Accélérateur de la transformation numérique

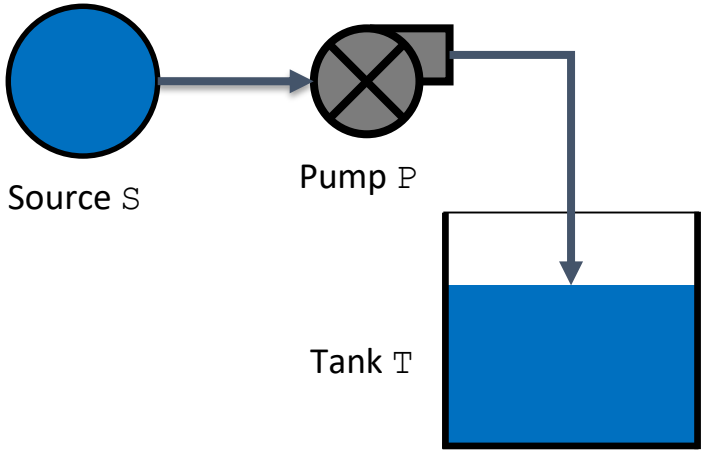
OpenAltaRica - Example

A (Simple) Water Supply System

www.irt-systemx.fr

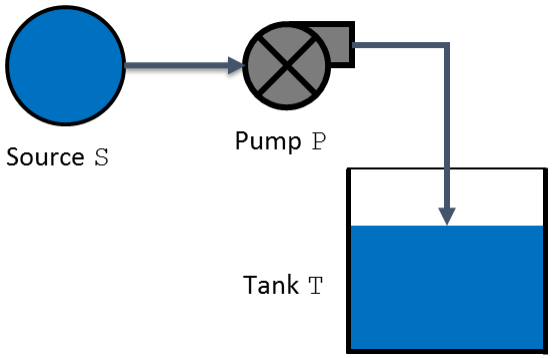
OpenAltaRica - Example

A (Simple) Water Supply System

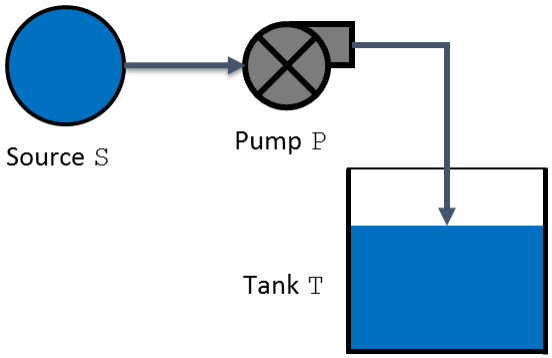


WHAT Feel the tank

HOW Move the water from the source S to the tank T by means of a pump P

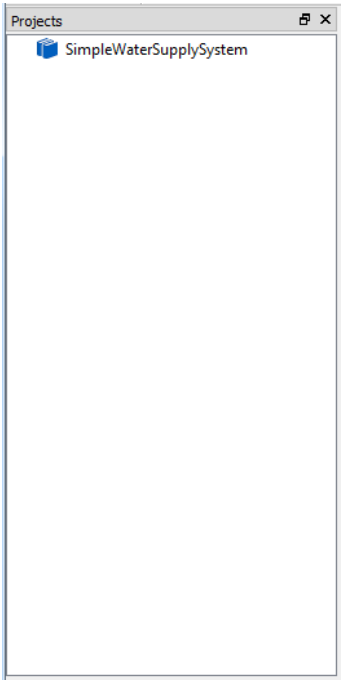


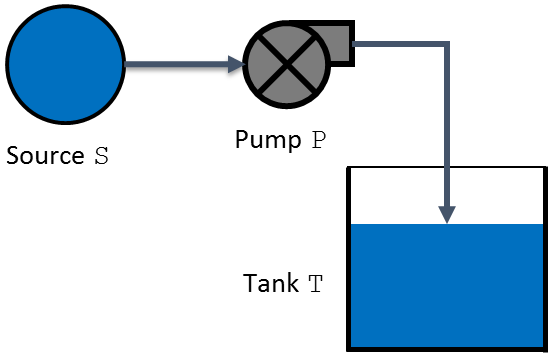
A. To design your model



A. To design your model

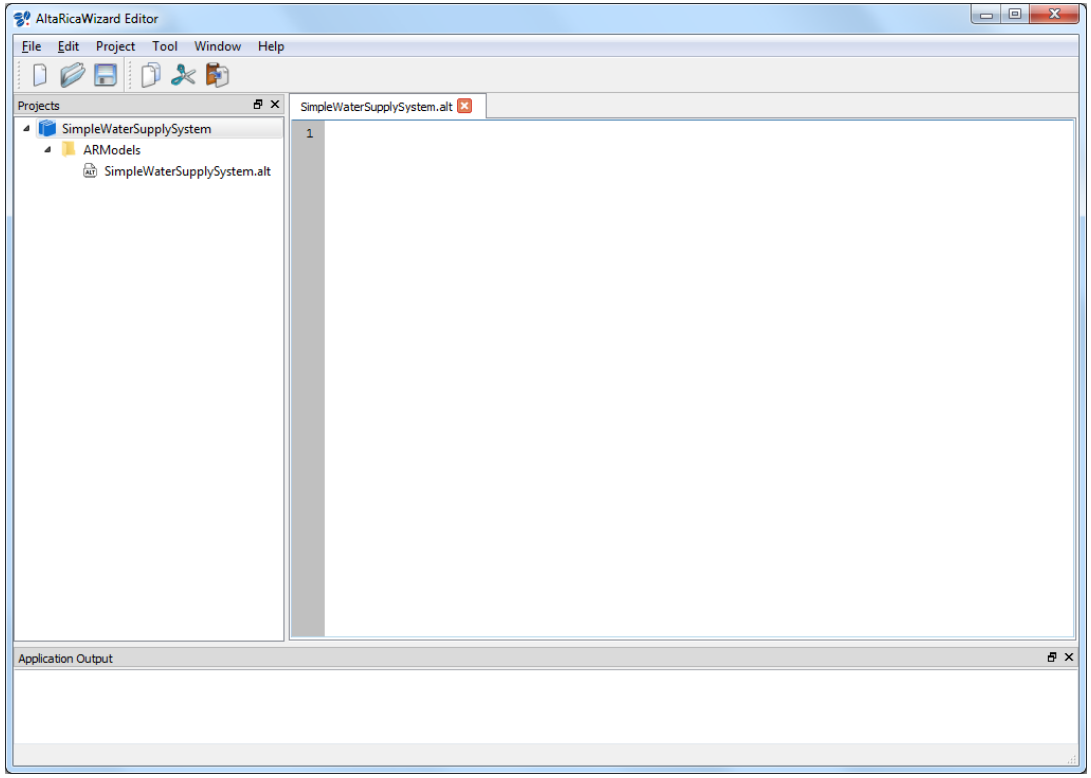
1. Create a new project into the AltaRicaWizard:
   a. ‘File’->‘New File or Project’->’New Project’;
   b. ‘SimpleWaterSupplySystem’ -> ‘OK’.

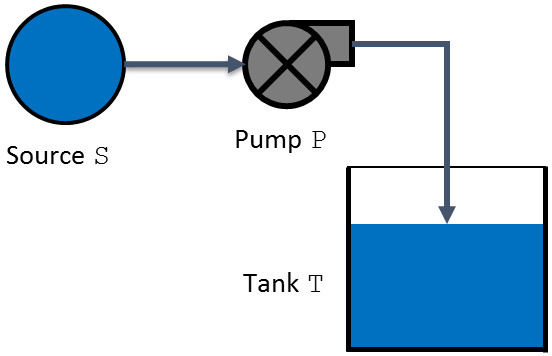




A. To design your model

2. Create a new AltaRica 3.0 model:
   a. Right-click on the project ‘SimpleWaterSupplySystem’;
   b. ‘Add new file to “SimpleWaterSupplySystem”’;
   c. You can create a new folder (e.g. ‘ARModels’);
