# Transient View Generation in Eclipse

Christian Schneider

Real-Time Systems and Embedded Systems Group
Department of Computer Science
Christian-Albrechts-Universität zu Kiel
`www.informatik.uni-kiel.de/rtsys`

ACademics Modelling with Eclipse, July 2, 2012

**K**iel **I**ntegrated **E**nvironment for **L**ayout
Eclipse Rich Client

# Outline

## KIELER

Message

KIELER @ Work – Employment in MENGES

## Related Work

## Transient Views

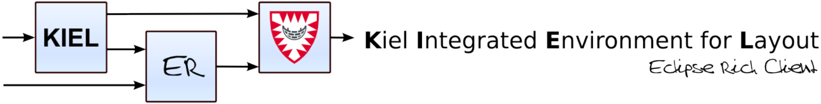
Motivating Examples

Characterization

KRendering Description Model

## Conclusion

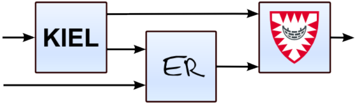
# KIELER Message



Challenge: Free user of manual mechanical work

Message: While modeling focus on *model-ing*

# KIELER Message



**K**iel **I**ntegrated **E**nvironment for **L**ayout
Eclipse Rich Client

Challenge: Free user of manual mechanical work

Message: While modeling focus on *model-ing*

Our focus: *Pragmatics* of modeling languages

- Apply the MVC to the users' perspective

**Key enabler: flexible & content-aware automatic layout**

H. Fuhrmann and R. v. Hanxleden, Taming Graphical Modeling (MoDELS'10)

# KIELER @ Work – Employment in MENGES KoSSE[1]

- Joint research project of industry & academia
- Aims at developing DSLs for railway signaling systems
- Specifications are of textual nature
  - ☺ easy to formulate
  - ☺ version control
  - ☹ comprehensibility reduces
- Shall be extended by graphical views on various aspects

[1] https://menges.informatik.uni-kiel.de/

# KIELER @ Work – Employment in MENGES[1]

KoSSE

- Joint research project of industry & academia
- Aims at developing DSLs for railway signaling systems
- Specifications are of textual nature
  - ☺ easy to formulate
  - ☺ version control
  - ☹ comprehensibility reduces
- Shall be extended by graphical views on various aspects
  - Somehow . . .

[1] https://menges.informatik.uni-kiel.de/

# Related Work

- Akos Ledeczi et al.
  The Generic Modeling Environment
  IEEE Workshop on Intelligent Signal Processing (WISP 2001)
- Gergely Mezei et al.
  Visual presentation solutions for domain specific languages
  IASTED International Conference on Software Engineering (2006)
- GMF Tooling

# Related Work

- Akos Ledeczi et al.
  The Generic Modeling Environment
  IEEE Workshop on Intelligent Signal Processing (WISP 2001)

- Gergely Mezei et al.
  Visual presentation solutions for domain specific languages
  IASTED International Conference on Software Engineering (2006)

- GMF Tooling

- Hauke Fuhrmann and Reinhard von Hanxleden
  Taming Graphical Modeling
  ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2010)

# Related Work

Akos Ledeczi et al.
The Generic Modeling Environment
IEEE Workshop on Intelligent Signal Processing (WISP 2001)

Gergely Mezei et al.
Visual presentation solutions for domain specific languages
IASTED International Conference on Software Engineering (2006)

GMF Tooling

Hauke Fuhrmann and Reinhard von Hanxleden
Taming Graphical Modeling
ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2010)

Benjamin B. Bederson et al.
Toolkit Design for Interactive Structured Graphics
IEEE Transactions on Software Engineering, Vol. 30, No. 8 (2004)

# Related Work

Robert Ian Bull
Towards A Model Driven Engineering Approach For Information Visualization
Ph.D. thesis, University of Victoria, BC, Canada (2008)

Graphiti project (http://www.eclipse.org/graphiti/)

Jan Koehnlein
Discovery Diagrams for the Generic Graphical View
http://koehnlein.blogspot.de/2012/01/discovery-diagrams-for-generic.html

