Kieler Compiler

Deprecated since 0.13

This article is deprecated. The described features are no longer available in current releases.

Project Overview

Responsible:

- Christian Motika, Steven Smyth

Related Theses:

- none yet

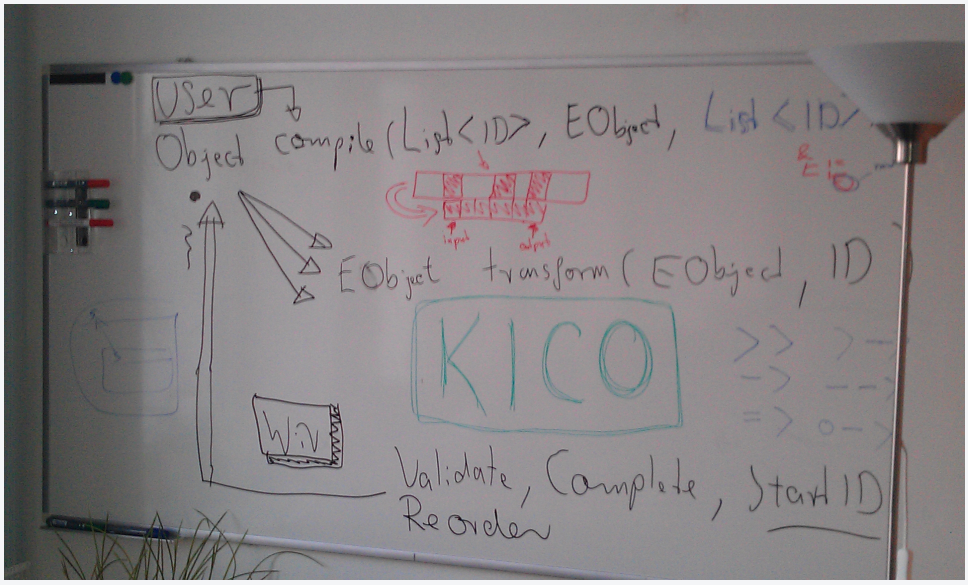
To see the KielerCompiler in action, we provide an Online SCCharts Compiler here.

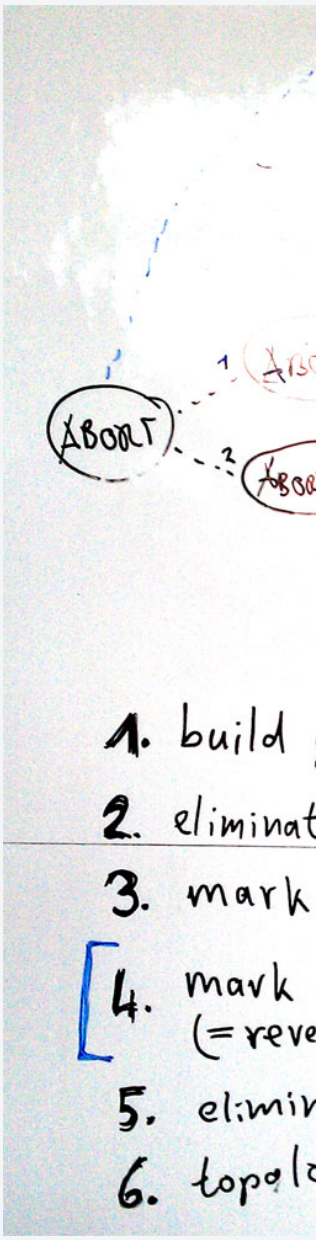
Kieler Compiler (KiCo)

In order to integrate and be able to evaluate our compiler chain from SCCharts to C or VHDL code we use the KiCo project as a generic framework that allows to register setp-by-step transformations on EObjects. These can then be handled by the generic KIEM KiCo DataComponent. **To see the KielerCompiler in action, we provide an Online SCCharts Compiler here.**

General

The KIELER Compiler (KiCo) project allows to register step-by-step model transformations on EObjects that could be written in Xtend or Java. These transformations are registered using an extension point provided (see below). After registering transformations these can be used by simply call the KielerCompiler compilation method as also explained further below.





