Improving Your
Software Reliability and Security

August 22, 2013

George Mason University
Arlington, Virginia

Cyber Security & Information Systems
Information Analysis Center

Improving Your Software Reliability and Security

8:30 – 9:15 am	Your situation, your needs
9:15 -10:00 am	Goal Setting … *How Good Do You Need To Be?*
10:00 - 10:15 am	**Break**
10:15 - 11:00 am	Designing-In Reliability
11:00 - 11:45 am	Building-In Reliability
11:45 - 12:45 pm	**Lunch Provided**
12:45 - 1:30 pm	Watching As You Go … *Assessment and Mid-Course Corrections*
1:30 - 2:15 pm	Release Decision … *When to Let Go*
2:15 - 2:30 pm	**Break**
2:30 - 3:30 pm	Resources for the Journey
3:30 - 4:30 pm	Consolidation and Commitment … *Where to Go From Here*

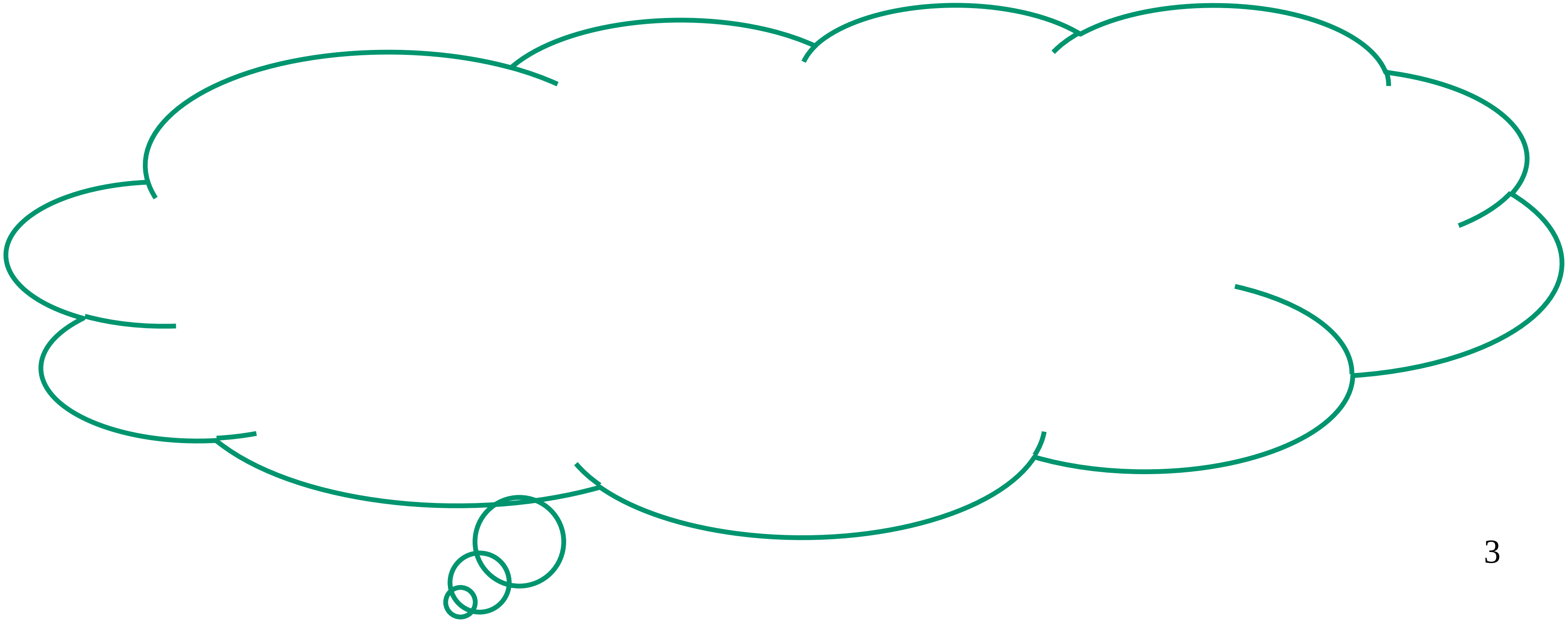
Improving Your Software Reliability and Security

Do you need to reduce the frequency and severity of failures due to software defects?

