



JEAN-CHRISTOPHE LE LANN
ENSEIGNANT-CHERCHEUR

EMBEDDED SYSTEMS, MODEL-DRIVEN ENGINEERING



From System-level models to heterogeneous embedded systems

Jean-Christophe Le Lann

Joel Champeau

Papa Issa Diallo

ENSTA-Bretagne / Labsticc

Pierre-Laurent

Lagalaye

Modaë Technologies



Overview

- Introduction
- Experimental toolchain
- Models of computation
- Result and discussion
- Conclusions



Introduction

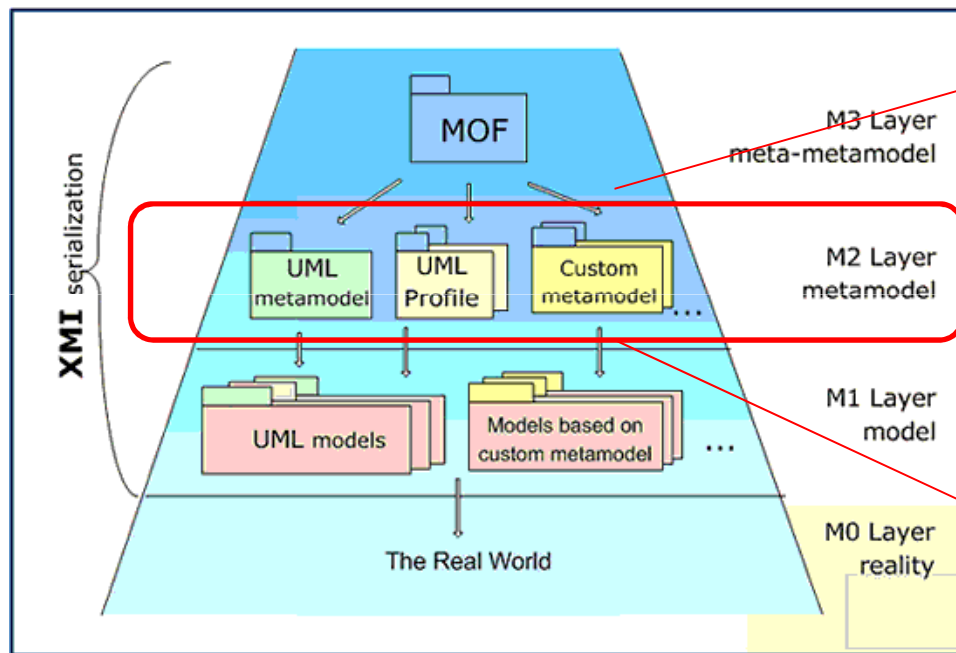
- History
 - 1999 : Thomson Multimedia / Technicolor
 - **System-level specification** for SoC design + **Mopcom ANR**
 - Video compression system
 - Multicore + SIMD ~15M gates
 - Needs :
 - ease of **algorithm capture** : data+ high-level control flow
 - Simulation : untimed, functional, data movements, events
 - Synthesis/compilation on heterogeneous platforms
 - Allowing quick iterations in the design flow
 - 2009 : startup Modae Technologies
 - **Interpreted languages** as input + DSL



New needs

- **System-engineering** practical aspects
 - IBM Rhapsody + UML 2.0 at the front
- **Software engineering** for embedded systems
 - Not only algorithmic, nor event-driven
 - Importance of object-oriented
- **Need for openness**
 - insurance of independence wrt tool providers
 - Quite different from classical ESL business-model
 - Facilitate toolchains development

Metamodeling for tool development

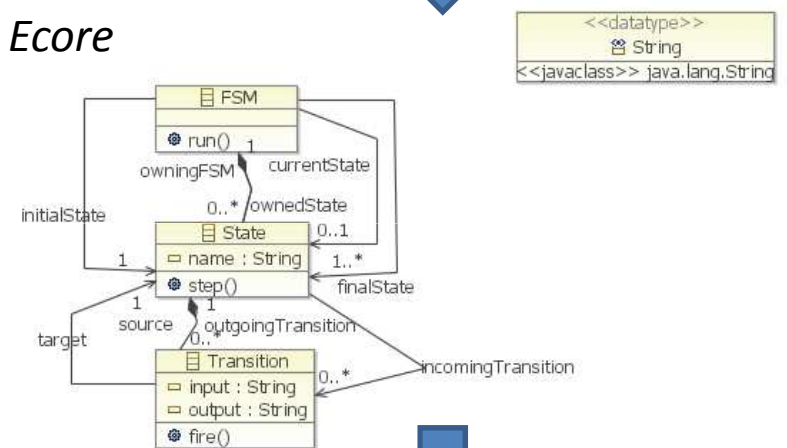


MDD : model-driven development

Eclipse EMF support
Sodius MDWorkbench, ...

