Software reliability, safety and testing classes by Softrel, LLC

	Intended audience	Class duration	Available as open session	Available online	Available at your US site
IEEE 1633 Recommended Practices for Software Reliability Training	Reliability engineers, acquisitions personnel, systems engineers, software management, software testing	2 days	Yes	Yes	Yes
Integrating Software and hardware reliability predictions	Reliability engineers, acquisitions personnel, systems engineers	1 day	Yes	No	Yes
Software FMEA	Software engineers, software test engineers, reliability engineers	3 days	Yes	Yes	Yes
Software FTA	Software engineers, software test engineers, reliability engineers	1 day	No	No	Yes
Software safety analysis	Software engineers, software test engineers, safety engineers	1 day	No	No	Yes
Reliability driven software testing	Software test engineers, software engineers	2 days	No	No	Yes
Software analyses	Software engineers, software test engineers	2 days	No	No	Yes
Software reliability and FMEA Boot camp	Reliability engineers, software test engineers, software management	3 days	Yes	No	Yes
How to Specify Software Reliability on Government Contracts	DoD personnel	2 days	No	Yes	Yes

IEEE 1633 Recommended Practices for Software Reliability Training

Who should attend: Reliability engineers, systems engineers, software QA, software test engineers, software management, and acquisitions personnel.

1.0 Getting started

- Greetings and Introductions Software Reliability Timeline
- Industry guidance available for software reliability Vocabulary
- Overview of models that predict and estimate software reliability models
- Hard facts
- Mapping software to hardware reliability
- Failure modes that do and do not apply
- Where software fits within the product lifecycle Common myths
- Top list of things that everyone thinks is related to reliable software (but really isn't)
- Why software reliability growth is more limited than you think Overview of methods for reliability testing

2.0 Planning for software reliability

Topics	**Section of IEEE 1633 2016**
Characterize the software system	5.1.1
Define failures and criticality	5.1.2
Perform an initial risk assessment	5.1.3

3.0 Apply software reliability during development

Section of this presentation	**Section of IEEE 1633 2016**
1. Predict normalized effective size	5.3.2.3.1
2. Predict testing or fielded defect density using the SEI CMMi, industry type, Shortcut Model	
3. Predict total testing and fielded defects	
4. Predict when defects will be discovered over time	5.3.2.3.2
5. Predict failure rate and MTTF	5.3.2.3.3
5.1. Sanity check the predictions	5.3.3
6. Predict reliability	5.3.2.3.4
7. Predict availability	5.3.2.3.5
8. Sensitivity analysis	5.3.7
9. Apply predictions with incremental development	5.3.2.4
10. Predict defect pileup	5.3.6
11. Predict staff required to maintain software	5.5
Detailed methods for steps 1-8	
Step 1. Predicting size of COTS components	5.3.2.5
Step 2. Advanced models for predicting defect density - Quick Assessment, Full-scale, Neufelder, Rome Laboratory, Historical Data	5.3.2.3.1, 6.2 and Annex B
Step 4. Other options for predicting growth rate	Not included
Step 8. Advanced sensitivity analysis	5.3.7

Integrating software and hardware predictions

Who should attend: Reliability engineers, systems engineers, software QA, software test engineers, software management and acquisitions personnel.

1.0 Combining software and hardware reliability

Section of this training	**Section of IEEE 1633 Recommended Practices for Software Reliability 2016**
1.0 Identify an initial system reliability objective	5.3.1
2.0 Determine an appropriate overall software reliability requirement	5.3.5
3.0 Merge the software reliability predictions into the system prediction	5.3.4
4.0 Allocate the required reliability to the software LRUs	5.3.8
5.0 If the system objective can't be met, perform a system level sensitivity analysis	5.3.7.2

2.0 Apply software reliability during testing

Section of this training	**Section of IEEE 1633 Recommended Practices for Software Reliability 2016**
Overview -The reliability growth curve for software	5.4.4
How to know where the program is on that curve	
1.0 Collect the data	5.4.4
2.0 Plot the data	5.4.4 and 5.4.5
How to estimate the failure rate, MTBF from that curve	
3.0 Select the best model for the current trend	5.4.5
4.0 Compute the reliability figures of merit	6.3 and Annex B
5.0 Validate the accuracy of the estimation	5.4.7
6.0 Make a release decision	5.5

One Day Use Case Software FMEA

The "skinny" software FMEA is here. Apply the SFMEA to the most critical use cases and the most common software failure modes. This 1 day class can be taken before or after the 2 day software FMEA class. The one day class covers the application of the SFMEA to use cases while the 2 day class covers application of the SFMEA to software requirements and design. This class is intended for software engineers, firmware engineers, software architects, system architects, software requirements engineers, systems engineers, reliability engineers, software test engineers, software test managers, and acquisitions personnel.

