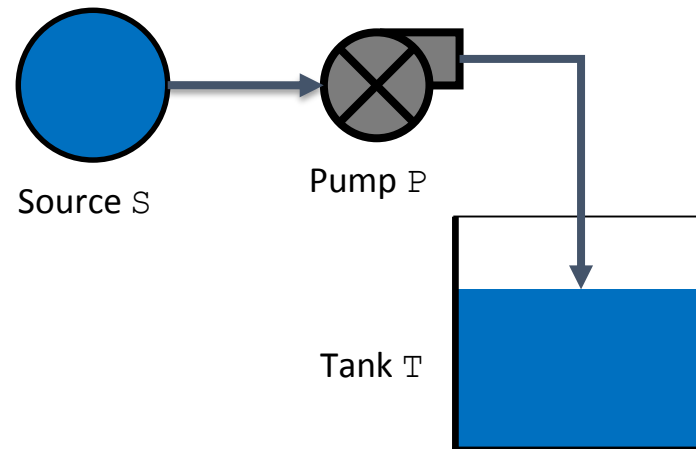


# OpenAltaRica - Example

A (Simple) Water Supply System

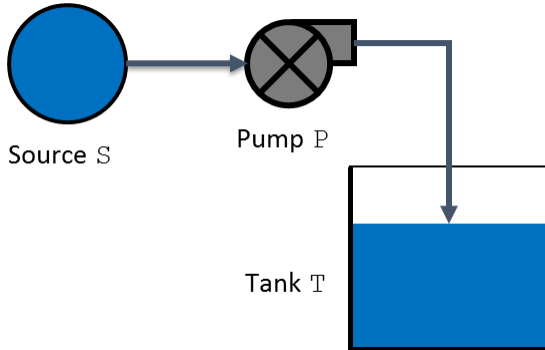
[www.irt-systemx.fr](http://www.irt-systemx.fr)



