Towards an FDIR Software Fault Tree Library for Onboard Computers

IEEE Aerospace 2020

Sascha Müller, Kilian Höflinger, Michal Smisek, Andreas Gerndt

DLR German Aerospace Center
Institute of Simulation and Software Technology
Software for Space Systems and Interactive Visualization
Braunschweig

March 13, 2020

Outline

- FDIR Analysis with Fault Trees
- Problem Statement
- FDIR Software Fault Tree Library
- Dependability Quality Model
- Use Case: MMX
- Results
- Conclusion

Fault Detection, Isolation and Recovery

FDIR

Even well designed systems cannot avoid the existence of faults

- But not every fault is a **failure**
- FDIR tries to prevent faults from turning into failures

Fault Detection, Isolation and Recovery

FDIR

Even well designed systems cannot avoid the existence of faults

- But not every fault is a **failure**
- FDIR tries to prevent faults from turning into failures

Fault → Detection → (Isolation) → Recovery

Detection
- Fault detected?

(Isolation)
- Where?
- Why?

Recovery
- Can we recover?
- How?

High Performance On-Board Computers and FDIR

On-Board Computing in Space

- Demand for powerful on-board computing is increasing (navigation, loss-less compression, artificial intelligence) due to communication limitations

High Performance On-Board Computers and FDIR

On-Board Computing in Space

- Demand for powerful on-board computing is increasing (navigation, loss-less compression, artificial intelligence) due to communication limitations
- Idea: Use Common of the shelf components

High Performance On-Board Computers and FDIR

On-Board Computing in Space

- Demand for powerful on-board computing is increasing (navigation, loss-less compression, artificial intelligence) due to communication limitations
- Idea: Use Common of the shelf components
- But: Space is a harsh environment
  - Radiation effects
  - Limited room for human intervention

High Performance On-Board Computers and FDIR

On-Board Computing in Space

- Demand for powerful on-board computing is increasing (navigation, loss-less compression, artificial intelligence) due to communication limitations
- Idea: Use Common of the shelf components
- But: Space is a harsh environment
  - Radiation effects
  - Limited room for human intervention

Need FDIR to provide stable operation

How to plan and assess the FDIR concept in the **design phase**?

Modeling the F in FDIR

Fault Model

Relationship between basic faults and how they lead to failures

- Failure Modes and Effects Analysis (FMECA)
- Reliability Block Diagrams
- Markov Modeling
- **Fault Tree Analysis (FTA)**
- ... and many more

Basics of Fault Trees

Fault Tree

How do faults propagate through components?

- Propagation model using gates (AND, OR, SPARE, PAND, etc.)

Basics of Fault Trees

Fault Tree

How do faults propagate through components?

- Propagation model using gates (AND, OR, SPARE, PAND, etc.)
- Leaves: Basic Events (Resolution of analysis)

Basics of Fault Trees

Fault Tree

How do faults propagate through components?

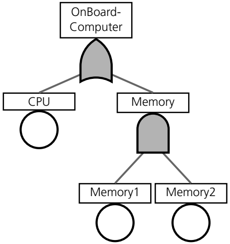
- Propagation model using gates (AND, OR, SPARE, PAND, etc.)
- Leaves: Basic Events (Resolution of analysis)
- Basic Events can have failure rates for quantitative evaluation

Basics of Fault Trees

Fault Tree

How do faults propagate through components?

- Propagation model using gates (AND, OR, SPARE, PAND, etc.)
- Leaves: Basic Events (Resolution of analysis)
- Basic Events can have failure rates for quantitative evaluation

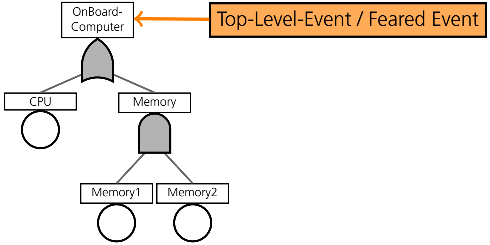


Basics of Fault Trees

Fault Tree

How do faults propagate through components?

- Propagation model using gates (AND, OR, SPARE, PAND, etc.)
- Leaves: Basic Events (Resolution of analysis)
- Basic Events can have failure rates for quantitative evaluation

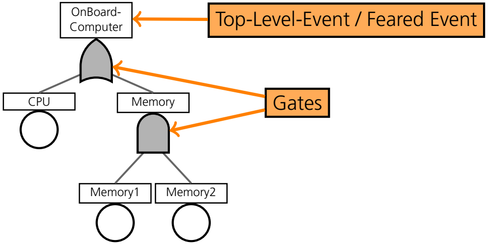


Basics of Fault Trees

Fault Tree

How do faults propagate through components?

