25th Brazilian Symposium on Formal Methods (SBMF 2022)

Some Applications of Formal Methods

07 December 2022

Valdivino Alexandre de Santiago Júnior

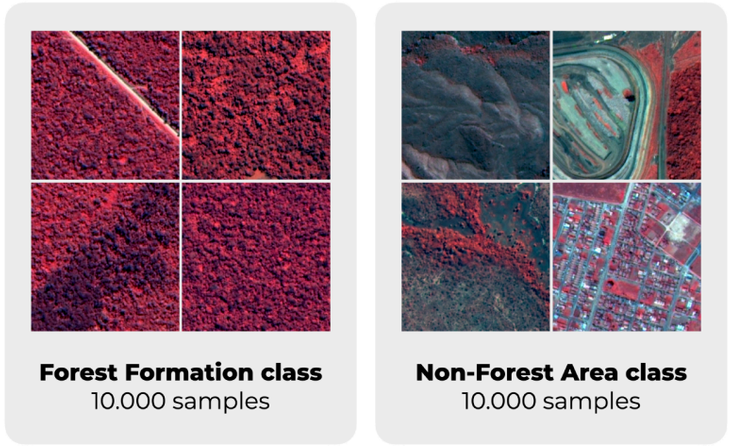
Coordenação de Pesquisa Aplicada e Desenvolvimento Tecnológico (COPDT)
Instituto Nacional de Pesquisas Espaciais (INPE)
São José dos Campos, SP, Brazil

About INPE

Headquarters:
São José dos Campos/SP, Brazil.

About INPE

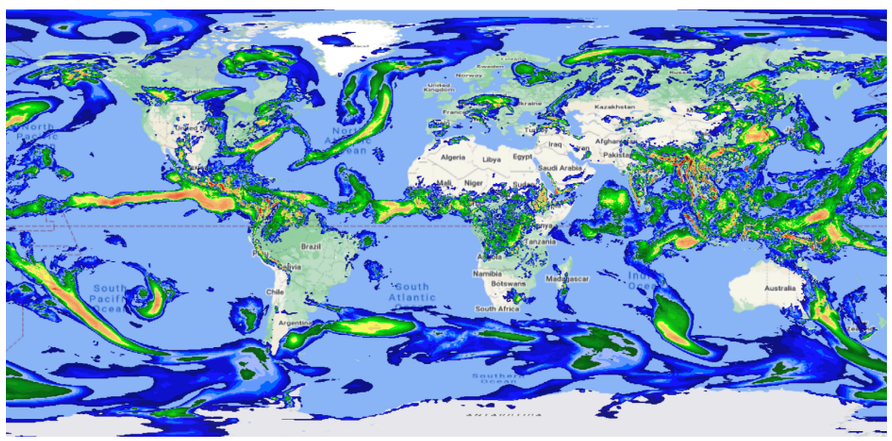
- Science and technology for the outer space and the Earth system, for the benefit of the Brazil and the world.



Forest Formation class
10.000 samples

Non-Forest Area class
10.000 samples

Remote Sensing



Weather and Climate

Space Engineering

About INPE

- Seven post-graduate programs.

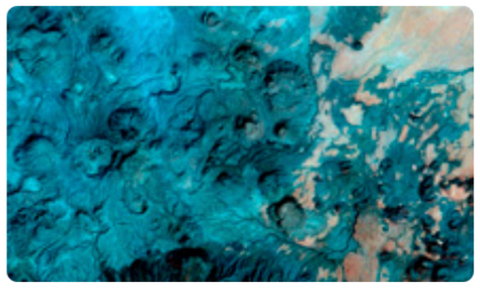
Astrophysics

Space Engineering and Technology

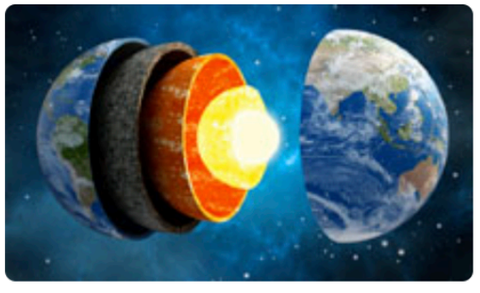
Space Geophysics

Applied Computing

Meteorology



Remote sensing



Earth System Science

About INPE

- Seven post-graduate programs.

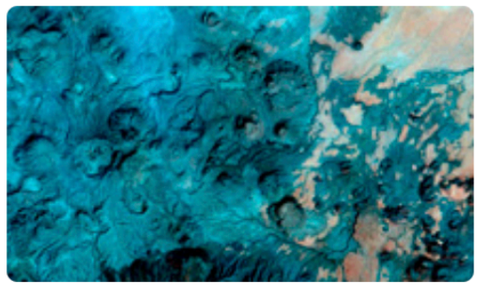
Astrophysics

Space Engineering and Technology

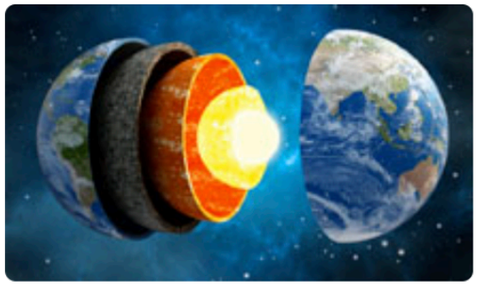
Space Geophysics

Applied Computing

Meteorology



Remote sensing



Earth System Science

In This Talk…

- Some applications of formal methods (FM).

Source: https://nasa.tumblr.com/post/189210016829/from-discovering-the-secrets-of-the-universe-to

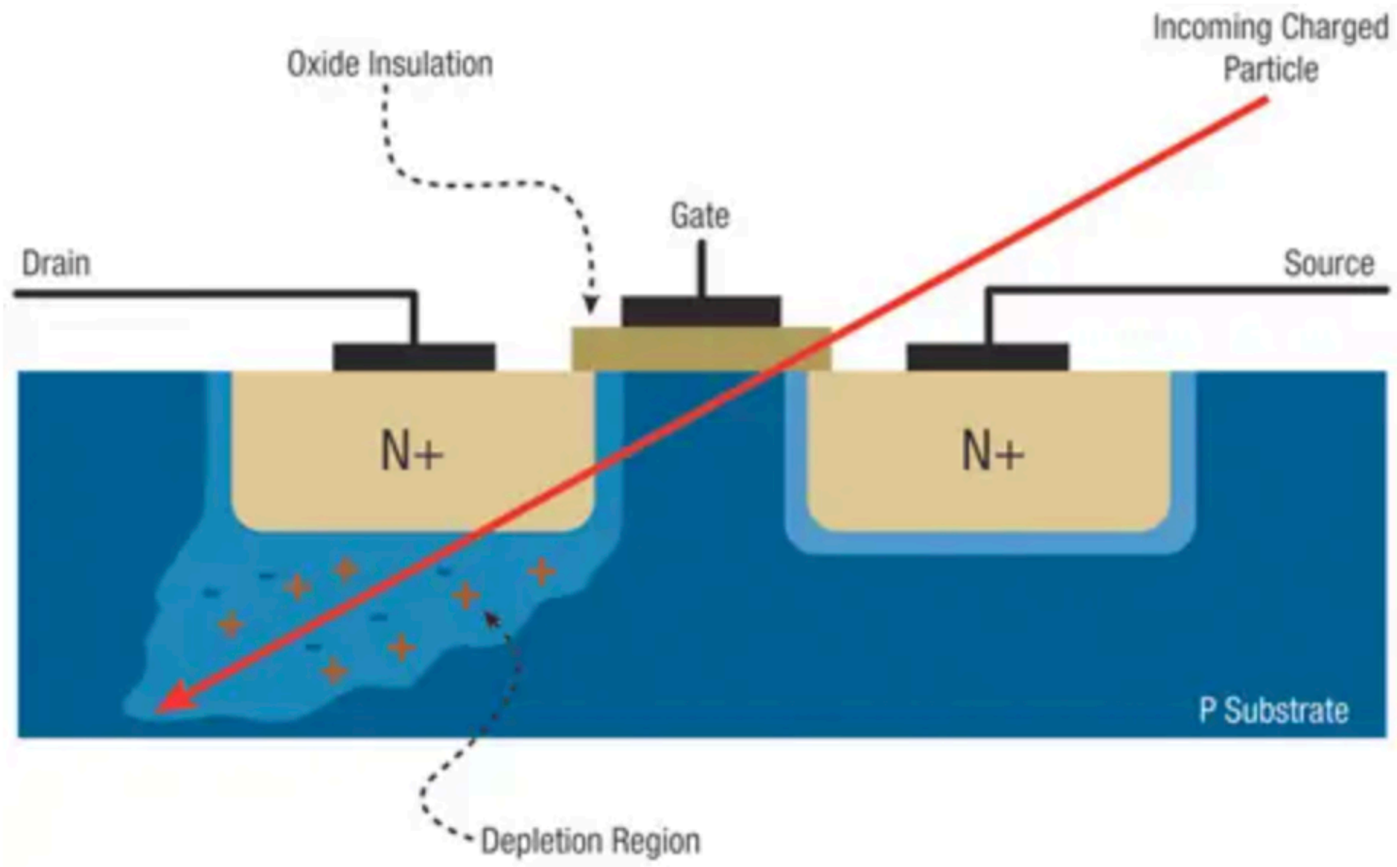
Application 1

Mitigation problem of single event upsets (SEUs) in field-programmable gate arrays (FPGAs)

Single Event Effect (SEE)

- “SEEs occur when atmospheric radiation, comprising high energy particles, collide with specific locations on semiconductor devices contained in aircraft systems. Memory devices, microprocessors and FPGAs are most sensitive to SEE”. [FAA 2016]

SEE



Thus, SEEs affect AEROSPACE systems!

SEUs in SRAM FPGAs

- SEU is a type of SEE.

- An SEU causes a change of state in a storage cell (memory devices, latches, sequential logic, …). **Bit-flip!**

- Hence, SRAM FPGAs are susceptible to SEUs.

SEUs in SRAM FPGAs

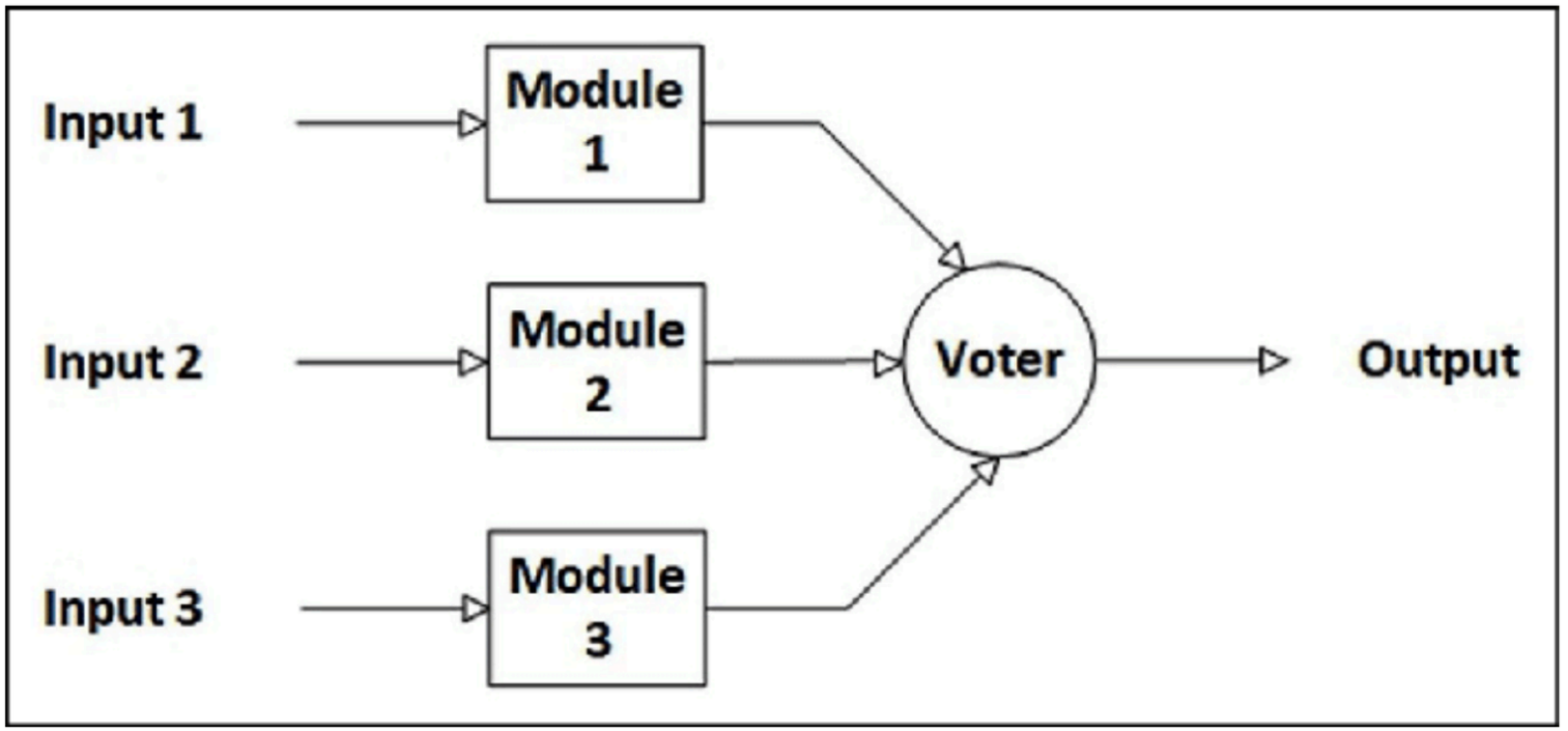
- Traditional approaches: usually costly (design and implementation of techniques in FPGAs, ...).

- Thus, one interesting path:
  - Obtaining the results in the initial stage of the project;
  - No risk to damage the devices (FPGAs);
  - Model-driven development (MDD).