Business knowledge

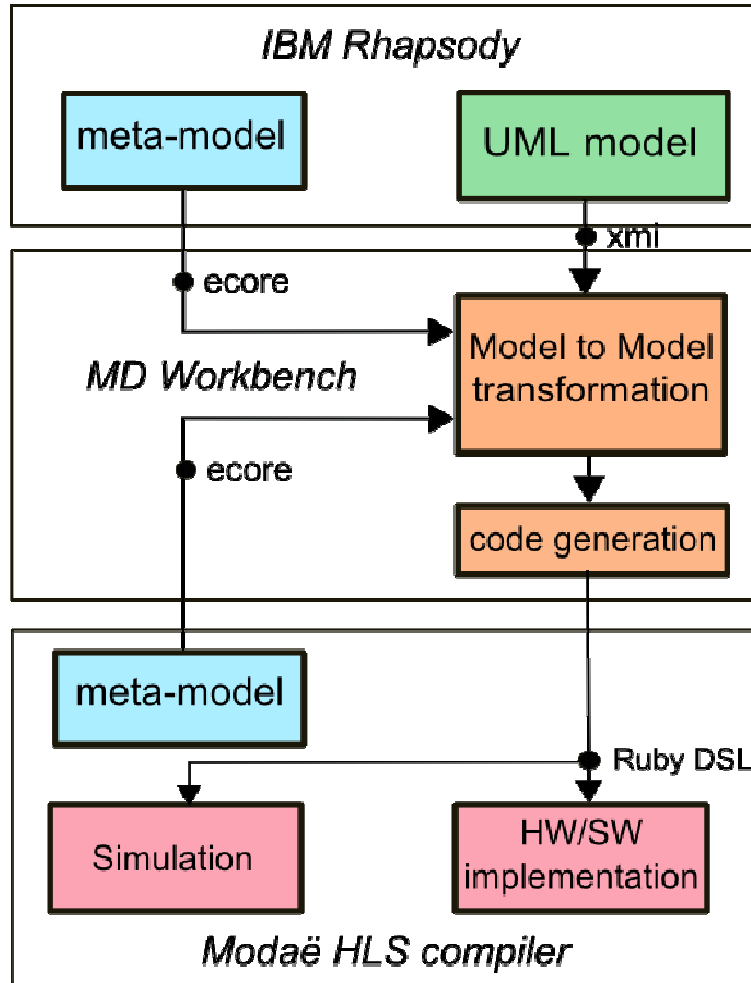
Ecore



Model edition

Code generation, ...