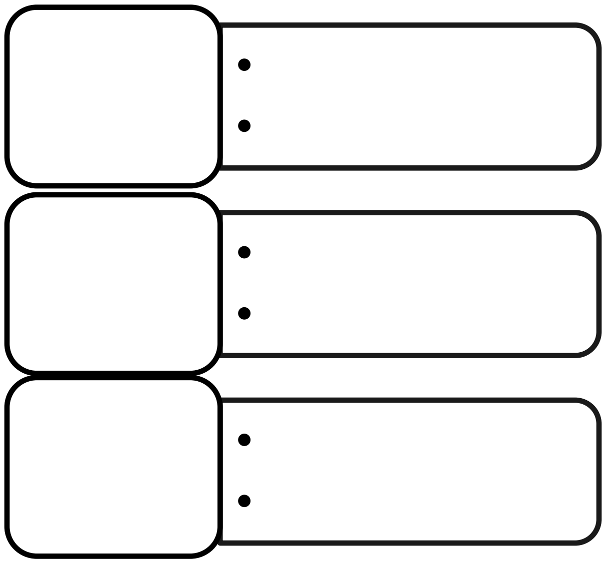
Do you need to assure adequate confidence in your systems?

This workshop is intended to help equip you with techniques and tools to meet those goals … and do so in a cost-effective way.

What are YOUR OWN goals for today?