Class Outline
Introduction - statement of goals for class and schedule
1. Preparing the use case Software FMEA. Identify riskiest use cases, ground rules, failure definition scoring criteria, most likely failure modes, identify personnel and time required for SFMEA.
2. Identify and analyze failure modes that a) Span all use cases b) Affect one use case c) Affect specific steps in a use case. Understand how these failure modes apply to use cases: a) faulty functionality b) faulty processing c) faulty timing d) faulty sequences e) faulty state management f) faulty data g) faulty error handling Understand the root causes for these failure modes. Several real examples will be presented.
3. Identify consequences of the software failure modes
4. Mitigate the software failure modes
5. Generate the Critical Items List
How to NOT conduct a SFMEA.
Closing, Q & A

Two Day Software Failure Modes Effects Analysis

This class outline maps directly to the below process for performing a software FMEA. This class maps to section 2.0 of the IEEE 1633 Recommended Practices for Software Reliability, 2016.

Who should attend: Reliability engineers, systems engineers, software QA, software test engineers, software management, software architects, software requirements engineers, and acquisitions personnel. Class handouts include the "Effective Application of Software Failure Modes Effects Analysis" book. The software FMEA toolkit is optional. A third day of hands on application is optional.

Class outline
Introduction - statement of goals for class and schedule Real examples of how software FMEAs were used to find serious defects in a cost effective manner.
1. Preparing the SFMEA. a. Identify the scope of the SFMEA - the riskiest and most critical part of the software. b. Identify the people and artifacts needed to do the SFMEA. Identify the viewpoints that are most applicable for the current phase of development and project risks. c. Identify the ground rules for the SFMEA. d. Identify the failure definition and scoring criteria to be used for the SFMEA
2. Brainstorm past failure modes. Employ a defect root cause analysis or software fault tree analysis.
3. Identify failure modes for the functional SFMEA viewpoint
4. Identify consequences
5. Mitigate
6. Generate the Critical Items List
Class example - The class will see a real example of a functional SFMEA
Identify failure modes for the interface SFMEA viewpoint. Interface FMEAs analyze failure modes between software, firmware and hardware. Class example - The class will see a real example of an interface SFMEA
Identify failure modes for the detailed SFMEA viewpoint. A detailed design FMEA is performed on the design or code.
Identify failure modes for the maintenance SFMEA viewpoint. A maintenance process FMEA analyzes the failure modes related to how people support the software once it is deployed. The focus is on failure modes that would cause previously functional software to stop functioning.
How to perform a vulnerability SFMEA. This is a detailed SFMEA that focuses on the design and coding failure modes that are also related to vulnerabilities
How to perform a production process FMEA. A production process FMEA analyzes the failure modes related to how people produce the software product. It's possible for the requirements, design and code to be working, but for the software to be unusable because there is no source control.
How to perform a serviceability SFMEA. An installation process FMEA analyzes the failure modes related to an end user's or system installation. For example, the software could be working properly but the installation of it might fail. Or the end user may have an incorrect user's manual and be unable to use the software. Class exercise - The entire FMEA process will be executed from analyzing resources to improving the product.
How to NOT conduct a SFMEA

Online Software FMEA Training

The online training consists of 3 modules. Module 1 is a required prerequisite for modules 2 and 3.

Online SFMEA training outline
MODULE 1 - All students must take module 1 before proceeding to the other modules. Module 1 is appropriate for reliability engineers, systems engineers, software engineers, software architects, software managers, software requirements engineers, system requirements engineers, software test engineers, software test managers, acquisition personnel.
Introduction - statement of goals for class
1. Preparing the Software FMEA. a. Identify the scope of the Software FMEA - the riskiest and most critical part of the software. b. Identify the people and artifacts needed to do the SFMEA. c. Identify the viewpoints that are most applicable for the current phase of development and project risks. d. Identify the groundrules for the Software FMEA. e. Identify the failure definition and scoring criteria to be used for the Software FMEA
2. Brainstorm past software failure modes and root causes
3. Identify failure modes for the functional Software FMEA viewpoint
4. Identify consequences
5. Mitigate the software failure modes
6. Generate the Critical Items List
7. How to NOT perform a Software FMEA
MODULE 2 - This module is geared towards software and system architects. However, it can be taken by reliability engineers, systems engineers, software engineers, software managers, software test engineers, software test managers, acquisition personnel.
Identify failure modes for the interface SFMEA viewpoint. Interface FMEAs analyze failure modes between software, firmware and hardware.
Identify failure modes for usability SFMEA viewpoint. A system failure can occur if an end user makes a mistake due to overly cumbersome software or instructions or due to faulty assumptions about the end users.
MODULE 3 - This module is geared towards software engineers who are developing the code. It is also appropriate for software architects, software managers.
Identify failure modes for the detailed SFMEA viewpoint.. A detailed design SFMEA is performed on the design or code or pseudocode.
Identify failure modes for the maintenance SFMEA viewpoint.. A maintenance process FMEA analyzes the failure modes related to how people support the software once it is deployed. The focus is on failure modes that would cause previously functional software to stop functioning.
Identify failure modes for the cyber/vulnerability SFMEA. This is a detailed SFMEA that focuses on the design and coding failure modes that are also related to vulnerabilities.
Identify failure modes for the installation SFMEA. An installation SFMEA analyzes the failure modes related to a software installation package or an update to an installation. For example, the software could be working properly but the installation of it might fail.