Experimental toolchain



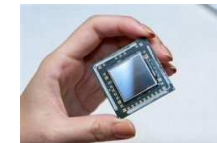
Modeling in UML 2.0



Transformation scripts
in MDWorkbench

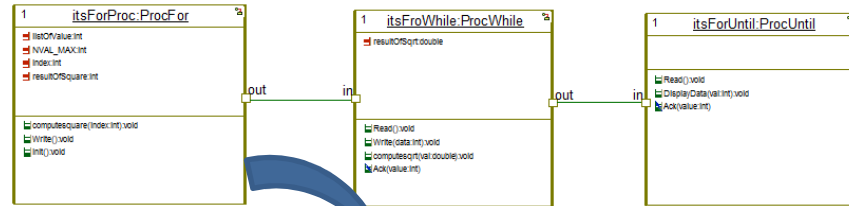


« Backend »
System-level synthesis
Modaë SLS

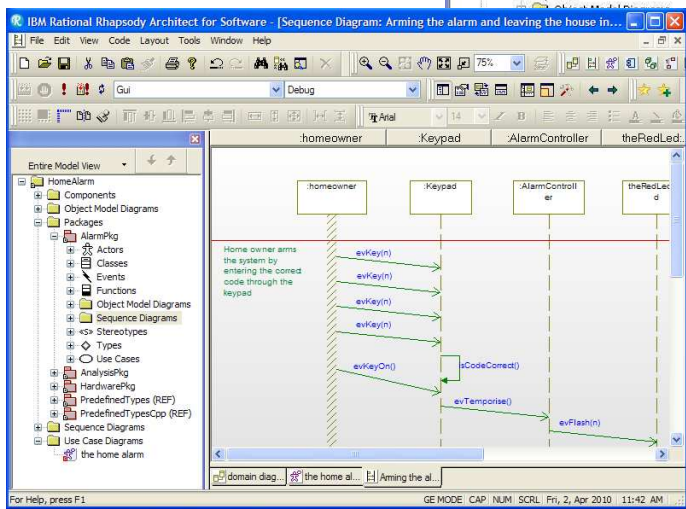
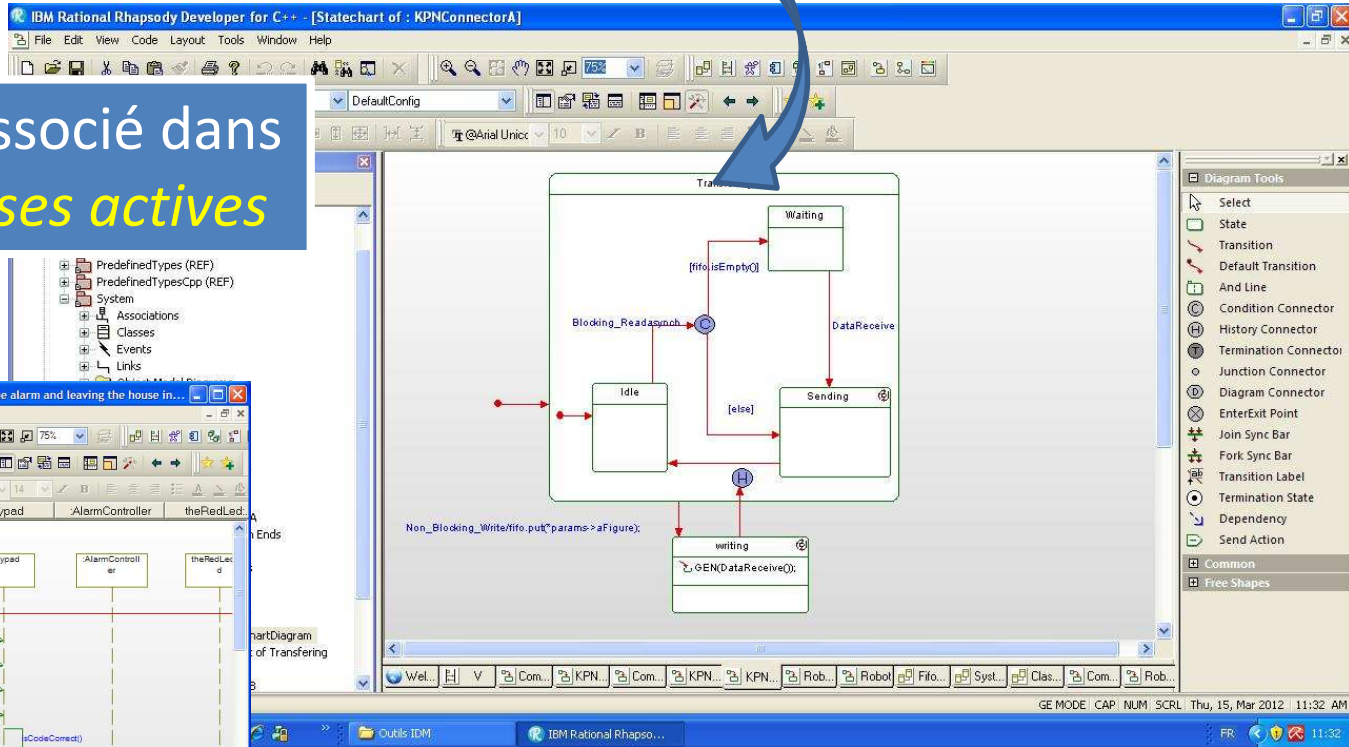


IBM Rhapsody UML 2.0

Diagramme de **composants**
(éventuellement composite)



Statecharts associé dans
le cas de **classes actives**



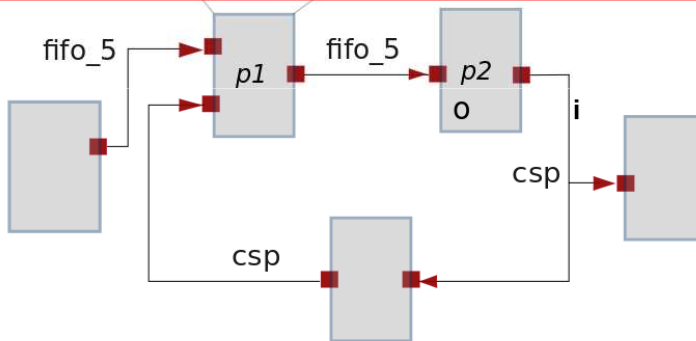
System modeling with *Modaë*



```
class MyProcessing < Reactive
  inports :i1,i2
  outports :o1
  def initialize x,y
    @x,@y = x,y
  end
  def behavior
  end
  def method_1
  end
  ...
end

x = receive :i1
while...
  ...
end
for i in 0..100
  if i>x
    ...
    send :o1,i
    ...
  end
end
```

