

ALA CARTE™ SOFTWARE RELIABILITY

The State of Software Reliability Programs Today

Companies developing products containing embedded software use the following approaches to create reliable software:

- Traditional software reliability programs
- Process control
- Quality through software testing

Traditional software reliability programs treat the development process as a software-generating black box. Predictive models are generated, usually by a separate team of reliability engineers, to provide estimates of the number of faults in the resulting software; greater consistency in reliability leads to increased accuracy in the modeling output. Within the black box, a combination of reliability techniques like failure analysis, e.g., FMEAs (Failure Mode and Effects Analysis), FTAs (Fault Tree Analysis), defect tracking, and operational profile testing are used to identify defects and produce software reliability metrics.

Process control assumes a correlation between process maturity and latent defect density in the final software. Organizations implementing CMM Level 3 processes generate software containing 2.0-3.5 faults per KSLOC (1000 source lines of code). If the current process level does not yield the desired software reliability, audits and stricter process controls are implemented.

Quality through software testing is the most prevalent approach for implementing software reliability within small or unstructured development organizations. This approach assumes that reliability can be increased by expanding the types of system tests (e.g., integration, performance and loading) and increasing the duration of testing. Software reliability is measured by various methods of defect counting and classification. Generally, these approaches fail to achieve their software reliability targets. Companies find that their software engineers spend more time debugging than designing or coding, and accurate software reliability measurements are not available at deployment to share with customers.

What can development organizations do to increase software reliability without significantly increasing schedules or budgets?



Software Reliability Integration throughout the Product Life Cycle

Ops A La Carte offers software reliability training and consulting services that *integrate* with the above approaches to produce better results.

Our training and consulting services focus directly on the practices of software and test engineers.

We offer services at the Organizational Level which will improve software processes and therefore will improve the reliability of all products. We also offer Product Level services that are aimed at improving specific software programs

Ops A La Carte shall tailor a program to fit your product, market, and organization so that you get the *optimal* solution. We provide Educational Services for smooth transitions.

OPS A LA CARTE™ SOFTWARE RELIABILITY

Software Reliability Integration for the Organization

Software Reliability Assessment

Software Development Best Practices Assessment

Before assisting with specific programs, we recommend that an organization **assess** its development practices against industry best practices to ensure they have a solid foundation upon which to integrate our reliability services. If necessary, we will help an organization fill in gaps by identifying existing, internal best practices and tools to yield the desired results. Then, we help define a set of reliability practices to move defect prevention and detection as far upstream of the development cycle as possible.



Once we have assessed the Software Reliability Program, we have a number of software reliability services that *integrate* with specific software programs.

Software Reliability in the Concept Phase

Software Reliability Goal Setting

Software Reliability Program and Integration Plan

Software Reliability in Implementation Phase

Facilitation of Code Reliability Reviews

Software Robustness and Coverage Testing Techniques

Software Reliability in the Design Phase

Facilitation of Team Design Template Reviews

Facilitation of Team Design Reviews

Software Failure Analysis

Software Fault Tolerance

Software Reliability in the Testing Phase

Software Reliability Measurements and Metrics

Usage Profile-based Testing

Software Reliability Estimation Techniques

Software Reliability Demonstration Tests

Software Reliability Integration in the Concept Phase

In the concept phase, the software team should be able to assist in defining system level reliability and availability software goals, which are different from hardware goals. These goals become part of the overall Reliability Program and Integration Plan and are applied to the design and testing phases.

Software Reliability Integration in the Design Phase

In the design phase, group pre-design review meetings provide engineers with forums to expand their knowledge base of design techniques by exchanging design templates. Design inspection results will be greatly improved if they are preceded by brief, informal reviews that are highly-interactive at multiple points throughout the progression from system architecture through low-level design. Prior to the final stage of design, software failure analysis is used to identify core and vulnerable sections of the software which may benefit from additional run-time protection by incorporating software fault tolerance techniques.

Software Reliability Integration in the Implementation Phase

In the implementation phase, reliability reviews target only the core and vulnerable sections of code to allow the owner of the source code to develop sufficient synergy with a small team of developers in finding defects. Unit testing efforts focus on efficient detection of software faults using robustness and coverage testing techniques for through module-level testing.

Software Reliability Integration in the Testing Phase

In the system testing phase, reliability measurements and metrics are used to track the number of remaining software defects, the software MTTF (Mean Time To Failure) and to anticipate when the software is ready for deployment. The test engineers will be able to apply usage profiling mechanisms to emphasize test cases based on their anticipated frequency of execution in the field.