Tools and Procedures at GRS to Automatically Compare and Modify Fault and Event Trees

Joachim Herb, GRS

2012-12-10/11

Next Generation PSA Software, Methods, and Model Representation Standards

Paris, France

Motivation

PSA have become state-of-the-art since several decades (e.g. in Germany PSA are obligatory since the 1990s)

⇒ New versions of a PSA are based on already reviewed older versions

⇒ Therefore it is important to identify the differences between different versions of a PSA

Several fault tree analysis cases (e.g. of highly redundant systems) require structurally similar fault trees

- Modeling of common cause failures
- Modeling of digital I&C systems

⇒ Generate fault trees “automatically”, i.e. let the computer do the work

⇒ Create a programming language (domain specific language) to model fault trees

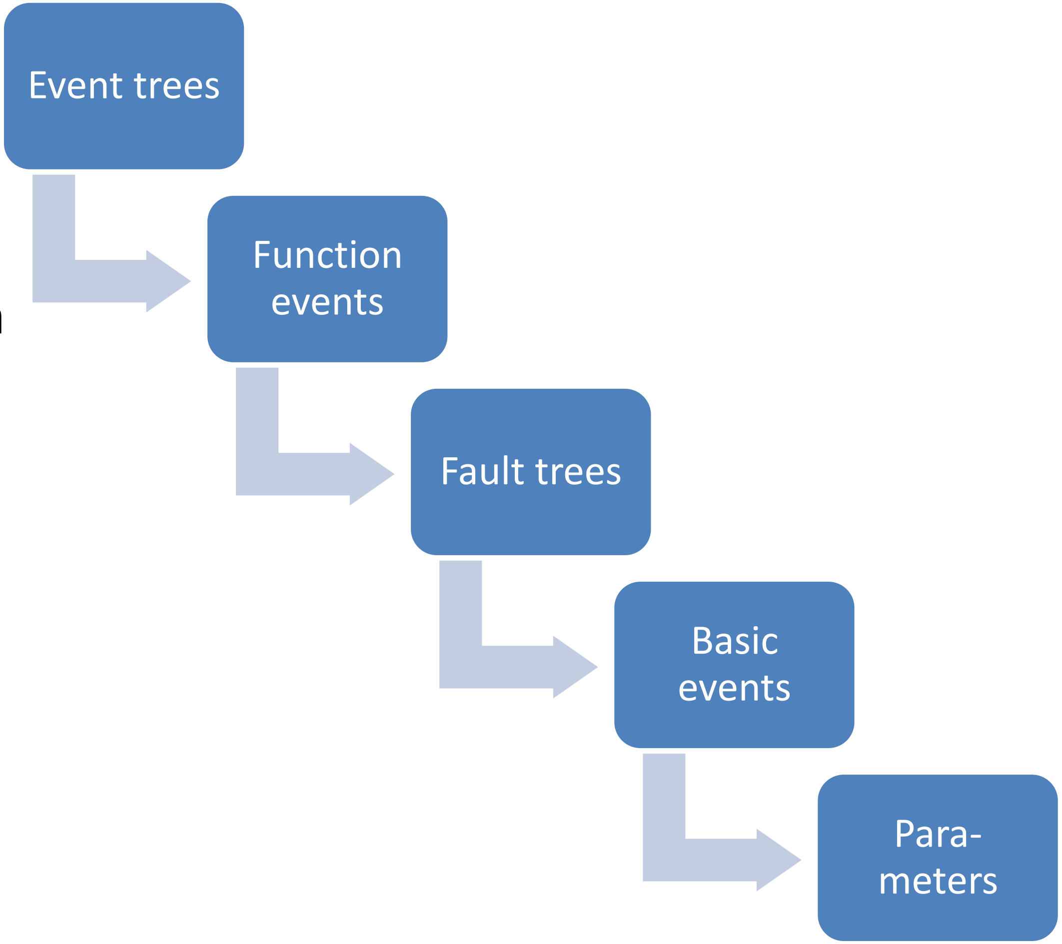
- Most PSA in the world (and all within Germany) use RiskSpectrum®