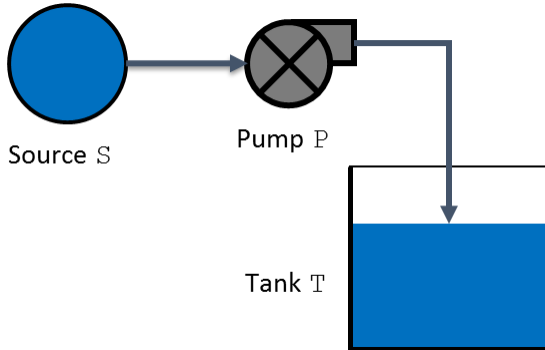


WHAT Feel the tank

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

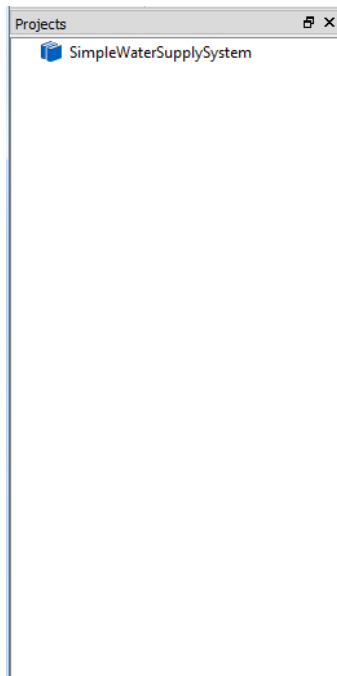


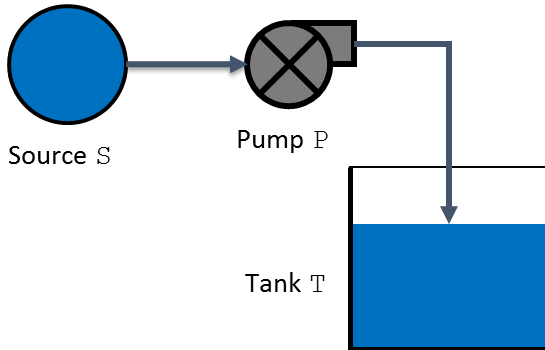
## A. To design your model



### A. To design your model

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

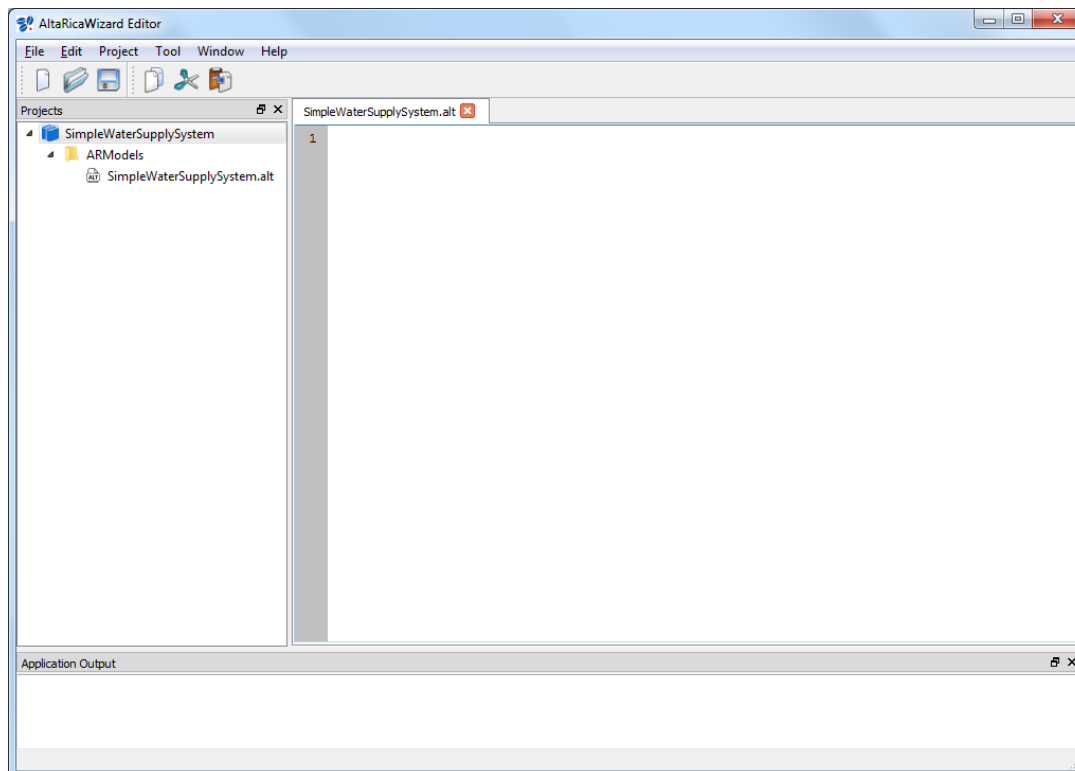


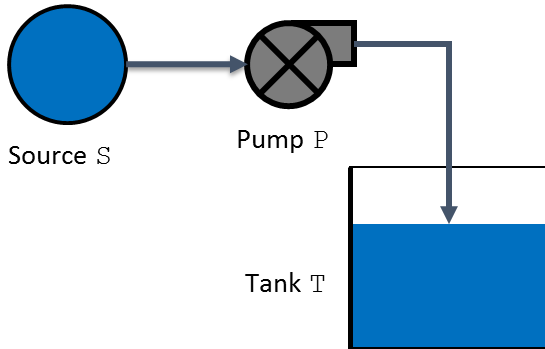


### A. To design your model

#### 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'.



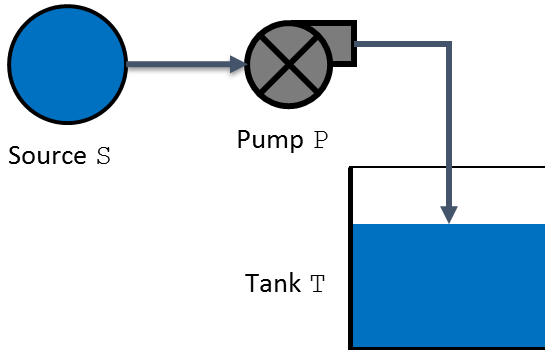


## A. To design your model

### 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

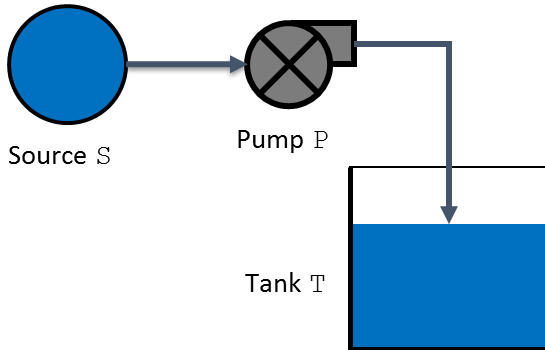
```
class Source
  extends RepairableComponent;
  Boolean output (reset = false);
  assertion
    output := if working then true else false;
end
```

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.

