Presentation
Paper
Bio

P R E S E N T A T I O N

WG3

Wednesday, May 12, 1999
4:30PM

HOW TESTERS CAN USE A SOFTWARE RELIABILITY ENGINEERING MATURITY MODEL

John Musa
International Consultant

International Conference On
Software Testing, Analysis & Review
MAY 10-14, 1999 • ORLANDO, FL

How Testers Can Use a Software Reliability Engineering (SRE) Maturity Model

John D. Musa

j.musa@ieee.org

Outline

- What is SRE?
- SRE process
- What is the SRE Maturity Model?
- Effects of SRE maturity level
- What SRE does for testers
- To explore further

What is SRE?

SRE is a practice that reduces risks of unreliability or unavailability of released product, missed schedules, and cost overruns by *quantitatively* planning and guiding software development and test to meet user needs.

How Does SRE Work?

- SRE improves all major quality characteristics (reliability, availability, delivery time, life cycle cost) of product by quantitatively characterizing expected use and employing this information to
  - Precisely focus resources on most used and/or most critical functions or modules
  - Make test more effective by realistically representing field conditions

How Does SRE Work?

- SRE matches major quality characteristics to user needs more precisely by:
  - Setting quantitative objectives for availability and/or reliability as well as schedule and life cycle cost
  - Engineering project strategies to meet objectives
  - Tracking reliability of each system in test against its objective as one of the release criteria

Reliability and Availability Definitions

- *Failure:* departure of system behavior in execution from user needs
- *Failure intensity (FI)*: failures per unit time
- *Reliability:* probability a system functions without failure for a specified time in a specified environment
- *Availability:* the probability that a system is functional at a given time in a specified environment

Activities of SRE Process and Relation to Software Development Process

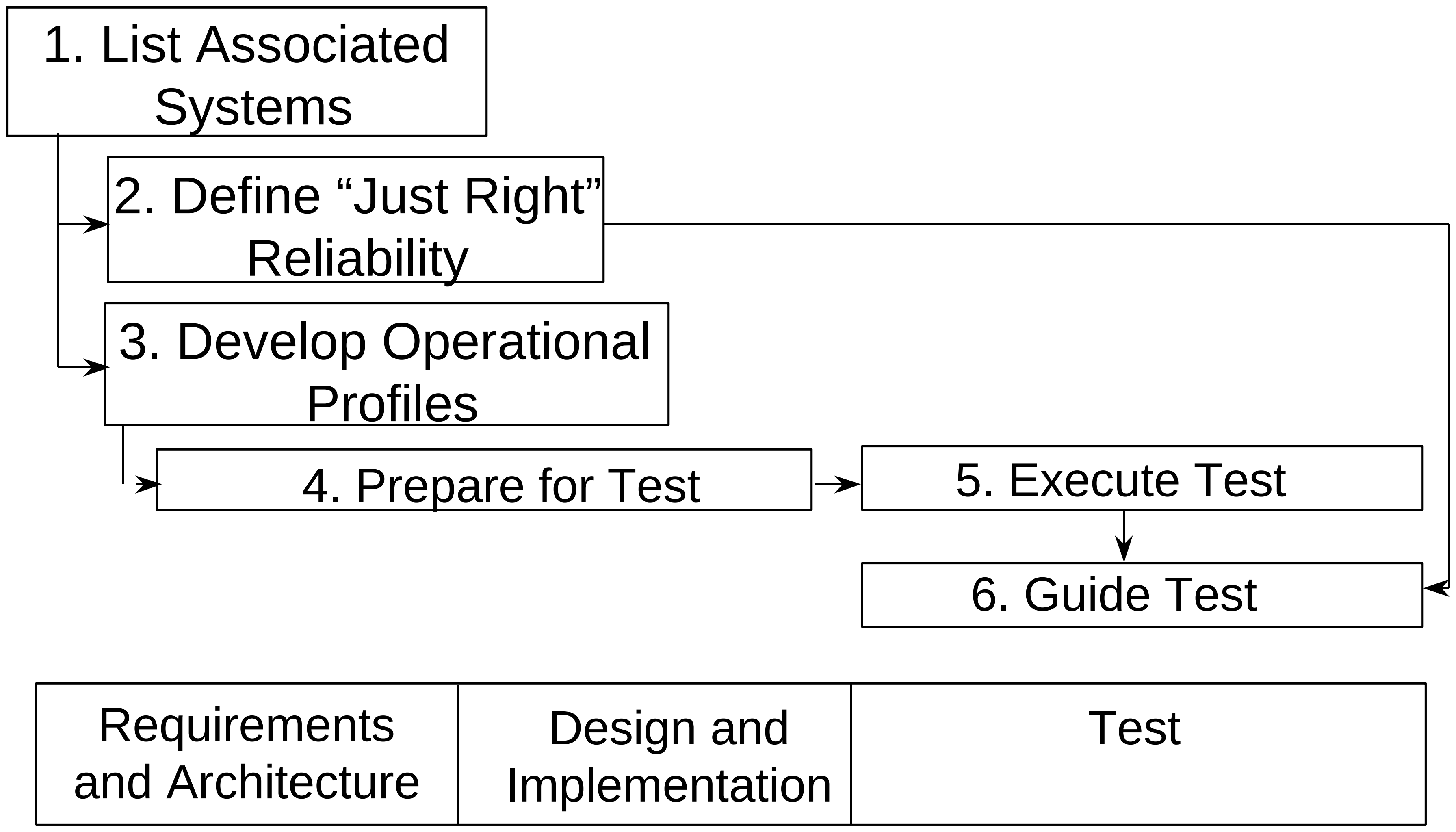

1. List Associated Systems
2. Define "Just Right" Reliability
3. Develop Operational Profiles
4. Prepare for Test
5. Execute Test
6. Guide Test
Requirements and Architecture
Design and Implementation
Test