⇒ Common interface to stored models and data

Review Approaches for PSA Updates

Top-down

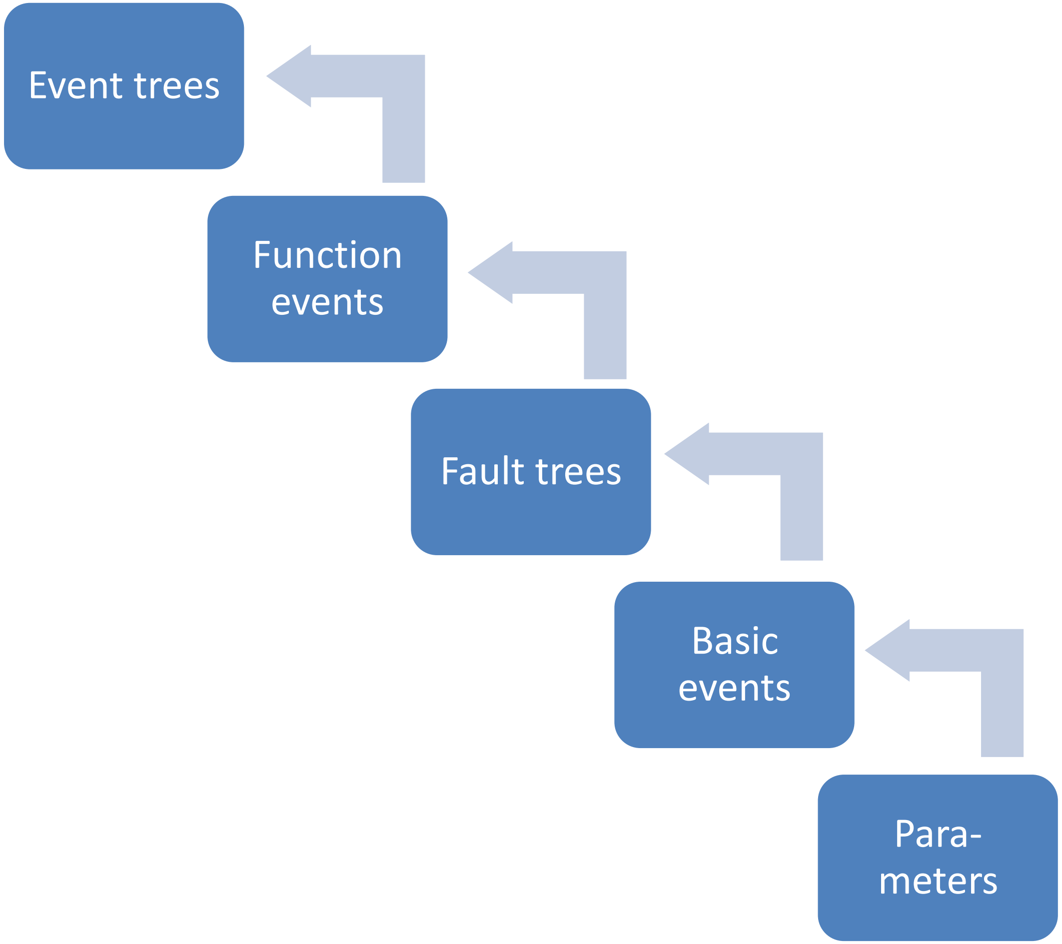
- Start from event trees:
  - Compare sequences and consequences
- Recursively check for changes in
  - Function events
  - Fault trees
  - Basic event
  - Parameters



Review Approaches for PSA Updates

Bottom-up

- Start from parameters
- Find effects on
  - Basic events
  - Fault trees
  - Function events
  - Event trees' sequences and consequences



Architecture of GRS Tools to Support the Review of PSA Changes

Library of classes representing:

- Event trees
- Fault trees (Fault tree pages)
- Parameters

Different tools:

- Extract data, fault/event tree models from RiskSpectrum®
- Identify differences of fault/event trees

Find fault trees which are part of different versions of the PSA

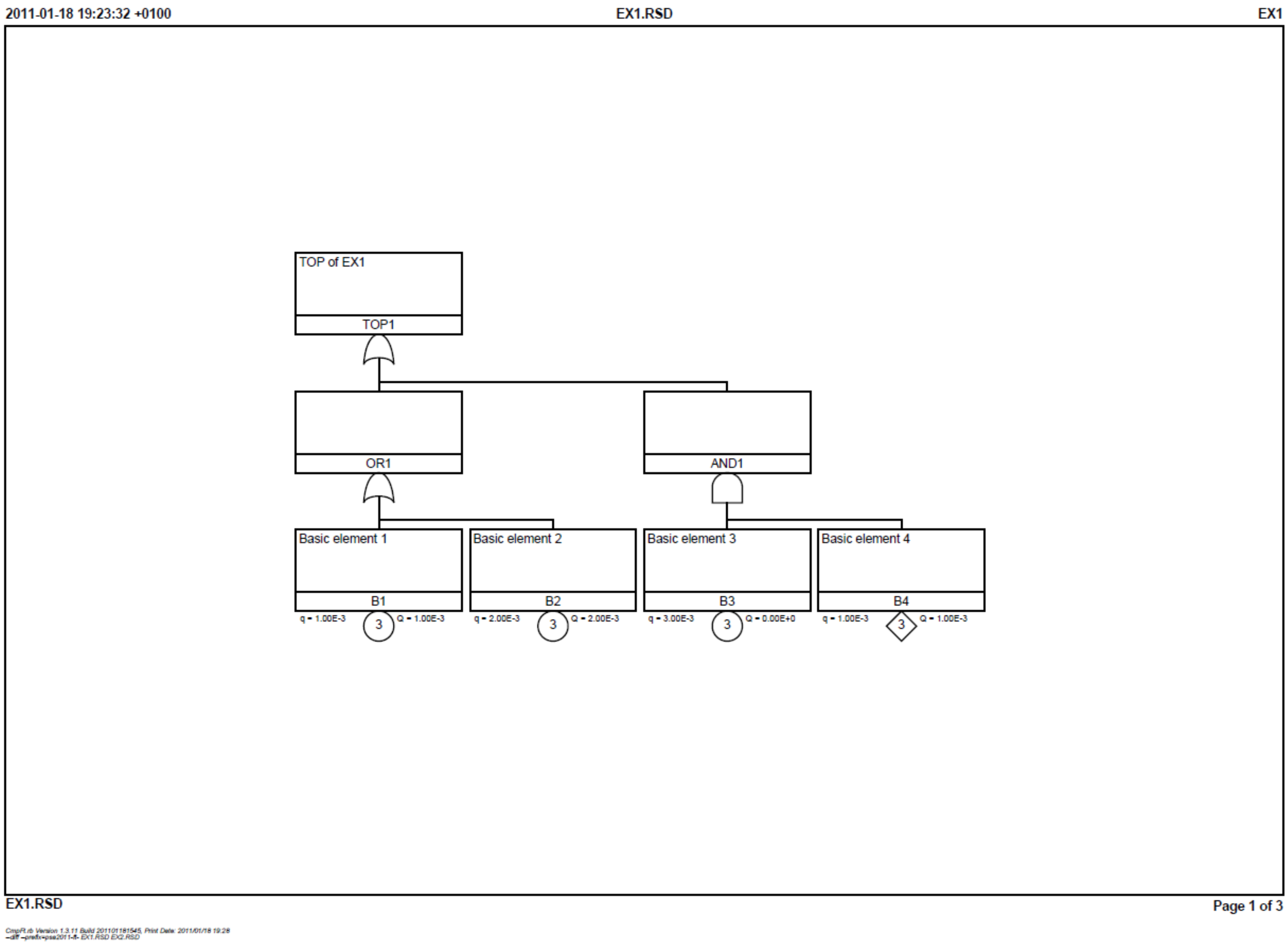
Starting from the TOP gate, check for changes of these fault trees:

- Type of node (AND, OR, K/N, basic event, transfer gate, …)
- Transfer source (fault tree used as input for transfer gate)
- ID of node

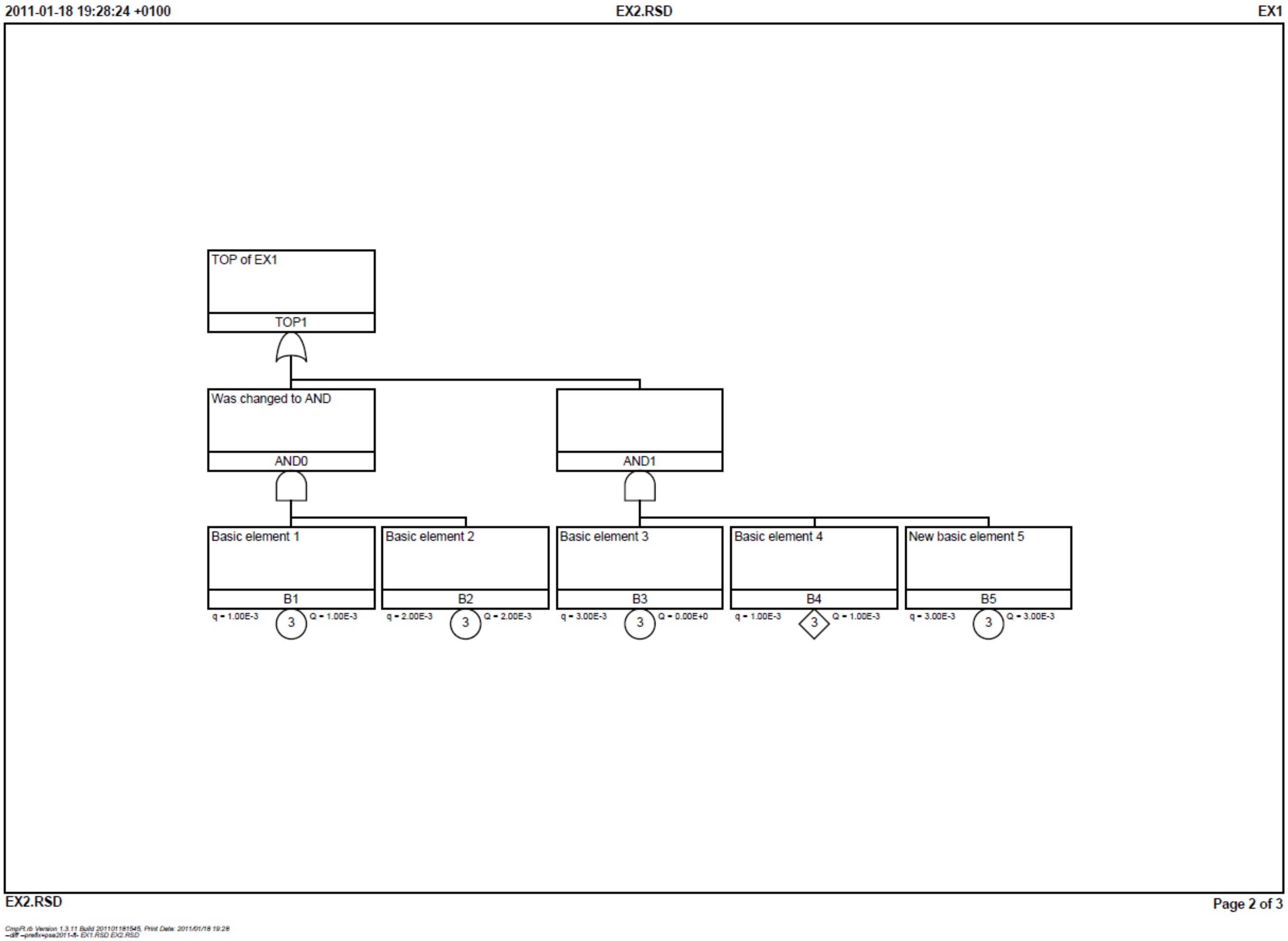
Repeat recursively for all children nodes

