

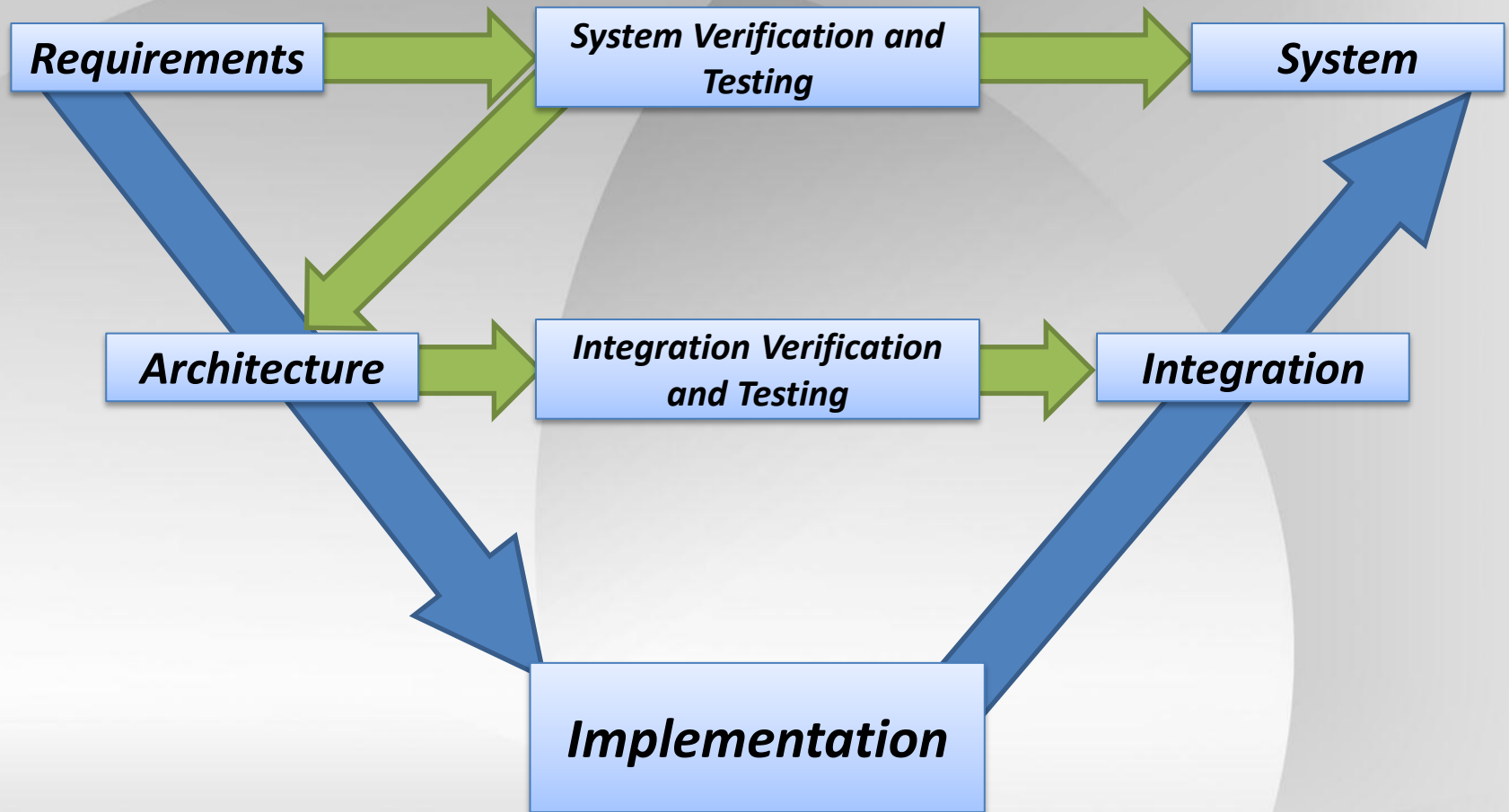
NuSMV3: a framework for Formal Model Based Safety Assessment

*Marco Bozzano, Roberto Cavada,
Alessandro Cimatti, Cristian Mattarei
Fondazione Bruno Kessler, Trento (Italy)*

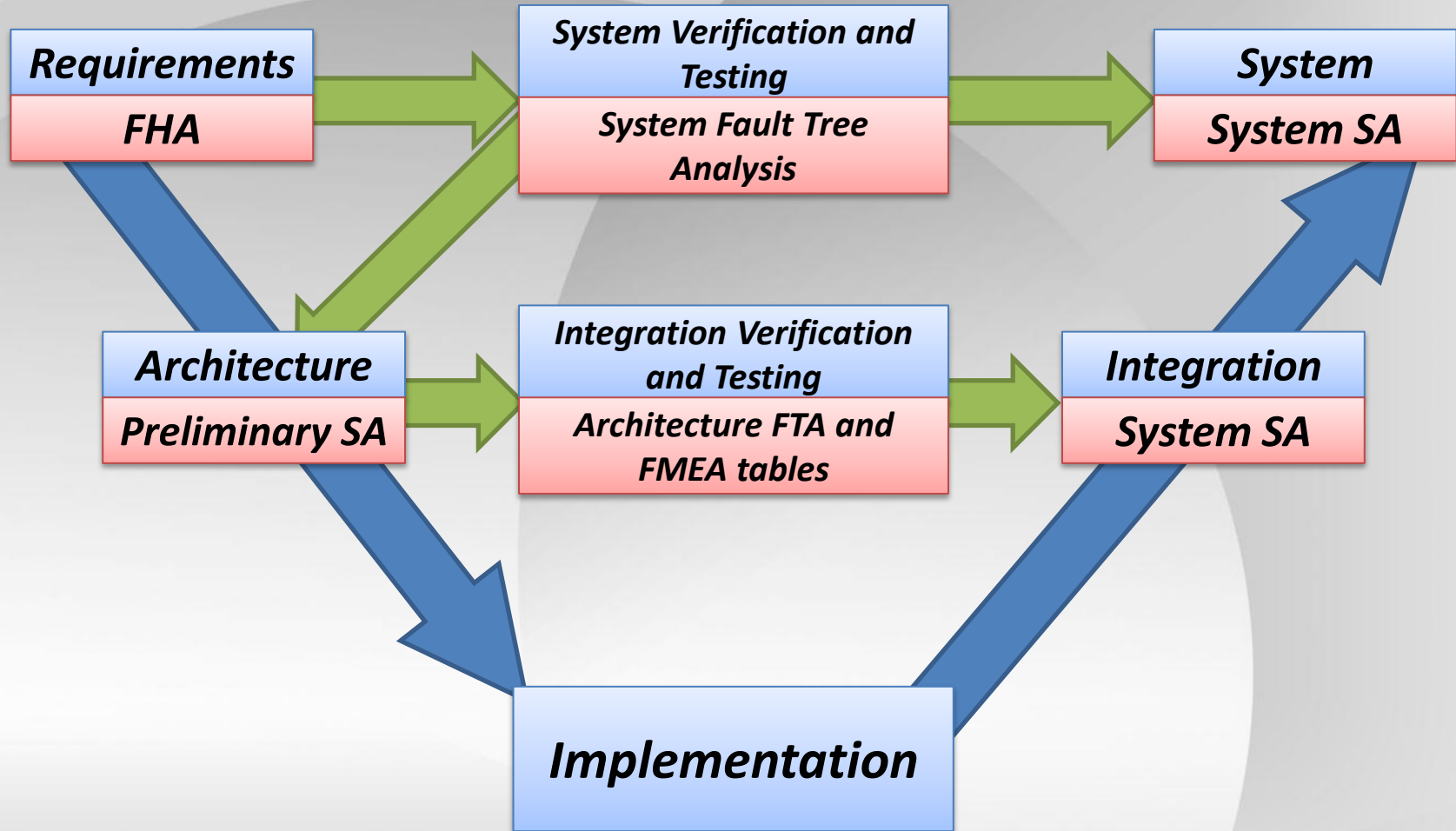
Roadmap

- ***Formal Model Based Safety Assessment***
- ***Formal Safety Assessment***
 - *Current approach*
 - *Automated Fault Extension*
- ***NuSMV3 formal verification framework***
- ***Next challenges***