Illustration - FONE FOLLOWER (FF) - Product Description

- Subscriber calls FF and enters planned phone numbers for where he/she plans to be vs time
- Incoming calls (voice or fax) from network to subscriber are forwarded as per program. Incomplete voice calls go to pager (if user has one) and then voice mail.

Define “Just Right” Reliability

- Set FIO for each associated system as part of requirements, balancing among major quality characteristics users need, using data from similar release or product
  - Customer satisfaction surveys vs failure intensity
  - Analysis of competing products
- Engineer project software reliability strategies to meet these objectives

Develop Operational Profiles - What Are They?

Operational profile: complete set of functions with probabilities of occurrence

Illustration - FF:

Operation	Occ. Prob.
Process voice call, no pager, ans.	0.18
Process voice call, no pager, no ans.	0.17
Process fax call	0.15
•	
•	
•	
	1

Some Applications of Operational Profiles

- System engineering: reduce number of low use operations, focus resources
- Allocate test cases to operations

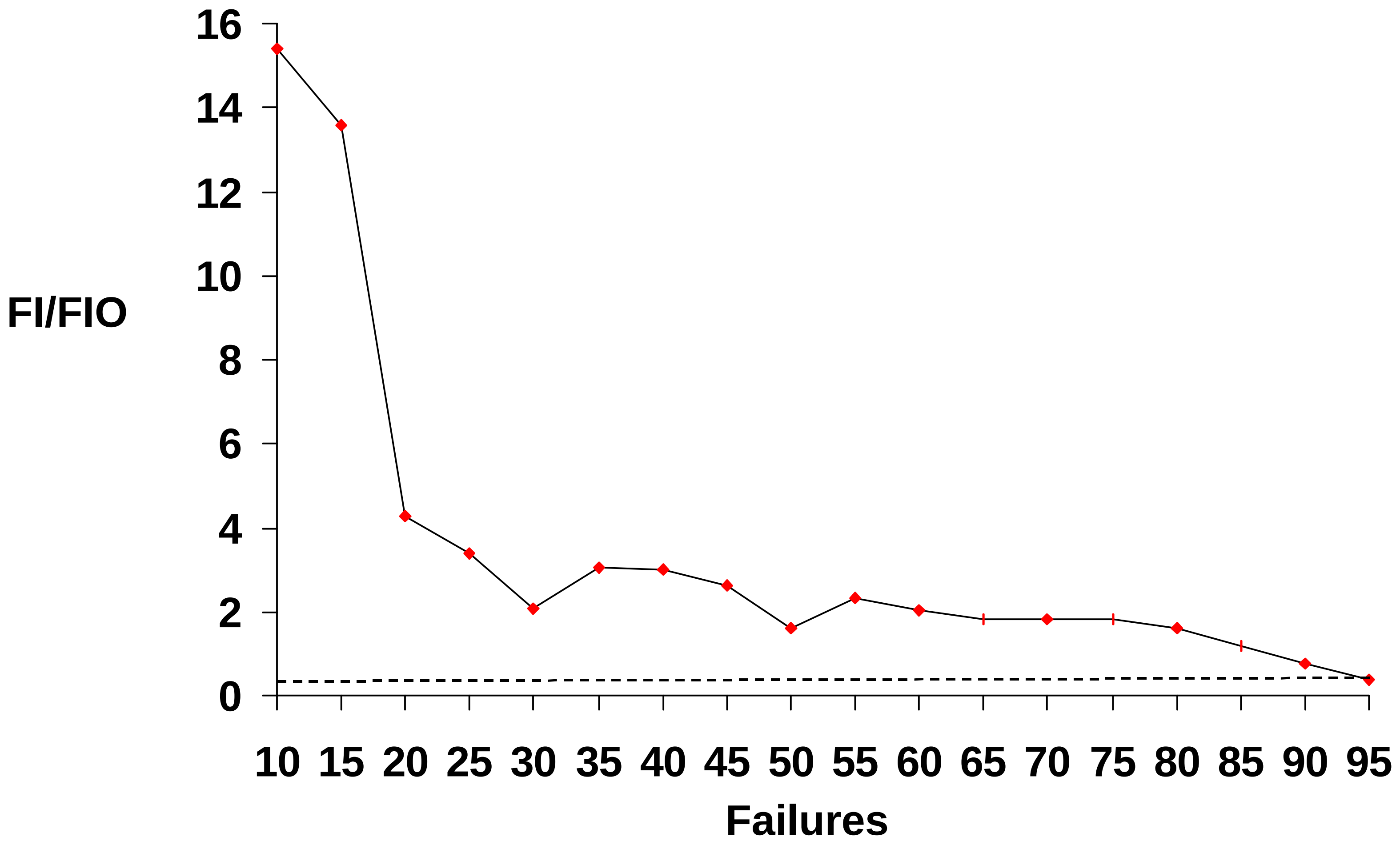
  Illustration - FF:

  Allocate 15% of test cases to Process fax call operation
- Allocate test time among operations

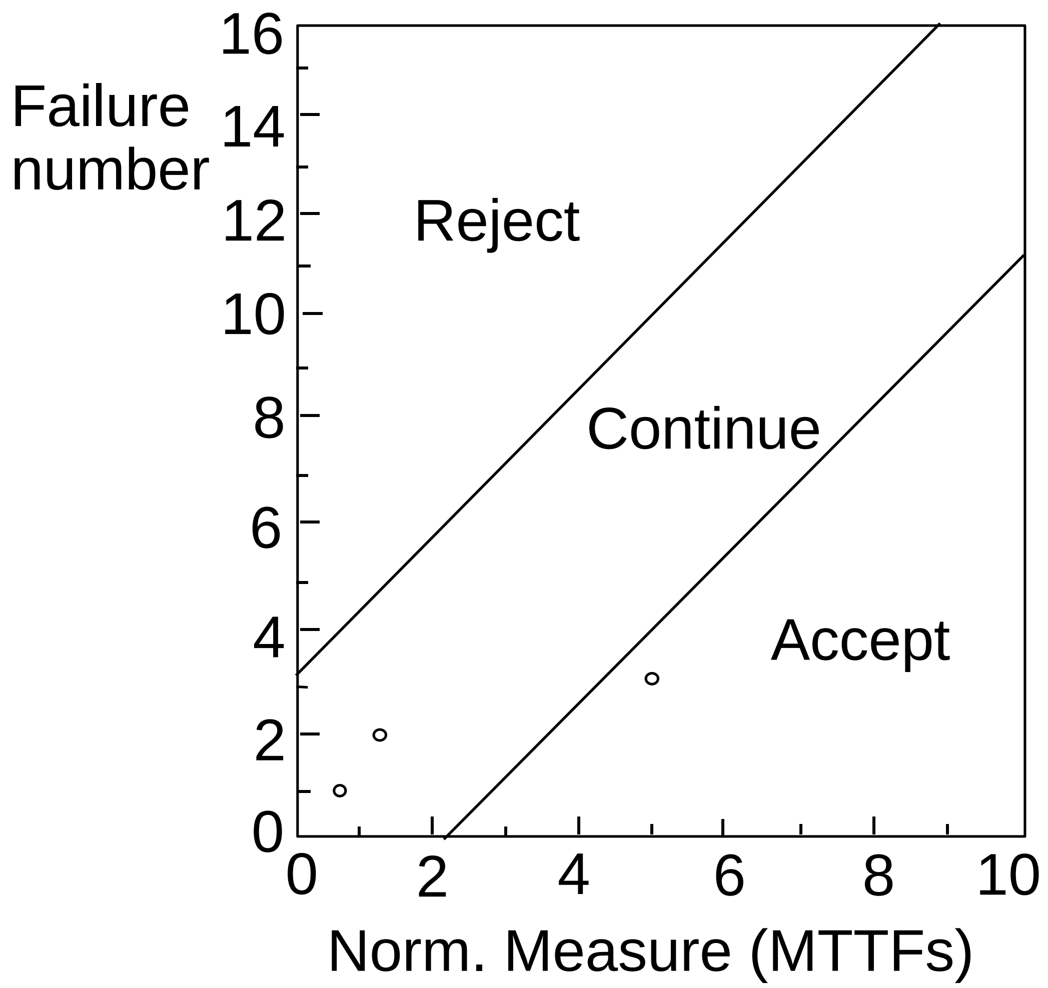
Guide Test

- Use failure data from test to:
  - Track reliability growth of developed software using software reliability estimation program
  - Certification test (accept or reject) certain systems such as acquired components

Estimate Total FI / FIO Ratio and Plot Trend : Illustration - FF



FF Illustration: Certification Test

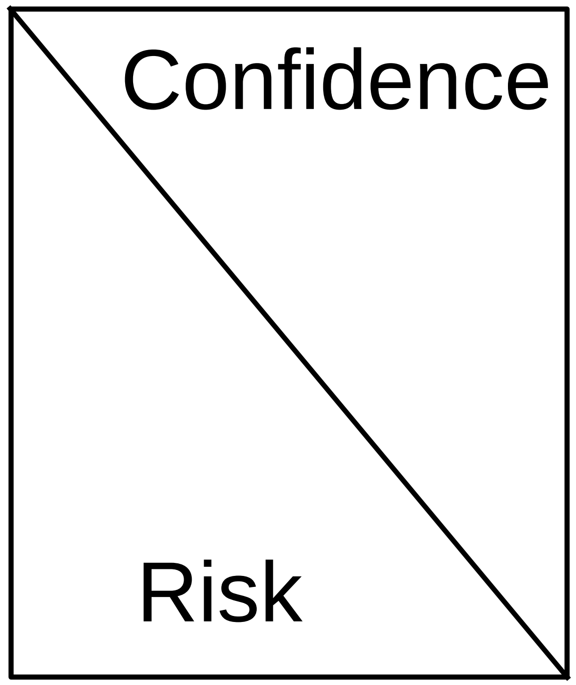


Purposes of SRE Maturity Model

- Provide SRE implementers with guideline for staging SRE deployment, relating benefits realized with level of SRE process improvement
- Provide SRE implementers with guideline for *self*-assessment