Objective

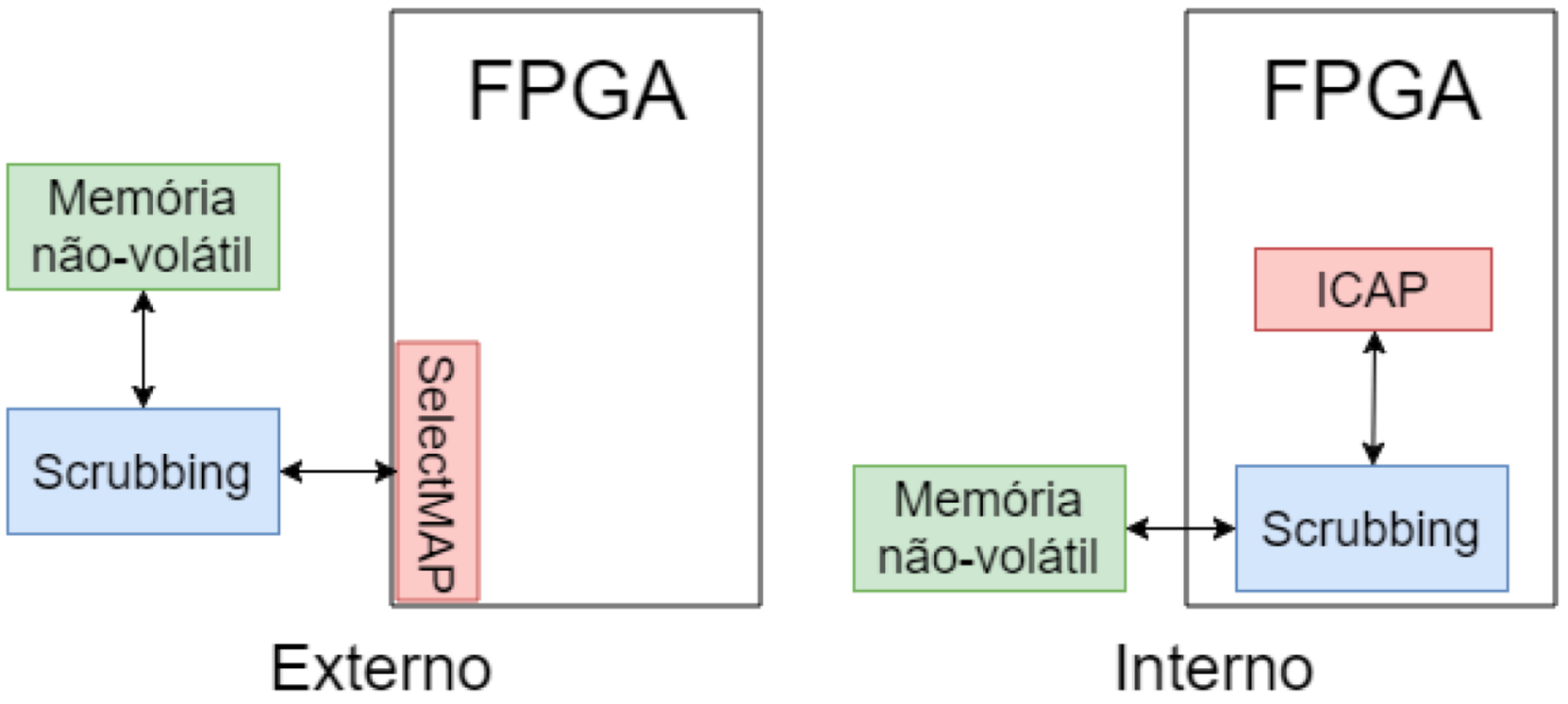
- To investigate the feasibility, in the context of space applications, of **probabilistic model checking** to determine what would be, among a set of solutions, the best technique for mitigating SEU on SRAM FPGAs. [Pereira, Santiago Júnior and Manea 2017][Pereira, 2018].

Triple Modular Redundancy (TMR)



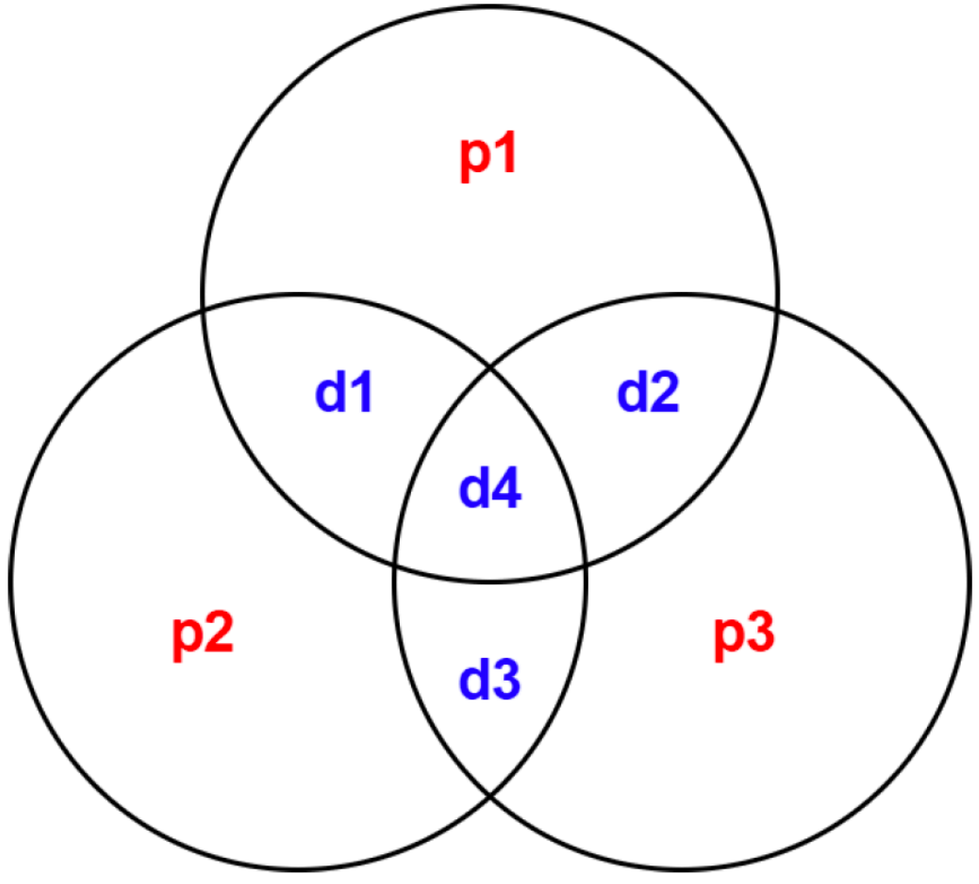
Scrubbing

- Rewrites part of the memory. Correct the fault rather than masking it.
  - Blind: ΔT defines when original data are copied.



Hamming Code

- Error-correction code. Detects and corrects faults.
  - Makes use of redundant bits.



Determining the SEU rates

- Mean Time Between Failures (MTBF).

- Cosmic Ray Effects on Microelectronics (CREME)

- CREME96: phenomenological models to predict SEE rates.

Determining the SEU rates

- CREME96: add info regarding the orbit of the satellite, type of radiation, …

- Example: CBERS-4A.
  - Mean orbital altitude: 628.614 km.
  - Inclination: 97.8963 degrees.
  - Apogee, perigee, …

Determining the SEU rates

- Example: CBERS-4A (heavy ions; SEEs/bit/second).

```
REPORT NO.    1:   virtex5 T O D O
RPP Dimensions: X =    1.82000  Y =    1.82000  Z =    1.00000 microns.
Funnel length =    0.00000 microns.
CROSS-SECTION INPUT   4  WEIBULL FIT:
    ONSET   =     0.100 MeV-cm2/milligram
    WIDTH   =    10.860 MeV-cm2/milligram
    POWER   =     1.750 (dimensionless)
    PLATEAU =     3.330 square microns/bit
Number of bits =   1.00000E+00
Rates:      SEEs/bit/second     /bit/day    /device/second    /device/day
*****   1     1.28824E-12     1.11304E-07     1.28824E-12     1.11304E-07

REPORT NO.    2:   virtex5 T O D O
RPP Dimensions: X =    1.00000  Y =    1.00000  Z =    1.00000 microns.
Funnel length =    0.00000 microns.
CROSS-SECTION INPUT   4  WEIBULL FIT:
    ONSET   =     0.250 MeV-cm2/milligram
    WIDTH   =   100.000 MeV-cm2/milligram
    POWER   =     2.950 (dimensionless)
    PLATEAU =     1.000 square microns/bit
Number of bits =   1.00000E+00
Rates:      SEEs/bit/second     /bit/day    /device/second    /device/day
*****   2     1.53736E-16     1.32828E-11     1.53736E-16     1.32828E-11
```

Probabilistic Model Checking

- Continuous-Time Markov Chain (CTMC).

- Continuous Stochastic Logic (CSL).

- PRISM model checker.

Case Studies

- Both cases: FPGA Xilinx Virtex-5.

- First case study: High Elliptical Orbit [Hoque 2016].

- Second case study: CBERS-4A.

Developing the CTMC Models

- CREME96 (SEEs/bit/second).

- Fault (Event) rate: $\lambda = 1/MTBF$.

Developing the CTMC Models

- Scrubbing:
  - Based on [Hoque 2016].
  - But 10 Adders (A) e 10 Multipliers (M). Higher!
  - $\Delta T = 1$, 4, and 9 days.

Developing the CTMC Models

- TMR:
  - 2 × 4-bit inputs.
  - Individual voting modules.
- Hamming code:
  - 4 bits of data and 3 bits of parity.
  - Encoding and decoding modules.

Developing the CTMC Models

- TMR in PRISM.

```
module adder1
    a1 : [0..15] init 0;
    a2 : [0..15] init 0;
    outA1 : [0..31] init 31;
    [input] (a1=0 & a2=0) -> 1: (a1'=5) & (a2'=7);
    [adder] ((a1 > 0 | a2 > 0) & (Na > 0)) -> Na*(1-lambda_A) :
    (outA1'= a1+a2) + Na*lambda_A : (outA1'= a1+a2-1);
endmodule
(...)

formula fail = ((outMf!=m1*m2) | (outAf!=a1+a2));
(...)

rewards "timeFailure"
        fail : 1;
endrewards
(...)
```

Kogge-Stone adder in VHDL

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.Carry_Tree_Adder_pkg.all;

entity Kogge_Stone_Adder is
    Port  (A : in std_logic_vector(3 downto 0);
           B : in std_logic_vector(3 downto 0);
           C_in : in STD_LOGIC_VECTOR(0 downto 0) := "0";
           C_out : out STD_LOGIC;
           Sum_out : out STD_LOGIC_VECTOR(3 downto 0));
end Kogge_Stone_Adder;

architecture Behavioral of Kogge_Stone_Adder is
-- Declaring necessary variables
    -- Output variables for each stage
    -- Stage 0
    signal g_0 : std_logic_vector(3 downto 0);
    signal p_0 : std_logic_vector(3 downto 0);
```

```vhdl
    -- Stage 3 Operations
    Storing_3 :
       for i in 0 to 3 generate
           g3(i) <= g2(i);
           p3(i) <= p2(i);
       end generate;

    carry_gen :
       for i in 0 to 3 generate
           car(i) <= g3(i) OR (C_in(0) AND p3(i));
       end generate;

    C_out <= car(3);
    Sum_out(0) <= C_in(0) XOR p_0(0);

    -- Calculating final sum
    Sum :
       for i in 1 to 3 generate
           Sum_out(i) <= car(i-1) XOR p_0(i);
       end generate;

