Lightweight Diagrams (KLighD)

KLighD has been moved to GitHub.

Project Overview

Related Publications:

- Christian Schneider, Miro Spönemann, and Reinhard von Hanxleden. *Just Model!* – Putting Automatic Synthesis of Node-Link-Diagrams into Practice. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'13)*, San Jose, CA, USA, 15–19 September 2013. Original publication is available via IEEE Xplore® (pdf / bib / poster).
- Christian Schneider, Miro Spönemann, and Reinhard von Hanxleden. Transient view generation in Eclipse. In *Proceedings of the First Workshop on Academics Modeling with Eclipse (AcME)*, Kgs. Lyngby, Denmark, 2012. Original publication is available in the joint proceedings of the co-located events on the website of the 8th European Conference on Modelling Foundations and Applications (ECMFA'12) (pdf / bib).

Get the **demo example** DSL and KLighD diagram synthesis shown at **XtextCon 2015** as well as the test model from here!

This page has been updated to KIELER Pragmatics 0.11.0

Contents

The Big Picture

The KIELER Lightweight Diagrams (KLighD) project aims at offering transient lightweight representations of models or parts of them, without incorporating complex editing facilities like graphical editors.
Instead graphical or textual representations are to be synthesized from a chosen fraction of a model base and dismissed if they are not needed anymore. This way the Model-View-Controller paradigm (MVC) shall be established at the users' front end of modeling tools. Although KLighD is intended to address graphical as well as textual transient views, the graphical ones are currently in the focus. The automatic arrangement of those views (macro layout) is contributed by KIML.

Feel free to watch the following videos illustrating the idea and possibilities of transient views of models:

- http://rtsys.informatik.uni-kiel.de/~kieler/videos/klighd/InstantModelBrowsingAnnotated.mp4
- http://rtsys.informatik.uni-kiel.de/~kieler/videos/klighd/TextualModeling.mp4
- http://rtsys.informatik.uni-kiel.de/~kieler/videos/klighd/SCT.mp4 (textual modeling continued)
- http://rtsys.informatik.uni-kiel.de/~kieler/videos/klighd/SCG.mp4
- http://rtsys.informatik.uni-kiel.de/~kieler/videos/ptolemyViewer/PtolemyViewerHQ.mp4

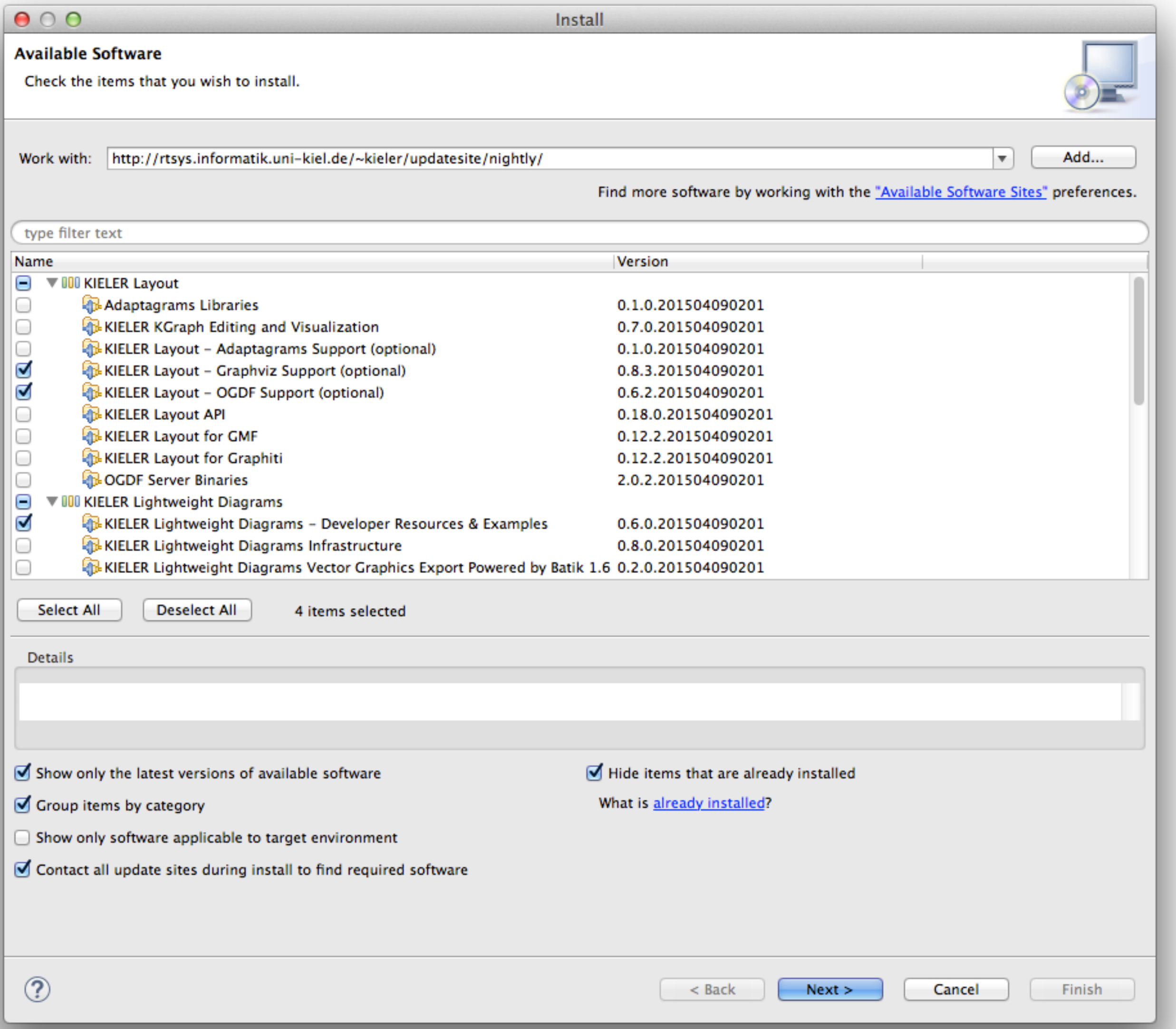
For details on the objectives of our KLighD framework have a look at the above mentioned publication *Just Model! – Putting Automatic Synthesis of Node-Link-Diagrams into Practice* (Sec. I, III, & V) first.

Getting Started

Installation

In order to build up your own diagrams by means of KLighD we suggest to install some of our provided features from our update site http://rtsys.informatik.uni-kiel.de/~kieler/updatesite/:

- KIELER Lightweight Diagrams – Developer Resources & Examples:
  This feature includes KLighD's & KIML's runtime as well as a few developer tools (e.g. a convenient project wizard) and a few prepared example diagram synthesis implementations. Besides, all source bundles will be installed.
  Note: *KIELER Lightweight Diagrams Demo Examples* is part of an older release as indicated by the timestamp in the version number. It is superseded by *Developer Resources & Examples.*
- KIELER Layout – Graphviz support (optional):
  This feature enables the employment of Graphviz' layouters, `dot` is very popular for simple graphs with edge labels. Note that employing Graphviz requires to also install the native library, see http://www.graphviz.org/
- KIELER Layout – OGDF support (optional):
  This feature enables the employment of OGDF' layouters, its `planarization` is the recommended algorithm for arranging UML Class Diagrams. The required OGDF Server Binaries will be automatically installed, too!


