The Open-PSA Initiative

Standard Representation Format for Probabilistic Safety Analyses

Towards a New Generation of Models and Tools

The Open-PSA Initiative

Credits

Author: Antoine B. Rauzy

Version: 1.1a

Date: September the 5th 2007

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

The Open-PSA Initiative

Who Are We?

- Informal group
- Our goals
  - To develop a Standard Representation Format for PSA
  - To exchange about the new generation of PSA models and tools
  - To create working groups on subjects of interest
  - To organize workshops
  - ...
- Website
  - www.open-psa.org

Events

- Past events
  - Workshop Goesgen (Switzerland), 06/12/2007, KKG
  - Working Group, Paris (France), 07/19/2007, EdF
- Forthcoming events
  - Presentation ACRS, Washington DC (USA), 10/02/2007
  - Workshop Washington DC (USA), 10/03/2007, EPRI
  - Workshop Osaka (Japan), 11/13-14/2007, NEL
  - Workshop Vienna (Austria), ~12/10/2007, IAEA

The Open-PSA Initiative

Rationale for the Standard

Where Are We?

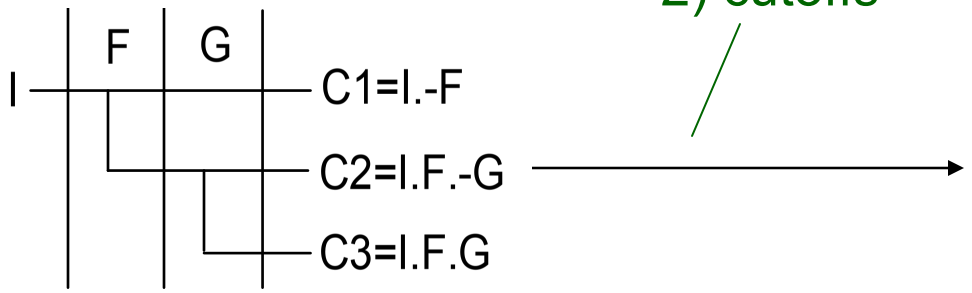
- Detailed models have been developed for level 1 and level 2 PSA
- Good tools have been developed to design and assess models

... but

- Models are hard to master, to check for completeness, to maintain...
- Models are tool-dependent
- Calculation engines have flaws

Success Branches

- MCS calculations

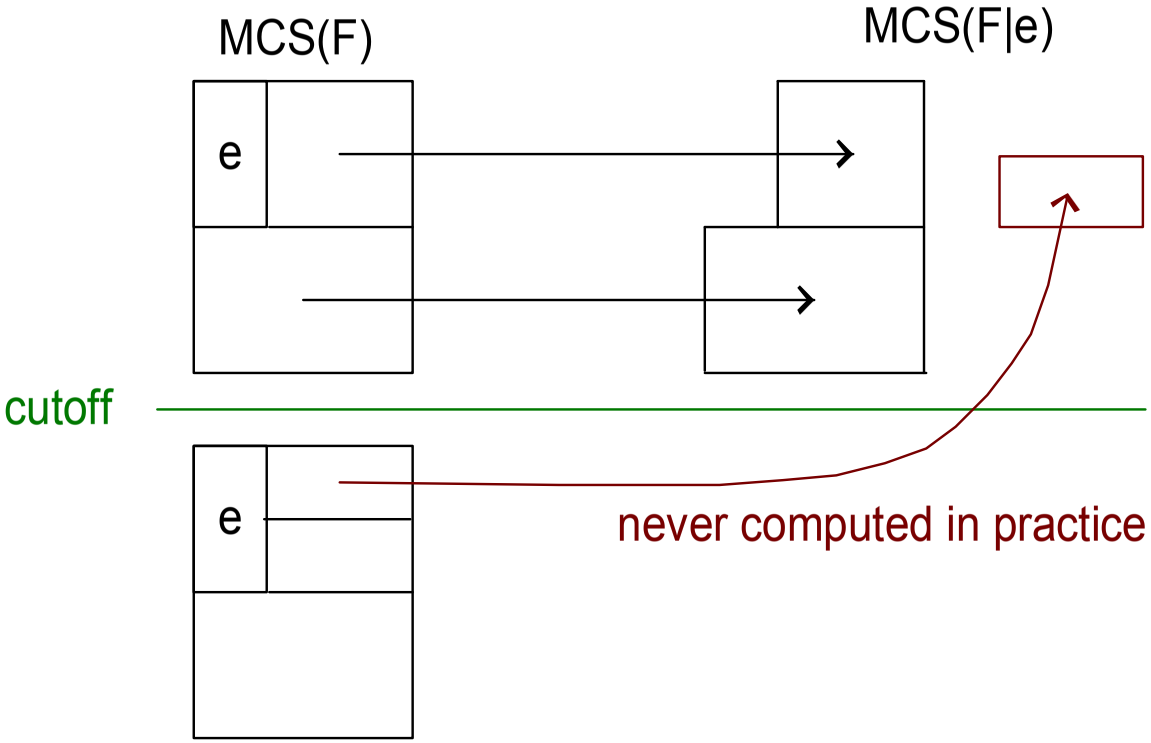


double approximation:
1) non-coherent vs coherent
2) cutoffs

- [Epstein, Rauzy 04], japanese PSA
  - CDF overestimated ($1.21\ 10^{-4}$ vs $2.27\ 10^{-5}$)
  - 1/3 sequences underestimated
  - 2/3 sequences overestimated (up to a factor 100!)

Importance Factors

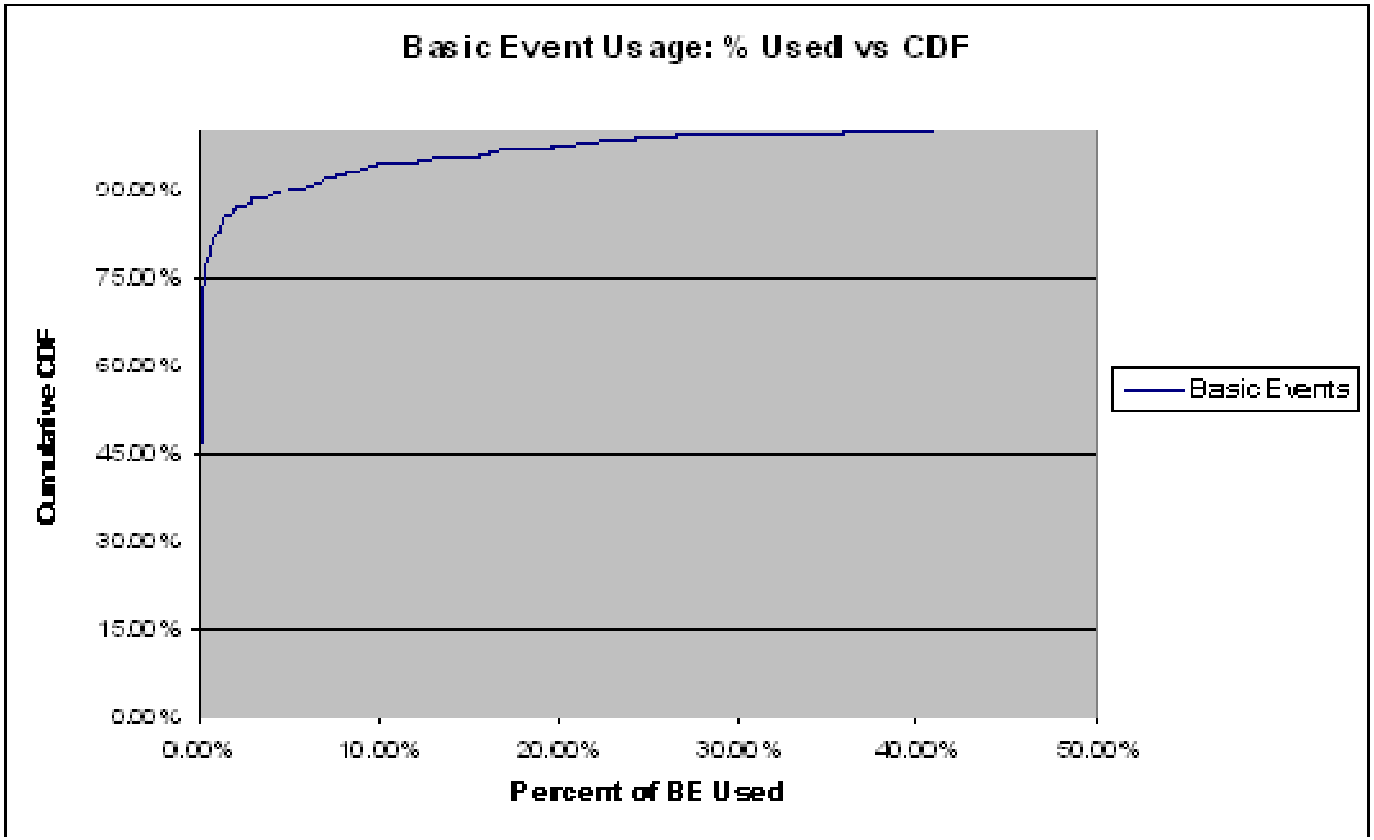
- Impact of cutoffs : all importance factors require to compute conditional probabilities