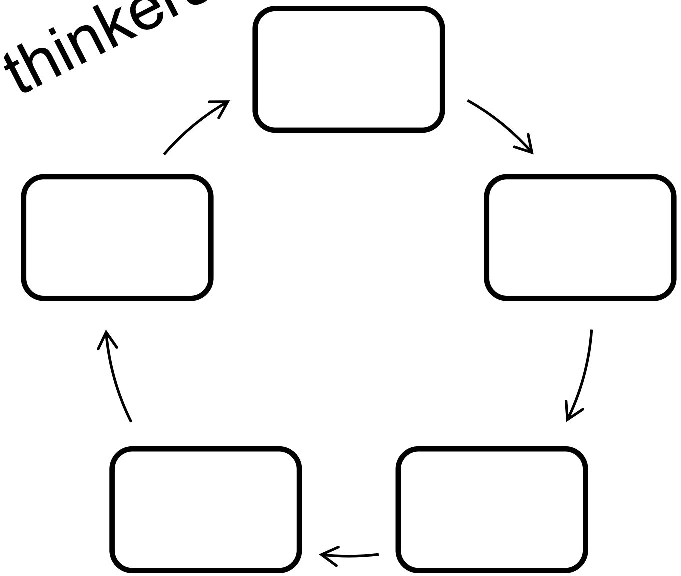


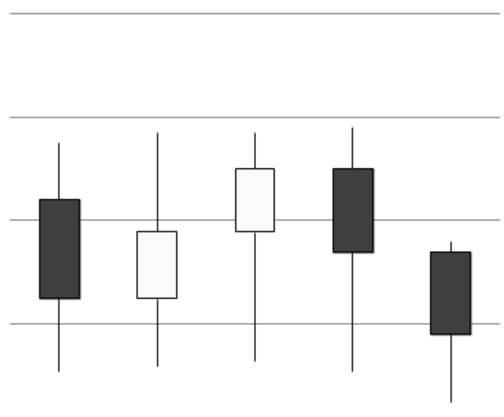
Take notes …



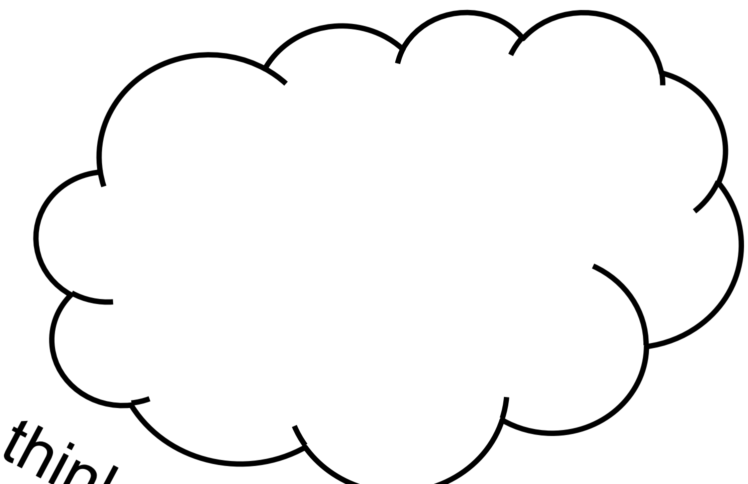
linear thinkers

process thinkers





quantitative thinkers



visual thinkers

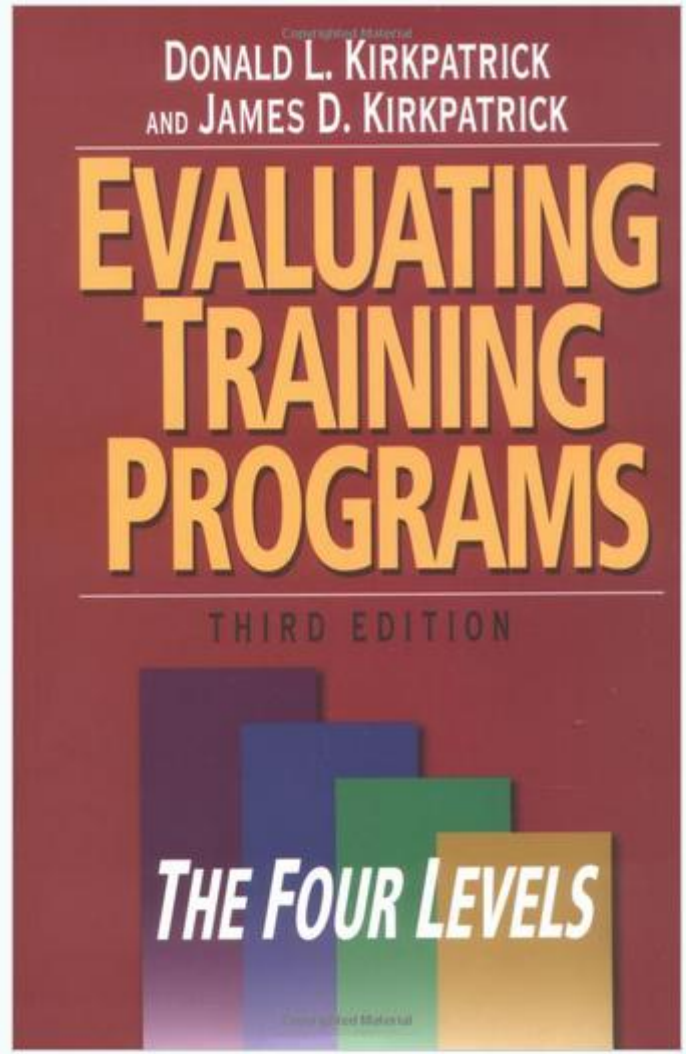
Evaluating …

… *reaction*

… *learning*

… *behavior*

… *results*



REACTION

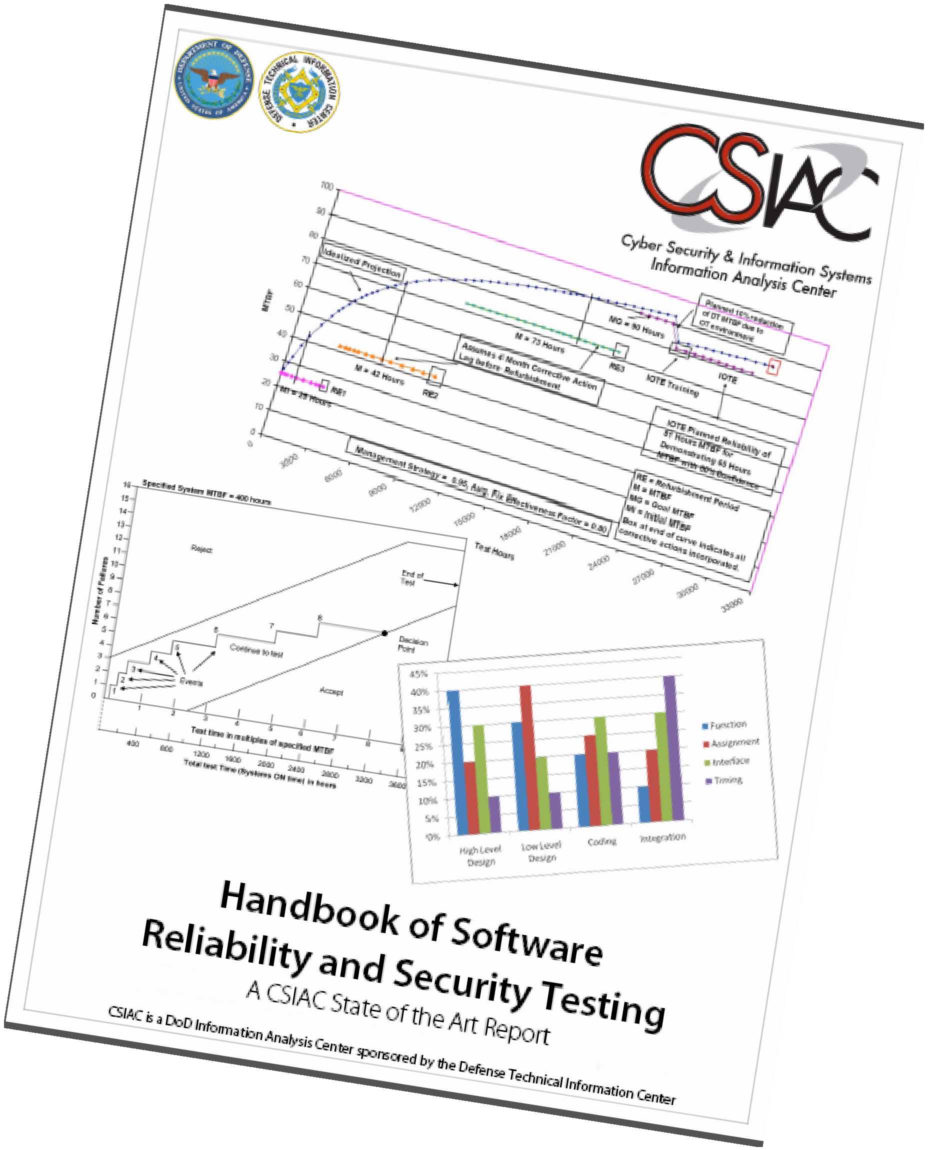
Feedback sheet provided

LEARNING

Should I quiz you on content?

BEHAVIOR + RESULTS

... Call me … or I'll call you


CSIAC
Cyber Security & Information Systems
Information Analysis Center
Handbook of Software Reliability and Security Testing
A CSIAC State of the Art Report
CSIAC is a DoD Information Analysis Center sponsored by the Defense Technical Information Center


Handbook of Software Reliability and Security Testing
DISTRIBUTION A. Approved for public release; distribution unlimited
CSIAC

Files Currently on the Disc (21)

- BNL_sw in PRA
- BNL_sw rel survey
- BSIMM3
- CRITCAL CODE
- cyber attack on SCADA
- DOD reliability directive DTM-11-003
- evidence for dependable systems_NAP
- glossary_nistir-7298-revision1
- Managing resilience
- NASA_FTA aerospace
- NIST InfoSec Risk Mang_ SP800-39
- NIST InfoSec testing_SP800-115
- NIST Sp Pub 800-53
- Open Source Security Testing Methodolo...
- OWASP_Testing_Guide_v3
- SAMM-1.0
- Secure Software Development_SEI_2005
- SecurityTesting
- SW impact on system reliability_Gooden...
- SW Reliability Test Handbook June 2012
- Systems Security Engineering_2010

8:30 – 9:15 am	Your situation, your needs
9:15 -10:00 am	Goal Setting … *How Good Do You Need To Be?*