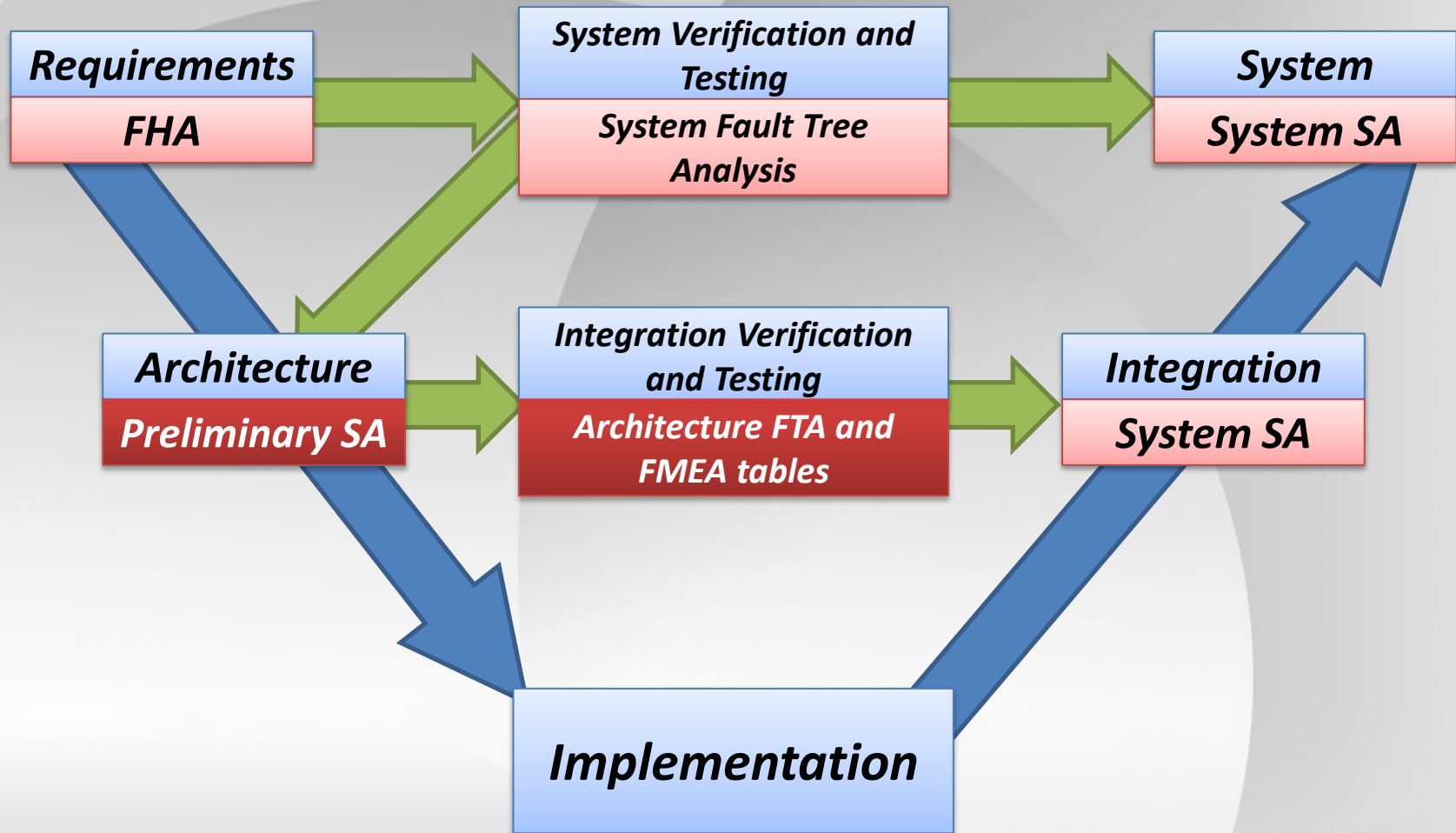
Model Based Safety Assessment



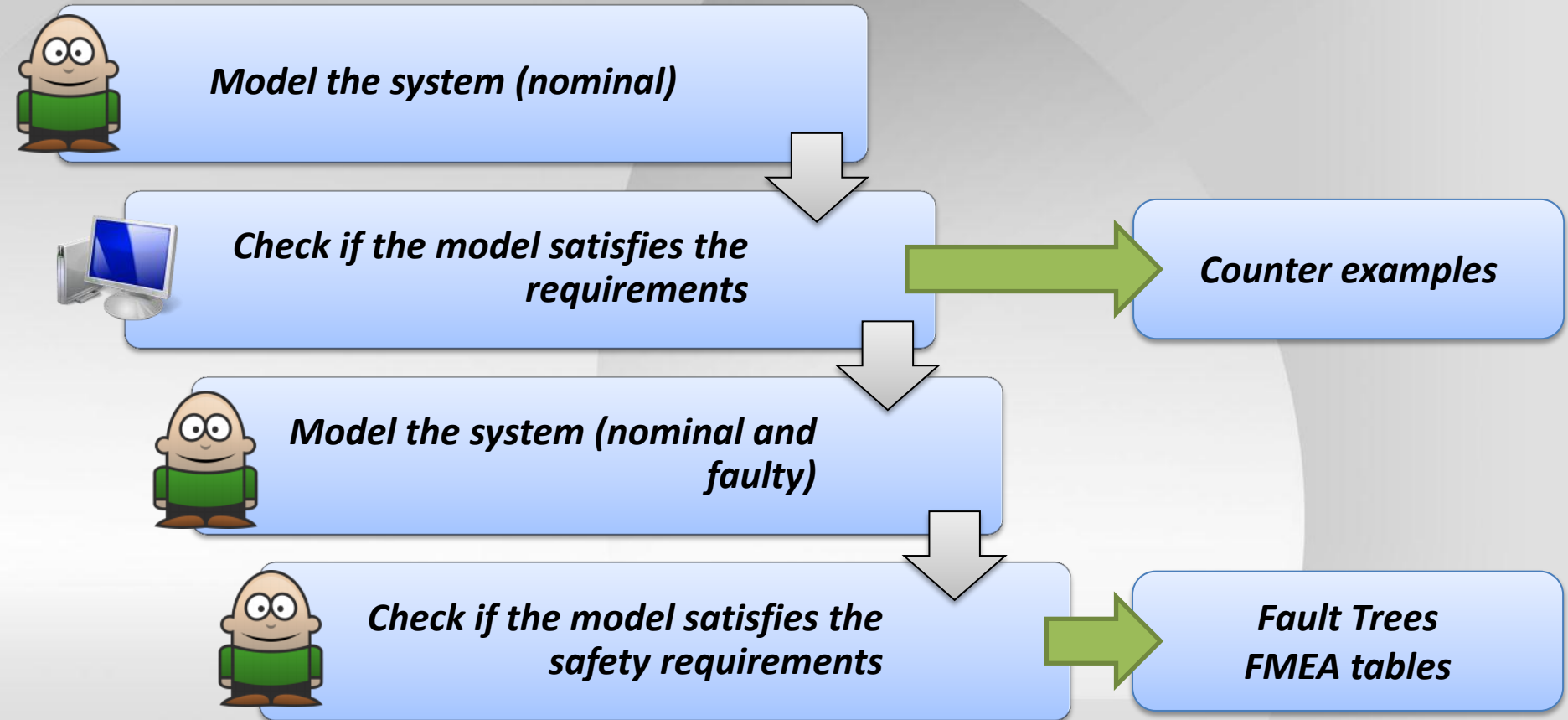
Model Based Safety Assessment



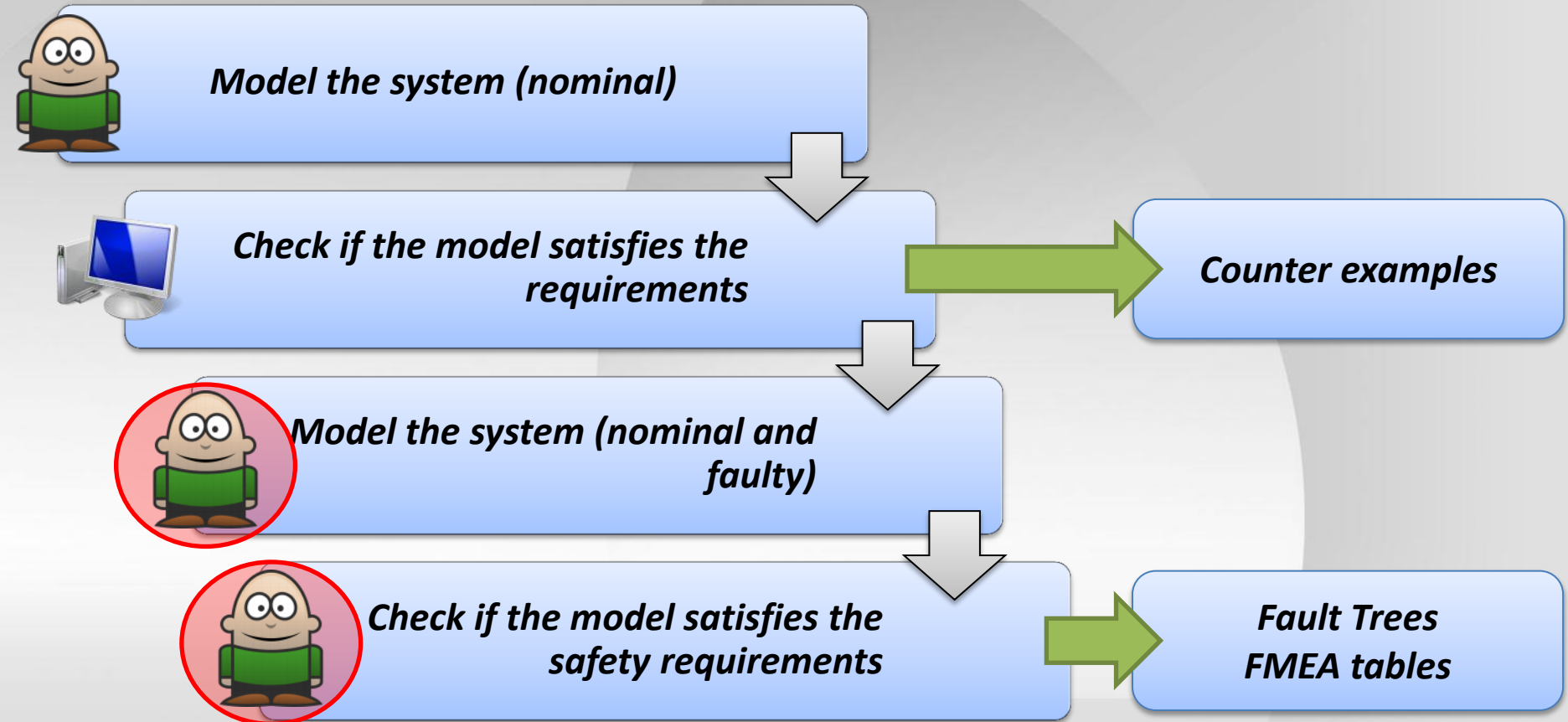
Model Based Safety Assessment



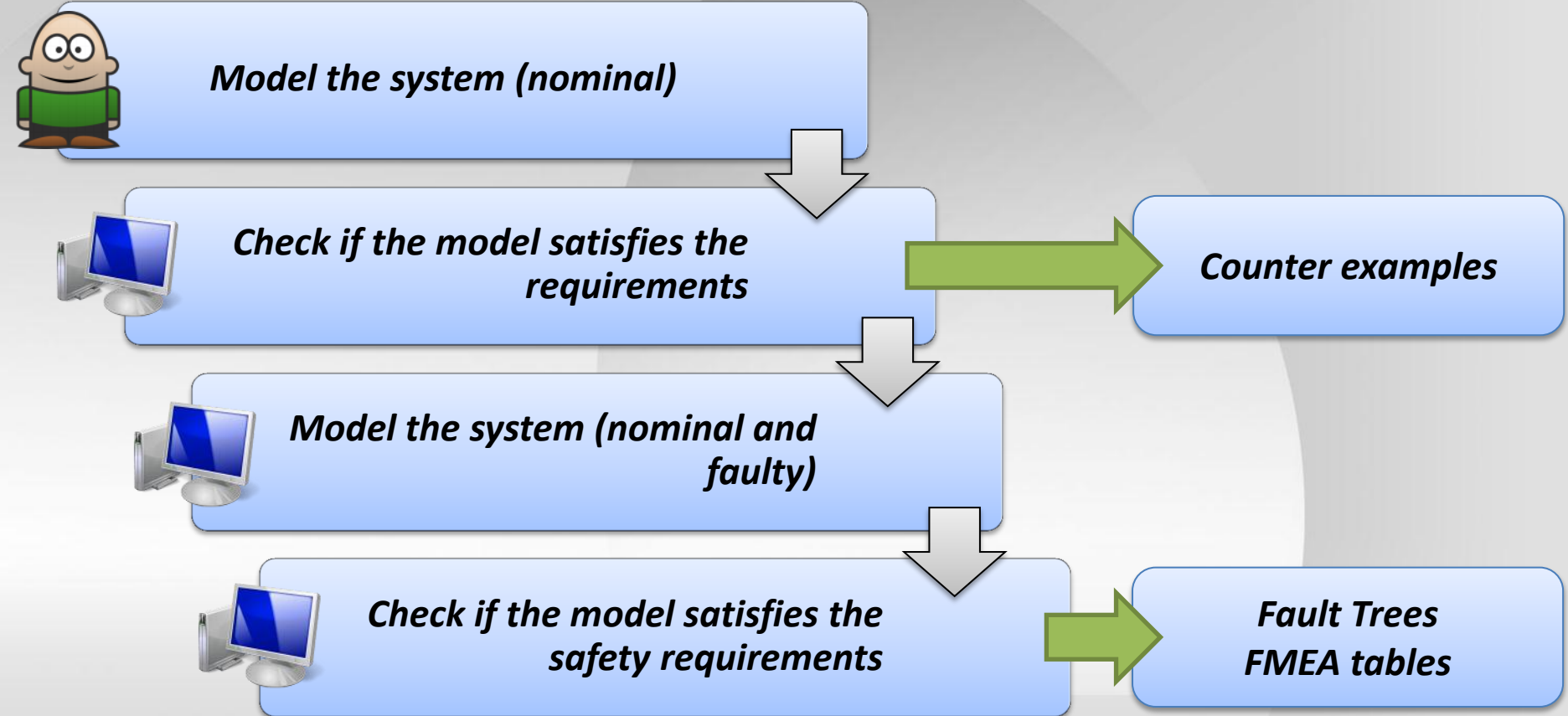
Model Based Safety Assessment



Model Based Safety Assessment



Model Based Safety Assessment

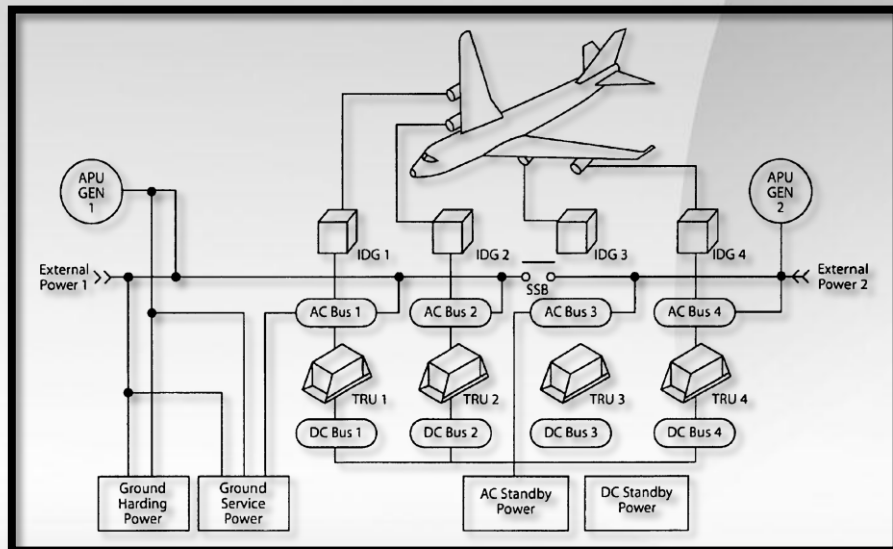


Roadmap