Install

Available Software

Check the items that you wish to install.

Work with: http://rtsys.informatik.uni-kiel.de/~kieler/updatesite/nightly/ [Add...]

Find more software by working with the "Available Software Sites" preferences.

type filter text

	Name	Version
▣	▼ KIELER Layout	
☐	Adaptagrams Libraries	0.1.0.201504090201
☐	KIELER KGraph Editing and Visualization	0.7.0.201504090201
☐	KIELER Layout – Adaptagrams Support (optional)	0.1.0.201504090201
✓	KIELER Layout – Graphviz Support (optional)	0.8.3.201504090201
✓	KIELER Layout – OGDF Support (optional)	0.6.2.201504090201
☐	KIELER Layout API	0.18.0.201504090201
☐	KIELER Layout for GMF	0.12.2.201504090201
☐	KIELER Layout for Graphiti	0.12.2.201504090201
☐	OGDF Server Binaries	2.0.2.201504090201
▣	▼ KIELER Lightweight Diagrams	
✓	KIELER Lightweight Diagrams – Developer Resources & Examples	0.6.0.201504090201
☐	KIELER Lightweight Diagrams Infrastructure	0.8.0.201504090201
☐	KIELER Lightweight Diagrams Vector Graphics Export Powered by Batik 1.6	0.2.0.201504090201

[Select All] [Deselect All] 4 items selected

Details

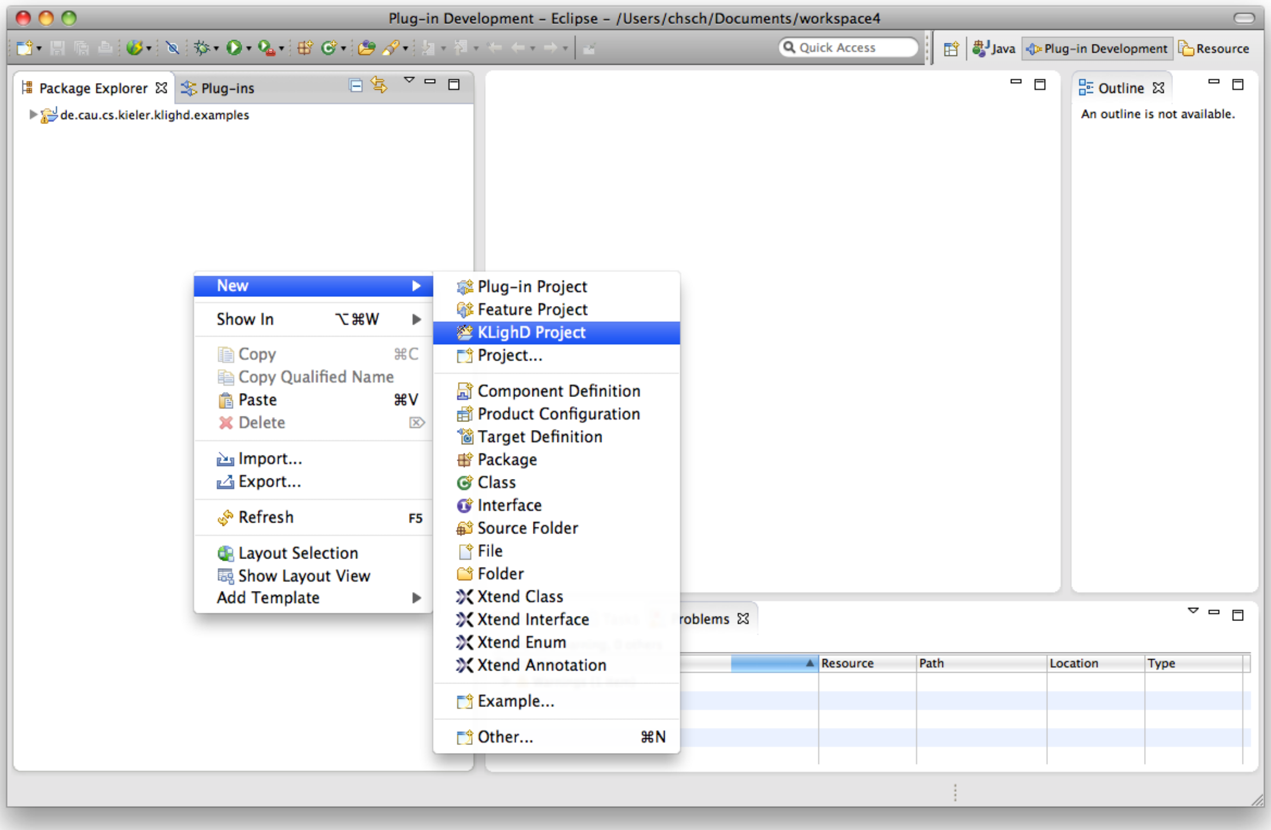
- ✓ Show only the latest versions of available software
- ✓ Group items by category
- ☐ Show only software applicable to target environment
- ✓ Contact all update sites during install to find required software
- ✓ Hide items that are already installed

What is already installed?

