DEVS Visual Modeling and Simulation Environment

Hongyan Song

MSDL, McGill University

August, 2005

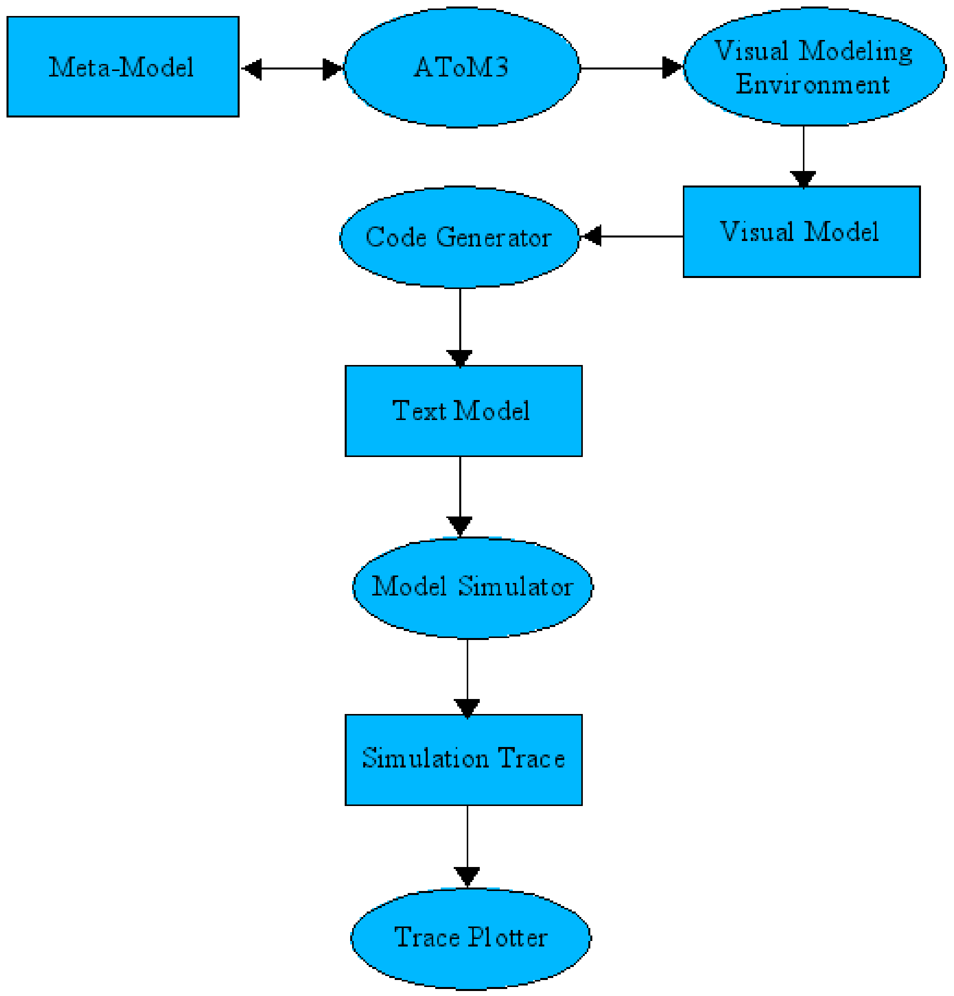
Outline

- Motivations and Purposes
- The Architecture
- DEVS Meta-Model
- Visual Modeling Environment
- Code Generator
- Simulation Environment and Trace Plotter

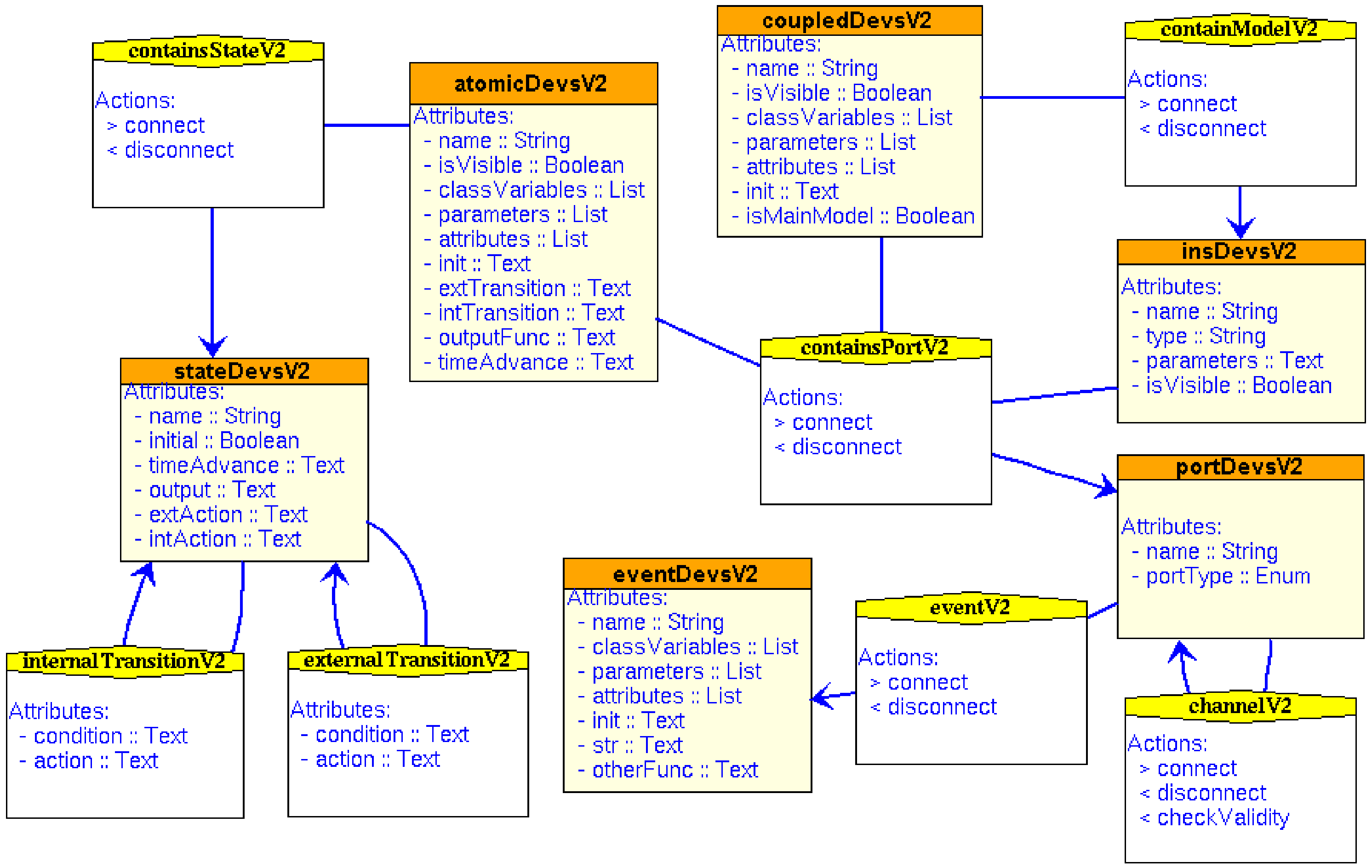
Motivations and Purposes

- Meta-model the DEVS formalism
- Visualize the Modeling and Simulation Process
- Visualize the Simulation Trace
- Exploring techniques of building integrated visual modeling and simulation environment