# Outline

KIELER

- Message
- KIELER @ Work – Employment in MENGES

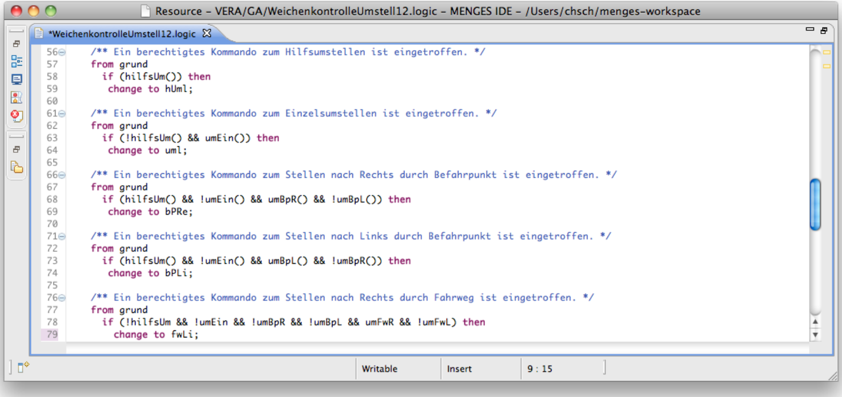
Related Work

Transient Views

- Motivating Examples
- Characterization
- KRendering Description Model

Conclusion

# Example 1: Textual DSLs

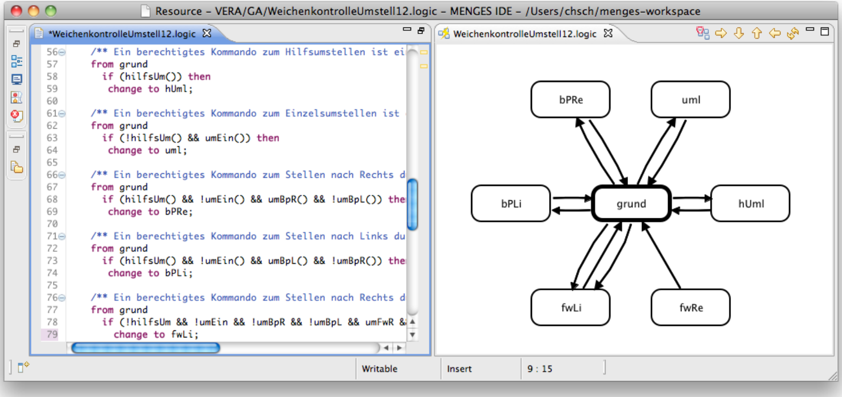


Resource – VERA/GA/WeichenkontrolleUmstell12.logic – MENGES IDE – /Users/chsch/menges-workspace

*WeichenkontrolleUmstell12.logic

```
/** Ein berechtigtes Kommando zum Hilfsumstellen ist eingetroffen. */
from grund
  if (hilfsUm()) then
   change to hUml;

/** Ein berechtigtes Kommando zum Einzelsumstellen ist eingetroffen. */
from grund
  if (!hilfsUm() && umEin()) then
   change to uml;

/** Ein berechtigtes Kommando zum Stellen nach Rechts durch Befahrpunkt ist eingetroffen. */
from grund
  if (hilfsUm() && !umEin() && umBpR() && !umBpL()) then
   change to bPRe;

/** Ein berechtigtes Kommando zum Stellen nach Links durch Befahrpunkt ist eingetroffen. */
from grund
  if (hilfsUm() && !umEin() && umBpL() && !umBpR()) then
   change to bPLi;

/** Ein berechtigtes Kommando zum Stellen nach Rechts durch Fahrweg ist eingetroffen. */
from grund
  if (!hilfsUm && !umEin && !umBpR && !umBpL && umFwR && !umFwL) then
    change to fwLi;
```

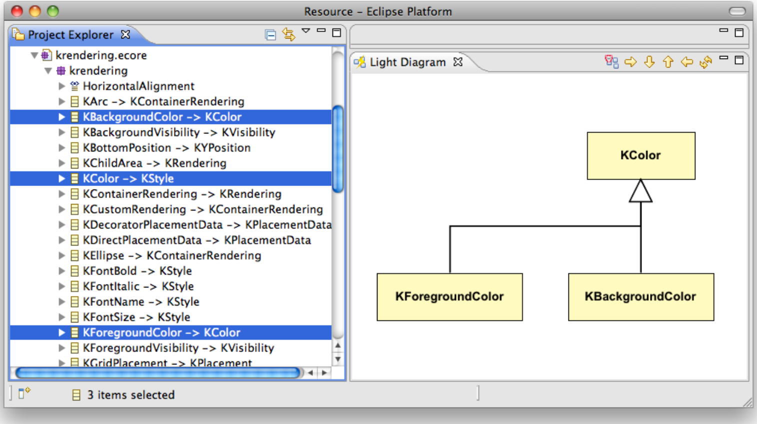
# Proposal: Transient Graphical Diagram – Demo