- Propagation model using gates (AND, OR, SPARE, PAND, etc.)
- Leaves: Basic Events (Resolution of analysis)
- Basic Events can have failure rates for quantitative evaluation

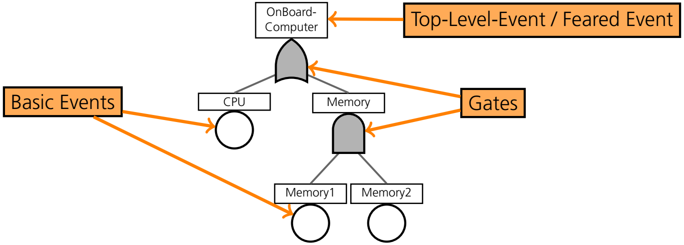


Basics of Fault Trees

Fault Tree

How do faults propagate through components?

- Propagation model using gates (AND, OR, SPARE, PAND, etc.)
- Leaves: Basic Events (Resolution of analysis)
- Basic Events can have failure rates for quantitative evaluation



Fault Trees and RAMS

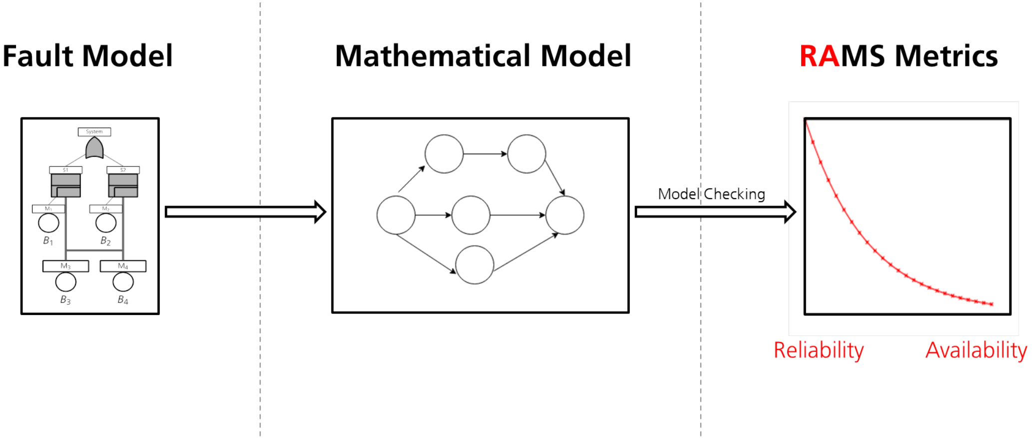


Figure: Fault Tree Evaluation

Problem Statement

Fault trees provide great analysis benefits but...

- also require a lot of modeling effort!
- need to be redone mostly from scratch for each new system!

Problem Statement

Fault trees provide great analysis benefits but...

- also require a lot of modeling effort!
- need to be redone mostly from scratch for each new system!

Can we develop..

- a generic FDIR software library
- **generic fault tree models for FDIR Software library?**
- **a methodology to easily generate fault trees incorporating calls to the library?**

FDIR Software Library

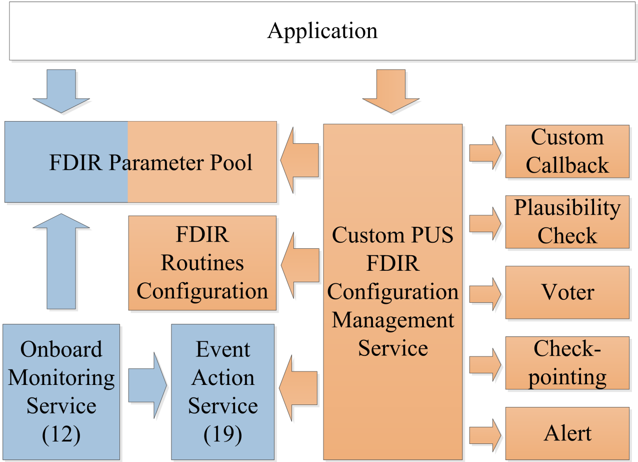


Figure: FDIR C++ Library Architecture

Fault Tree Generation

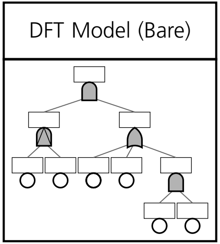


Figure: Generation process with DFT model and service library models.