- Chaotic ranking of events, [Duflot 2006], French PSA

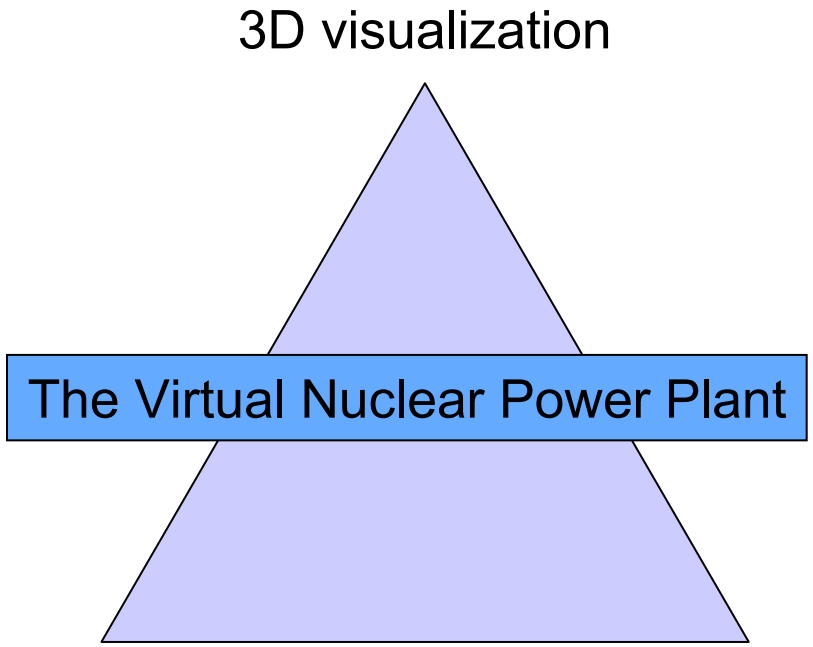
Complexity of Models

- [Epstein, Rauzy 05], US PSA
  - Up to 52 and/or alternations from top event to basic events !
  - 5% of basic events are actually used for (most of the) calculations



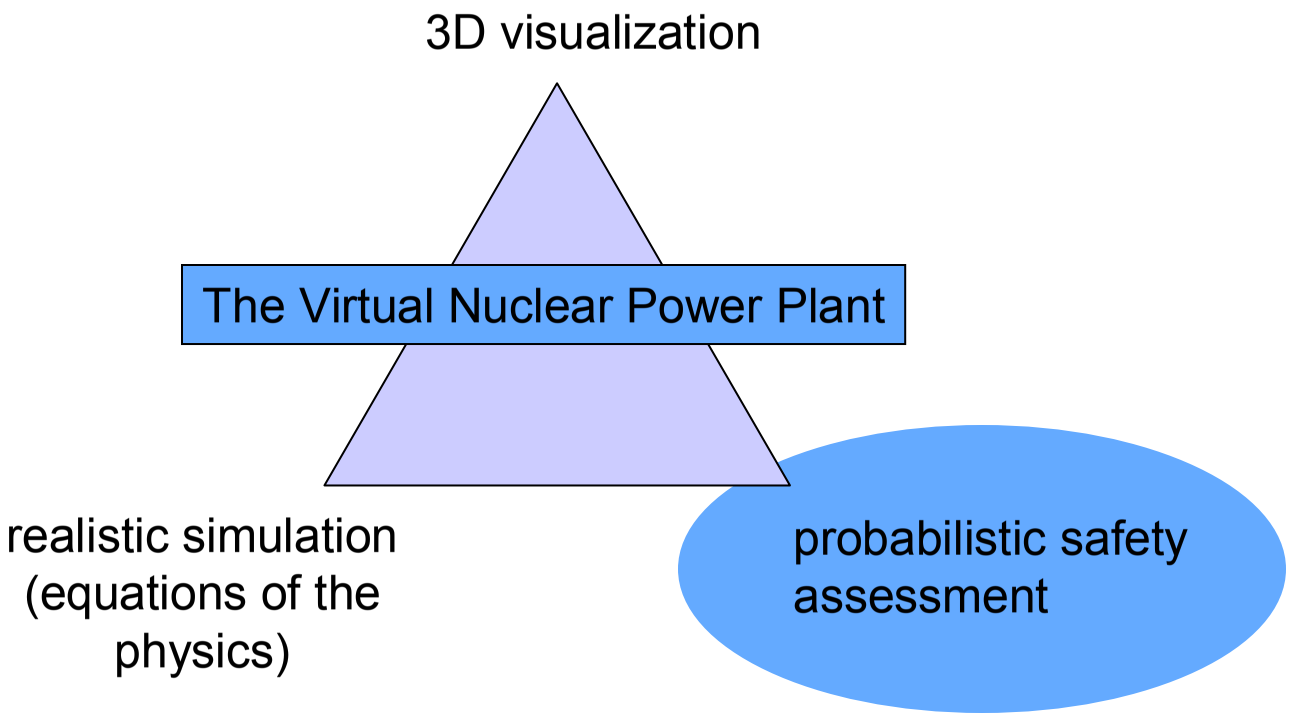
Where We Want to Go?

The future ...



The Open-PSA Initiative

A First Step...

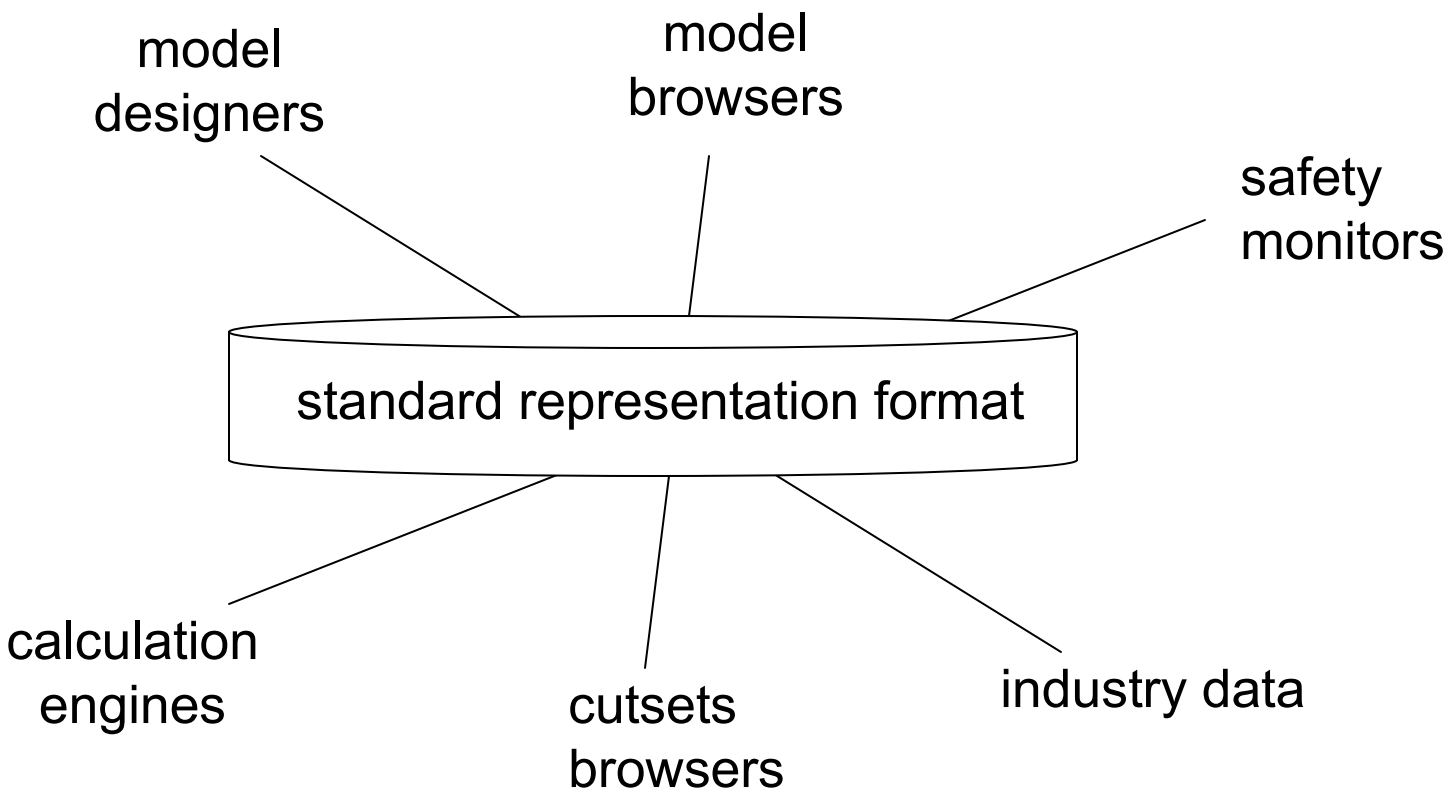


An International Standard Representation Format for PSA Models

Why Do We Need a Standard?