Resource – VERA/GA/WeichenkontrolleUmstell12.logic – MENGES IDE – /Users/chsch/menges-workspace
*WeichenkontrolleUmstell12.logic
WeichenkontrolleUmstell12.logic
bPRe
uml
bPLi
grund
hUml
fwLi
fwRe
Writable
Insert
9 : 15

```
    /** Ein berechtigtes Kommando zum Hilfsumstellen ist ei
    from grund
      if (hilfsUm()) then
       change to hUml;

    /** Ein berechtigtes Kommando zum Einzelsumstellen ist
    from grund
      if (!hilfsUm() && umEin()) then
       change to uml;

    /** Ein berechtigtes Kommando zum Stellen nach Rechts d
    from grund
      if (hilfsUm() && !umEin() && umBpR() && !umBpL()) the
       change to bPRe;

    /** Ein berechtigtes Kommando zum Stellen nach Links du
    from grund
      if (hilfsUm() && !umEin() && umBpL() && !umBpR()) the
       change to bPLi;

    /** Ein berechtigtes Kommando zum Stellen nach Rechts d
    from grund
      if (!hilfsUm && !umEin && !umBpR && !umBpL && umFwR &
        change to fwLi;
```

# Example 2: Class Diagrams – Demo



- krendering.ecore
  - krendering
    - HorizontalAlignment
    - KArc -> KContainerRendering
    - KBackgroundColor -> KColor
    - KBackgroundVisibility -> KVisibility
    - KBottomPosition -> KYPosition
    - KChildArea -> KRendering
    - KColor -> KStyle
    - KContainerRendering -> KRendering
    - KCustomRendering -> KContainerRendering
    - KDecoratorPlacementData -> KPlacementData
    - KDirectPlacementData -> KPlacementData
    - KEllipse -> KContainerRendering
    - KFontBold -> KStyle
    - KFontItalic -> KStyle
    - KFontName -> KStyle
    - KFontSize -> KStyle
    - KForegroundColor -> KColor
    - KForegroundVisibility -> KVisibility
    - KGridPlacement -> KPlacement

3 items selected

# Transient Graphical Views

Characteristics:

- Lightweight
- Flexible
- “Intelligent”
- Easy to define & contribute

# Transient Graphical Views

## Characteristics:

- Lightweight
- Flexible
- “Intelligent”
- Easy to define & contribute

## Requirements:

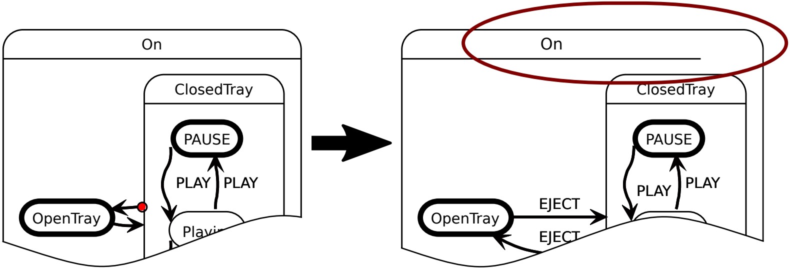
- Automatic arrangement of depicted elements – macro layout
- Local arrangement of the figures’ primitives – micro layout
- Drawing by means of an efficient graphics framework
- Appropriate description language to formulate diagrams

