

# ABC Encapsulation

## *Fault tree generator*

---

Copyright © 2008 Dassault Systemes

Version 1.0

### *Abstract*

*ABC* is a new generation fault-tree generator based on AltaRica DataFlow language. Developed as part of ISAAC project, it aims to overtake limits of historical generator of BPA DAS.

Firstly, *ABC* overtakes current fault tree generation limit by considering:

- instantaneous events
- different types of synchronizations (not only Common Cause Failures)
- models using flows in guards of transitions.

Eventually, it will manage systems with loops.

This chapter deals with the integration of this engine in BPA DAS.

# Table of Contents

---

<b>FaultTree generation with ABC .....</b>	<b>4</b>
Install/Parameter .....	4
FaultTree generation requirements .....	4
FaultTree generation launching .....	5
Result of FaultTree generation .....	7

# FaultTree generation with ABC

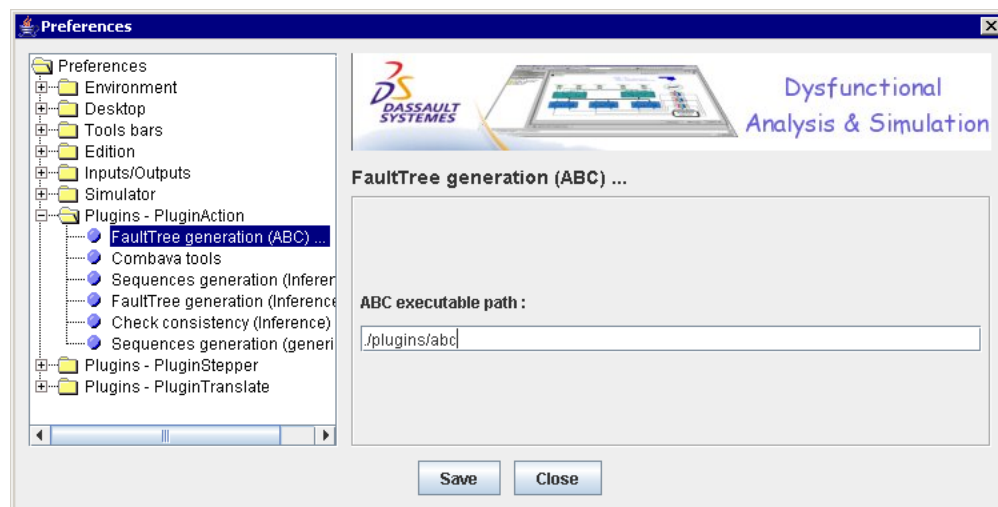
FaultTree generation consists in compiling AltaRica model into boolean formula. Compilation algorithm use a deductive approach from groups of "independant" components. Boolean formula generation is made with reachability graphs associated to each group.

Fault-trees are generated in *Aralia* format. This format contains equations (tree structure), laws, attributs associated with model events, and parameters of generated laws. Generated trees can be either used directly in *Aralia* with command scripts, or imported in BPA SSA or in another fault-tree-editor which supports *Aralia* native format.

## Install/Parameter

FaultTree generation is based on *ABC* software. This software is available in plugin installation directoty.

Users preferences allow to specify *ABC* location.



## FaultTree generation requirements

It's important to notice that AltaRica model can be dynamic (reconfiguration, maintenance, ...) and that a fault-tree is an instantaneous view of a system, that's to say a static view. Therefore, an AltaRica model may not be compilable into boolean formula.

FaultTree generation implies some model restrictions. These restrictions are:

- Dynamic behavior of components

Because fault-tree is a static model, it's impossible to compile a model whose final state depends on faults order. The two following restrictions ensure a static behavior:

- For each fault-combination, if a fault-permutation is workable, every other permutation must be workable and must lead to the same state of the component.
- Transitions must be independant of component flows. In other words, component reachability graph must only depends on its initial state and its local failures.

- Uncomplete or unconsistant component(s)
- Too complex component(s)

A too complex component will generate a local combinative explosion during calcul of its reachability graph.

- Looped System

A looped system is a system with circular definition of its flow variables. That is to say, a X variable is define with a set of AltaRica assertions which depend on variable X.

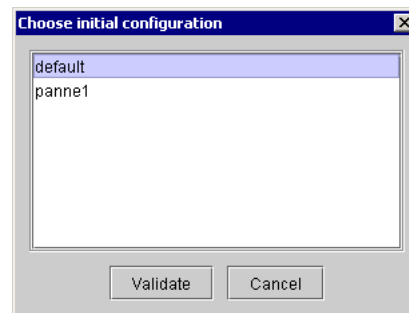
These restrictions ensure a good behavior of compilation algorithm. Check properties (when config is correct) enables to spot things that don't fit these restrictions.