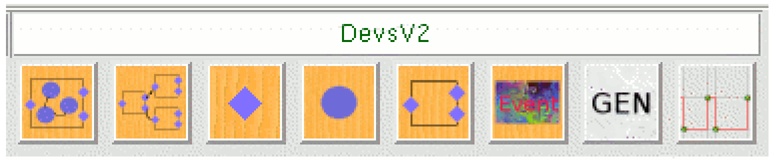
The Architecture



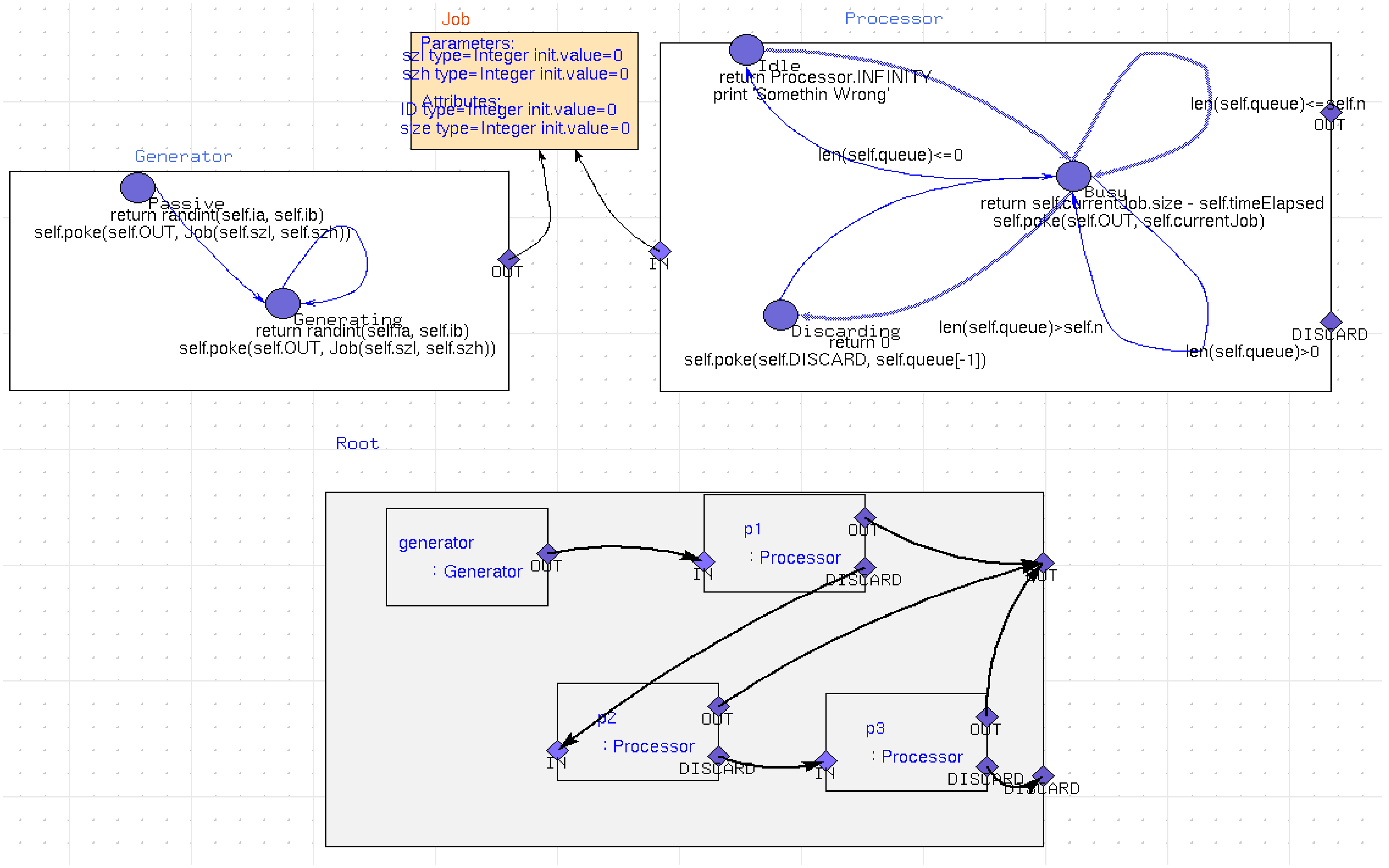
DEVS Meta-Model



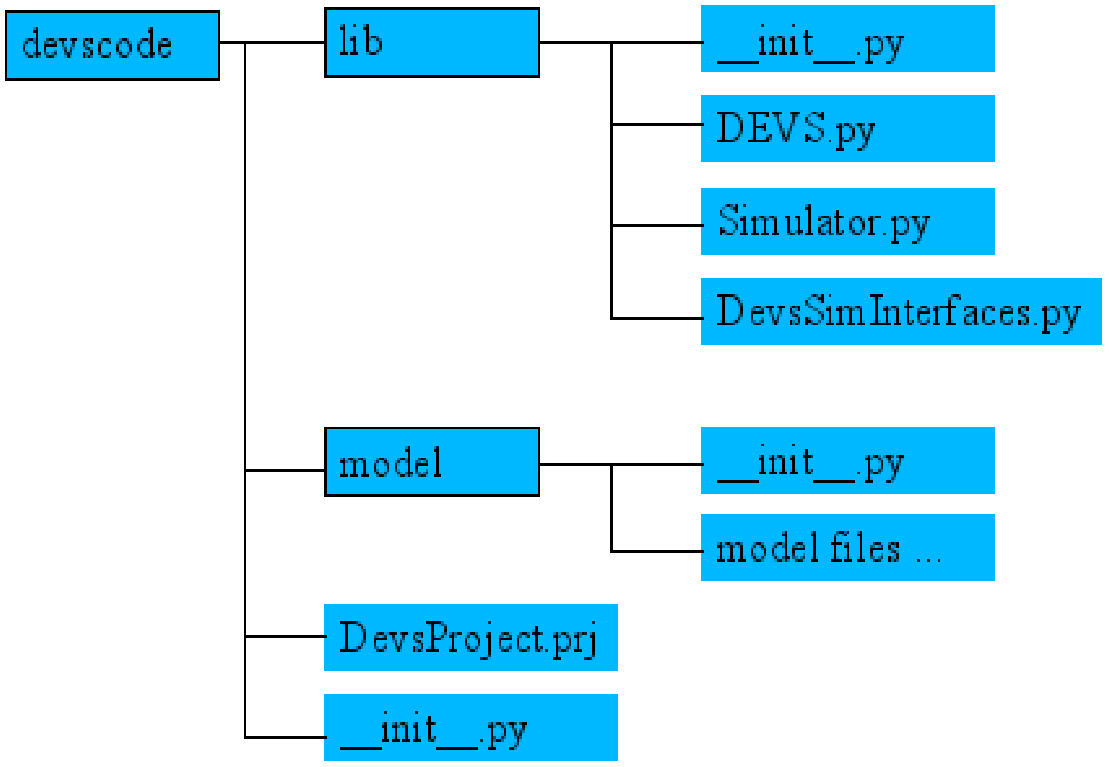
Modeling Environment



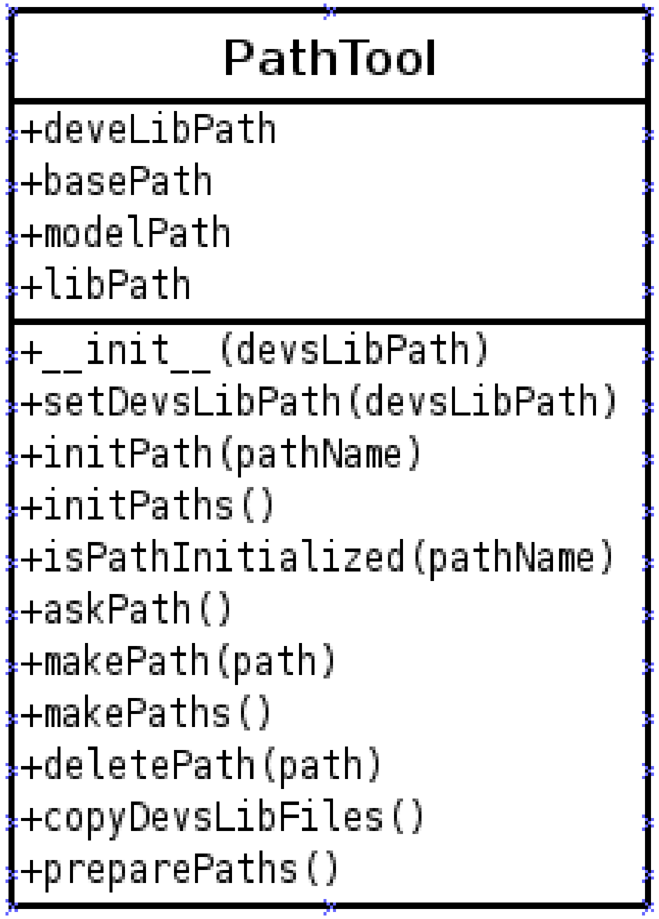
The Queue Example


Job
Parameters:
szl type=Integer init.value=0
szh type=Integer init.value=0
Attributes:
ID type=Integer init.value=0
size type=Integer init.value=0
Generator
Passive
return randint(self.ia, self.ib)
self.poke(self.OUT, Job(self.szl, self.szh))
Generating
return randint(self.ia, self.ib)
self.poke(self.OUT, Job(self.szl, self.szh))
OUT
Processor
Idle
return Processor.INFINITY
print 'Somethin Wrong'
len(self.queue)<=self.n
OUT
len(self.queue)<=0
Busy
return self.currentJob.size - self.timeElapsed
self.poke(self.OUT, self.currentJob)
IN
Discarding
return 0
self.poke(self.DISCARD, self.queue[-1])
len(self.queue)>self.n
len(self.queue)>0
DISCARD
Root
generator
: Generator
OUT
p1
: Processor
OUT
IN
DISCARD
OUT
p2
: Processor
IN
OUT
DISCARD
p3
: Processor
IN
OUT
DISCARD
DISCARD

