# 21310 - Verifying Software Control Categories (SCCs) Using Quantitative Fault Tree Analyses (FTAs)

Robert E. Smith, CSP
Booz Allen Hamilton
2018 NDIA 21st SE Conference
Tampa, FL
25 October 2018

# Agenda

- Introduction
- Background / Hypothesis
- Assumptions and Decomposing SCC Definitions
- Fault Tree Examples
- Case Study Verification
- Results
- Conclusion and Recommendations
- References

# Disclaimer

- Any views or opinions presented in this presentation are solely those of the author/presenter and do not represent those of Booz Allen Hamilton nor the U.S. Department of Defense (DoD)

# Introduction

- Proposes an approach to use quantitative Fault Tree Analyses (FTAs) to determine and/or verify the Software Control Category (SCC) assignments in accordance with MIL-STD-882E
- Evaluates and decomposes SCC definitions from MIL-STD-882E
- Uses quantitative FTAs and sets software failures to 1 (always fails) to assess the impact of the top event failure probability to help determine the proper alignment of the SCC assessment to the SCC definitions

***Goal – Evaluating the feasibility of using quantitative FTAs to verify SCCs and SwCIs***

# Background

- Software Failure Probability
  - Extremely difficult to determine -- *this paper does not attempt to research this topic*
    - MIL-STD-882E - *"Determining the probability of failure of a single software function is difficult at best and cannot be based on historical data. Software is generally application-specific and reliability parameters associated with it cannot be estimated in the same manner as hardware. Therefore, another approach shall be used for the assessment of software's contributions to system risk…"*
    - Joint Software Systems Safety Engineering Handbook (JSSSEH), Section 4.2.1.8.2 – *"Traditionally, and for the purpose of being conservative, software errors in fault trees must be set to a value of one (1) where no supporting analysis or assurance rationale is provided."*
  - For this reason, typically take a conservative approach to assessing software failures in FTAs as "always failure" or failure probability set to 1

# Background – Software Control Categories (MIL-STD-882E, Table IV)

- Five SCC Levels:
  - 1 – Autonomous (AT)
  - 2 – Semi- Autonomous (SAT)
  - 3 - Redundant Fault Tolerant (RFT)
  - 4 – Influential
  - 5 – No Safety (NSI)

TABLE IV. Software control categories

| SOFTWARE CONTROL CATEGORIES | | |
|---|---|---|
| Level | Name | Description |
| 1 | Autonomous (AT) | • Software functionality that exercises autonomous control authority over potentially safety-significant hardware systems, subsystems, or components without the possibility of predetermined safe detection and intervention by a control entity to preclude the occurrence of a mishap or hazard. *(This definition includes complex system/software functionality with multiple subsystems, interacting parallel processors, multiple interfaces, and safety-critical functions that are time critical.)* |
| 2 | Semi-Autonomous (SAT) | • Software functionality that exercises control authority over potentially safety-significant hardware systems, subsystems, or components, allowing time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap or hazard. *(This definition includes the control of moderately complex system/software functionality, no parallel processing, or few interfaces, but other safety systems/mechanisms can partially mitigate. System and software fault detection and annunciation notifies the control entity of the need for required safety actions.)*<br>• Software item that displays safety-significant information requiring immediate operator entity to execute a predetermined action for mitigation or control over a mishap or hazard. Software exception, failure, fault, or delay will allow, or fail to prevent, mishap occurrence. *(This definition assumes that the safety-critical display information may be time-critical, but the time available does not exceed the time required for adequate control entity response and hazard control.)* |
| 3 | Redundant Fault Tolerant (RFT) | • Software functionality that issues commands over safety-significant hardware systems, subsystems, or components requiring a control entity to complete the command function. The system detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition. *(This definition assumes that there is adequate fault detection, annunciation, tolerance, and system recovery to prevent the hazard occurrence if software fails, malfunctions, or degrades. There are redundant sources of safety-significant information, and mitigating functionality can respond within any time-critical period.)*<br>• Software that generates information of a safety-critical nature used to make critical decisions. The system includes several redundant, independent fault tolerant mechanisms for each hazardous condition, detection and display. |
| 4 | Influential | • Software generates information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap. |
| 5 | No Safety Impact (NSI) | • Software functionality that does not possess command or control authority over safety-significant hardware systems, subsystems, or components and does not provide safety-significant information. Software does not provide safety-significant or time sensitive data or information that requires control entity interaction. Software does not transport or resolve communication of safety-significant or time sensitive data. |