SRE Maturity Model Levels

- Optimized Level (5)
- Measured Level (4)
- Quality-aware Level (3)
- Use-aware Level (2)
- Ad Hoc Level (1)



Definition of Levels

Activity	Level				
	1				
Set FIOs	Remove bugs				
Engineer strategies	in time available				
Develop, apply OPs	Uniform				
Test emphasis	Feature				

Definition of Levels - Notes

- “Uniform” means resources (effort, test cases, test time, etc.) are divided among all functions equally, except for critical functions
- Feature test executes all the test cases independently of each other
- Load test executes all the test cases together, with all the many combinations of interactions and different conditions of the field environment
- Prerequisites to attaining Level 2:
  - Awareness activities, training, planning, acquiring SRE estimation program
  - Listing associated systems

Definition of Levels

Activity	Level				
		2			
Set FIOs	Remove bugs				
Engineer strategies	in time available				
Develop, apply OPs		E			
Test emphasis		Load			

E = Estimated as necessary

Definition of Levels

Activity	Level				
			3		
Set FIOs			E		
Engineer strategies			E		
Develop, apply OPs			E		
Test emphasis			Load		
Track rel. growth			M		

E = Estimated as necessary, M = Measured in test

Definition of Levels

Activity	Level				
				4	
Set FIOs				F1	
Engineer strategies				F1	
Develop, apply OPs				E	
Test emphasis				Load	
Track rel. growth				M	
Certification test				M	
Measure field FI				F1	

E = Estimated as necessary, M = Measured in test , F1 = Using field data from similar release or project

Definition of Levels

Activity	Level				
					5
Set FIOs					FM
Engineer strategies					FM
Develop, apply OPs					F1
Test emphasis					Load
Track rel. growth					M
Certification test					M
Measure field FI					F1
Measure field OPs					F1

M = Measured in test, F1 = Using field data from similar release or project,
FM = Using field data from multiple projects

Definition of Levels

Activity	Level				
	1	2	3	4	5
Set FIOs	Remove bugs in time available		E	F1	FM
Engineer strategies			E	F1	FM
Develop, apply OPs	Uniform	E			F1
Test emphasis	Feature	Load			
Track rel. growth			M		
Certification test				M	
Measure field FI				F1	
Measure field OPs					F1

E = Estimated as necessary, M = Measured in test, F1 = Using field data from similar release or project, FM = Using field data from multiple projects

Benefits of SRE Maturity Model

- Guide to adopting SRE by stages
- Framework for characterizing your practice
- Understand your practice better in relation to best practice
- Can help you communicate with management, get support for change

Productivity Improvement

Level	Improvement
1	0
2	++
3	+++
4	++++
5	+++++

Confidence Improvement- Matching Quality Characteristics

Level	Improvement
1	0
2	0
3	+
4	++
5	+++

What SRE Does for Testers

- Reduces end of project time squeeze through better, earlier planning
- Increases productivity and efficiency by focusing efforts with operational profile
- Clarifies user needs and makes test more user-focused by participation in system engineering
- Reduces risk by providing greater visibility into test process and its current status
- Gives concrete powerful tool for arguing for more time or money for test when needed
- Makes you more competitive

To Explore Further

- Software Reliability Engineering website: lots of useful material including bibliography of articles by software reliability engineering users, course information:

  http://members.aol.com/JohnDMusa/
- Musa, J. D., *Software Reliability Engineering: More Reliable Software, Faster Development and Testing*, ISBN 0-07-913271-5, McGraw-Hill, 1998.
- Lyu, M. (Editor), *Handbook of Software Reliability Engineering* , ISBN 0-07-039400-8, McGraw-Hill, 1996 (includes CD/ROM of CASRE software reliability engineering estimation program).

To Explore Further

- Musa, Iannino, Okumoto; *Software Reliability: Measurement, Prediction, Application,* ISBN 0-07-044093-X, McGraw-Hill, 1987.
- Musa, J. D., “Software Reliability Engineering,” *Duke Distinguished Lecture Series Video*, University Video Communications, 415-813-0506.
- IEEE Computer Society Technical Committee on Software Reliability Engineering (publishes newsletter, sponsors ISSRE annual international conference): membership application at http://www.tcse.org/tcseform.html

To Explore Further

- Electronic mailing list: send email to:

  sw-rel@computer.org

  - Subscribe: put ONLY “subscribe” in body of message
  - Post (you must first subscribe): put text to be posted in body

How Testers Can Use a Software Reliability Engineering (SRE) Maturity Model

John D. Musa
j.musa@ieee.org

Recent rapid growth of interest in the application of software reliability engineering (SRE) (Musa 1998a, Musa 1998b, Tierney 1997) has raised the issue of how to best deploy this very promising practice. The rapid growth of interest has come about because SRE helps solve the most important software development problem, making you and your organization more competitive, by *quantitatively* planning and guiding software development and test to meet user needs.