end Behavioral;
```

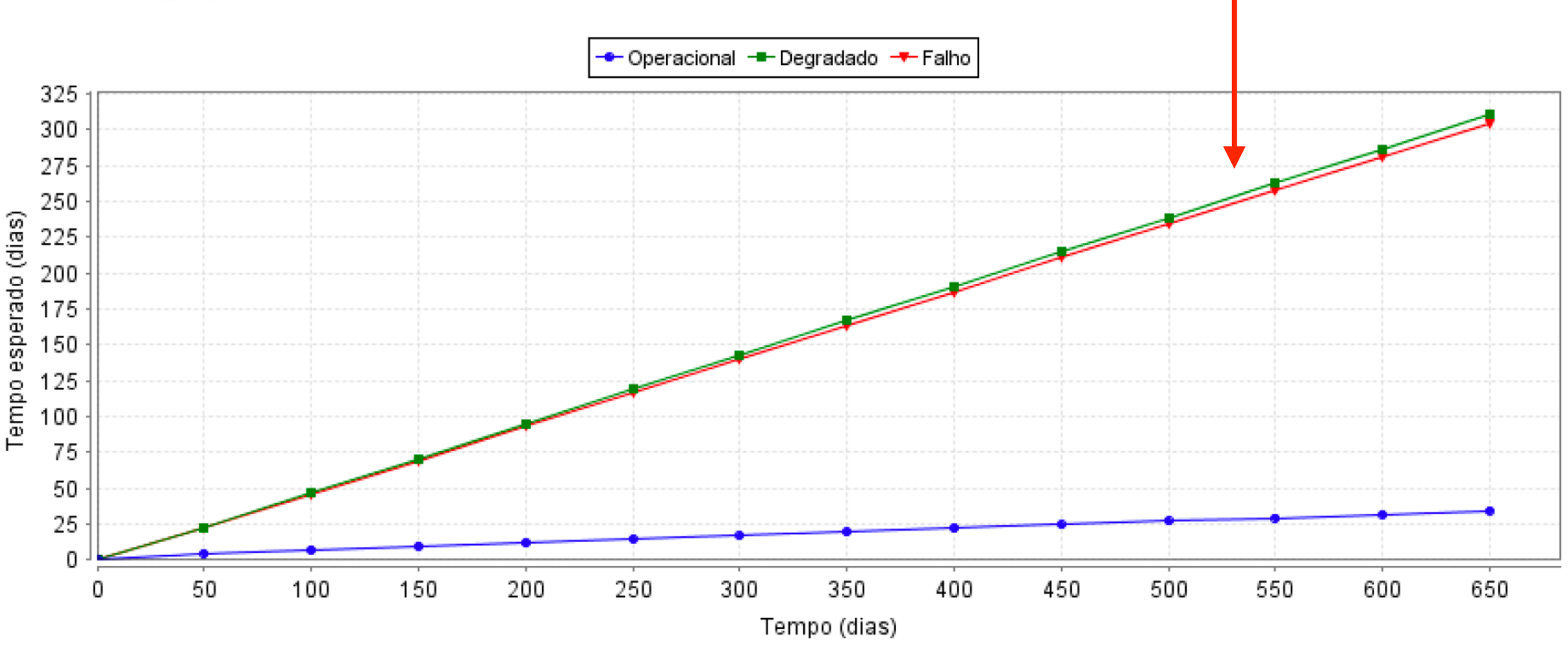
Source: https://github.com/NatsuDrag9/Kogge-Stone-Adder/blob/master/Carry_Tree_Adder.vhd

CSL Extended With Rewards

Attribute	Property	Meaning
Availability	$\mathcal{R}\{''timeOperational''\} =?[C <= t]$	Accumulated time in the operational mode within interval $[0, t]$
Availability	$\mathcal{R}\{''timeDegraded''\} =?[C <= t]$	Accumulated time in the degraded mode within interval $[0, t]$
Availability	$\mathcal{R}\{''timeFailure''\} =?[C <= t]$	Accumulated time in the failure mode within interval $[0, t]$
Availability	$\mathcal{S} =?[fail]$	The steady-state probability of the failure of the system
Relilability	$\mathcal{P} =?[\Box[0, t]oper \vee degrade]$	The probability that the system is operational or degraded in the first t days
Safety	$\mathcal{P} =?[\Box[0, t]oper \vee degrade \vee fail_{safe}]$	The probability that the system is operational, degraded, or in fail safe in the first t days

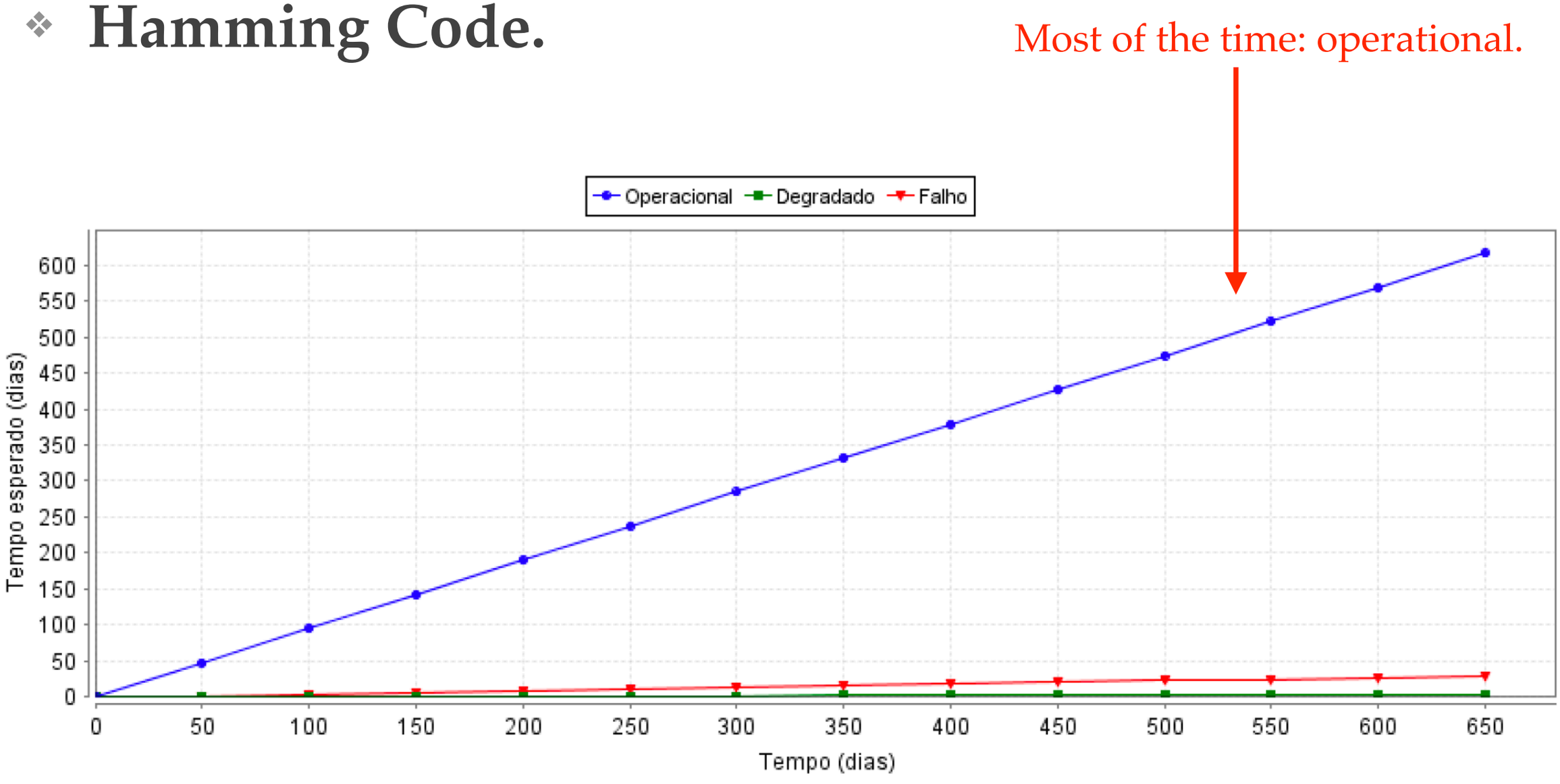
Results: Availability

- CBERS-4A (PUP=Indirect Ionisation). Mission time: 650 days.
- **TMR.**



Results: Availability

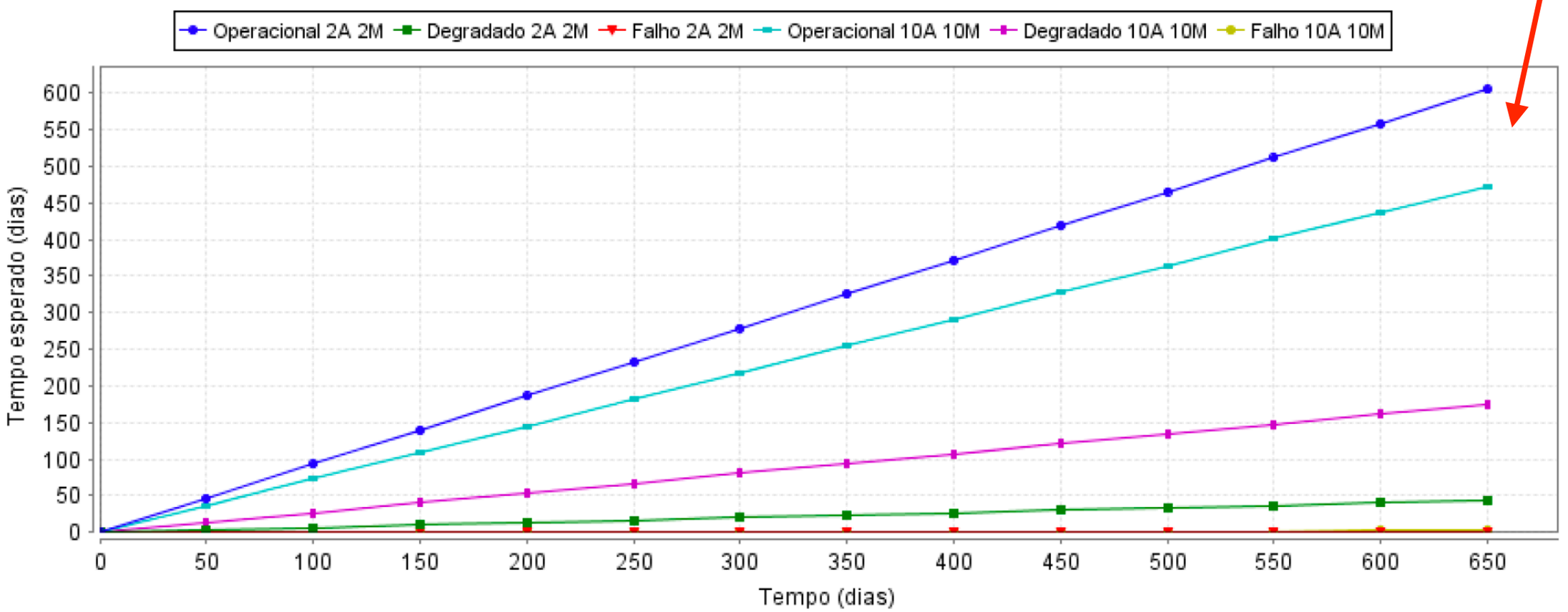
- CBERS-4A (PUP). Mission time: 650 days.
- **Hamming Code.**



Results: Availability

- CBERS-4A (PUP). Mission time: 650 days.
- 2A 2M [Hoque 2016]; 10A 10M (ours).
- **Scrubbing. $\Delta T = 1$ day.**

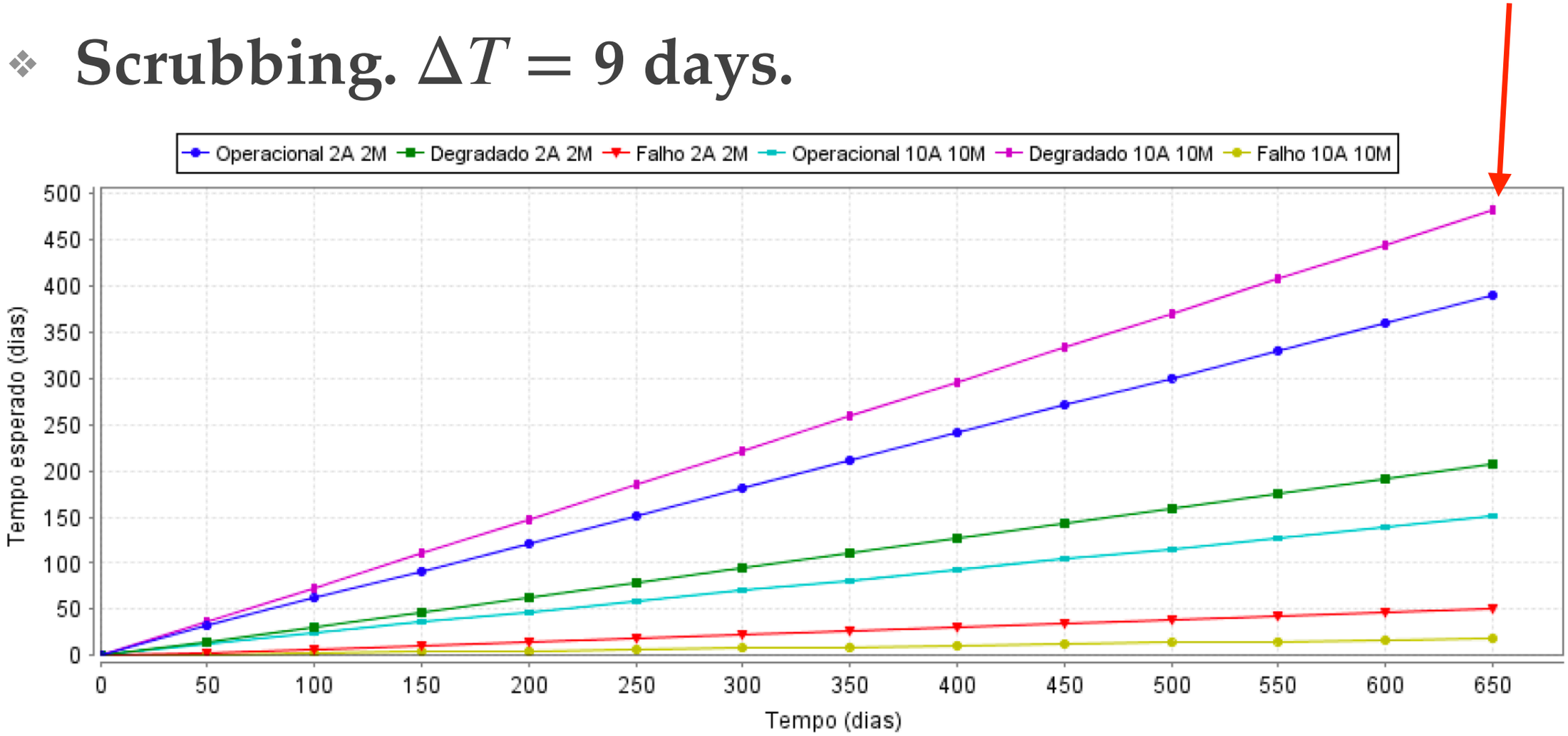
2A 2M and 10A 10M: operational (good).



Results: Availability

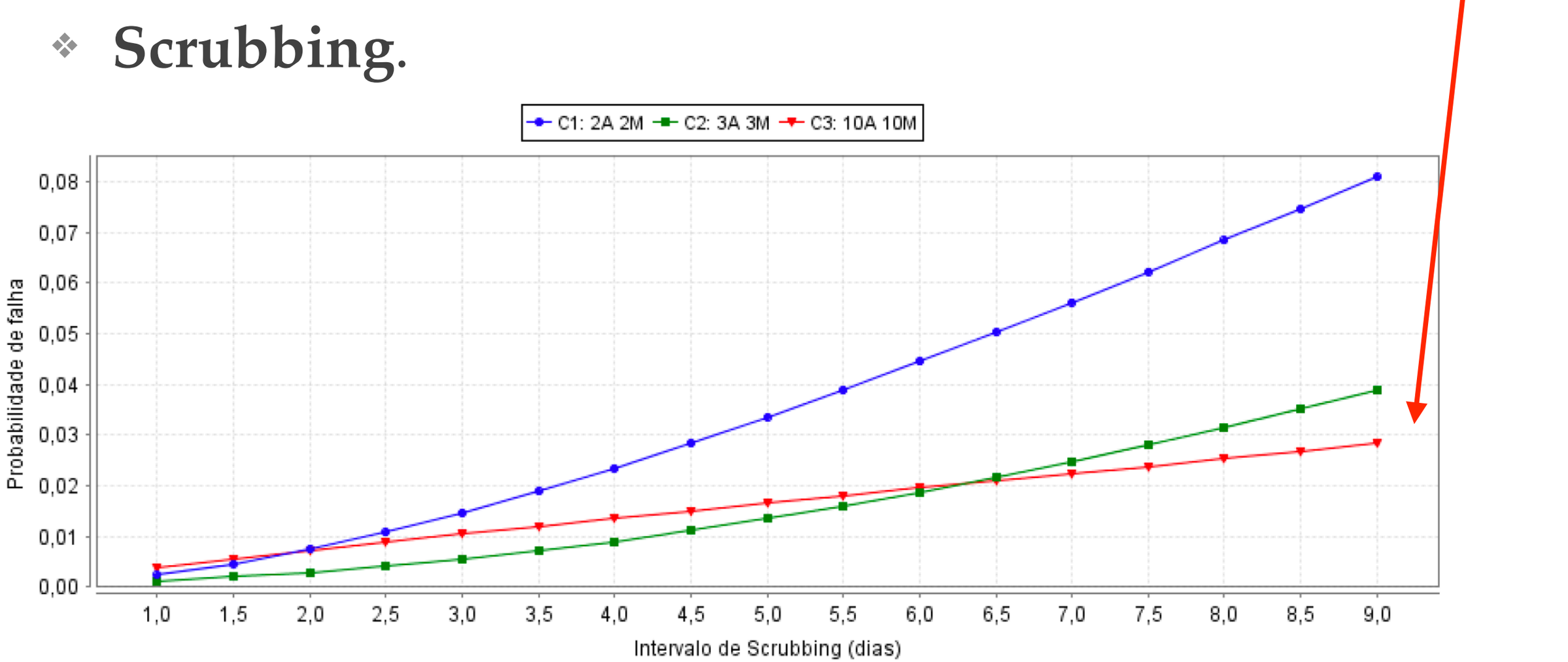
- CBERS-4A (PUP). Mission time: 650 days.
- 2A 2M [Hoque 2016]; 10A 10M (ours).
- **Scrubbing. $\Delta T = 9$ days.**

2A 2M still good.
But, in 10A 10M, most of the time degraded.



Results: Availability

- CBERS-4A (PUP). Steady-state probability:
  - TMR: 0.4701; Hamming Code: 0.0046;
  - **Scrubbing.**



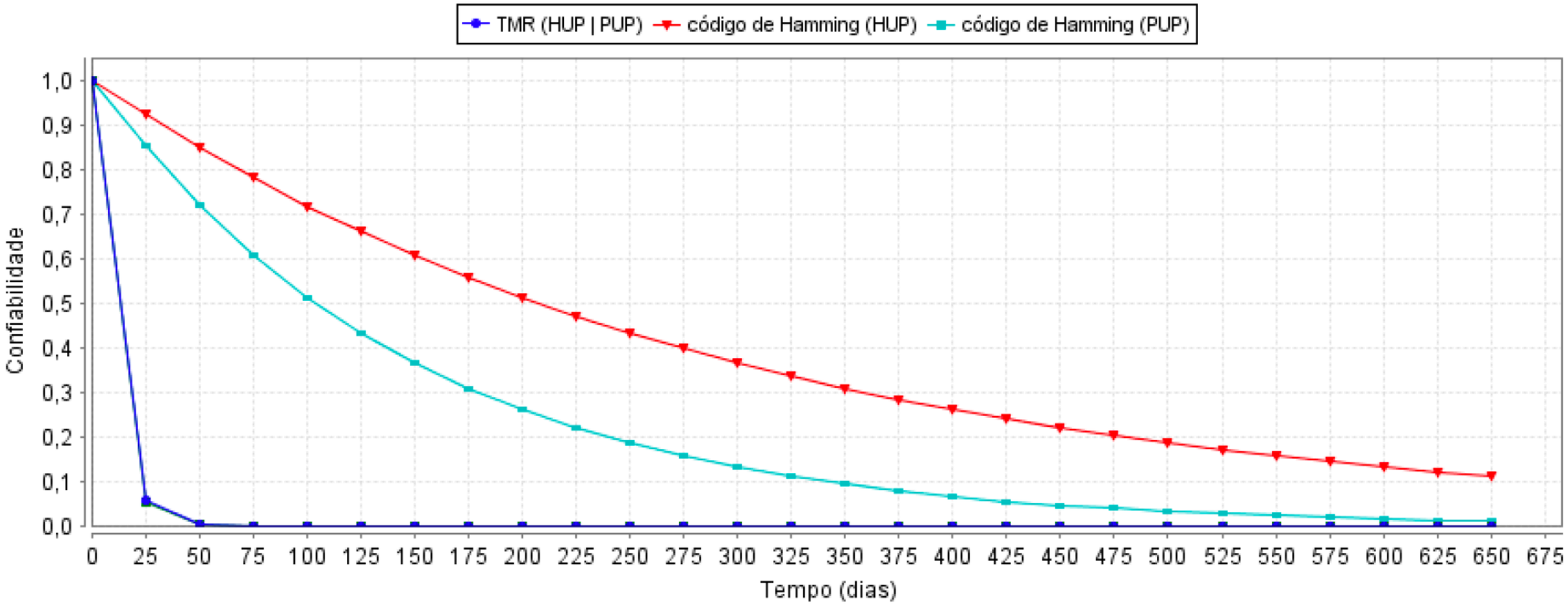
Conclusions: Availability

- Probabilistic model checking: consistent results considering all evaluated techniques and systems.
- TMR: most of the mission time in the failure mode.
- Hamming code: high availability.
- For scrubbing: highest ΔT affected more the smaller systems.
  - For larger systems, it might be ok a highest ΔT.

Results: Reliability

- CBERS-4A (PUP, HUP = Direct Ionisation). Mission time: 650 days.
- **TMR** and **Hamming Code**.

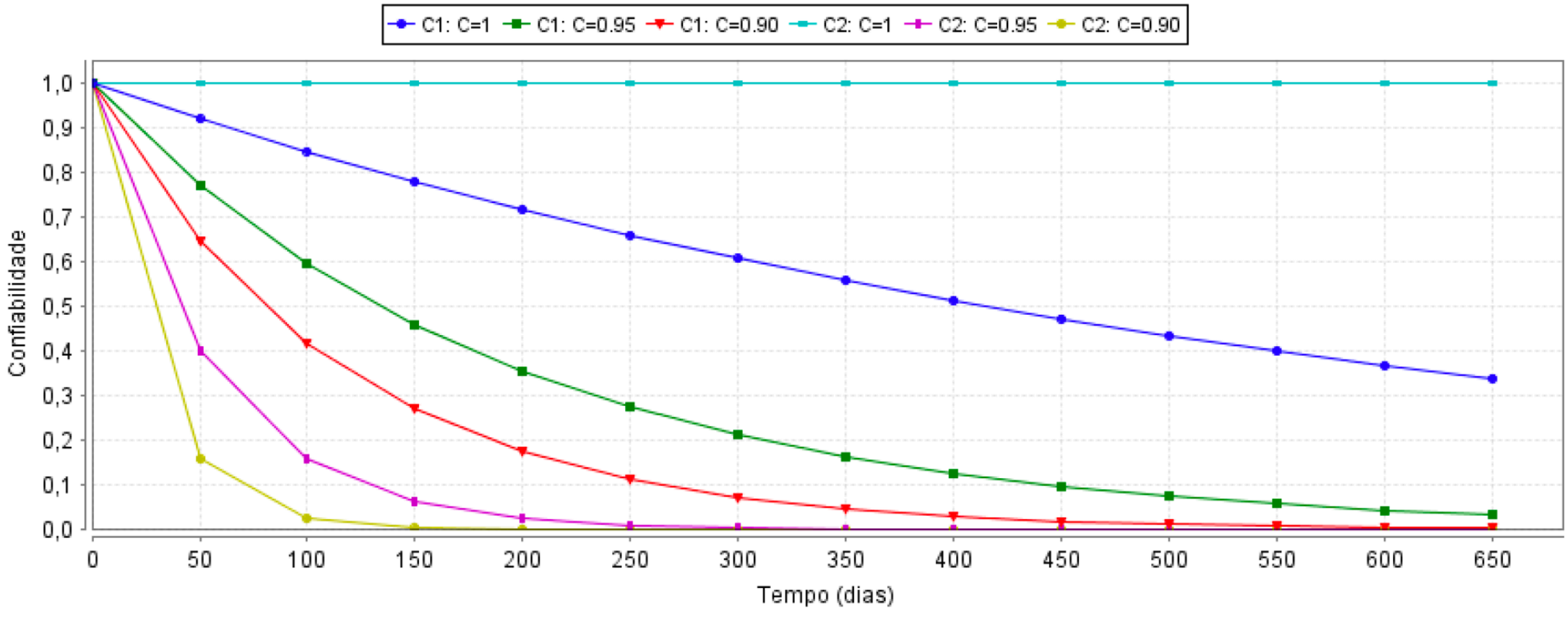
TMR: really bad!.



Results: Reliability

- CBERS-4A (PUP). Mission time: 650 days.
- C1: 2A 2M [Hoque, 2016]; C2: 10A 10M (ours).
- **Scrubbing. $\Delta T = 1$ day.**

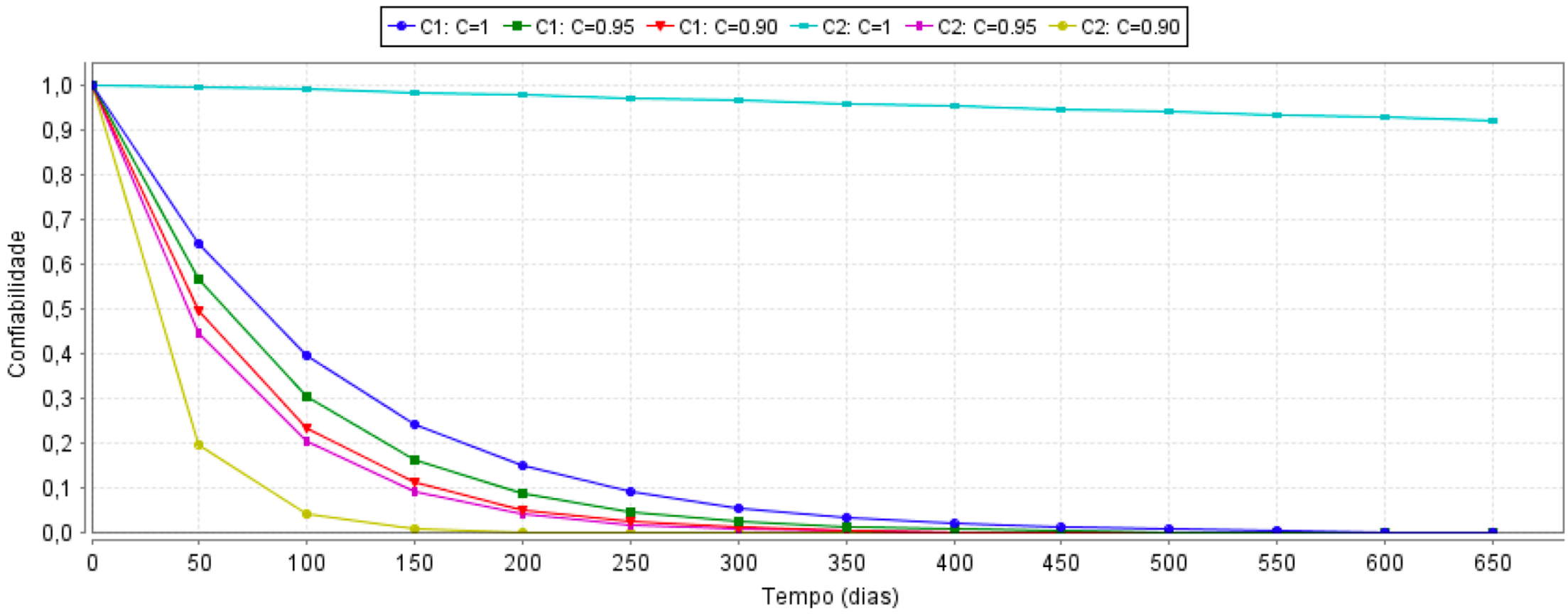
C = Cover Rate = $P(detect\_fault \mid fault\_exists)$



Results: Reliability

- CBERS-4A (PUP). Mission time: 650 days.
- C1: 2A 2M [Hoque, 2016]; C2: 10A 10M (ours).
- **Scrubbing. $\Delta T = 9$ days.**

Reliability: highly affected by ΔT.



Overall Conclusions

- Using a formal method approach to complement a traditional one can worth the value.
- TMR: bad performance.
- Hamming Code: promising results.
- Scrubbing:
  - Reliability related to ΔT.
  - Safety more affected by the cover rate.

Application 2

Automated unit test case generation based on C++ source code.

Software Testing

- Sometimes, we just have the source code (no docs).
- Unit testing level: widely used in industry. Lots of frameworks and tools available.

Software Testing

- TIOBE index: November/2022.

Nov 2022	Nov 2021	Change	Programming Language	Ratings	Change
1	1		Python	17.18%	+5.41%
2	2		C	15.08%	+4.35%
3	3		Java	11.98%	+1.26%
4	4		C++	10.75%	+2.46%
5	5		C#	4.25%	-1.81%