# Our Contribution: KRendering Meta Model

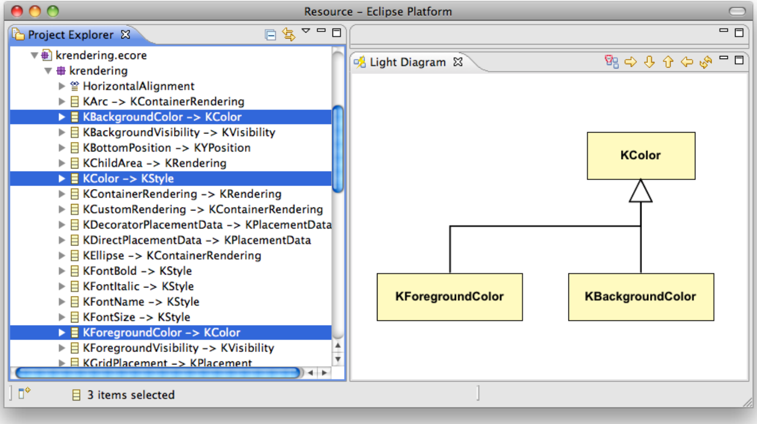
- Designed for describing concrete diagrams
- Is built upon the KGraph meta model used by the KIELER Infrastructure for Meta Layout (KIML)
- Provides primitive figures to be composed to complex ones
- Enables smart micro layout descriptions

# Our Contribution: KRendering Meta Model

- Designed for describing concrete diagrams
- Is built upon the KGraph meta model used by the KIELER Infrastructure for Meta Layout (KIML)
- Provides primitive figures to be composed to complex ones
- Enables smart micro layout descriptions



# KRendering Diagram Description – Example



- krendering.ecore
  - krendering
    - HorizontalAlignment
    - KArc -> KContainerRendering
    - KBackgroundColor -> KColor
    - KBackgroundVisibility -> KVisibility
    - KBottomPosition -> KYPosition
    - KChildArea -> KRendering
    - KColor -> KStyle
    - KContainerRendering -> KRendering
    - KCustomRendering -> KContainerRendering
    - KDecoratorPlacementData -> KPlacementData
    - KDirectPlacementData -> KPlacementData
    - KEllipse -> KContainerRendering
    - KFontBold -> KStyle
    - KFontItalic -> KStyle
    - KFontName -> KStyle
    - KFontSize -> KStyle
    - KForegroundColor -> KColor
    - KForegroundVisibility -> KVisibility
    - KGridPlacement -> KPlacement

3 items selected

# KRendering Diagram Description – Example

```
KNode { /* the diagram */
  KShapeLayout {
    algorithm = "ogdf.planarization",
    direction = UP, spacing = 75.0
  }
  KNode { /* "KColor" figure */
    KShapeLayout {
       width 180.0 height 80.0
    }
    Rectangle {
      lineWidth 2, backgroundColor 255 250 205
      Text "KColor" {
        bold, fontSize 20,
        backgroundColor 255 250 205
      }
    }
  }
  KNode { /* "KBackgroundColor" figure */
    KShapeLayout {
      width 242.0 height 80.0
    }
    Rectangle {
      lineWidth 2, backgroundColor 255 250 205
      Text "KBackgroundColor" {
         bold, fontSize 20,
         backgroundColor 255 250 205
      }
    }
  }
```

```
  --> "//children.0" {
    KEdgeLayout {
       edgeType = GENERALIZATION
    }
    Polyline  {
      lineWidth 2
      Polygon {
        lineWidth 2, backgroundColor 255 255 255
        polylinePlacementData {
          points:
            left   0.0 0.0 / top 0.0 0.0,
            right  0.0 0.0 / top 0.0 0.5,
            left   0.0 0.0 / bottom 0.0 0.0
          detailedPlacementData:
            decoratorPlacementData {
              relative,  location  1.0,
              xOffset  -35, yOffset -17.5,
              width 35, height 35
            }
        }
      }
    }
  }
  KNode { /* "KForegroundColor" figure */
    KShapeLayout {
      width 242.0 height 80.0
    }
  ...
}
```

# Benefits & Realization

- Enables view synthesis in model-based fashion
- Efficient application of automatic layout (no graph extraction)
- Views can be updated interactively by changing the model
- Forms a basis for efficient view management as proposed by

H. Fuhrmann and R. v. Hanxleden, Taming Graphical Modeling (MoDELS'10)

# Benefits & Realization

- Enables view synthesis in model-based fashion
- Efficient application of automatic layout (no graph extraction)
- Views can be updated interactively by changing the model
- Forms a basis for efficient view management as proposed by

  H. Fuhrmann and R. v. Hanxleden, Taming Graphical Modeling (MoDELS'10)

- Experimentally implemented in the KIELER Lightweight Diagrams project (KLighD)
  - Is provided with mappings
  - Chooses a fitting one if objects are to be depicted