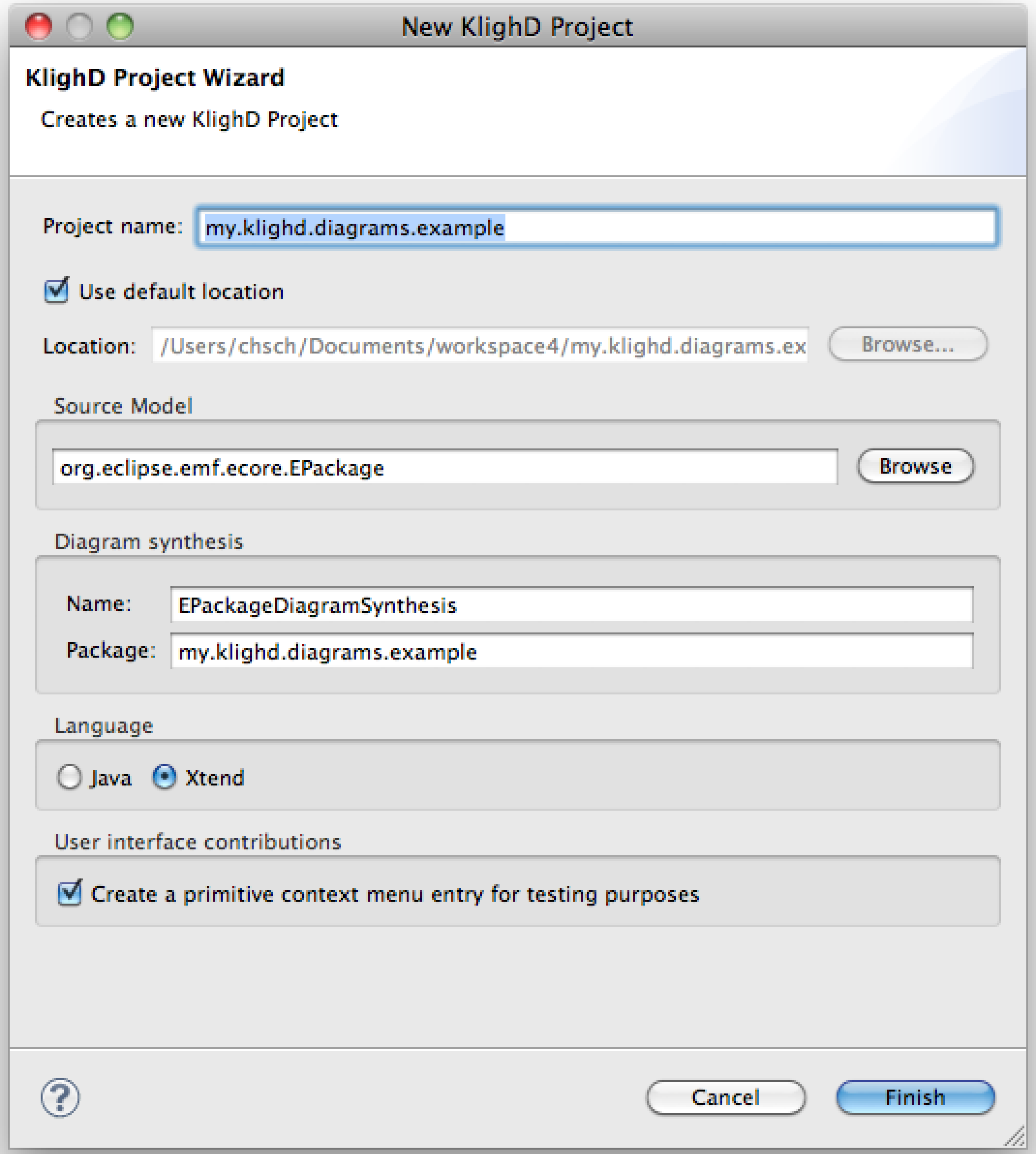
[< Back] [Next >] [Cancel] [Finish]

Create first diagrams

In order to let the framework draw your desired diagrams for representing given data we need to tell KLighD how to obtain nodes, edges, and hierarchy levels from the provided data. To this end we need to implement a so called *diagram synthesis*. Getting an initial stub of such an implementation is most easiest done by means of our KLighD *project wizard* (if the entry is not visible, switch the perspective or close and reopen it, or simply choose Other... KIELER Light-weight diagrams):



Provide the source model type, e.g. via the 'Browse' button. Note that the project containing that class must exist in the workspace or at least any other workspace project must have this class in its class path, e.g. via a *bundle dependency*. Otherwise it won't be offered by the list of available classes. Of course you might alter the project's name, the name of the diagram synthesis implementation and its containing package. Finally choose your favorite implementation language - we like Xtend very much for this purpose! 🙂 If you don't like to have a simple popup menu contribution for testing purposes, uncheck the corresponding checkbox.


New KlighD Project

KlighD Project Wizard

Creates a new KlighD Project

Project name: my.klighd.diagrams.example

[✓] Use default location

Location: /Users/chsch/Documents/workspace4/my.klighd.diagrams.ex

Source Model

org.eclipse.emf.ecore.EPackage

Diagram synthesis

Name: EPackageDiagramSynthesis

Package: my.klighd.diagrams.example

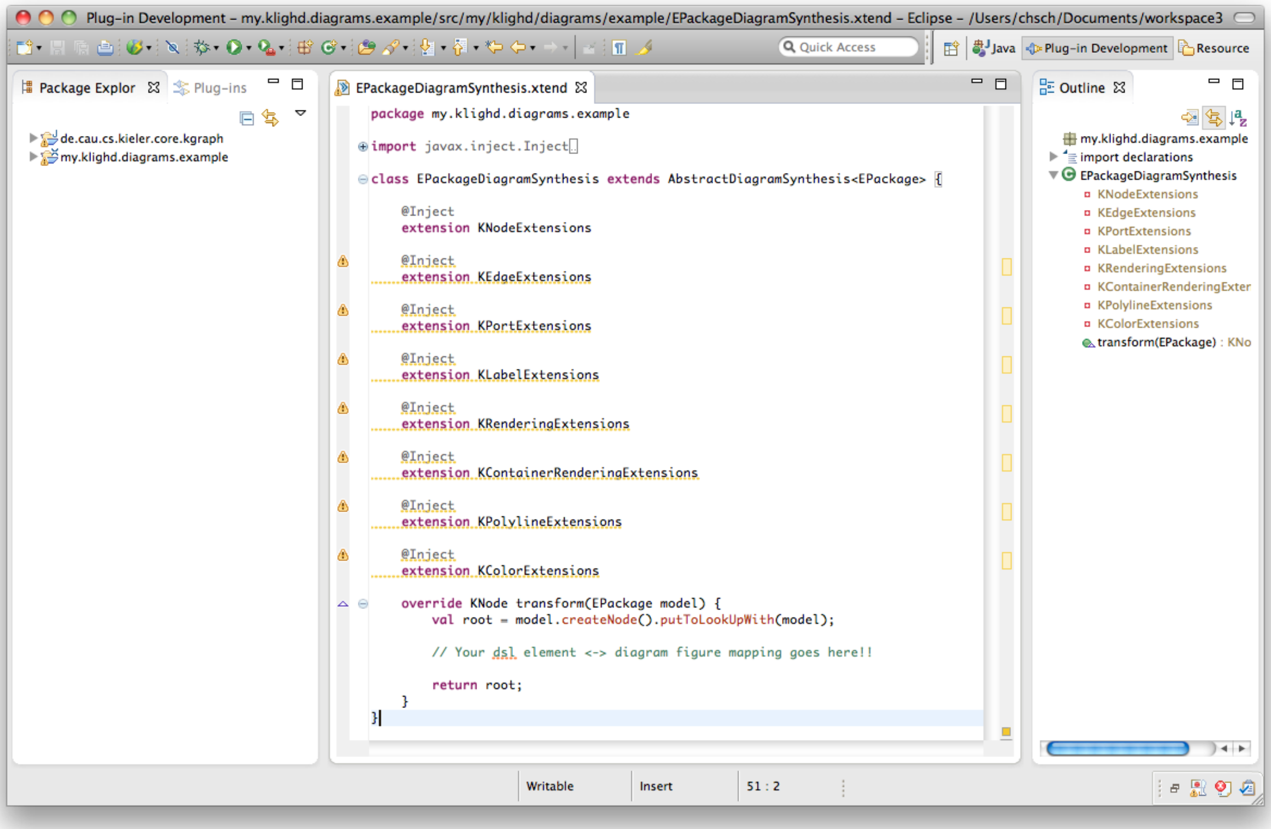
Language

Java [✓ Xtend]

User interface contributions

[✓] Create a primitive context menu entry for testing purposes

You will get the following implementation template, it is already registered via KLighD's dedicated extension point (see the project's plugin.xml).

