FAME System Requirement Assessment Tool

User Tutorial for the FAME System Requirement Assessment Tool

May 2, 2014

1.0 Summary

The FAME System Requirement Assessment Tool is a web-based tool that captures reliability for a system model that can be simulated and is subject to component faults.

2.0 Purpose

The DARPA Adaptive Vehicle Make program aims at developing a design flow that lets system designers adapt their designs through a tightly integrated build-test-modify loop with multiple points of feedback. In order for this workflow to yield reliable system designs, it is essential for FANG designers to have the ability to analyze faults, fault propagation, and system-level impact.

The FAME (Fault Augmented Model Extension) System Requirement Assessment Tool provides this functionality. The tool supports analyses of system reliability and performance under both continuous wear and catastrophic failure of critical system components. It also scores design configurations according to reliability metrics and provides feedback to the designer about preferable choices of components or design configurations.

"Reliability" describes the ability of a system to operate while meeting all requirements for a specified period of time or number of missions. Reliability is often quantified in terms of likelihood of failure, e.g. Mean Time to Failure (MTTF), Mean Time between Failure (MTBF), and Failures in Time (FIT) which captures system unreliability. The FAME System Requirement Assessment Tool captures reliability using Overall Probability of Mission Failure, which is equivalent to NASA's System Failure Probability quantity.

3.0 Background

3.1 Model

The example model currently used is driveLine and the test bench used is no_controls. A description of the system design can be found here. The model is expressed in CyPhy/Modelica files; CyPhy and Dymola can be used to explore the model and its components in detail. Please refer to CyPhy documentation for help on importing and exploring model (xme) files.

3.2 Installation

This is a web-based tool. No local installation is needed. The tool is available at http://fame-deploy.parc.com:2040/. It has been tested on Chrome, Firefox, Safari, IE10, and IE11.

3.3 Overview of the Tool

The tool allows you, as a system designer or design tester, to explore the consequences of different design choices, and to assess the effects of gradual wear and catastrophic failure of critical components as a result of mission stress. The tool will help you to discover answers to the following questions:

- What System Configurations are most reliable?
- Which Component Failure Modes causes critical performance loss?
- Why is a particular Component Failure Mode critical?
- What Performance Metrics are most at risk?
- How do these factors vary with Number of Missions?

4.0 Getting Started

4.1 Launch the tool

In a supported browser, navigate to http://fame-deploy.parc.com:2040/. The web site initially looks like this:

Figure 1: FAME System Requirement Analysis Tool User Interface

4.2 Open the System Configuration and Component Faults Listings

Click to open scrolling lists of System Configurations and Component Faults.
Drag the scroll handles to scroll through the lists.
Press radio buttons to select a System Configuration and a component subject to fault.