Ruby/Python algorithms,
object-oriented

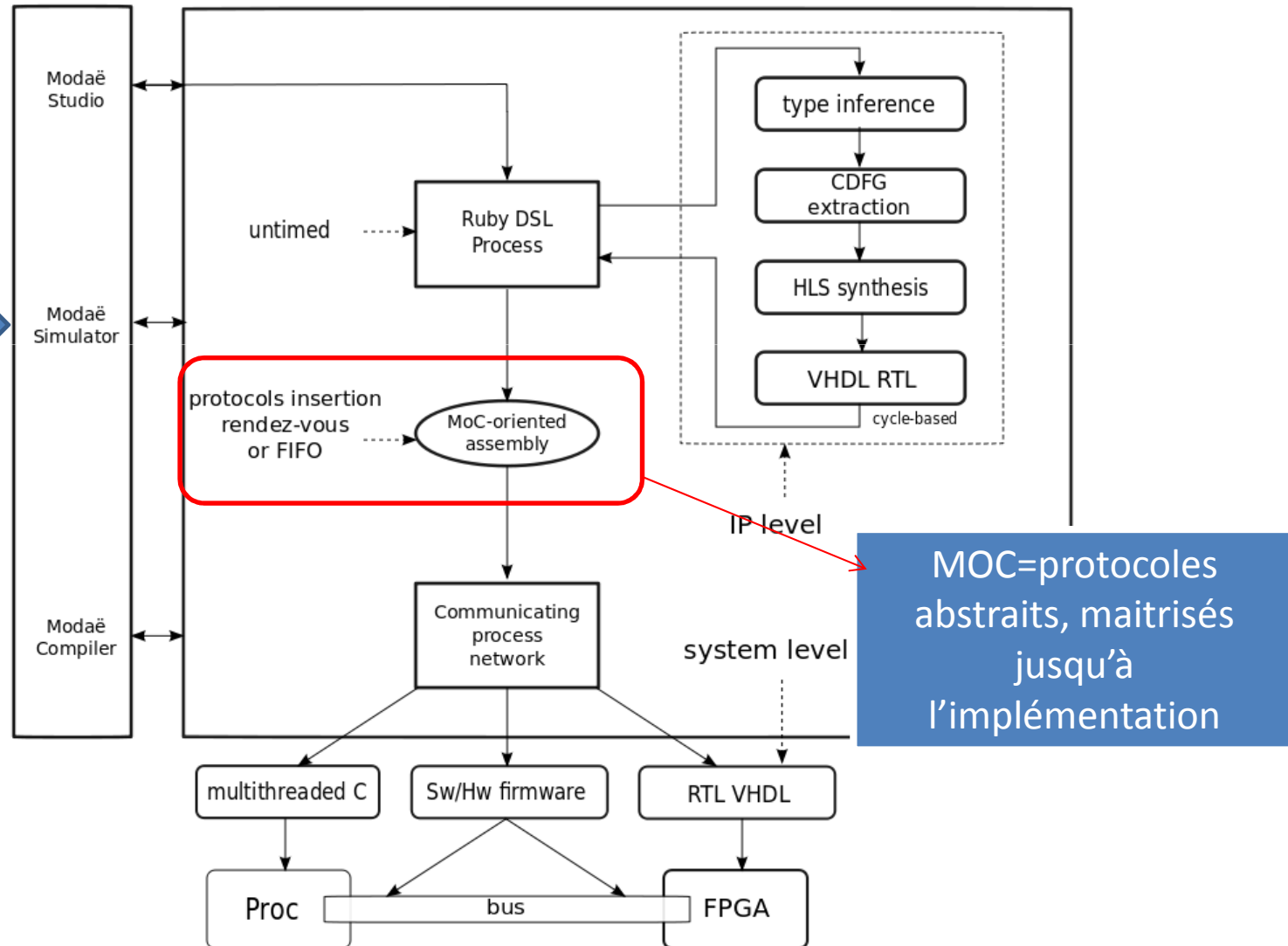
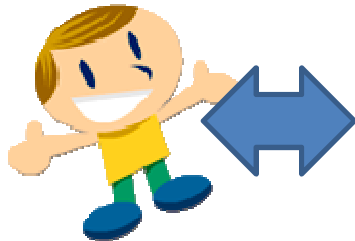