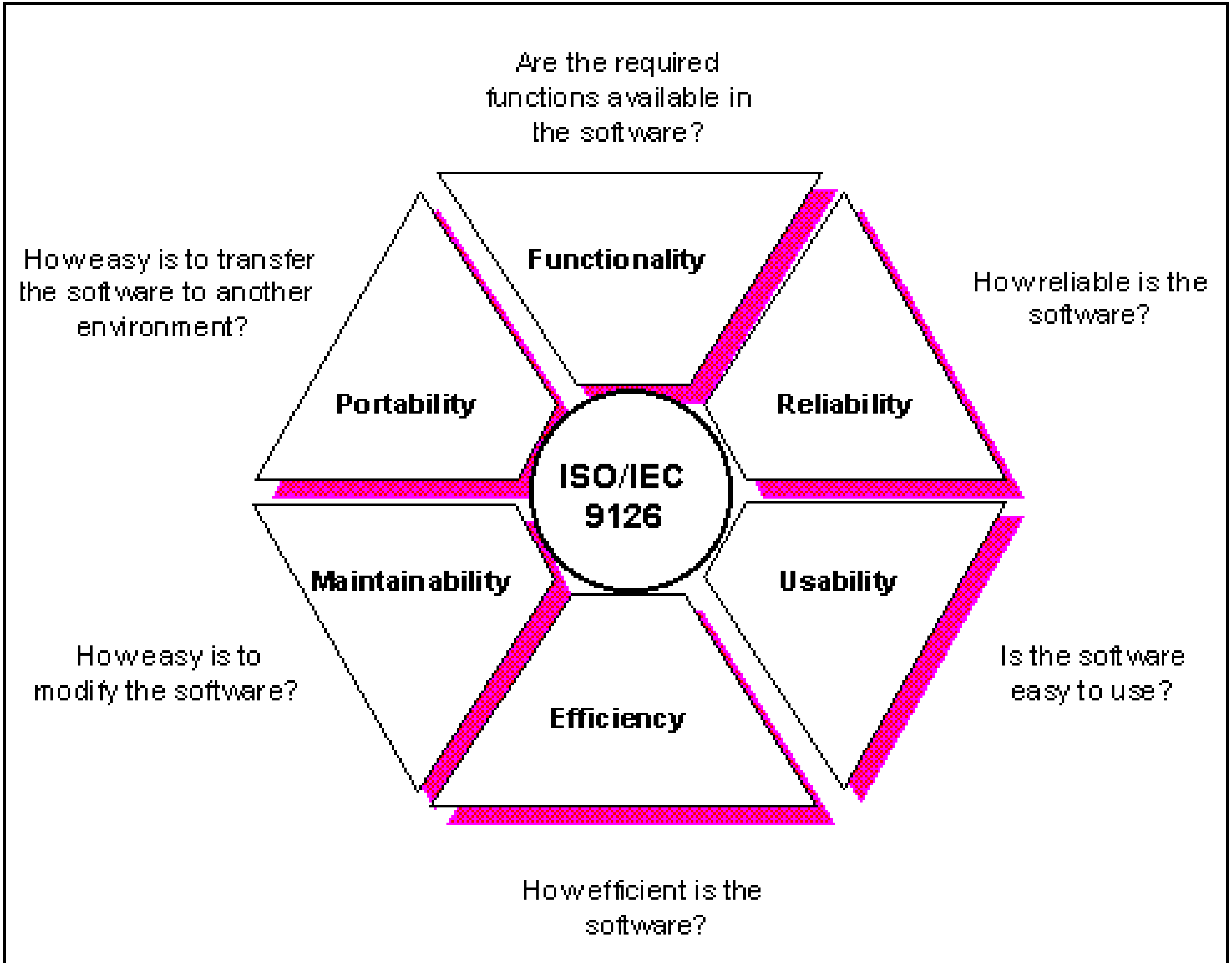
Handbook Topics 1.2 and 2.0

Glossary of Key Information Security Terms

Evaluating Software's Impact on System and System of Systems Reliability

Review of Quantitative Software Reliability Methods

Project Management
Information Assurance
Systems Engineering
Software Acquisition
Software Assurance
Software Engineering
Test & Evaluation
Information Systems Security Engineering
Safety and Security