File Structure



Path Management Tool

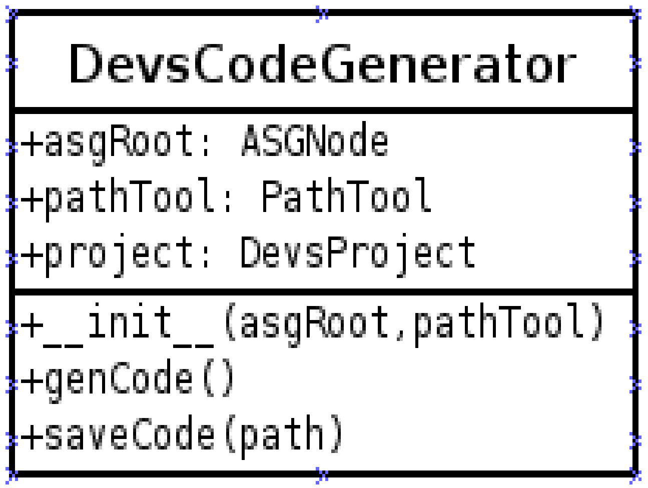


- Make the file structure
- Initialize the paths with __init__.py
- Copy DEVS model templates and simulator to the lib path
- Clean the paths when generate new project

Code Generators

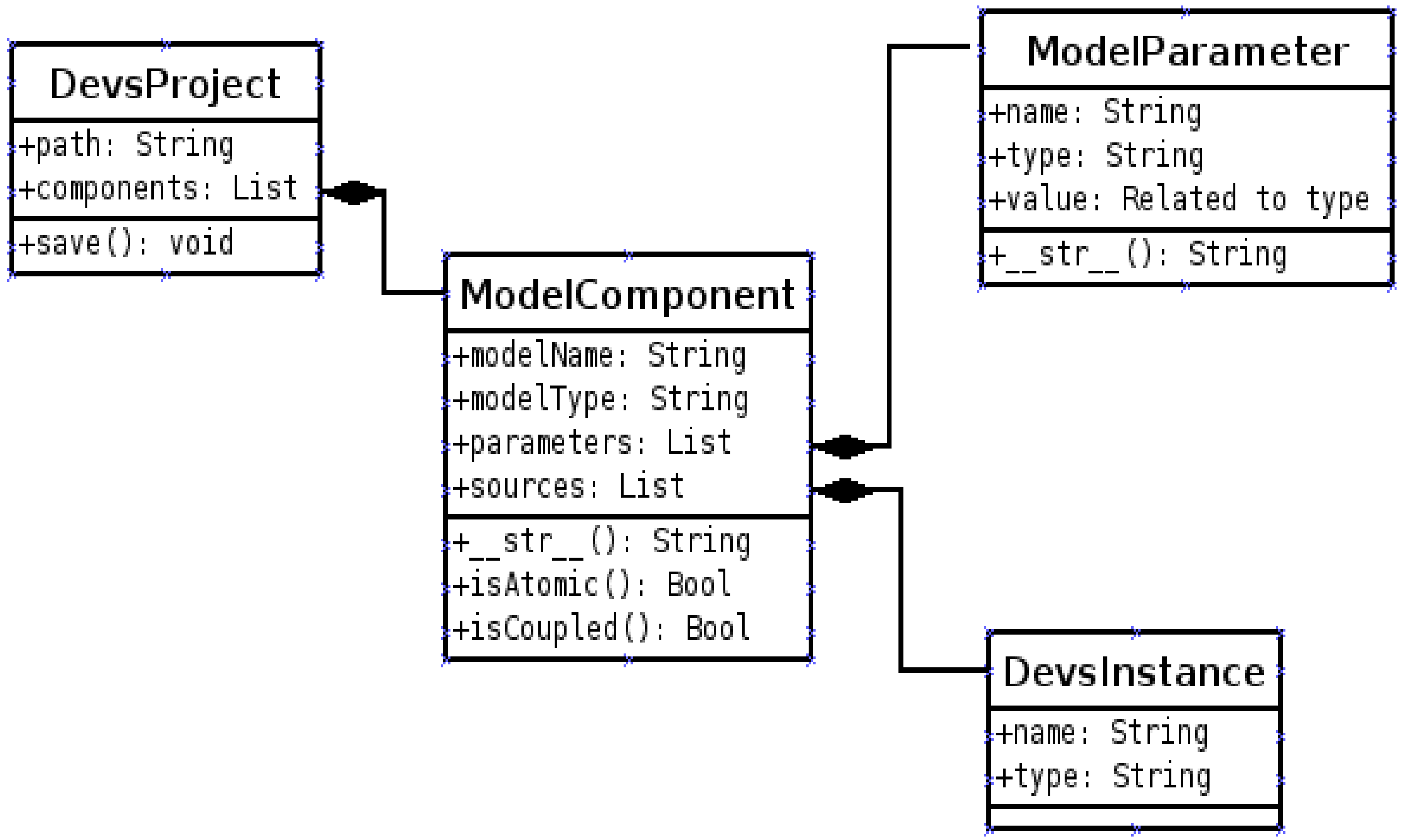
- Project Generator
- Atomic Code Generator
- Coupled Code Generator
- Event Code Generator