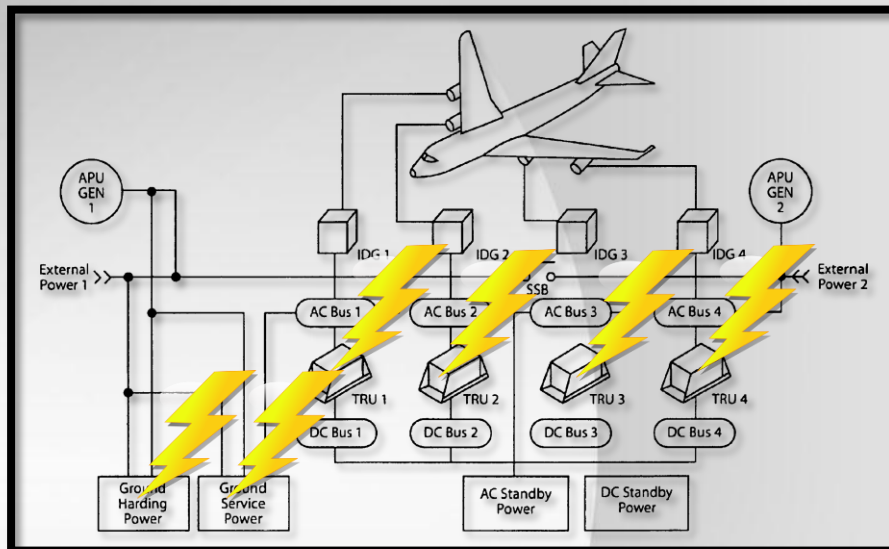
- *Formal Model Based Safety Assessment*
- ***Formal Safety assessment***
 - *Current approach*
 - *Automated Fault Extension*
- *NuSMV3 formal verification framework*
- *Next challenges*

Fault Extension: the idea

Formal model (nominal)



Faulty model (extended)



Manual Extension

Manual Extension

PROS

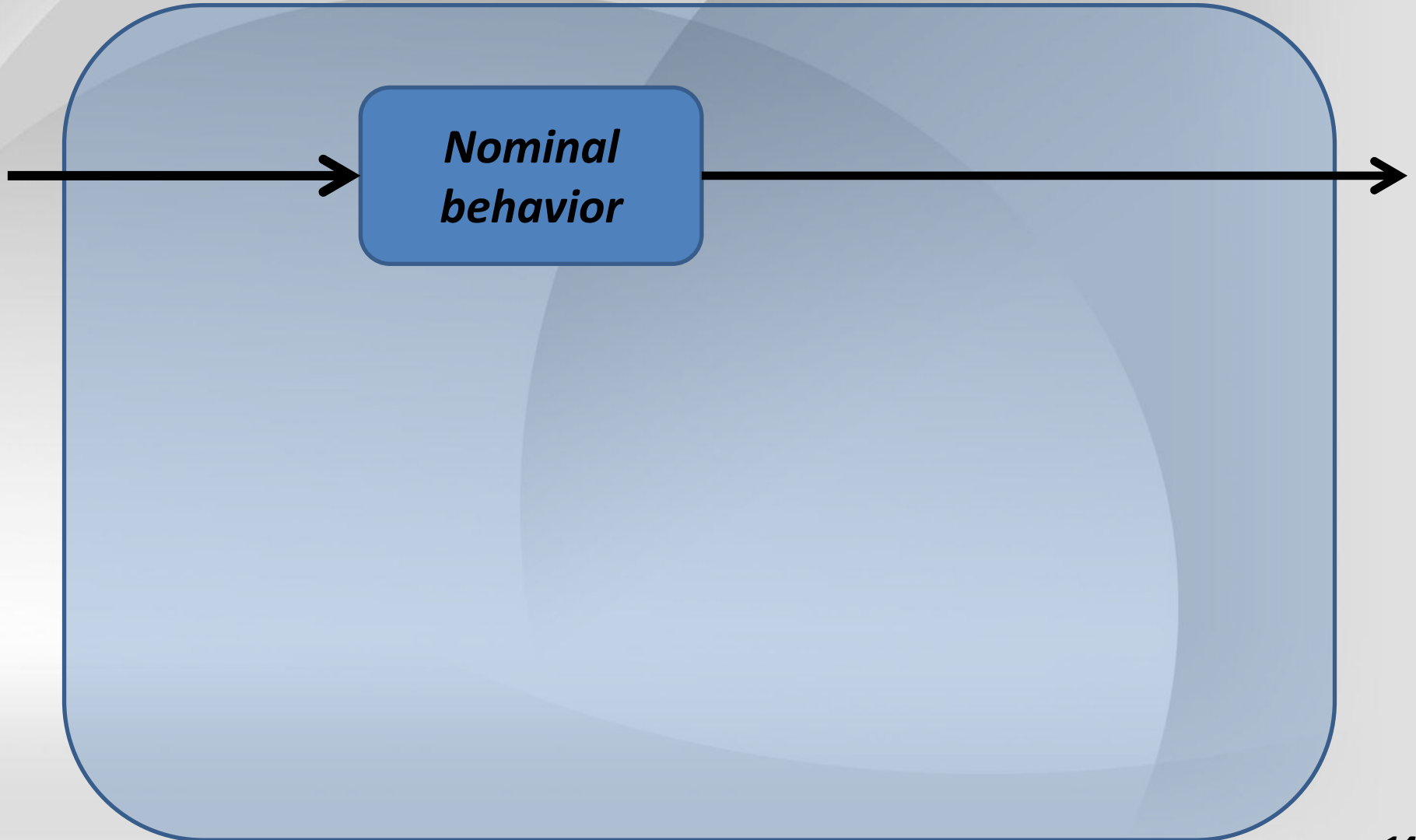
- *Highly expressive*
- *Does not need extra tools*