   d. ‘SimpleWaterSupplySystem.alt’ -> ‘Save’.





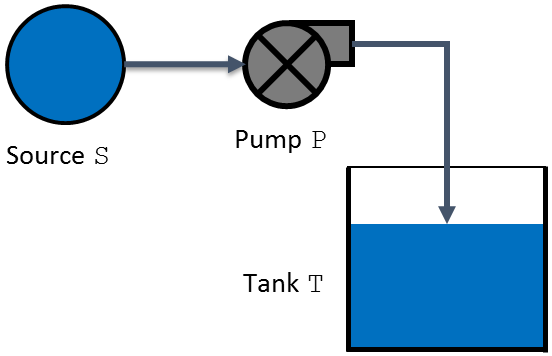
A. To design your model

3. Design your AltaRica 3.0 model.

The source and the pump are repairable components

=> a generic class representing repairable components

```
class RepairableComponent
    Boolean working (init = true);
    parameter Real lambda = 0.0001;
    parameter Real mu = 0.01;
    event failure (delay = exponential(lambda));
    event repair (delay = exponential(mu));
    transition
        failure: working -> working := false;
        repair: not working -> working := true;
end
```



A. To design your model

3. Design your AltaRica 3.0 model.

1

```
class RepairableComponent
    Boolean working (init = true);
    parameter Real lambda = 0.0001;
    parameter Real mu = 0.01;
    event failure (delay = exponential(lambda));
    event repair (delay = exponential(mu));
    transition
        failure: working -> working := false;
        repair: not working -> working := true;
end
```

