Chap 4. Software Reliability

4.4 Software Reliability Engineering (SRE)

1. Introduction
2. SRE Process
3. Reliability Targets
4. Operational Profile
5. Reliability Validation
6. Example

1. Introduction

-Software reliability engineering (SRE) is the prediction, measurement, and management of software intensive systems, with the purpose of achieving the highest customer satisfaction possible.

÷SRE can achieve significant reduction in testing and maintenance cost, better resource and schedule control, and improved customer satisfaction.

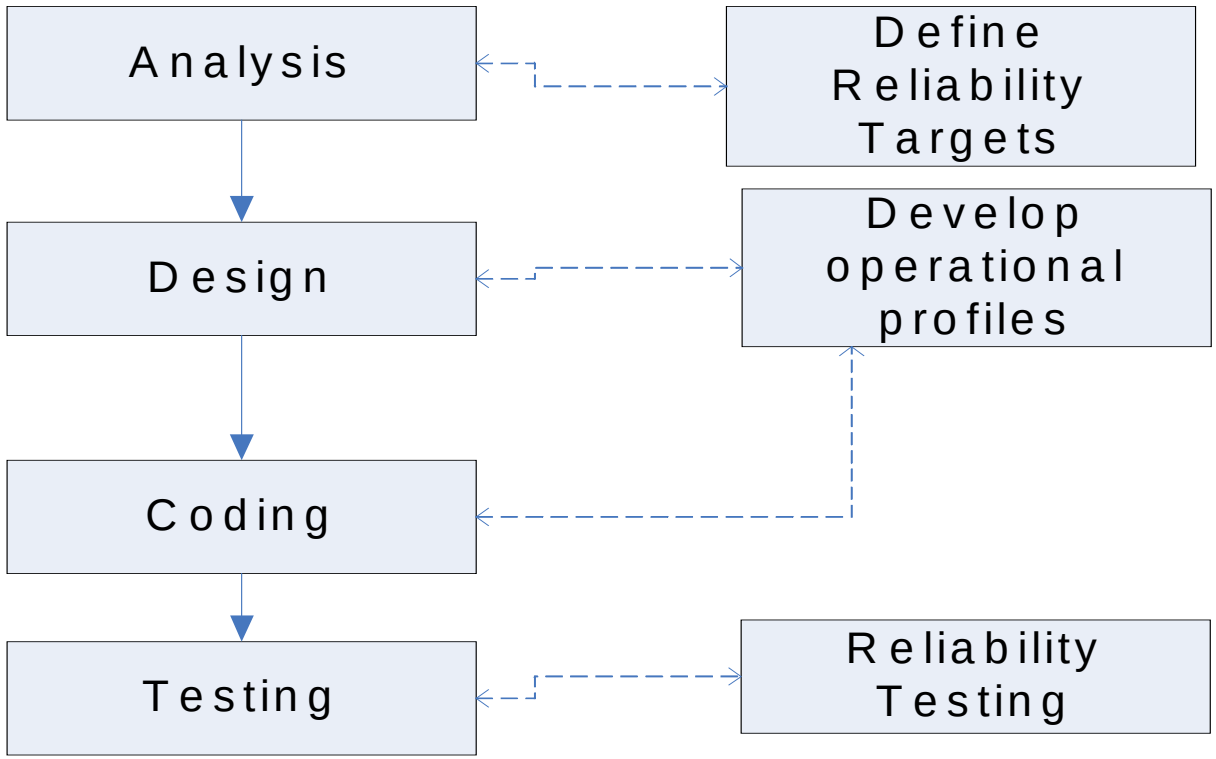
-Software Reliability Engineering (SRE) is a multi-faceted discipline covering the software product lifecycle. It involves both ***technical*** and ***management*** activities in three target areas:

÷Software reliability measurement, prediction, and estimation.

÷Use of reliability target to guide software development process and maintenance.

÷Study of the impact of software reliability metrics and activities on operational software behavior.