# Background – SCC and Software Criticality Indices (SwCIs) (MIL-STD-882E, Table V)

- Five SwCIs Levels for performing Level of Rigor (LOR)
  - SwCI 1 – Analysis of requirements, architecture, design and code; conduct in-depth safety-specific testing
  - SwCI 2 – Analysis of requirements, architecture, design; conduct in-depth safety-specific testing
  - SwCI 3 - Analysis of requirements and architecture; conduct in-depth safety-specific testing
  - SwCI 4 – Conduct safety-specific testing
  - SwCI 5 – Not Safety

TABLE V. Software safety criticality matrix

| SOFTWARE SAFETY CRITICALITY MATRIX | | | | |
|---|---|---|---|---|
| | SEVERITY CATEGORY | | | |
| SOFTWARE CONTROL CATEGORY | Catastrophic (1) | Critical (2) | Marginal (3) | Negligible (4) |
| 1 | SwCI 1 | SwCI 1 | SwCI 3 | SwCI 4 |
| 2 | SwCI 1 | SwCI 2 | SwCI 3 | SwCI 4 |
| 3 | SwCI 2 | SwCI 3 | SwCI 4 | SwCI 4 |
| 4 | SwCI 3 | SwCI 4 | SwCI 4 | SwCI 4 |
| 5 | SwCI 5 | SwCI 5 | SwCI 5 | SwCI 5 |

| SwCI | Level of Rigor Tasks |
|---|---|
| SwCI 1 | Program shall perform analysis of requirements, architecture, design, and code; and conduct in-depth safety-specific testing. |
| SwCI 2 | Program shall perform analysis of requirements, architecture, and design; and conduct in-depth safety-specific testing. |
| SwCI 3 | Program shall perform analysis of requirements and architecture; and conduct in-depth safety-specific testing. |
| SwCI 4 | Program shall conduct safety-specific testing. |
| SwCI 5 | Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required. |

NOTE: Consult the Joint Software Systems Safety Engineering Handbook and AOP 52 for additional guidance on how to conduct required software analyses.

# Hypothesis – With Software Failures set to 1

- If the top event failure probability is 1, this function is likely **SCC 1 - Autonomous (AT)**
- If the top event failure probability is 1e-3 or less, then there are still hardware and/or operator influences on the function, so the SCC is likely **SCC 2 - Semi-Autonomous (SAT)** or lower
- If the top event failure probability is a very small number [less than 1e-6], then the SCC is likely **3 - Redundant Fault Tolerant (RFT); 4 - Influential, or even 5 - No Safety Impact (NSI)**

# Overall Assumptions

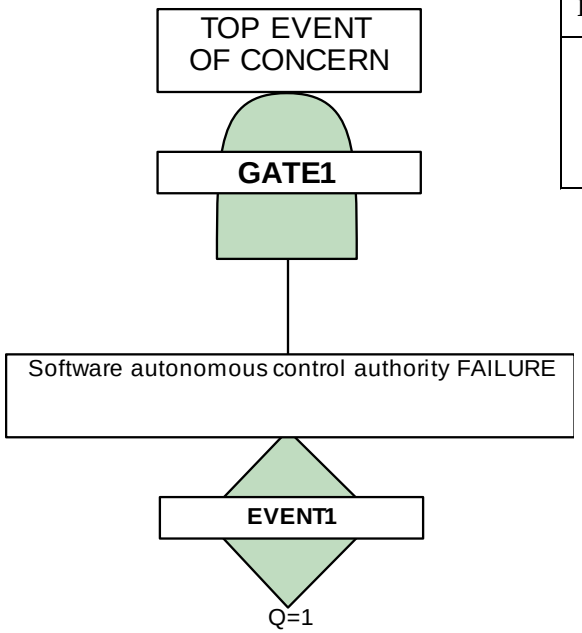
- Nominal failure probability values
  - Hardware failures = 1e-4
  - Operator errors = 1e-3