Software Analyses

This is a 2 day class intended for software engineers.

Topic
Day 1
Introduction to software analysis • Industry standards relevant to software analysis • Summary of each analysis and when used • Overview of each analysis • Tailoring guidelines • Artifacts to collect for each analysis
Software Analyses during requirements phase • Purpose of the analysis, Inputs and outputs of the analysis, Steps and techniques to perform the analysis, Documentation of results, Use the results to improve the software, Automated tools for the analysis, Examples
Day 2
Software Analyses during design phase • Purpose of the analysis, Inputs and outputs of the analysis, Steps and techniques to perform the analysis, Documentation of results, Use the results to improve the software, Automated tools for the analysis, Examples
Software Analyses during coding • Purpose of the analysis, Inputs and outputs of the analysis, Steps and techniques to perform the analysis, Documentation of results, Use the results to improve the software, Automated tools for the analysis, Examples
Day 3
Software Analyses during testing • Purpose of the analysis, Inputs and outputs of the analysis, Steps and techniques to perform the analysis, Documentation of results, Use the results to improve the software, Automated tools for the analysis, Examples
Conclusions

2 Day Reliability Driven Software Testing

This course is intended to provide software testing principles that are geared towards reducing defects effectively and efficiently. These software testing techniques are the very same techniques that have been found to be quantitatively associated with fewer escapes. The target audience is software engineers, software verification engineers, acquisitions personnel, software quality engineers, software managers. This class maps to clauses 5.4.1 and 5.4.3 of the IEEE 1633 Recommended Practices for Software Reliability.

Agenda

Getting started - greeting, introductions

a) Understand the difference between developer and system level testing

b) Understand the difference between and white box, grey box and black box test

Developer level testing

a) How to determine what to test, when to test , the expected coverage and the tools that you will need

b) White box tests

- Path/Logic
- Domain/Boundary
- Mathematical testing
- Examples

c) Grey box tests

- Design verification
- Unit level exceptions
- Examples

System level testing

a) When to start the test plan

- Know how to review the Software Requirements Spec (SRS)
- Know how to translate the SRS to a test strategy
- Know how to streamline the plans and reports to support traceability
- Know how to streamline the plans and reports to easily estimate coverage
- Know how to order the test plan so that most critical tests are run first

b) Know what to put in the test plan and when to test

- Requirements validation
- Use cases
- State transitions/timing/sequence
- Stress
- System level exceptions
- Performance
- Special tests
- Client/Server
- Database
- Web
- Examples

c) Things that you should not do with regards to systems testing

d) Tools

Know when to stop testing

- How to establish entrance and exit criteria for developer and systems level testing
- Metrics that are useful to determining when to stop testing

Software Fault Tree Analysis

The goals of the Software Fault Tree Analysis training course are:

- To be able to perform a software fault tree analysis on software during any phase of the software development process
- To be proficient in software fault tree analysis immediately after the course is complete
- To have examples of real software fault trees from real software systems

While knowledge of the software engineering process is desirable, it is not required. Knowledge of reliability engineering and/or experience with fault trees on hardware systems is not required. This class is ideal for a group of software engineers, systems engineers and reliability engineers.

Topic
Introduction - statement of goals for class and schedule
The process for executing a fault tree with managed resources and schedule • Plan resources • Brainstorm failure events • Create the tree • Assess probability and severity and determine if within mitigation threshold • Revise applicable product documents (requirements, design, code, test plan)
Break
How to perform a fault tree during the requirements phase.
Class exercise - We will execute the entire process from planning resources to revising the applicable product
How to perform a fault tree during the design phase - We will execute the entire process from planning resources to revising the applicable product
Class exercise
How to perform a fault tree during the coding/unit testing phase - We will execute the entire process from planning resources to revising the applicable product
Class exercise
How to perform a fault tree during system testing and integration and maintenance - We will execute the entire process from planning resources to revising the applicable product
How to compute the probability of success of an event from the fault tree
• Recap how the fault trees helped to define the product - the goal was a better product within the resource and schedule constraints - not an exquisite fault tree! • Closing • Q&A

Software Safety Analysis

There are an assortment of training classes on software safety. This 2 day training class merges the criteria from AOP-52 Generic Software Safety Design Requirements with a skinny Software Failure modes Effects Analysis and a Software Fault Tree Analysis. The target Audience is software engineers, software test engineers, software safety engineers.