Addition of an internal DSL
...graphical

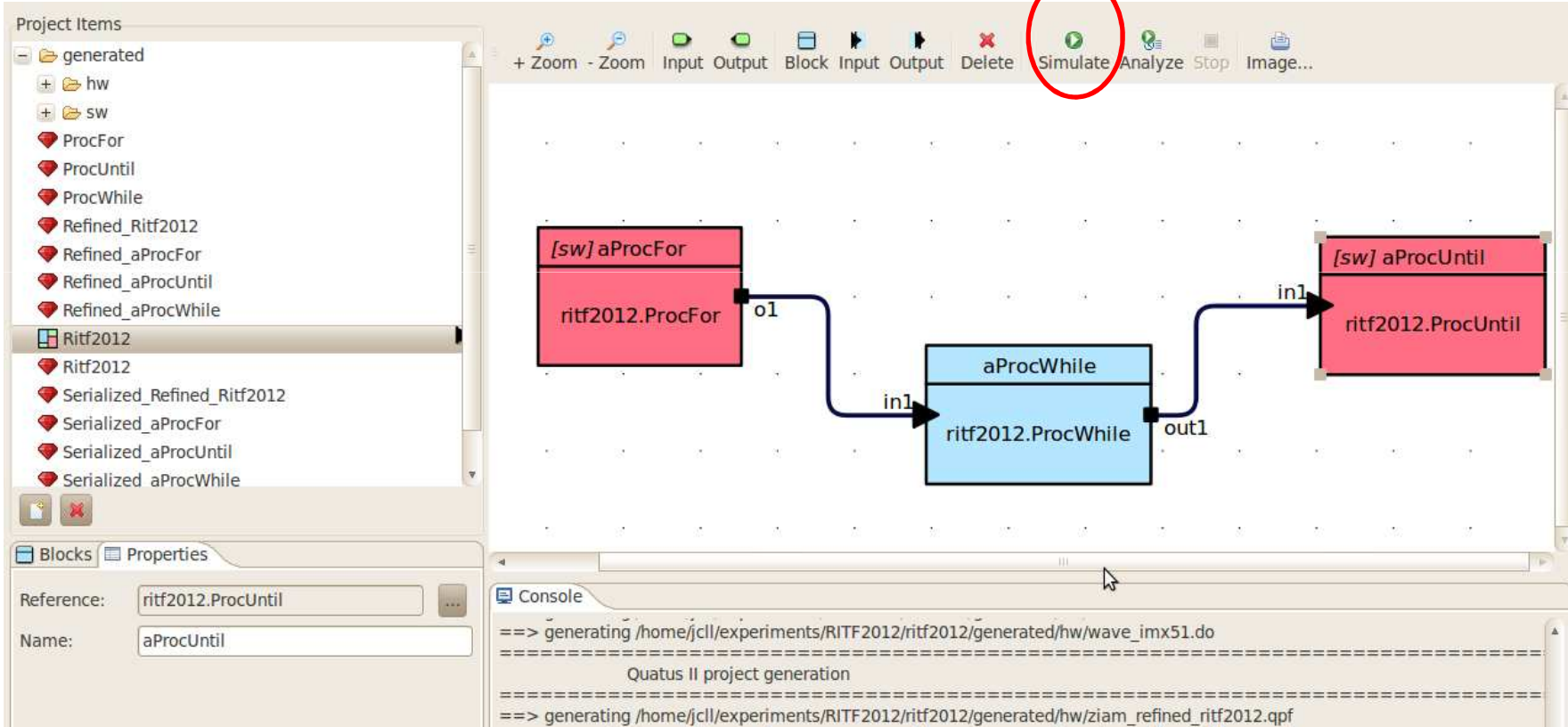
```
Network.new('example') do
  p1=MyProcessing.new('p1')
  p2=...
  ...
  connect :fifo_5, p1.o => p2.i
  ...
end
```

...textual

System modeling with *Modaë*



System modeling with *Modaë*



The screenshot displays the Modaë system modeling software interface. The main workspace shows a block diagram with three components: a pink block labeled `[sw] aProcFor` containing `ritf2012.ProcFor`, a blue block labeled `aProcWhile` containing `ritf2012.ProcWhile`, and another pink block labeled `[sw] aProcUntil` containing `ritf2012.ProcUntil`. The blocks are interconnected: the output `o1` of the first block connects to the input `in1` of the second block, and the output `out1` of the second block connects to the input `in1` of the third block.