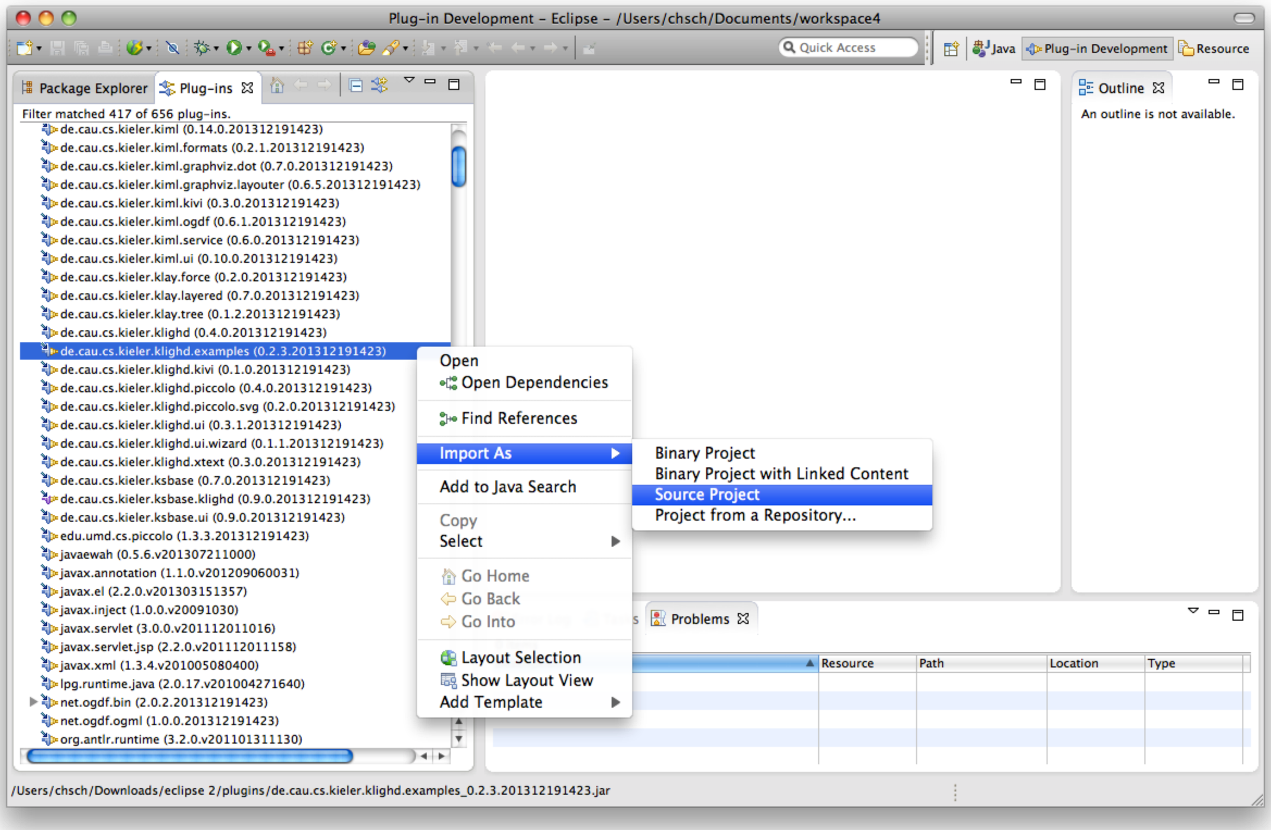

Plug-in Development – my.klighd.diagrams.example/src/my/klighd/diagrams/example/EPackageDiagramSynthesis.xtend – Eclipse – /Users/chsch/Documents/workspace3
Quick Access
Java
Plug-in Development
Resource
Package Explor
Plug-ins
de.cau.cs.kieler.core.kgraph
my.klighd.diagrams.example
EPackageDiagramSynthesis.xtend
Outline
my.klighd.diagrams.example
import declarations
EPackageDiagramSynthesis
KNodeExtensions
KEdgeExtensions
KPortExtensions
KLabelExtensions
KRenderingExtensions
KContainerRenderingExter
KPolylineExtensions
KColorExtensions
transform(EPackage) : KNo
Writable
Insert
51 : 2

```
package my.klighd.diagrams.example

import javax.inject.Inject

class EPackageDiagramSynthesis extends AbstractDiagramSynthesis<EPackage> {

    @Inject
    extension KNodeExtensions

    @Inject
    extension KEdgeExtensions

    @Inject
    extension KPortExtensions

    @Inject
    extension KLabelExtensions

    @Inject
    extension KRenderingExtensions

    @Inject
    extension KContainerRenderingExtensions

    @Inject
    extension KPolylineExtensions

    @Inject
    extension KColorExtensions

    override KNode transform(EPackage model) {
        val root = model.createNode().putToLookUpWith(model);

        // Your dsl element <-> diagram figure mapping goes here!!

        return root;
    }
}
```

Before starting with the implementation let us point you at some hints on using Xtend!

Create a `KNode` (via `createNode()`) for each element comprised by the given instance `model` (usually called the business, domain, or semantic model) of your data type, here `EPackage`. Add those nodes to the children of `root`. Create a `KEdge` for each relation or link to be contained in your diagram. Similarly to `createNode()` there is `createEdge()` for that purpose. Set source and target node of those edges accordingly, this will implicitly add the edges to the diagram (via EMF's `EOpposite` mechanism). You can reveal an already created node representing a certain domain element by calling *domainElement*.`getNode()`, or simply *domainElement*.node. Have a look at UML2UseCaseDiagramSynthesis, which is part of our examples project, on how to do that. If you are interested in diagrams with nodes and edges connected via ports (as depicted in Figure 2 in *Just Model! - ...*) have a look at CircuitDiagramSynthesis. For convenience you might also want to import the examples project into your workspace. Simply switch to the *Plug-ins* view and import `de.cau.cs.kieler.klighd.examples` as *Source Project*, see screen shot below.