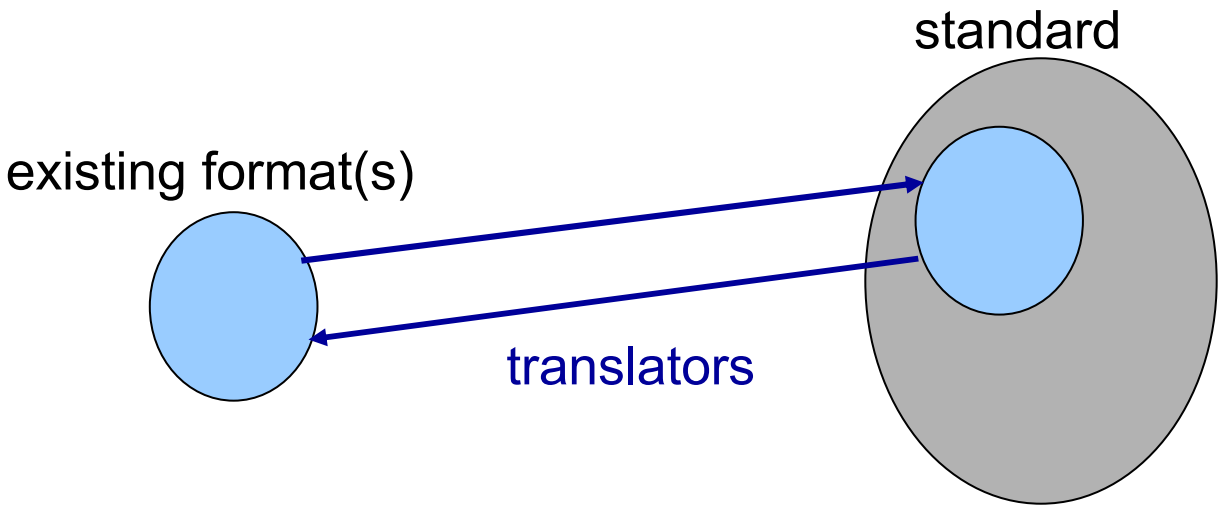
- Reduce tool dependency
- Have a better confidence in approximations (quality insurance)
- Cross check calculations
- Develop new calculation engines
- Design new model browsers and safety monitors
- Review and document (existing) models
- Clarify (unify?) modeling methodologies
- Call external tools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