2

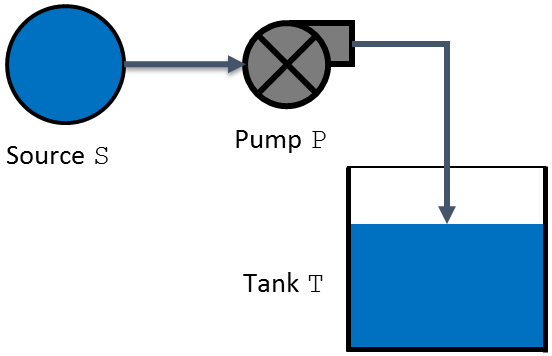
```
class Source
    extends RepairableComponent;
    Boolean output (reset = false);
    assertion
        output := if working then true else false;
end
```

3

```
class Pump
    extends RepairableComponent;
    Boolean input, output (reset = false);
    assertion
        output := if working then input else false;
end
```

4

```
block SimpleWaterSupplySystem
    Source S;
    Pump P;
    block Tank
        Boolean input (reset = false);
    end
    assertion
        P.input := S.output;
        Tank.input := P.output;
end
```



A. To design your model

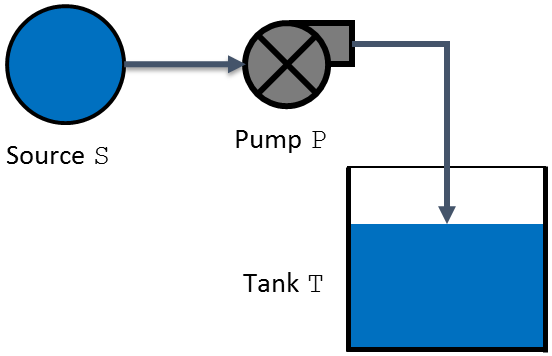
3. Design your AltaRica 3.0 model.

Add an observer providing information about the input flow of the tank. There is no input flow to the tank if:

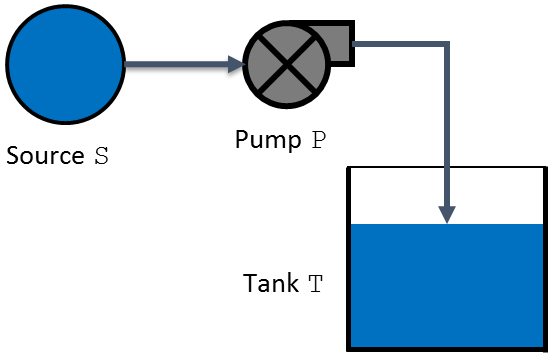
the value of the flow variable '`input`' of the tank '`Tank`' is equal to '`false`'.

```
block SimpleWaterSupplySystem
    Source S;
    Pump P;
    block Tank
        Boolean input (reset = false);
    end
    assertion
        P.input := S.output;
        Tank.input := P.output;

    observer Boolean TE_tankIsNotSupplied = Tank.input == false;
end
```



