Credits

Author: Antoine B. Rauzy
Version: 2.0d
Date: December the 4th 2008
Content

- Open-PSA Initiative
- Rationale for the Standard
- Anatomy of the Standard
- Fault Tree Layer
- Stochastic Layer
- Extra-Logical Layer
- Event Tree Layer
- Report Layer
Why Do We Need a Standard?

- Reduce tool dependency
- Have a better confidence in approximations (quality insurance)
- Cross check calculations
- Develop new calculation engines
- Design new model browsers and safety monitors
- Review and document (existing) models
- Clarify (unify?) modeling methodologies
- Call external tools (Level 2 PSA)
- Extend fault trees/events trees formalism
- ...
The Open-PSA Initiative

The Open-PSA Architecture

- Standard representation format
- Model designers
- Model browsers
- Safety monitors
- Calculation engines
- Cutsets browsers
- Industry data
The Open-PSA Initiative

Requirements

- It should be possible to cast any existing model.
- The role of each element should be clearly identified and have an unambiguous semantics.
- The standard should be easy to embed in existing tools and easy to extend.

(existing format(s) translators standard)

... XML format
The Open-PSA Initiative

Anatomy of the Standard
Methodology

- We considered models built with the main tools available on the market
  - Cafta, Saphire, RiskSpectrum, Riskman, Fault Tree free...
  - US, Japanese and European PSA
- We made of taxonomy of all syntactic categories we found in these models
  - Gates, basic events, house events, sequences...
- We gave to each category a formal operational semantics
- We designed a XML representation of categories
The Open-PSA Initiative

Five Layers Architecture

- Report Layer
  - Results of calculation...
- Event Tree Layer
  - Event trees, initiators, sequences, consequences
- Extra-Logical Layer
  - CCF-groups, delete terms, exchange events...
- Fault Tree Layer
  - Fault Trees, gates, basic events, house events
- Stochastic Layer
  - Probability distributions, parameters
Fault Tree Layer
<define-fault-tree name="FT1">
  <define-gate name="top">
    <or>
      <gate name="G" />
      <basic-event name="C" />
    </or>
  </define-gate>
  <define-gate name="G">
    <and>
      <basic-event name="A" />
      <basic-event name="B" />
    </and>
  </define-gate>
</define-fault-tree>
The Open-PSA Initiative

Declarations of Gates

<define-gate name="valve-failed-closed">
  <or>
    <basic-event name="valve-hardware-failure" />
    <gate name="valve-human-failure" />
    <basic-event name="valve-test-failure" />
  </or>
</define-gate>

*the standard provides a complete set of logical connectives*
Declarations of Basic Events

<define-basic-event name="valve-hardware-failure" >
  <exponential>
    <parameter name="failure-rate-valves" />
    <mission-time />
  </exponential>
</define-basic-event>
The Open-PSA Initiative

Stochastic Layer
The Open-PSA Initiative

- Consequence-terms
- Consequences
- Initiator-sets
- Initiators
- Branches
- States
- Functional-events
- Rules
- Named-rules
- CCF-group
- Delete-terms
- Recovery-rules
- Fault-trees
- Formulae
- Gates
- House-events
- Basic-events
- Expressions
- Parameters
- Event-trees
- Stochastic layer
- Fault tree layer
- Extra-logical layer
1. Stochastic expression and parameters
   role and definition
2. Operations
   Arithmetic operations, logical operations, conditional operations
3. Built-ins
   usual time-dependent distributions
4. Random Deviates
   uniform, normal, lognormal deviates, histograms
1. Associate (possibly time-dependent) probabilities with basic events. E.g.

   <define-basic-event name="BE">
     <exponential>
       <parameter name="lambda" />
       <mission-time />
     </exponential>
   </define-basic-event>

2. Define distributions for these probabilities (and more generally for parameters). E.g.

   <define-basic-event name="BE2">
     <uniform-deviate>
       <float value="1.0e-4" />
       <float value="2.0e-4" />
     </uniform-deviate>
   </define-basic-event>
Set of predefined function to describe time-dependent distributions.

*E.g.*

- `<exponential>
  <parameter name="failure-rate-pump" />
  <mission-time />
</exponential>`

- `<Weibull>
  <parameter name="shape1" />
  <parameter name="scale1" />
  <sub>
    <mission-time />
    <parameter name="locality1" />
  </sub>
</Weibull>`

- `...`
To perform sensitivity analyses. E.g.