The Open-PSA Architecture



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

… XML format

The Open-PSA Initiative

Anatomy of the Standard

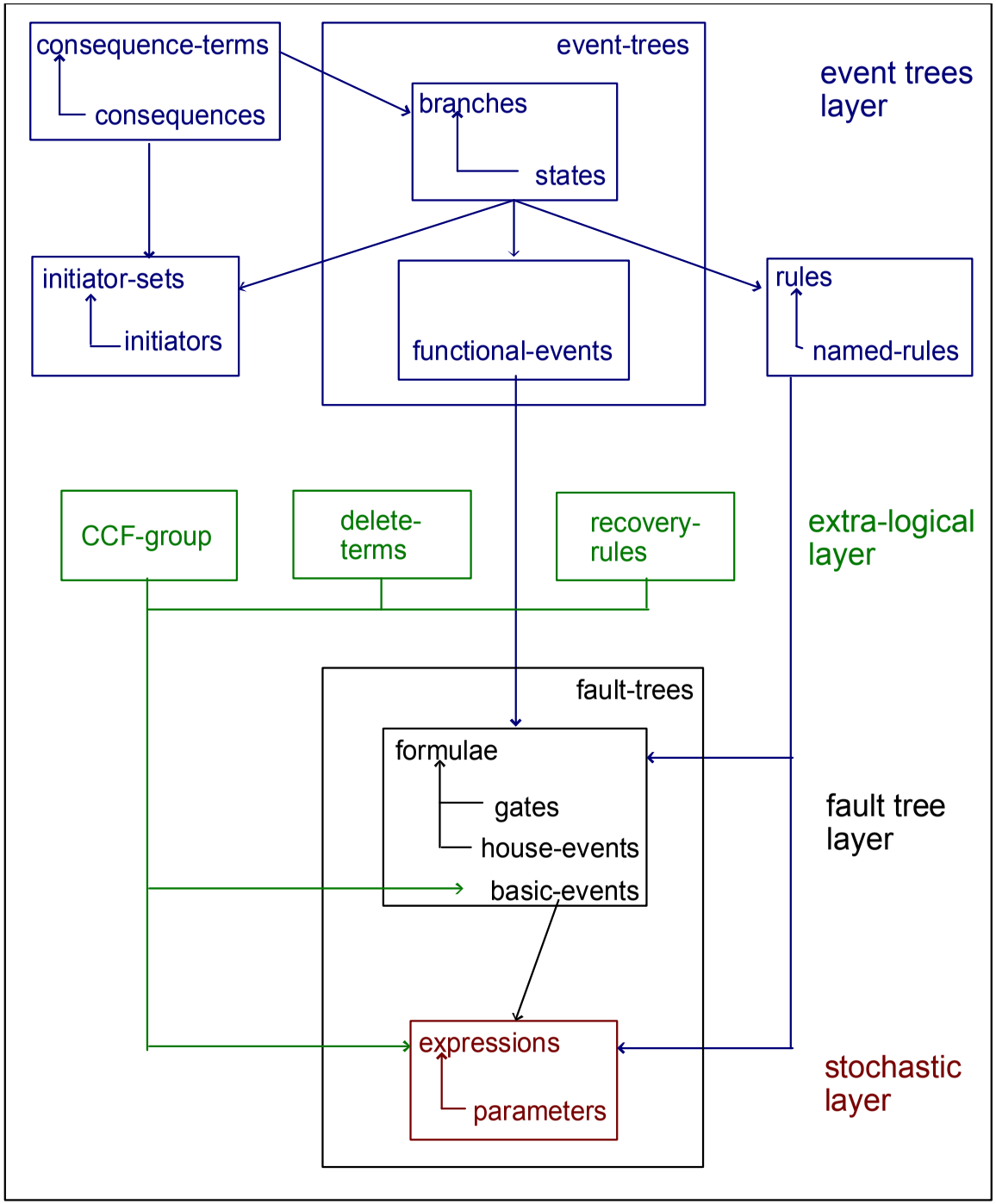
Methodology

- 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

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

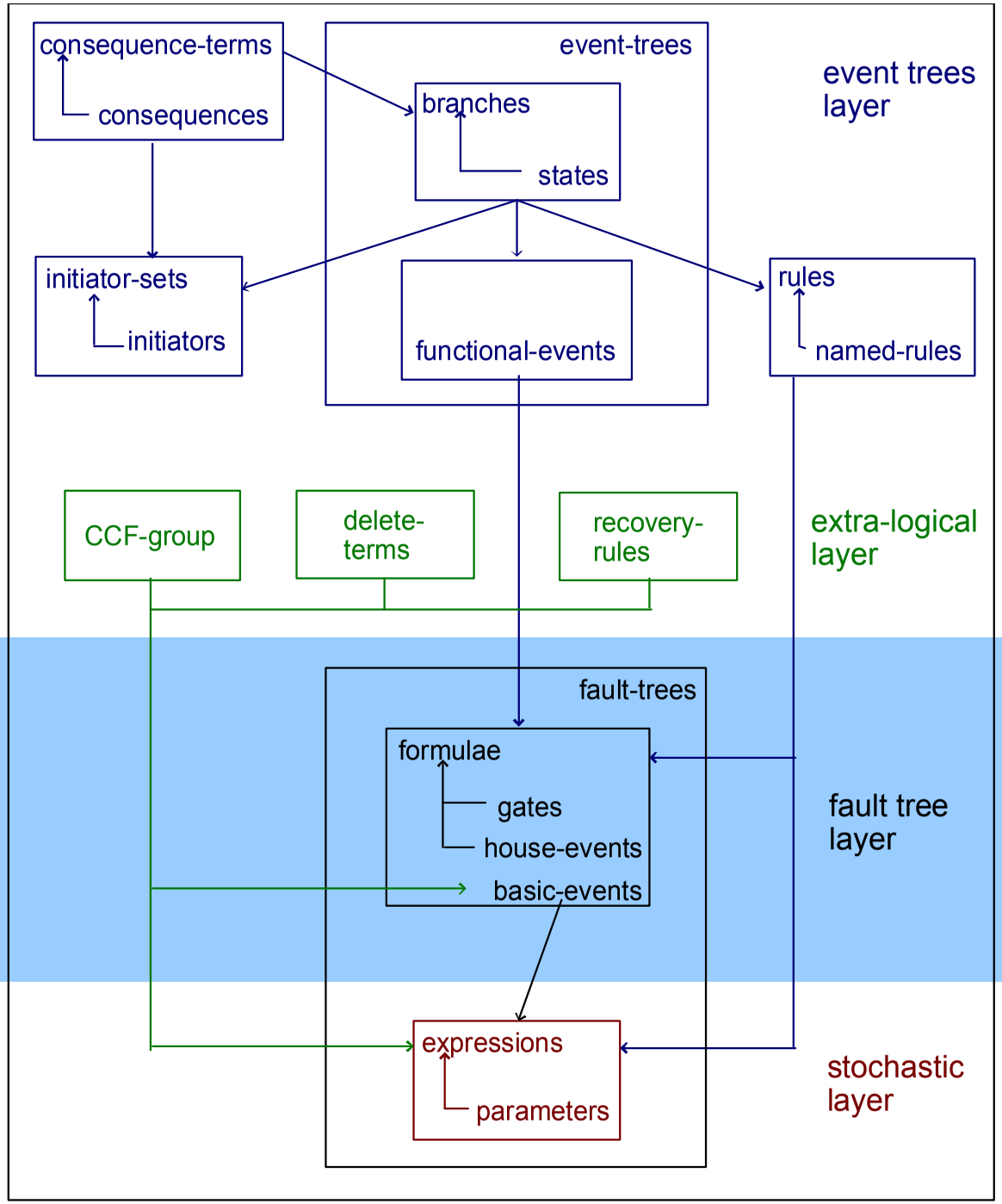
The Open-PSA Initiative



The Open-PSA Initiative

Fault Tree Layer

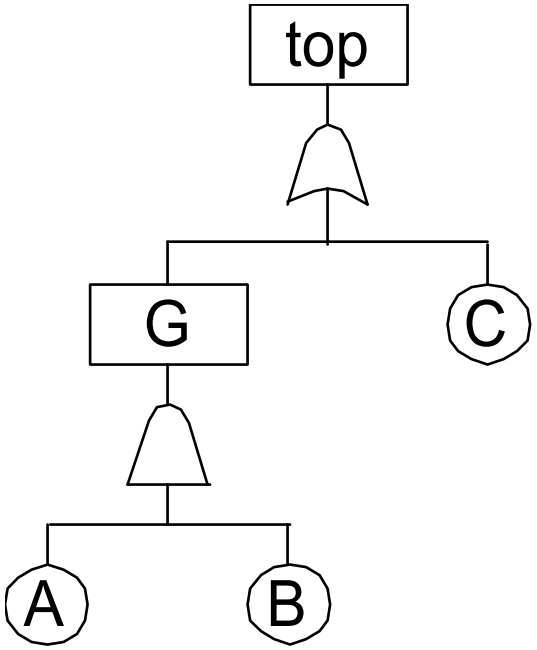
The Open-PSA Initiative



Fault Tree Layer (Content)

1. Declarations
   - Fault trees, gates, house events, basic events
2. Logical operations
   - Boolean formulae
3. Documentating and structuring models
   - Labels, attributes, components
4. Resolution of name conflicts
   - Local versus global elements

Declarations of Fault Trees




```
<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>
```

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>
```

Declarations of House Events

```
<define-house-event name="HE1" >
    <constant value="false" />
</define-house-event>
```

default value (optional)

Formulae (1)

formula ::=

<constant value="true" />

| <constant value="false" />

| <gate name="*identifier*" />

| <house-event name="*identifier*" />

| <basic-event name="*identifier*" />

...

Formulae (2)

formula ::=

...

| <not> *formula* </not>

| <or> *formula*+ </or>

| <and> *formula*+ </and>

| <iff> *formula*+ </iff>

| <xor> *formula*+ </xor>

| <nor> *formula*+ </nor>

| <nand> *formula*+ </nand>

...

F iff G = F.G + -F.-G

F xor G = F.-G + -F.G

F nor G = -(F+G)

F nand G = -(F.G)

Formulae (3)

formula ::=

...

| <atleast min="*integer*">

formula+

</atleast>

| <cardinality min="*integer*" max="*integer*" >

formula+

</cardinality >

k-out-of-n

at least min and at most max

Labels