CONS

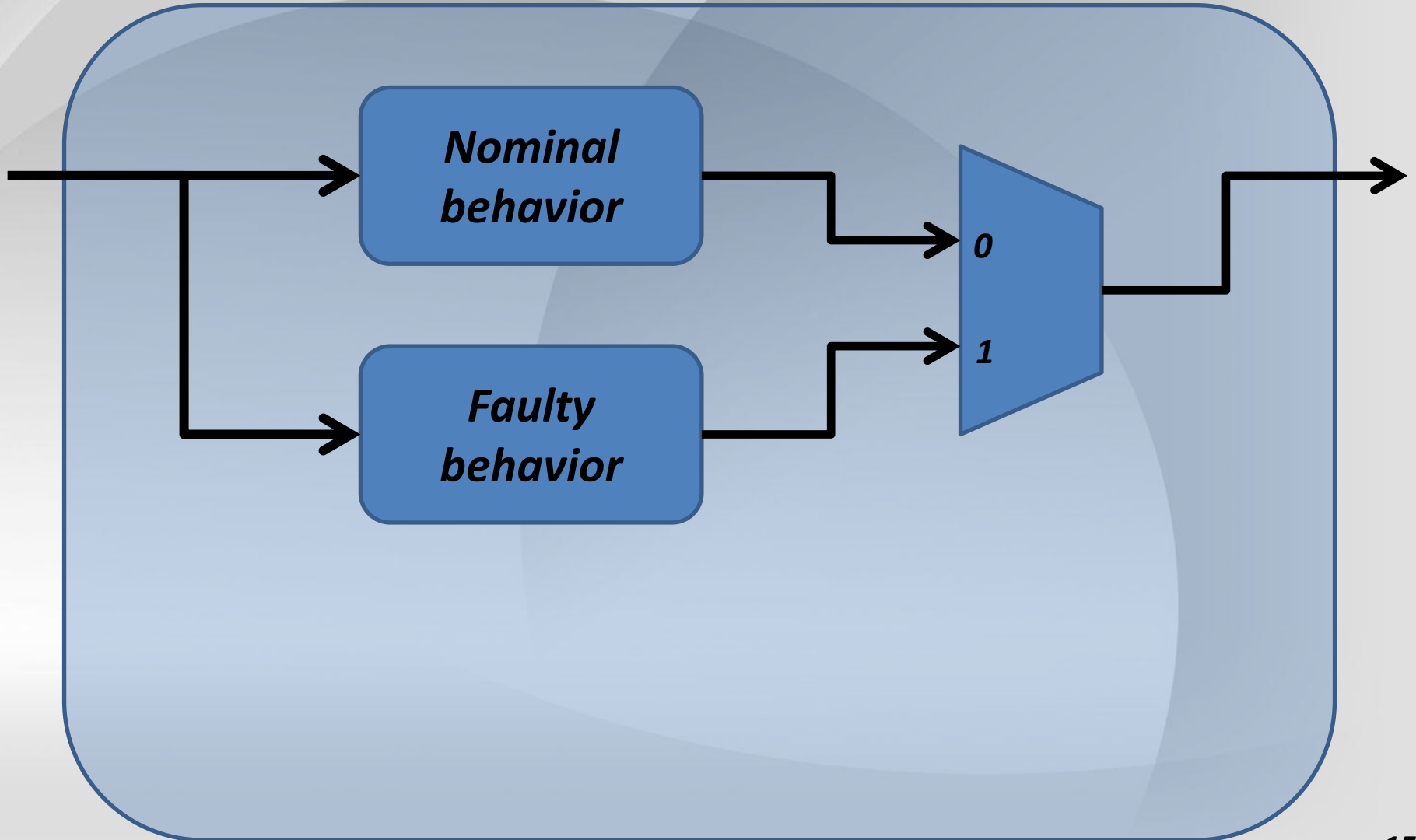
- *Error prone*
- *Not traceable process*
- *Time consuming*

