

## 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 Contents

#### Course Introduction

#### Know the Basics About Software Reliability

1. Software reliability defined
2. The software development process mapped to software reliability tasks
3. Primitives that you need to understand
  - A. Defects, faults, failures and errors
  - B. Escapes
  - C. Defect density
  - D. Total effective size
  - E. Initial failure rate –  $\lambda_0$
  - F. Inherent defects –  $N_0$
  - G. Delivered failure rate –  $\lambda_{del}$
  - H. Inherent delivered defects –  $N_{del}$
  - I. Growth rate during testing –  $Q_0$
  - J. Growth rate after delivery –  $Q_{del}$
  - K. Types of software defects
4. Predicting versus estimating
5. The process for predicting
6. How predictor models work
7. The process for estimating
8. Some simple metrics for any organization
9. When software reliability is similar to hardware reliability and when it's not
10. Some factors that impact defects, failure rate and reliability
11. Some common myths about software reliability
12. Incremental life cycle models

#### Measurements Used Early in the Lifecycle

1. How to predict failure rate or MTTF before testing or development
2. How to predict failure rate or MTTF by categories of criticality
3. How to predict inherent delivered defects before testing or development
4. How to determine how much testing time will be needed before testing or development starts
5. How to determine how many people will be needed to support the software before testing or development starts
6. How to combine software measures with hardware measures before testing or development starts
7. How to predict Mean Time To Software Restore (MTSWR)
8. How to predict availability before testing or development
9. How to predict reliability before testing or development
10. How to improve failure rate, MTTF, reliability, inherent defects before development is complete
11. How to check your prediction results for reasonableness

## **Measurements Used Later in the Lifecycle**

1. How to determine how much more testing time is needed to meet an objective
2. How to estimate inherent defects during testing
3. How to estimate failure rate or MTTF
4. How to estimate failure rate or MTTF by categories of criticality
5. How to determine which estimator is the best choice
6. How to check your results for reasonableness
7. How to determine when it is OK to stop testing
8. How to combine software measures with hardware measures during testing
9. How to improve failure rate, MTTF, reliability, inherent defects once testing has started
10. How to measure actual defect density (and therefore validate a prediction)
11. How to perform a root cause analysis on software
12. How to determine removal rate – the percentage of defects that are removed prior to delivery

## **Systems Analysis**

1. How to include software in a reliability block diagram
2. How to allocate system reliability to software
3. How to apply fault trees to software
4. How to apply FMEAs to software
5. How software can be redundant
6. Reliability concepts for software engineers
7. Software concepts for reliability engineers