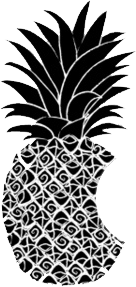
Pineapple DataflowNet Modeler  
Quick guide



Created by Ákos Hajdu, József Makai and Márton Búr as a homework for the “Model Driven Software Development” course at the Budapest University of Technology and Economics (<http://www.inf.mit.bme.hu/en>)

# Technologies used

* Eclipse Modeling Framework 2.0.2 for creating the metamodel (Has to be installed in Eclipse)
* EMF IncQuery 0.8.0 (<http://viatra.inf.mit.bme.hu/incquery>) for validation (Has to be installed in Eclipse)
* Xtext 2.5.3 for creating textual language (Has to be installed in Eclipse)
* Xtend 2.5.3 for generating GraphML and Java code (Has to be installed in Eclipse)
* yED 3.12.2 for visualization (<http://www.yworks.com/en/products_yed_about.html>)
* Akka 2.2.2 (<http://akka.io/>) and Paho 0.4.0 (<http://www.eclipse.org/paho/>) for remote communication (Included as dependecies)

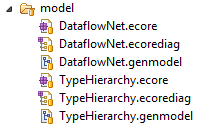
# How to use it

## Load metamodel and sample model

Clone the project and load the following projects into the workspace:

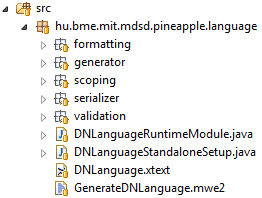
* hu.bme.mit.mdsd.pineapple
* hu.bme.mit.mdsd.pineapple.dataflownet.common
* hu.bme.mit.mdsd.pineapple.dataflownet.generator
* hu.bme.mit.mdsd.pineapple.dataflownet.generator.incquery
* hu.bme.mit.mdsd.pineapple.dataflownet.graphml
* hu.bme.mit.mdsd.pineapple.dataflownet.incquery
* hu.bme.mit.mdsd.pineapple.dataflownet.ui.generator
* hu.bme.mit.mdsd.pineapple.language
* hu.bme.mit.mdsd.pineapple.ui.buttons

The metamodel can be found in the hu.bme.mit.mdsd.pineapple project.



Generate the metamodels using the .genmodel files. Generate TypeHierarchy.genmodel first, and then DataflowNet.genmodel.

The textual language (Xtext) can be found in the hu.bme.mit.mdsd.pineapple.language project.



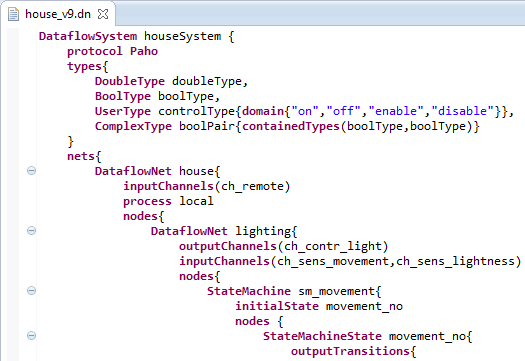
Right click on the file GenerateDNLanguage.mwe2 and select Run as 🡪 MWE2 Workflow.

Start a runtime Eclipse application and load the following projects:

* hu.bme.mit.mdsd.pineapple.sample
* hu.bme.mit.mdsd.pineapple.dataflownet.bundle

The sample instance model can be found in the file house\_v9.dn in the project hu.bme.mit.mdsd.pineapple.sample.

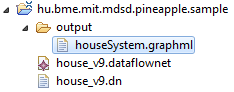




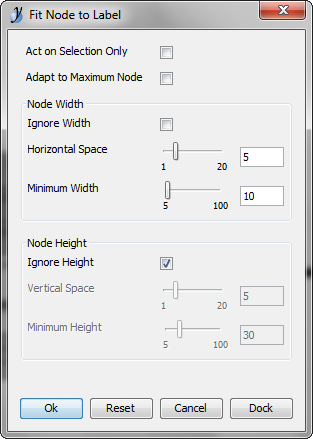
Right click on the file house\_v9.dn and select “Convert textual dataflownet to XMI” to get the file house\_v9.dataflownet. This file can also be edited using the tree editor.