Project Generator

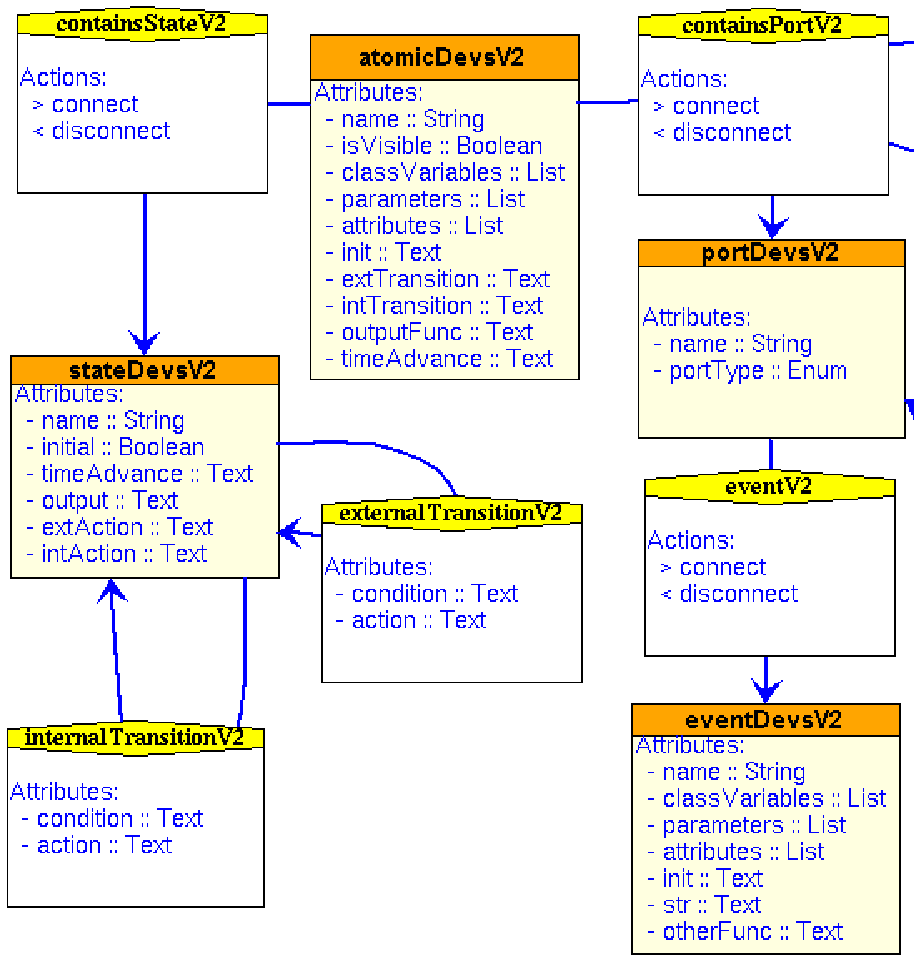


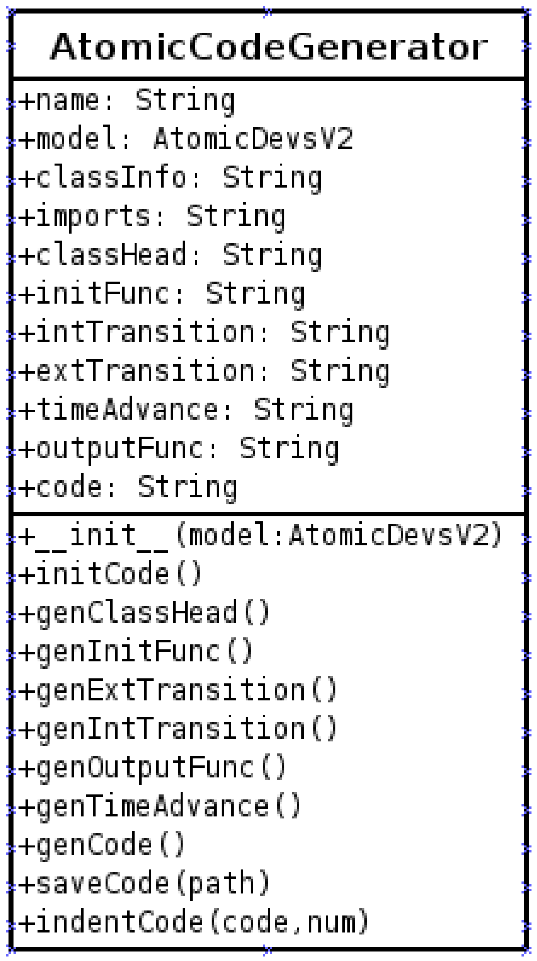
- Get the asgRoot
- Prepare the file structure using pathTool
- Generate DevsProject.prj

Project Data Structure



Atomic Code Generator





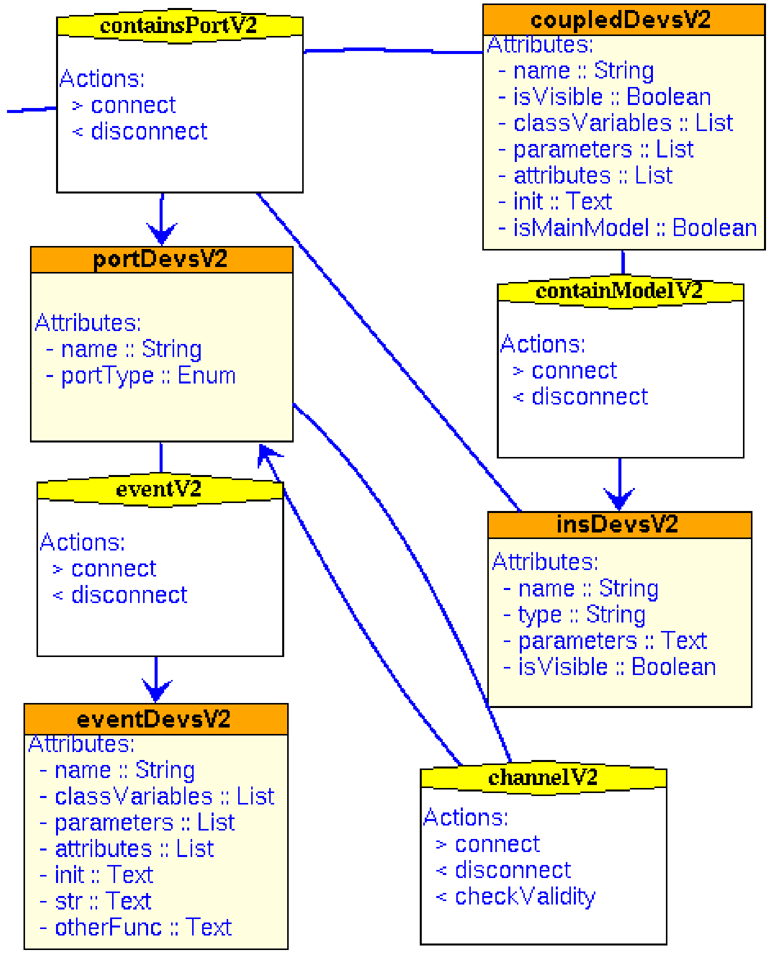
Atomic Example - Processor.py

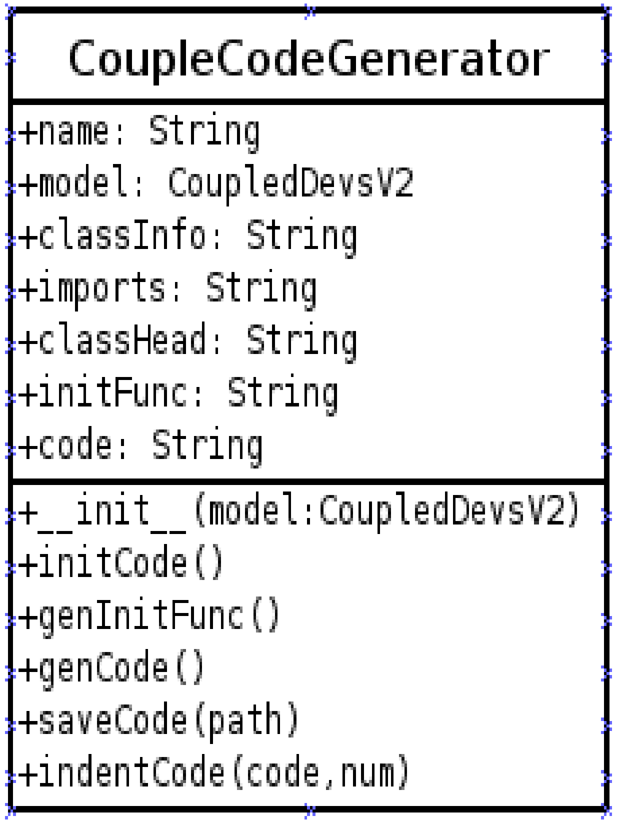
```
# AtomicDEVS model: Processor


from lib.DEVS import *
from lib.Simulator import *
from whrandom import *
from Job import *

class Processor(AtomicDEVS):
  Idle='Idle'
  Busy='Busy'
  Discarding='Discarding'
  INFINITY = 1000000

  def __init__(self, n):
    AtomicDEVS.__init__(self)
    self.n = n
    self.queue = []
    self.currentJob = None
    self.timeElapsed = 0.0
    self.state = Processor.Idle
    self.IN = self.addInPort()
    self.IN.instName = 'IN'
    self.IN.instType = 'InPort'
    self.OUT = self.addOutPort()
    self.OUT.instName = 'OUT'
    self.OUT.instType = 'OutPort'
    self.DISCARD = self.addOutPort()
    self.DISCARD.instName = 'DISCARD'
    self.DISCARD.instType = 'OutPort'
```

```
def intTransition(self):
  if(self.state==Processor.Busy):
    self.timeElapsed = 0.0
    if(len(self.queue)<=0):
      self.currentJob = None
      return Processor.Idle
    elif(len(self.queue)>0):
      self.currentJob = self.queue[0]
      self.queue = self.queue[1:]
      return Processor.Busy
  elif(self.state==Processor.Discarding):
    self.queue=self.queue[:-1]
    return Processor.Busy

def extTransition(self):
  p = self.peek(self.IN)
  self.queue.append(p)
  if(self.state==Processor.Idle):
    self.currentJob = self.queue[0]
    self.queue = self.queue[1:]
    return Processor.Busy
  elif(self.state==Processor.Busy):
    self.timeElapsed = self.timeElapsed + self.elapsed
    if(len(self.queue)>self.n):
      return Processor.Discarding
    elif(len(self.queue)<=self.n):
      return Processor.Busy
```

```
def timeAdvance(self):
  if(self.state==Processor.Idle):
    return Processor.INFINITY
  elif(self.state==Processor.Busy):
    return self.currentJob.size - self.timeElapsed
  elif(self.state==Processor.Discarding):
    return 0

def outputFnc(self):
  if(self.state==Processor.Idle):
    print 'Somethin Wrong'
  elif(self.state==Processor.Busy):
    self.poke(self.OUT, self.currentJob)
  elif(self.state==Processor.Discarding):
    self.poke(self.DISCARD, self.queue[-1])
```