Extension Point

In order to add a transformation to KiCo you must follow these steps:

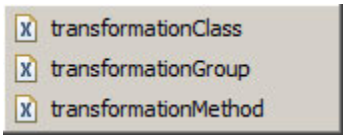
1. Add dependency to

   `de.cau.cs.kieler.kico`

2. Add the extension

   `de.cau.cs.kieler.kico.transformation`

3. Add one of the following extension element



Extension Element	Description
transformationClass	The defined class must extend "de.cau.cs.kieler.kico.Transformation" and must implement the methods defined in "de.cau.cs.kieler.kico.ITransformation". These are • getId(): Returns a unique String ID for this transformation. This is the ID the transformation will be referenced throughout KiCo. • getName(): Optionally return a String name for this transformation. If null is returned here the ID will be used as a name. • getDependencies(): Optionally return a List<String> of other transformation IDs that must run BEFORE this transformation. If null is returned then this means there are no dependencies. • transform(EObject): Returns an EObject and does the central transformation.
transformationMethod	The defined class can be freely chosen and does not need to extend or implement any other class or interface. Although you have to give more information in the extension element now: • class: The class where the transform method is implemented in • method: The name of the transformation method. Its signature must ensure that it returns an EObject an only take an EObject argument. Otherwise it cannot be found by KiCo. • id: The unique String ID for this transformation. This is the ID the transformation will be referenced throughout KiCo. • name: An optional String name for this transformation. If nothing is entered here the ID will be used as a name. • dependencies: An optional String as comma separated list of other transformation IDs that must run AFTER this transformation down the compile-chain. This means if the current transformation is selected to run, all transformations that have dependencies on this will be run BEFORE. If nothing is entered here then this means there are no dependencies.
transformationGroup	Sometimes you may want to group other transformations and give this group a specific transformation ID as a kind of shortcut. You can do this by using the transformationGroup element giving the following information: • id: The unique String ID for this transformation group. This is the ID the transformation group will be referenced throughout KiCo. It may also again be referenced by other transformation groups! • transformations: Groups must specify their transformations. Use a String as comma separated list of other transformation IDs or transformation group IDs that should represent this group. Note that the order will be implied by the referenced transformations itself although if there is a free degree of order it can be influenced by the order specified here in the group. • name: An optional String name for this transformation. If nothing is entered here the ID will be used as a name. • alternatives: When selected this group specifies alternatives only. When this group is referenced, the first transformation is selected if not any other transformation from this group is already in the list of selected transformations. This is an advanced feature.

Example

plugin.xml

```
<extension
      point="de.cau.cs.kieler.kico.transformation">
   <transformationGroup
         id="NORMALIZE"
         dependencies="TRIGGEREFFECT, SURFACEDEPTH"
         name="Transform All Normalize">
   </transformationGroup>

   <transformationMethod
         class="de.cau.cs.kieler.sccharts.extensions.SCChartsCoreTransformation"
         id="TRIGGEREFFECT"
         method="transformTriggerEffect"
         name="Transform Trigger and Effect">
   </transformationMethod>

   <transformationMethod
         class="de.cau.cs.kieler.sccharts.extensions.SCChartsCoreTransformation"
         id="SURFACEDEPTH"
         method="transformSurfaceDepth"
         name="Transform Surface Depth">
   </transformationMethod>

   <transformationGroup
         id="ALL"
         dependencies="CORE NORMALIZE"
         name="Transform All">
   </transformationGroup>
</extension>
```

Compilation

Once a bunch of model transformations are registered, these can simply be called using the KiCo central "KielerCompiler" class with its method compile(). This will be given a List<String> of transformation IDs or a comma separated String of transformation IDs as the first parameter. The second parameter is the EObject that is being transformed. It should meet the signature of the first model transformation called. Note that the actual model transformations that are done may vary because KiCo will automatically inspect the dependencies of each transformation requested (deep-recursively). If you do not like this to happen as an advanced user you can use a third parameter that will skip this autocompletion. Note that if you switch this off also NO transformation groups can be processed. Here is an overview and examples how to use the compile() method:

Method	Description
`EObject KielerCompiler. compile(List<String> transformationIDs, EObject eObject)`	• transformationIDs: List of Strings representing the transformation IDs and a pre-ordering. Note that KiCo may automatically modify the order to meet the dependencies of the referenced transformation IDs or transformation group IDs. • eObject: The EObject that is the input to the compilation process. • Returns: The EObject returned from the last model transformation called by KiCo.
`EObject KielerCompiler. compile(String transformationIDs, EObject eObject)`	This is a convenient method only which can be used to give transformation IDs or transformation group IDs as a comma separated String. For eObject and the return value see above.
`EObject KielerCompiler. compile(List<String> transformationIDs, EObject eObject, boolean autoexpand)`	This is an advanced compile method which can turn of auto-expansion with the last parameter. Use this with care! Note that if switching autoexpand off you cannot use transformation group IDs any more. Also no dependencies will be considered. The transformations will be applied straight forward in the order defined by the transformationIDs list.

