Open-PSA in GRIF

Cyrille Folleau (SATODEV)

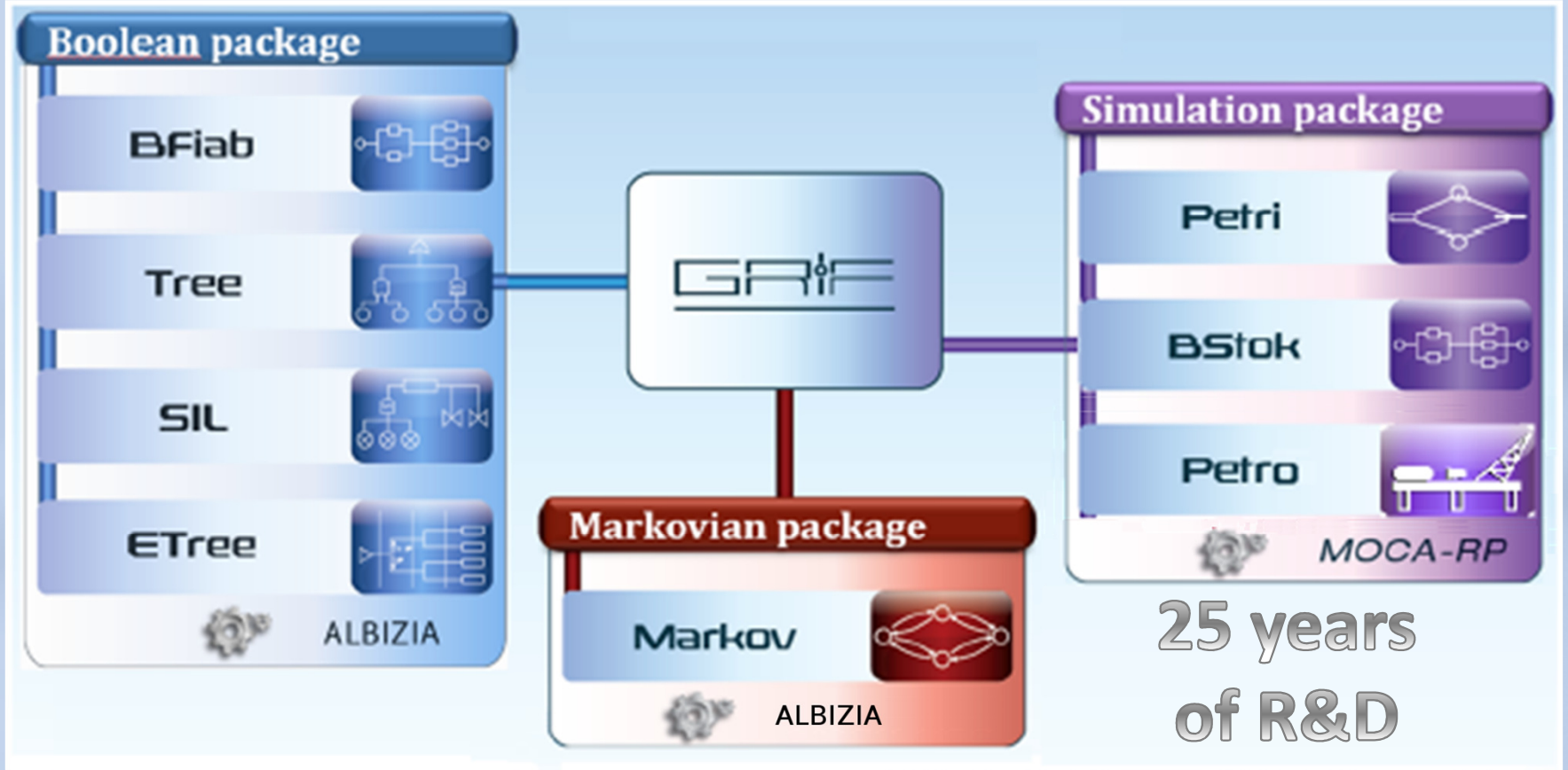
Clamart 10/12/2011 – Open-PSA Workshop 2012

Summary

- GRIF description
- Open-PSA in GRIF 2013 (Fault-Tree only)
- Needs of functions
- New models that could be handled in Open-PSA
  - Markov graphs
  - Reliability Bloc Diagram

GRIF Description

- **GRIF** is a systems analysis software platform for determining the essential indicators of dependability.



Open-PSA in GRIF

OPSA Import made in	OPSA Export made in
2013 ? Needs format improvement	2013 ? Needs format improvement
2011	2012
N/A	N/A
2014 ? Needs GRIF improvement	2013
2013 ? Needs format improvement	2013 ? Needs format improvement

Functions

- Open-PSA v2.0d :
  - Software X provides “advanced” distribution named “MyDistribution” for its basic events. Distribution is saved is open-PSA format for each basic event.
  - Software X opens the file, how to know easily that a distribution is “MyDistribution” ?
  - Software Y opens the file, what can be displayed to the user ? For a basic event with “MyDistribution”
- Open-PSA V3 ? : we need functions
  - let developers create any distribution and identify it easily.
  - Import unknown distribution from other software
  - Decrease file size : if a distribution is used 100 times, the function is written once.