Coupled Code Generator



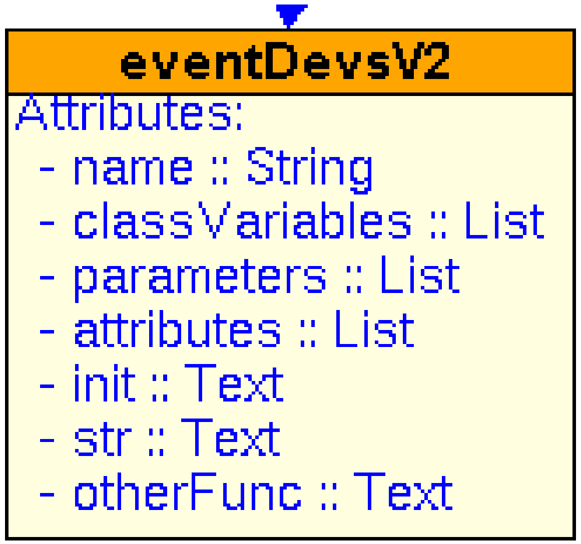


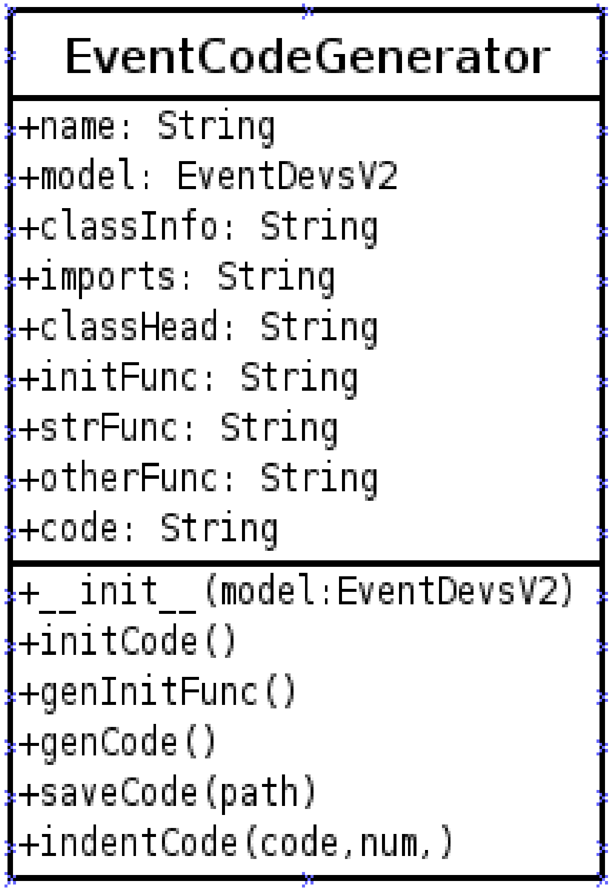
```
CoupleCodeGenerator
+name: String
+model: CoupledDevsV2
+classInfo: String
+imports: String
+classHead: String
+initFunc: String
+code: String
+__init__(model:CoupledDevsV2)
+initCode()
+genInitFunc()
+genCode()
+saveCode(path)
+indentCode(code,num)
```

Coupled Example - Root.py

```
# CoupledDEVS model: Root


from lib.DEVS import *
from lib.Simulator import *
from whrandom import *
from Generator import *
from Processor import *

class Root(CoupledDEVS):

  def __init__(self, qs, ia, ib, sa, sb):
    CoupledDEVS.__init__(self)
    self.qs = qs
    self.ia = ia
    self.ib = ib
    self.sa = sa
    self.sb = sb
    self.OUT = self.addOutPort()
    self.OUT.instName = 'OUT'
    self.OUT.instType = 'OutPort'
    self.DISCARD = self.addOutPort()
    self.DISCARD.instName = 'DISCARD'
    self.DISCARD.instType = 'OutPort'
    self.generator = self.addSubModel(Generator(ia=self.ia, ib=self.ib, szl=self.sa, szh=self.sb))
    self.generator.instName = 'generator'
    self.generator.instType = 'Generator'
    self.p1 = self.addSubModel(Processor(self.qs))
    self.p1.instName = 'p1'
    self.p1.instType = 'Processor'
    self.p2 = self.addSubModel(Processor(self.qs))
    self.p2.instName = 'p2'
    self.p2.instType = 'Processor'
    self.p3 = self.addSubModel(Processor(self.qs))
    self.p3.instName = 'p3'
    self.p3.instType = 'Processor'
    self.connectPorts(self.generator.OUT, self.p1.IN)
    self.connectPorts(self.p1.OUT, self.OUT)
    self.connectPorts(self.p1.DISCARD, self.p2.IN)
    self.connectPorts(self.p2.OUT, self.OUT)
    self.connectPorts(self.p2.DISCARD, self.p3.IN)
    self.connectPorts(self.p3.OUT, self.OUT)
    self.connectPorts(self.p3.DISCARD, self.DISCARD)
```