Source: https://www.tiobe.com/tiobe-index/

Software Testing

- TIOBE index: November/2022.

Nov 2022	Nov 2021	Change	Programming Language	Ratings	Change
1	1		Python	17.18%	+5.41%
2	2		C	15.08%	+4.35%
3	3		Java	11.98%	+1.26%
4	4		C++	10.75%	+2.46%
5	5		C#	4.25%	-1.81%

Source: https://www.tiobe.com/tiobe-index/

Objective

- To contribute to the improvement of quality of software products created in the C++ programming language, via automated test case generation and FM [Eras, Santiago Júnior and Santos 2019][Eras 2020].

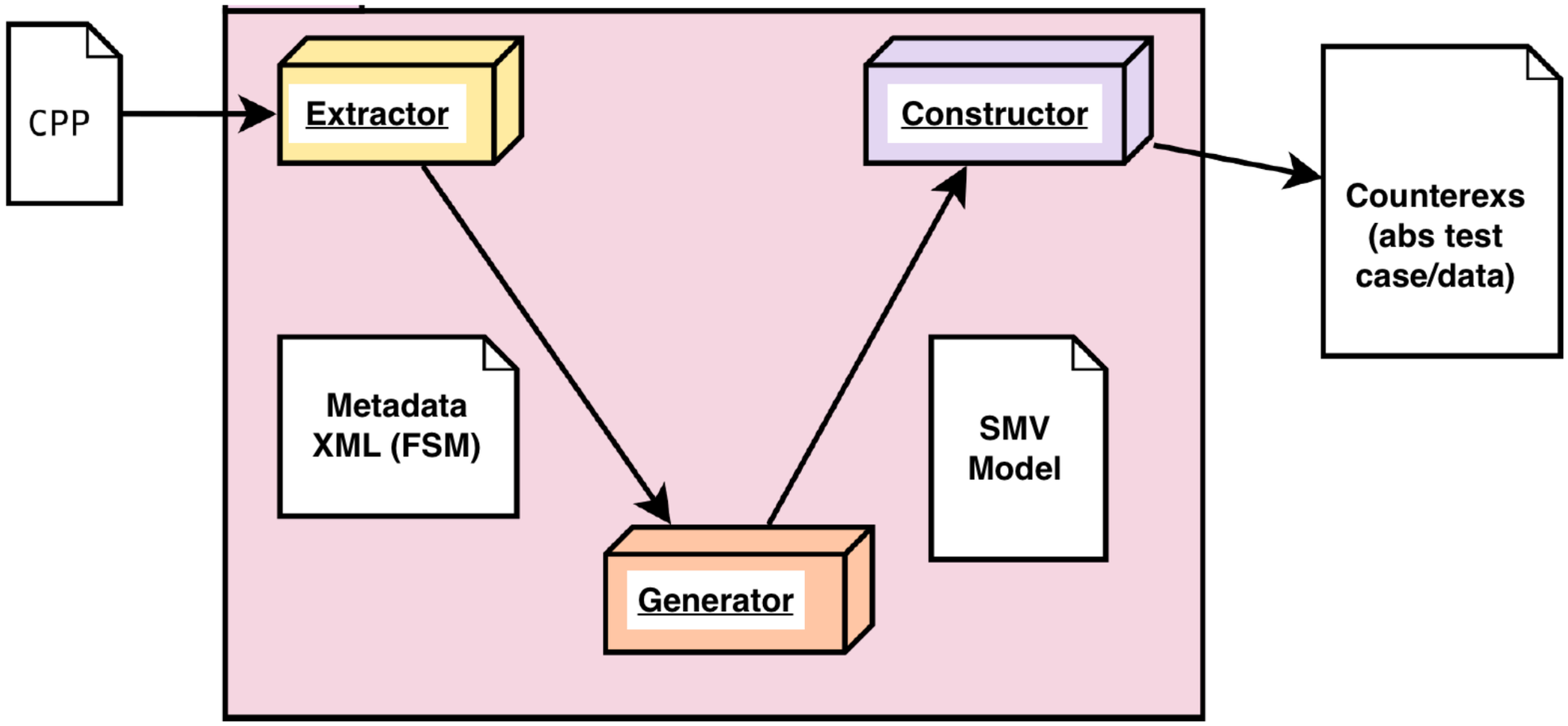
The Singularity Method

- Automated unit test case generation based on C++ source code.

- Reads the source code and performs successive transformations: Finite State Machine (FSM) ⟶ Control-Flow Graph (CFG) ⟶ NuSMV model checker (functional model checking).

- Trap properties: **counterexamples as test case/data.**

The Singularity Method



The Generator Module

- Translates the generated metadata (FSM) into an SMV model (NuSMV model checker).

- Trap properties: variation of the HiMoST method [Santiago Júnior and Silva 2017].
  - They force counterexample generation.
  - Formalised in Computation Tree Logic (CTL).

Trap Properties

- 1.) Negation of the events.

$$\forall \Box \neg (event \quad = e_i)$$

- 2.) Negation of the boolean transitions.

$$\forall \Box \alpha \wedge \gamma \rightarrow \exists \bigcirc \neg \beta$$

- 3.) Negation of the non-boolean transitions.

$$\begin{cases} \forall \Box \alpha \wedge \gamma \rightarrow \exists \bigcirc \neg \beta \\ \forall \Box \alpha \wedge \neg \gamma \rightarrow \exists \bigcirc \beta \end{cases}$$

Tool

Source: https://github.com/eduardoeras/Singularity

Case Studies

- C++ Geoinformatics applications (mature; non-trivial).

Classes evaluated: 24.

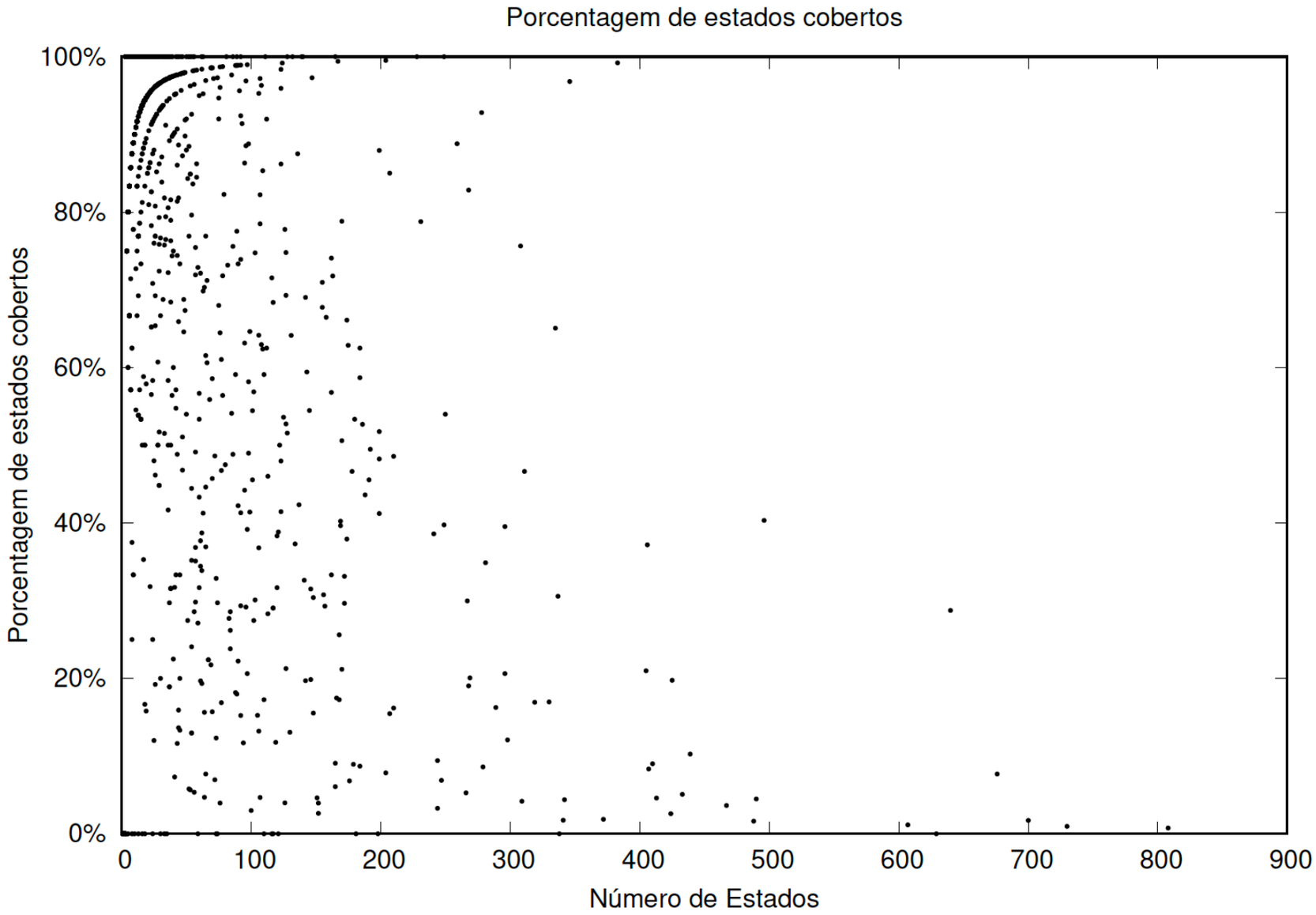
Classes evaluated: 1,063.

Results: TerraLib

- Metrics:
  - State coverage: #states of the model that appear in the counterexamples (test cases/data);
  - Transition coverage: #transitions of the model that appear in the counterexamples (test cases/data).

Results: TerraLib

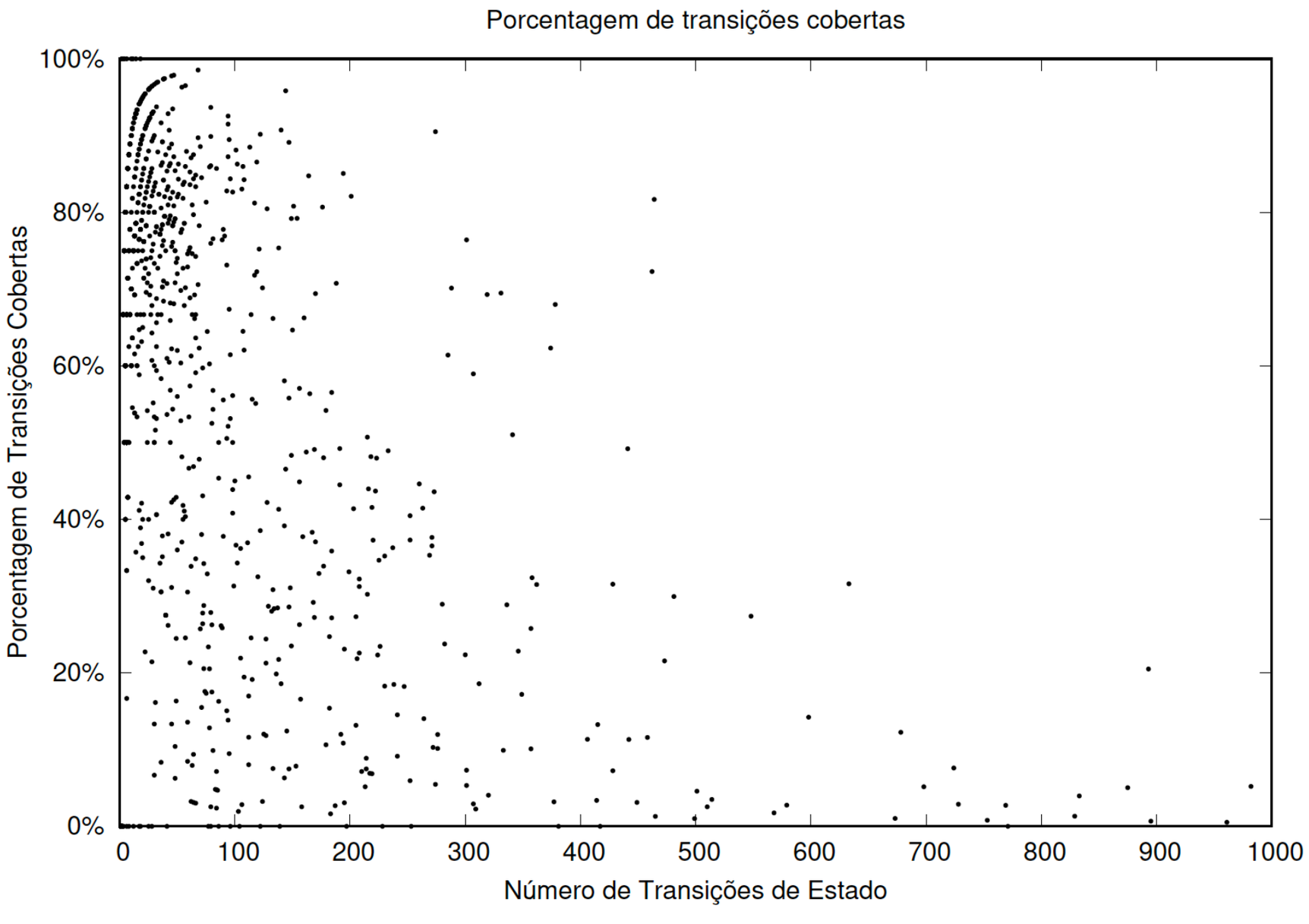
- state coverage X #states/class.



Up to 100 states/class: good state coverage.

Results: TerraLib

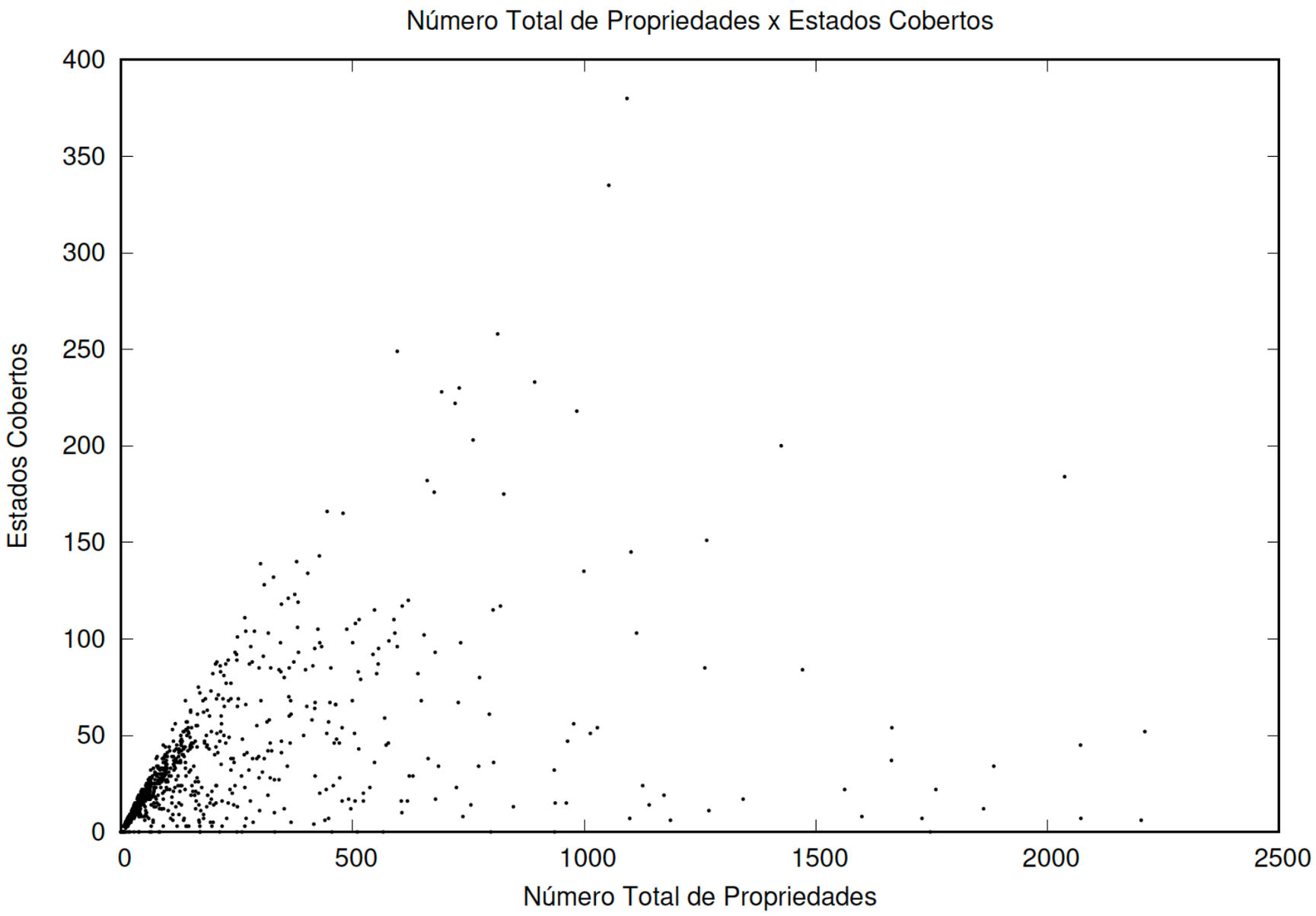
- transition coverage X #state transitions/class.



Up to 100 state transitions/class: good transition coverage.

Results: TerraLib

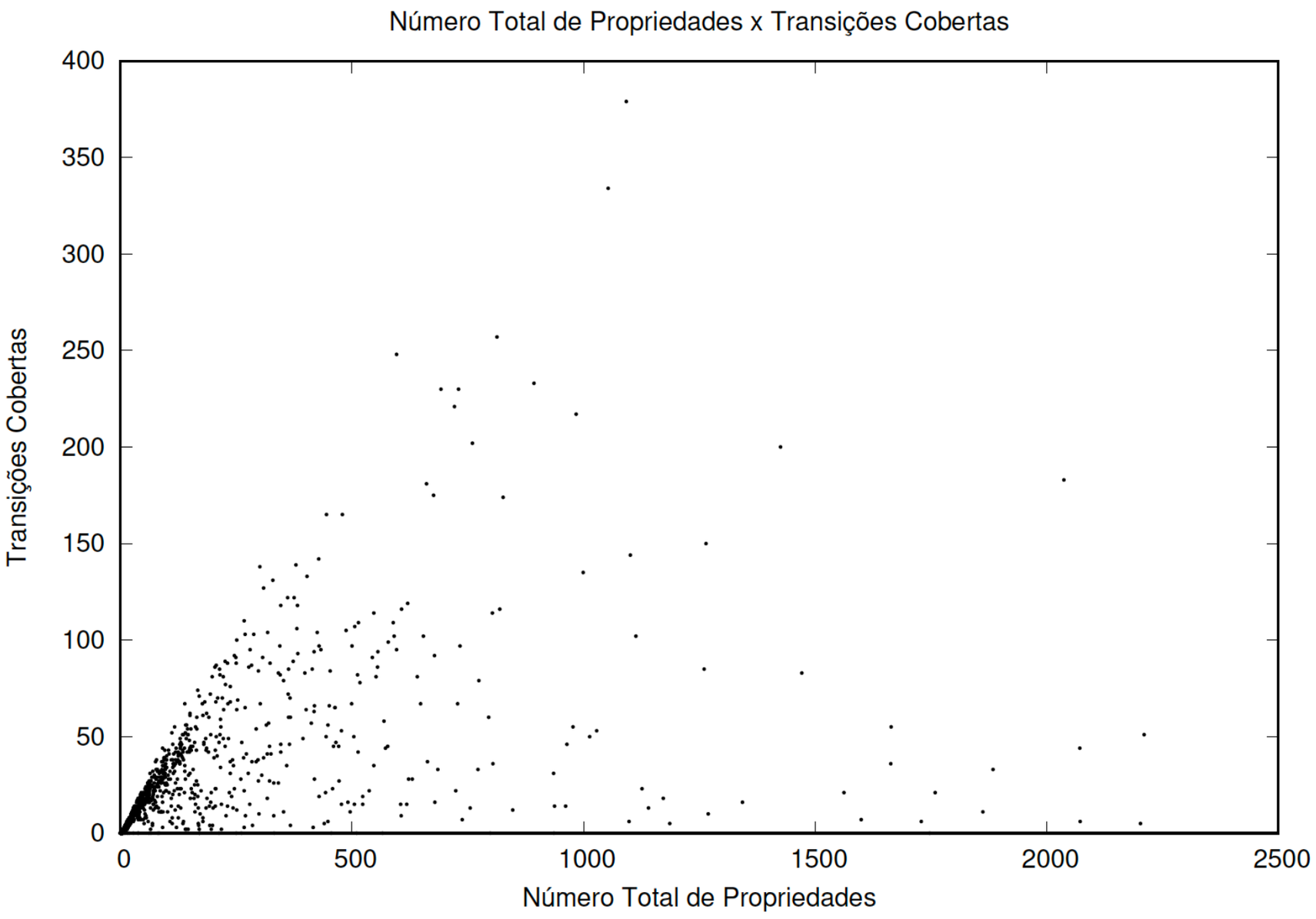
- state coverage X #generated properties/class.



Up to 100 properties/class: good state coverage.

Results: TerraLib

- transition coverage X #generated properties/class.

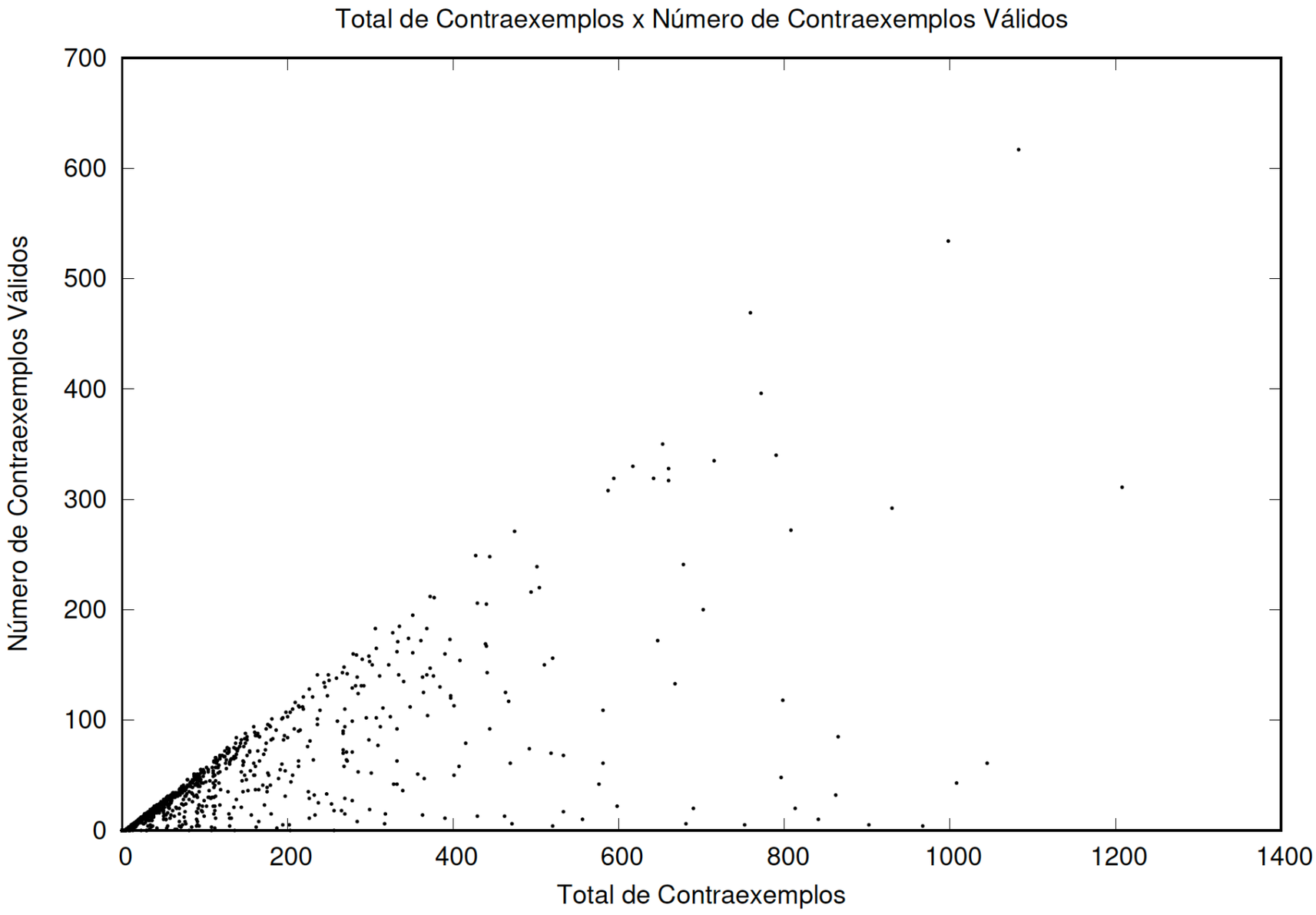


Up to 100 properties/class: good transition coverage.

Generating a large number of properties does not imply a good coverage of the model (source code).

Results: TerraLib

- #valid counterexamples X #all counterexamples/class.

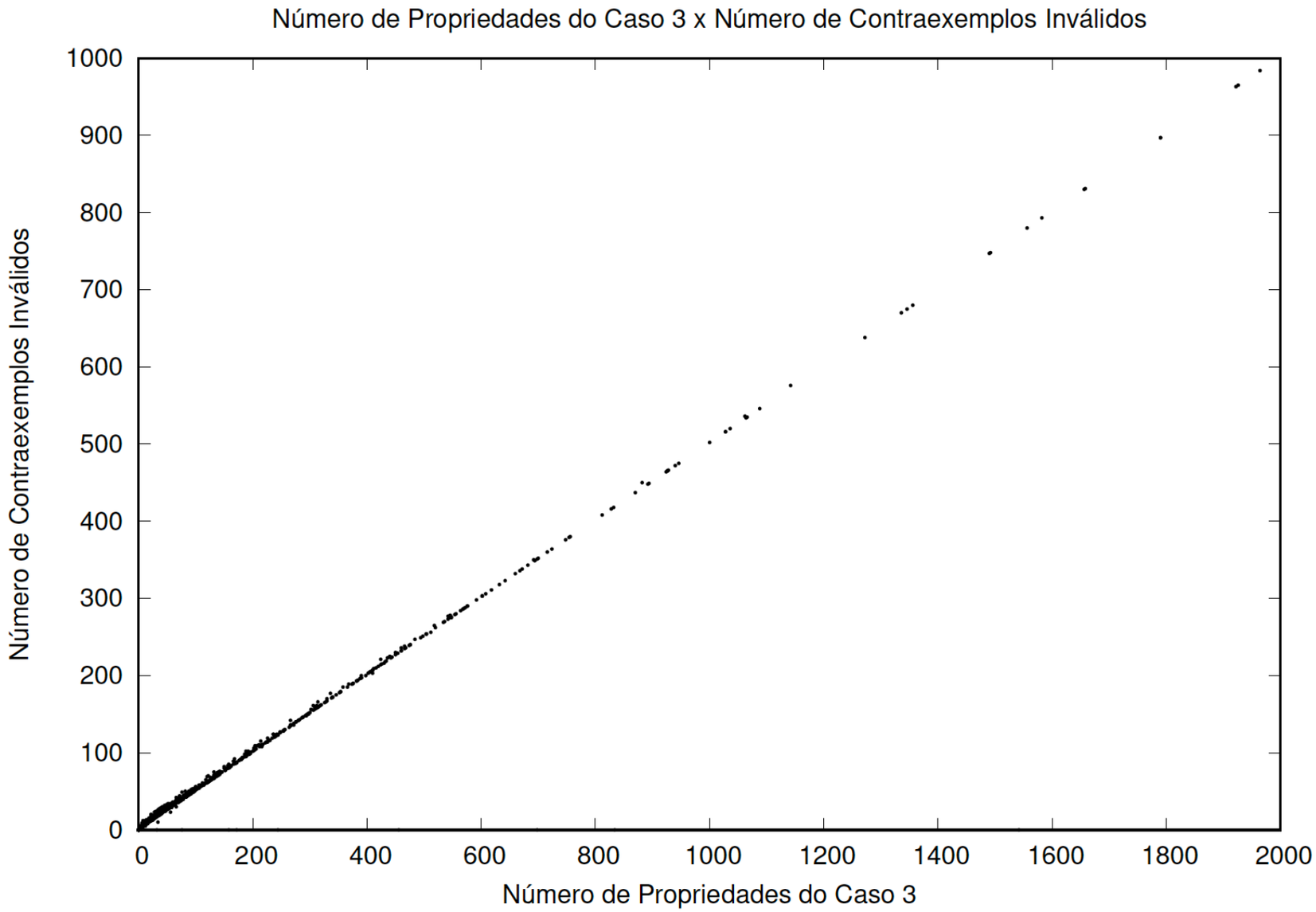


Up to 100 counterexamples/class: good.

Valid counterexample: > 2 states.

Results: TerraLib

- #invalid counterexamples X #case 3 properties/class.



$$\begin{cases} \forall \Box \alpha \wedge \gamma \rightarrow \exists \bigcirc \neg \beta \\ \forall \Box \alpha \wedge \neg \gamma \rightarrow \exists \bigcirc \beta \end{cases}$$

Case 3: generated the highest number of properties.

Invalid counterexample: ≤ 2 states.

Overall Conclusions

- The Singularity method: “effective” counterexamples (test cases) up to 100 states/class and 100 transitions/class.