Experience in helping many organizations deploy SRE has convinced me that a staged deployment has many advantages. Change, no matter how beneficial, always causes some disruption. A staged deployment lets you control the rate of change and hence the degree of disruption by spreading the deployment over a period of time. It gives practitioners more time to learn SRE and to integrate it with the development process they are using. And it helps practitioners predict how the benefits they realize should increase as more and more of SRE is deployed.

Thus there is a need to determine a set of levels of deployment, with each step being part of an overall plan. Each step should consist of an integrated set of SRE tasks that function together to yield some concrete benefits and reflect a higher level of maturity in using SRE. The levels have been determined from a careful delineation and analysis of the SRE process, and the interdependencies and benefits associated with various process tasks. This paper outlines the levels, providing the basis for a project to plan a staged deployment. It also furnishes to projects the means for self-assessing their present status in deploying SRE.

First we will examine what SRE is and the we will outline the SRE process. After defining the Software Reliability Engineering Maturity Model, we will show how maturity level affects project productivity and improvement in the confidence level with which you meet user needs for the major quality characteristics of reliability, availability, delivery time, and life-cycle cost.

What is SRE?

SRE reduces the risks of unreliability or unavailability (in the sense of service) of a released product, missed schedules, and cost overruns. These risks can all lead to loss of market share and profitability.
Reliability is the probability that a system or a capability of a system functions without failure for a specified time in a specified environment. *Availability* is the probability at any

given time that a system or a capability of a system functions satisfactorily in a specified environment. For a given average down time per failure, availability implies a certain reliability. A *failure* is a departure of system behavior in execution from user needs. *Failure intensity* is simply the number of failures per unit time. Because of its simplicity, failure intensity is often used instead of reliability in the software reliability engineering field.

SRE improves all the major quality characteristics (reliability, availability, delivery time, and life cycle cost) of the product by quantitatively characterizing expected use and employing this information to precisely focus project resources on the most used and/or most critical functions or modules. Also, the expected use information makes test more effective by creating a test environment that realistically represents field conditions.

SRE matches major quality characteristics to user needs more precisely by setting quantitative objectives for availability and/or reliability as well as schedule and life cycle cost, and engineering project strategies to meet these objectives. Then it tracks reliability of each system in test against its objective as one of the product release criteria.

SRE is a proven, standard, widespread best practice that is widely applicable (Musa 1998a, Musa 1998b). As an example, Tierney (1997) reported the results of a survey taken in late 1997 that showed that Microsoft has applied software reliability engineering in 50 percent of its software development groups, including projects such as Windows NT and Word. The benefits they observed were increased test coverage, improved estimates of amount of test required, useful metrics that helped them establish ship criteria, and improved specification reviews.

SRE is low in cost and its deployment has only minor schedule impact. It encourages greater communication among different project roles. This is particularly true for testers and systems engineers. When SRE is implemented, testers typically participate as members of the system engineering team. They help develop operational profiles, set failure intensity objectives, and select project reliability strategies.

SRE is very customer-oriented. It involves direct interaction with customers, and this enhances your image as a supplier if you have any reasonable degree of competence, improving customer satisfaction. SRE is highly correlated with attaining Levels 4 and 5 of the SEI Capability Maturity Model.

SRE Process

The SRE process is illustrated in Figure 1. There are six principal activities. The software development process is shown side by side with the SRE process, so you can relate the activities of one to those of the other. Both processes follow spiral models, but the feedback paths are not shown for simplicity.

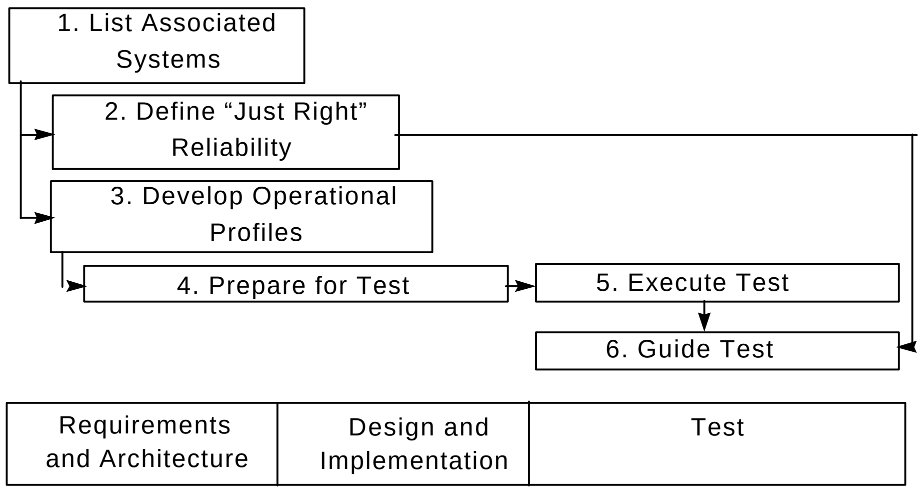

1. List Associated Systems
2. Define “Just Right” Reliability
3. Develop Operational Profiles
4. Prepare for Test
5. Execute Test
6. Guide Test
Requirements and Architecture
Design and Implementation
Test

Figure 1. SRE Process

We will illustrate the SRE process with Fone Follower, an example adapted from an actual project at AT&T. The name and certain details were changed to keep the explanation simple and protect proprietary data. Subscribers to Fone Follower call and enter the phone numbers to which they want their calls forwarded as a function of time. Incoming calls (voice or fax) from the network to the subscriber are forwarded in accordance with the program the subscriber entered. Incomplete voice calls go to the subscriber’s pager (if the subscriber has one) and then to voice mail.