Event Code Generator

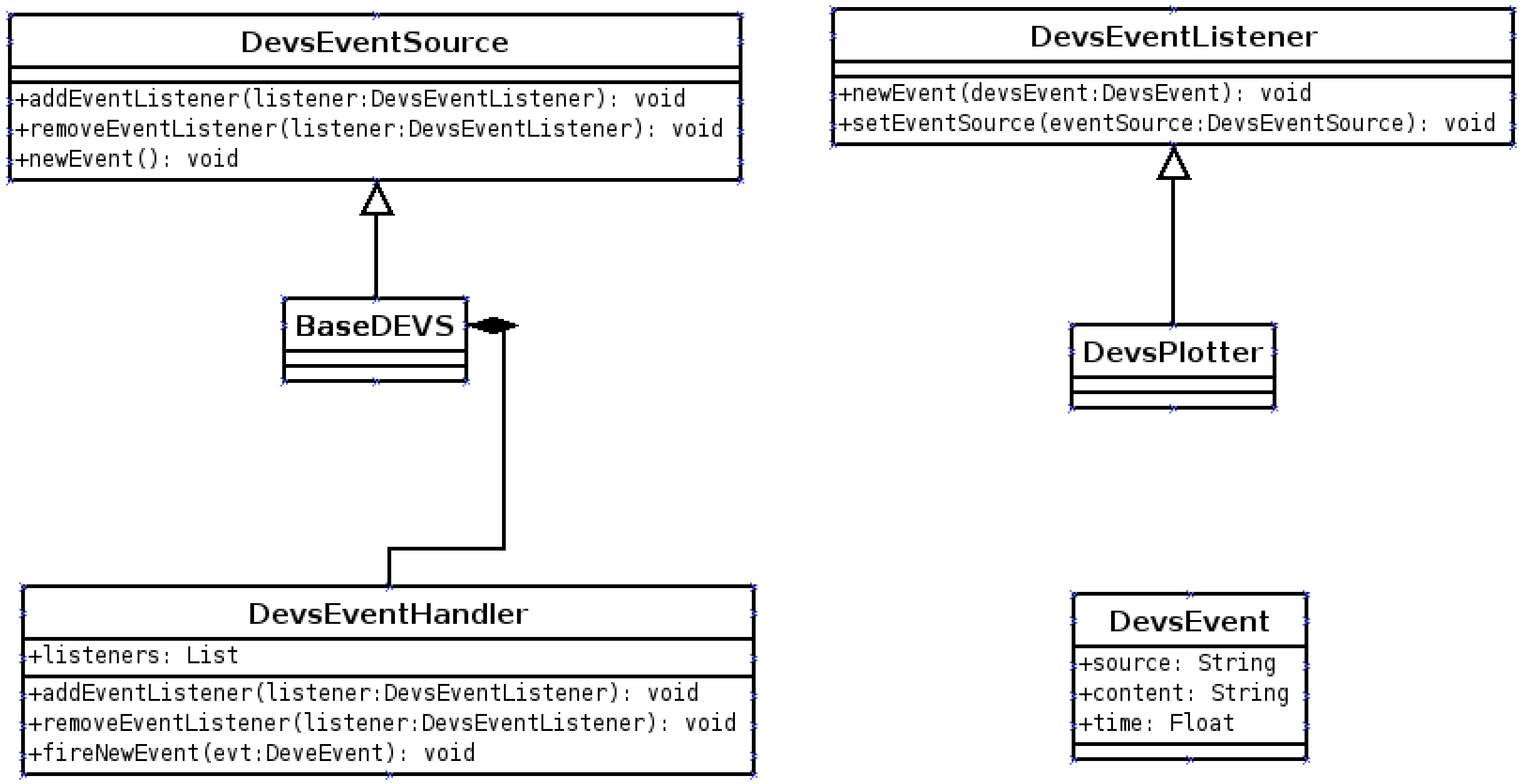




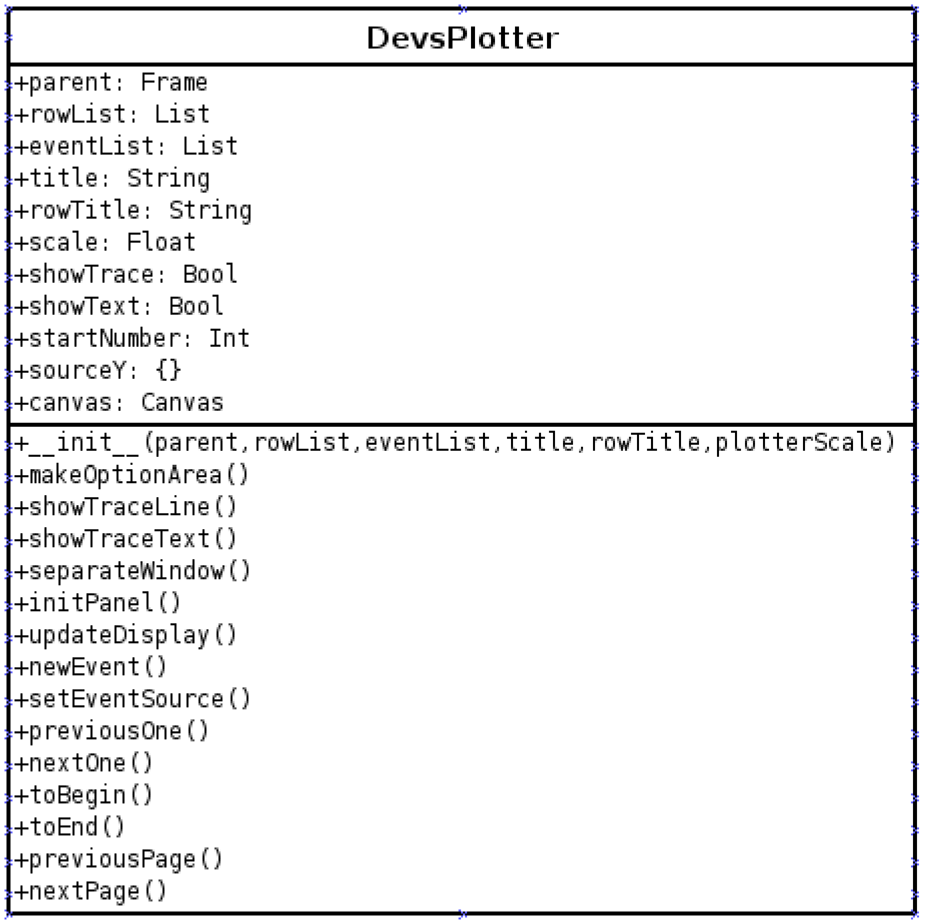
Event Example

```
# Event: Job

from whrandom import *

class Job(object):
  IDCounter = 0

  def __init__(self, szl, szh):
    self.szl = szl
    self.szh = szh
    self.ID = 0
    self.size = 0
    self.ID = Job.IDCounter = Job.IDCounter + 1
    self.size = randint(self.szl, self.szh)

  def __str__(self):
    return "(job %d, size %d)" % (self.ID, self.size)
```

Interface for Model and Plotter


DevsEventSource
+addEventListener(listener:DevsEventListener): void
+removeEventListener(listener:DevsEventListener): void
+newEvent(): void
DevsEventListener
+newEvent(devsEvent:DevsEvent): void
+setEventSource(eventSource:DevsEventSource): void
BaseDEVS
DevsPlotter
DevsEventHandler
+listeners: List
+addEventListener(listener:DevsEventListener): void
+removeEventListener(listener:DevsEventListener): void
+fireNewEvent(evt:DeveEvent): void
DevsEvent
+source: String
+content: String
+time: Float

The Plotter

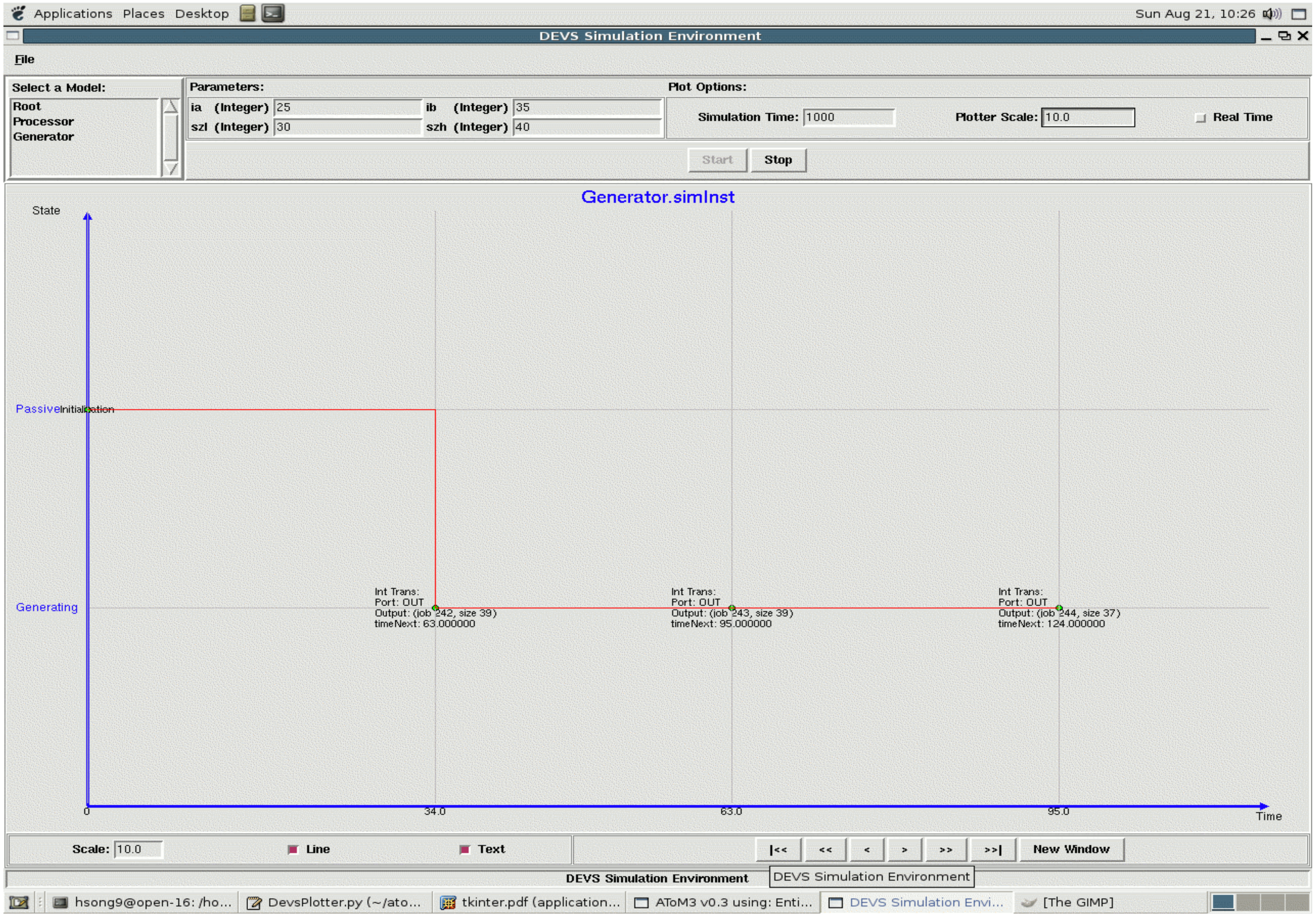


DevsPlotter

```
+parent: Frame
+rowList: List
+eventList: List
+title: String
+rowTitle: String
+scale: Float
+showTrace: Bool
+showText: Bool
+startNumber: Int
+sourceY: {}
+canvas: Canvas
```

```
+__init__(parent,rowList,eventList,title,rowTitle,plotterScale)
+makeOptionArea()
+showTraceLine()
+showTraceText()
+separateWindow()
+initPanel()
+updateDisplay()
+newEvent()
+setEventSource()
+previousOne()
+nextOne()
+toBegin()
+toEnd()
+previousPage()
+nextPage()
```