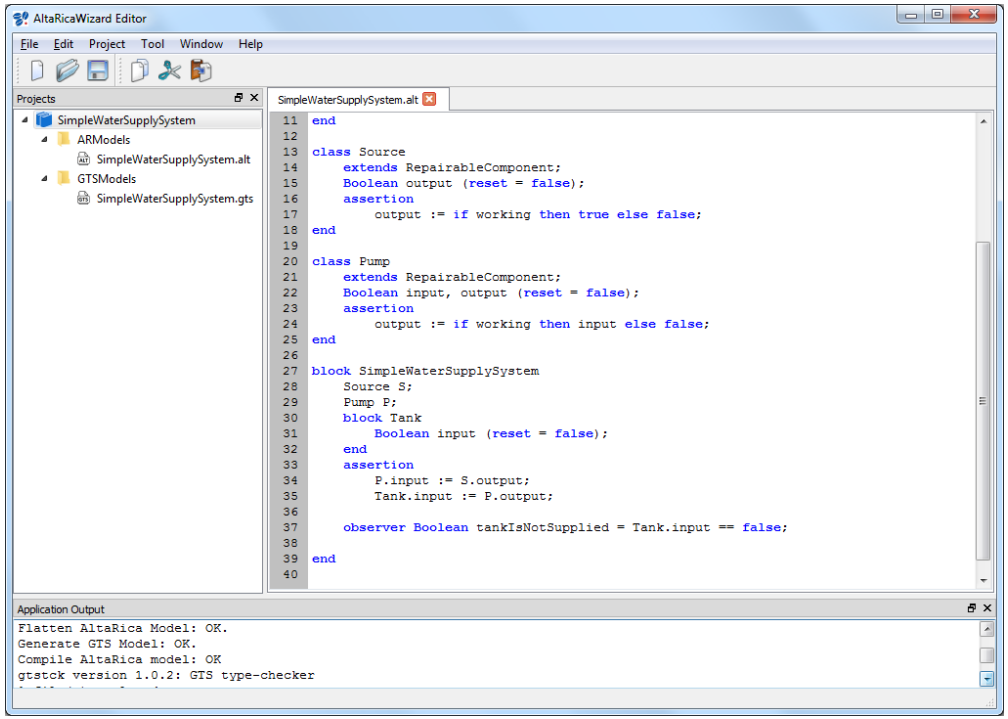
B. To check and compile the model

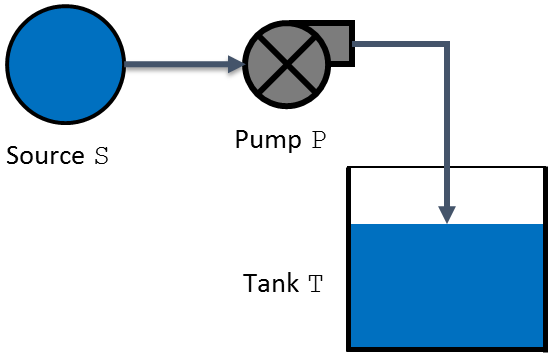


B. To check and compile the model

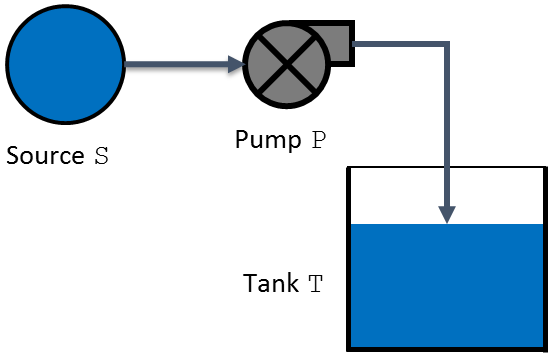
Launch the AltaRica 3.0 compiler:

a. Click on ‘Tool’ -> ‘Flattening’;
b. You can create a new output folder (e.g. ‘GTSModels’);
c. Results are printed into the ‘Application Output’ part. A new file is created into the output folder (e.g. ‘GTSModels’).