```
<define-gate name="k2-relay-failure" >
    <label>
        k2 relay fails to open when k5 relay
        closed for t > 60s
    </label>
    <or>
        <basic-event name="k2-relay-fail-to-open" />
        <gate name="EMF-not-removed-from-k2" />
    </or>
</define-gate>
```

any text

Attributes

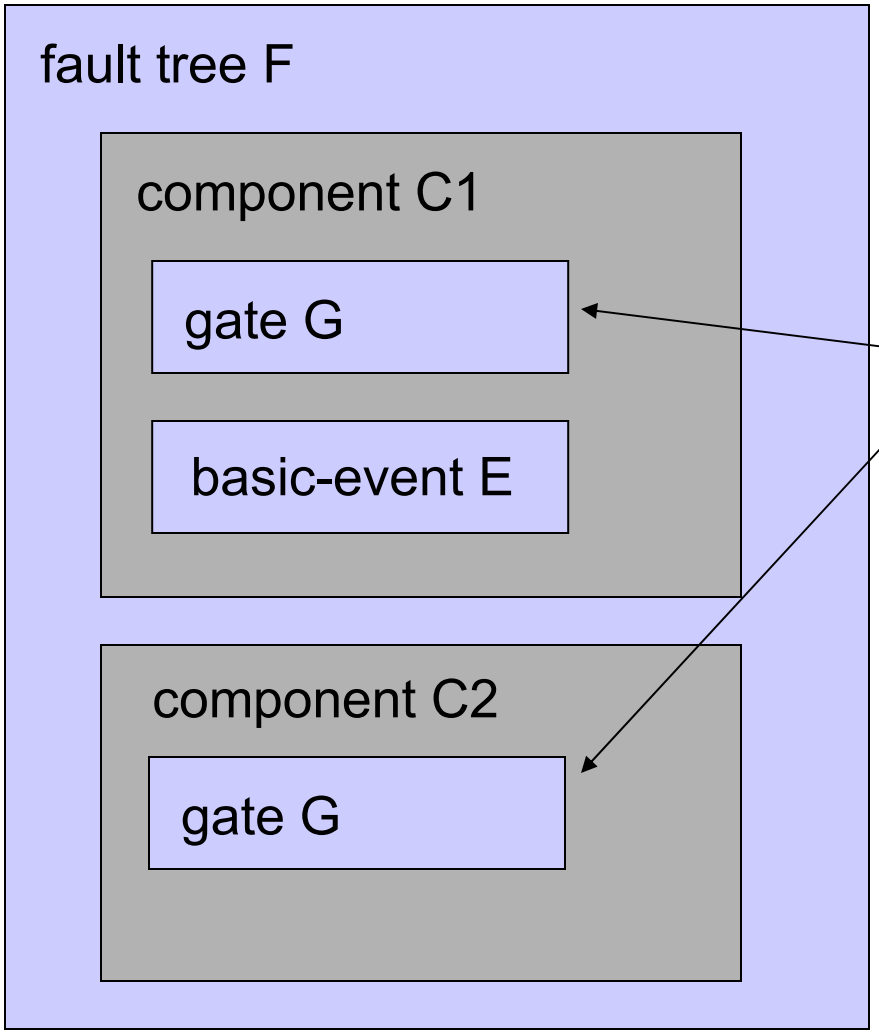
```
<define-gate name="pump-1" >
    <label> ... </label>
    <attributes>
        <attribute name="room" value="33A" />
        <attribute name="year" value="2005" />
        ...
    </attributes>
    <and> ... </and>
</define-gate>
```

Declarations of Components

Components make it possible to group a set of declarations in order to structure models

<define-component name="*identifier*" >
(*gate-declaration*
| *basic-event-declaration*
| *house-event-declaration*
| *parameter-declaration*
| *component-declaration*
)*
</define-component>

Resolution of Name Conflicts



Global elements: conflict !

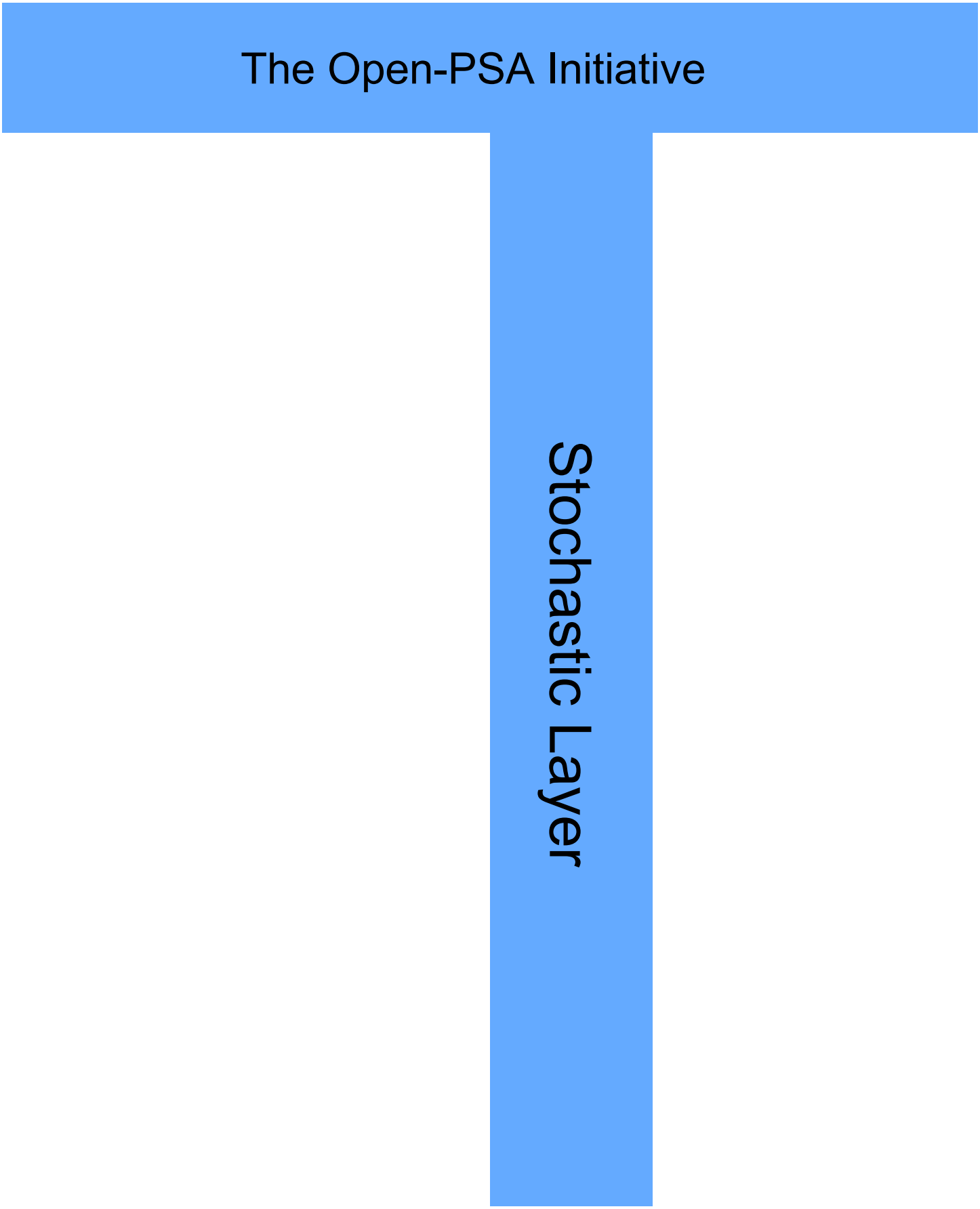
Local elements: OK
- F.C1.G
- F.C2.G

Resolution of Name Conflicts (continued)

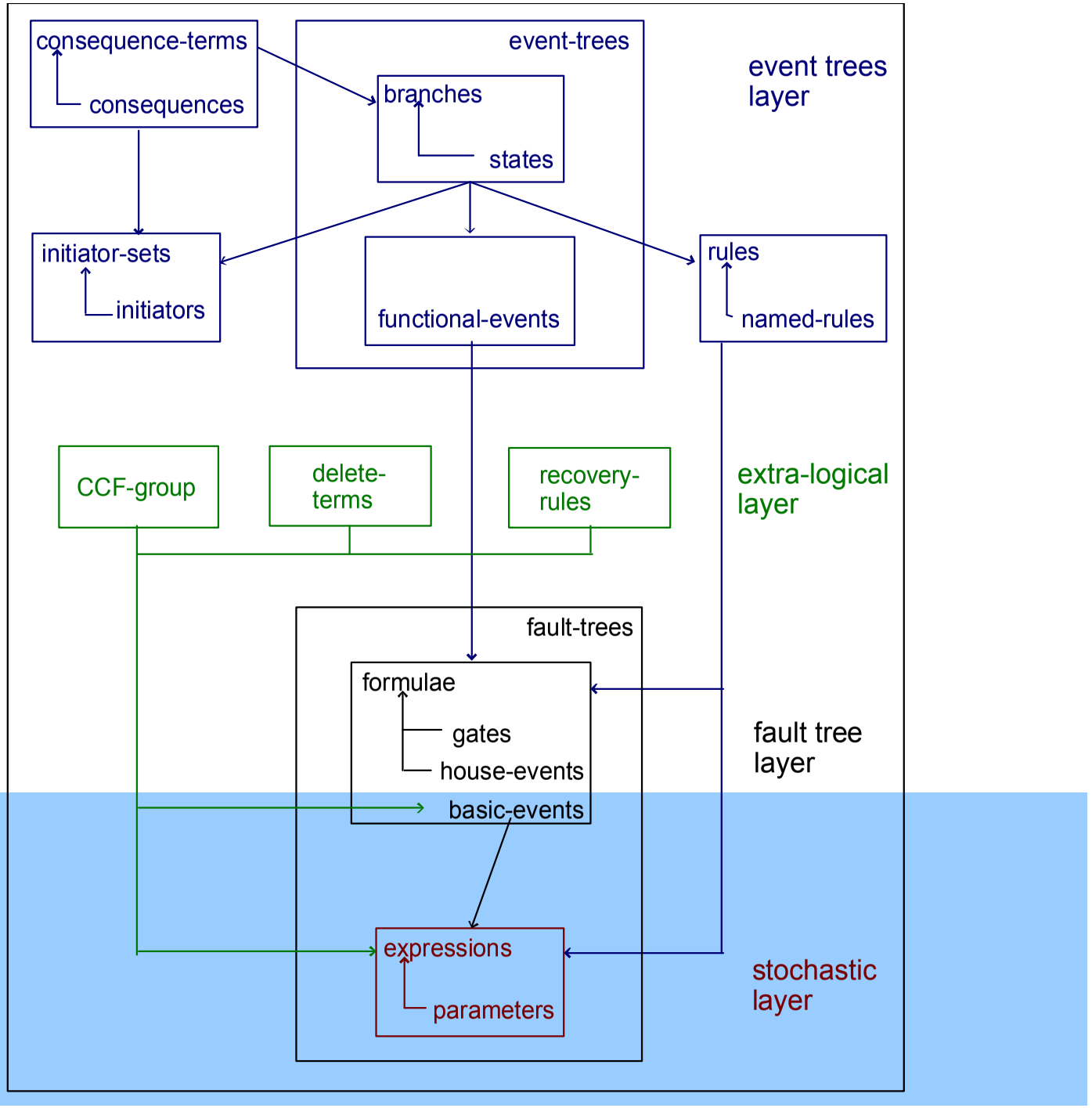
- By default, all of the elements of a model are visible everywhere in the model
  - This implies that they are uniquely identified
- Gates and components can be declared as 'local' (as opposed as 'global'). E.g.

```
<define-gate name="g1" status="local" >
...
</define-gate>
```

- Local components are accessed through the '.' notation
  - F.C1.g1
- By default, elements are local if their innermost container is, and global otherwise.


The Open-PSA Initiative
Stochastic Layer

The Open-PSA Initiative



Stochastic Layer (Content)

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

Role of Stochastic Expressions

1. Associate (possibly time-dependent) probabilities with basic events. E.g.