- DOES NOT assess or construct safety interlocks in the FTA examples that are considered "software interlocks."
  - JSSSEH states "safety interlocks can be either hardware or software oriented. As an example, a hardware safety interlock would be a key switch that controls a safe/arm switch. Software interlocks generally require the presence of two or more software signals from independent sources to implement a particular function. Examples of software interlocks are checks and flags, firewalls, come-from programming techniques, and bit combinations."
  - Modeling software interlocks in quantitative FTAs would be another excellent paper topic for a future conference, but are excluded from this presentation

# SCC Assumptions

- Following slides and tables breakdown each SCC definition (SCCs 1-5) by software failure description, hardware failure description, and operator failure description
  - This will form the basis for the Fault Tree model for each SCC level
  - If the column contains “Not Applicable (N/A)” then this failure mode is assumed not present in the SCC definition
  - The last column includes the number of events included in the cut set
  - Text in **BOLD AND UNDERLINED** indicate added descriptors to convert the SCC definitions into failure descriptions to model as failure events in the sample FTAs

# FTA Examples – SCC 1




| SCC Level | Name | Software Failure Description | Hardware Failure Description | Operator Failure Description | # of Events (in Cut Sets) |
|---|---|---|---|---|---|
| 1 | Autonomous (AT) | Software autonomous control authority **FAILURE** | N/A – no possibility of predetermined safe detection and intervention by a control entity | N/A - no possibility of predetermined safe detection and intervention by a control entity | 1 |

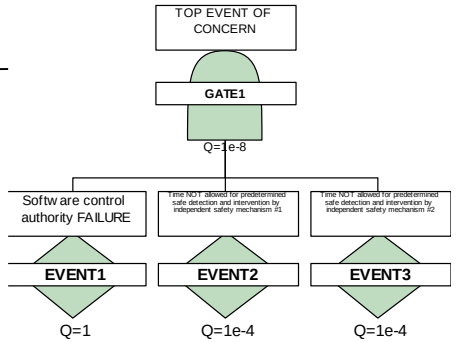
| EVENT1 - Software Failure Probability (Q – Unavailability) | GATE1 – Top Event Probability (Q – Unavailability) | SCC Determination Trend | SCC Rationale |
|---|---|---|---|
| 1 (always fail) | 1 (always fail) | 1 | Equal failure probability values show Top Event is directly dependent on the Software Functionality |
| 0.10 | 0.10 | | |
| 1e-2 | 1e-2 | | |
| 1e-3 | 1e-3 | | |
| 1e-6 | 1e-6 | | |
| 1e-9 | 1e-9 | | |

# FTA Examples – SCC 2 – Two options

| | SCC Level | Name | Software Failure Description | Hardware Failure Description | Operator Failure Description | # of Events (in Cut Sets) |
|---|---|---|---|---|---|---|
| Option 1 | 2 | Semi-Autonomous (SAT) | Software control authority **FAILURE** | Time **NOT** allowed for predetermined safe detection and intervention by independent safety mechanisms<br><br>(Assume at least two (2) independent safety mechanisms **FAIL**) | N/A | 3 (or more than 3) |
| Option 2 | 2 | Semi-Autonomous (SAT) | Software **FAILS** to display safety-significant information | N/A | Immediate operator entity **FAILS** to execute a predetermined action for mitigation or control in the appropriate timeframe. | 2 |

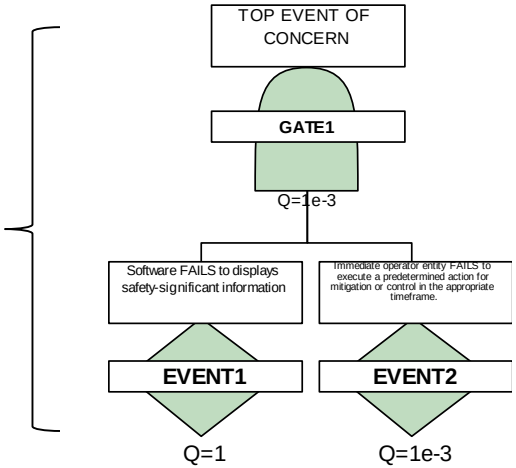
# FTA Examples – SCC 2 – Two options (cont)