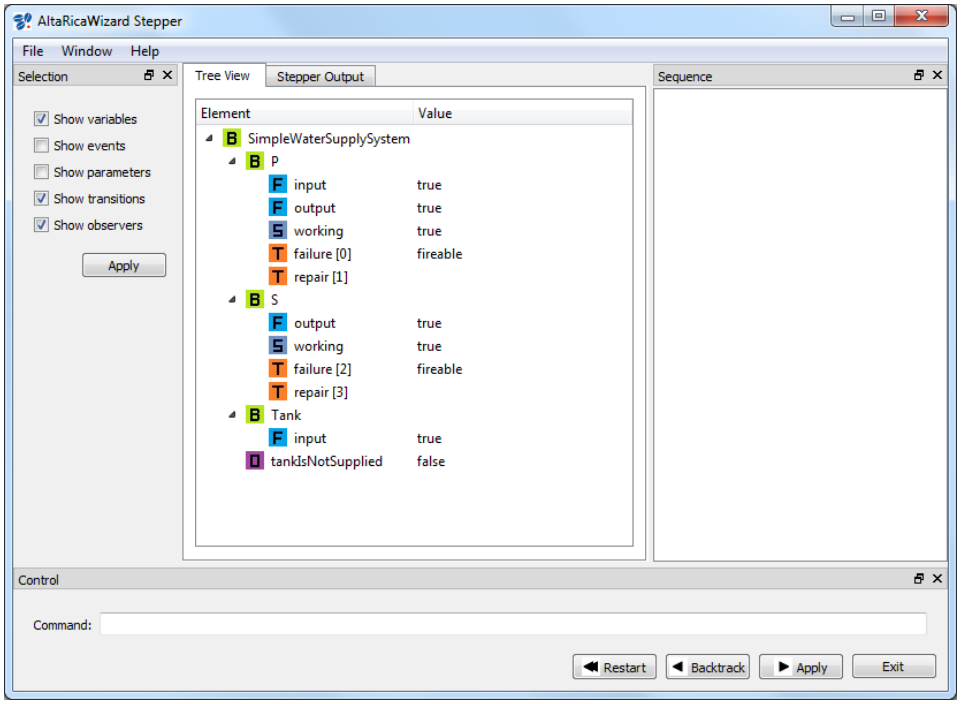


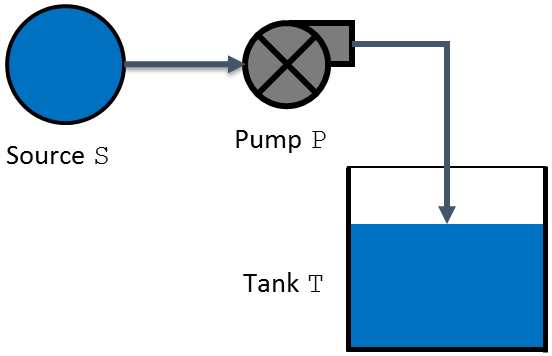
C. To simulate the model



C. To simulate the model

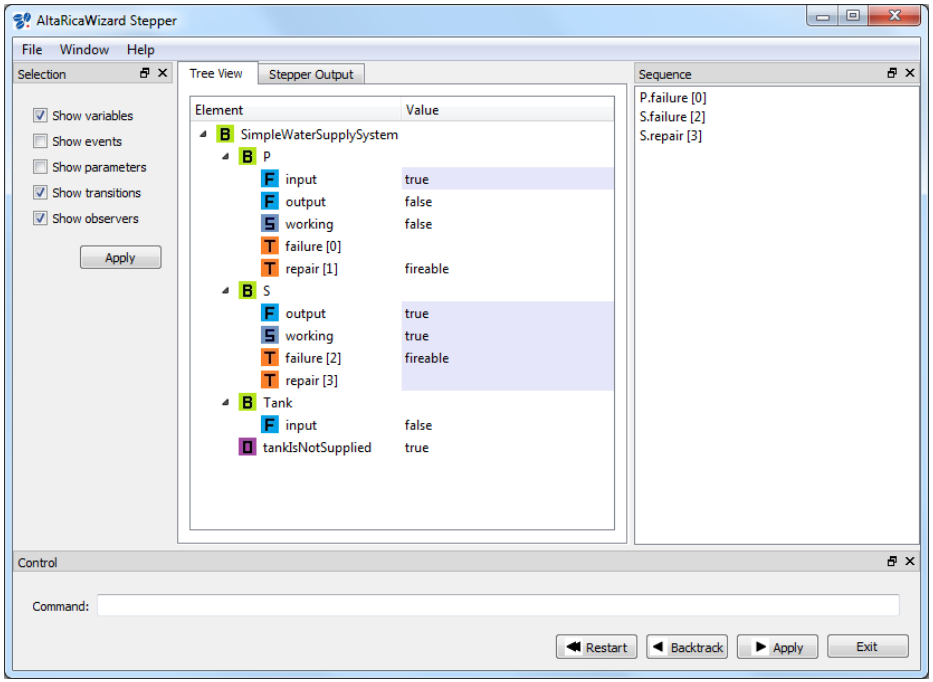
1. Launch the stepwise simulator:
   a. Click on ‘Tool’ -> ‘Stepwise Simulation’;
   b. Click on ‘Next’ for the flattening part (it compiles the model if it has been modified) -> Click on ‘Next’ for the ‘Stepwise Simulation’ part;
   c. A new window opens.