Each course attendee learn how to analyze each software use case, requirement statement, design statement and determine which of the AOP-52 compliance criteria is applicable, met, not met. Each course attendee will learn how to use the results of the analysis to drive other analyses such as the software FMEA.

Class Outline
Learn how the AOP-52 which is applicable even for non-weapon systems
Learn how to compliance of software requirements with the AOP-52 Generic Software Safety Design Requirements
Learn how to use the software fault tree analysis to drive the identification of the safety related hazard
Conduct a software fault tree analysis(SFTA) with focus on safety related hazards
Learn how a software FMEA can facilitate the software safety hazards analysis.
Conduct a software failure mode effects analysis (SFMEA) with focus on safety related hazards and safety critical requirements, interfaces, design
Learn how to use the AOP-52 assessment, software FMEA and software fault tree analysis to drive the software safety hazards assessment
Learn how to Perform a software safety hazards assessment with application of DoD MIL-STD-882E System

Software Reliability and SFMEA Boot camp

This three day class covers topics from the following courses:

- IEEE 1633 Recommended Practices for Software Reliability. This 2 day course is shortened to 1.5 days. Section 2 is removed. Section 3 items 6 and beyond are removed.
- One Day Use Case Software Failure Modes Effects Analysis.
- Integrating Hardware and Software Reliability. This 1 day course is shortened to .5 days. Section 1 items 2, 3, 5 are removed. Section 2 is shortened to cover one model.

The below table illustrates which training classes cover the topics in the IEEE 1633 Recommend Practices

Training class	**Software reliability task**	
IEEE 1633 Recommended Practices for software reliability, Software Reliability Boot camp	5.1. Planning for software reliability	
	5.1.1	Characterize the software system
	5.2. Develop Failure Modes Model	
Software failure modes effects analysis – 2 or 3 day	5.2.1	Perform a defect root cause analysis
Software failure modes effects analysis – 1, 2 or 3 day	5.2.2	Perform Software Failure Modes Effects Analysis (SFMEA)
Software Fault Tree Analysis	5.2.3	Include Software in the System Fault Tree Analysis
	5.3. Apply software reliability during development	
Integrating software and hardware reliability	5.3.1	Identify/Obtain the initial system reliability objective – The required or desired MTBF, failure rate, availability, reliability for the entire system.
IEEE 1633 Recommended Practices for software reliability, Software Reliability Boot camp	5.3.2	Perform a Software Reliability Assessment and Prediction – Predict the MTBF, failure rate, availability, reliability and confidence bounds for each software LRU.
	5.3.3	Sanity Check the Prediction – Compare the prediction to established ranges for similar software LRUs
Integrating software and hardware reliability	5.3.4	Merge the predictions into the overall system predictions – Various methods exist to combine the software and hardware predictions into one system prediction.
IEEE 1633 Recommended Practices for software reliability, Software Reliability Boot camp	5.3.5	Determine an appropriate overall Software Reliability Requirement – Now that the system prediction is complete, revisit the initial reliability objective and modify as needed.
IEEE 1633 Recommended Practices for software reliability, Software Reliability Boot camp	5.3.6	Plan the reliability growth – Compute the software predictions during and after a specific level of reliability growth testing.
IEEE 1633 Recommended Practices for software reliability	5.3.7	Perform a Sensitivity Analysis – Identify the practices, processes, techniques, risks that are most sensitive to the predictions. Perform tradeoffs.
Integrating software and hardware reliability	5.3.8	Allocate the Required Reliability to the Software LRUs– The software and hardware components are allocated a portion of the system objective based on the predicted values for each LRU.

Training class	**Software reliability task**	
	5. 4. Apply Software Reliability during Testing	
Reliability Driven Software Testing	5.4.1	Develop a reliability test suite
	5.4.3	Measure test coverage
Integrating software and hardware reliability, Software Reliability Bootcamp	5.4.4	Collect Fault and Failure Data– This data is collected during testing
	5.4.5	Select Reliability Growth Models – The best models are those that
	5.5 Make a release decision	
	5.5.1	Determine release stability
	5.5.2	Forecast additional test duration
	5.5.3	Forecast remaining defects and effort required to fix them
	6.0 Software reliability models	
IEEE 1633 Recommended Practices for software reliability, Software Reliability Bootcamp	6.1	Overview
	6.2	Models that can be used before testing
Integrating software and hardware predictions, Software Reliability Bootcamp	6.3	Models that can be used during and after testing