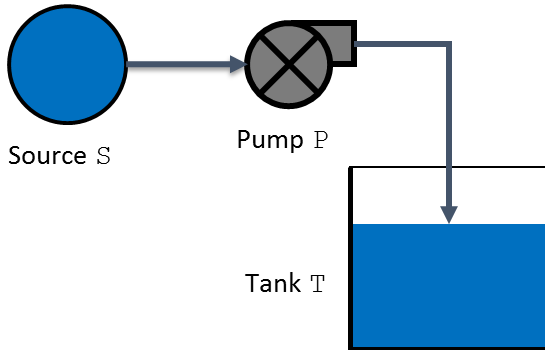
Add an observer providing information about the input flow of the tank. There is no input flow to the tank if: the value of the flow variable 'input' of the tank 'Tank' is equal to 'false'.

```

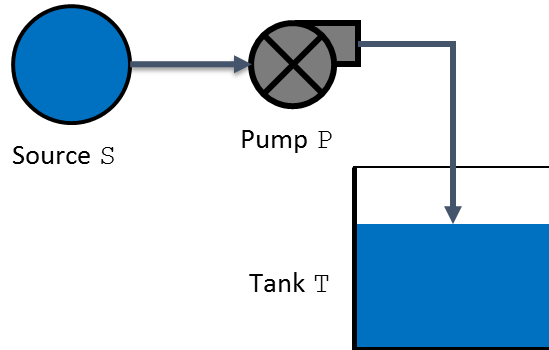
block SimpleWaterSupplySystem
  Source S;
  Pump P;
  block Tank
    Boolean input (reset = false);
  end
  assertion
    P.input := S.output;
    Tank.input := P.output;
  observer Boolean TE_tankIsNotSupplied = Tank.input == false;
end

```





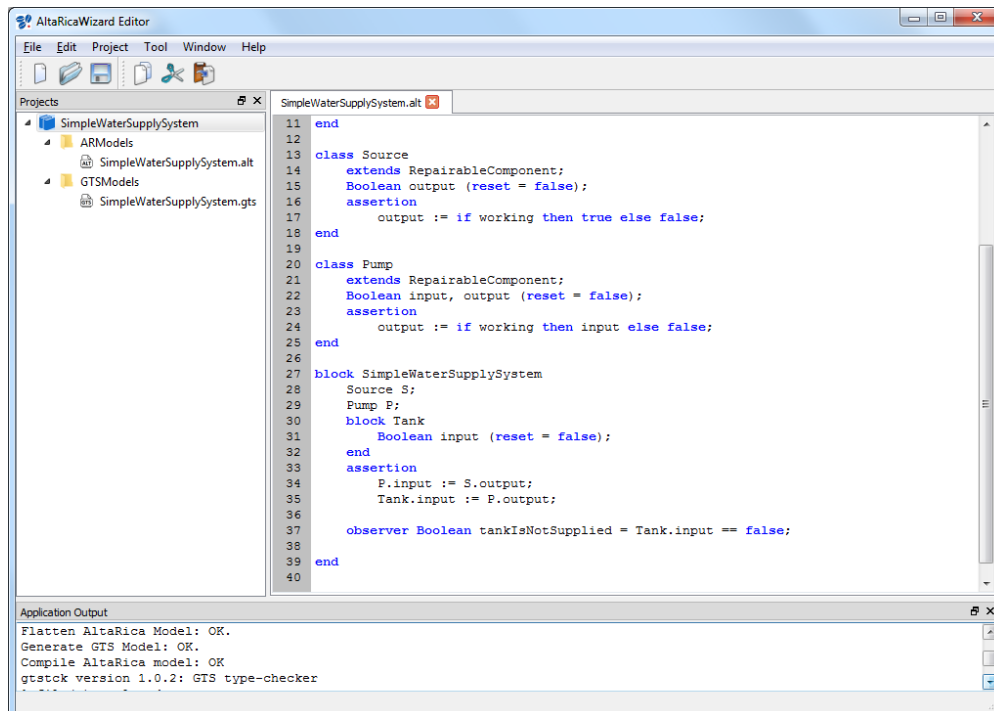
## B. To check and compile the model



## B. To check and compile the model

Launch the AltaRica 3.0 compiler:

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



```

11 end
12
13 class Source
14   extends RepairableComponent;
15   Boolean output (reset = false);
16   assertion
17     output := if working then true else false;
18 end
19
20 class Pump
21   extends RepairableComponent;
22   Boolean input, output (reset = false);
23   assertion
24     output := if working then input else false;
25 end
26
27 block SimpleWaterSupplySystem
28   Source S;
29   Pump P;
30   block Tank
31     Boolean input (reset = false);
32   end
33   assertion
34     P.input := S.output;
35     Tank.input := P.output;
36
37   observer Boolean tankIsNotSupplied = Tank.input == false;
38
39 end
40

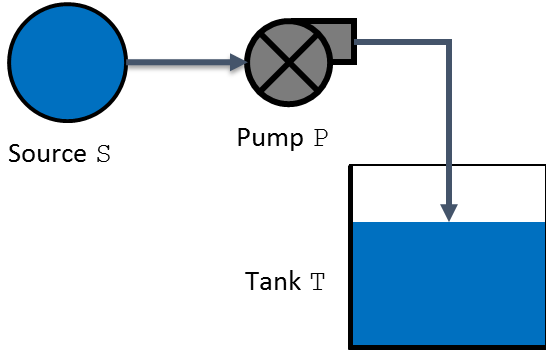
```

Application Output

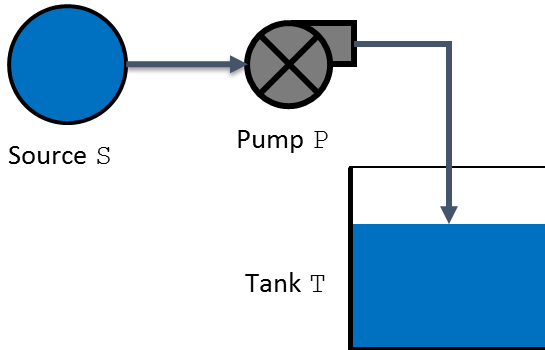
```

Flatten AltaRica Model: OK.
Generate GTS Model: OK.
Compile AltaRica model: OK
gtstocK version 1.0.2: GTS type-checker

