2024 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
DIGITAL ENGINEERING / SYSTEMS ENGINEERING TECHNICAL SESSION
AUG. 13-15, 2024 - NOVI, MICHIGAN

AUTOMATED SYSTEM/SOFTWARE SAFETY ANALYSIS USING MODEL-BASED SYSTEMS ENGINEERING

Gregory P. Czerniak[1] and Rodolfo Proenza[1]

[1]Software Engineering Center, DEVCOM US Army Ground Vehicle Systems Center, Warren, MI

ABSTRACT

We present a data model for performing system and software safety in a way that is compatible with Digital Engineering and Model-Based Systems Engineering. This requires imposing structure into the system/software safety process that allows for interfacing with other data models and for other data models to interface with the system safety model. Doing this allows for high amounts of traceability, and it has allowed for Safety Assessment Reports for small projects to be generated and analyzed in weeks.

Citation: G. Czerniak, R. Proenza, "Automated System/Software Safety Analysis Using Model-Based Systems Engineering," In *Proceedings of the 2024 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*, NDIA, Novi, MI, Aug. 13-15, 2024 .

1. INTRODUCTION

Our goal was to produce a computer system capable of modeling military system and software safety using a digital engineering approach. The military uses MIL-STD-882E [1] and the Joint Software Systems Safety Engineering Handbook (JSSSEH) [2] to define system safety for ground vehicles. However, in order to implement digital engineering and model-based systems engineering against system safety, it is required to provide traceability against Authoritative Source of Truth (ASoT) per the Department of Defense (DoD) Digital Engineering Fundamentals [3], which requires applying more rigid structure to the problem space. We implemented and now describe such an approach toward making software safety compatible with digital engineering.

2. HAZARD DEFINITION

According to the DoD Digital Engineering Fundamentals [3], organizations should develop models for performing engineering work. MIL-STD-882E defines a hazard as "A real or potential condition that could lead to an unplanned event or series of events (i.e. mishap) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment." While this is a useful definition, it lacks the structure of a data model. This section will describe a structured way to represent a hazard.

In "Hazard Analysis Techniques for System Safety" [4], the author defines a hazard as three components:

1. Threat Outcome: A bad event that we want to prevent (a mishap)
2. Hazard Source: Usually a source of energy (kinetic, chemical, etc).
3. Initiating Mechanism(s): Events that must happen simultaneously for the hazard to happen.

An example of a hazard in this form is:

- Threat Outcome: A person is run over
- Hazard Source: by a large moving vehicle
- Initiating Mechanisms
  - because a person is in front of the vehicle
  - and the vehicle does not stop

By defining hazards in this manner, the hazard can naturally be transformed into a fault tree, as Figure 1 illustrates.

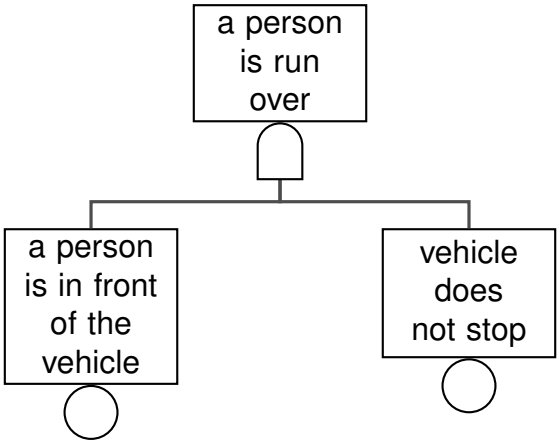

Figure 1: Sample fault tree

A fault tree is advantageous because the events in a fault tree can be traced to both mitigations and failures in system/software functions. Fault tree boxes are reusable throughout multiple hazards.

Per the JSSSEH, software mitigation failures are automatically assigned a probability of 1, since there is currently no standard way to quantify the probability of software failure. Instead, software must be assigned a Software Control Category (SCC) and designed to the Level of Rigor (LOR) that corresponds to its SCC and severity.

With the Initiating Mechanisms serving as the primary branches of the tree, the user can then populate them further in as much detail required to describe the circumstances that can lead up to a hazard.