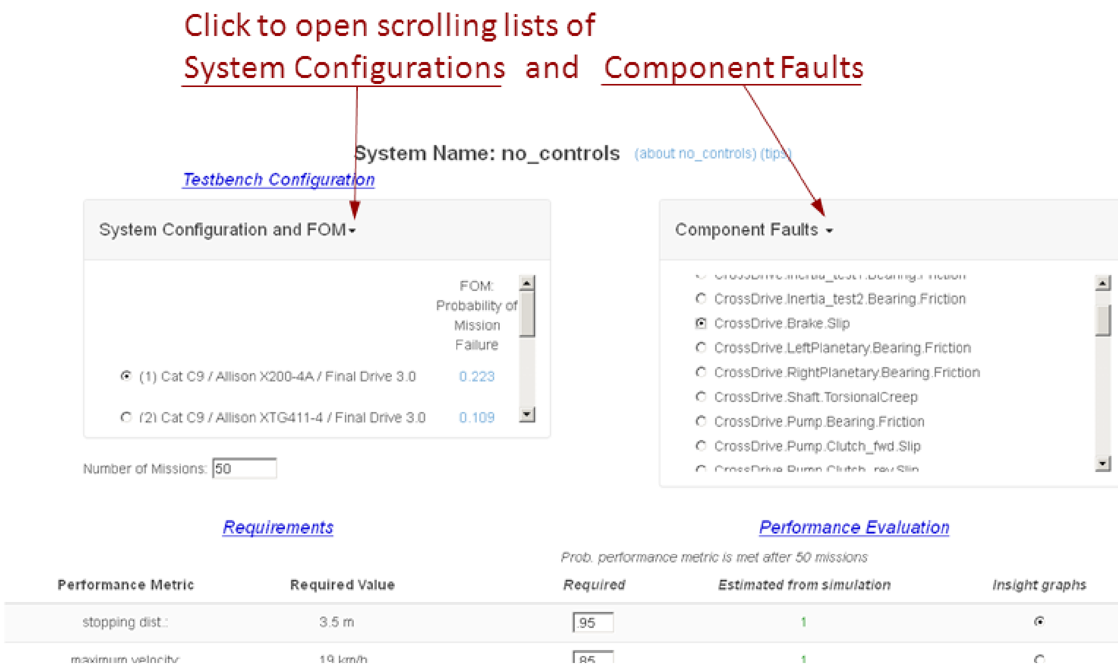

Figure 2: Selecting System Configuration and Component Fault

5.0 Use Case I: Explore the Impact of Particular Component Failure Modes

5.1 Select Desired Options

1. System Configuration
2. Component Fault mode
3. Number of Missions
4. Required Probability for Meeting a Performance Metric
5. Performance Metric to plot Insight Graphs

Key outputs to analyze the impact of particular component failure modes are highlighted in purple.