Option 1




| EVENT1 - Software Failure Probability | EVENT2 - Mechanism #1 Failure Probability | EVENT3 - Mechanism #2 Failure Probability | GATE1 - Top Event Probability | SCC Determination Trend | SCC Rationale |
|---|---|---|---|---|---|
| 1 (always fail) | 1e-4 | 1e-4 | 1e-8 | 2 | Top event failure probability is the product of the two (2) mechanism failures |
| 0.10 | 1e-4 | 1e-4 | 1e-9 | | |
| 1e-2 | 1e-4 | 1e-4 | 1e-10 | | |
| 1e-3 | 1e-4 | 1e-4 | 1e-11 | | |
| 1e-6 | 1e-4 | 1e-4 | 1e-14 | | |
| 1e-9 | 1e-4 | 1e-4 | 1e-17 | | |

Option 2



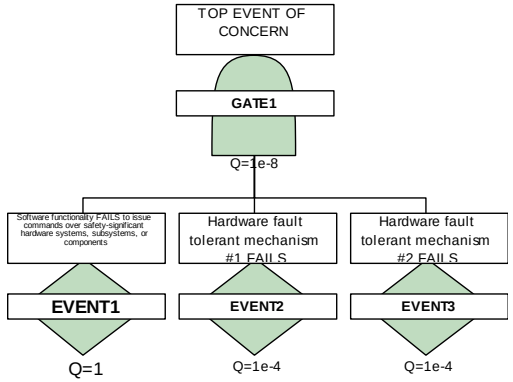

| EVENT1 - Software Failure Probability | EVENT2 - Operator Failure Probability | GATE1 - Top Event Probability | SCC Determination Trend | SCC Rationale |
|---|---|---|---|---|
| 1 (always fail) | 1e-3 | 1e-3 | 2 | Top event failure probability is the failure probability of operator failure |
| 0.10 | 1e-3 | 1e-4 | | |
| 1e-2 | 1e-3 | 1e-5 | | |
| 1e-3 | 1e-3 | 1e-6 | | |
| 1e-6 | 1e-3 | 1e-9 | | |
| 1e-9 | 1e-3 | 1e-12 | | |

# FTA Examples – SCC 3 – Two options

| | SCC Level | Name | Software Failure Description | Hardware Failure Description | Operator Failure Description | # of Events (in Cut Sets) |
|---|---|---|---|---|---|---|
| Option 1 | 3 | Redundant Fault Tolerant (RFT) | Software functionality **FAILS** to issue commands over safety-significant hardware systems, subsystems, or components | At least two (2) hardware fault tolerant mechanisms **FAIL** | Control entity **INADVERTENTLY** completes the command function.<br><br>(Does this count as a failure in the FTA if the software fails to issue appropriate commands?) | 3 or more |
| Option 2 | 3 | Redundant Fault Tolerant (RFT) | Software **FAILS TO** generate information of a safety-critical nature used to make critical decisions | Several (at least three (3)) redundant, independent fault tolerant mechanisms **FAIL** for each hazardous condition, detection and display. | | 4 or more |

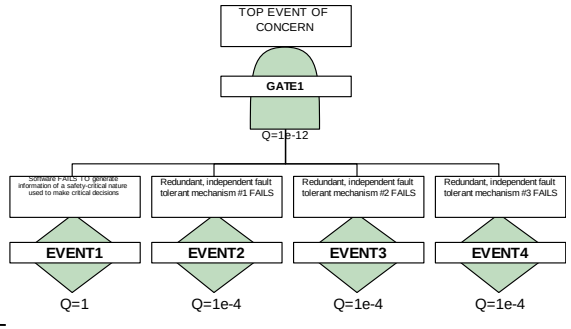
# FTA Examples – SCC 3 – Two options (cont)

Option 1




| EVENT1 - Software Failure Probability | EVENT2 – Hardware Fault Tolerant Mechanism #1 Failure Probability | EVENT3 - Hardware Fault Tolerant Mechanism #2 Failure Probability | GATE1 - Top Event Probability | SCC Determination Trend | SCC Rationale |
|---|---|---|---|---|---|
| 1 (always fail) | 1e-4 | 1e-4 | 1e-8 | 3 | Top event failure probability is the product of the two (2) mechanism failures |
| 0.10 | 1e-4 | 1e-4 | 1e-9 | | |
| 1e-2 | 1e-4 | 1e-4 | 1e-10 | | |
| 1e-3 | 1e-4 | 1e-4 | 1e-11 | | |
| 1e-6 | 1e-4 | 1e-4 | 1e-14 | | |
| 1e-9 | 1e-4 | 1e-4 | 1e-17 | | |