```
<define-basic-event name="BE">
  <exponential>                          negative exponential distribution
    <parameter name="lambda" />          failure rate
    <mission-time />                     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>                      uniform random deviate
    <float value="1.0e-4" />             lower bound
    <float value="2.0e-4" />             upper bound
  </uniform-deviate>
</define-basic-event>
```

Declarations of Parameters

```
<define-parameter name="failure-rate-valves" >
    <logormal-deviate>
        <float value="1.0e-3" />
        <float value="3" />
        <float value="0.95" />
    </lognormal-deviate>
</define-parameter>
```

mean

error factor

confidence level

Arithmetic Operations

- <add> *expression*+ </add>
- _{*expression*+}
- <mul> *expression*+ </mul>
- *expression*+
- <neg> *expression* </neg>
- ...
- <pow> *expression expression* </pow>
- <exp> *expression* </exp>
- <log> *expression* </log>
- ...
- <min> *expression*+ </min>
- <max> *expression*+ </max>
- <mean> *expression*+ </mean>
- ...

Logical Operations

- <and> *expression*+ </and>
- <or> *expression*+ </or>
- <not> *expression* </not>

- <eq> *expression expression* </eq>
- <df> *expression expression* </df>
- <lt> *expression expression* <lt>
- <leq> *expression expression* </leq>
- <gt> *expression expression* <gt>
- <geq> *expression expression* </geq>

Conditional Operations

- *If-Then-Else*

  <ite> *expression expression expression* </ite>

- *Switch-Case*

  <switch>

  <case> *expression expression* </case>

  ...

  <case> *expression expression* </case>

  <else> *expression*

  </switch>

Built-ins

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>
- ...

Random-Deviates

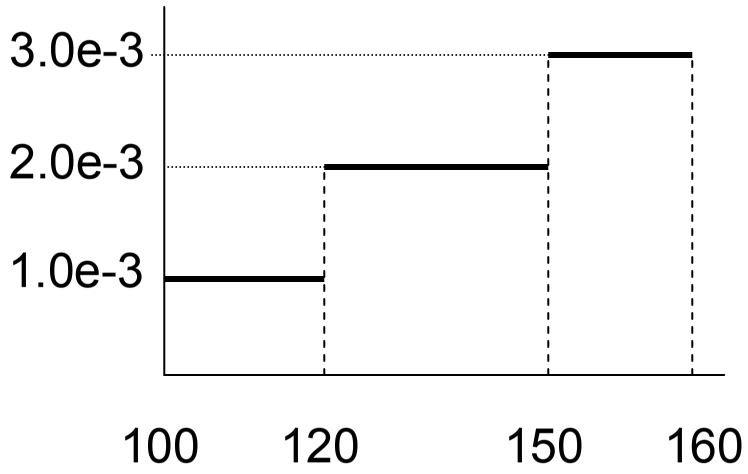
To perform sensitivity analyses. E.g.

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

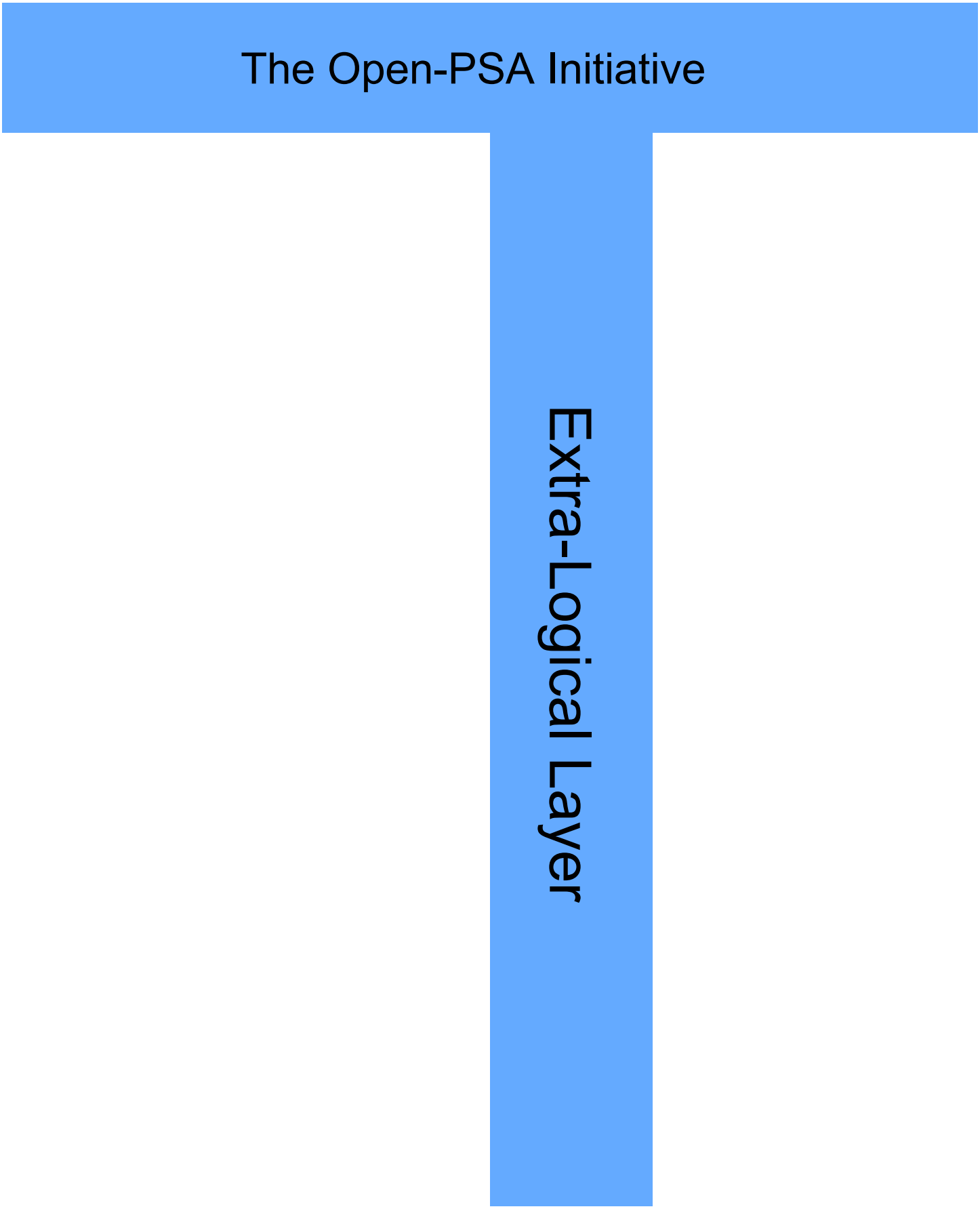
- <lognormal>
  <float value="1.23e-4" /> *mean*
  <int value="3" /> *error-factor*
  <float value="0.90" /> *confidence*
  </lognormal>

- ...

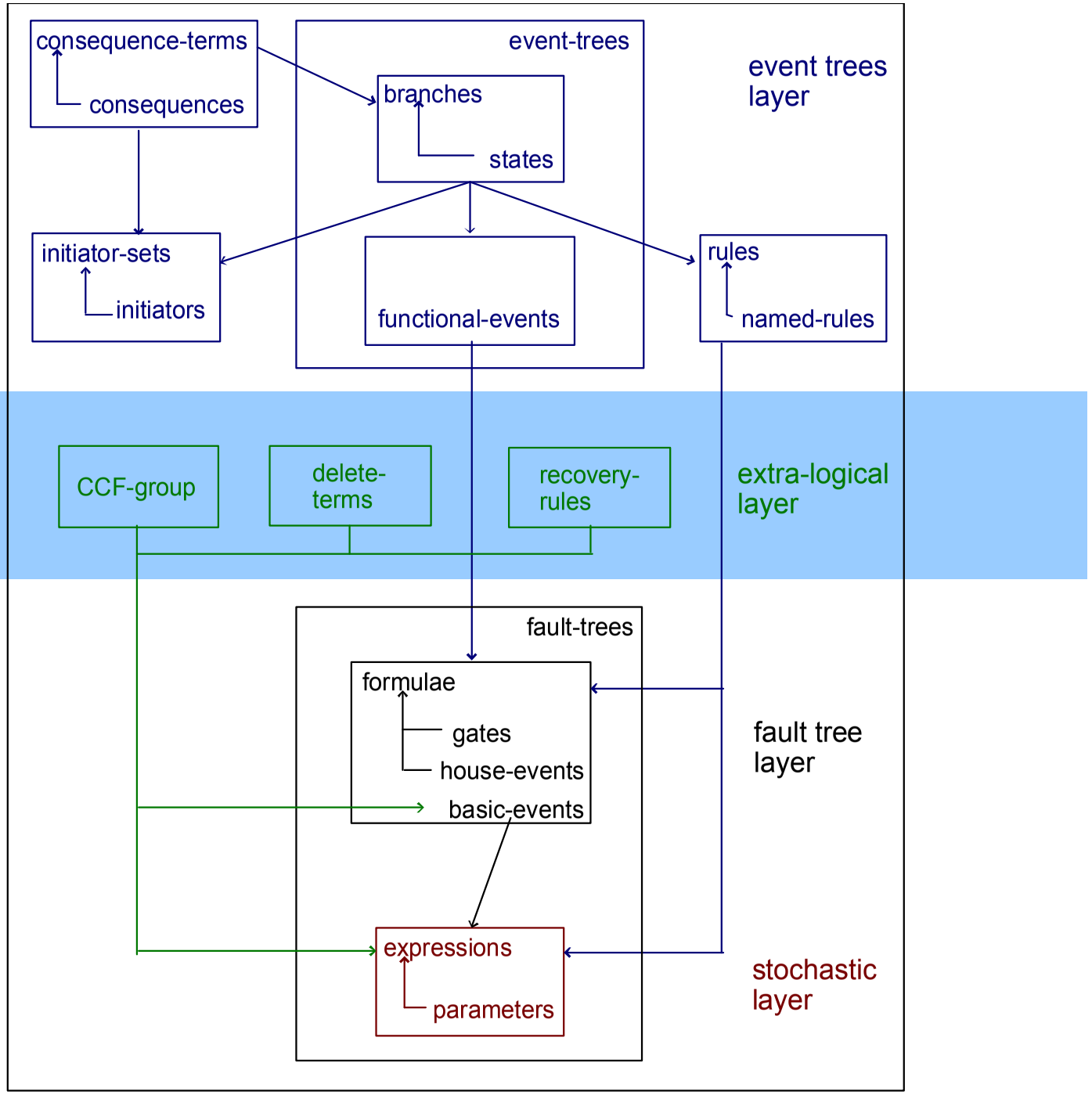
Histograms