Simulation Environment



Select a Model:
Root
Processor
Generator

Parameters:

ia (Integer)	25	ib (Integer)	35
szl (Integer)	30	szh (Integer)	40

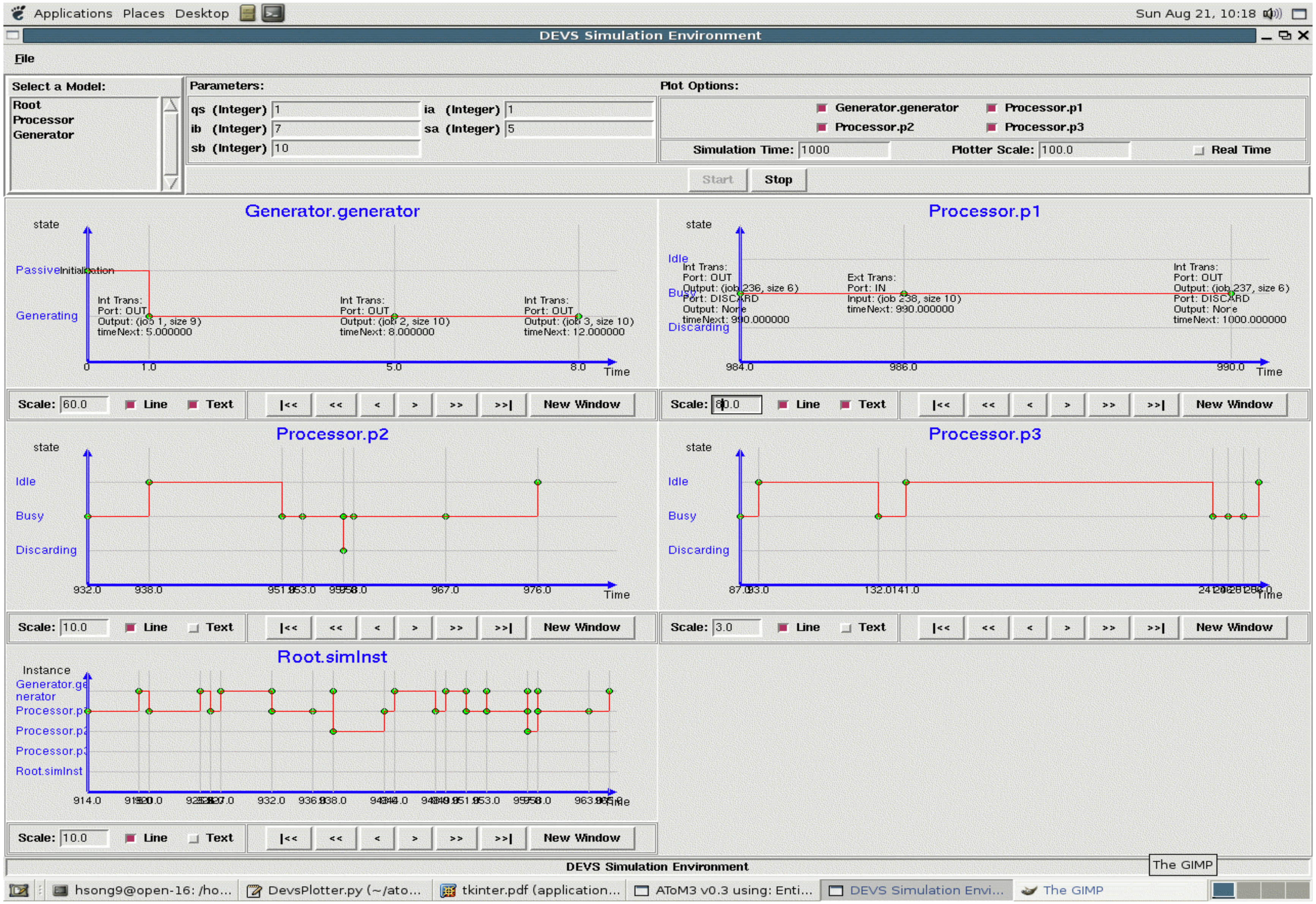
Plot Options:
Simulation Time: 1000
Plotter Scale: 10.0

```
Int Trans:
Port: OUT
Output: (job 242, size 39)
timeNext: 63.000000

Int Trans:
Port: OUT
Output: (job 243, size 39)
timeNext: 95.000000

Int Trans:
Port: OUT
Output: (job 244, size 37)
timeNext: 124.000000
```

Scale: 10.0

DEVS Simulation Environment

Simulation Environment - continue


Applications Places Desktop
Sun Aug 21, 10:18
DEVS Simulation Environment
File
Start
Stop
|<< << < > >> >>| New Window
state
Time
Instance
hsong9@open-16: /ho...
DevsPlotter.py (~/ato...
tkinter.pdf (application...
AToM3 v0.3 using: Enti...
DEVS Simulation Envi...
The GIMP

Select a Model:
Root
Processor
Generator

Parameters:

qs (Integer)	1	ia (Integer)	1
ib (Integer)	7	sa (Integer)	5
sb (Integer)	10		

Plot Options:
🟥 Generator.generator 🟥 Processor.p1
🟥 Processor.p2 🟥 Processor.p3

Simulation Time: 1000 Plotter Scale: 100.0 ☐ Real Time

Generator.generator

Passive Initialization
Generating

```
Int Trans:
Port: OUT
Output: (job 1, size 9)
timeNext: 5.000000

Int Trans:
Port: OUT
Output: (job 2, size 10)
timeNext: 8.000000

Int Trans:
Port: OUT
Output: (job 3, size 10)
timeNext: 12.000000
```

0 1.0 5.0 8.0

Scale: 60.0 🟥 Line 🟥 Text

Processor.p1

Idle
Busy
Discarding

```
Int Trans:
Port: OUT
Output: (job 236, size 6)
Port: DISCARD
Output: None
timeNext: 990.000000

Ext Trans:
Port: IN
Input: (job 238, size 10)
timeNext: 990.000000

Int Trans:
Port: OUT
Output: (job 237, size 6)
Port: DISCARD
Output: None
timeNext: 1000.000000
```

984.0 986.0 990.0

Scale: 80.0 🟥 Line 🟥 Text

Processor.p2

Idle
Busy
Discarding

932.0 938.0 951.953.0 957958.0 967.0 976.0

Scale: 10.0 🟥 Line ☐ Text

Processor.p3

Idle
Busy
Discarding

87.093.0 132.0141.0 241.0...