Option 2




| EVENT1 - Software Failure Probability | EVENT2 – Hardware Mechanism #1 Failure Probability | EVENT3 - Hardware Mechanism #2 Failure Probability | EVENT4 - Hardware Mechanism #3 Failure Probability | GATE1 - Top Event Probability | SCC Determination Trend | SCC Rationale |
|---|---|---|---|---|---|---|
| 1 (always fail) | 1e-4 | 1e-4 | 1e-4 | 1e-12 | 3 | Top event failure probability is the product of the three (3) mechanism failures |
| 0.10 | 1e-4 | 1e-4 | 1e-4 | 1e-13 | | |
| 1e-2 | 1e-4 | 1e-4 | 1e-4 | 1e-14 | | |
| 1e-3 | 1e-4 | 1e-4 | 1e-4 | 1e-15 | | |
| 1e-6 | 1e-4 | 1e-4 | 1e-4 | 1e-18 | | |
| 1e-9 | 1e-4 | 1e-4 | 1e-4 | 1e-21 | | |

# FTA Examples – SCC 4

- SCC 4 definition implies:
  - 1) there is no mishap (or FTA top event) if this safety failure occurs and
  - 2) safety-related nature is related to a Marginal / Negligible severity.
- Therefore, there is no need to develop a FTA since there is no mishap / top event if software failure occurs.

| SCC Level | Name | Software Failure Description | Hardware Failure Description | Operator Failure Description | # of Events (in Cut Sets) |
|---|---|---|---|---|---|
| 4 | Influential | Software **FAILS** to generate information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap. | N/A | N/A | No Mishap / Top Event if software failure occurs |

# FTA Examples – SCC 5

- Based on the SCC 5 definition:
  - No need to develop a FTA since software functionality is not related to any safety significant function and there is no mishap / top event if software failure occurs

| SCC Level | Name | Software Failure Description | Hardware Failure Description | Operator Failure Description | # of Events (in Cut Sets) |
|---|---|---|---|---|---|
| 5 | No Safety Impact (NSI) | N/A | N/A | N/A | Software functionality **NOT** related to any safety significant function. No Mishap / Top Event if software failure occurs |

# Initial Results

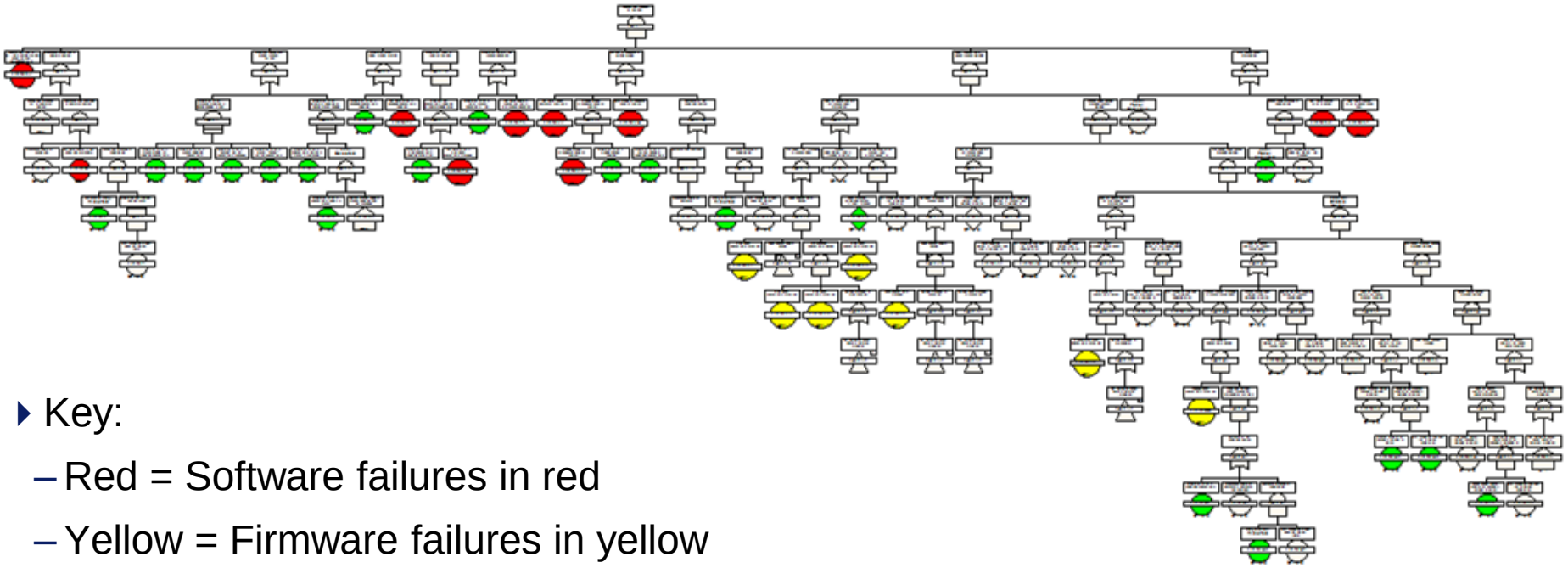
- While the sensitivity analyses did not expose any revelations, they did provide the following range of failure probabilities
- Probability ranges could be used as a “rule of thumb” during SCC assessment when software failure probabilities are set to 1