```

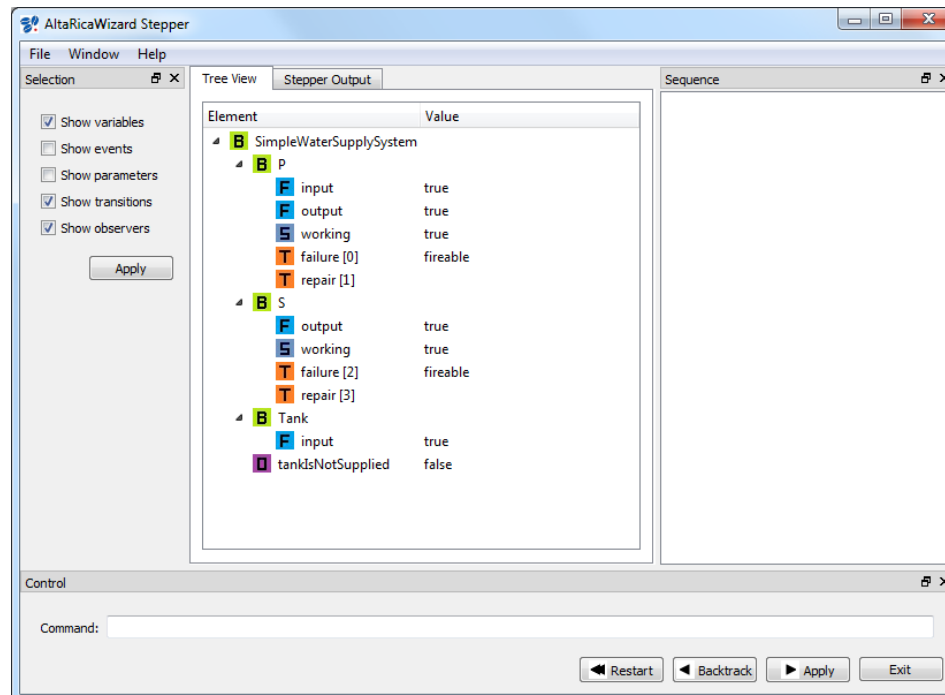


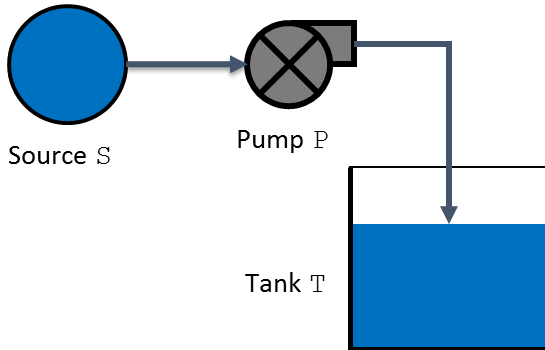
### C. To simulate the model



### 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.

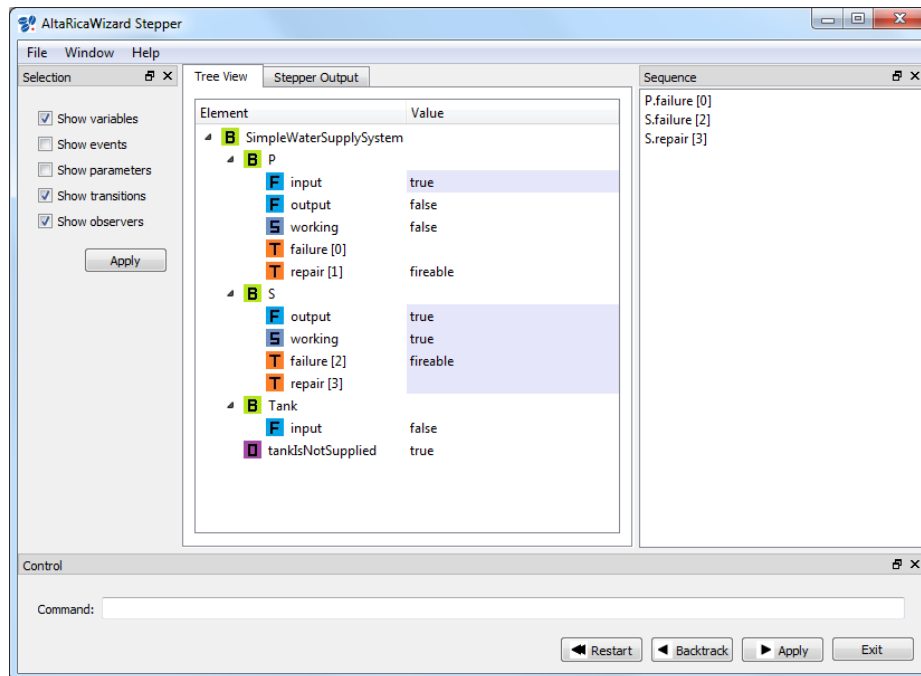




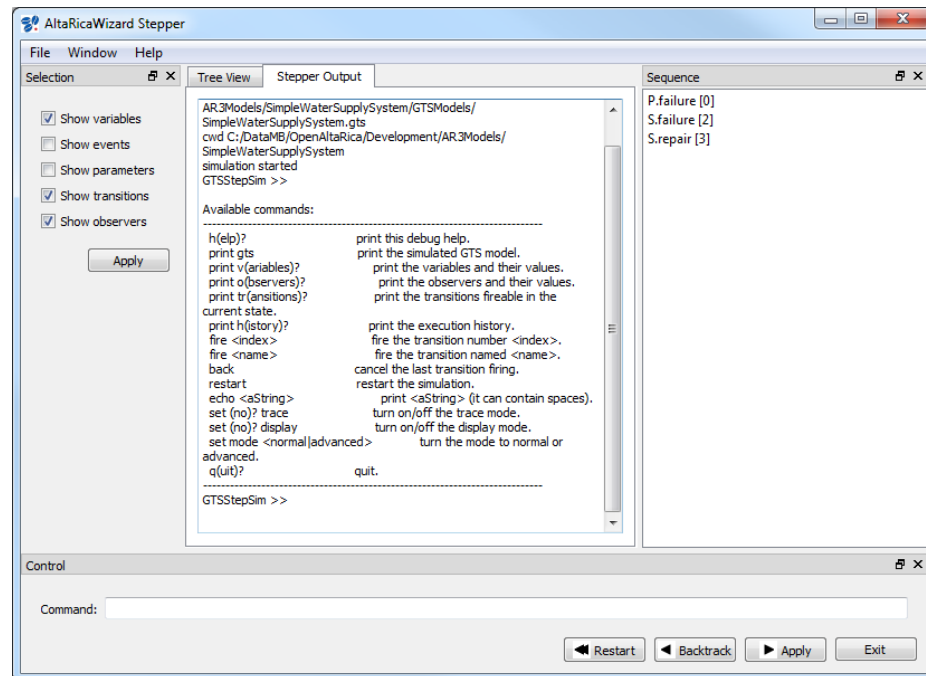
### C. To simulate the model

#### 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).



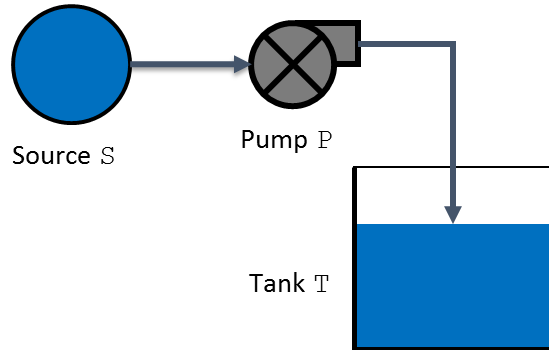
Element	Value
SimpleWaterSupplySystem	
P	
input	true
output	false
working	false
failure [0]	
repair [1]	fireable
S	
output	true
working	true
failure [2]	fireable
repair [3]	
Tank	
input	false
tanksNotSupplied	true



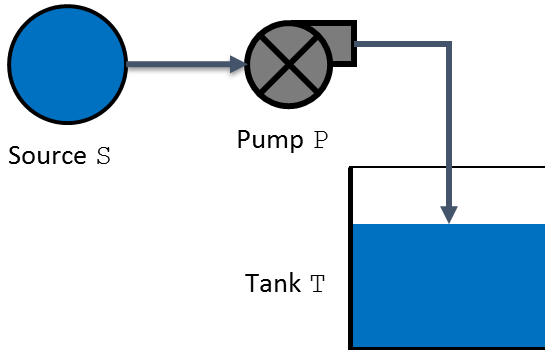
```

AR3Models/SimpleWaterSupplySystem/GTSMODELS/
SimpleWaterSupplySystem.gts
cwd C:/DataMB/OpenAltaRica/Development/AR3Models/
SimpleWaterSupplySystem
simulation started
GTSSStepSim >>

Available commands:
-----
h(elp)?           print this debug help.
print gts        print the simulated GTS model.
print v(ariables)? print the variables and their values.