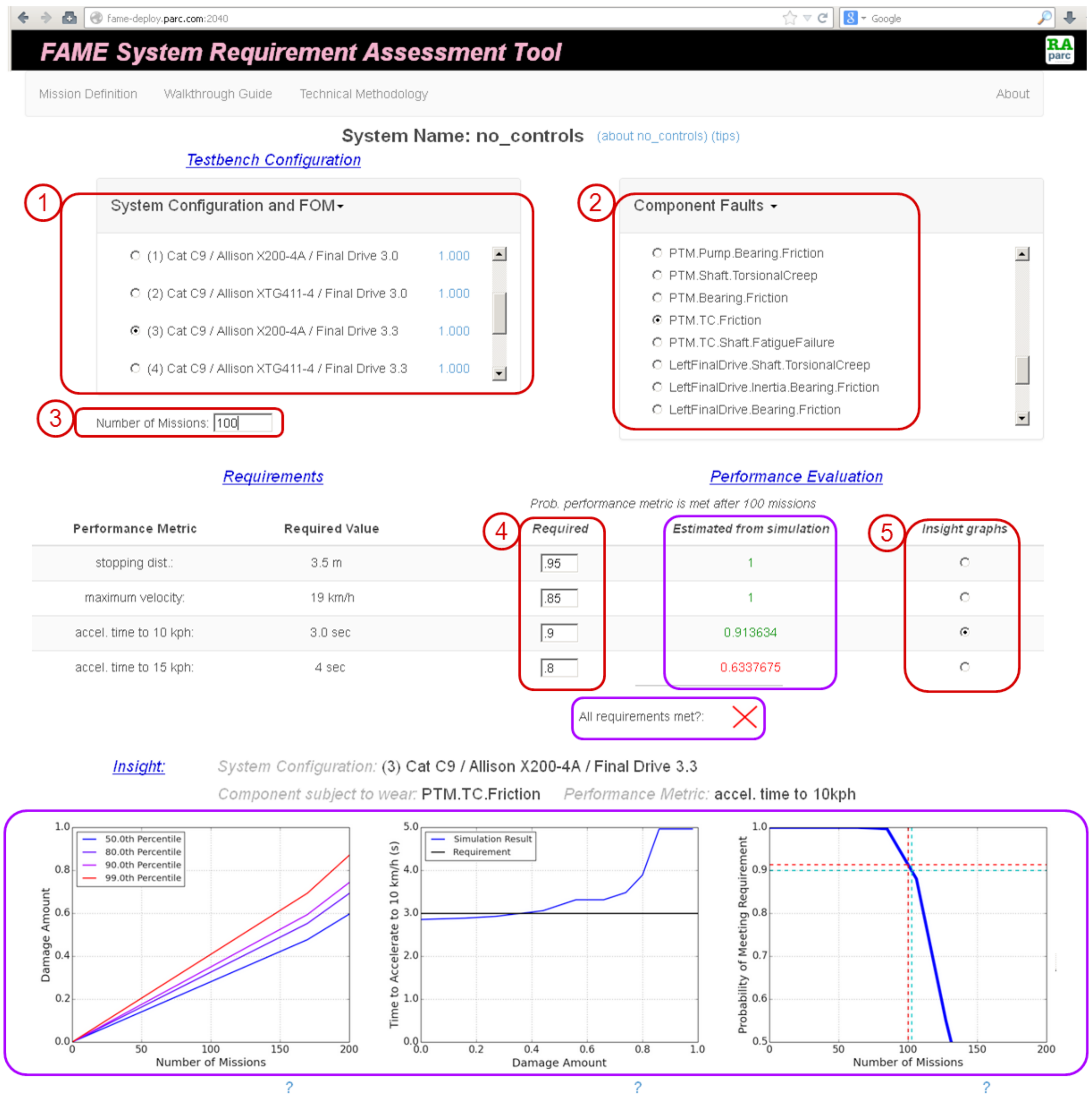

Figure 3: Selecting Desired Options for Use Case I

Note that the faults that significantly affect the performance of each configuration can be unique. Please click on the tips link to determine which faults are relevant to which configurations.

For this tutorial, please select the following options:

1. (3) Cat C9 / Allison X200-4A / Final Drive 3.3
2. PTM.TC.Friction
3. 100 missions
4. [0.95, 0.85, 0.9, 0.8]
5. accel. time to 10 kph

5.2 Key Insights

Given these settings, the Insight graphs at the bottom of the page reveal how component damage reduces probability of mission success.

5.2.1 Critical damage for component failure mode

Critical damage is defined as the minimal damage amount that results in failing any required performance metric. This is determined by interpolating Modelica simulation results. Graphically, critical damage for a particular performance metric is represented by the damage amount where simulated performance intersects the required performance as a function of damage amount (middle graph, Performance Metric vs. Damage Amount).

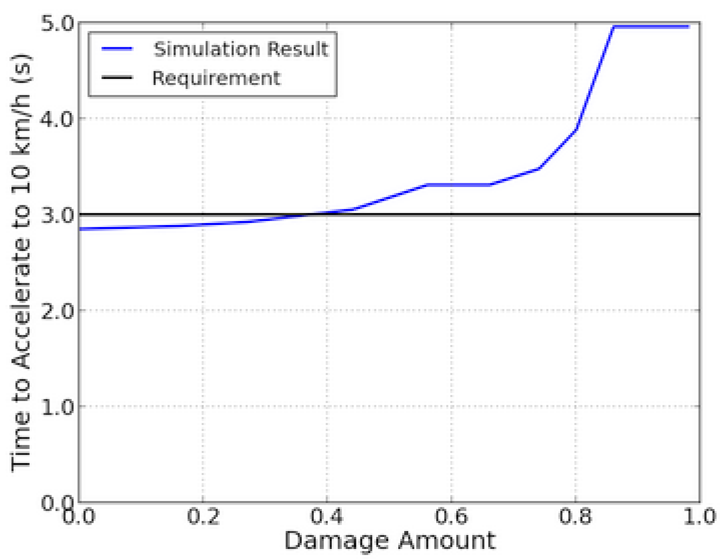

Figure 4: Performance Metric vs. Damage Amount

For the selected options, the critical damage amount is 0.35. Also, notice that the critical damage amount is sensitive to the required value. If the required value changes by 0.2 in either direction (shifting the black line up and down), then the critical damage amount could change from 0 to 0.5. This provides useful information about the design trade-off between different performance metrics and system reliability.

5.2.2 Damage parameter map for component failure mode

The Damage parameter map captures the damage accumulation process for a particular failure mode. The tool plots this damage accumulation as a damage amount vs. number of missions for four different percentile values (see below).