Output differences-graphs in pdf files

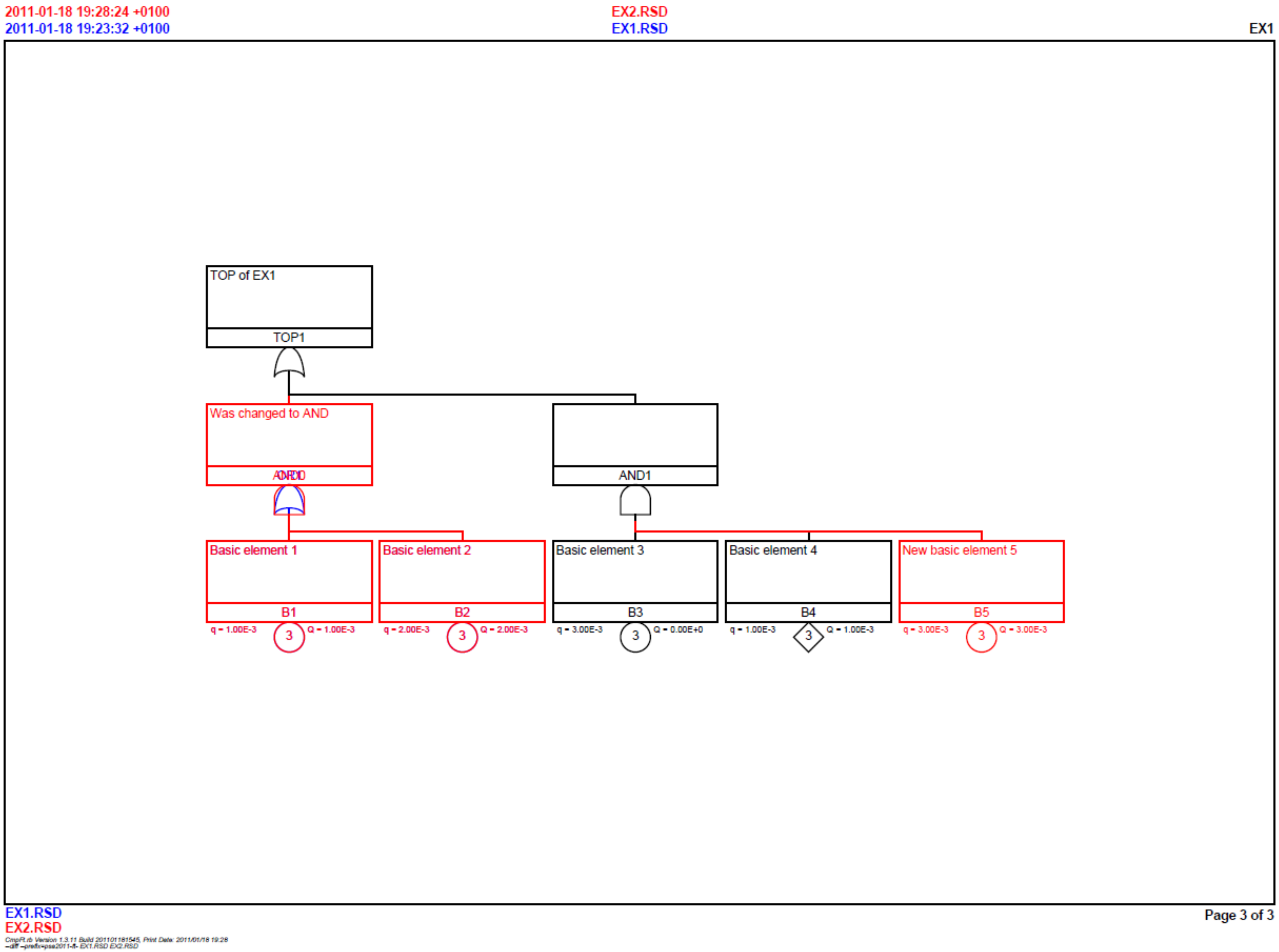
GRS Tool: Generate Differences-Graphs of Fault Trees



GRS Tool: Generate Differences-Graphs of Fault Trees



GRS Tool: Generate Differences-Graphs of Fault Trees



GRS Tool: Trace Dependencies and Changes of Fault Tree Logic

”Expand“ fault trees up- or downwards:
Replace recursively all transfer gates through the referenced fault trees