print o(bservers)? print the observers and their values.
print t(ransitions)? print the transitions fireable in the
current state.
print h(istory)?  print the execution history.
fire <index>      fire the transition number <index>.
fire <name>       fire the transition named <name>.
back            cancel the last transition firing.
restart         restart the simulation.
echo <aString>   print <aString> (it can contain spaces).
set (no)? trace turn on/off the trace mode.
set (no)? display turn on/off the display mode.
set mode <normal|advanced> turn the mode to normal or
advanced.
q(uit)?         quit.
GTSSStepSim >>
  
```



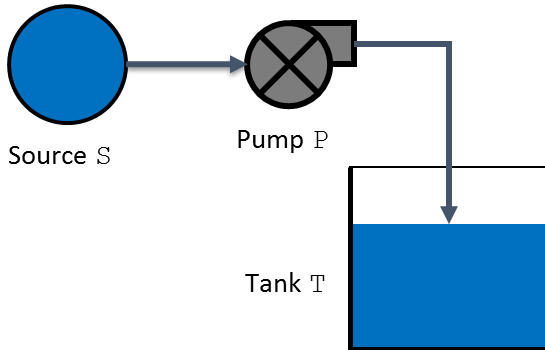
## D. To generate and study a fault tree



## 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

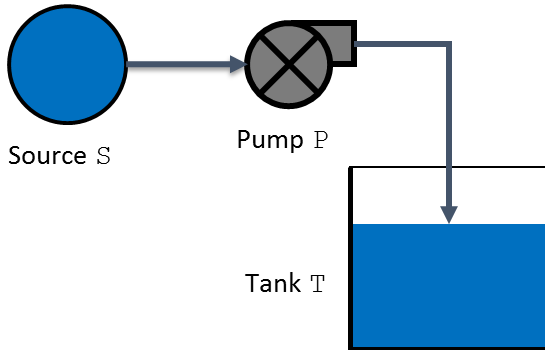
2. Launch the compiler to fault trees and analyze the generated fault tree:

- 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 observe ('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'.