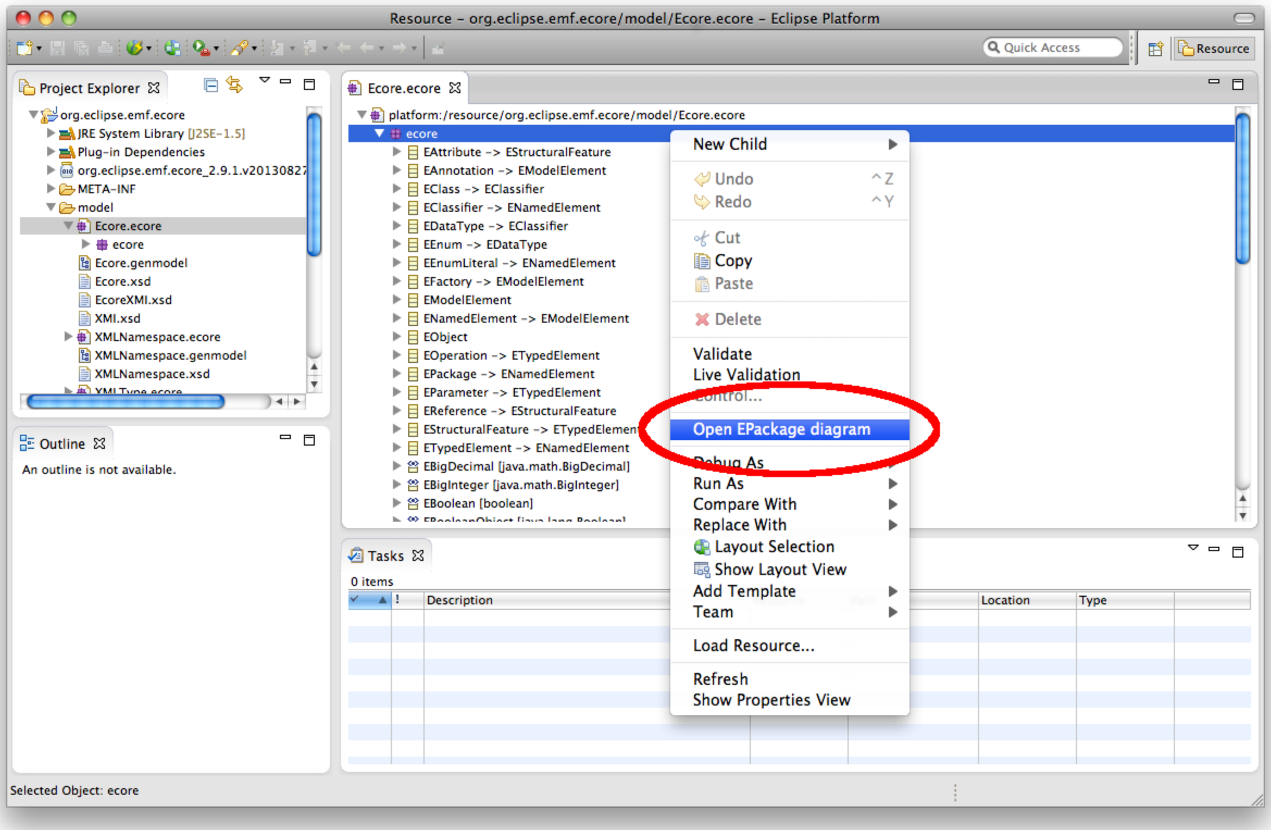
Finally attach figure specifications to the nodes and edges, most conveniently by means of the extension methods provided by the `...Extension` classes - see the use case example on that, too. A documentation of those methods in form of a list of all extension methods sorted by the type they can be used with can be found here. (And yes, there is still some lack of documentation...)



```
Filter matched 417 of 656 plug-ins.
de.cau.cs.kieler.kiml (0.14.0.201312191423)
de.cau.cs.kieler.kiml.formats (0.2.1.201312191423)
de.cau.cs.kieler.kiml.graphviz.dot (0.7.0.201312191423)
de.cau.cs.kieler.kiml.graphviz.layouter (0.6.5.201312191423)
de.cau.cs.kieler.kiml.kivi (0.3.0.201312191423)
de.cau.cs.kieler.kiml.ogdf (0.6.1.201312191423)
de.cau.cs.kieler.kiml.service (0.6.0.201312191423)
de.cau.cs.kieler.kiml.ui (0.10.0.201312191423)
de.cau.cs.kieler.klay.force (0.2.0.201312191423)
de.cau.cs.kieler.klay.layered (0.7.0.201312191423)
de.cau.cs.kieler.klay.tree (0.1.2.201312191423)
de.cau.cs.kieler.klighd (0.4.0.201312191423)
de.cau.cs.kieler.klighd.examples (0.2.3.201312191423)
de.cau.cs.kieler.klighd.kivi (0.1.0.201312191423)
de.cau.cs.kieler.klighd.piccolo (0.4.0.201312191423)
de.cau.cs.kieler.klighd.piccolo.svg (0.2.0.201312191423)
de.cau.cs.kieler.klighd.ui (0.3.1.201312191423)
de.cau.cs.kieler.klighd.ui.wizard (0.1.1.201312191423)
de.cau.cs.kieler.klighd.xtext (0.3.0.201312191423)
de.cau.cs.kieler.ksbase (0.7.0.201312191423)
de.cau.cs.kieler.ksbase.klighd (0.9.0.201312191423)
de.cau.cs.kieler.ksbase.ui (0.9.0.201312191423)
edu.umd.cs.piccolo (1.3.3.201312191423)
javaewah (0.5.6.v201307211000)
javax.annotation (1.1.0.v201209060031)
javax.el (2.2.0.v201303151357)
javax.inject (1.0.0.v20091030)
javax.servlet (3.0.0.v201112011016)
javax.servlet.jsp (2.2.0.v201112011158)
javax.xml (1.3.4.v201005080400)
lpg.runtime.java (2.0.17.v201004271640)
net.ogdf.bin (2.0.2.201312191423)
net.ogdf.ogml (1.0.0.201312191423)
org.antlr.runtime (3.2.0.v201101311130)

/Users/chsch/Downloads/eclipse 2/plugins/de.cau.cs.kieler.klighd.examples_0.2.3.201312191423.jar
```

Invoking the diagram synthesis

Testing the diagram synthesis is most easiest done by means of the test menu contribution created by the project wizard. Thus, start a test *Eclipse Application* via *'Run As'*, e.g. via 'Run' in the main menu. Import or create a project with some of your model data. If your model format has been described by means of Ecore, you can open your model with a matching EMF tree editor as shown in the following screenshot (here we used the Ecore format itself, so we used an "arbitrary" Ecore model 😉).
Open the context menu while an instance of your source model type is selected, and hit the *Open ... diagram* entry. Btw.: The menu entry is enabled as far as the selected elements are of your source model type. Hence, there may be other ways (views or editors) to open the diagram beyond those EMF tree editors.