The "Hazard Source" generally determines the hazard's severity. For example, a person getting run over by a small radio-controlled car may injure a shin, but a person run over by a military vehice is potentially fatal.

3. MITIGATIONS

According to HCRQ Inc.'s "Fault Tree Analyses - When to Accept, When to Reject," [5] fault trees shall represent potential failures of the mitigations, and they shall have traceability to the relevant safety requirements.

In this system, mitigations may be marked as hardware, software, procedural, Personal Protective Equipment (PPE), post-mishap recovery equipment (such as an environmental spill kit), training or signage. This annotation is used later to provide approximations for SCCs, and it can be used to help design products according to the MIL-STD-882E design order of precedence.

Once defined, mitigations may be traced to a box in the above-mentioned hazard fault trees. Once traced, boxes traced as a potential failure of a mitigation will be marked with a letter "M", satisfying the HCRQ requirement of including mitigation failures in fault trees.

Mitigations can also be assigned as either test-specific or not, and fault tree boxes traced

as the failure of a test-specific mitigation will be marked with a letter "T". This satisfies a requirement in MIL-STD-882E Task 301 to mark test or other event-specific mitigation masures necessary to reduce risks.

The mitigation types are also used later in cut-set analysis to approximate the SCC of a given safety-significant function for a given scenario.

4. FUNCTIONAL HAZARD ANALYSIS

The system uses a hierarchical structure to represent the system in a logical (not geometric) structure. System components may be registered either as hardware, software, human, or Hazardous Materials (HAZMAT).

The system provides the ability to import system components from a SysML model in XML format. When this is done, system components will maintain the xmi:id of the given SysML component, providing more traceability against the ASoT.

System components may have any number of functions represented underneath. These functions have the following attributes:

- Dependent Functions: These are represented as a "if X function fails, then this provokes Y failure in this function." This can be used to model failures such as a device losing electrical power or a computer operating system failing to give a task enough time to process.
- Requirements: Functions may be traced to any number of requirements. These are stored only as requirement ID strings, which can then be used by other tools to trace against a requirements database (an ASoT).
- Failures: For the failure types listed later, any number of fault tree boxes can be linked to a failure type. This is vital, since it establishes traceability from the functional breakdown to any hazard that uses that fault tree box in its hazard definition. Fault tree boxes that are also marked as mitigations also receive traceability to a linked function's requirements, satisfying the HCRQ requirement mentioned in "Mitigations."

The current failure types come from "Hazard Analysis Techniques for System Safety" [4] and the military standards [1] [2], and include:

- Function is unavailable
- Malfunction
- Out of Sequence
- Too early
- Too late
- Outside of defined window
- Unable to stop
- Receives erroneous data
- Sends erroneous data
- Conflicting data or information
- Signage ignored

Once function failures are traced to fault tree boxes, it is possible to perform cut-set analysis on the fault trees to determine how close system functions are to the hazard.

5. CUT-SET ANALYSIS

Safety analysis is a driver of requirements and a driver of design. Arguably the most important aspect of this is the assignment of a Software Criticality Index (SwCI) to safety-significant functions, as that will determine the LOR required for each function. The LOR drives the amount of activities required to test and verify each safety-significant function as per the JSSSEH Implementation Guide [6].

Toward that end, the system performs analysis in order to provide approximations of SwCI levels for functions specified in the database. In order to do this, the system performs a cut-set analysis on all fault trees representing hazards. If any cut-sets include the failure mode of a function in the function database, it gets aggregated under that function. Each cut-set is then individually analyzed to determine how close that function's failure is to the hazard being able to happen.

Description	Probability
a person is in front of the vehicle	1
drive-by-wire software malfunctions	SCC1
software-based emergency stop fails	SCC1

Table 1: Cut-Set with software-based emergency stop

In Table 1, there are no hardware interlocks or human actions that can prevent the hazard, since the emergency stop is implemented by software. As a result, both software components in the cut-set are given SCC 1. The event of a person being in front of the vehicle is a house event, so it has a probability of 1 for argument's sake.