2. SRE Process



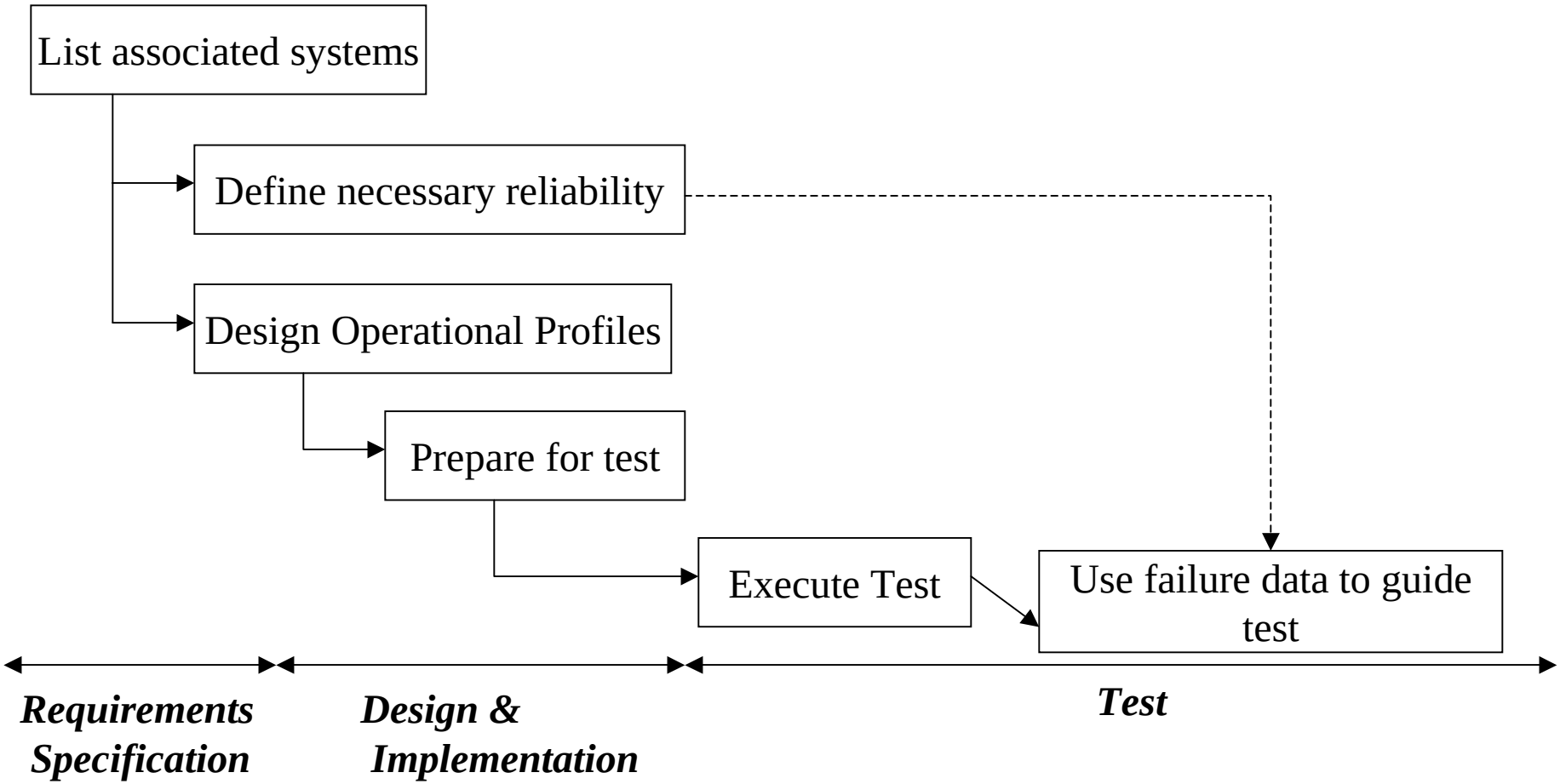
- Typical SRE process involves three main steps:

÷*Reliability goals definition*: consists of capturing the customer expectations in terms of reliable product or operations.

÷*Operational profiles definition*: consists of identifying and specifying the **probable** patterns of usage of the software.

÷*Reliability testing*: the operational profiles are used to design test cases and drive the reliability validation process.

Detailed SRE Process



-Listing associated systems (of product) involves defining:

÷the base product,

÷the major variations of the base product

÷the frequently used components of the based product or variations.

3. Reliability Targets

-This involves:

1. Defining consistently the notion of failure for the software product.
2. Choosing common reference measure for all failure intensities (e.g., failures per some natural unit or per hour).
3. Setting system Failure Intensity Objective (FIO) for each associated system (software/hardware)
4. For any software you develop:
   A. Find developed software FIO by subtracting the total of the FIOs of all hardware and acquired software components from the system FIOs.
   B. Choose software reliability strategies to meet developed software FIO and schedule objectives with lowest development cost (i.e., Fault prevention, Fault removal, Fault tolerance, Fault/failure forecasting)

Example 4.3.1: Various FIO may be defined for various parts of a system:

-System failure intensity objective μ= 30 failure/1,000,000 transactions
-MTTF for OS is 3,000 hours for 10 million transactions
-MTTF for hardware is 1 per 30 hours of operation

One must define a unique scale for all FIOs

To compute failure intensity objective for the *developed software*:

1. Set FIO for the whole system
2. Set a common measurement unit for failure intensity for the whole system
3. Subtract expected failure intensity for acquired components from the FIO.
4. Subtract expected failure intensity for the environment (OS, interface systems) that the developed software will run on
5. The remaining will be failure intensity objective for the developed software components.