Fault Tree Generation

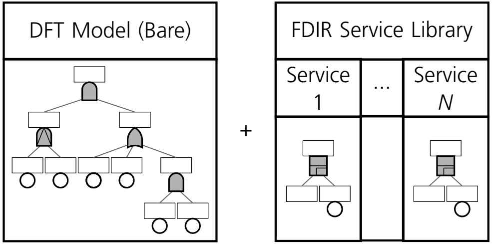


Figure: Generation process with DFT model and service library models.

Fault Tree Generation

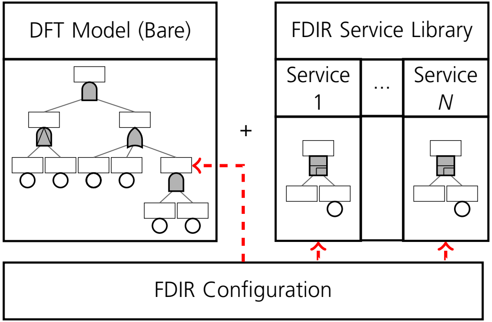


Figure: Generation process with DFT model and service library models.

Fault Tree Generation

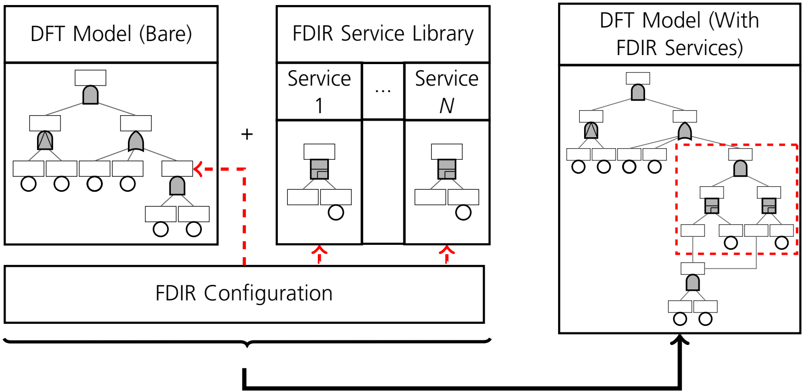


Figure: Generation process with DFT model and service library models.

Quality Model

Characteristic	Sub Characteristic	Metric	Threshold
Reliability	Reliability Evidence	Reliability after t	> 95% in 50 d
		Structural Coverage	100%*
Maintain-ability	Complexity	Cyclomatic Complexity	< 12*
	Modularity	Modular Coupling	< 4*
Availabilty	Availability Evidence	MTTF	> 50 d

Figure: Quality mode using factor-criteria-metric model (Based on ECSS)

Implementation - Virtual Satellite 4 FDIR

Virtual Satellite

- Model Based Systems Engineering tool
- FDIR Extension supporting Model-Based Fault Tree Analysis
- https://github.com/virtualsatellite

Implementation - Virtual Satellite 4 FDIR

Virtual Satellite

- Model Based Systems Engineering tool
- FDIR Extension supporting Model-Based Fault Tree Analysis
- https://github.com/virtualsatellite

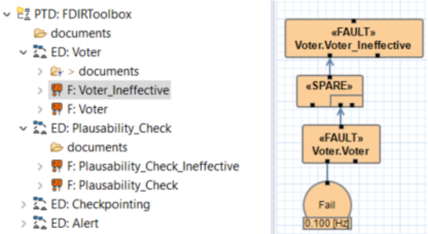


Figure: Excerpt of the library in VirSat

MMX

Mission

- Martian Moon eXploration (Phobos)
- Carries rover exploration with various payloads
- Limited communication windows and delays: High degree of autonomy required
- Single OBC based on COTS components

MMX

Mission

- Martian Moon eXploration (Phobos)
- Carries rover exploration with various payloads
- Limited communication windows and delays: High degree of autonomy required
- Single OBC based on COTS components

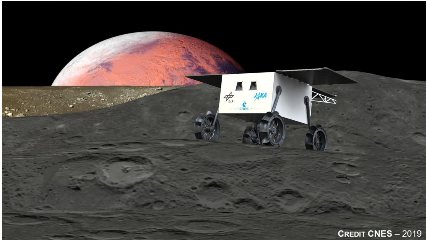


Figure: Artist impression of the MMX rover on Phobos (credit: CNES)