Scale: 3.0 🟥 Line ☐ Text

Root.simInst

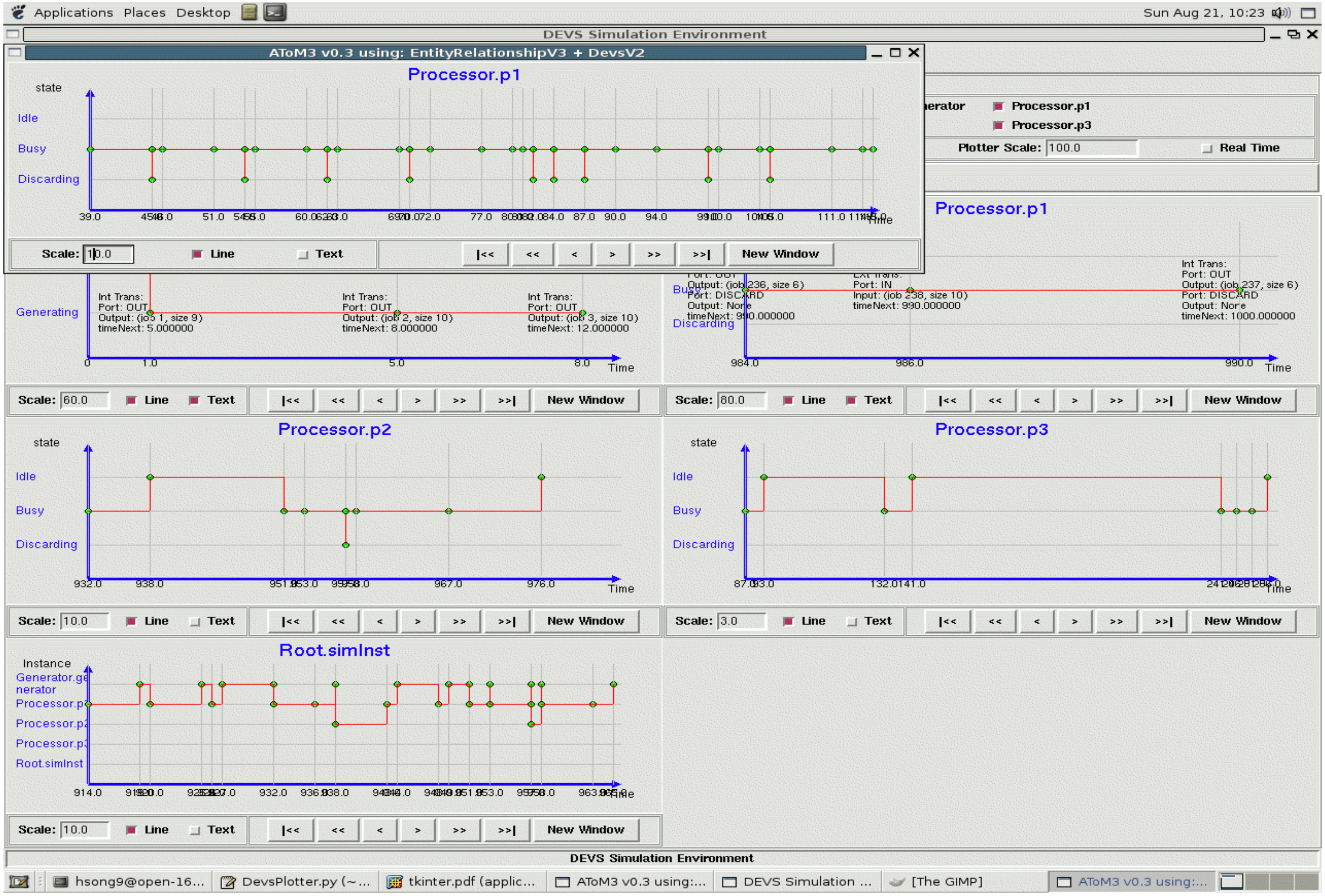
Generator.generator
Processor.p1
Processor.p2
Processor.p3
Root.simInst

914.0 919.920.0 925.926.927.0 932.0 936.938.0 943.944.0 948.949.951.953.0 957.958.0 963.965.0

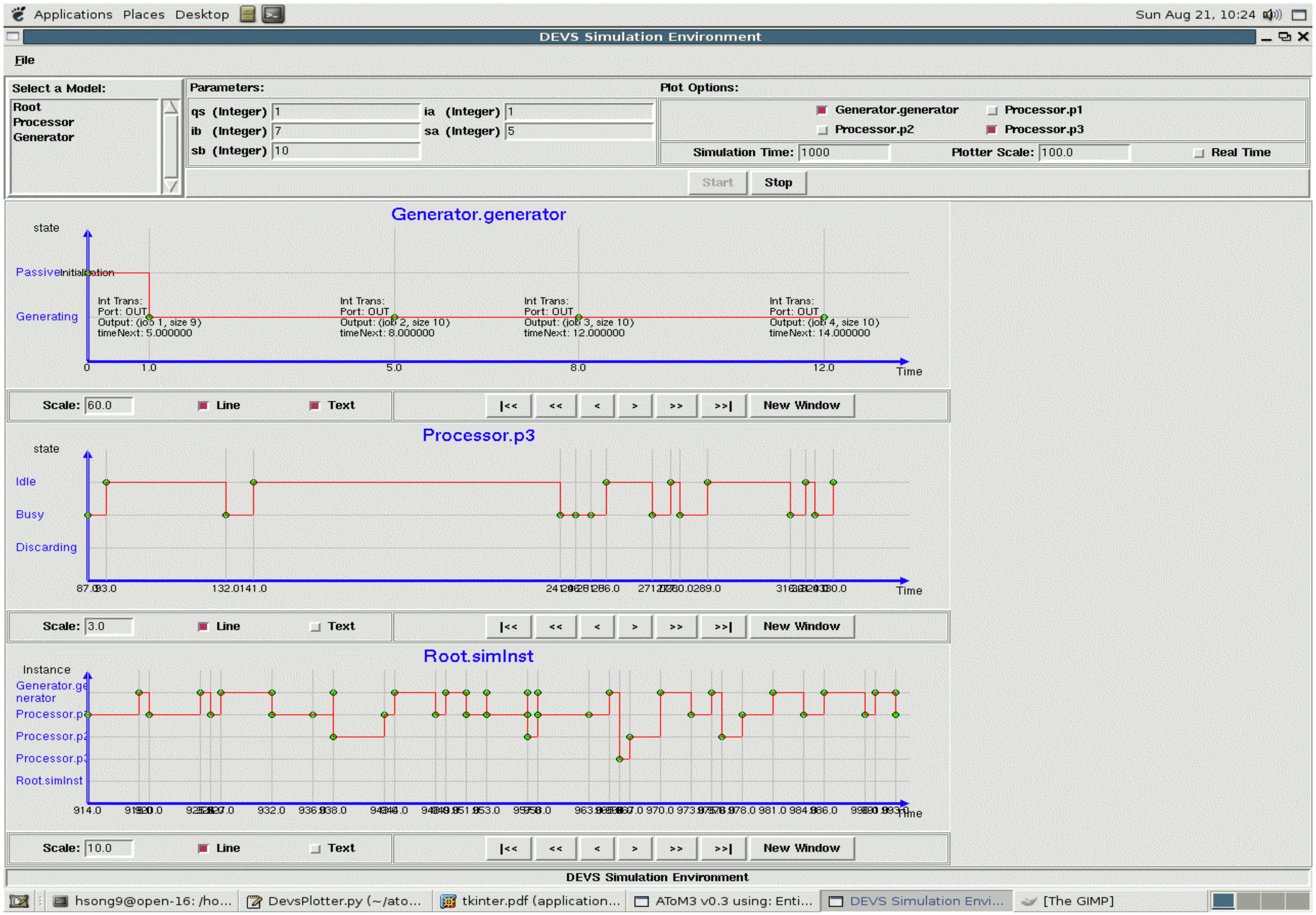
Scale: 10.0 🟥 Line ☐ Text

DEVS Simulation Environment

Simulation Environment – continue



Simulation Environment - continue



Select a Model:
Root
Processor
Generator

Parameters:

qs (Integer)	1	ia (Integer)	1
ib (Integer)	7	sa (Integer)	5
sb (Integer)	10		

Plot Options:
🟥 Generator.generator ☐ Processor.p1
☐ Processor.p2 🟥 Processor.p3
Simulation Time: 1000 Plotter Scale: 100.0 ☐ Real Time

```
Int Trans:
Port: OUT
Output: (job 1, size 9)
timeNext: 5.000000

Int Trans:
Port: OUT
Output: (job 2, size 10)
timeNext: 8.000000

Int Trans:
Port: OUT
Output: (job 3, size 10)
timeNext: 12.000000

Int Trans:
Port: OUT
Output: (job 4, size 10)
timeNext: 14.000000
```

Scale: 60.0 🟥 Line 🟥 Text

Scale: 3.0 🟥 Line ☐ Text

Scale: 10.0 🟥 Line ☐ Text

Future Works

- Continue to refine the current work
- Define a proper sub-modelica language for expressing DEVS
- Build a compiler for the sub-modelica language
- Generate Python code

Acknowledgment

- Many thanks for Hans' insightful guidance and valuable advices in the whole developing process
- Thanks for the original versions of Denis' DevsV2, Ernesto's DEVS Code Generator, and Jean-Sébastien's DEVS Templates and Simulator