| Software Failure Probability | GATE1 - Top Event Probability Range | SCC Determination Trend |
|---|---|---|
| 1 (always fail) | 1 | 1 – Autonomous (AT) |
| 1 (always fail) | 1e-3 to 1 | *(further SCC determination needed to decide if SCC 1 or SCC 2)* |
| 1 (always fail) | 1e-3 to 1e-8 | 2 - Semi-Autonomous (SAT) |
| 1 (always fail) | 1e-8 to 1e-12 | 3 - Redundant Fault Tolerant (RFT) |
| 1 (always fail) | N/A – no mishap/top event | 4 – Influential (I) |
| N/A – no software functionality | N/A – no mishap/top event | 5 - No Safety Impact (NSI) |

# Case Study to Verify Initial FTA Examples

- Using a more realistic model, a sample fault tree of a fictitious launching system with a variety of fault events
- Top event is “Inadvertent Launch”

# Case Study to Verify Initial FTA Examples (cont)

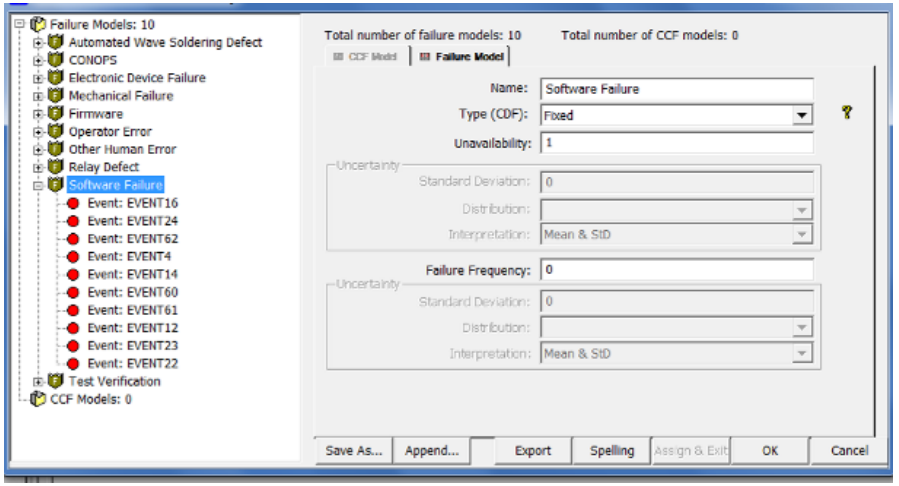


- Key:
  - Red = Software failures in red
  - Yellow = Firmware failures in yellow
  - Green = Operator failures in green
  - Default White = Other failures (e.g., mechanical failures)

# Case Study to Verify Initial FTA Examples (cont)

- For this sample fault tree, a Fault Tree application is used including the use of a failure model library to easily and quickly change failure probabilities to perform sensitivity analyses of software failures

- All other failure probabilities for the various failure models are kept at their assumed failure levels
  - Hardware failures = 1e-4
  - Operator errors = 1e-3