- Generating a large number of properties does not imply a good coverage of the model (source code).
- Lots of invalid counterexamples particularly due to case 3. Need to carefully choose the properties templates.
- We tried some properties templates based on the specification patterns. But, in general, no good results considering our case studies.

Application 3

Assessment of the safety of navigation systems for aircrafts.

System Safety Assessment

- Safety assessment: mandatory for certification of aircraft complying with Part 25 airworthiness standards.

- Airworthiness: measure of an aircraft's suitability for safe flight.

Source: https://www.embraercommercialaviation.com/commercial-jets/e195/

System Safety Assessment

- Industry guidelines include FAA AC 25.1309-1A (1B) and SAE ARP-4761.
- Several methods for compliance demonstration analysis are predicted:
  - Failure Mode and Effect Analysis (FMEA);
  - Failure Hazard Analysis (FHA);
  - Fault Tree Analysis (FTA), …

System Safety Assessment

- Alternative to the traditional FTA: probabilistic models (predicted in the SAE ARP-4761).

- Advantages of probabilisitic model checking over FTA:
  - Expressing a temporal sequence of events or state-dependent behaviour of systems;
  - Possible automation of certain parts of the safety assessment activity.

Objective

- Investigating the contribution of FM (particularly probabilistic model checking) to the activity of assessing the safety of aircraft navigation systems, providing an alternative to **complement** the study carried out by using classical but non-formal methods [Pasa and Santiago Júnior, 2021].

Failure Conditions and Hazard Classifications

Table 1. Relationship between probability and severity of failure condition as per AC 25.1309-1B [7].

Failure condition classification	No safety effect	Minor	Major	Hazardous	Catastrophic
Effect on occupants	Inconvenience	Physical discomfort	Physical distress, possibly including injuries	Serious or fatal injury to a small number of passengers or cabin crew	Multiple fatalities
Allowable qualitative probability	No Probability Requirement	Probable	Remote	Extremely Remote	Extremely Improbable
Allowable quantitative probability	No Probability Requirement	$<10^{-3}$	$<10^{-5}$	$<10^{-7}$	$<10^{-9}$

Failure Conditions and Hazard Classifications

Table 2. Hazard classifications for the subset of aircraft positioning and navigation functions being evaluated (adapted from FAA AC 20-138D [8]).

	Advisory Vertical Guidance	Enroute/Terminal Area/Non-precision Approach (LNAV or RNP 0.3)	Non-precision Approach with Vertical Guidance (LNAV/VNAV)	LP/LPV Approach	GLS Approach
Loss of Navigation	No Effect	Major	Major	Major	Major
Misleading Information	Minor	Major	Major	Hazardous	Hazardous

System safety assessment for navigation systems of a generic commercial transport category aircraft.

Failure Conditions and Hazard Classifications

Table 2. Hazard classifications for the subset of aircraft positioning and navigation functions being evaluated (adapted from FAA AC 20-138D [8]).

	Advisory Vertical Guidance	Enroute/Terminal Area/Non-precision Approach (LNAV or RNP 0.3)	Non-precision Approach with Vertical Guidance (LNAV/VNAV)	LP/LPV Approach	GLS Approach
Loss of Navigation	No Effect	Major	Major	Major	Major
Misleading Information	Minor	Major	Major	Hazardous	Hazardous

Failure conditions.

Failure Conditions and Hazard Classifications

Table 2. Hazard classifications for the subset of aircraft positioning and navigation functions being evaluated (adapted from FAA AC 20-138D [8]).