Example 4.3.2: System FIO = 100 failure/1,000,000 transactions.
Failure intensity for hardware = 0.1 failure/hour
OS failure for a load of 100,000 transactions = 0.4 failure/hour
Therefore, developed software FIO = 95 failure/1,000,000 transactions

Example 4.3.3: Database system running on Win 2K
System failure intensity objective = 30 failure/1,000,000 transactions
MTTF for Win 2K is around 3,000 hours for 10 million transactions
Average hardware failure is 1 per 30 hours
Failure rate for other systems is 9 for one million transactions
What is FIO for the developed software?

Solution: $\mu_{developed\_software}$=300-191=109 for 10,000,000 transactions

4. Operational Profile

-The operational profile of the software reflects how it will be used in practice.

÷It consists of a specification of classes of input and the probability of their occurrence.

÷It is relatively easy to assess the operational profile for a new software system that is replacing an existing manual or automated system.

÷However, when the system is new and innovative it is more difficult to anticipate how it will be used.

-The operational profile is based on the characteristics of the software environment, the functions used, their inputs and outputs, and their usage frequencies.

-Defining the operational profile involves identifying major system operations, their occurrence probabilities and their initiators:

1. Identify initiators of operations (i.e., user types, external systems, and the system itself)
2. Identify and list the operations invoked by each initiator
3. Determine occurrence rates of the operations
4. Determine occurrence probabilities by dividing the occurrence rates by the total occurrence rate

Example 4.3.4 - FONE FOLLOWER (FF)
Product Description

1. Subscriber calls FF, enters planned phone numbers (forwardees) to which calls are to be forwarded vs. time.
2. FF forwards incoming calls (voice or fax) from network to subscriber as per program. Incomplete voice calls go to pager (if subscriber has one) and then voice mail.

-***Operation*:** major system logical task performed for initiator, which returns control to system when complete.

Illustrations - FF: Process fax call, Phone number entry, Audit section of phone number database

-***Operational profile (OP):*** complete set of operations with probabilities of occurrence

Illustration - FF:

Operation	Occur. Prob.
Process voice call, no pager, ans.	0.21
Process voice call, pager, ans.	0.19
Process fax call	0.17
Process voice call, pager, ans. on page	0.13
÷	
÷	
÷	

Operational Profile Design

-The design of the operational profile involves some intermediary profiles which are ultimately compounded. These include:

÷*Customer profiles*: describe individuals, groups of persons, or organizations acquiring the system.

÷*User profiles*: describe individuals, groups of persons, or organizations employing the system.

÷*System mode profiles*: describe sets of related business functions or operations.

÷*Functional profiles*: divide the system into a set of user-oriented functions or operations with corresponding usage frequencies.

-The design of the operational profile consists of collecting and analyzing the different conditions or modes under which the software is used.

÷An operational profile is defined for each mode, and can be based on user profiles or other considerations such as risk or business criticality.

÷An operational profile combines a collection of functional profiles underlying specific usage scenarios, corresponding software commands and input states, and associated probabilities

Example 4.3.5: Operational profile for a telephone switch system

Customer profile

Customer group	**Probability**
Local carrier	0.7
Inter-city carrier	0.3

User profile

User group	**Local (0.7)**		**Inter-city (0.3)**		**Total**
	within	total	within	total	
Households	0.6	0.42	0.2	0.06	0.48
Businesses	0.3	0.21	0.6	0.1	0.39
Emergency services	0.05	0.035	0.001	.0003	.0353
Other	0.05	0.035	0.199	0.0597	0.0947

System Mode Profiles

÷Assume that 99% of inter-city and 90% of local household traffic is voice, while only 70% of business traffic is voice for both customer categories.

÷Assume that system administration accounts for 30% and maintenance for 70% of the “Other” user category, while the rest of the traffic is DATA.

Mode	**Probability**
Voice (personal)	0.4374
Voice (business)	0.273
Data	0.1596
Emergency	0.0353
System admin.	0.02841
Maintenance	0.06629

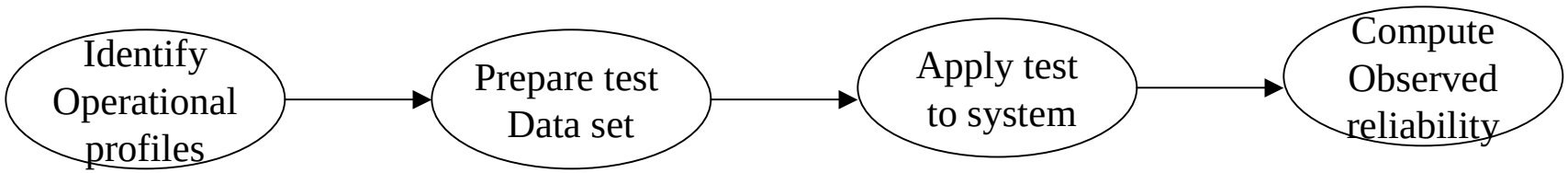
5. Reliability Validation

Overview

-Reliability validation is achieved through ***statistical testing***.