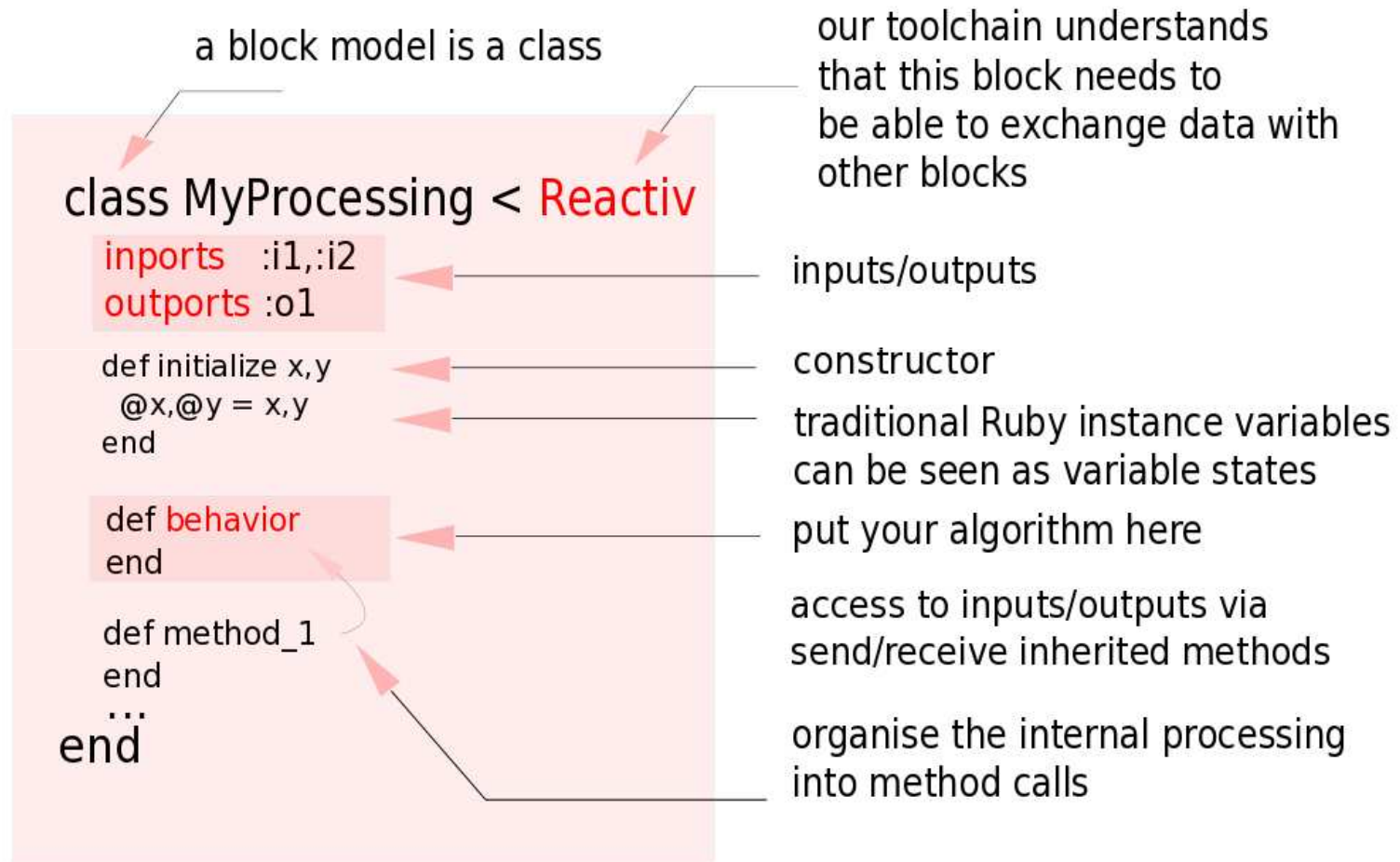
The top toolbar contains several icons, with the `Simulate` icon (a green play button) circled in red. Other icons include `+ Zoom`, `- Zoom`, `Input`, `Output`, `Block`, `Delete`, `Analyze`, `Stop`, and `Image...`.

On the left, the `Project Items` pane shows a tree view with folders `generated`, `hw`, and `sw`, and various process blocks like `ProcFor`, `ProcUntil`, `ProcWhile`, and their refined and serialized versions. The `Ritf2012` folder is selected.

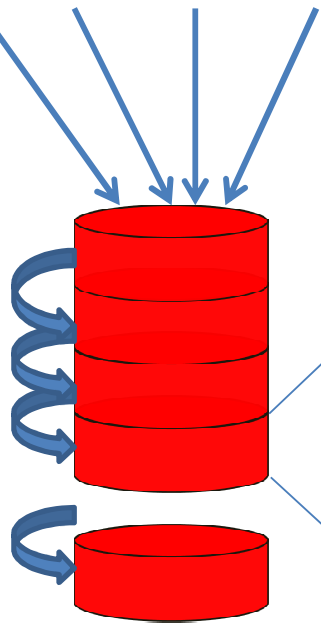
At the bottom, the `Console` window shows the following output:

```
==> generating /home/jccl/experiments/RITF2012/ritf2012/generated/hw/wave_imx51.do
-----
Quatus II project generation
-----
==> generating /home/jccl/experiments/RITF2012/ritf2012/generated/hw/ziam_refined_ritf2012.qpf
```