Fault Injection

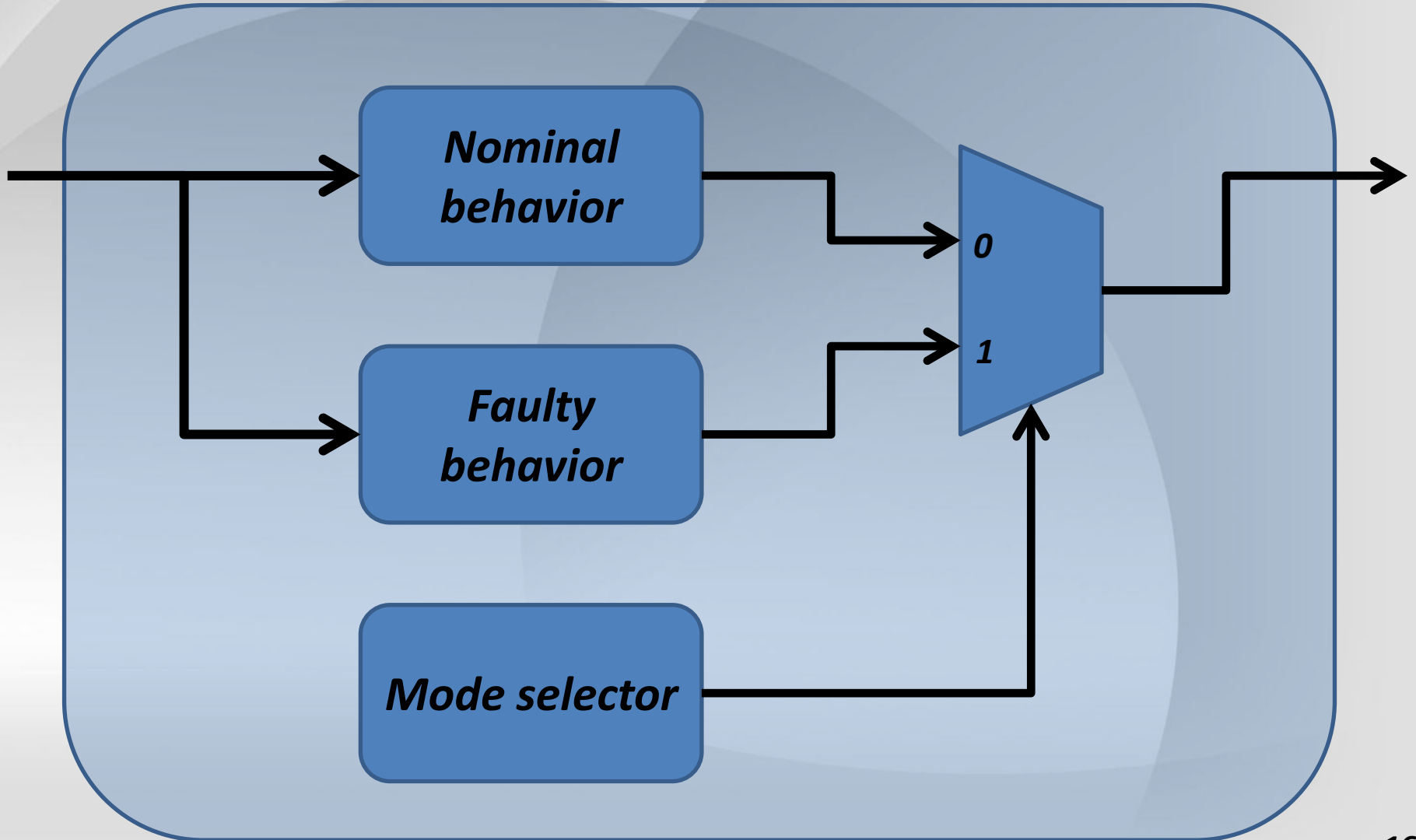
Fault Injection



Fault Injection



Fault Injection



Fault Injection

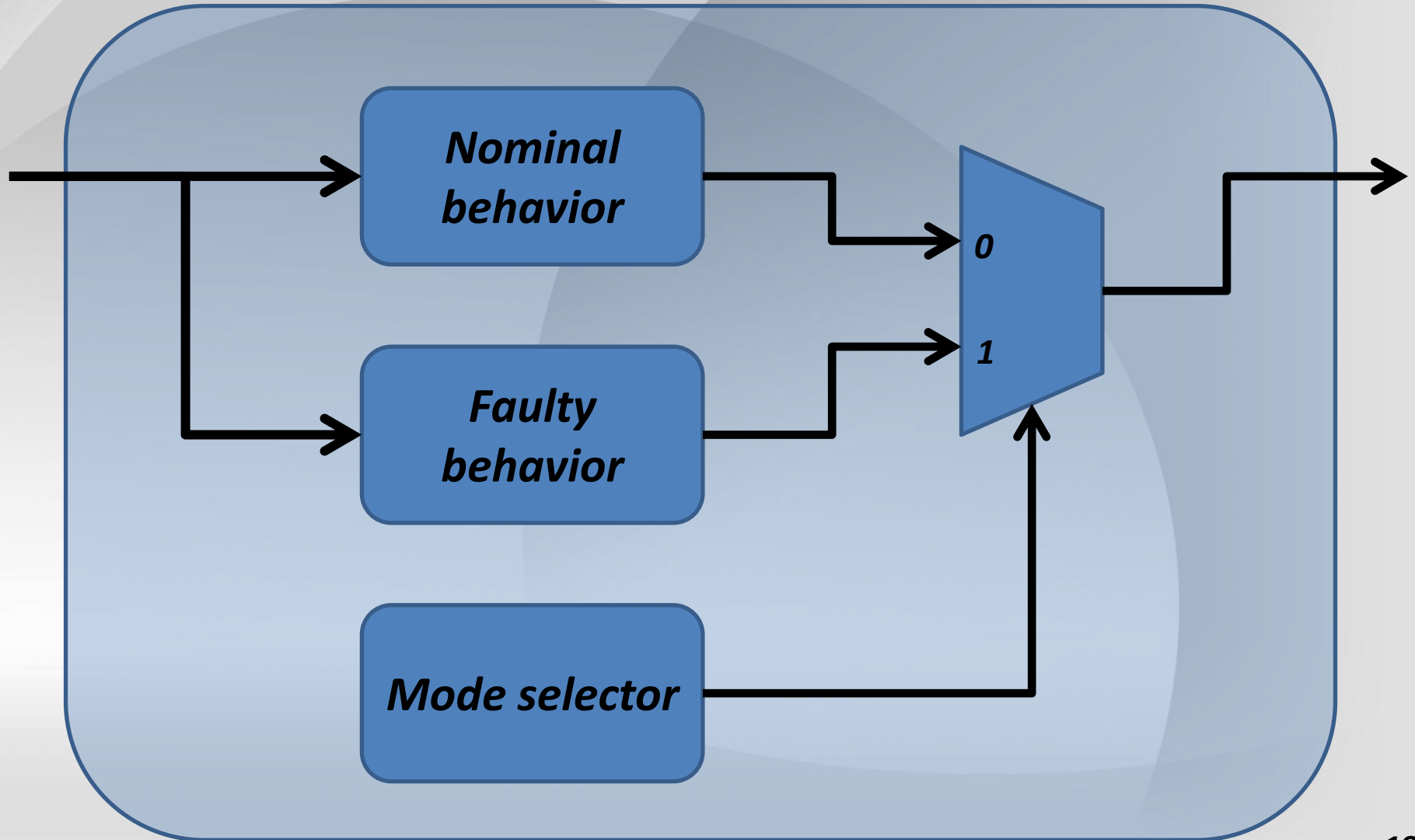
PROS

- ***Keeps nominal and fault model disjoint***
- ***Traceable process***
- ***Automatic technique***
- ***“Once and for all” validation***










CONS

- ***Needs functional modeling***

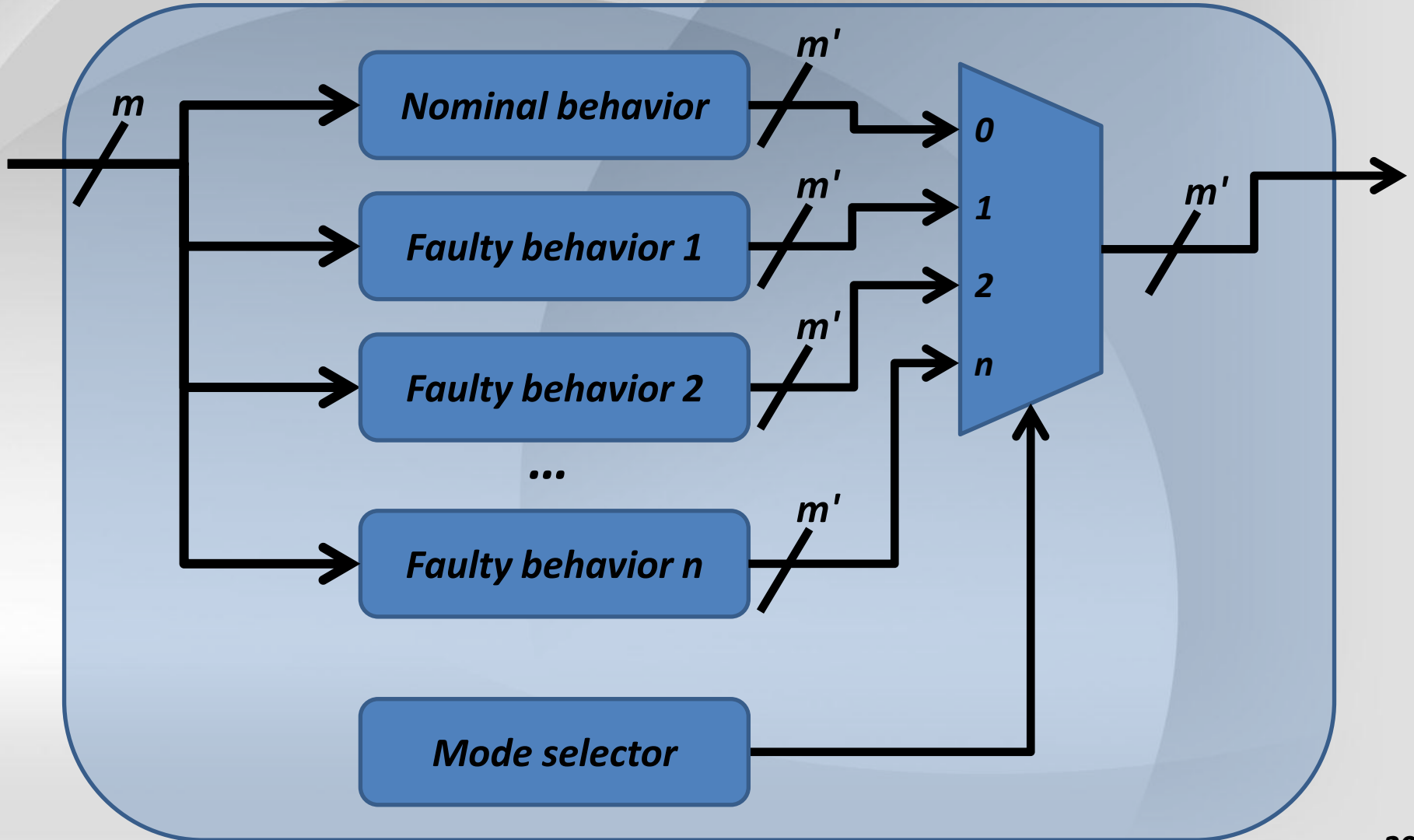
Fault Injection (FSAP)



Fault Extension approaches

	Expressiveness	Traceability	Time investment
Manual extension			
Fault Injection <i>FSAP</i>			
Library Based FI <i>NuSMV3</i>			

Library Based Fault Injection



Faults Libraries

- *Effects model library*

One effect model describes the effects on the associated nominal component when a fault occurs

e.g.: stuck at a value, invert a value, a value ramps down, ...