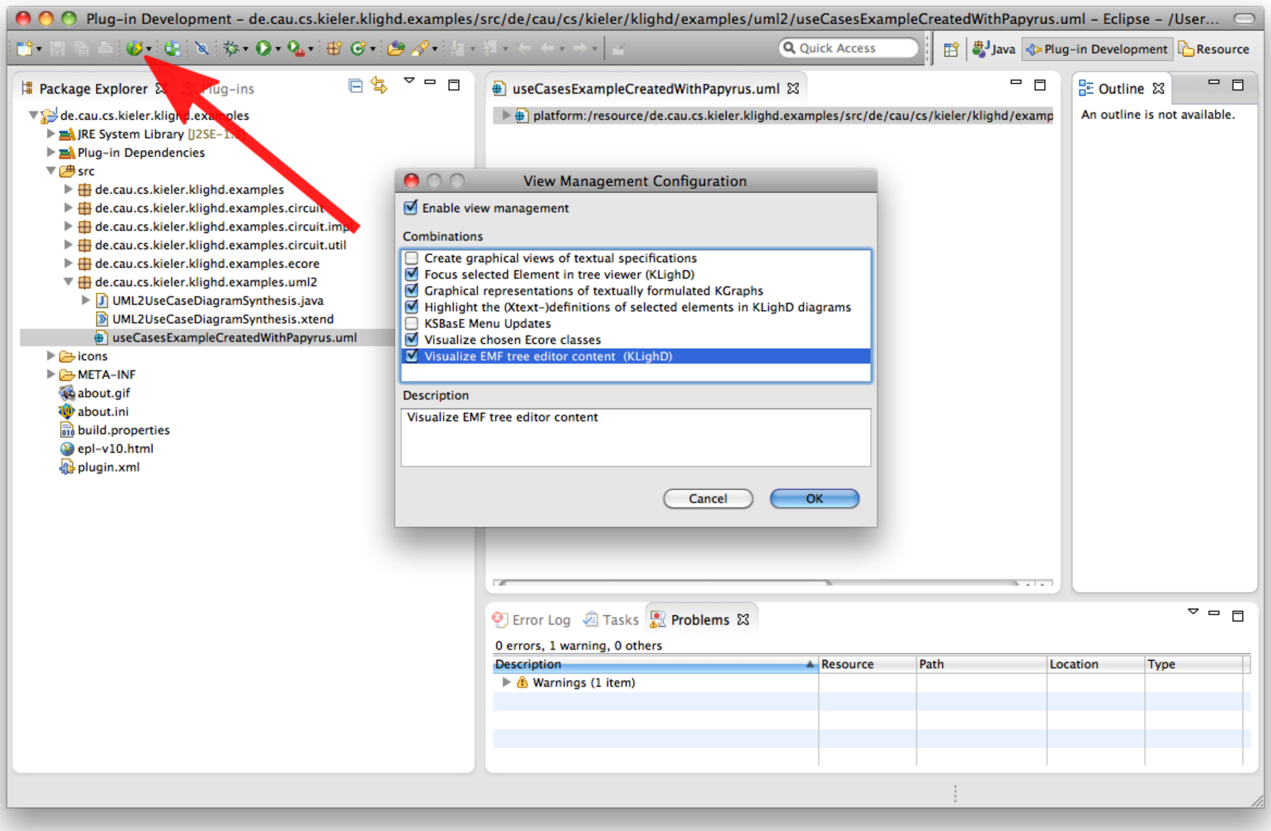
Alternatively, you might want to create your own menu contribution. Here are some instructions using the Eclipse 3.X API: Register a command with some arbitrary id and a default handler. In the `execute()` method of your handler call

```
DiagramViewManager.getInstance().createView(<<someId>>, <<someName>>, <<yourConcreteData>>,
KlighdSynthesisProperties.emptyConfig());
```

This statements will open an eclipse view part showing your desired diagram. `<<someId>>` is an the Id that can be used to access, update, or close the view part later on programmatically, and `<<someName>>` is the name of the view (shown at the view's tab), `<<yourConcreteData>>` is an instance of the input type of your diagram synthesis implementation. Via the last parameter some additional instructions can be handed over. For those that are not yet familiar with Eclipse menu stuff have a look on at a corresponding tutorial, e.g. at vogella.com.

Exemplary diagram syntheses

Exemplary use case diagrams (created by the UML2UseCaseDiagramSynthesis) and circuit diagrams (created by the CircuitDiagramSynthesis) are shown in following screenshots, the source UML & Circuit models are part of the examples project, too. In order to obtain those diagrams enable the automatic diagram creation by activating *Visualize EMF tree editor content (KLighD)* as shown below.


Plug-in Development - de.cau.cs.kieler.klighd.examples/src/de/cau/cs/kieler/klighd/examples/uml2/useCasesExampleCreatedWithPapyrus.uml - Eclipse - /User...
Quick Access
Java
Plug-in Development
Resource
Package Explorer
Plug-ins
de.cau.cs.kieler.klighd.examples
JRE System Library [J2SE-1.5]
Plug-in Dependencies
src
de.cau.cs.kieler.klighd.examples
de.cau.cs.kieler.klighd.examples.circuit
de.cau.cs.kieler.klighd.examples.circuit.impl
de.cau.cs.kieler.klighd.examples.circuit.util
de.cau.cs.kieler.klighd.examples.ecore
de.cau.cs.kieler.klighd.examples.uml2
UML2UseCaseDiagramSynthesis.java
UML2UseCaseDiagramSynthesis.xtend
useCasesExampleCreatedWithPapyrus.uml
icons
META-INF
about.gif
about.ini
build.properties
epl-v10.html
plugin.xml
useCasesExampleCreatedWithPapyrus.uml
platform:/resource/de.cau.cs.kieler.klighd.examples/src/de/cau/cs/kieler/klighd/examp
Outline
An outline is not available.
Cancel
OK
Error Log
Tasks
Problems

View Management Configuration

☑ Enable view management

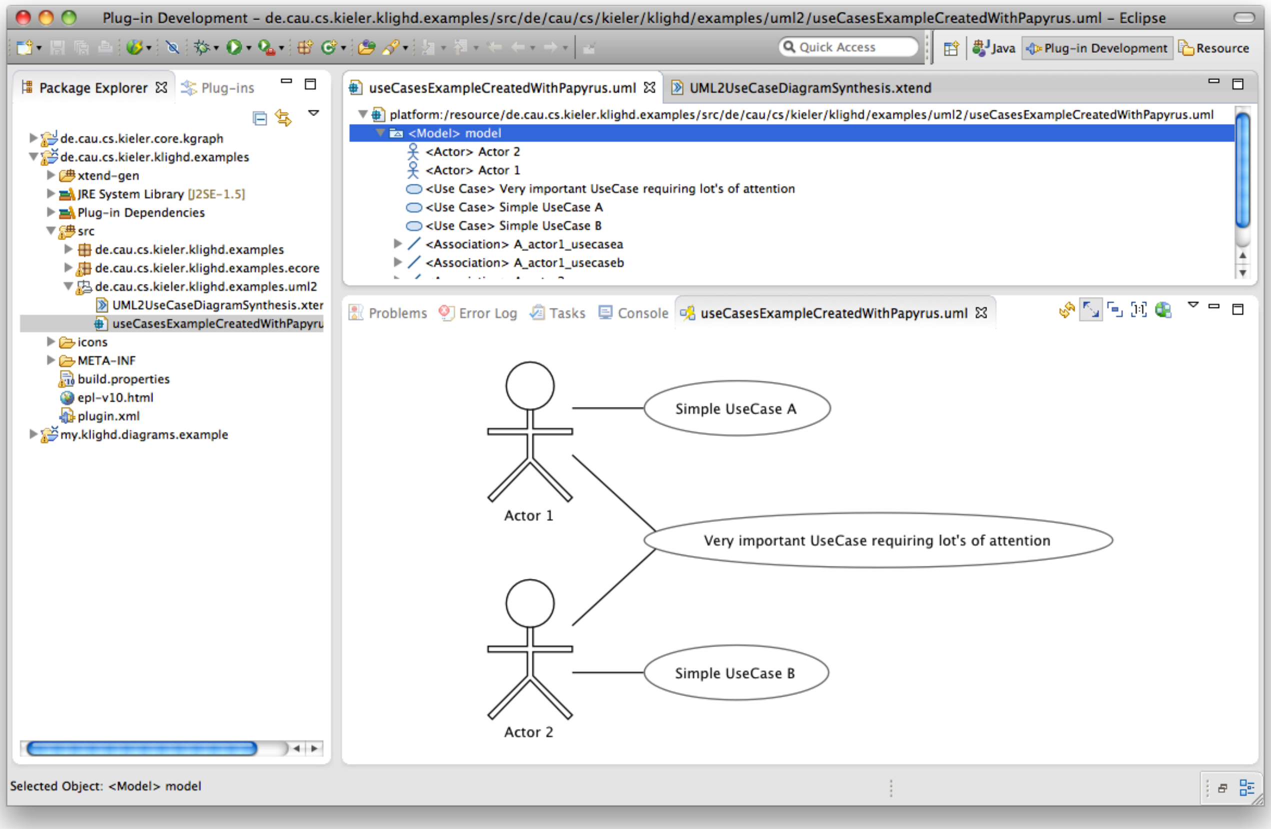
Combinations
- ☐ Create graphical views of textual specifications
- ☑ Focus selected Element in tree viewer (KLighD)
- ☑ Graphical representations of textually formulated KGraphs
- ☑ Highlight the (Xtext-)definitions of selected elements in KLighD diagrams
- ☐ KSBasE Menu Updates
- ☑ Visualize chosen Ecore classes
- ☑ Visualize EMF tree editor content (KLighD)