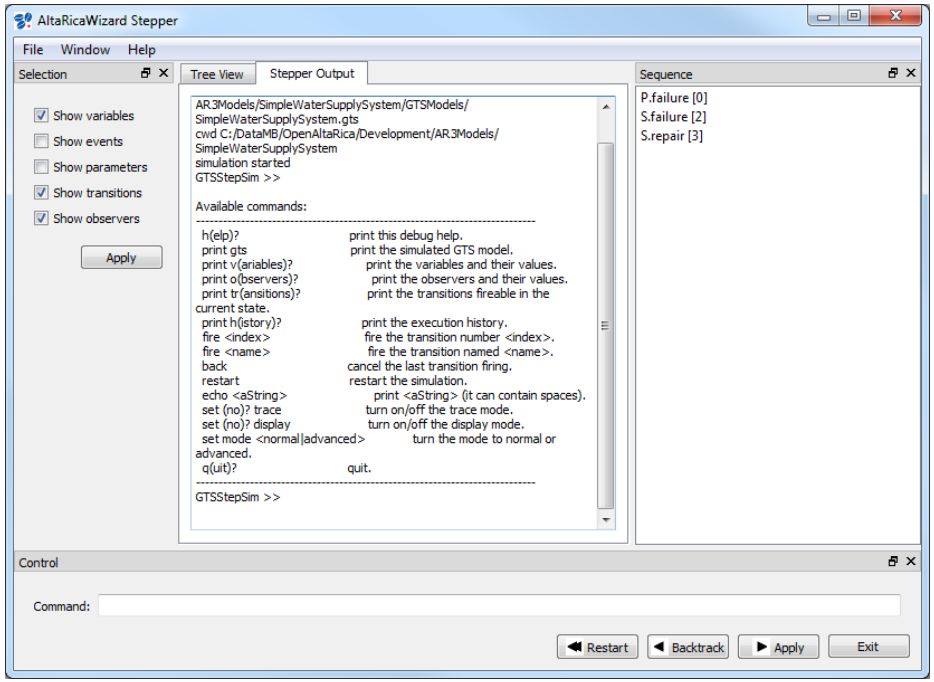


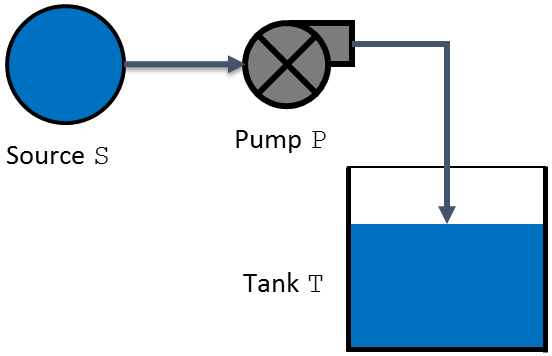


C. To simulate the model

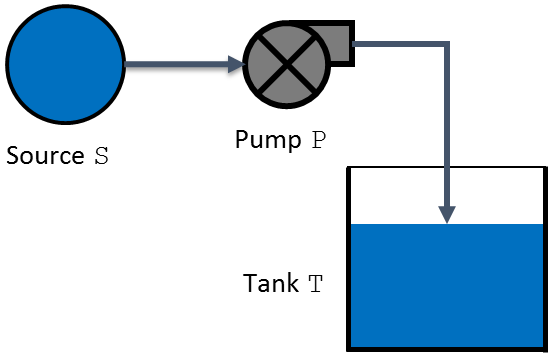
2. Launch commands to perform experiments:
   - Select elements you want to see (variables, events, etc.);
   - Click on fireable transitions to perform a simulation;
   - You can backtrack or restart to a new simulation;
   - You can run other commands and see results onto the 'Stepper Output' part (type 'help' and click on 'Apply' to see available commands).







D. To generate and study a fault tree

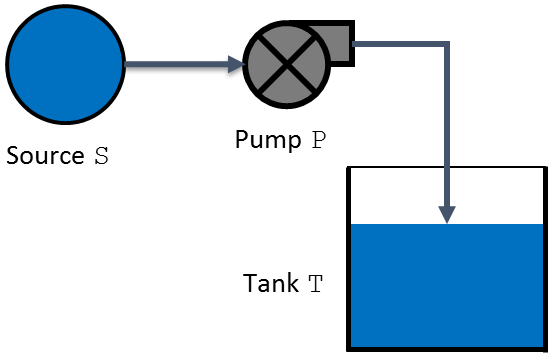


D. To generate and study a fault tree

1. We consider the top event defined by the observer named ‘TE_tankIsNotSupplied’ and meaning “There is no input flow to the tank”

```
block SimpleWaterSupplySystem
    Source S;
    Pump P;
    block Tank
        Boolean input (reset = false);
    end
    assertion
        P.input := S.output;
        Tank.input := P.output;

    observer Boolean TE_tankIsNotSupplied = Tank.input == false;
end
```

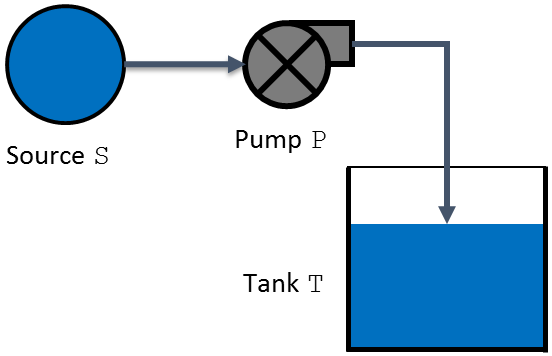


D. To generate a fault tree

2. Launch the compiler to fault trees and analyze the generated fault tree:
   a. Click on ‘Tool’ -> ‘Compilation into Fault Tree’;
   b. Click on ‘Next’ for the flattening part (it compiles the model if it has been modified) -> Click on ‘Next’ for the ‘Compilation into Fault Tree’ part;
   c. Select the name of the generated fault tree (you can create a new output folder: e.g. ‘FaultTrees’) -> click on the case ‘Launch Fault Tree Assessment’ and select the top event you want to observer (‘TE_tankIsNotSupplied’ with the value ‘true’) -> ‘Next’