Behavioral blocks in *Modaë*



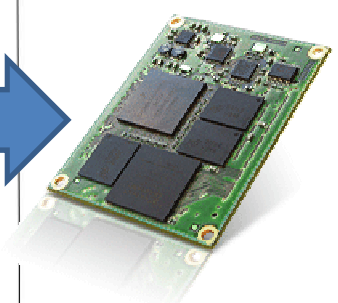
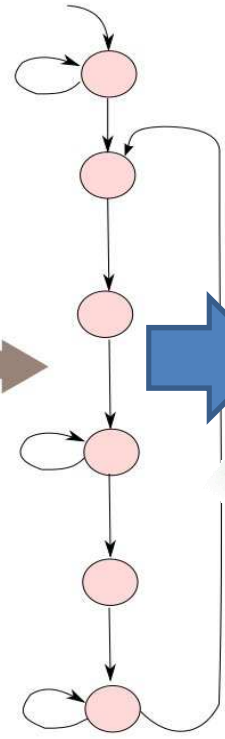
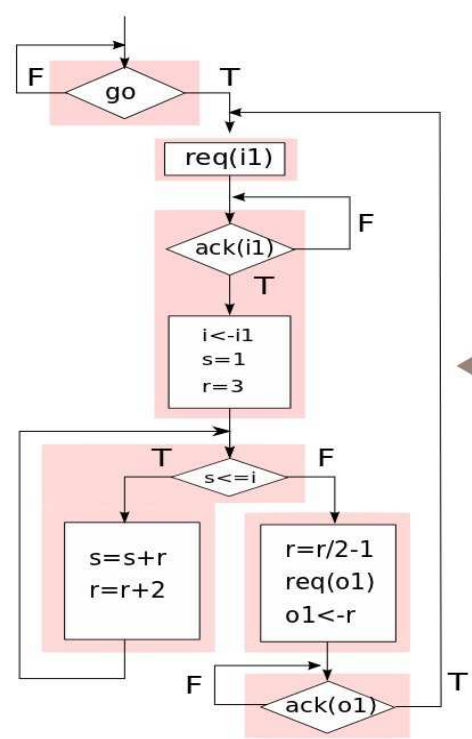
Modaë SLS (2/2)