÷This contrasts with ***defect testing*** whose aim is to discover system faults.

÷The aim of statistical testing is to assess system reliability.



-The process of validating the reliability of a system involves the following steps:

1. *Identify operational profiles:* an operational profile identifies different classes of system inputs and the probability of these inputs in normal use.
2. *Prepare test data:* a set of test data is constructed that reflect the operational profile.
3. *Apply test to system:* the system is tested with these data and the number of failures are also logged.
4. *Compute reliability:* after a statistically significant number of failures is observed, the software reliability can then be computed.

-However, statistical testing is not easy to apply in practice. The principal difficulties include:

÷Operational profiles may not be an accurate reflection of the real use of the system.

÷Defining a large amount of test data may be time consuming and expensive if it is not possible to generate this data automatically.

÷It is important to generate a statistically significant number of failures to allow accurate reliability measurements.

Types of Test

- ***Certification Test:*** Accept or reject (binary decision) an acquired component for a given target failure intensity.
- ***Feature Test:*** A single execution of an operation with interaction between operations minimized.
- ***Load Test:*** Testing with field use data and accounting for interactions
- ***Regression Test:*** Feature tests after every build involving significant change, i.e., check whether a bug fix worked.

Sequence of System Test

1. ***Acquired components:*** Certification test only
2. ***Developed product:*** Feature test and then load test for a new product; Feature test, and then regression test for subsequent releases
3. ***Other systems interface:*** Load test only

-It is possible to change this sequence or testing different systems in parallel.

Test Preparation

-Use the operational profiles to prepare test cases and test procedures.

-This involves the following activities:

1. Specify new test cases for new operations
   A. Distribute new test cases to new operations based on operational profile

   Illustration - FF:

   Allocate 17% of test cases to Proc. fax call operation

   B. Detail new test cases for each new operation by selecting from possible choices of input variable values with equal probability

   Illustration - FF:

   Forwardee = Local calling area

2. Specify test procedure, based on the test operational profile. The test procedure is the controller that invokes test cases during execution.

Test Execution

-This involves the following tasks:

1. Determine and allocate test time among associated systems and types of test (i.e., feature, load, regression)
2. Invoke test in accordance with operational profile: this involves choosing operations randomly in accordance with the operational profile.
3. Identify system failures and when they occurred
4. Use data to guide testing.

-In the reliability growth test, perform *feature test* first and then *load test*. Then conduct *regression test* after each build that has a significant change.

÷Invocation of test cases should occur at random times.

÷In feature test select in random order from the set of all new test cases plus the regression test cases of the previous release.

÷In load test, invoke each operational mode for its allocated proportion of time. The number of test cases invoked will be determined based on the time the operational mode runs and its occurrence rate.

-Test case selection should be with replacement for test cases in load test but not in feature or regression test.

÷In feature or regression test, the runs are much less likely to be different because indirect input variables are tightly controlled.

÷In load test, the resulting runs will almost be different, because the indirect variables can vary, causing to explore the failure behavior of different runs.

÷Repetitions are inefficient. But in load test, the number of runs is so large that the probability of wasting test resources by repeating many runs is infinitesimal.

- Therefore, in load test, after each test, replace the element in the population, allowing reselection.

-As operations may be associated with multiple faults, if selection is performed without replacement, the operation can be selected only once therefore reducing the probability of finding another bug.