```
<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

The Open-PSA Initiative



Extra-Logical Layer (Content)

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

Common Cause Failures

Models

- β-factor
- Multiple Greek Letters
- α-factor
- ϕ-factor *new model suggested by the working group*

β-factor

- Description:
  - members: E1,...,En
  - factor β
  - probability distribution Q
- Each Ei is rewritten as
  - Ei-alone or E1-E2-...-En
- Distributions
  - pr(Ei-alone) = (1 - β).Q
  - pr(E1-E2-...-En) = β.Q

β-factor (continued)

```
<define-CCF-group model="beta-factor" >
    <members>
        <basic-event name="BE1" />
        <basic-event name="BE2" />
        <basic-event name="BE3" />
    </members>
    <factor>
        <float value="0.3" />
    </factor>
    <distribution>
        <exponential>
            <parameter name="lambda" />
            <mission-time />
        </exponential>
    </distribution>
</define-CCF-group>
```

Multiple Greek Letters

- Generalizes β-factor by considering groups of 2, 3, ...,n components
- Description:
  - members E1,...,En
  - Distribution Q
  - Factors ρ2,...,ρn
- Distribution of a group with k elements

$$Q_k = \frac{1}{\binom{n-1}{k-1}} \times \left(\prod_{i=2}^{k} \rho_i \right) \times \left(1 - \rho_{k+1}\right) \times Q$$

Multiple Greek Letters (continued)

```
<define-CCF-group model="MGL" >
    <members> basic-event+ </members>
    <factors>
        <factor level="2" >
            expression          e.g. <float value="0.3" />
        </factor>
        <factor level="3" >
            expression
        </factor>
        ...
        <factor level="n" >          n = size of the group
            expression
        </factor>
    </factors>
    <distribution> expression </distribution>
</define-CCF-group>
```

α-factor

- Another way to set factors
- Description:
  - members E1,...,En
  - Distribution Q
  - Factors α1,...,αn
- Distribution of a group with k elements

$$Q_k = \frac{1}{\binom{n-1}{k-1}} \times \frac{\alpha_k}{\sum_{i=1}^{n} \alpha_i} \times Q$$

α-factor (continued)

```
<define-CCF-group model="alpha-factor" >
    <members> basic-event+ </members>
    <factors>
        <factor level="1" >
            expression          e.g. <float value="0.3" />
        </factor>
        <factor level="2" >
            expression
        </factor>
        ...
        <factor level="n" >          n = size of the group
            expression
        </factor>
    </factors>
    <distribution> expression </distribution>
</define-CCF-group>
```

ϕ-factor

- A third and direct way to set factors
- Description:
  - members E1,...,En
  - Distribution Q
  - Factors ϕ1,...,ϕn
- Distribution of a group with k elements

$$Q_k = \phi_k \times Q$$

ϕ-factor (continued)

```
<define-CCF-group model="phi-factor" >
    <members> basic-event+ </members>
    <factors>
        <factor level="1" >
            expression          e.g. <float value="0.3" />
        </factor>
        <factor level="2" >
            expression
        </factor>
        ...
        <factor level="n" >          n = size of the group
            expression
        </factor>
    </factors>
    <distribution> expression </distribution>
</define-CCF-group>
```

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={e1,e2}

1. Post-processing of cutsets
   - (e1 and e2 and ...) deleted
2. Global constraint
   - NewTopEvent = TopEvent and [not (e1 and e2)]
3. Local subsitution
   - e1 $\rightarrow$ ge1 = (e1 and not e2)
   - e2 $\rightarrow$ ge2 = (e2 and not e1)

Delete Terms (continued)

XML representation

```
<define-exclusive-group name="g1" >
    <basic-event name="e1" />
    <basic-event name="e2" />
    <basic-event name="e3" />
</define-exclusive-group>
```

Recovery Rules

Recovery rules are pairs (hypothesis,consequence) where

- hypothesis is a set of basic events, and
- consequence is a basic event

They are used as a post-processing of minimal cutsets:

- When the custet contains the hypothesis, the consequence is added

XML representation

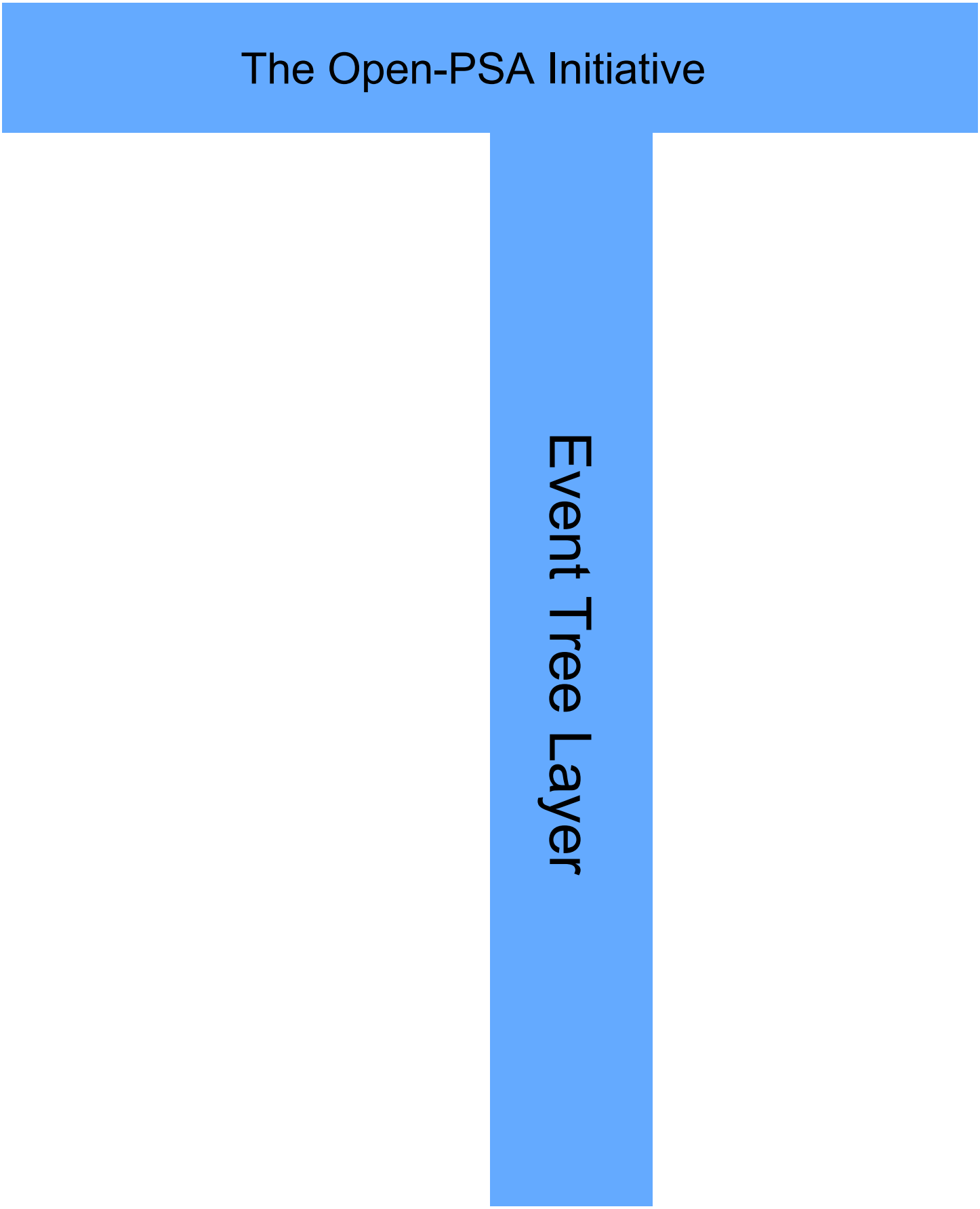
```
<define-recovery-rule name="R1" >
  <hypothesis>
    <basic-event name="e1" />
    <basic-event name="e2" />
  <hypothesis>
  <consequence>
    <basic-event name="e3" />
  <consequence>
<define-recovery-rule>
```