	Advisory Vertical Guidance	Enroute/Terminal Area/Non-precision Approach (LNAV or RNP 0.3)	Non-precision Approach with Vertical Guidance (LNAV/VNAV)	LP/LPV Approach	GLS Approach
Loss of Navigation	No Effect	Major	Major	Major	Major
Misleading Information	Minor	Major	Major	Hazardous	Hazardous

Operations.

Each failure condition is applicable to a certain type of operation, mostly related to approach procedures (navigation leading to landing).

Failure Conditions and Hazard Classifications

Table 2. Hazard classifications for the subset of aircraft positioning and navigation functions being evaluated (adapted from FAA AC 20-138D [8]).

	Advisory Vertical Guidance	Enroute/Terminal Area/Non-precision Approach (LNAV or RNP 0.3)	Non-precision Approach with Vertical Guidance (LNAV/VNAV)	LP/LPV Approach	GLS Approach
Loss of Navigation	No Effect	Major	Major	Major	Major
Misleading Information	Minor	Major	Major	Hazardous	Hazardous

Operation Enroute/Terminal Area/Non-precision Approach (LNAV or RNP 0.3): The "Enroute/Terminal Area/Non-precision Approach" function is essentially the capacity for the aircraft to determine its lateral position (latitude and longitude) with the appropriate level of position uncertainty. Lateral Navigation (LNAV) is a mode of operation where the aircraft lateral trajectory is typically controlled by the Flight Management System (FMS). Required Navigation Performance (RNP) 0.3 is a similar mode, but it has a stricter position uncertainty limit of 0.3 nautical miles.

High-level System Architecture

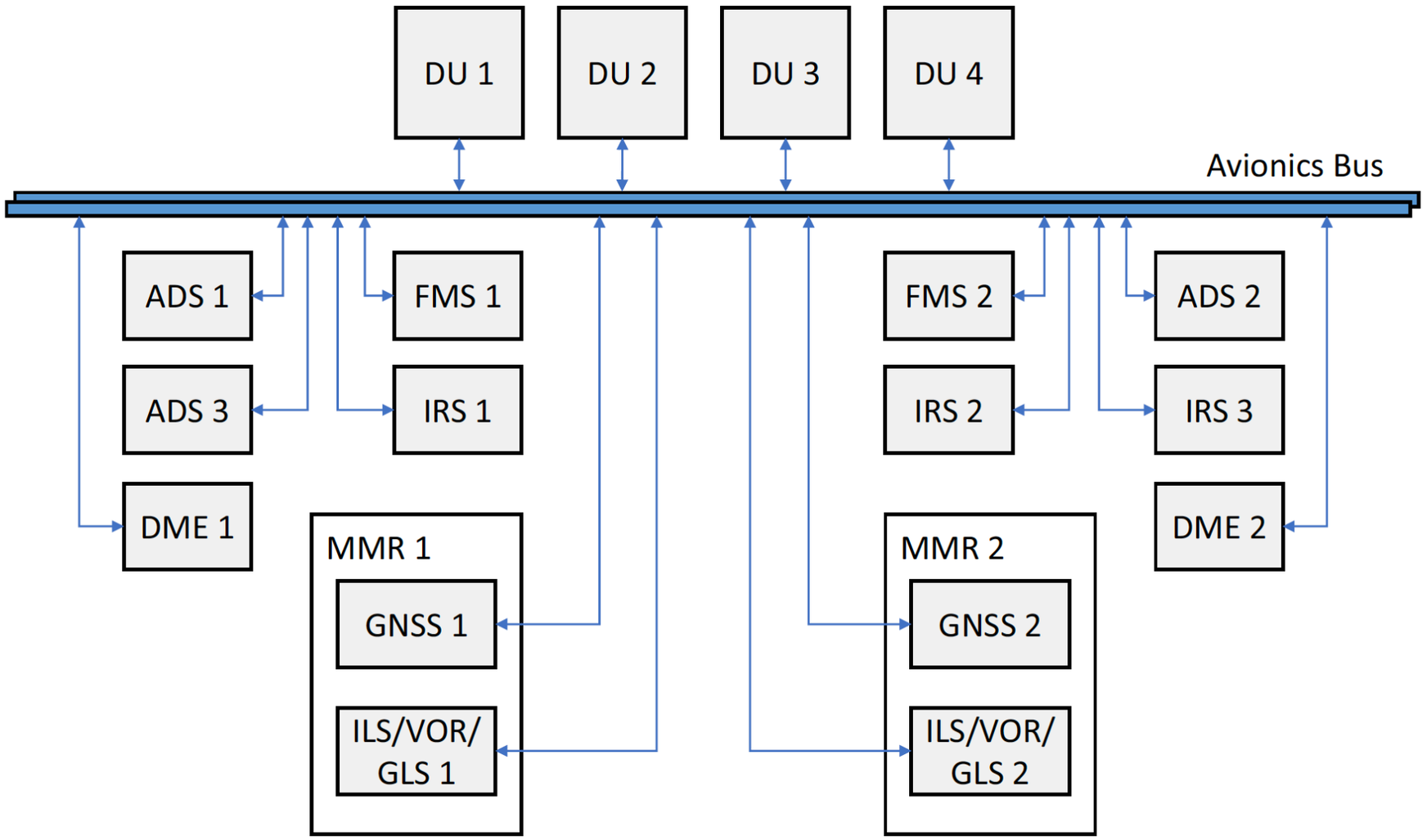


Fig. 1. High-level avionics architecture, including Display Units (DU), Air Data System (ADS), Flight Management System (FMS), Inertial Reference System (IRS), Distance Measuring Equipment (DME), Multi-Mode Receiver (MMR), Global Navigation Satellite System (GNSS), Instrument Landing System (ILS), VHF Omnidirectional Range (VOR), GNSS Landing System (GLS).

Flight Phases and Exposure Times

Percentage of fatal accidents and onboard fatalities | 2009 through 2018

	Taxi, load/ unload, parked, tow	12%		Climb (flaps up)	Cruise	Descent	Initial approach	49%	
		Takeoff	Initial climb					Final approach	Landing
Fatal accidents	10%	6%	6%	10%	14%	2%	4%	25%	24%
Onboard fatalities	0%	5%	1%	12%	31%	3%	12%	27%	10%
		6%						37%	
Exposure (percentage of flight time estimated for a 1.5-hour flight)		1%	1%	14%	57%	Initial approach fix 11%	Final approach fix 12%	3%	1%

Note: Percentages may not sum to 100% because of numerical rounding.

Fig. 2. Commercial airplanes statistics on fatal accidents, on-board fatalities, and exposure times per flight phase. Adapted from [2].

Flight Phases and Exposure Times

Critical phases.

Percentage of fatal accidents and onboard fatalities | 2009 through 2018

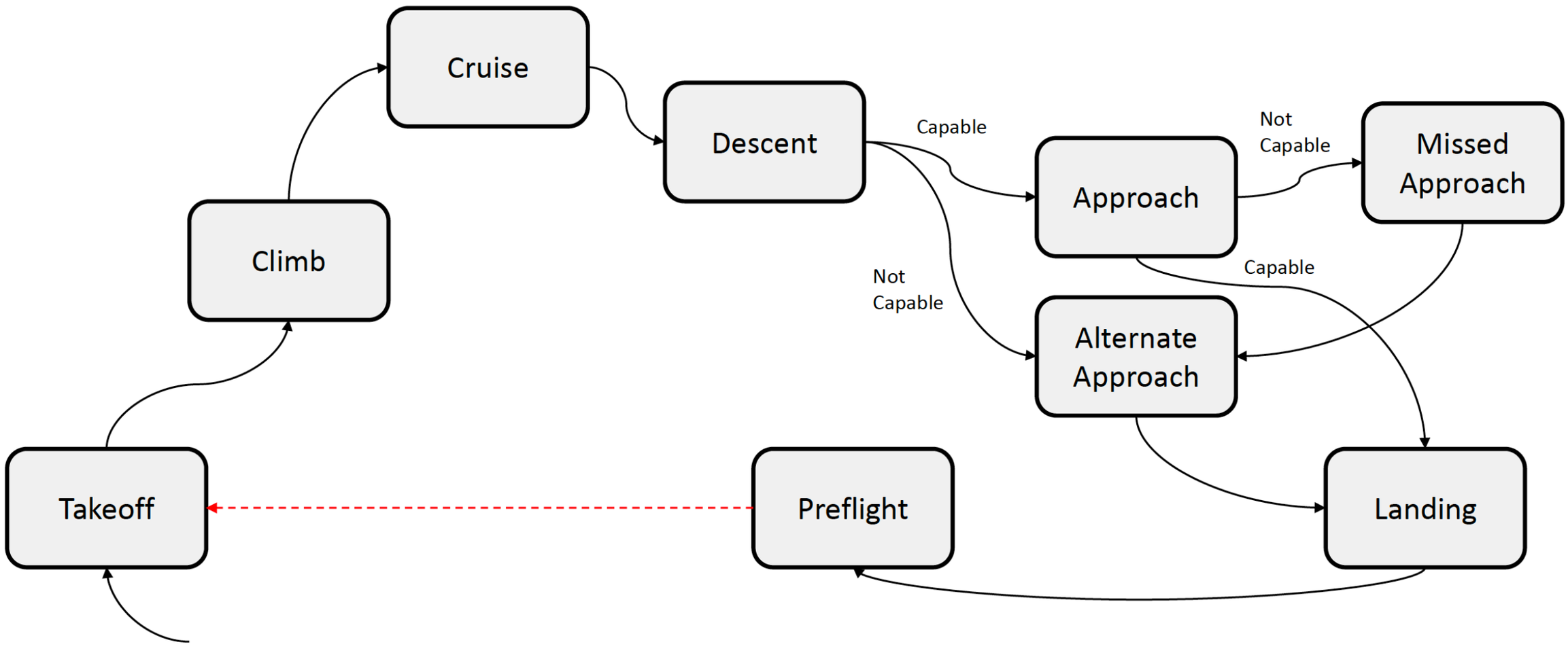
	Taxi, load/ unload, parked, tow	12%		Climb (flaps up)	Cruise	Descent	Initial approach	49%	
		Takeoff	Initial climb					Final approach	Landing
Fatal accidents	10%	6%	6%	10%	14%	2%	4%	25%	24%
Onboard fatalities	0%	5%	1%	12%	31%	3%	12%	27%	10%
		6%						37%	
Exposure (percentage of flight time estimated for a 1.5-hour flight)		1%	1%	14%	57%	Initial approach fix 11%	Final approach fix 12%	3%	1%

Note: Percentages may not sum to 100% because of numerical rounding.

Fig. 2. Commercial airplanes statistics on fatal accidents, on-board fatalities, and exposure times per flight phase. Adapted from [2].

CTMC Models

Flight phases



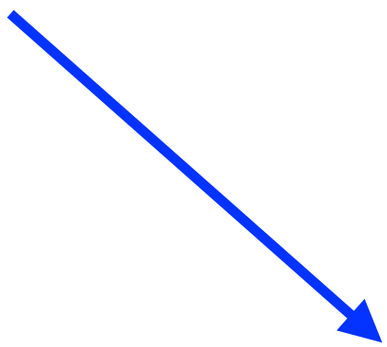
CTMC Models

Flight phase state transitions (PRISM).

