

## **System Software Reliability**

This training course is tailored for reliability engineers, systems engineers, quality assurance engineers, and software engineers and testers. Featuring hands-on software reliability measurement, analyses and design, it is intended for those individuals responsible for measuring, analyzing, designing, automating, implementing or ensuring software reliability for either commercial or government programs. Practical approaches are stressed with many examples included. It is not necessary to have a software background or a reliability background to attend the course, however, either or both are helpful. Hand-outs include a training course manual and a copy of Ann Marie Neufelder's "Ensuring Software Reliability."

### **Course Instructor**

Ann Marie (Leone) Neufelder is the President of SoftRel, a software reliability research, development and testing organization. For more than 20 years Ann Marie has been measuring and improving software reliability on a variety of systems in defense, aerospace, space, healthcare, electronics, and commercial industries. She co-authored two industry standards on software reliability and has published "Ensuring Software Reliability" with Marcel Dekker Inc. Her experience is in the practical hands on application of software reliability measurement and her course reflects this. Ann Marie is a 1983 graduate of Georgia Tech.

### **Course Contents**

#### *Course Introduction*

#### *Section I - Get Started*

1. What will software reliability metrics do for me?
2. Know the vocabulary, terms, acronyms, myths and comparisons to hardware reliability
3. Know the characteristics that impact software defects, failure rate, reliability and late delivery
4. Know some simple metrics that can be used by any organization
5. Identify some of the software reliability prediction models and what they can do for you
6. Identify some software reliability growth models and what they can do for you

#### *Section II - Predict defects; reduce them effectively and efficiently; plan development and test resources to meet objectives*

1. **Collect data**
  - Administer software reliability prediction survey(s) and determine benchmarking score
  - Predict effective size

- Predict growth rate, growth period, duty cycle, restore time, percentage of critical defects
  - Class exercise
2. **Quantitative analysis**
    - Predict defect density
    - Predict fielded defects and in test defects
      - Class exercise
    - Transform defect prediction to failure rate prediction
      - Class exercise
    - Compute customer related metrics
      - Predict Mean Time To Software Restore (MTSWR)
      - Predict availability
      - Predict reliability
      - Class exercise
    - Compute resource related metrics
      - Determine how much testing time will be needed before testing or development starts
      - Determine how many people will be needed to support the software before testing or development starts
      - Class exercise
  3. **Determine key practices that will optimize defects with the least cost and schedule time and maximum probability of success**
    - Class exercise

*Section III – Estimate reliability growth during testing; determine when to stop testing and plan support resources*

1. **Collect defect and usage data during system and verification testing**
2. **Estimate parameters needed for models**
  - Graph defect rate versus cumulative defects
3. **Estimate inherent testing defects, current failure rate and MTTF and extrapolate into the future**
  - Filter by severity
4. **Determine accuracy of each growth model and compare**
  - Determine which estimator is the best choice for this phase of this project
  - Check your results for reasonableness
5. **Determine additional testing hours needed to reach some quantitative objective**
6. **Determine how many defects will be fielded if testing stops now**
7. **Determine when to stop testing**
  - How to improve failure rate, MTTF, reliability, inherent defects once testing has started
  - How to perform a root cause analysis on software

- A real example from a real software system

## **Section IV – Software fault trees and FMEAs**

### **1. Software fault trees**

- Plan resources for the software fault tree
- Gather applicable product documents (Systems requirements spec, software requirements spec, software design spec)
- Brainstorm system failure events
- Place each event on the fault tree
  - Common mistakes made when creating a fault tree
- Assess severity and probability
- Rank mitigation effort
- Revise the applicable product
- An example software fault tree from a real software system

### **2. Software Failure Modes Effects Analyses (FMEAs)**

- Plan resources for the Component, Interface and Module software FMEA
- Analyze the applicable product documents (Functional requirements, interface design or component design)
- Brainstorm failure modes applicable to this product document
- For each failure mode identify/assess
  - Probability
  - Severity
  - Failure effects
  - Corrective actions
  - Detection monitors
  - Compensating provisions
  - Criticality (If performing a FMECA)
- Identify failure mode equivalence
- An example software FMEA from a real system

## **Section V – Combine software and hardware predictions and make system level tradeoffs**

- 1. How to allocate system reliability objectives to software**
- 2. How to combine software and hardware reliability predictions**
- 3. How to make system level tradeoffs**
- 4. How software can be redundant**
- 5. How to include software in a reliability block diagram**