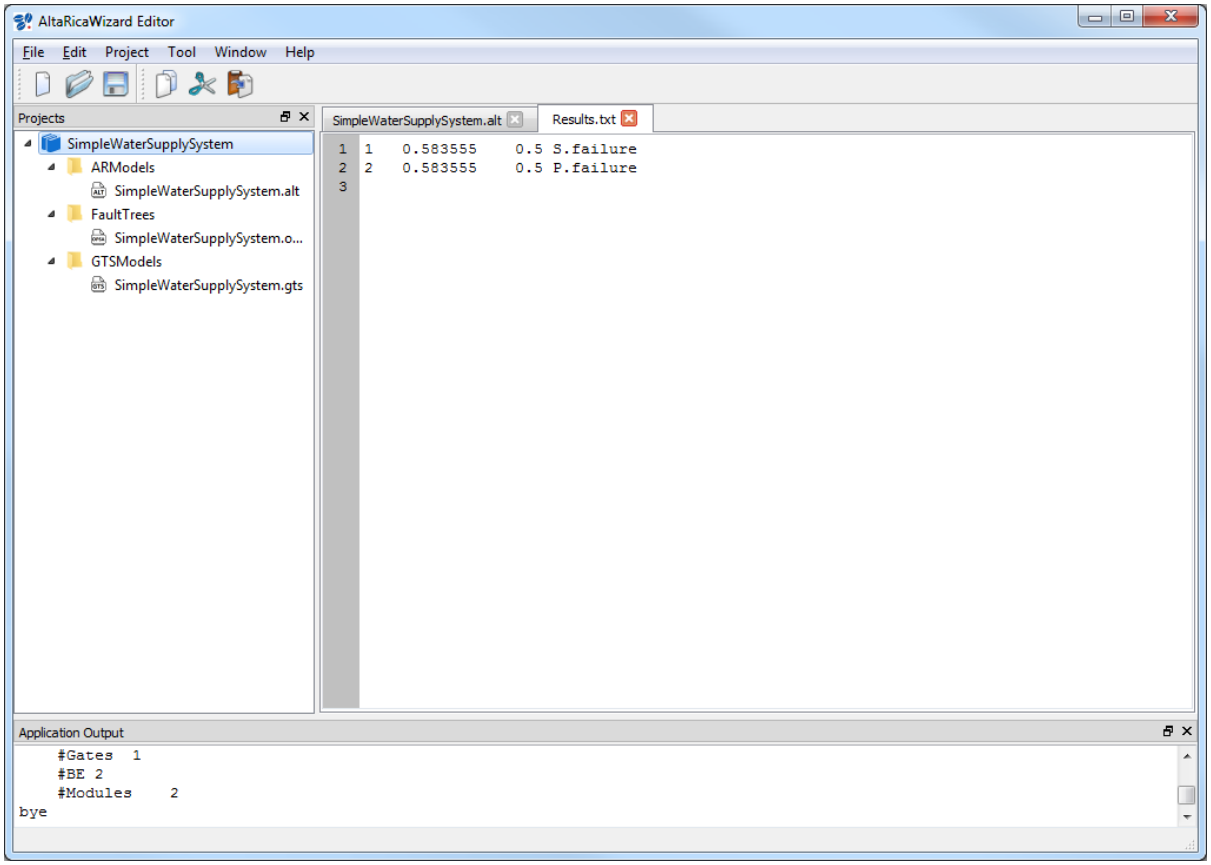
   A new window opens

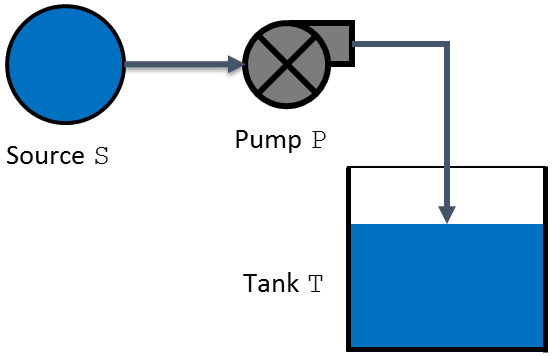
   d. Define the XFTA command file (into the created output folder e.g. ‘FaultTrees’) and select features you want to compute (only orders of minimal cutsets with results into the file ‘FaultTrees/Results.txt’) -> click on ‘Create’;
   e. Into the ‘Calculations’ part, click on ‘Launch’.



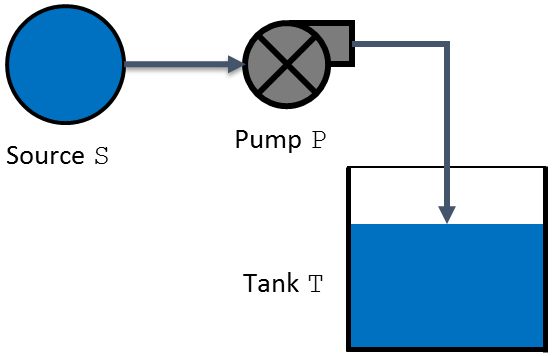
D. To generate a fault tree

2. Open results:
   a. Click on ‘File’ -> ‘Open File or Project’ -> ‘Open File’;
   b. Select the file of results ‘FaultTrees/Results.txt’;