# Outline

KIELER

- Message
- KIELER @ Work – Employment in MENGES

Related Work

Transient Views

- Motivating Examples
- Characterization
- KRendering Description Model

Conclusion

# Conclusion & Further Work

- Lots of productivity wasted with drawing diagrams manually
- Transient views appear to be a promising means for visualizing models or model excerpts
- Form a means for browsing and even modifying models

# Conclusion & Further Work

- Lots of productivity wasted with drawing diagrams manually
- Transient views appear to be a promising means for visualizing models or model excerpts
- Form a means for browsing and even modifying models
- Currently also investigated:
  - Synthesis of transient views in a generative way
  - General treatment of diagram labels
  - Smart layout configuration
- `http://www.informatik.uni-kiel.de/rtsys/kieler/`

# Conclusion & Further Work

- Lots of productivity wasted with drawing diagrams manually
- Transient views appear to be a promising means for visualizing models or model excerpts
- Form a means for browsing and even modifying models
- Currently also investigated:
  - Synthesis of transient views in a generative way
  - General treatment of diagram labels
  - Smart layout configuration
- `http://www.informatik.uni-kiel.de/rtsys/kieler/`

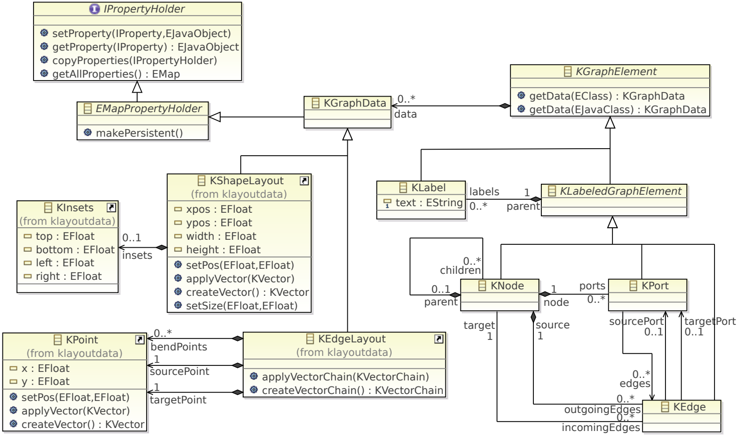
thanks!

questions or comments?

# Appendix

# Appendix – KGraph Meta Model

## with KLayoutData



*IPropertyHolder*
- setProperty(IProperty,EJavaObject)
- getProperty(IProperty) : EJavaObject
- copyProperties(IPropertyHolder)
- getAllProperties() : EMap

*EMapPropertyHolder*
- makePersistent()

KGraphData

*KGraphElement*
- getData(EClass) : KGraphData
- getData(EJavaClass) : KGraphData

KShapeLayout (from klayoutdata)
- xpos : EFloat
- ypos : EFloat
- width : EFloat
- height : EFloat
- setPos(EFloat,EFloat)
- applyVector(KVector)
- createVector() : KVector
- setSize(EFloat,EFloat)

KInsets (from klayoutdata)
- top : EFloat
- bottom : EFloat
- left : EFloat
- right : EFloat

KLabel
- text : EString

*KLabeledGraphElement*

KNode

KPort

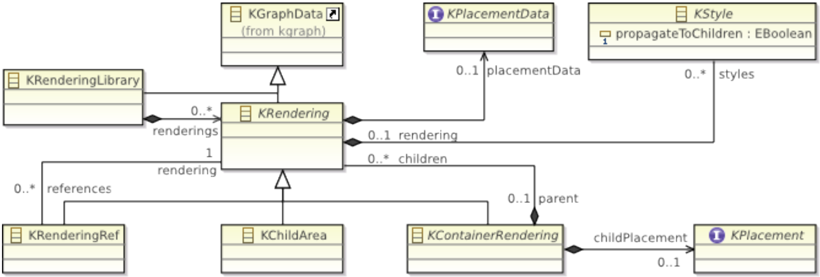
KEdge

KPoint (from klayoutdata)
- x : EFloat
- y : EFloat
- setPos(EFloat,EFloat)
- applyVector(KVector)
- createVector() : KVector

KEdgeLayout (from klayoutdata)
- applyVectorChain(KVectorChain)
- createVectorChain() : KVectorChain