- *Local dynamics model library*

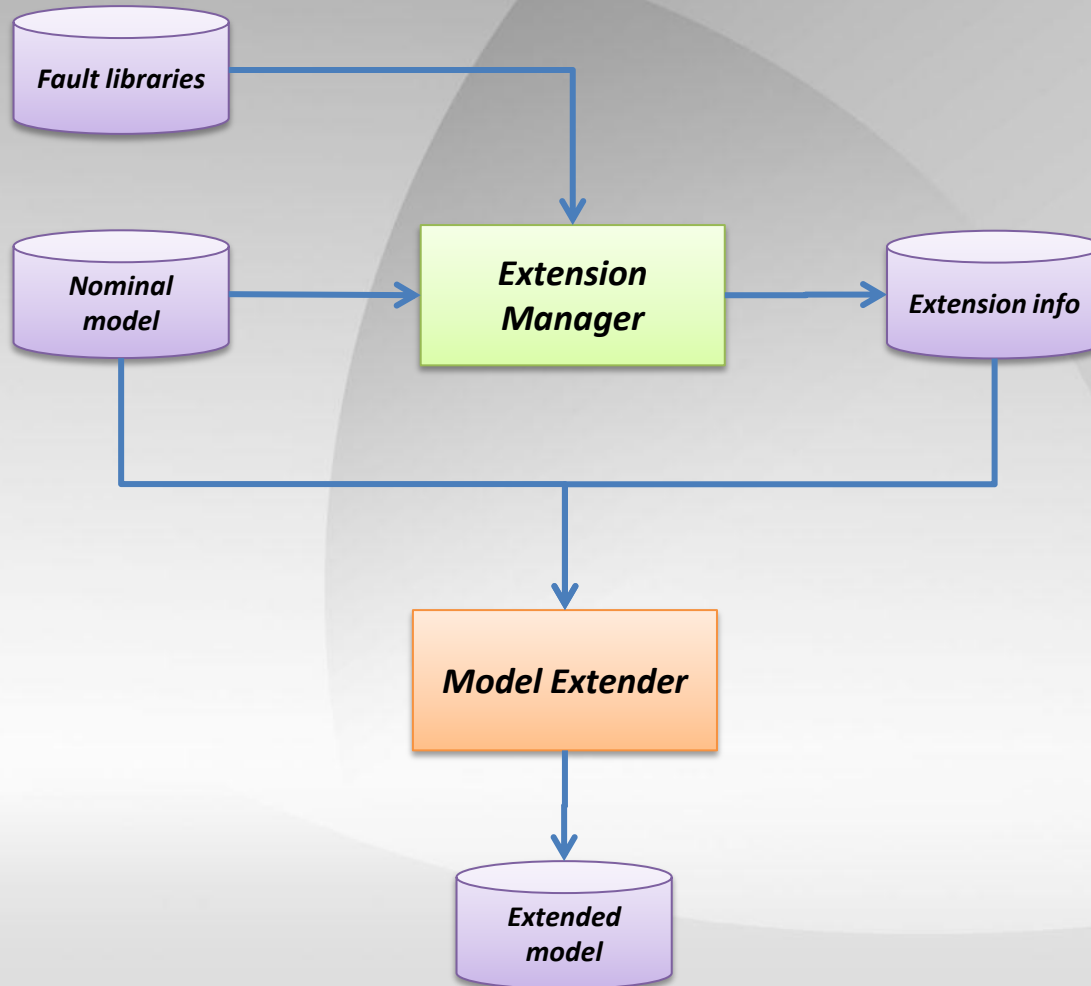
One local dynamic model describes the behavior of the fault

e.g.: a permanent or transient fault, self repair after 10 seconds, ...

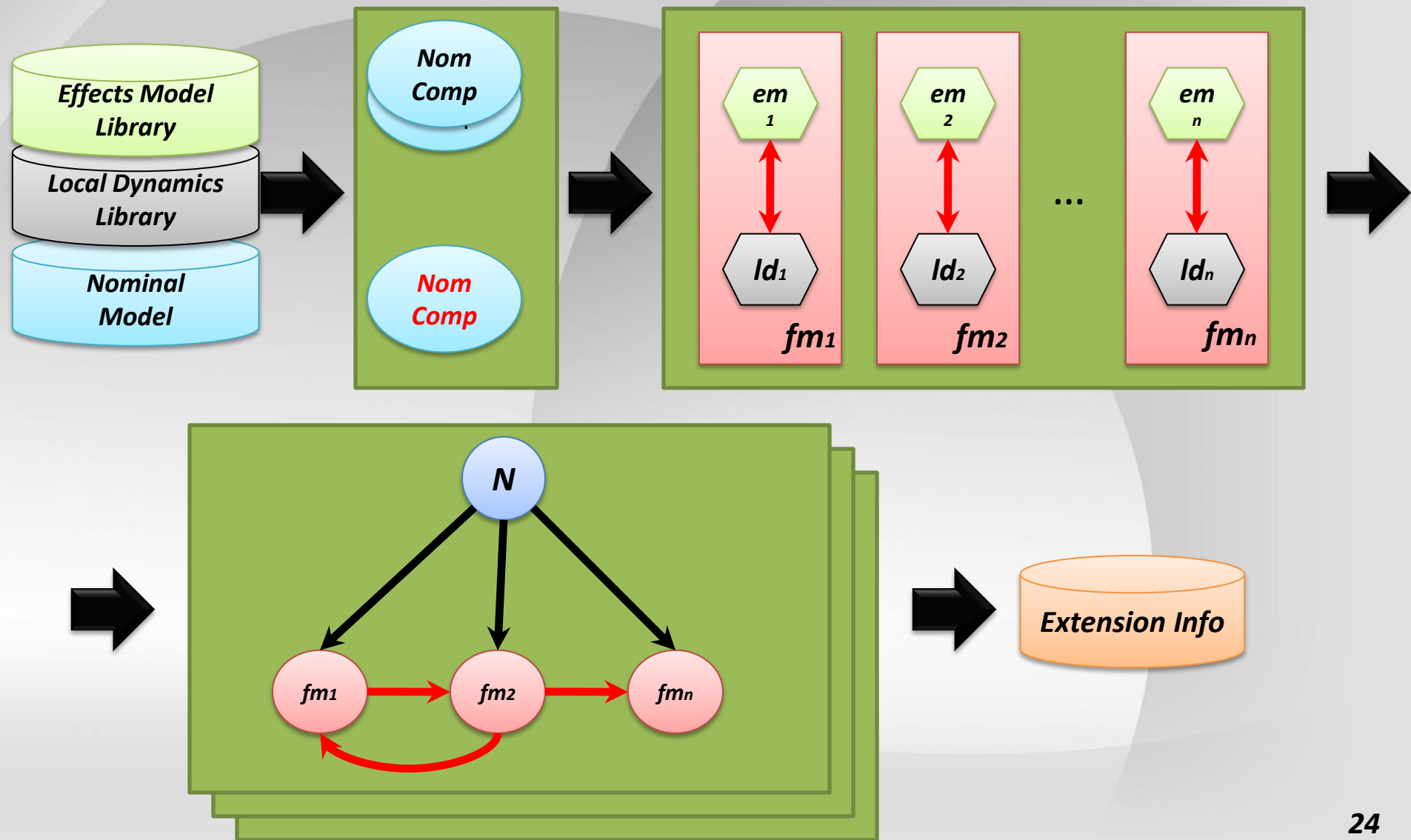
Library Based Fault Injection

- *Support for complex behavior*
 - *hybrid and discrete semantics*
 - *multiple input support*
 - *global dynamics interaction*
- *Easily extendable library definition*
 - *effects model and local dynamics*
- *User friendly and aided approach*
 - *human readable files definition*
 - *guided extension via GUI*

Flow of the Fault Extension



Flow of the Fault Extension

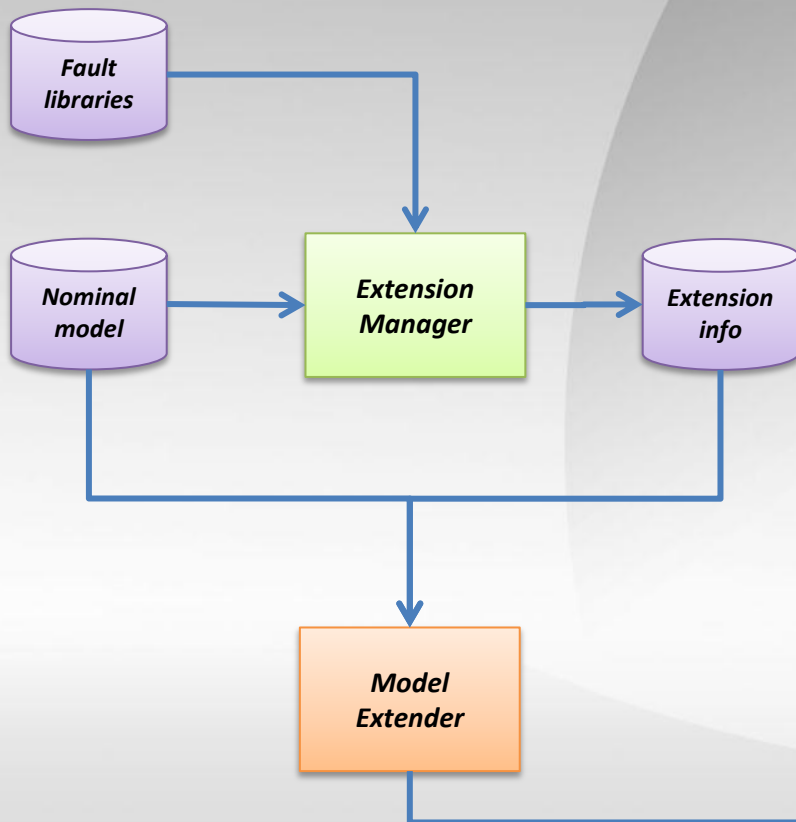


Roadmap

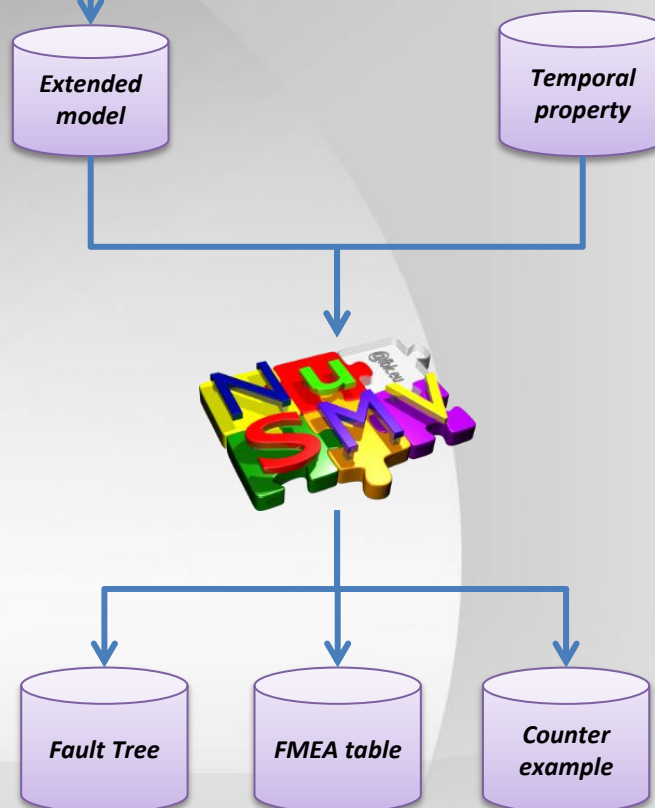
- *Formal Model Based Safety Assessment*
- *Formal Safety assessment*
 - *Current approach*
 - *Automated Fault Extension*
- ***NuSMV3 formal verification framework***
- *Next challenges*