```
// Typical flight duration (in hours)
const double flight_duration = 1.5;

// Mean times spent on each flight phase (in hours)
const double time_in_takeoff = 0.01 * flight_duration;
const double time_in_climb = 0.15 * flight_duration;
const double time_in_cruise = 0.57 * flight_duration;
const double time_in_descent = 0.11 * flight_duration;
const double time_in_approach = 0.15 * flight_duration;
const double time_in_landing = 0.01 * flight_duration;
const double time_in_missed_approach = time_in_climb;

module flight_phase
  // Phases of flight: 0 = pre-flight; 1 = takeoff;
  // 2 = climb; 3 = cruise; 4 = descent;
  // 5 = approach; 6 = alternate approach;
  // 7 = landing; 8 = missed approach
  phase: [0..8] init 1;
```



```
  // Takeoff to climb
  [] (phase=1) -> 1/time_in_takeoff: (phase'=2);
  // Climb to cruise
  [] (phase=2) -> 1/time_in_climb: (phase'=3);
  // Cruise to descent
  [] (phase=3) -> 1/time_in_cruise: (phase'=4);
  // Descent to approach / alternate approach
  [] (phase=4) & (!loss_of_navigation) -> 1/time_in_descent: (phase'=5);
  [] (phase=4) & (loss_of_navigation) -> 1/time_in_descent: (phase'=6);
  // Approach to landing / missed approach
  [] (phase=5) & (!loss_of_navigation) -> 1/time_in_approach: (phase'=7);
  [] (phase=5) & (loss_of_navigation) -> 1E5: (phase'=8);
  // Alternate approach to landing
  [] (phase=6) -> 1/time_in_approach: (phase'=7);
  // Missed approach to alternate approach
  [] (phase=8) -> 1/time_in_missed_approach: (phase'=6);
  // Landing to pre-flight
  [] (phase=7) -> 1/time_in_landing: (phase'=0);
endmodule
```

CTMC Models

Table 3. Equipment failure mode rates.

System	Failure Rate (per hour)	Erroneous Rate (per hour)
FMS	1E-6	1E-7
GNSS	1E-4	1E-6
IRS	1E-6	1E-8
DME	1E-5	1E-6
ADS	1E-6	1E-8
GLS	1E-5	1E-7

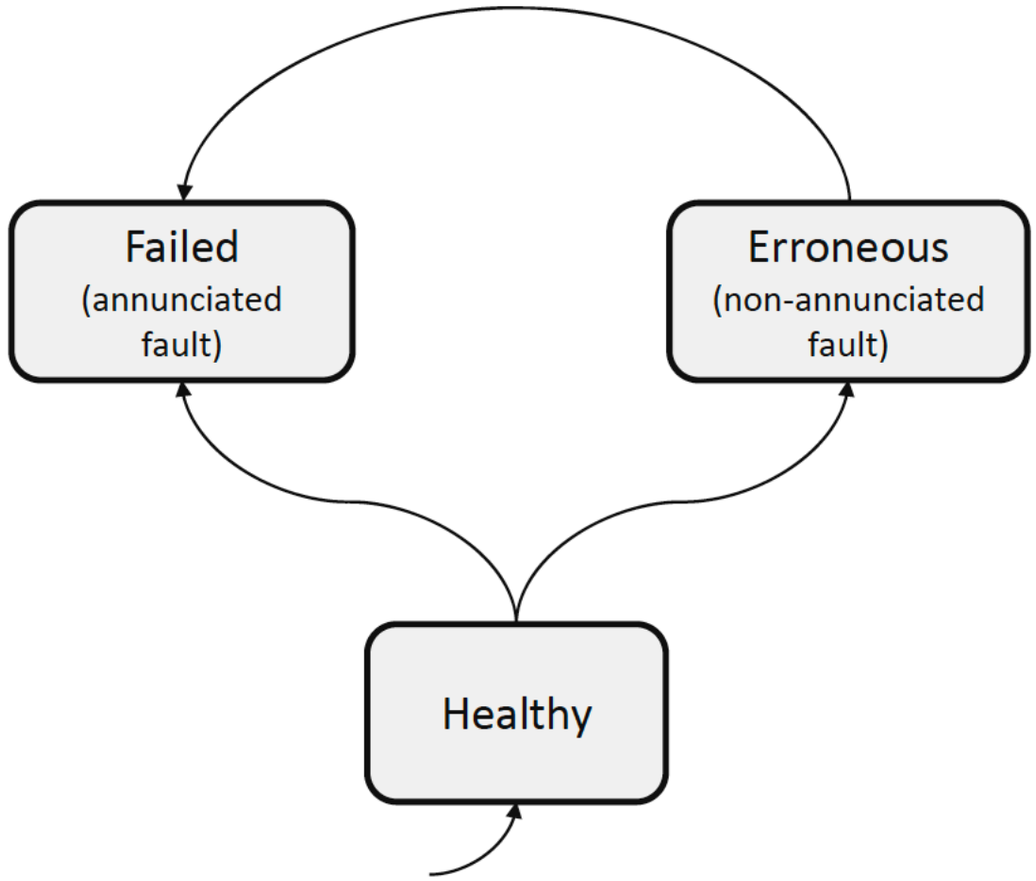


Fig. 3. Equipment state transitions.

All systems/subsystems start as healthy and may present one out of two possible failure modes: they may become failed (annunciated fault) or may present erroneous information (non-annunciated fault).

CTMC Models

