Setup and Documentation

First step is to set up an Eclipe to develop KIELER and the KEITH Language Server in. You might need both to test all features on both platforms.

If you find anything in the Setup guides that seems unclear or is wrong feel free to contact us.

Set up KEITH and KIELER development Eclipse

Follow the Running KEITH guide and select only KIELER semantics (you don't need ELK). Be sure to select the **keith** stream.

Follow the guide to set up an Eclipse to develop the language server and set up Keith.

KEITH does not work on Mac yet. If you only have a Mac you are only able to develop for KIELER.

Developing with other use cases

The scope of this project is not only to work with examples of SCCharts diagrams, but also other current uses cases, where bigger diagrams may be improved with new concepts from this project.

One of these use cases is the *C to Dataflow* extraction to view the flow of data in C programs as SCCharts dataflow. As this feature is still under development and is not yet merged into the main branches, simply check out the *nre/df* branch in the semantics repository. Opening a C program in the editor and compiling that program in the KIELER Compiler widget via the *C to Dataflow* compilation chain (in KEITH: press F1 and type *>Kicool: Compile model with C to Dataflow*) to create an SCChart model that will be visualized in the diagram widget.

Another use case is the visualization of models defining the structure of OSGi models, see OSGiViz.
To use this, clone the OSGiViz repository and import all existing projects of that repository into your Eclipse installation. When building, not all of these plugins will be able to be build and show errors. That is because Some dependencies are not set up for running this in your current setup. For that, first download the *org.eclipse.emf.ecore.xcore.lib_1.4.0.v20190401-0856.jar* file from this site and store it at a location, where you don't accidentally delete it again while developing for this project. Then in Eclipse go to *WindowPreferencesPlug-in DevelopmentTarget Platform* and click on the currently active Keith platform and click *Edit...* There, click on *Add...* and *Installation* and browse folder containing the previously downloaded jar. Click finish everywhere. Now you can set up a run configuration similar to launching the Semantics Language Server, but choose the project *de.cau.cs.kieler.osgiviz.language.server* and the main class *de.cau.cs.kieler.osgiviz.language.server.OsgivizLanguageServer* and add the new plugins to the classpath. Lastly, go to the client code in *keith-language* in *common/index.ts* and add these two lines into the *languageDescriptions* array to support the OSGi model languages in KEITH:

```
    {id: "model", name: "OSGi Model"},
    {id: "osgiviz", name: "OsgiViz Model"},
```

The diagram may not show anything for the provided OSGi models for you yet, as something in this setup is not quite correct. I did not find the root cause of it, but a workaround to display the diagram anyways:

In the de.cau.cs.kieler.klighd.lsp plugin, navigate to the KGraphDiagramUpdater to line 152 and just remove the if statement / replace it with an if(true). Now OSGi models (and other models with 'errors') are still visualized, which is all we want.

Repositories that you might need

Repo	Location	Contents
KIELER Semantics	https://git.rtsys.informatik.uni-kiel.de/projects/KIELER/repos/semantics/browse	Languages such as SCCharts, Esterel, Lustre as well as compilation and simulation tools as well as the semantic language server for KEITH. In here is the AdaptiveZoom class. You might not want to change much in there. It is mostly to show what approaches already exist.
KIELER Pragmatics	https://git.rtsys.informatik.uni-kiel.de/projects/KIELER/repos/pragmatics/browse	The KGraph synthesis.
KlighD	https://github.com/kieler/KLighD	The KGraph language, the pragmatics language server, and the KIELER lightweight diagram framework.
KEITH	https://git.rtsys.informatik.uni-kiel.de/projects/KIELER/repos/keith/browse	Typescript client that connects to the language server. Renders diagram via Sprotty.

Theia Documentation

https://github.com/eclipse-theia/theia/blob/master/doc/Developing.md

General Idea of a Language Server

https://microsoft.github.io/language-server-protocol/overviews/lsp/overview/

Our diagram server

https://github.com/eclipse/sprotty/wiki/Client-Server-Protocol

https://github.com/eclipse/sprotty/wiki/Architectural-Overview

Ticket Management

If you encounter any problems in KIELER, KEITH, or somewhere else in the used technology feel free to make a ticket in our Jira (or an issue in Github for KLighD).

Make sure to label your issue appropriately and add the right components to it. Otherwise we might not see it or people not responsible for it get assigned a ticket.