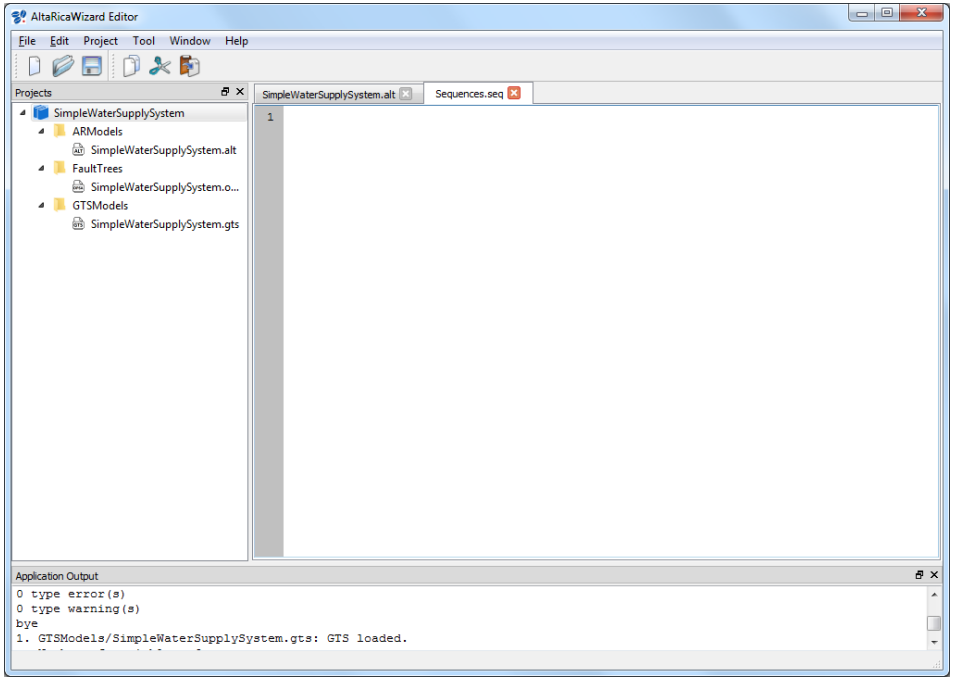
E. To script simulation

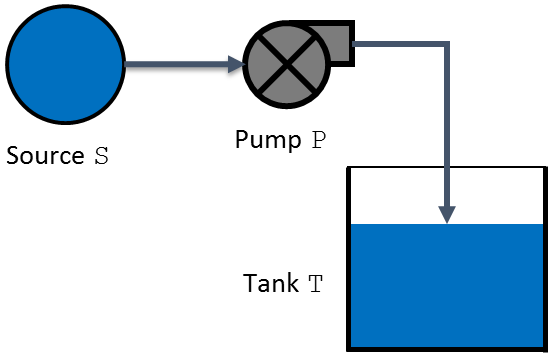


E. To script simulation

1. Create the script file for simulation:
   a. Click on ‘File’ -> ‘New File or Project’ -> ‘New File’ (an untitled file opens);
   b. Save this untitled file : click on ‘File’ -> ‘Save File as…’ (you can create a dedicated folder: e.g. ‘Scripts’) and save it (e.g. ‘Sequences.seq’);

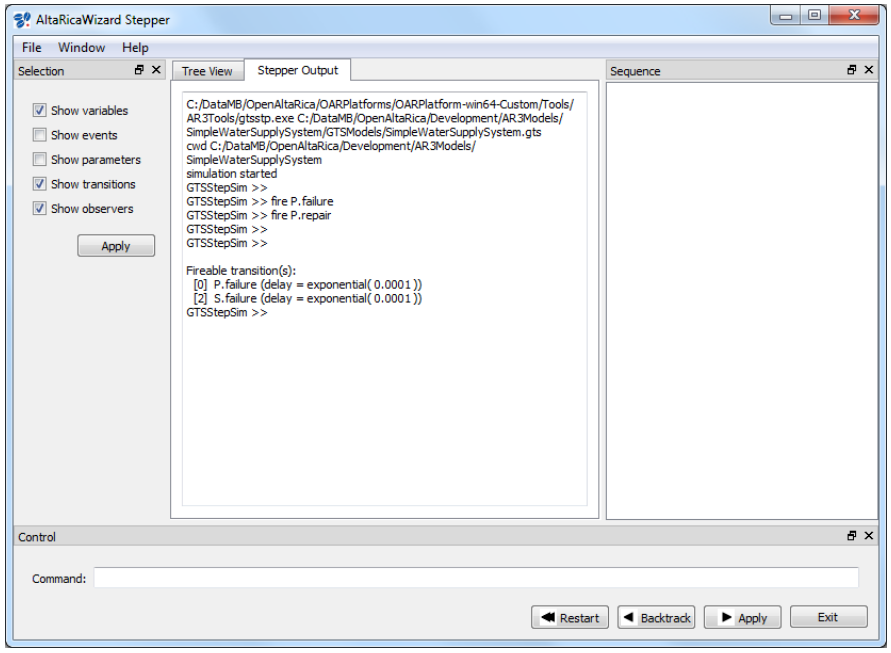
REM: the dedicated folder and the script file are not listed to the project.





E. To script simulation

2. Edit the script file for simulation:
   a. Enter directly commands into this script file;
   b. Launch the stepwise simulator and select the ‘Stepper Output’ part;
   c. Type the command ‘run’ with the path to the script file (‘run ./Scripts/Sequences.seq’)



CONTACTS

The OpenAltaRica project

www.openaltarica.fr

contact.oar@irt-systemx.fr

www.irt-systemx.fr