The first activity is to list all the systems that are associated with the product that for various reasons must be tested independently.

To define the “just right” level of reliability for the product, you set the failure intensity objective (FIO) for each associated system as part of the requirements, balancing among major quality characteristics users need. To determine this need you will require field data for a similar release or product. This data includes customer satisfaction surveys related to measured failure intensity, and an analysis of competing products. Then you engineer project software reliability strategies to meet these objectives.

An *operational profile* is a complete set of functions with their probabilities of occurrence. Table 1 shows an illustration of an operational profile from Fone Follower.

You can use operational profiles in system engineering to reduce the number of operations to those that are cost effective with respect to life-cycle system costs and benefits, to plan a competitive release strategy (schedule a small number of most-used operations for a speeded-up first version and defer the others to a later version), and to focus resources on the functions and modules that are most used or most critical.

Operation	Occurrence Probability
Proc. voice call, no pager, ans.	0.18
Proc. voice call, no pager, no ans.	0.17
Proc. voice call, pager, ans.	0.17
Proc. fax call	0.15
Proc. voice call, pager, ans. on page	0.12
Proc. voice call, pager, no ans. on page	0.10
Phone number entry	0.10
Audit sect. - phone number data base	0.009
Add subscriber	0.0005
Delete subscriber	0.0005
Recover from hardware failure	0.000001
Total	1

Table 1. Fone Follower Operational Profile

A very important use of the operational profile (plus criticality information) is in allocating test cases to operations during the Prepare for Test activity. For example, in Fone Follower we allocated 15% of the test cases to the Process fax call operation. We also use the operational profile and criticality information to allocate test time among operations during the Execute Test activity.

Execute Test includes feature test, load test, and regression test. *Feature test* executes all the test cases independently of each other. *Load test* executes all the test cases together, with all the many combinations of interactions and different conditions of the field environment. *Regression test* executes some (including all critical operations) or all of feature test after each system build with significant change; it is designed to reveal failures caused by faults introduced by program changes.

During the Guide Test activity, we apply failure data collected during Execute Test to track reliability growth and conduct certification tests. We track reliability growth of developed software using a software reliability estimation program. Reliability growth test couples test with attempts to remove faults and hence cause reliability growth. You input failure data that you collect during reliability growth test to a reliability estimation program such as CASRE (Lyu 1996). Normalize the data by multiplying by the failure intensity objective in the same units. Execute this program periodically and plot the FI/FIO ratio as shown in Figure 2 for Fone Follower.

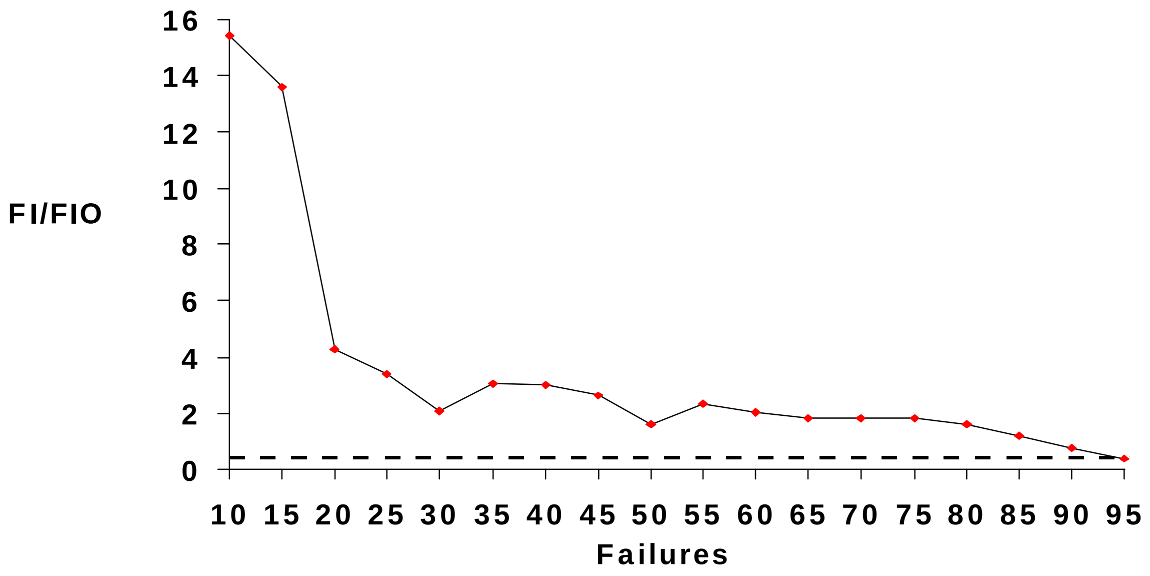
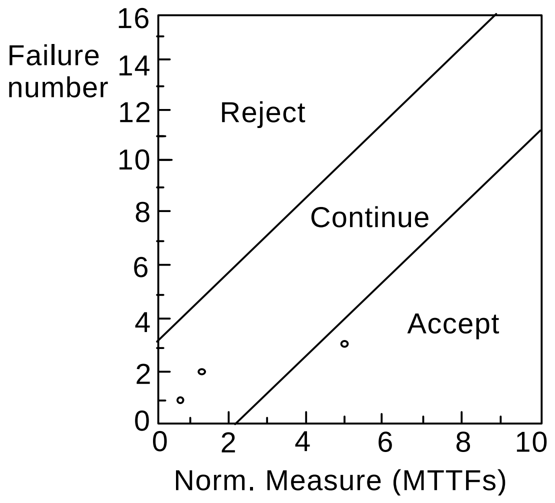