Functions

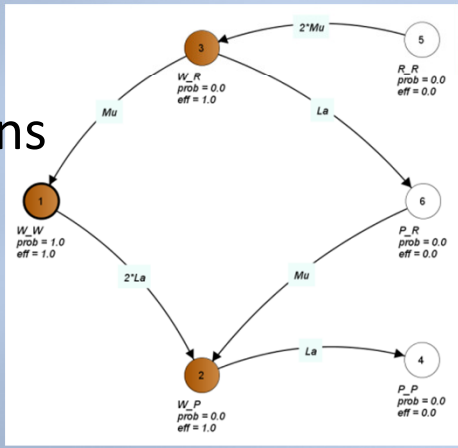
- For Exemple : $f(t) = b*(a-c*t)$

```
<define-functions>
    <define-function name="myFirstFunction">
        <define-function-param name="a"/>
        <define-function-param name="b"/>
        <define-function-param name="c"/>
        <define-function-formulae>
            <mul>
                <function-param name="b"/>
                <sub>
                    <function-param name="a"/>
                    <mul>
                        <function-param name="c"/>
                        <system-mission-time/>
                    </mul>
                </sub>
            </mul>
        </define-function-formulae/>
    </define-function>
</define-functions>
```

```
<define-basic-event name='myEvt'>
    <distribution>
        <function-distribution name="myFirstFunction">
            <function-param name="a">
                <float value='0.1'/>
            </function-param>
            <function-param name="b">
                <float value='0.5'/>
            </function-param>
            <function-param name="c">
                <float value='50'/>
            </function-param>
        </function-distribution>
    </distribution>
</define-basic-event>
```

Markov

- A Markov graphs analysis needs :
  - One or many graphs made of states and transitions
    - For each state, we needs information (attributes ?)
      - The system works or not
      - The probability to be in that state at t=0
      - ….



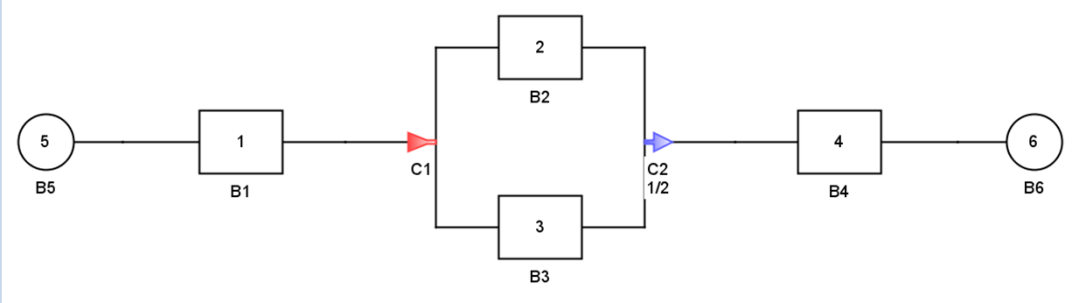
  - A system can have many different behaviors (graphs) during its life
  - Chaining matrixes specifying the way to switch from one model to another
  - A scheduler specifying when and how to use models and matrixes

Markov

```
<define-markov name='Exemple02'>
      <graph number='1' name='operation'>
        <state number='1' name='W' initial='1.0' available='true'/>
        <state number='2' name='R'/>
        <state number='3' name='F'/>
        <transition source='1' target='3'><parameter name='lambda'/></transition>
        <transition source='2' target='1'><parameter name='mu'/></transition>
      </graph>
      <graph number='2' name='maintenance'>
        <state number='1' name='W' available='true'/>
        <state number='2' name='R'/>
        <transition source='2' target='1'><parameter name='mu'/></transition>
      </graph>
      <matrix number='1' name='Matrix1'>
        <transition source='1' target='1'><float value='1.0'/></transition>
        <transition source='2' target='2'>float value='1.0'/></transition>
        <transition source='3' target='2'><float value='1.0'/></transition>
      </matrix>
      <matrix number='2' name='Matrix2'>
        <transition source='1' target='1'><float value=‘0.6'/></transition>
        <transition source=‘1' target='2'>float value=‘0.4'/></transition>
        <transition source=‘2' target='2'><float value='1.0'/></transition>
      </matrix>
      <sheduler>
        <phase number='1' graph='1'><parameter name='T></phase>
        <phase number='2' graph='2'><parameter name='pi'/></phase>
        <first phase='1'/>
        <next phase='2' matrix='1'/>
        <next phase='1' matrix='2'/>
      </sheduler>
</define-markov>
```

Reliability Bloc Diagram

- RBD are made of nodes (blocks and connectors)
- A block needs
  - A Name
  - A distribution
  - A previous-node
  - A next-node



- A block can be defined with a basic event or a gate or a sequence …
- A Connector needs
  - A Name
  - A list of previous nodes
  - A list of next nodes
  - A K out of N configuration

The End !