- `<uniform>
  <float value="1.0e-3" /> lower-bound
  <float value="2.0e-3" /> upper-bound
</uniform>

- `<lognormal>
  <float value="1.23e-4" /> mean
  <int value="3" /> error-factor
  <float value="0.90" /> confidence
</lognormal>

- ...
Histograms

<histogram lower-bound="100">
  <bin upper-bound="120">
    <float value="1.0e-3 />
  </bin>
  <bin upper-bound="150">
    <float value="2.0e-3 />
  </bin>
  <bin upper-bound="160">
    <float value="3.0e-3 />
  </bin>
</histogram>
The Open-PSA Initiative

Extra-Logical Layer
Extra-Logical Layer (Content)

1. Common Cause Failures
   • models, declarations
2. Exclusive events (delete terms)
   • model, declaration
3. Recovery rules
   • model, declaration
Delete Terms

Delete terms are groups of exclusive (basic) events.
- Used to model physically impossible configurations such as simultaneous maintenance

Three possible interpretations/uses of the exclusive group $g=\{e_1, e_2\}$

1. Post-processing of cutsets
   - (e1 and e2 and ...) deleted

2. Global constraint
   - NewTopEvent = TopEvent and [not (e1 and e2)]

3. Local substitution
   - $e_1 \rightarrow ge_1 = (e_1 \text{ and not e2})$
   - $e_2 \rightarrow ge_2 = (e_2 \text{ and not e1})$
XML representation
<define-exclusive-group name="g1" >
  <basic-event name="e1" />
  <basic-event name="e2" />
  <basic-event name="e3" />
</define-exclusive-group>
Event Tree Layer

The Open-PSA Initiative
Graphical presentation of Event Trees

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td></td>
<td>S1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S6</td>
</tr>
</tbody>
</table>

Interpretation

S1 = I and not F and not H
S2 = I and not F and H
S3 = I and F and not G and not H
S4 = I and F and not G and H
S5 = I and F and G and not F
S6 = I and F and G and H

A priori simple but …
• Fault trees may be given flavors (by setting house events)
• These flavors may depend on the current branch
• There may have several initiating events
• Some success branches may be interpreted as a bypass
• There may have multi-states branches
• Branches may be defined as references to other branches
• …
Preliminaries (2)

- Fault trees may be given flavors (by setting house events)
- These flavors may depend on the current branch
- There may have several initiating events
- Some success branches may be interpreted as a bypass
- There may have multi-states branches
- Branches may be defined as references to other branches
- …

Event Trees should be seen as a graphical programming language!
- The graphical view described the structure of the tree, i.e. the different sequences
- Instructions are provided to give flavors to fault trees
- The interpretation of sequences (Boolean formula) is built while walking along the branches
Walk:
- 0, 1, 2, 3, 4 (S1)
- 0, 1, 2, 3, 5 (S2)
- 0, 1, 6, 7, 3, 4 (S1)
- ...

at each point some instructions can be executed in order to set values of house events and parameters and/or to collect functional event
Structure of Event Trees (2)

<define-event-tree name="ET1">
  <define-functional-event name="F">
    <fault-tree name="FTF" gate="top" />
  </define-functional-event>
  ...
  <define-consequence name="S1" />
  ...
  <path>
    <fork functional-event="F">
      <path>
        <collect functional-event="F" polarity="success" />
        <fork functional-event="H">
          ...
        </fork>
      </path>
    </path>
    ...
    <fork>
    </path>
  </path>
</define-event-tree>

declarations of functional events

declarations of consequences

definition of the structure

instruction
Instructions to set parameters/house event values
   - <set house-event="H1">
     <constant value="false"/>
   </set-parameter>
   - <set parameter="lambda">
     <float value="0.001"/>
   </set-parameter>

Instructions to collect functional events
   - <collect functional-event="F" polarity="failure"/>

Conditional instructions
   - <if>
     <collected functional-event="F"/>
     <set house-event="H2">
       <constant value="true"/>
     </set>
   </if>
Blocks
  - <block>
    instruction+
  </block>

Rules (named blocks of instructions)
  - <define-rule name="R1" >
    <set house-event="H1"> <constant value="false" /> </set>
    <set house-event="H2"> <constant value="true" /> </set>
    <set house-event="H3"> <constant value="true" /> </set>
    ...
  </define-rule>
The Open-PSA Initiative

Report Layer
1. Description of Calculations
   – model, tool, algorithm, mission-time, cutoff...

2. Description of Results
   – minimal cutsets
   – probabilistic measures
Description of Calculations

- Software
  - version, contact organization (editor, vendor)

- Calculation algorithm
  - name
  - limits (number of basic events, cutsets...)
  - preprocessing techniques
  - cutoffs
  - handling of success branches, use of delete terms
  - external routines
  - calculation time
  - ...

- Feedback
  - success, failure

The standard provides examples rather than a strict syntax for these items
<sum-of-products name="MCS1" basic-events="3" products="2">
  <product order="2">
    <basic-event name="A" />
    <basic-event name="B" />
  </product>
  <product order="2">
    <not>
      <basic-event name="A" />
    </not>
    <basic-event name="C" />
  </product>
</sum-of-products>
Descriptions of Results

<measure name="RAW" system="TopEvent" component="BE33" >
   <mean value="0.00149807" />
   <standard-deviation value="0.000385405" />
   <error-factor percentage="90" value="1.00056" />
   <histogram lower-bound="0" >
      <bin upper-bound="0.25"> <float value="0.00112081"> </bin>
      <bin upper-bound="0.50"> <float value="0.00136203"> </bin>
      <bin upper-bound="0.75"> <float value="0.0016188"> </bin>
      <bin upper-bound="1.00"> <float value="0.00186128"> </bin>
   </histogram>
</measure>