Examples

Java Code

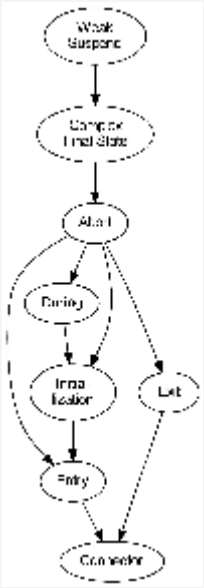
```
import de.cau.cs.kieler.kico.KielerCompiler;
...
private MyEObjectClass myMethod(EObject eObject) {
        ...
        transformed = (MyEObjectClass) KielerCompiler.compile("ABORT, SIGNAL", eObject);
        ...
        return transformed
}
```

Xtend Code

```
import de.cau.cs.kieler.kico.KielerCompiler
...
def dispatch MyEObjectClass myMethod(EObject eObject) {
        transformed = KielerCompiler.compile("ABORT, SIGNAL", eObject) as MyEObjectClass
        ...
        transformed
}
```

Requirement Completion

Original



Original Dependency Graph

Two alternative transformation implementations for Abort

Example 1

Selected for transformation

Auto selected requirements

Example 2

Selected for transformation

Auto selected requirements using DEFAULT of alternative group

Example 3

Selected for transformation

Auto selected requirements using selected alternative

Help / Problems / FAQs

Maybe you get into problems when using KiCo. The following list should give you hints to solve these. If you have a problem not considered here please write us an e-mail (see above for contact information of the persons in charge of KiCo).

Symptom	Reason	Solution

You get the following run time error:
 	
`ENTRY de.cau.cs.kieler.klighd 4 0 2014-03-17 11:08:46.009`
`!MESSAGE`
`!STACK 0`
`java.lang.RuntimeException: Cannot find a transformation with the ID 'ABORT2'. Make sure that the transformation with this ID is registered and its declaring plugin is loaded.`
`Make sure that the ID does exactly match (case sensitive). Maybe you forgot to separate multiple ID's by a comma.`
`    at de.cau.cs.kieler.kico.KielerCompiler.getTransformation(KielerCompiler.java:61)`
`    at de.cau.cs.kieler.kico.KielerCompiler.getDependencies(KielerCompiler.java:82)`
`    at de.cau.cs.kieler.kico.KielerCompiler.isDependingOn(KielerCompiler.java:102)`
`    at de.cau.cs.kieler.kico.KielerCompiler.insertTransformationID(KielerCompiler.java:136)`
`    at de.cau.cs.kieler.kico.KielerCompiler.expandDependencies(KielerCompiler.java:164)`
`...`
	There is a transformation with ID "ABORT2" referenced either by the initial call to KielerCompiler.compile() or by some of the dependent transformations / transformation group.
	
	But KiCo could not find any registered transformation with ID "ABORT2".
	
	Maybe the plugin declaring "ABORT2" was not loaded or the ID is misspelled.
		Check why "ABORT2" may not be found by KiCo, more specifically, check if the declaring can be loaded (sometimes compiler error prevent it from being loaded or it has unsatisfied dependencies).
		Also check the spelling of the ID, maybe the declaring plugin defines the transformation with the ID "abort2".

You get the following error:
 	
`!ENTRY de.cau.cs.kieler.kico 2 2 2014-03-17 11:26:13.818`
 	
`!MESSAGE Extension 'TERMINATION' from component: de.cau.cs.kieler.sccharts cannot be loaded becaus this ID is already taken. (de.cau.cs.kieler.kico)`
	The trasformation with ID "TERMINATION" is already registered and this ID cannot be used a second time.
	The component "de.cau...sccharts" tries to register a transformation with this ID while this ID is already taken.
		Rename one of the IDs or form a group of transformations.