ISO/IEC 9126-1:2001

QUALITY =
meeting requirements

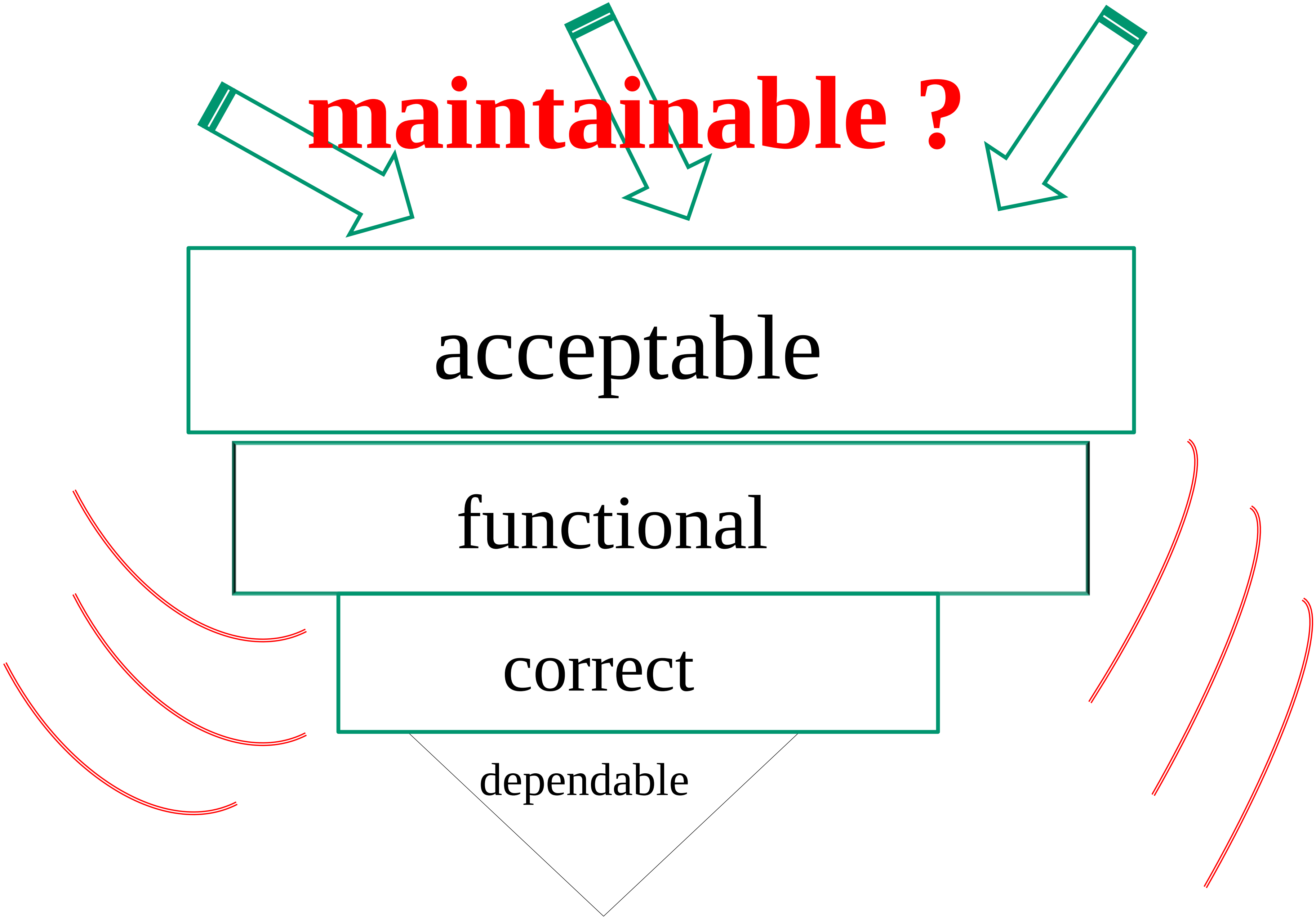
QUALITY =
"fitness for use"

Make it.