Flow of Formal MBSA

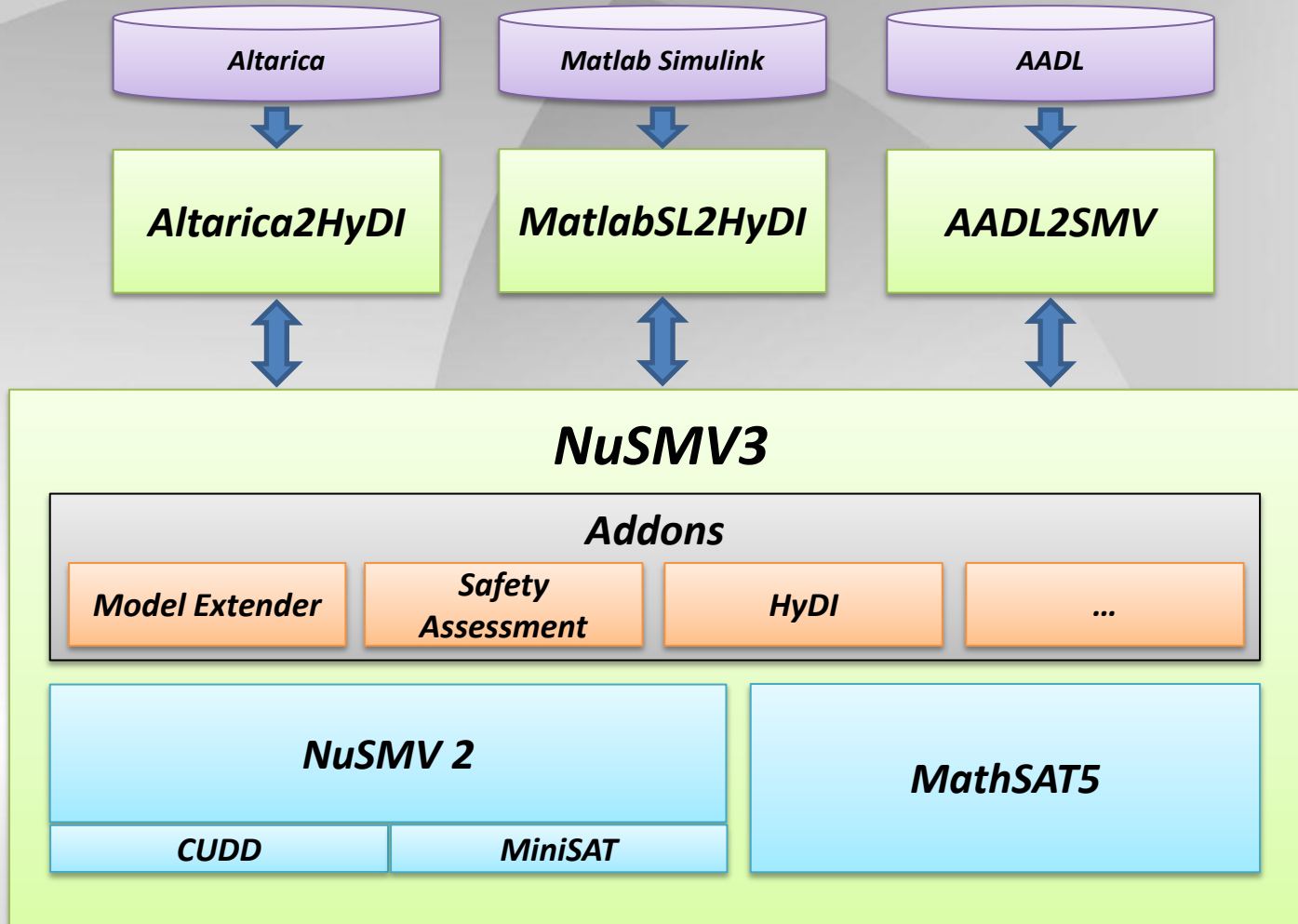
Fault Extension



Formal Verification

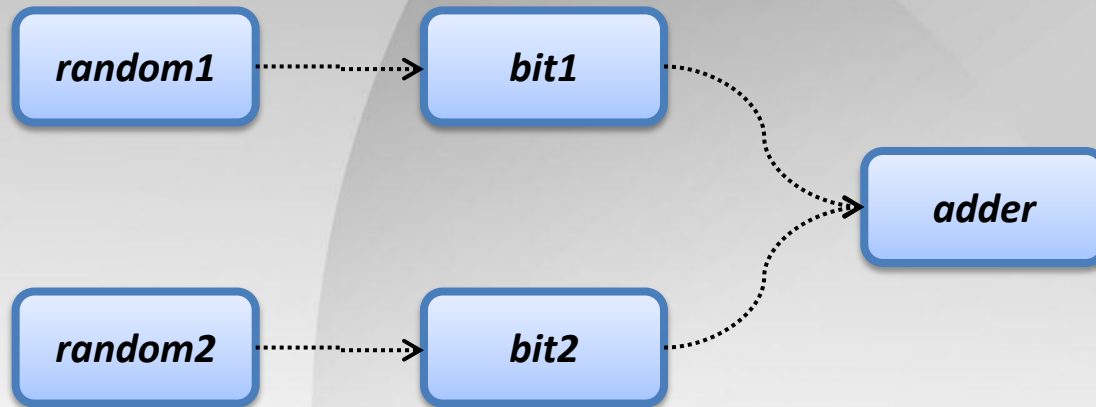


NuSMV3: Architecture

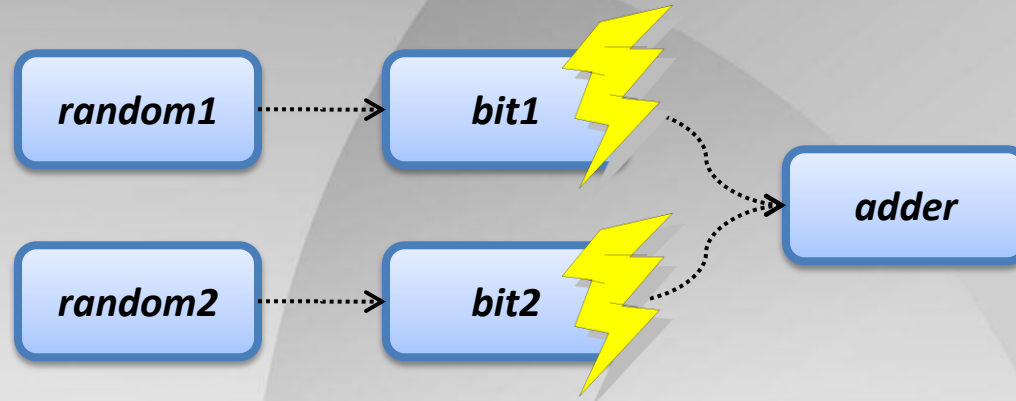


Adder Example

Adder example: Nominal Model

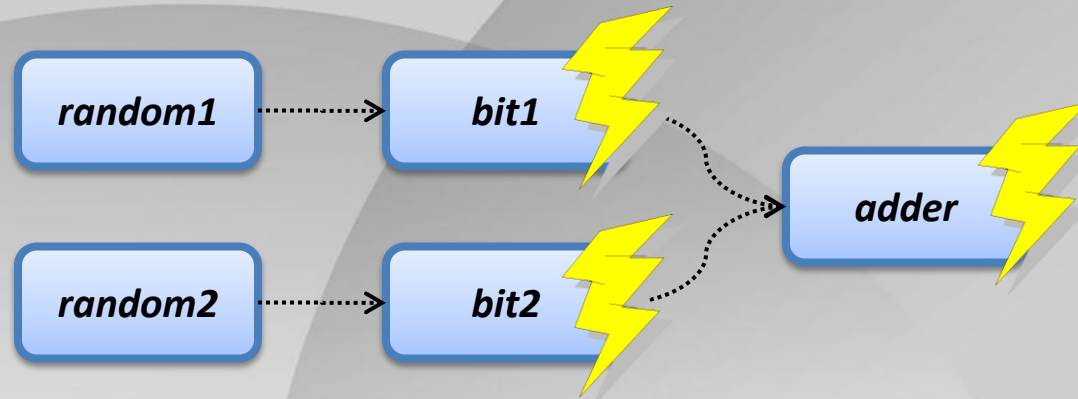


Adder example: components may fail



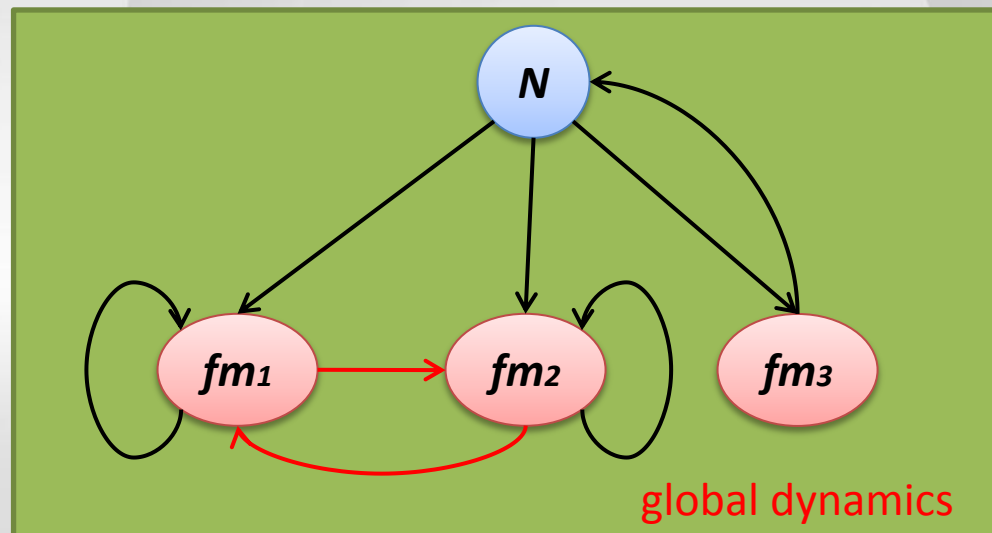
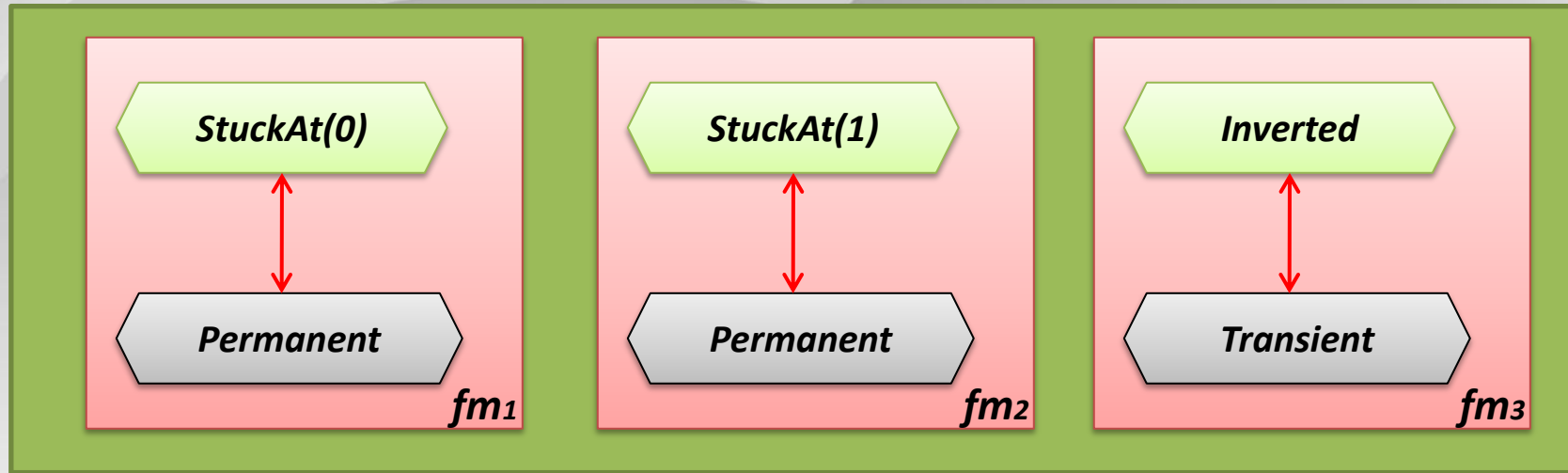
<i>Components</i>	<i>Effect Model</i>	<i>Local Dynamics</i>
<i>bit1, bit2</i>	<i>StuckAt(0)</i>	<i>Permanent</i>
<i>bit1, bit2</i>	<i>StuckAt(1)</i>	<i>Permanent</i>
<i>bit1, bit2</i>	<i>Inverted</i>	<i>Transient</i>

Adder example: components may fail



<i>Components</i>	<i>Effect Model</i>	<i>Local Dynamics</i>
<i>bit1, bit2</i>	<i>StuckAt(0)</i>	<i>Permanent</i>
<i>bit1, bit2</i>	<i>StuckAt(1)</i>	<i>Permanent</i>
<i>bit1, bit2</i>	<i>Inverted</i>	<i>Transient</i>
<i>adder</i>	<i>StuckAt(0)</i>	<i>Permanent</i>
<i>adder</i>	<i>StuckAt(1)</i>	<i>Permanent</i>

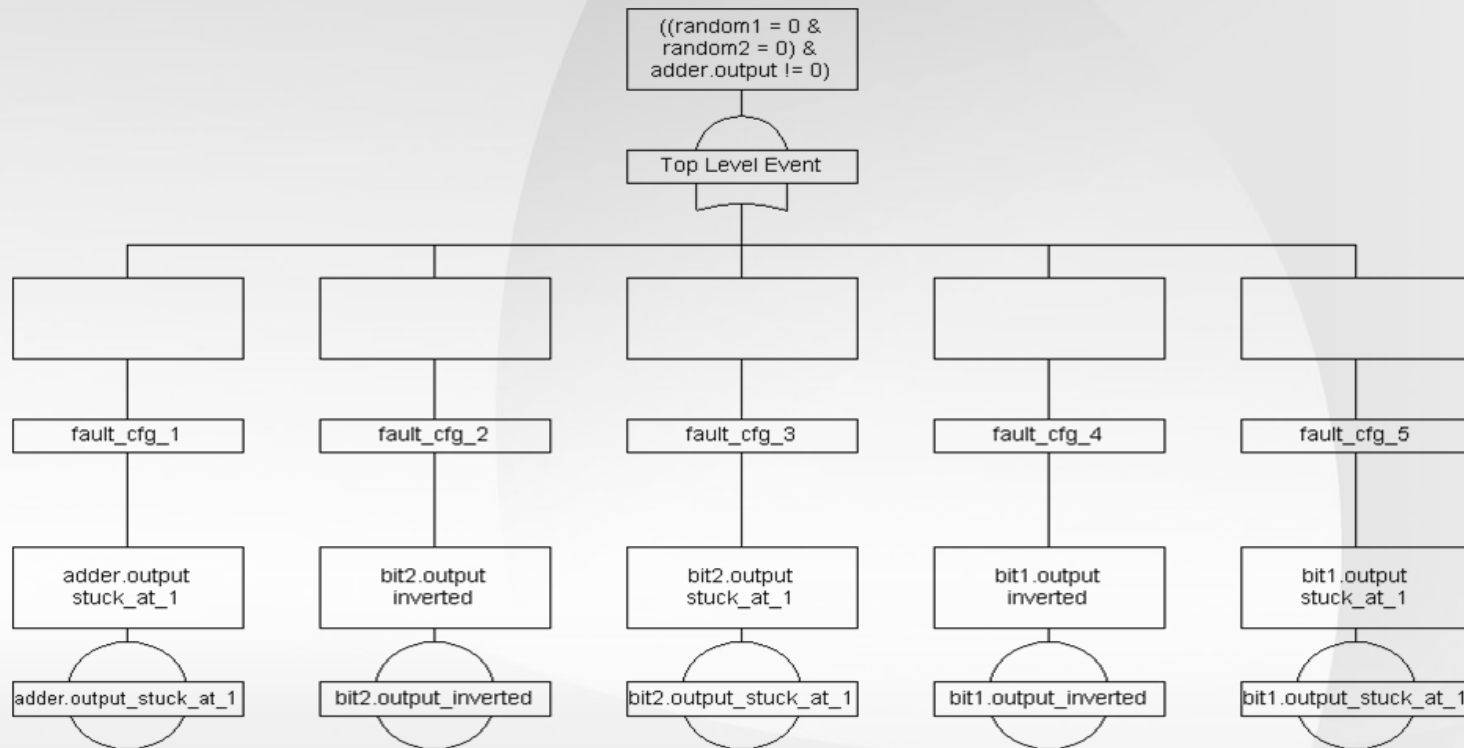
Example: bits fault model composition



Analysis Results

Example: Fault Tree Analysis

Top Level Event:
 $random1 = 0 \ \& \ random2 = 0 \ \& \ adder.output \neq 0$
(check when $0 + 0 \neq 0$)



Example: FMEA tables

FMEA TABLE ORDER 1		
Id.Nr.	Failure Mode	Failure Effects
1	<i>bit1.output inverted</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
2	<i>bit1.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
3	<i>bit2.output inverted</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
4	<i>bit2.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
5	<i>adder.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>

FMEA TABLE ORDER 2		
Id.Nr.	Failure Mode	Failure Effects
1	<i>bit1.output inverted</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
2	<i>bit1.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
3	<i>bit2.output inverted</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
4	<i>bit2.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
5	<i>adder.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
6	<i>bit1.output inverted & bit1.output stuck_at_0</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
7	<i>bit1.output inverted & bit1.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
8	<i>bit1.output inverted & bit2.output inverted</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
9	<i>bit1.output inverted & bit2.output stuck_at_0</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
10	<i>bit1.output inverted & bit2.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
11	<i>bit1.output inverted & adder.output stuck_at_1</i>	<i>((random1=0 & random2 = 0) & adder.output !=0)</i>
12	...	

Conclusion

Library based fault extension

- ***Highly Expressive***
- ***Automated technique***
- ***Time saving***
- ***Traceable process***

Next challenges

- ***Extension of expressiveness for library based fault injection***
- ***Integration with industrial design tools***

Thank you!

Cristian Mattarei

FBK ES-Group

mattarei@fbk.eu