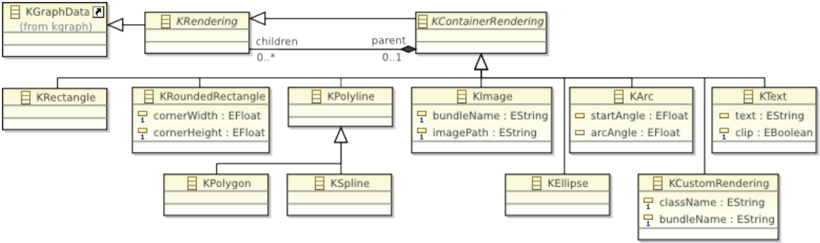
# Appendix – KRendering Meta Model 1/3

## Core Elements



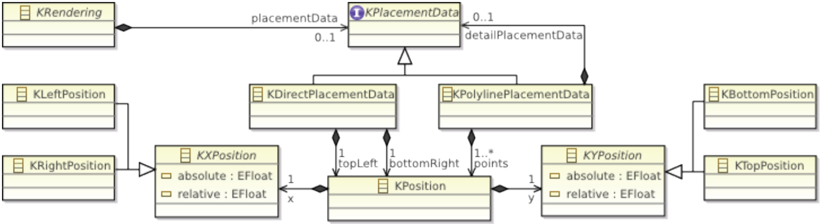
# Appendix – KRendering Meta Model 2/3

## Rendering Primitives

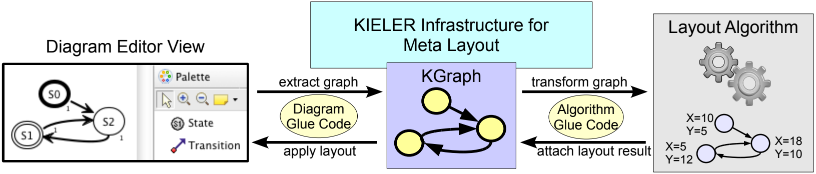


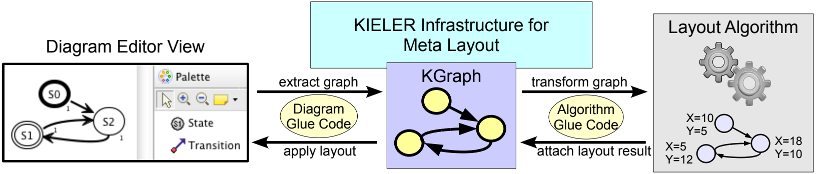
# Appendix – KRendering Meta Model 3/3

## Declarative Micro Layout Definitions (Excerpt)



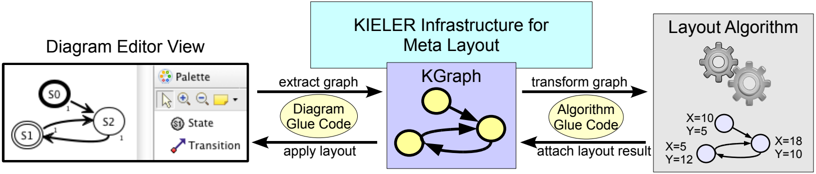
# Infrastructure for Meta Layout





- GMF
- Graphiti
- Papyrus
- Yakindu

- Graphviz (Dot, Neato, FDP, Twopi, Circo)
- Open Graph Drawing Framework (OGDF) (Layer-based, Planarization, Force-directed)
- Own Implementations (Data flow diagrams)



- GMF
- Graphiti
- Papyrus
- Yakindu

- Graphviz (Dot, Neato, FDP, Twopi, Circo)
- Open Graph Drawing Framework (OGDF) (Layer-based, Planarization, Force-directed)
- Own Implementations (Data flow diagrams)