Figure 2. Plot of FI/FIO Ratio

Note that the FI/FIO ratio is close to 16 early in system test, a not unusual situation. The failure intensity at this point in test is almost 16 times what users want it to be in the released product. This is not a cause for concern unless time remaining for system test is short. The two substantial upward swings that you can see in the ratio may signify a problem that needs your attention. The most common causes are system evolution, which may indicate poor change control, and changes in test selection probability, which may indicate a poor test process. In theory, when the FI/FIO ratio reaches 1, you should consider release. However, we wait until it reaches 0.5 to allow for the error involved in any statistical estimation. Also, we require that essential documentation be complete and that outstanding high severity failures be resolved (the faults causing them have been removed).

We also certification test (accept or reject) certain systems such as some acquired components. For certification test you also first normalize failure data by multiplying by the failure intensity objective, yielding a measure in MTTFs. Plot each new failure as it occurs on a reliability demonstration chart as shown in Figure 3. Note that the first two failures fall in the Continue region, which signifies that there is not enough data to reach an accept or reject decision. The third failure falls in the Accept region, which means that you can accept the software, subject to the levels of risk associated with the chart you are using.



FAIL. NO.	MCALLS AT FAILURE	MTTFs
1	0.1875	0.75
2	0.3125	1.25
3	1.25	5

Failure intensity objective:
4 failures / Mcalls

Figure 3. Reliability Demonstration Chart

What is the SRE Maturity Model?

As we have already pointed out, the principal reasons for creating the SRE Maturity Model were to:
1. Provide SRE implementers with a guideline for determining how to stage the deployment of SRE, by relating a given deployment stage or level with the amounts of improvement expected in productivity and confidence in matching the balance among major quality characteristics users need.
2. Provide SRE implementers with a guideline for *self*-assessment.

The levels defined for the SRE Maturity Model are:

- Optimized (5)
- Measured (4)
- Quality aware (3)
- Use aware (2)
- Ad hoc (1)

The detailed definitions of the levels are illustrated in Figure 4. Level 1 represents software development without SRE. Projects are allotted a certain amount of test time and are expected to remove as many faults or bugs as possible during that time. Test emphasis is on feature test. There is either no load test, or load test that is very limited in representing field conditions accurately. "Uniform" in Figure 4 means resources (effort, test cases, test time, etc.) are divided among all functions equally, except for critical functions.

Activity	Level				
	1	2	3	4	5
Set FIOs	Remove bugs		E	F1	FM
Engineer Strategies	in time available		E	F1	FM
Develop, apply OPs	Uniform	E			F1
Test emphasis	Feature	Load			
Track rel. growth			M		
Certification test				M	
Measure field FI				F1	
Measure field OPs					F1

E = Estimated as necessary, M = Measured in test, F1 = Using field data from similar release or project, FM = Using field data from multiple projects

Figure 4. Definitions of SRE Maturity Model Levels

In order to attain Level 2, an organization must conduct awareness activities and training. Awareness activities typically involve the presentation of an overview of about two hours to an organization by an SRE expert, someone who has experience in helping projects implement SRE. The presentation should be very interactive, with about half the time devoted to questions and comments. Practitioners need to explore how SRE can help them. Training generally involves a two day course in which participants learn the SRE process in detail. The course should incorporate workshops in which participants apply what they have learned to their own projects. Thus they learn how to integrate SRE into their own development processes. The organization must plan the SRE deployment and acquire an SRE estimation program such as CASRE (Lyu 1996).

To reach Level 2, you must implement the development of operational profiles for the associated systems you have listed, using measured field data whenever it is available, but making estimates when necessary. You must also apply the operational profiles. This involves employing them in system engineering to reduce the number of low use operations and to plan a competitive release strategy. In test, you apply them in allocating both test cases and test time to operations. And throughout the project, you use operational profiles to allocate development resources to different functions, resulting in greater efficiency by focusing on the functions that receive the greatest use (critical functions also receive special attention). Although your reliability strategy is still the simple one of removing as many bugs (faults) as possible in the time available to you, the primary emphasis in test shifts from feature test to load test. At Level 2, you are "Use aware."

In order to move to Level 3, you must implement those activities of SRE that set failure intensity objectives, engineer strategies for meeting them, and track reliability growth, using failure data collected during system test. By dealing with reliability quantitatively, as one of the major quality characteristics that you fit to user needs, you have become "Quality aware" in the most sophisticated sense.

Reaching Level 4 requires that you measure failure intensity of your product in the field and use this data for setting failure intensity objectives and engineering reliability strategies for the next product release. Hence Level 4 is known as the "Measured" level. You also start employing certification test as appropriate; for example, for certain acquired components.

