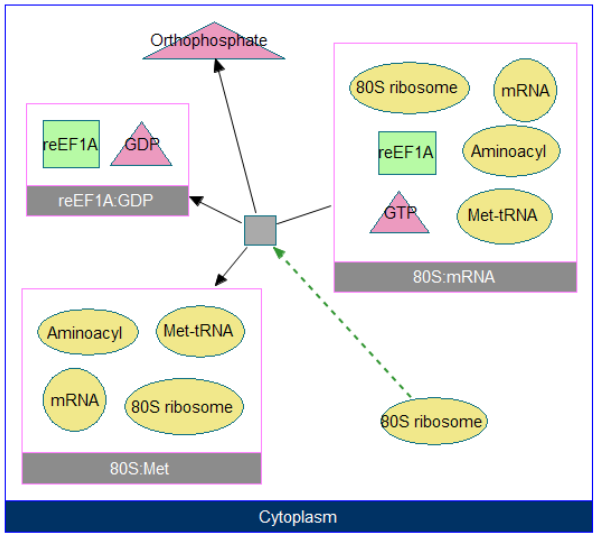
DOT Metamodel

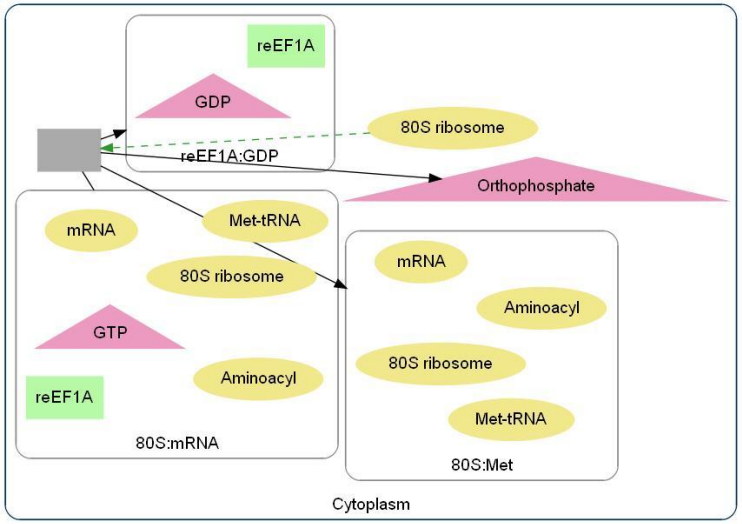


M2M Example


platform:/resource/Graph2DOT/Models/Pathway.xmi
Graph root
Compound Node Cytoplasm
Node Properties Rectangle
Color 0
Color 0
Color 255
Point 38
Graph
Compound Node reEF1A:GDP
Compound Node 80S:Met
Compound Node 80S:mRNA
Node Orthophosphate
Node 80S ribosome
Node Hydrolysis of reEF1A


Graph
name root
type digraph
compound true
nodes
xsi:type SubGraph
name cluster_Cytoplasm
style rounded
color "#003264"
labelloc b
label
nodes
nodes
DirectedArc
DirectedArc


Orthophosphate
reEF1A
GDP
reEF1A:GDP
80S ribosome
mRNA
reEF1A
Aminoacyl
GTP
Met-tRNA
80S:mRNA
Aminoacyl
Met-tRNA
mRNA
80S ribosome
80S:Met
80S ribosome
Cytoplasm


reEF1A
GDP
80S ribosome
reEF1A:GDP
Orthophosphate
mRNA
Met-tRNA
80S ribosome
GTP
reEF1A
Aminoacyl
80S:mRNA
mRNA
Aminoacyl
80S ribosome
Met-tRNA
80S:Met
Cytoplasm

Discussion and Lessons Learned

- Experience about the benefits of MDSD
  - Interoperability
  - Documentation-code sync
- Metamodeling is hard
  - Metathinking requires reflection
- Metamodeling requires broader knowledge
- M1 – M2 Level Discrimination
- Metamodels expressive than Grammars
- UML Profiling not as easy as it sounds

Discussion and Lessons Learned

- Model Transformations requires considerable
  - source metamodel & target metamodel knowledge
- Learned and Used new technologies
  - EMF
    - Incubating, many internal problems
  - TOPCASED plugin
  - Openarchitectureware
    - Not comprehensive tutorials
    - Many different syntaxes
  - ATL
    - Poor exception handling
- Tools to improve as MDSD becomes popular

Conclusion & Future Work

- Constructed Comprehensive and extended graph metamodel
  - Graph Visualization Capabilities
- Performed Model Transformations
  - validation and testing
  - interoperability
- MDSD steps applied successfully
- Need reverse-engineering (Text-to-Model Transformation) actual tool interoperability
- MD-CHISIO
  - customizable graph visualization tool
  - next step of domain specific visualization software

References

- [1] http://www.cs.bilkent.edu.tr/~ivis/chisio.html. CHISIO website.
- [2] http://www.graphviz.org/doc/info/lang.html. DOT file specification.
- [3] http://graphml.graphdrawing.org/. GraphML file specification.
- [4] Benguria, G.; Larrucea, X., "Data Model Transformation for Supporting Interoperability," *Commercial-off-the-Shelf (COTS)- Based Software Systems, 2007. ICCBSS '07. Sixth International IEEE Conference on , vol., no., pp.172-181, Feb. 26 2007-March* 2 2007
- [5] http://en.wikipedia.org/wiki/Graph_(data_structure)
- [6] http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf EBNF specification
- [7] http://www.topcased.org/. Topcased offical website.
- [8] http://www.eclipse.org/ Eclipse offical website.
- [9] OCL1.5 Specification: http://www.omg.org/cgibin/apps/doc?formal/03-03-13.
- [10] OCL 2.0 Specification: http://www.omg.org/cgibin/apps/doc?ptc/2003-10-14.
- [11] Unified Modeling Language (UML), Version 1.5, OMG documentformal/03-03-01, Object Management Group (2003),http://www.omg.org/cgi-bin/doc?formal/03-03-01.
- [12] http://www.eclipse.org/modeling/emf/
- [13] Kovse J. *'Generic Model-to-Model Transformations in MDA: Why and How?' Workshop Generative Techniques in the Context*
- od Model Driven Architecture, OOPSLA 2002.
- [14] http://www.eclipse.org/m2m/atl/ Atlas model transformation language.
- [15] UML 2.0 Superstructure Specification, OMG document formal/05-07-04, Object Management Group, Inc. (2005), http://www.omg.org/cgi-bin/doc?formal/05-07-04.

QUESTIONS ?

- THANKS FOR LISTENING