H. Fuhrmann and R. v. Hanxleden, Taming Graphical Modeling (MoDELS'10)

# KRendering Diagram Description – Example

```
KNode { /* the diagram */
  KShapeLayout {
    algorithm = "ogdf.planarization",
    direction = UP, spacing = 75.0
  }
  KNode { /* "KColor" figure */
    KShapeLayout {
       width 180.0 height 80.0
    }
    Rectangle {
      lineWidth 2, backgroundColor 255 250 205
      Text "KColor" {
        bold, fontSize 20,
        backgroundColor 255 250 205
      }
    }
  }
  KNode { /* "KBackgroundColor" figure */
    KShapeLayout {
      width 242.0 height 80.0
    }
    Rectangle {
      lineWidth 2, backgroundColor 255 250 205
      Text "KBackgroundColor" {
         bold, fontSize 20,
         backgroundColor 255 250 205
      }
    }
  }
```

```
  --> "//children.0" {
    KEdgeLayout {
       edgeType = GENERALIZATION
    }
    Polyline  {
      lineWidth 2
      Polygon {
        lineWidth 2, backgroundColor 255 255 255
        polylinePlacementData {
          points:
            left  0.0 0.0 / top 0.0 0.0,
            right  0.0 0.0 / top 0.0 0.5,
            left  0.0 0.0 / bottom 0.0 0.0
          detailedPlacementData:
            decoratorPlacementData {
              relative,  location  1.0,
              xOffset  -35, yOffset -17.5,
              width 35, height 35
            }
        }
      }
    }
  }
  KNode { /* "KForegroundColor" figure */
    KShapeLayout {
      width 242.0 height 80.0
    }
  ...
}
```

# Appendix – Ecore Mapping in Xtend 1/3

```
Inject
extension KRenderingUtil

Inject
extension KRenderingColors

override KNode transform(EModelElementCollection model, TransformationContext<
      EModelElementCollection, KNode> transformationContext) {

  val rootNode = KimlUtil::createInitializedNode;
  rootNode.KShapeLayout.setProperty(LayoutOptions::ALGORITHM, "de.cau.cs.kieler.kiml.ogdf.
        planarization");
  rootNode.KShapeLayout.setProperty(LayoutOptions::SPACING, 50.float);
  rootNode.KShapeLayout.setProperty(LayoutOptions::DIRECTION, Direction::UP);

  val classifier = model.filter(typeof(EClassifier)).toList;
  classifier.createClassifierFigures(rootNode);
  classifier.createAssociationConnections;
  classifier.createInheritanceConnections;

  model.filter(typeof(EPackage)).forEach[
    val classifiers = it.EClassifiers;
    classifiers.createClassifierFigures(rootNode);
    classifiers.createAssociationConnections;
    classifiers.createInheritanceConnections;
  ];

  return rootNode;
}
```

# Appendix – Ecore Mapping in Xtend 2/3

```
def createClassifierFigures(Iterable<EClassifier> classes, KNode rootNode) {
  classes.forEach[
    val boxWidth = if (it.name.length < 10) 180 else it.name.length*12+50;
    val classNode = it.createRectangulareNode(80, boxWidth);
    classNode.KRendering.add(
    factory.createKText.of(it.name).add(factory.createKFontSize.of(20))
      .add(factory.createKFontBold.setbold).add("lemon".bgColor)
    );
    classNode.KRendering.add(factory.createKLineWidth.of(2)).add("lemon".bgColor);
    rootNode.children.add(classNode);
  ];
}

def createAssociationConnections(Iterable<EClassifier> classes) {
  val list = classes.toList;
  list.filter(typeof(EClass)).forEach[
    it.EStructuralFeatures.filter(typeof(EReference))
      .filter[list.contains(it.EType)]
      .forEach[it.createAssociationConnection];
  ];
}

def createAssociationConnection(EReference ref) {
  val edge = ref.createPolyLineEdge;
  edge.KRendering.add(factory.createKLineWidth.of(2));
  (edge.KRendering as KPolyline).addConnectionArrow(2, true);
  edge.source = ref.eContainer.node;
  edge.target = ref.EType.node;
  ref.eContainer.node.outgoingEdges.add(edge);
}
```

# Appendix – Ecore Mapping in Xtend 3/3

```
def createInheritanceConnections(Iterable<EClassifier> classes) {
  val list = classes.toList;
  list.filter(typeof(EClass)).forEach[
    child | child.ESuperTypes.filter[ list.contains(it) ]
      .forEach[ parent | child.createInheritanceConnection(parent) ];
  ];
}

def createInheritanceConnection(EClass child, EClass parent) {
  val edge = new Pair(child, parent).createPolyLineEdge;
  val line = edge.KRendering as KPolyline
  edge.KEdgeLayout.setProperty(LayoutOptions::EDGE_TYPE, EdgeType::GENERALIZATION)
  line.add(factory.createKLineWidth.of(2))
  line.addInheritanceConnectionArrow(2, true);
  edge.source = child.node;
  edge.target = parent.node;
  child.node.outgoingEdges.add(edge);
}
```