Finally, you attain the "Optimized" level , Level 5, by even more sophisticated collection and use of field data. You collect failure intensity and development strategy information on multiple projects and use this to extract guidelines for future projects. For example, a guideline for most rapidly and inexpensively attaining a particular range of failure intensity objectives might be to devote 50 percent of your resources to fault prevention, 30 percent to fault removal, and 20 percent to fault tolerance. There might also be more detailed guidelines, such as allocating 8 staff hours per thousand source lines for requirements inspection for a given failure intensity objective. At Level 5, you also measure the actual operational profiles experienced in the field and use this information to improve the operational profiles for the next release of the product.

Effects Of SRE Maturity Level

The productivity improvement and the improvement in confidence in matching quality characteristics resulting from an increase in level is shown in Figure 5. The plus signs indicate the cumulative amount of improvement.

Level	Improvement	
	Productivity	Confidence
1	0	0
2	++	0
3	+++	+
4	++++	++
5	+++++	+++

Figure 5. Effects of SRE Maturity Level

The SRE Maturity Model provides a framework for characterizing your practice. With it, you can understand your practice better in relation to the best practice. It provides a guide to adopting SRE by stages. And it can help you communicate with management and get support for change.

A related model, the Testing Maturity Model, has been recently proposed for testing (Burnstein, Homyen, Grom, and Carlson (1998)). Some of the concepts of the Testing Maturity Model were derived from Gelperin and Hetzel's Evolutionary Testing Model (Gelperin and Hetzel 1988). And obviously, the SRE maturity model owes some of its concepts to the SEI Capability Maturity Model (Humphrey 1989), which deals with the entire software development process.

For testers, SRE reduces the end of project time squeeze with better, earlier planning. It increases productivity and efficiency by focusing efforts with the operational profile. SRE clarifies user needs and makes test more user-focused by having testers participate in system engineering. It reduces risk by providing greater visibility into the test process and its current status. SRE provides a concrete tool for arguing for more time or money for test when needed. Overall, SRE is a powerful practice that makes you more competitive.

Acknowledgements

I am indebted to Dr. Nancy Eickelmann of the NASA Independent Verification and Validation Facility, Software Research Laboratory, Fairmont, WV and Dr. Bill Everett, Software Process and Reliability Engineering, Inc., Albuquerque, NM for their very helpful reviews and many comments. I also wish to thank Dr. David Gelperin, Software Quality Engineering, for issuing the challenge that led to this work.

References

Burnstein, Homyen, Grom, and Carlson. 1998. "A model to assess testing process maturity." *Crosstalk,* Nov. 1998, pp. 26-30.

Gelperin, D. and B. Hetzel. 1988. "The growth of software testing," *CACM,* Vol. 31, No. 6, pp. 687-695.

Humphrey, W.S. 1989. *Managing the Software Process,* ISBN 0-201-18095-2, Addison-Wesley, Reading, MA.

Lyu, M. (Editor). 1996. *Handbook of Software Reliability Engineering*, ISBN 0-07-039400-8, McGraw-Hill, New York (includes CD/ROM of CASRE program).

Musa, J. D. 1998a. *Software Reliability Engineering: More Reliable Software, Faster Development and Testing*, ISBN 0-07-913271-5, McGraw-Hill, New York.

Musa, J. D. 1998b (updated regularly). *Software Reliability Engineering* website: overview, briefing for managers, bibliography of articles by software reliability engineering users, course information, useful references, Question of the Month:
http://members.aol.com/JohnDMusa/

Musa, J.D., A. Iannino, and K. Okumoto. 1987. *Software Reliability: Measurement, Prediction, Application*, ISBN 0-07-044093-X, McGraw-Hill, New York.

Tierney, J. 1997. *SRE at Microsoft*. Keynote speech At 8th International Symposium On Software Reliability Engineering, November 1977, Albuquerque, NM.

John D. Musa

John D. Musa is an Independent Senior Consultant in software reliability engineering. He has more than 30 years' experience as software practitioner and manager in a wide variety of development projects. He is one of the creators of the field of software reliability engineering and is widely recognized as the leader in reducing it to practice. He was recently Technical Manager of Software Reliability Engineering (SRE) at AT&T Bell Laboratories, Murray Hill, NJ.

He has been involved in SRE since 1973. His many contributions include the two most widely-used models (one with K. Okumoto), the concept and application of the operational profile, and the integration of SRE into all phases of the software development cycle. Musa has published some 100 articles and papers, given more than 175 major presentations, and made a number of videos. He is principal author of the widely-acclaimed pioneering book, *Software Reliability: Measurement, Prediction, Application* and author of the new book, *Software Reliability Engineering: More Reliable Software, Faster Development and Testing.*

He organized and led the transfer of SRE into practice within AT&T, spearheading the effort that defined it as a "best current practice." He was actively involved in research to advance the theory and practice of the field. Musa has been an international leader in its dissemination.

Listed in *Who's Who in America* and *American Men and Women of Science*, he is an international leader in software engineering and a Fellow of the IEEE, cited for "contributions to software engineering, particularly software reliability". He was recognized in 1992 as the individual who had that year contributed the most to testing technology. He was co-founder of the IEEE Committee on SRE. He has very extensive international experience as a lecturer and teacher.