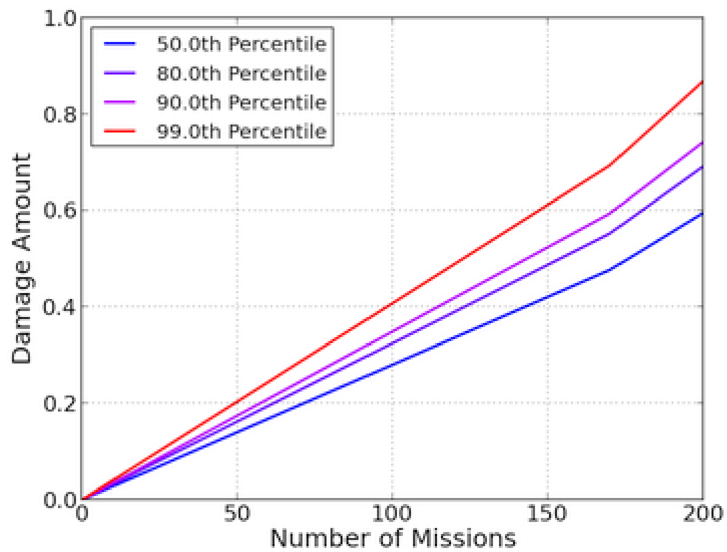

Figure 5: Damage Amount vs. Number of Missions

For this particular component, damage accumulation is mostly linear with a slight increase near 180 to 200 missions. After 100 missions, with about 50% probability the component will have a damage amount of 0.28 or greater, and there's a 1% chance that damage will exceed 0.40.
Note that the damage parameter map for catastrophic fatigue failure modes (e.g. PTM.TC.Shaft.FatigueFailure) is plotted slightly differently by showing only a single plot of probability of failure vs. number of missions, without percentiles.

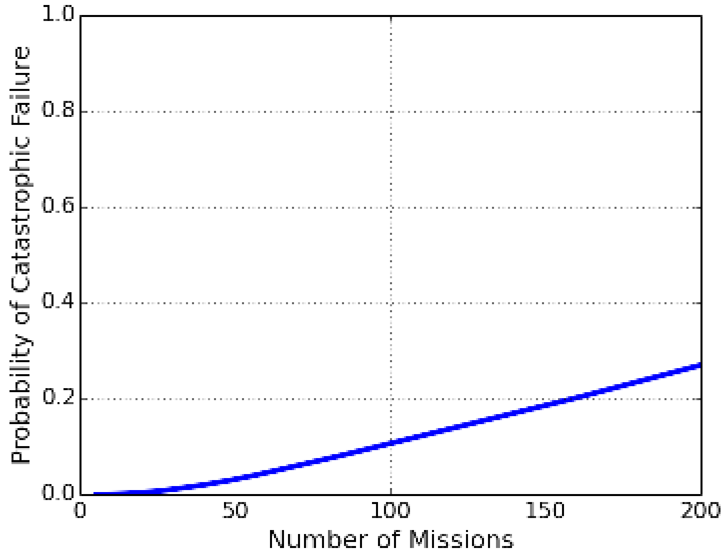

Figure 6: Damage Amount vs. Number of Missions for Catastrophic Failure Mode

5.2.3 Probability of not meeting requirement vs. the given number of missions

For a selected component subject to failure, the probability of meeting the selected required performance metric is obtained by combining the critical damage amount with the probability of incurring that much damage after the given number of missions. This is shown in the rightmost graph.

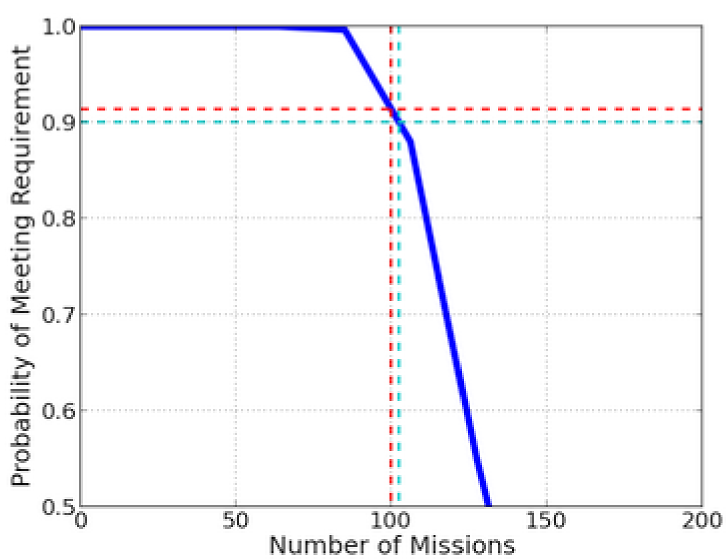

Figure 7: Probability of Not Meeting Requirement vs. Number of Missions

The locus of where the target number of missions (vertical dashed red line) crosses the probability of meeting performance metric required value (dark blue line), is where one can read off the probability of meeting the performance requirement (horizontal dashed red line).