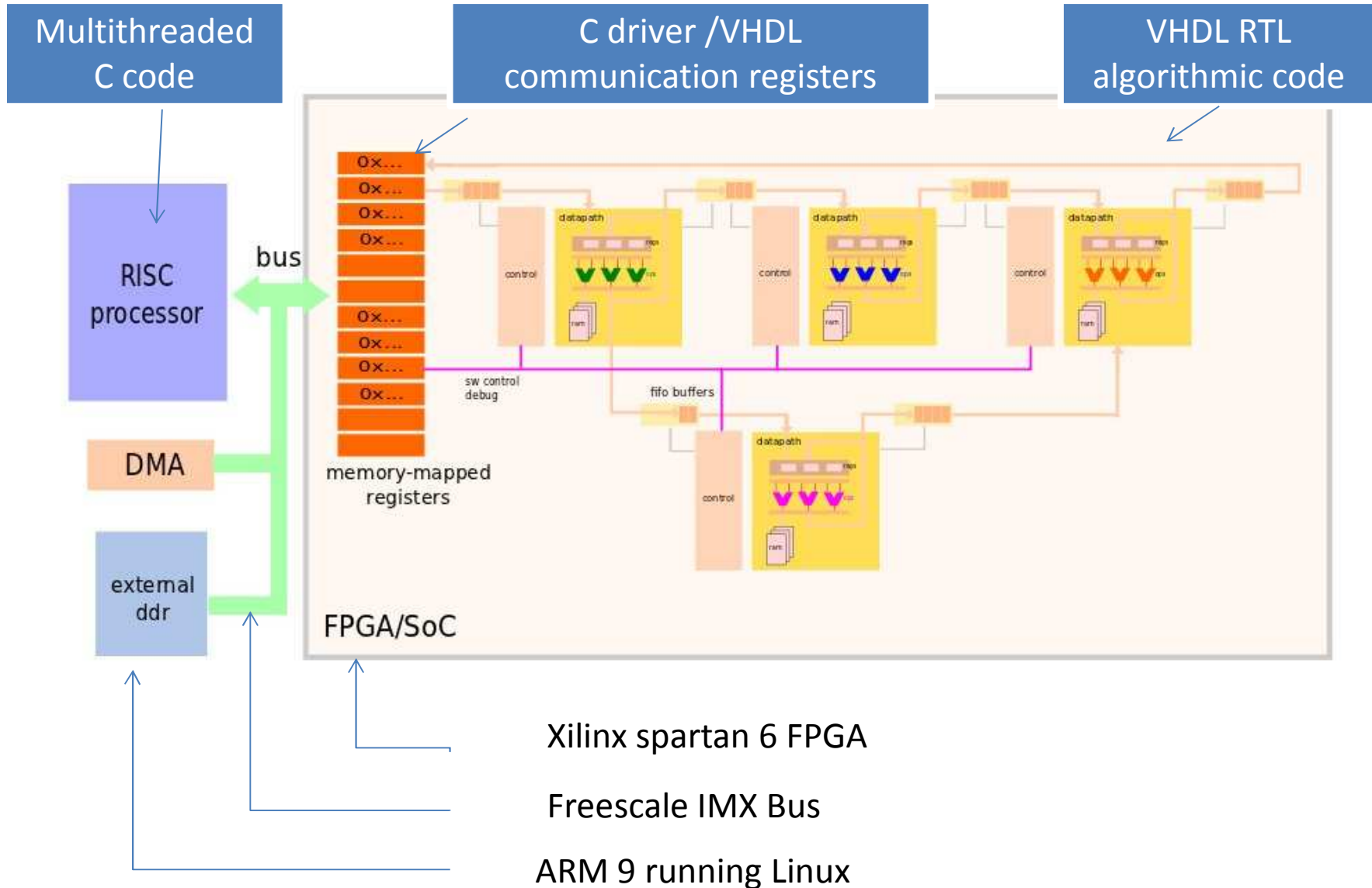
Internal representation

CDFG : Control-data flow graph

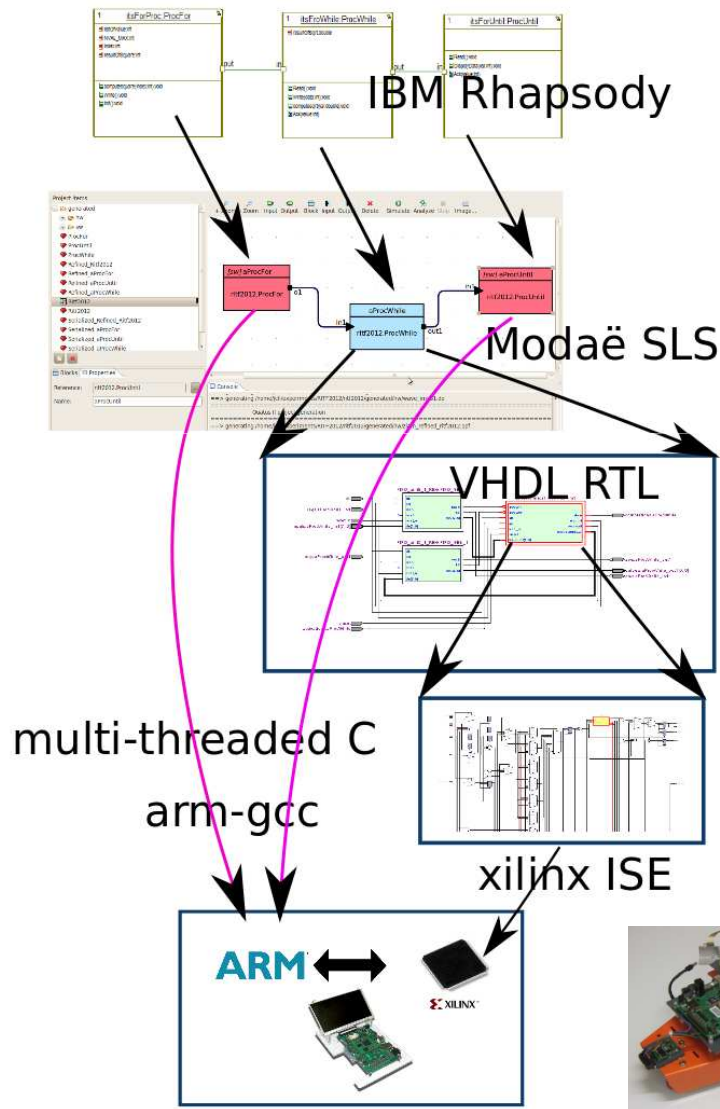
Synthesizable HDL code



Architecture template



Preliminary results



Simple UML 2.0 model – simple action language

HW/SW mapping annotated

Software synthesis + HLS

VHDL RTL synthesis

Porting on platform



Conclusions

- Ceremonial system-level processes vs agile processes and languages
 - Possible interactions
 - Complementary
- MDD : several technologies to develop a system
 - Endogeneous vs Exogeneous battle ?
- Example :
 - Is UML easier / more expressive than Ruby +DSL ?



QUESTIONS ?

Jean-christophe.le_lann@ensta-bretagne.fr