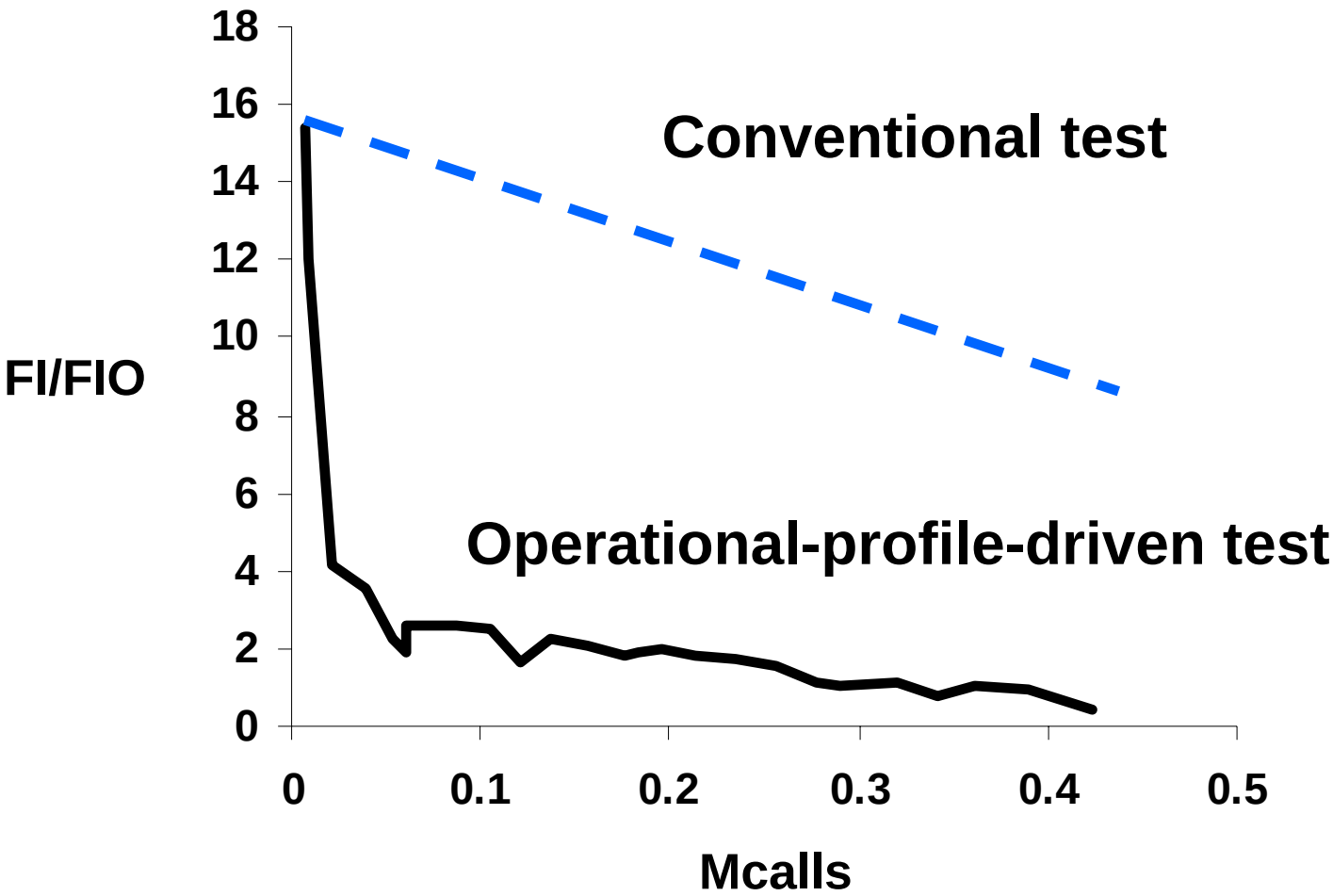
Use of Failure Data To Guide Test

-Process system failure data collected from the test to direct the testing process. This involves:

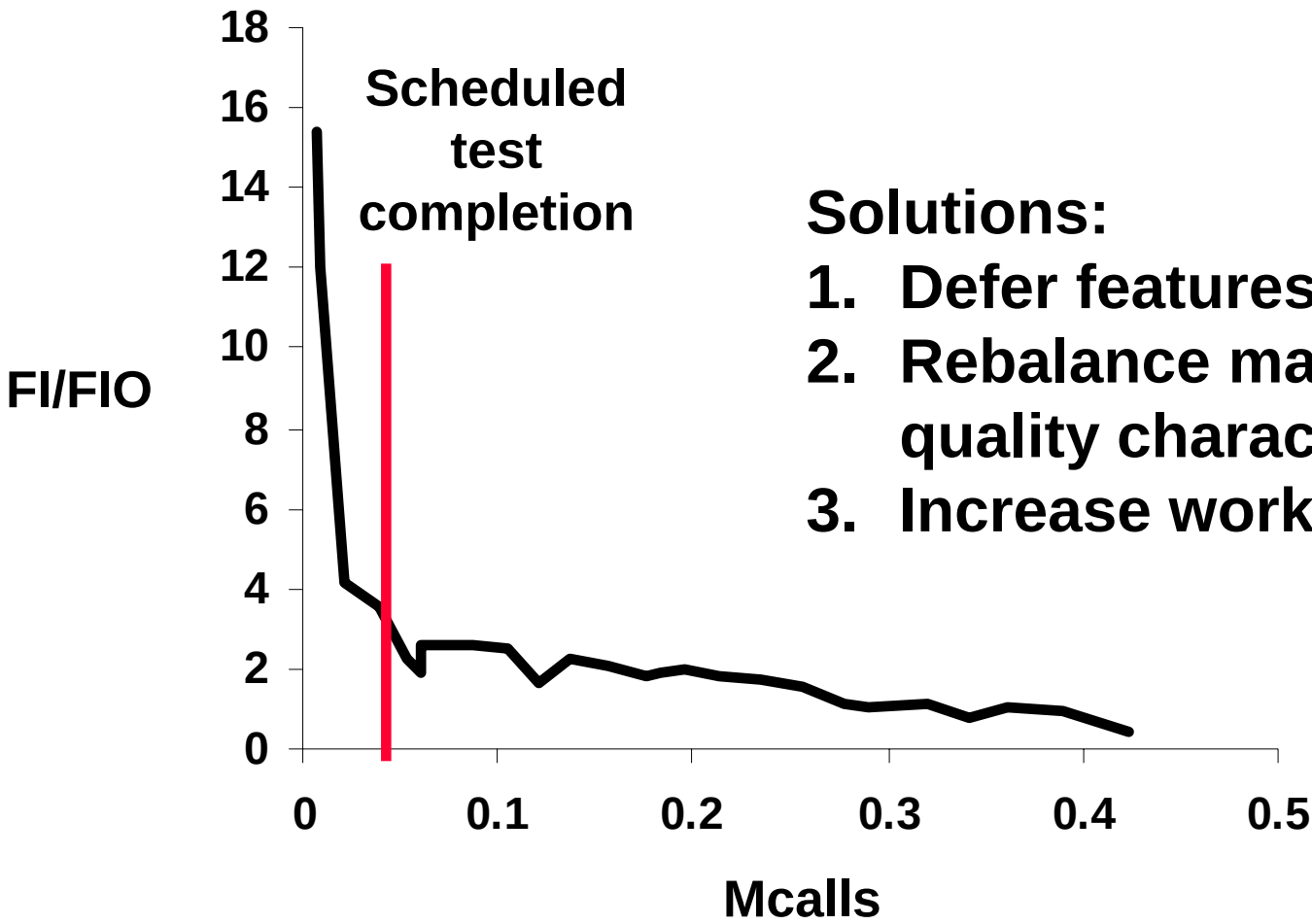
1. Tracking reliability growth, as faults are removed, of developed software of base product and variations:
   a. Estimate FI / FIO ratio
   b. Plot FI / FIO ratio against time: Plot each new failure as it occurs on a reliability demonstration chart.
   c. Interpret plot
2. Certifying reliability of base product and variations that customers will acceptance test: accept or reject software (operations) using reliability demonstration chart.
3. Guiding product release