## FaultTree generation launching



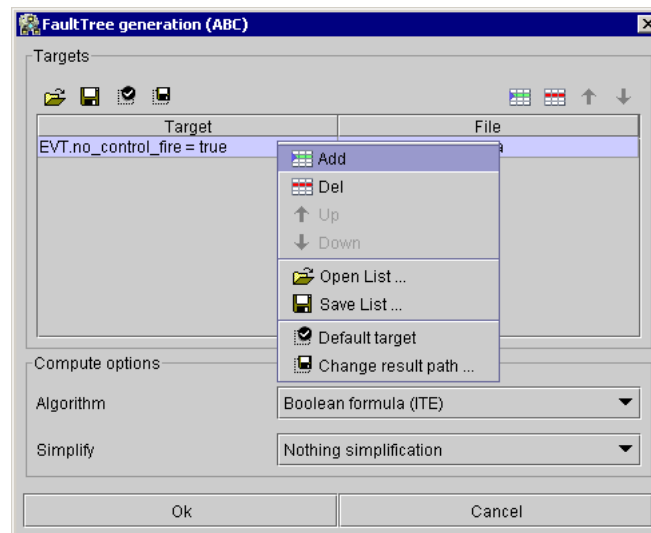
In order to launch FaultTree generation, use the **FaultTree generation (ABC)...** command.

A window is displayed when model has many initial configurations (**System** menu, **Initial states ...** command), in order to select initial configuration to take into account.



Feared event is defined thanks to a calculation-target. The target corresponds to a specific value of a model variable (If the variable is equal to the selected target the feared event happens). Calculation parameters are usually associated with a calculation-target (for exemple, the result file name).









The following dialog window enables to define one (or several) target(s) which must be taken into account during FaultTree generation. A fault-tree will by generated for each computation target.



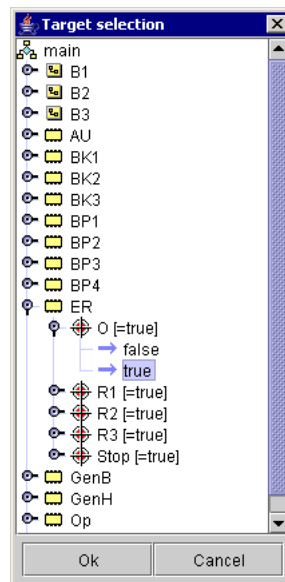
This window manage a list of computation-targets. When this window is validate, every typed information are validate to ensure that there is no incomplete line, no wrong line, targets correspond to a variable of processed model, file can be written ....

Moreover, if result-files already exist, a dialog window will ask for old file deletion.

Some icons/actions enables to modify the list of calculation targets.

	Add a calculation-target (a line in the list). The target is duplicated from a default calculation target.
	Remove selected computation target.
	Climb selected target up.
	Go selected target down.
	Define default computation-target from selected target. Every new target will be created using this default one.
	Save target-list in a XML file in order to re-use it.
	Load a list of computation targets.
	It enables to define result files associated to targets thanks to a pattern which take into account informations linked to calculation targets. The result file name will be define replacing generation tag %xxx% by their contents : %num% => order number in the list, %var% => variable name, %val% => value to use, ...

Double-click on an element of the list enables to edit it with an appropriate editor



Target editor opens a window and shows every model variable in a tree view. Select the value corresponding to the feared event to take into account.

File editor displays a standard file chooser window.

Two **Algorithm** are available in order to generate boolean formula :

1. **Boolean Formula (SOP)** : compile sum of product into simple boolean formule.
2. **Boolean Formula (ITE)** : compile sum of product into a tree structure 'If ... then ... else ...' (more powerful in case of weighty systems).

Three **Simplify** levels are available :

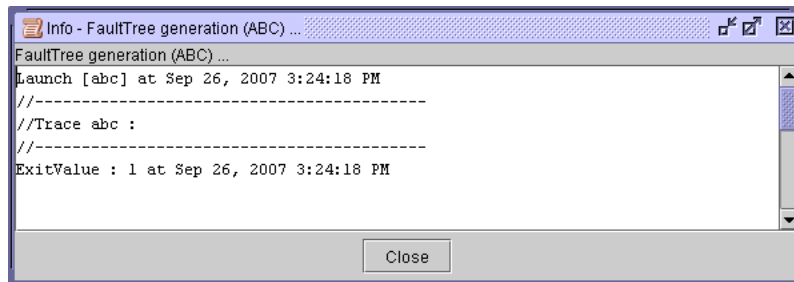
- **Nothing simplification**
- **Constant propagation** : In generated fault tree, boolean connectors having true or false arguments are simplified.
- **Unreferenciation** : simplifies the tree by deleting useless intermediate variables.

When this window is correctly filled and validated, **ABCTree** task is added to the task manager of BPA DAS

## Result of FaultTree generation

At the end of fault tree generation, a window is displayed. It shows *ABC* execution trace.

If everything is allright, this trace should be empty.



In the contrary (looped system, ...), *ABC* displays an error message.