Services

Feared Events

- Loss of Position Service
- Loss of Obstacle Detection Service

Services

Feared Events

- Loss of Position Service
- Loss of Obstacle Detection Service

Applications

- **Rectification:** Reverses lens distortions
- **Depth Image Computation:** Computes disparity image and depth image
- **Visual Odometry:** Measures rover's egomotion
- **Obstacle Detection Algorithm:** Utilizes camera images and depth images to detects obstacles and terrain featuress

Basic Events

Main Events

- Short mission duration (50 days): Focus on short-term effects rather than long-term accumulation effects
- Main focus: Single-Event Effects (SEE)

Basic Events

Main Events

- Short mission duration (50 days): Focus on short-term effects rather than long-term accumulation effects
- Main focus: Single-Event Effects (SEE)

Main Hardware Components for Analysis

- Processing Logic (PL):
  - Sub-Components: BRAM, CRAM
  - SEE/day: 3.21
- Processing System (PL):
  - Sub-Components: OCM, D-CACHE, ALU, FPU, Peripheral
  - SEE/day: 8.22E-02

Bare Fault Tree Model

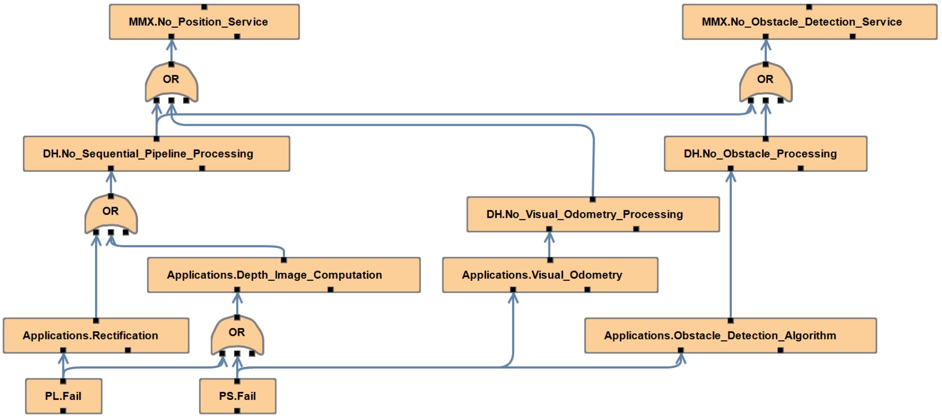


Figure: Bare fault tree model without FDIR

Summary of Evaluation Results

Experiment Setup

- Defined (sensible) configurations of increasing complexity
- FT size, configuration costs, MTTF, and reliability after 50 days

Summary of Evaluation Results

Experiment Setup

- Defined (sensible) configurations of increasing complexity
- FT size, configuration costs, MTTF, and reliability after 50 days

Results

- Could answer the question if simpler configurations would suffice

Summary of Evaluation Results

Experiment Setup

- Defined (sensible) configurations of increasing complexity
- FT size, configuration costs, MTTF, and reliability after 50 days

Results

- Could answer the question if simpler configurations would suffice (sadly: No)

Summary of Evaluation Results

Experiment Setup

- Defined (sensible) configurations of increasing complexity
- FT size, configuration costs, MTTF, and reliability after 50 days

Results

- Could answer the question if simpler configurations would suffice (sadly: No)
- Generated fault trees with ∼100 nodes

Summary of Evaluation Results

Experiment Setup

- Defined (sensible) configurations of increasing complexity
- FT size, configuration costs, MTTF, and reliability after 50 days

Results

- Could answer the question if simpler configurations would suffice (sadly: No)
- Generated fault trees with ∼100 nodes
- Reduced modeling effort by 80% for most complex configuration

Conclusion

Recap

- Separated software fault model (Bare model) and mitigation fault models (Fault Tree Library)
- Reusable
- Achieved significant reduction on modeling effort

Conclusion

Recap

- Separated software fault model (Bare model) and mitigation fault models (Fault Tree Library)
- Reusable
- Achieved significant reduction on modeling effort

Towards the future

- Automatic optimization of configurations? Constraints defining what a "sensible" configuration is?
- Coupling with code generation

Conclusion

Recap

- Separated software fault model (Bare model) and mitigation fault models (Fault Tree Library)
- Reusable
- Achieved significant reduction on modeling effort

Towards the future

- Automatic optimization of configurations? Constraints defining what a "sensible" configuration is?
- Coupling with code generation

Thank You!! Questions?