# Case Study Results

| Software Failure | Firmware Failure | Top Event Probability – Inadvertent Launch | SCC Determination Trend | SCC Rationale |
|---|---|---|---|---|
| 1 (always fail) | 1 | 1.01e-6 | 2-SAT | Top Event probability is between 1e-3 to 1e-8 as listed in Table on Slide 17 |
| 0.10 | 1 | 3.77e-12 | 3-RFT | Top Event probability is close to 1e-13 as listed in Option 2 Table on Slide 14 when Software Failure is 0.1. |
| 1e-2 | 1 | 5.18e-18 | 3-RFT | Top Event probability is close to 1e-14 as listed in Option 2 Table on Slide 14 when Software Failure is 1e-2. |
| 1e-3 | 1 | 4.87e-23 | 3-RFT | Top Event probability is close to 1e-15 as listed in Option 2 Table on Slide 14 when Software Failure is 1e-3. |
| 1e-6 | 1 | 1.22e-29 | 3-RFT | Top Event probability is close to 1e-18 as listed in Option 2 Table on Slide 14 when Software Failure is 1e-6. |
| 1e-9 | 1 | 1.01e-32 | 3-RFT | Top Event probability is close to 1e-21 as listed in Option 2 Table on Slide 14 when Software Failure is 1e-9. |

Cut Set Report (partial)

| Gate Name | No | Cut Set | Set Unavailability |
|---|---|---|---|
| GATE1 | 1 | EVENT8 ::EVENT12 ::EVENT14 ::EVENT16 ::EVENT22 ::EVENT32 ::EVENT25 ::EVENT26 ::EVENT27 ::EVENT28 ::EVENT36 ::EVENT42 ::EVENT44 ::EVENT61 ::EVENT62 | 1e-6 |
| GATE1 | 2 | EVENT8 ::EVENT12 ::EVENT14 ::EVENT16 ::EVENT24 ::EVENT32 ::EVENT25 ::EVENT26 ::EVENT27 ::EVENT28 ::EVENT36 ::EVENT42 ::EVENT44 ::EVENT60 ::EVENT62 | 1e-6 |
| GATE1 | 3 | EVENT8 ::EVENT12 ::EVENT14 ::EVENT16 ::EVENT22 ::EVENT32 ::EVENT25 ::EVENT26 ::EVENT27 ::EVENT28 ::EVENT36 ::EVENT42 ::EVENT44 ::EVENT60 ::EVENT62 | 1e-6 |

***Results – SCC assessed as 2-SAT (conservative assessment); additional verification could be provided by the Cut Set Report, Functional Hazard Analysis (FHA) and System Safety Working Group (SSWG)***

# Conclusion

- Quantitative FTAs can be used to verify the SCC level
- Rough approximation of the top event failure probability ranges to determine the SCC when software failure probabilities are set to 1
  - Further refinement may be necessary as additional case studies are performed for more complex and realistic fault tree models
  - Safety interlocks that are software interlocks (e.g., checks and flags, firewalls, bit combinations) are excluded from this paper - Modeling software interlocks could impact the conclusions of this paper
- Cut set reports can also be used to assess the combinations of failures required to cause the top event

# Recommendations

- If it is desired to reduce the SCC, the System Safety Practitioner should closely exam the results of FTAs and cut set reports
- Consider design enhancements that would add additional hardware controls, operator verification, and/or design features into the system functionality to reduce software autonomy
- Lowering the SCC could reduce the SwCI and LOR tasking required, which may save program resources when it comes to performing complex LOR tasks, such as code analysis
  - Understood that in some cases, operational and design requirements will dictate that software functionality for safety critical operations can only be autonomous
  - Inserting additional safety interlocks or operator interventions may not be feasible

# References


- MIL-STD-882E, “Department of Defense Standard Practice System Safety,” May 11, 2012.
- Department of Defense (DoD) Joint Software Systems Safety Engineering Handbook (JSSSEH), August 27, 2010.

# Questions?

**Robert E. Smith, CSP**
*Senior Lead Engineer*

Booz | Allen | Hamilton

Booz Allen Hamilton
1550 Crystal Dr, Suite 1100
Arlington, VA 22202
Tel (703) 412-7661
smith_bob@bah.com