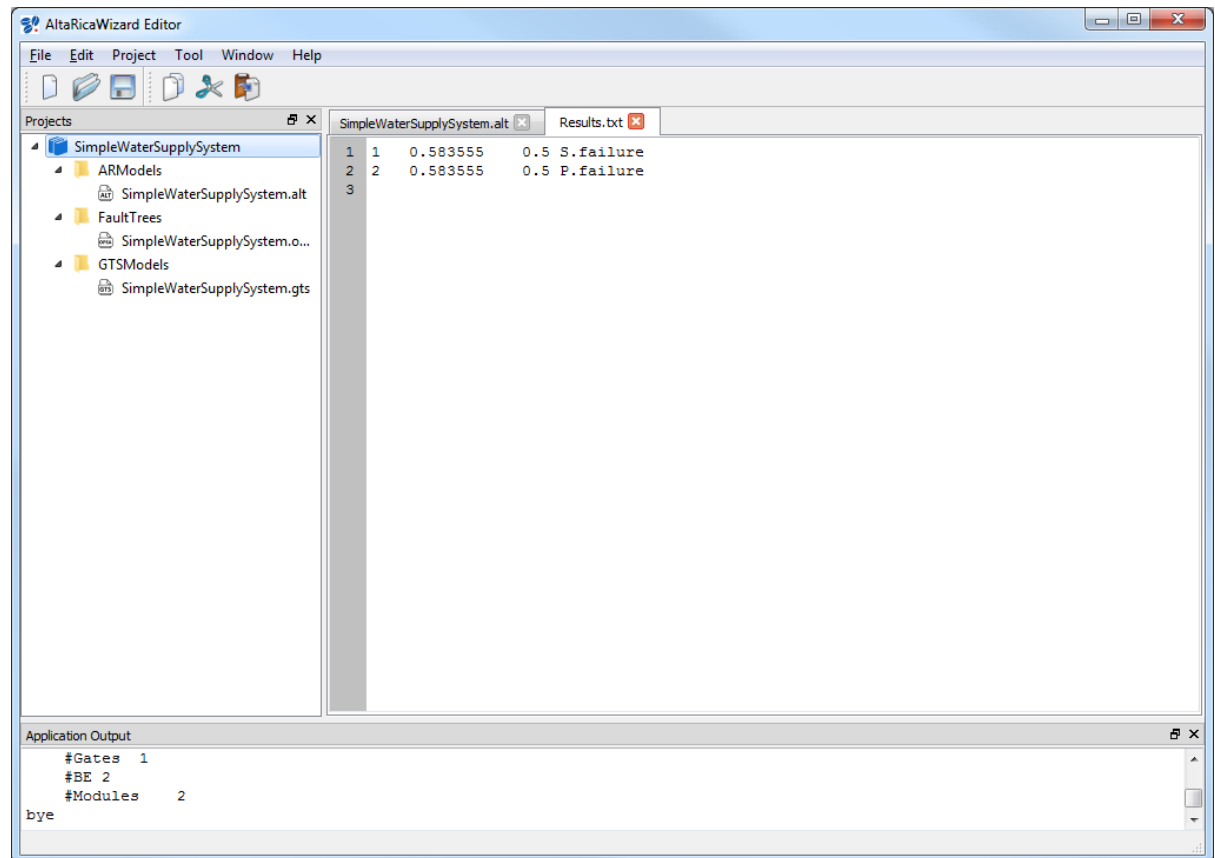


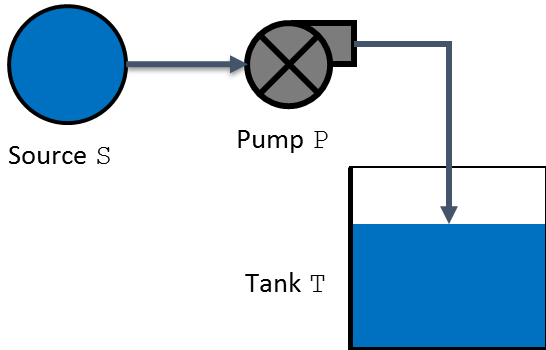


### D. To generate a fault tree

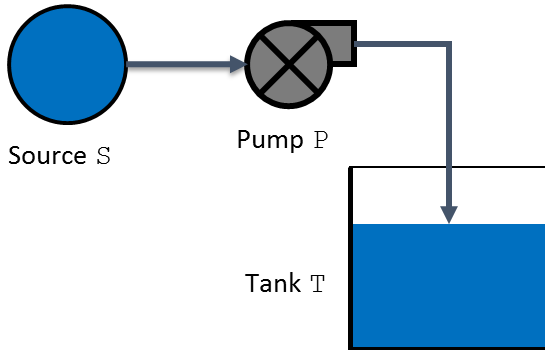
#### 2. Open results:

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





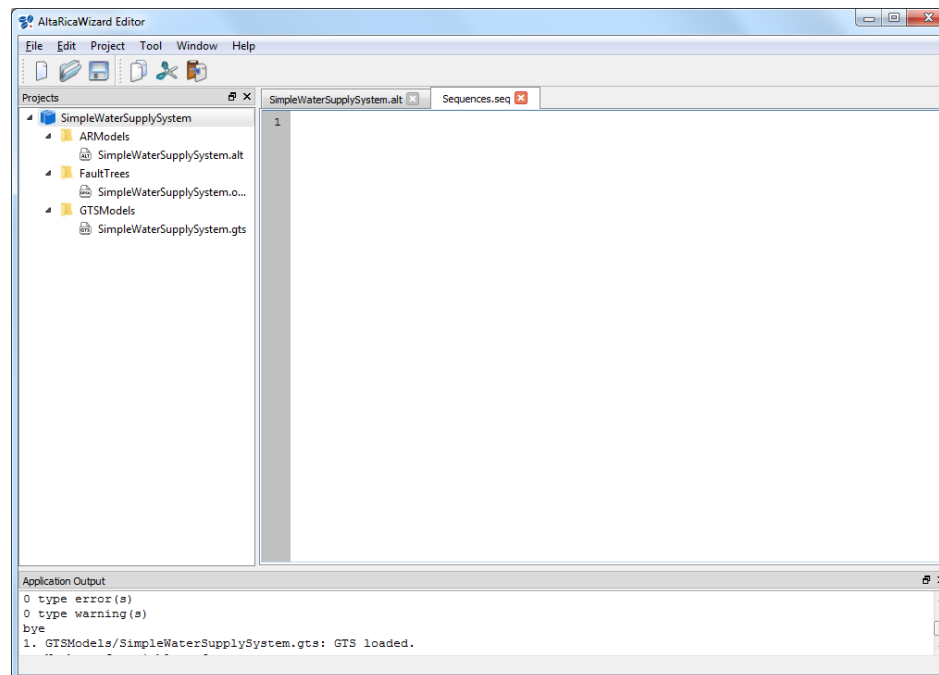
## E. To script simulation

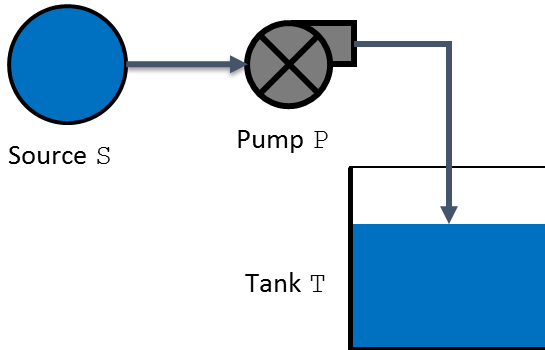


### E. To script simulation

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.*

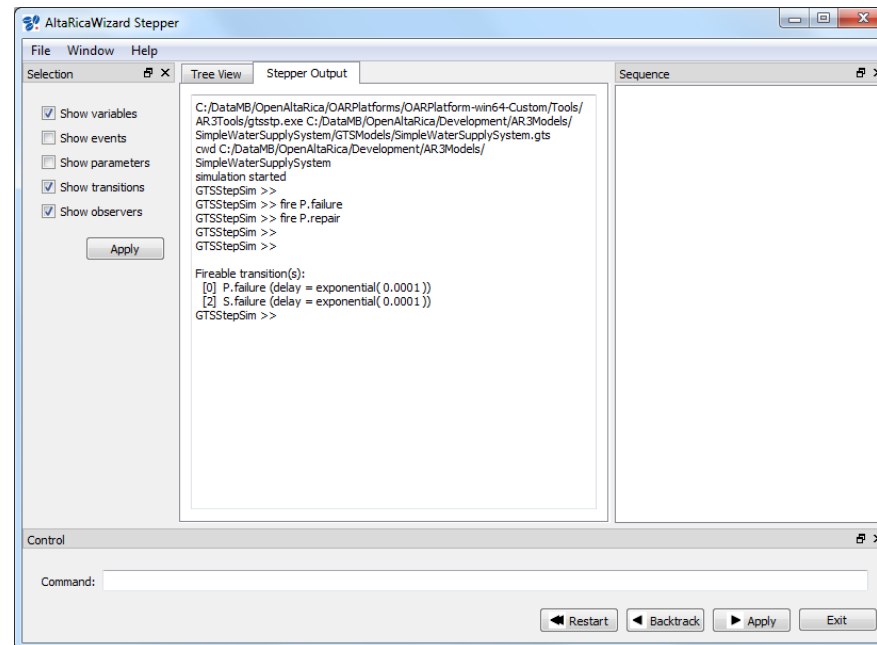




## E. To script simulation

### 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](http://www.openaltarica.fr)

[contact.oar@irt-systemx.fr](mailto:contact.oar@irt-systemx.fr)

[www.irt-systemx.fr](http://www.irt-systemx.fr)