Recovery Rules (continued)

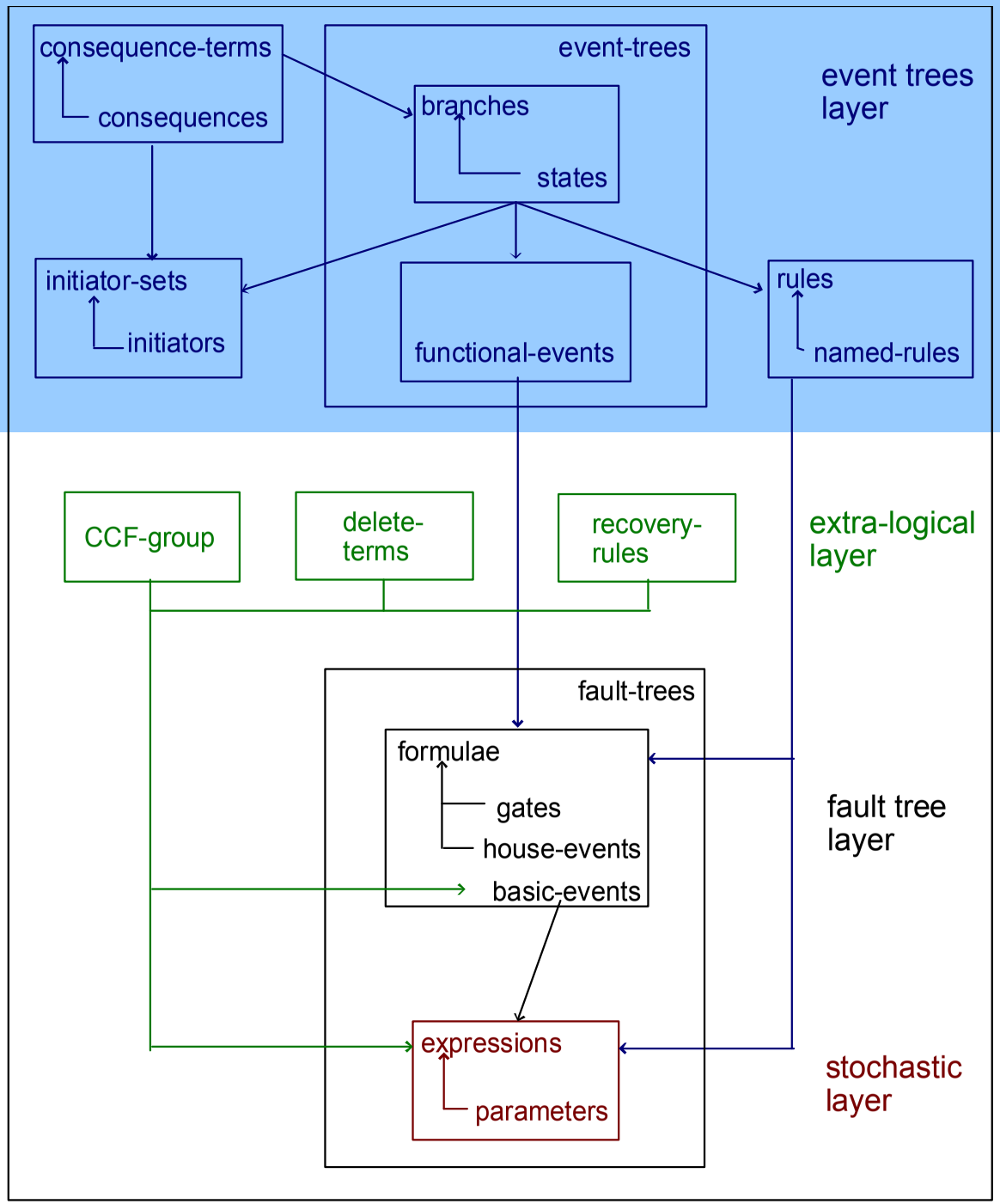
Two possible interpretations/uses of the recovery rule

R = ({e1,e2}, e3)

1. Post-processing of cutsets
   - (e1 and e2 and ...) $\rightarrow$ (e1 and e2 and e3 and ...)
   - (e1 and e2 and ...) $\rightarrow$ (e3 and ...) ?
2. Global constraint
   - NewTopEvent = TopEvent and [not (e1 and e2) or e3]
   - NewTopEvent = TopEvent and [(e1 and e2) $\Rightarrow$ e3]


The Open-PSA Initiative
Event Tree Layer

The Open-PSA Initiative

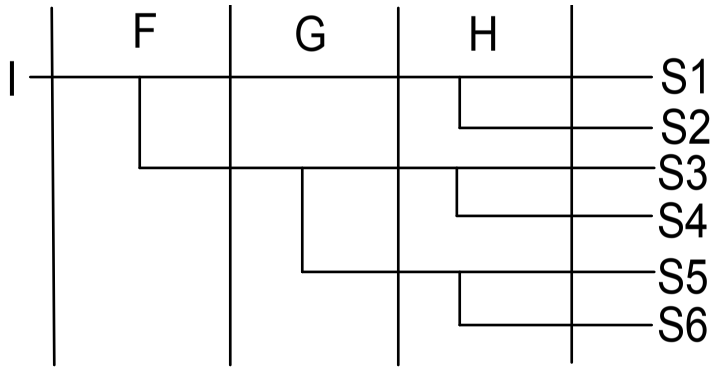


Event-Tree Layer (content)

1. Preliminaries
2. Structure of Event Trees
3. Instructions
4. Consequence groups, initiator groups
5. Mission profile

Preliminaries (1)

Graphical presentation of Event Trees



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 …

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
- …

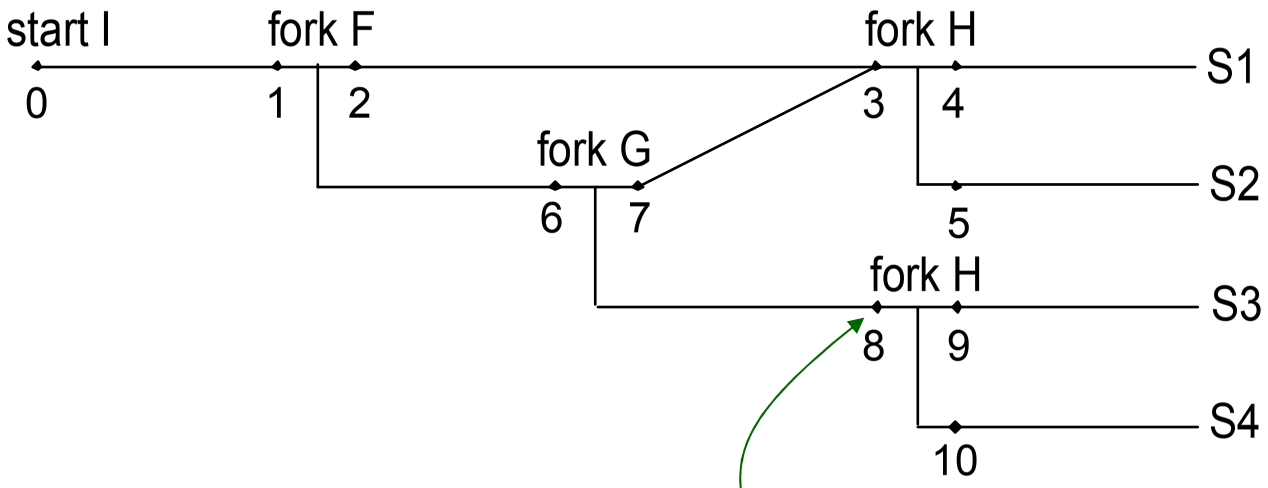
Preliminaries (2)

- 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

Structure of Event Trees (1)



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">          declarations of functional events
      <fault-tree name="FTF" gate="top" />
   </define-functional-event>
   …
   <define-consequence name="S1" />            declarations of consequences
   …
   <path>                                      definition of the structure
      <fork functional-event="F" >
         <path>
            <collect functional-event="F" polarity="success" />
            <fork functional-event="H" >
               …
            </fork>                                          instruction
         </path>
         …
      </fork>
   </path>
</define-event-tree>
```

Instructions (1)

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>
  </fi>

Instructions (2)

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>

Report Layer

Report Layer (content)

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

Descriptions of Results

```
<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>
```