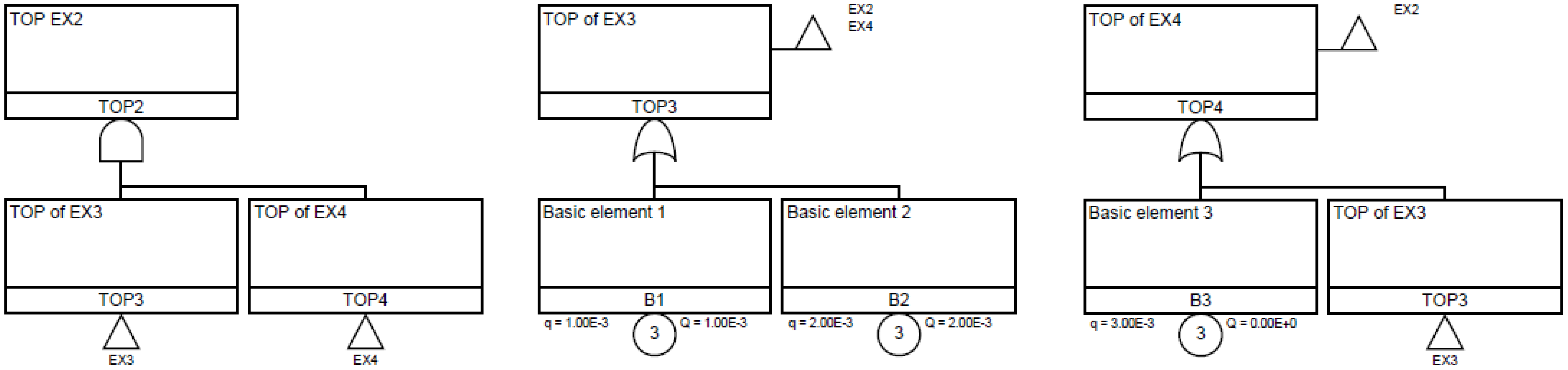
Downward expansion:

- Generate lists of all gates and basic events connected to a specified TOP gate
- Mark a specified fault tree as ***recursively equal***, if:
  - Fault tree (page) itself has not changed
  - All fault trees (pages) referenced by input transfer gates have not changed
- For a ***not recursively equal*** fault tree output the differing referenced fault trees (pages)

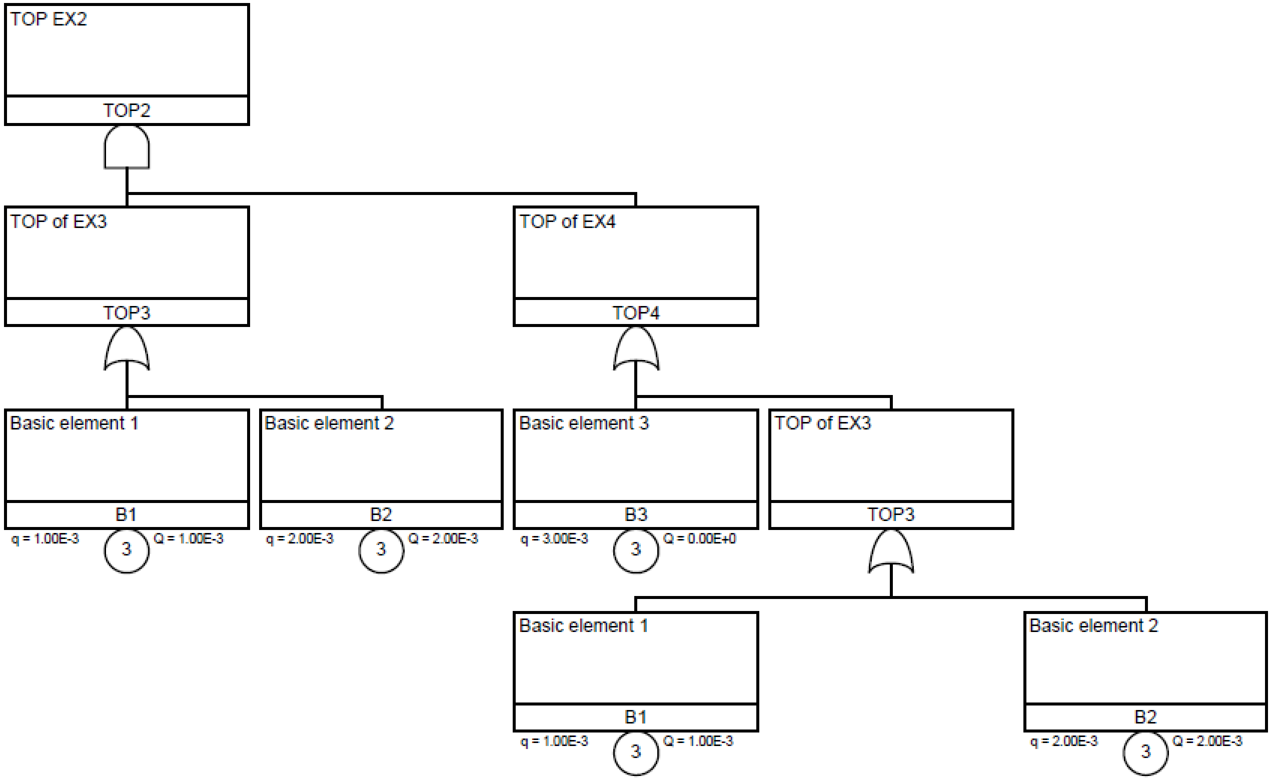
Upward expansion:

- Generate list of all TOP gates affected by changes in a certain fault tree

Example for Downward Expansion



Example for Downward Expansion



Example Applications of the GRS Tools

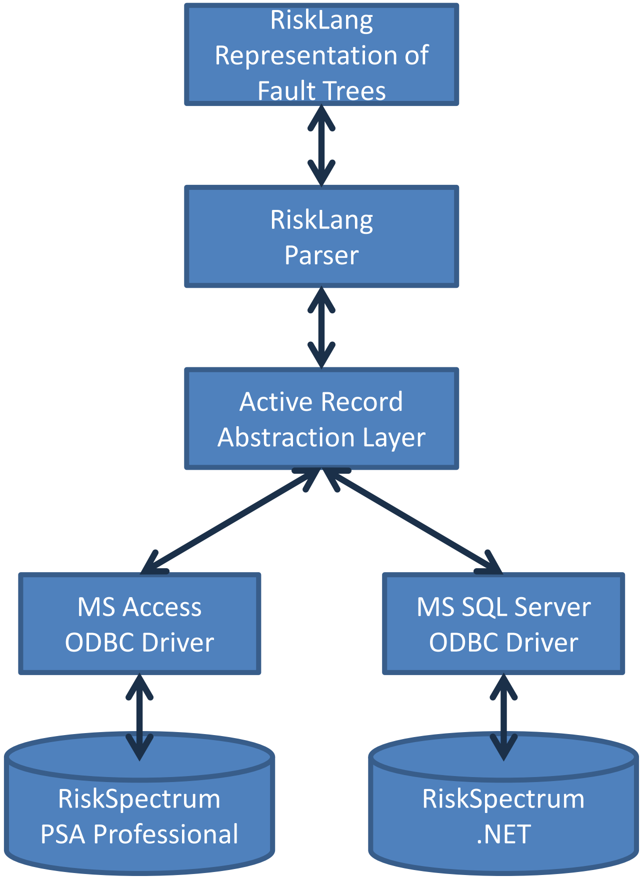
- Review of an updated L1 PSA
- Quality assurance processes during establishment of a Fire PSA

Case	# of Fault Trees	# of Differing Fault Trees	# of Event Trees	# of Differing Event Trees
Review (PSA L1)	2554/2570	53	37/38	17
Fire PSA	1902/1928	616	75/75	33

Data Link to Modify RiskSpectrum® Database using GRS RiskLang

Internal data link to RiskSpectrum®

- RiskLang parser generates Ruby objects
- Open source library ActiveRecord is used to map these objects onto the tables in the RiskSpectrum® database
- Data are added to RiskSpectrum® database through ODBC-drivers using SQL commands



Housekeeping

- All imported data are tagged in RiskSpectrum®
- User and edit date is set to new value
- Review/approval information is reset

