

Software Design for Reliability

Introduction

Group exercise: reliability self-assessment

Software reliability basic concepts

- ◆ Software quality vs. reliability
- ◆ Definition
- ◆ Terminology
- ◆ Failure classification
- ◆ Failure rates, distributions and MTTF
- ◆ Availability and failure rates
- ◆ Input robustness
- ◆ Fault tolerance

Best Practices approach to developing reliability software

- ◆ The paths to reliable software
- ◆ Software DfR options
- ◆ Formal methods
- ◆ Hardware-based
- ◆ Best practices
 - Comparing development behavior patterns
 - Best practices from “best in class” companies
 - Weakness within development organizations
- ◆ Defect removal strategy
 - Defect phase propagation
 - Integrating reliability practices into the development lifecycle
 - Comparing defect removal practices
 - Case study
- ◆ DfR based on a “Best Practices” approach
 - Defect lifecycle and DfR goals
 - Summary of DfR best practices

Reliability measurements, metrics and data analysis

- ◆ Measurements and metrics to be tracked
- ◆ Defect removal efficiency
- ◆ Project critical defect and failure tracking
- ◆ Failure rate data
- ◆ Data analysis
- ◆ Failure density analysis
- ◆ Failure causal analysis

Software reliability modeling

- ◆ Predictive
- ◆ Estimation

Unit test phase practices

- ◆ Software robustness failure modes
- ◆ Input equivalence classes
- ◆ Coverage-based unit testing
- ◆ Unit testing strategies

System test phase practices

- ◆ What is reliability testing
- ◆ Reliability growth
- ◆ Usage profiles
- ◆ Generating software reliability estimates
- ◆ Sample software fault tolerance techniques
- ◆ Thread / process monitoring
- ◆ Thread / process recovery
- ◆ System resource monitoring and recovery
- ◆ Software rejuvenation and reentrance

Wrap-up