ExCuSe – A Method for the Model-Based Safety Assessment of Simulink and Stateflow Models

MATLAB Expo 2018 | 2018-06-26 | München
Julian Rhein
Outline

- Introduction
- Property Proving
- Application to Safety Assessment
- Implementation
- Demo
- Summary & Outlook
Introduction

- Functional safety assessment is an integral part of software and systems development according to SAE ARP 4754A
- Task of functional safety assessment
  - Establish relations between component faults and system failure conditions
  - Validation & Verification of safety requirements

Typical methods:
- Failure Modes and Effects Analysis (FMEA)
  - Determine system level effects from (single) component faults
- Fault Tree Analysis (FTA)
  - Find possible causes for system failure conditions
Our goal: Partially automated generation of safety assessment artefacts from annotated models:

- Executable model of a technical system
- Formalized requirements
- hierarchically ordered
- component-oriented
- Signal flow models
- Physical models
- nominal behavior & failure behavior

FMEA
FTA
Conventional vs. Model-Based Safety Assessment

System Architecture

Reliability Models

Reliability Metrics

Conventional

Model-based

Automatic

Design Model

Safety Assessment Model

Institute of Flight System Dynamics

Julian Rhein – Model-based Safety Assessment
The ExCuSe Tool

System Architecture

Reliability Models

Reliability Metrics

MATLAB

Simulink

Design Model

Semi-Automatic

Safety Assessment Model

ExCuSe

Automatic

Automatic
Advantages of MBSA

- Support of analysts during repetitive, error-prone tasks
- Provides methods to system designers and software engineers to evaluate their designs prior to formal safety assessment
- Enables modularity of safety assessment and reusability of artefacts
- Additional validation of manual assessment results
Introduction to Property Proving

- Traditionally used in software verification
- Formal approach to prove that a property is satisfied or violated by a system
- Properties are disproved by counterexamples
- Infinite state and continuous systems are treated by inductive proving and SMT satisfiability
- Powerful free/open-source solvers available

Testing:

Model $M$ \rightarrow Properties $\mathcal{P}$

Property proving:

Satisfied \rightarrow Model $M$ \leftarrow Properties $\mathcal{P}$

Violated \leftarrow Counter example $u^*$

Satisfied \leftarrow Test vectors $u$
Safety Assessment as Property Proving Problem

**Formal concept**
- Fault injection
  \[ M(u) \Rightarrow M^*(u, f), f \in F \]
- Cut-sets computation
  \[ CS := \{ cs_i \in F | M^*(u^*, f^*) \not\in AG \mathcal{P} \land \exists j s.t. f_{i,j} \} \]
- Minimal cut-sets
  \[ MCS := \{ cs \in CS | \not\exists cs' \in CS \land cs' \subset cs \} \]
- Verification: \(|F| = 0, |\mathcal{P}| > 0\)
- FMEA: \(|F| = 1, |\mathcal{P}| > 0\)
- FTA: \(|F| > 0, |\mathcal{P}| = 1\)

**Intuitive explanation**
- Extension of the model by additional inputs to trigger fault events
- Computation of sets (combinations) of failures, which lead to a requirements violation from the counterexamples
- Minimal combinations of failures, i.e. failure configurations that are necessary for the occurrence of system failures
- Verify that the system fulfills all requirements in the failure free case
- Determine all possible effects (i.e. violation of requirements) of single failures
- Determine all possible causes of single a single effect (i.e. its MCSs)
Fault Injection

- Extension of the model by additional component failure models and additional inputs to trigger the activation and deactivation of component faults

\[
\begin{align*}
  & f \\
  & u \\
  \text{Model } M^* \\
  y & \rightarrow \text{Properties } P \\
  \end{align*}
\]

\[
\begin{align*}
  & f_i \\
  & u \\
  \text{Component Model } M_{i,Nominal}^* \\
  \end{align*}
\]

\[
\begin{align*}
  & f_i \\
  & u \\
  \text{Component Model } M_{i,Faulty}^* \\
  \end{align*}
\]

Satisfied

Violated
Implementation

- Implementation in Simulink/Stateflow by fault injection interface

**Idea:**
- Extend the model by a nondeterministic layer
- Provide modelling facilities for failure logic modeling or failure injection
- Provide interface for automatic analysis

- Share failure flow information between components without requiring additional signals
- Allows common cause modeling
- Predefined and custom fault models
- Automatic cut-set analysis based on the Simulink Design Verifier property proving function
Generic Failure Models

Fault Injector
ID: Failure

Uncertain parameters
Custom Failure Models

- Enable definition of arbitrary, user-defined fault models
- Uncertain parameters can be modelled using special source blocks

Custom failure models

Fault Injector
ID: Failure
Probabilistic Attributes

- Distributions of the component lifetime and repair time
- Similar to basic event models in fault tree analysis
- Common models are built-in:
  - Exponential distribution
  - Weibull distribution
  - Periodic test
- Custom models can be specified as custom expressions or histograms
Model-Based Safety Assessment with ExCuSe

Start

Design model

Inject failures to design model

Create failure logic model from design model

Using the ExCuSe blockset for failure modelling

Formalize functional requirements

Using the SLDV blockset for modelling of temporal properties

Analyze model

Minimal combinations of failures that cause a requirements violation

Evaluate safety requirements

Evaluate quantitative and probabilistic safety requirements

Modify system design

Refine failure modelling

Satisfactory?

Y

Plausible?

Y

N

N

End

Y

N

N

Plausible?

Satisfactory?

N

Y

N

Y
Demo: Ground Spoiler Deployment Logic

- Example problem: A320 SEC decision logic for ground spoiler deployment

- Hazards:
  - Inadvertent full/partial spoiler deflection inflight
  - Missing full/partial spoiler deflection on ground

- Question: Which combinations of sensor failures can potentially cause the hazards

[Airbus A320 AMM]
Demo: Ground Spoiler Deployment Logic
Summary

- Model checking provides a powerful method for model-based safety assessment
- Cut-set analysis can be expressed as property proving problem
  - Performance enhancement by incremental search
  - Anytime approximation of probability boundaries
- Successful integration in Simulink/Stateflow

Outlook

- Scalability considerations
- Creation of structured fault trees from the minimal cut-sets
- Using structural analysis to obtain initial guess of the minimal cut-sets
- Extension to undirected models
Thank you for your attention!

Questions?
Contact

Julian Rhein  
Research Associate

Lehrstuhl für Flugsystemdynamik /  
Institute of Flight System Dynamics  
Technische Universität München  
Boltzmannstraße 15  
D-85748 Garching  
Germany

Phone: +49 (89) 289-16051  
Fax: +49 (89) 289-16058  
Email: julian.rhein@tum.de