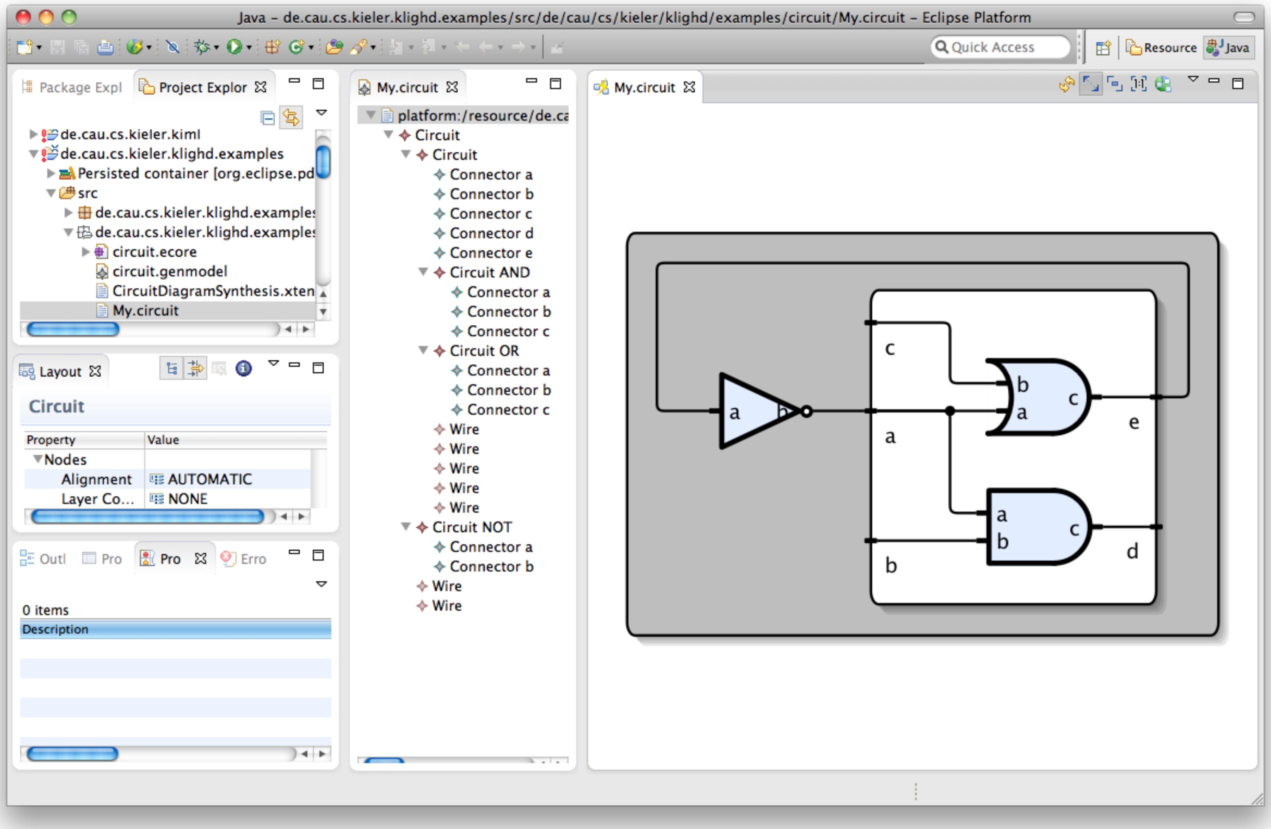
Description
Visualize EMF tree editor content