5.2.4 Mission success probability

Across performance metrics, these probability values are listed in the "Estimated from simulation" column of the Performance Evaluation Table. They are shown in green if the probability exceeds the "Required" probability for that performance metric, and in red if it does not.

Performance Evaluation

Prob. performance metric is met after 100 missions

Required	Estimated from simulation
.95	1
.85	1
.9	0.913634
.8	0.6337675

All requirements met?:

Figure 8: Performance Evaluation Table

The Performance Evaluation Table summarizes the importance of the selected component failure mode for all of the performance requirements at the specified number of missions. In this configuration, it is shown that acceleration time to 15 kph is the critical performance metric at most risk for not meeting the specified requirement.

6.0 Use Case II: Evaluate Overall Reliability of a System Configuration

The Use Case I analysis focuses on the impact of a particular component failure on a performance requirement.
A second use case is to look at reliability from a system level perspective.

6.1 Compare System Mission Failure Probabilities among System Configurations

Because system-level Probability of Mission Failure aggregates the impact of all component failure modes and performance metrics into a single probability, it depends only on the specified number of missions.

Figure 9: Overall System Reliability - Probability of Mission Failure

By comparing the Probability of Mission Failure for different system configurations and number of missions, you as a designer can focus on better configurations. For this case, System Configs (2) and (5) stand out favorably, while System Configs (4) and (6) have probability of failure equal to 1.000, meaning that both fail from the first mission.

6.2 Understand Critical Component Failure Modes

By clicking on the Probability of Mission Failure value for a particular system configuration, a separate page containing Probability of Mission Failure Breakdown will appear. By default this page will open a new browser tab, but you can open the page in a new separate browser window by holding the Shift key while clicking on the number.

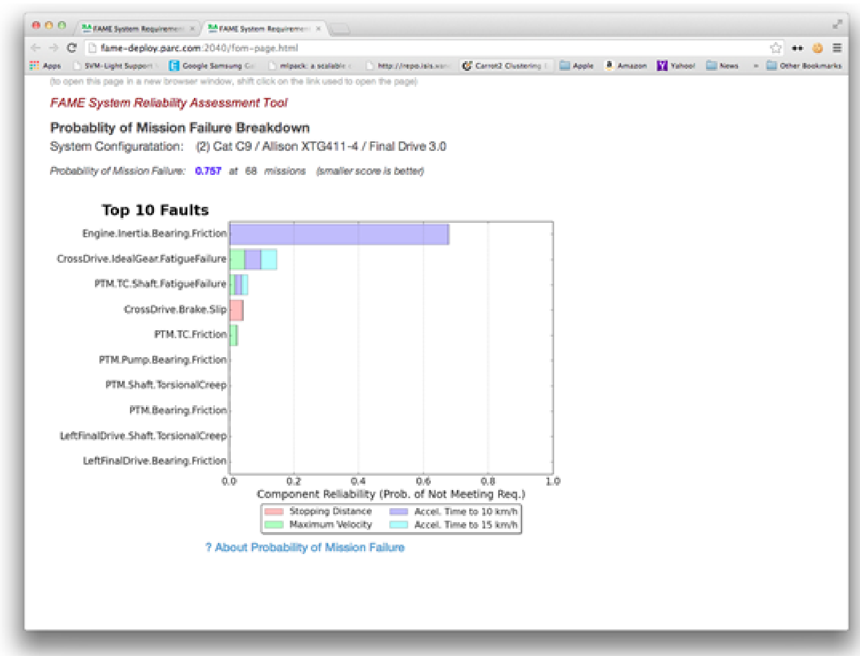

Figure 10: Probability of Mission Failure Breakdown for System Configuration (2) and 68 Missions

The Probability of Mission Failure Breakdown chart lists the ten most critical component failure modes. For this particular system configuration, the most critical failure mode is Engine.Inertia.Bearing.Friction. This bar's color coding indicates that it impacts the Performance Metric, Accel. Time to 10 km/h.

The second most important failure mode is CrossDrive.IdealGear.FatigueFailure. This is a catastrophic failure mode that impacts three different performance metrics.

7.0 Help and Guides

The user interface includes several help guides as highlighted in the green boxes below.

Figure 11: Hints and Guides in the FAME System Requirement Assessment Tool User Interface

These help guides include:

- Typical workflow
- Technical methodology
- Tips on other interesting configurations to explore for the *no_controls* test system.