Example 4.3.6: Track Reliability Growth

Interpret Plot : Illustration - FF



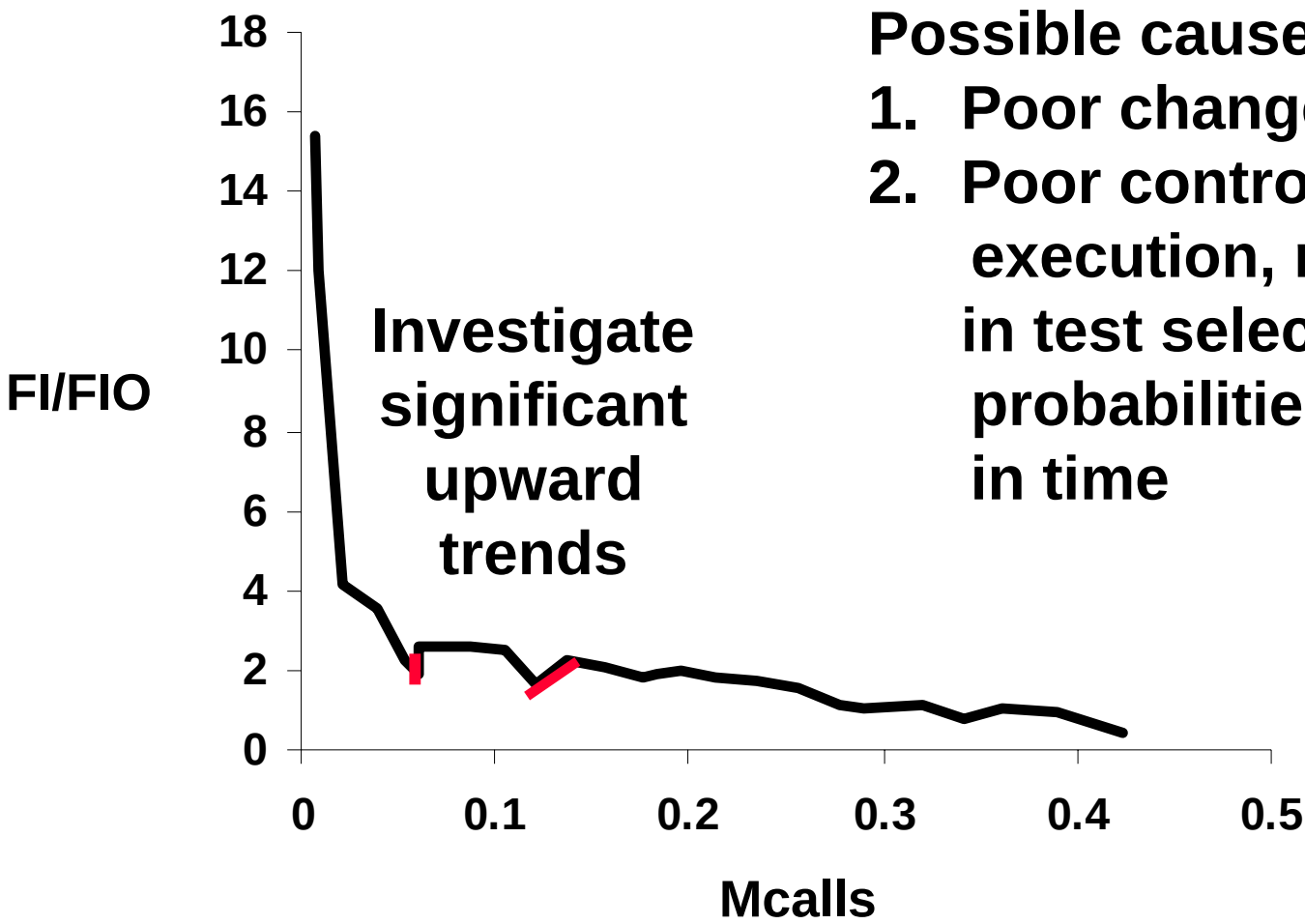
Interpret Plot : Illustration - FF



Solutions:

1. **Defer features**
2. **Rebalance major quality characteristics**
3. **Increase work hours**