Export fault trees from RiskSpectrum® in RiskLang

- Generate templates

RiskLang Specification to Model Fault Trees

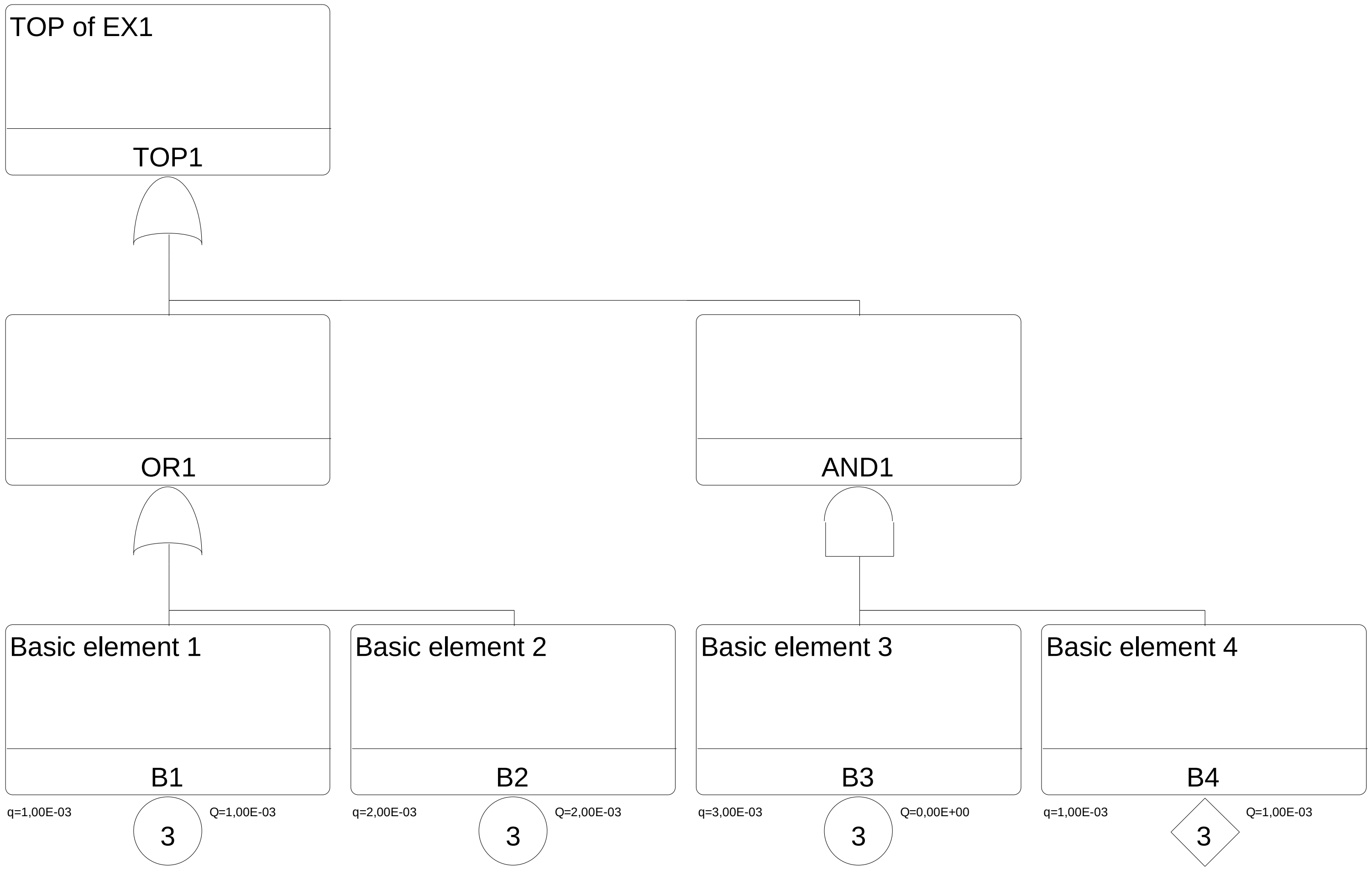
Based on the programming language Ruby

Add three commands to create fault trees, fault tree nodes, basic events/gates

Command	Argument	Data Type	Required
FT	:ID	String (max length 21)	Yes
	:Top	FTNode	Yes
	:Text	String	Optional
FTNode	:Event	Event	Yes
	:Transfer	Event	Yes
	:Pos	Integer	Optional
	:InLevel	Integer	Optional
	:Children	Array of FTNode	Optional
Event	:ID	String (max length 21)	Yes
	:Type	Event Type	Yes
	:Text	String	Optional
	:Model	Integer	Optional
	:CalcType	Integer	Optional

Event Types	RiskSpectrum® Equivalent
:circle	Basic event
:diamond	Diamond basic event
:house	House event
:orgate	OR gate
:andgate	AND gate
:kofn	K/N gate
:norgate	NOR gate
:nandgate	NAND gate
:xorgate	XOR gate
:comment	Comment gate
:continue	Continuation gate
:notfound	-