## Generate GraphML code

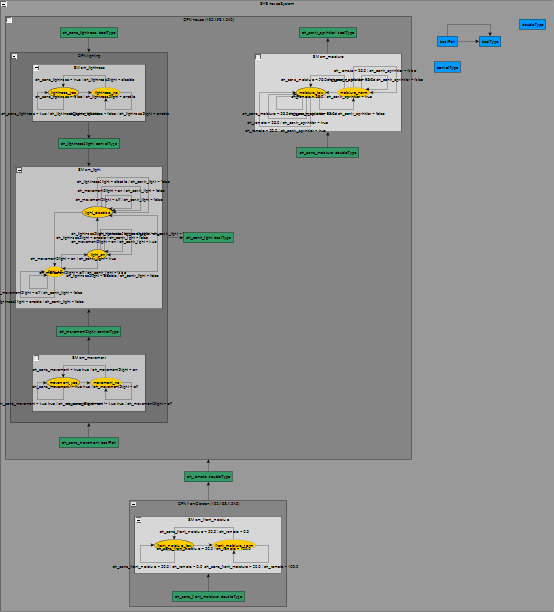
Right click on the house\_v9.dataflownet file and select “Generate GraphML Representation”. The .graphml file will be generated in the sample project under a folder named output.



Open the .graphml file in yED and select Tools 🡪 Fit Node to Label. Use the following settings and click ok.



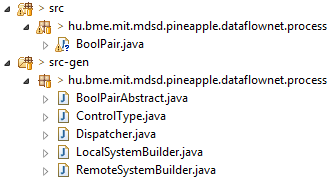
Use some layout algorithm, for example Layout 🡪 Orthogonal 🡪 Classic. The result can be seen below.



## Generate Java code

Right click on the house\_v9.dataflownet file and select “Generate Java Simulation Code” then select “Generate Java UI Code”.

The following files are generated in the bundle project:



Implement the unimplemented methods (compareTo and toString) of the custom complex types (BoolPair) under the src folder. For example use the following implementation:

|  |
| --- |
| @Override  public int compareTo(BoolPair boolPair) {  if(value1 == boolPair.value1 && value2 == boolPair.value2) return 0;  return 1;  }  @Override  public String toString(){  return value1 + "," + value2;  } |

A new project is generated for the UI, namely hu.bme.mit.mdsd.pineapple.dataflownet.process.ui.

## Try the generated Java code

Close the runtime Eclipse application and load the following projects into the host Eclipse:

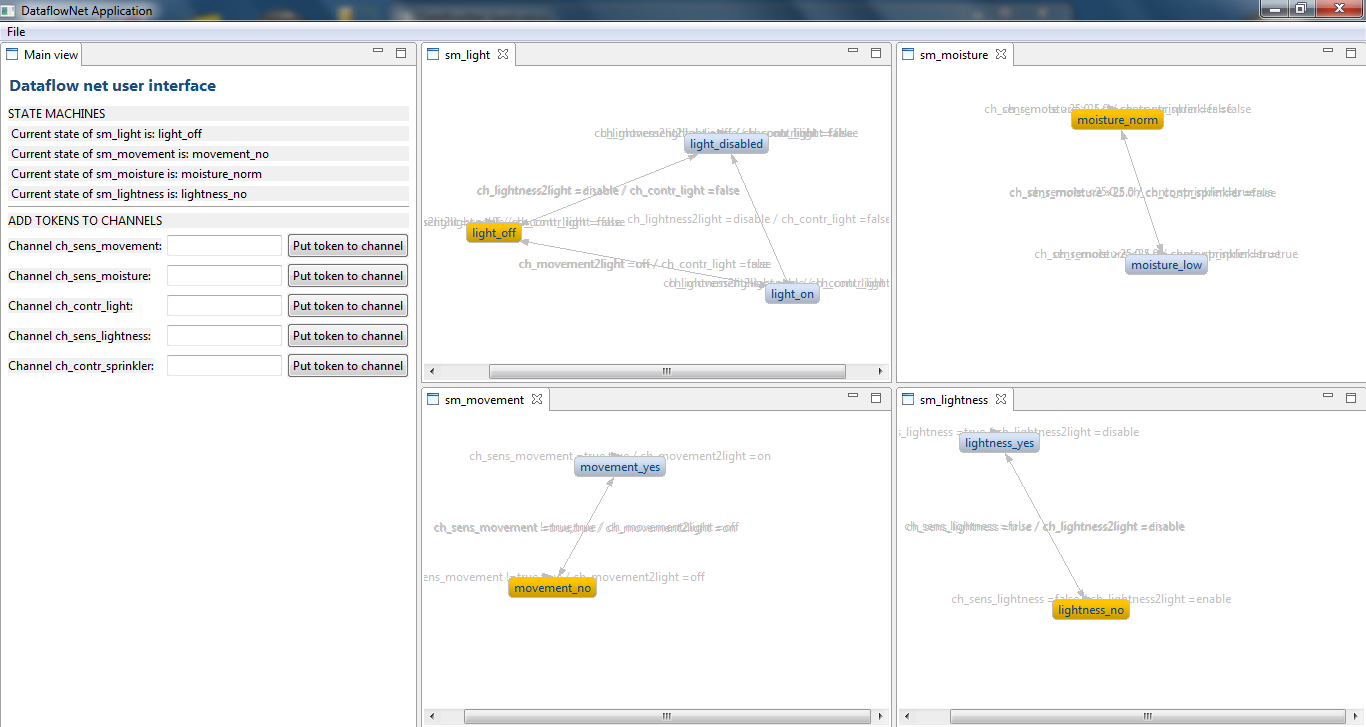
* hu.bme.mit.mdsd.pineapple.dataflownet.process.ui
* hu.bme.mit.mdsd.pineapple.dataflownet.bundle
* hu.bme.mit.mdsd.pineapple.dataflownet.bundle.dependencies
* hu.bme.mit.mdsd.pineapple.dataflownet.bundle.dependency.paho
* hu.bme.mit.mdsd.pineapple.dataflownet.ui.product

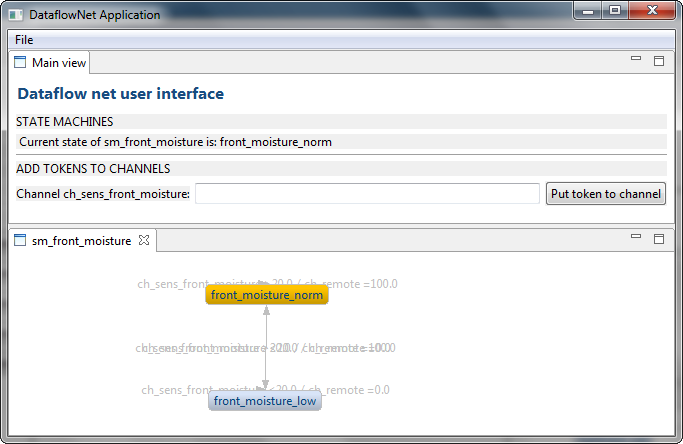
Export the RCP using the DataflownNetApplication.product in the project hu.bme.mit.mdsd.pineapple.dataflownet.ui.product. You may have to add the required plug-ins on the Dependencies tab of DataflownNetApplication.product.

Run the exported application DataflowNetUI.exe in two instances with the following command line arguments:

* The first one with “local –clearPersistedState”
* The second one with “remote –clearPersistedState”

The first parameter (“local” and “remote” in the example) is the name of the process defined in the metamodel. After arranging the tabs, you get the following screens:





You can now put tokens on the top-level channels by entering the value of the token as a string in the textboxes. You can try for example putting “10” on ch\_sens\_front\_moisture and putting “true true” on ch\_sens\_movement.