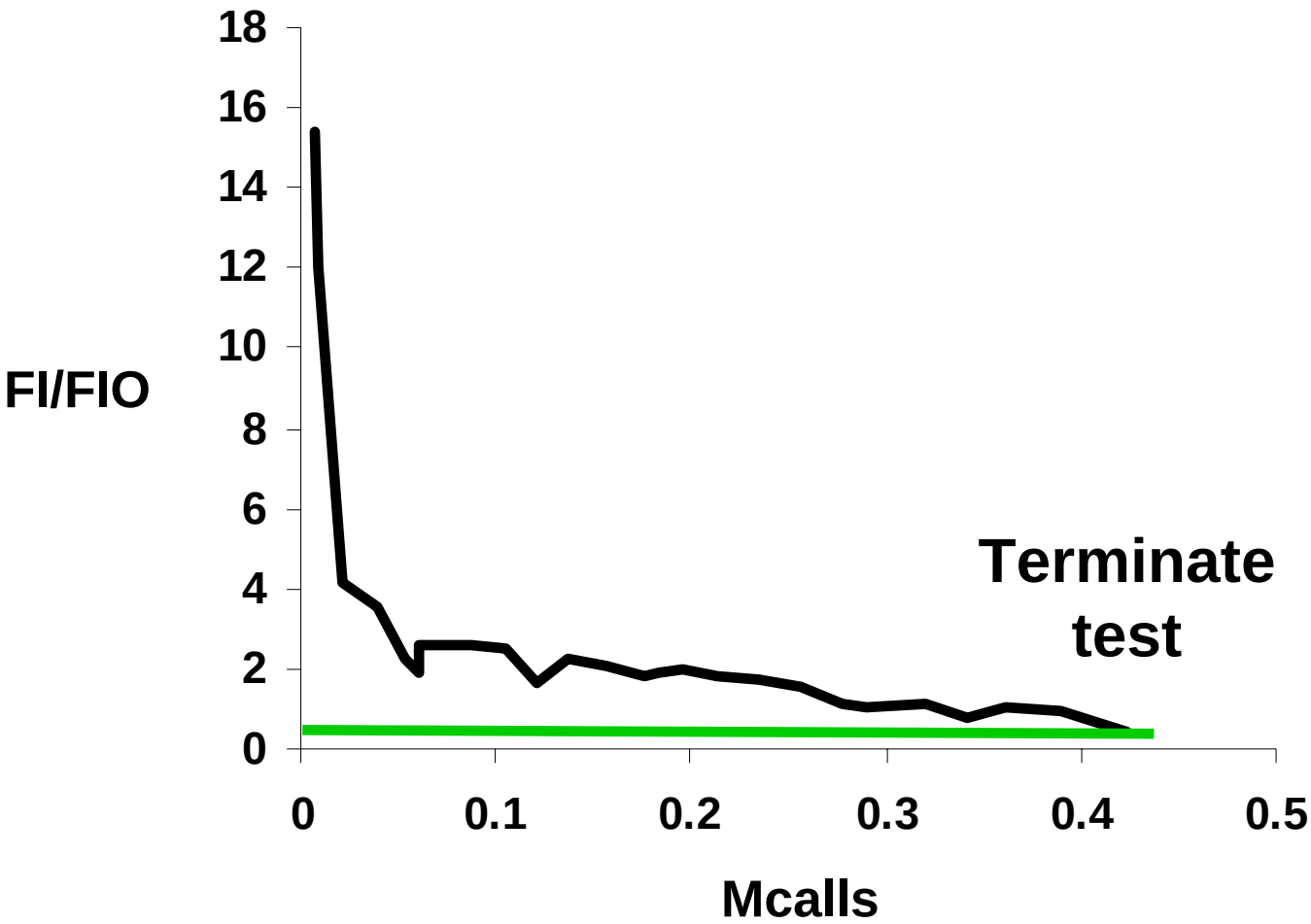
Interpret Plot : Illustration - FF



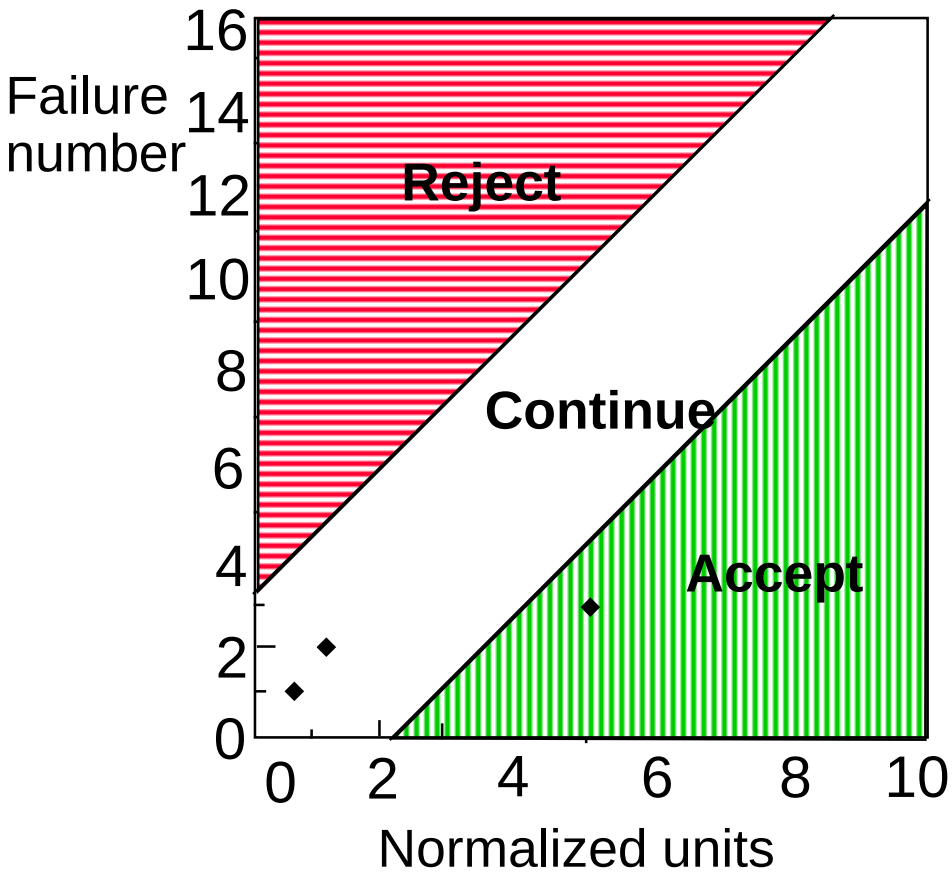
Possible causes:

1. **Poor change control**
2. **Poor control of test execution, resulting in test selection probabilities varying in time**

Interpret Plot : Illustration - FF



Certify Reliability – Dem. Chart: Illus. - FF – BP Supersystem



Fail. No.	Mcalls at Failure	Normalized Units
1	0.00375	0.75
2	0.00625	1.25
3	0.025	5

Failure intensity objective:
200 failures / Mcalls