0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				

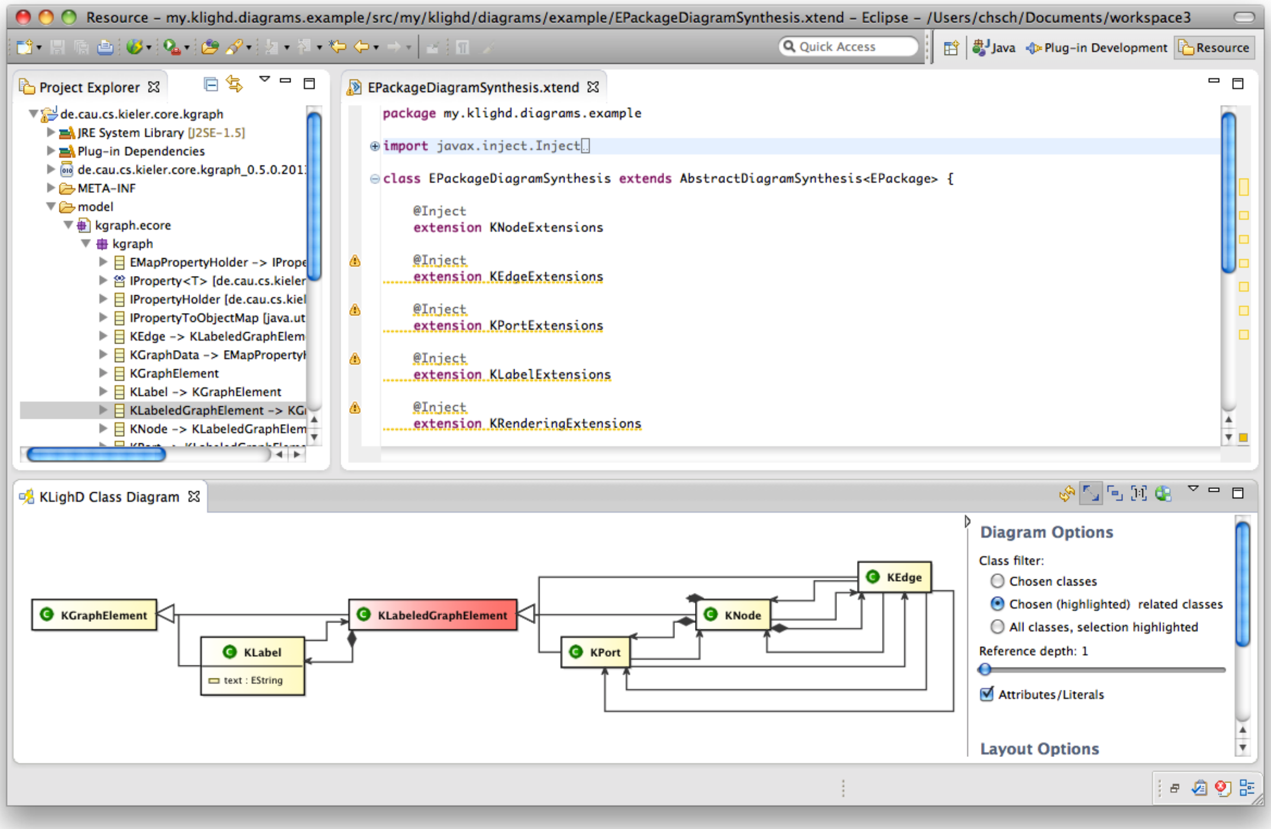
Then open the models and select one of its elements (open My.circuit with the *Sample Reflective Ecore Model Editor*).


Plug-in Development – de.cau.cs.kieler.klighd.examples/src/de/cau/cs/kieler/klighd/examples/uml2/useCasesExampleCreatedWithPapyrus.uml – Eclipse
Quick Access
Java
Plug-in Development
Resource
Package Explorer
Plug-ins
de.cau.cs.kieler.core.kgraph
de.cau.cs.kieler.klighd.examples
xtend-gen
JRE System Library [J2SE-1.5]
Plug-in Dependencies
src
de.cau.cs.kieler.klighd.examples
de.cau.cs.kieler.klighd.examples.ecore
de.cau.cs.kieler.klighd.examples.uml2
UML2UseCaseDiagramSynthesis.xter
useCasesExampleCreatedWithPapyru
icons
META-INF
build.properties
epl-v10.html
plugin.xml
my.klighd.diagrams.example
useCasesExampleCreatedWithPapyrus.uml
UML2UseCaseDiagramSynthesis.xtend
platform:/resource/de.cau.cs.kieler.klighd.examples/src/de/cau/cs/kieler/klighd/examples/uml2/useCasesExampleCreatedWithPapyrus.uml
<Model> model
<Actor> Actor 2
<Actor> Actor 1
<Use Case> Very important UseCase requiring lot's of attention
<Use Case> Simple UseCase A
<Use Case> Simple UseCase B
<Association> A_actor1_usecasea
<Association> A_actor1_usecaseb
Problems
Error Log
Tasks
Console
useCasesExampleCreatedWithPapyrus.uml
Simple UseCase A
Actor 1
Very important UseCase requiring lot's of attention
Simple UseCase B
Actor 2
Selected Object: <Model> model


Java – de.cau.cs.kieler.klighd.examples/src/de/cau/cs/kieler/klighd/examples/circuit/My.circuit – Eclipse Platform
Quick Access
Resource
Java
Package Expl
Project Explor
de.cau.cs.kieler.kiml
de.cau.cs.kieler.klighd.examples
Persisted container [org.eclipse.pd
src
de.cau.cs.kieler.klighd.example:
de.cau.cs.kieler.klighd.example:
circuit.ecore
circuit.genmodel
CircuitDiagramSynthesis.xten
My.circuit
Layout
Circuit
Property
Value
Nodes
Alignment
AUTOMATIC
Layer Co...
NONE
Outl
Pro
Pro
Erro
0 items
Description
My.circuit
platform:/resource/de.ca
Circuit
Circuit
Connector a
Connector b
Connector c
Connector d
Connector e
Circuit AND
Connector a
Connector b
Connector c
Circuit OR
Connector a
Connector b
Connector c
Wire
Wire
Wire
Wire
Wire
Circuit NOT
Connector a
Connector b
Wire
Wire
My.circuit
c
b
a
c
e
a
a
b
a
c
b
d

Advanced diagram synthesis features

Now you might want to use some advanced features of KLighD diagrams, like sophisticated figure styling or diagram synthesis options. Details on that will follow soon, the `EcoreDiagramSynthesis`, which is in charge of creating appealing class diagrams from Ecore models and which you will also find in the examples project, illustrates those features already. The following picture shows such a diagram and a list of options for tailoring such diagrams in the side bar on the right.


Resource - my.klighd.diagrams.example/src/my/klighd/diagrams/example/EPackageDiagramSynthesis.xtend - Eclipse - /Users/chsch/Documents/workspace3
Quick Access
Java
Plug-in Development
Resource
Project Explorer
de.cau.cs.kieler.core.kgraph
JRE System Library [J2SE-1.5]
Plug-in Dependencies
de.cau.cs.kieler.core.kgraph_0.5.0.201
META-INF
model
kgraph.ecore
kgraph
EMapPropertyHolder -> IPrope
IProperty<T> [de.cau.cs.kieler
IPropertyHolder [de.cau.cs.kiel
IPropertyToObjectMap [java.ut
KEdge -> KLabeledGraphElem
KGraphData -> EMapProperty
KGraphElement
KLabel -> KGraphElement
KLabeledGraphElement -> KGr
KNode -> KLabeledGraphElem
EPackageDiagramSynthesis.xtend
KLighD Class Diagram
KGraphElement
KLabel
text : EString
KLabeledGraphElement
KPort
KNode
KEdge

```
package my.klighd.diagrams.example

import javax.inject.Inject

class EPackageDiagramSynthesis extends AbstractDiagramSynthesis<EPackage> {

    @Inject
    extension KNodeExtensions

    @Inject
    extension KEdgeExtensions

    @Inject
    extension KPortExtensions

    @Inject
    extension KLabelExtensions

    @Inject
    extension KRenderingExtensions
```

Diagram Options

Class filter:

- Chosen classes
- [✓ Chosen (highlighted) related classes]
- All classes, selection highlighted

Reference depth: 1

[✓ Attributes/Literals]

Layout Options

Trouble shooting

- Be sure to compile your Eclipse plug-in project with Java 5 or Java 6 compatibility.
- Increase the `perm memory` size with the VM argument `-XX:MaxPermSize=128M` (*Arguments* tab of your Eclipse run configuration) if you get a `PermGen space` exception.

Incorporated Technologies

- The KRendering Notation Model
- Infrastructure for Meta Layout (KIML)
- Piccolo 2D