```
// ---------------------------------
// Inertial Reference System (IRS)
// ---------------------------------
const double irs_failure_rate = 1E-6;
const double irs_erroneous_rate = 1E-8;

module irs1
  irs1_fail: bool init false;
  irs1_err: bool init false;

  [] true -> irs_failure_rate : (irs1_fail'=true)&(irs1_err'=false);
  [] (!irs1_fail) -> irs_erroneous_rate : (irs1_err'=true);
endmodule
module irs2 = irs1[irs1_fail=irs2_fail, irs1_err=irs2_err] endmodule
module irs3 = irs1[irs1_fail=irs3_fail, irs1_err=irs3_err] endmodule
```

Listing 1.1. Sample of PRISM code implementing the IRS. All other systems are implemented similarly.

CTMC Models

Table 4. Characteristics of the CTMC models.

Operation	# Reachable states	# Transitions
LNAV/RNP 0.3	159,787	1,719,943
LNAV/VNAV	4,331,609	59,590,374
LP/LPV	16,771	146,329
GLS	158,011	1,692,511

Safety Properties

- CSL.

- Two types of failure conditions:
  - Loss of navigation;
  - Misleading information (the systems indications are erroneous to a point where the pilots could be mislead, which may be a more severe failure condition).

Safety Properties

- Loss of navigation safety property for the LNAV/RNP 0.3 approach.

$$\mathcal{P}_{<10^{-5}\times T}[\Diamond(\text{enroute\_terminal\_approach} \wedge \text{loss\_of\_navigation})]$$

- Misleading information safety property for the LNAV/RNP 0.3 approach.

$$\mathcal{P}_{<10^{-5}\times T}[\Diamond(\text{enroute\_terminal\_approach} \wedge \text{misleading\_information})]$$

Safety Properties

- Loss of navigation safety property for the LNAV/RNP 0.3 approach.

$$\mathcal{P}_{<10^{-5}\times T}[\Diamond(\text{enroute\_terminal\_approach} \wedge \text{loss\_of\_navigation})]$$

- Misleading information safety property for the LNAV/RNP 0.3 approach.

$$\mathcal{P}_{<10^{-5}\times T}[\Diamond(\text{enroute\_terminal\_approach} \wedge \text{misleading\_information})]$$

The bounds of the probabilities depend on the flight duration time (T).

Safety Properties

- Loss of navigation safety property for the GLS approach.

$$\mathcal{P}_{<10^{-5}\times T}[\Diamond(\text{phase} = 8)]$$

- Misleading information safety property for the GLS approach.

$$\mathcal{P}_{<10^{-7}\times T}[\Diamond(\text{phase} = 5 \wedge \text{misleading\_information})]$$

Results

Table 5. Safety properties results as evaluated by PRISM.

Operation	Failure mode	Classification	Objective	Probability	Compliant
LNAV/RNP 0.3	Loss of Navigation	Major	1.5E-5	2.97E-6	✓
	Misleading Information	Major	1.5E-5	6.24E-6	✓
LNAV/VNAV	Loss of Navigation	Major	1.5E-5	4.50E-7	✓
	Misleading Information	Major	1.5E-5	6.24E-6	✓
LP/LPV	Loss of Navigation	Major	1.5E-5	6.82E-9	✓
	Misleading Information	Hazardous	1.5E-7	3.71E-12	✓
GLS	Loss of Navigation	Major	1.5E-5	8.23E-9	✓
	Misleading Information	Hazardous	1.5E-7	4.41E-12	✓

Results

Table 5. Safety properties results as evaluated by PRISM.

Operation	Failure mode	Classification	Objective	Probability	Compliant
LNAV/RNP 0.3	Loss of Navigation	Major	1.5E-5	2.97E-6	✓
	Misleading Information	Major	1.5E-5	6.24E-6	✓
LNAV/VNAV	Loss of Navigation	Major	1.5E-5	4.50E-7	✓
	Misleading Information	Major	1.5E-5	6.24E-6	✓
LP/LPV	Loss of Navigation	Major	1.5E-5	6.82E-9	✓
	Misleading Information	Hazardous	1.5E-7	3.71E-12	✓
GLS	Loss of Navigation	Major	1.5E-5	8.23E-9	✓
	Misleading Information	Hazardous	1.5E-7	4.41E-12	✓

The maximum acceptable probability.

Results

- Extending flight duration: from $T = 1.5$ h to $T = 18$ h. Still Ok!

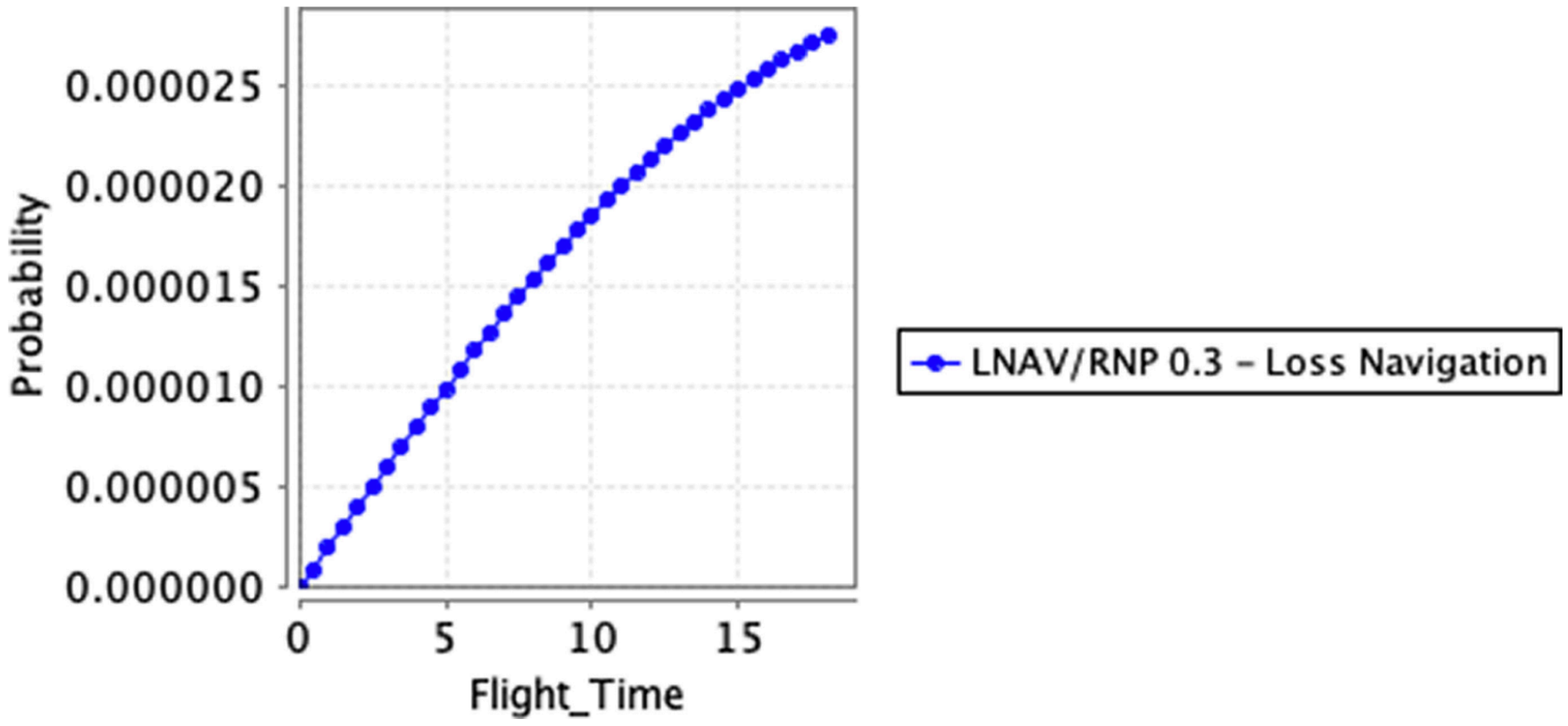


Fig. 5. LNAV/RNP 0.3: Loss of Navigation, $T = 18$ h.

Overall Conclusions

- Probabilistic model checking: feasible for the activity of assessing the safety of aircraft navigation systems.
- But, we must not neglect the classical methods (FTA, …).

Final Remarks

- FM (probabilistic and functional model checking): successfully used for non-trivial applications.

- In the current stage, FM should be used as a complementary approach, aiding non-formal/classical solutions.

Final Remarks

- Scalability and lack of transparency still seem to be big issues in the practical settings related to FM.

- In practical terms, be formal but not “so” formal!

References

- [FAA 2016]. FEDERAL AVIATION ADMINISTRATION (FAA). Single Event Effects Mitigation Techniques Report. 2016.
- [Hoque 2016]. HOQUE, KHAZA ANUARUL. Early dependability analysis of FPGA-based space applications using formal verification. PhD Thesis, Concordia University, Montréal, Québec, Canada, 2016.
- [Pereira, Santiago Júnior and Manea 2017]. PEREIRA, VINY CESAR; SANTIAGO JÚNIOR, VALDIVINO ALEXANDRE DE; MANEA, SILVIO. SEU Mitigation for SRAM FPGAs: A comparison via Probabilistic Model Checking. In: XVIII Workshop de Testes e Tolerância a Falhas (WTF), XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 2017, Belém, PA. Anais do XVIII Workshop de Testes e Tolerância a Falhas - WTF. Porto Alegre, RS: Sociedade Brasileira de Computação (SBC), 2017, p. 56-69.

References

- [Pereira 2018]. PEREIRA, VINY CESAR. Model Checking Probabilístico para Apoiar a Mitigação de Evento de Falta Única em Field Programmable Gate Arrays (FPGAs). 2018. Master in Applied Computing, Instituto Nacional de Pesquisas Espaciais (INPE).

References

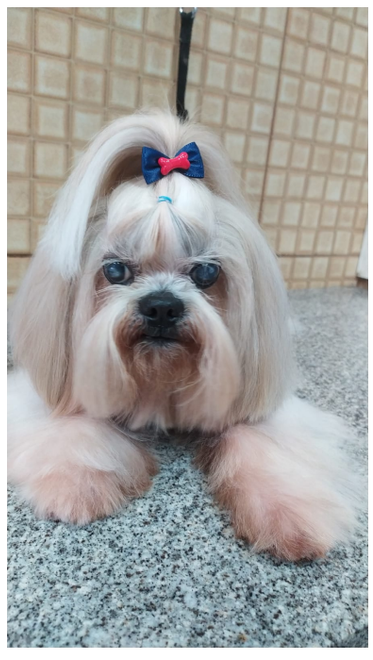
- [Eras, Santiago Júnior and Santos 2019]. ERAS, EDUARDO ROHDE; SANTIAGO JÚNIOR, VALDIVINO ALEXANDRE DE; SANTOS, LUCIANA BRASIL REBELO DOS. Singularity: A methodology for automatic unit test data generation for C++ applications based on Model Checking counterexamples. In: Proceedings of the IV Brazilian Symposium on Systematic and Automated Software Testing (SAST 19), 2019, Salvador, BA, p. 72-79.

- [Eras 2020]. ERAS, EDUARDO ROHDE. Singularity: Um Método para Geração Automática de Casos de Testes Unitários baseado em Contraexemplos de Verificador de Modelos para Aplicações em C++. 2020. Master in Applied Computing, Instituto Nacional de Pesquisas Espaciais (INPE).

References

- [Santiago Júnior and Silva 2017]. SANTIAGO JÚNIOR, VALDIVINO ALEXANDRE DE; SILVA, FELIPE ELIAS COSTA DA. From Statecharts into Model Checking: A Hierarchy-based Translation and Specification Patterns Properties to Generate Test Cases. In: Proceedings of the 2nd Brazilian Symposium on Systematic and Automated Software Testing (SAST 2017). ACM, New York, NY, USA, Article 2, 10 pages.
- PASA, GABRIEL DUARTE; SANTIAGO JÚNIOR, VALDIVINO ALEXANDRE DE. Aircraft Navigation Systems Safety Assessment via Probabilistic Model Checking. In: Gervasi, O. et al. (eds), Computational Science and Its Applications - ICCSA 2021. Lecture Notes in Computer Science, vol 12952, p. 465-480, Springer, Cham.

Thank You!



E-mail: valdivino.santiago@inpe.br

Web: http://www.lac.inpe.br/~valdivino/

GitHub: https://github.com/vsantjr