Description	Probability
a person is in front of the vehicle	1
drive-by-wire software malfunctions	SCC2
hardware-based emergency stop fails	3e-7

Table 2: Cut-Set with hardware-based emergency stop

In Table 2, there is a hardware-based emergency stop, which increases the software control category for "drive-by-wire software malfunctions" to SCC 2.

Currently, the system performs a simple count of hardware interlocks and human actions in order to determine the SCC. This approach has produced sensible results in all projects currently used by this system, but in the future we will add the ability to either manually assign SCCs to functions or to provide different logic.

After evaluating all cut-sets related to a function, the system determines the highest LOR found and reports it as the official LOR for that function.

Automating this process with clear rules eliminates a lot of debate and guess-work. It also makes it faster to evaluate alternative design choices (partitioning, separation of computers, implementation of mitigations) to find a solution to problems related to mitigation.

6. REPORTING

The system can report all data necessary to produce a Safety Assessment Report (SAR). The data gets exported as a set of LaTeX documents that can then be integrated into a full report.

Per the military standards [1] [2], the system produces a large list of tables and information that is required in a SAR. This includes, but is not limited to:

- Detailed records of system and subsystem-level hazards
- Risk matrix
- Software Control Category and Software Criticality Index matrices
- Functional Hazard Analysis
- Fault Trees
- List of probabilites used in fault trees with citations
- List of Computer Software Configuration Items, Computer Software Units, and functional interfaces
- Safety-significant requirements and their traceability
- List of HAZMAT

The system is also capable of producing an Army Deliberate Risk Assessement Worksheet (DRAW). While this worksheet uses a different risk matrix than MIL-STD-882E, representing risks in a one-to-one mapping from the Army matrix to the DoD matrix produces sensible results.

7. PROJECTS

This system has seen active use in multiple projects within the Ground Vehicle Systems Center (GVSC) Software Engineering Center (SEC), including an experiment for an optionally-manned vehicle and a test for an electrified vehicle. In both projects, this system was able to produce a detailed SAR and DRAW within weeks. Both SARs were accepted by the accreditor to proceed with these tests.

The system currently operates as a single-user single-project prototype. The eventual goal is to produce a multi-user multi-project version capable of handling safety across the GVSC SEC. The new version would also potentially support processing hazards using different standards such as ISO 26262 and DO-178C.

8. INTERFACE

The system provides access to its database, which allows for other systems to use it as an ASoT. These ASoTs could theoretically provide bi-directional traceability from hazards all the way to individual lines of computer code if the system were interfaced with a tool capable of tracing lines of code to requirements.

9. CONCLUSION

By incorporating a strict structure to system safety, this model facilitates digital engineering by providing traceability to ASoTs in SysML and requirements. It also allows for traceability from hazards to safety-significant functions. Imposing structure into the system safety process also produces new ASoTs that can then be interfaced with other tools, such as safety-significant functions, hazards, and fault tree boxes.

In addition, imposing structure on the system safety process seems to clarify the process, making it more efficient to input data and perform analyses.

10. REFERENCES

References

[1] Department of Defense, “Department of defense standard practice: System safety,” Wright-Patterson Air Force Base, OH, Standard, Sep. 2023. [Online]. Available: https://safety.army.mil/Portals/0/Documents/ON-DUTY/SYSTEMSAFETY/Standard/MIL-STD-882E-change-1.pdf.

[2] Department of Defense, “Joint software systems safety engineering handbook,” Indian Head, MD, Standard, Aug. 2010. [Online]. Available: https://www.acqnotes.com/Attachments/Joint-SW-Systems-Safety-Engineering-Handbook.pdf.

[3] Department of Defense, “Department of defense (dod) digital engineering fundamentals,” Tech. Rep. [Online]. Available: https://ac.cto.mil/wp-content/uploads/2022/03/DE-Fundamentals-2022.pdf.

[4] C. A. Ericson, *Hazard analysis techniques for system safety*. Hoboken, NJ: Wiley, 2016.

[5] HCRQ, Inc., “Fault tree analyses - when to accept, when to reject,” Williamsburg, VA, Tech. Rep., 2018.

[6] Joint Services - Software Safety Authorities, “Software system safety: Implementation process and tasks supporting MIL-STD-882E,” Standard, Oct. 2017.