Exemplary Fault Tree

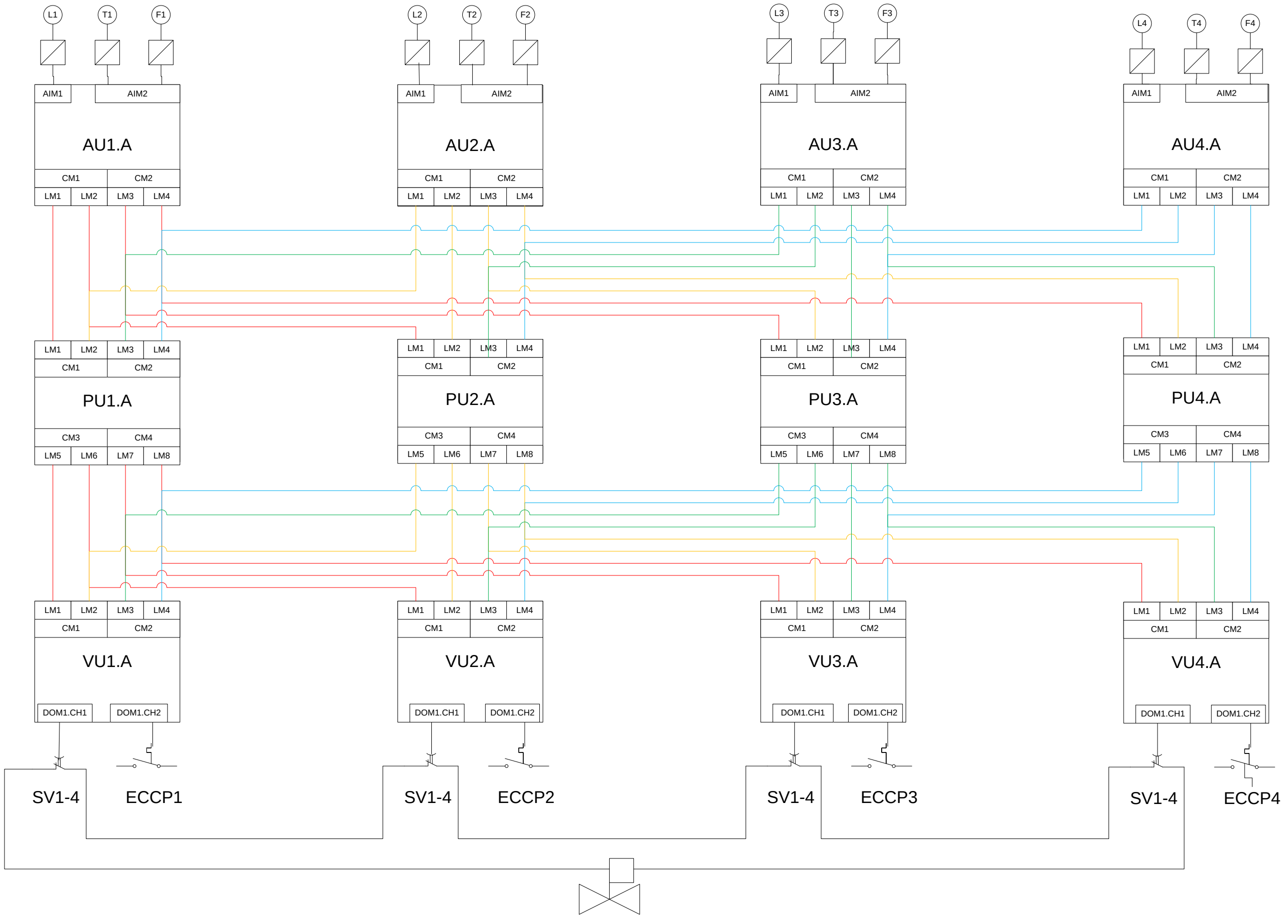


Exemplary Fault Tree

```
require "RiskRobot/ParseFT"
RiskSpectrumConnection.init('C:\temp\k2.rpp')

FT(:ID=>"EX1", :Text=>"Example fault tree", :top=>
  FTNode(:Event=>Event(:ID=>"TOP1", :Type=>:orgate, :CalcType=>1,
    :Text=>"TOP of EX1"), :Pos=>1, :InLevel=>0, :Children=>[
      FTNode(:Event=>Event(:ID=>"OR1", :Type=>:orgate, :CalcType=>1),
        :Pos=>1, :InLevel=>1, :Children=>[
          FTNode(:Event=>Event(:ID=>"B1", :Type=>:circle, :Model=>3, :CalcType=>1,
            :Text=>"Basic element 1"), :Pos=>1, :InLevel=>1),
          FTNode(:Event=>Event(:ID=>"B2", :Type=>:circle, :Model=>3, :CalcType=>1,
            :Text=>"Basic element 2"), :Pos=>2, :InLevel=>1)
      ]),
      FTNode(:Event=>Event(:ID=>"AND1", :Type=>:andgate, :CalcType=>1),
        :Pos=>3, :InLevel=>1, :Children=>[
          FTNode(:Event=>Event(:ID=>"B3", :Type=>:circle, :Model=>3, :CalcType=>1,
            :Text=>"Basic element 3"), :Pos=>3, :InLevel=>1),
          FTNode(:Event=>Event(:ID=>"B4", :Type=>:diamond, :Model=>3, :CalcType=>1,
            :Text=>"Basic element 4"), :Pos=>4, :InLevel=>1)
      ])
  ])
)
```

System Overview of Generic Digital I&C Example



AIM	Analogous Input Module
ASM	Analogous Signal Module
AU	Acquisition Unit
BP	Backplane
CM	Communication Module
CP	Communication Processor
DIM	Digital Input Module
DOM	Digital Output Module
ECCP	Emergency Core Cooling Pump
F	Flow Sensor
L	Level Sensor
LM	Link Module
PM	Processing Module
PU	Processing Unit
SV	Solenoid Valve
T	Temperature Sensor
VU	Voting Unit

Generic Fault Trees of Digital I&C System

Fault Tree ID (wildcards #{r}, #{r2}, #{m}, #{comb})	Description	Number of fault trees
2O4_VU#{r}.A_#{comb}_#{m}	OR of all combinations leading to a failure of 2 out of for 4 voter	372
AU#{r}.A_#{m}_FNSS	Non-self-signaling failure AU	12
AU#{r}.A_FSS	Self-signaling failure AU	4
COM_AU#{r}.A_PU#{r2}.A	Failure communication AU->PU	16
COM_PU#{r}.A_VU#{r2}.A	Failure communication PU->VU	16
ECCP#{r}_#{m}	Failure to start ECCP	8
F2M_PU#{r}.A_#{comb}_#{m}	**All combinations leading to a failure of second min processing unit**	**516**
OUT_AU#{r}.A_#{m}_FNSS	Non-self-signaling failure of output AU	12
OUT_AU#{r}.A_#{m}_FSS	Self-signaling failure of output AU	12
OUT_PU#{r}.A_#{m}_FNSS	Non-self-signaling failure of output PU	12
OUT_PU#{r}.A_#{m}_FSS	Self-signaling failure of output PU	12
PU#{r}.A_FNSS	Non-self-signaling failure PU	4
PU#{r}.A_FSS	Self-signaling failure PU	4
PU#{r}_F2M_#{m}_FNSS	Non-self-signaling failure of second min due to input failures from AU in PU	12
SIG_RED#{r}_FSS_#{m}	Self-signaling failure of analogous input AU	12
SV#{r}_#{m}	Failure to control SV	8
VU#{r}.A_FNSS	Non-self-signaling failure VU	4
VU#{r}.A_FSS	Self-signaling failure VU	4
VU#{r}_2O4_#{m}_FNSS	Non-self-signaling failure of 2 out of 4 due to input failures from PU in VT	12

$\sum = 1052$

Conclusions

Use cases exists besides the „normal“ application of PSA software (FT, consequence, sequence, importance, MCS analyses):

- Provide data for documentation
- Modify existing ET/FT models or create new ones automatically
- Trace chances
- Review
- …

Lessons learned: One software product can not foresee all user requirements

⇒ Need an **interface** (open file format or programming interface) to extend PSA software (i.e. RiskSpectrum®):

- ✓ Import/export PSA models and data (partly implemented by GRS database link/via OpenPSA file format)
- ✗ Trigger computations
- ✗ Access results of analyses
- ✗ Call (external) user defined functions from PSA software