Make it work.

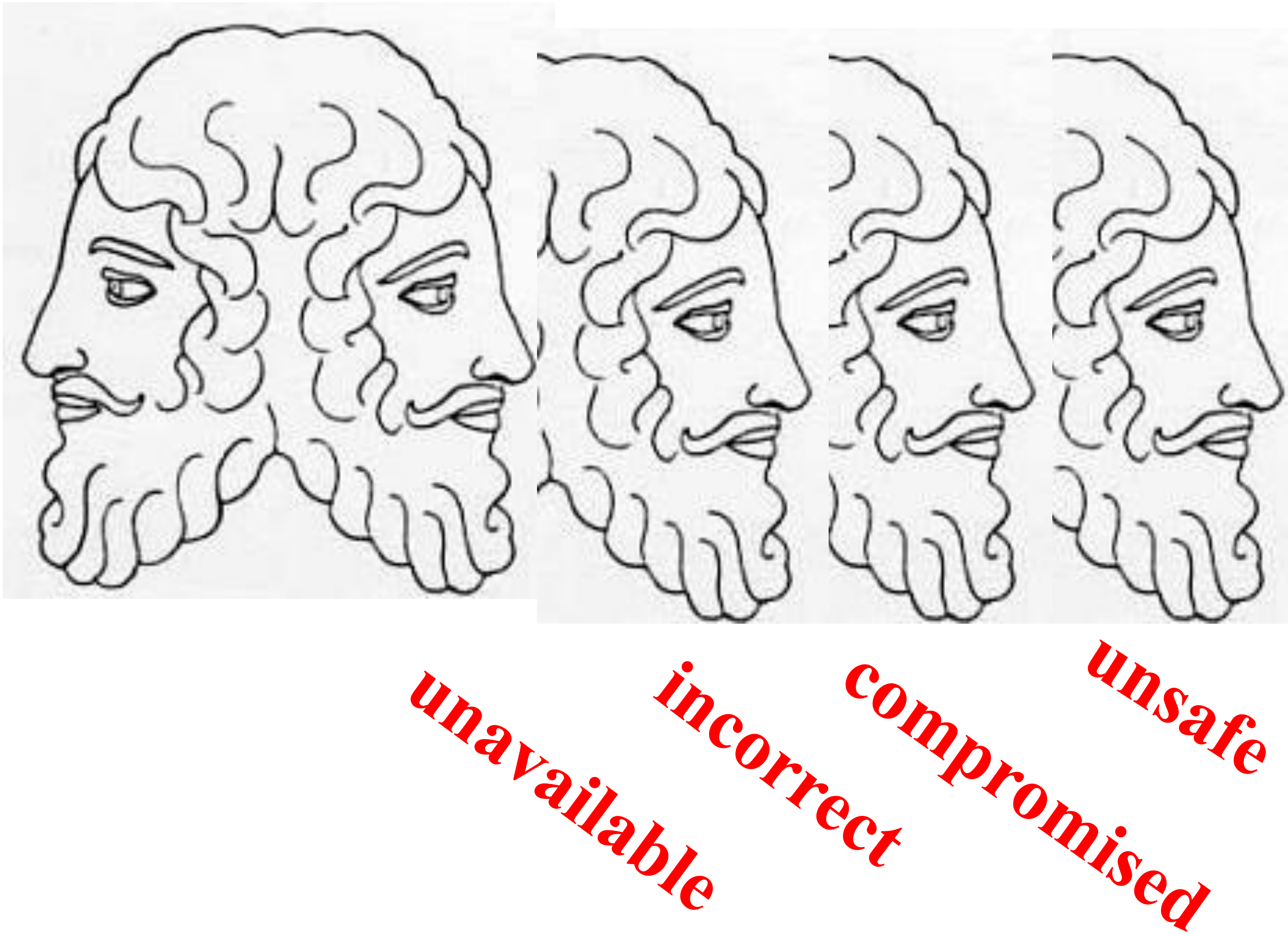
Make it work right.

Make it work right, regardless …


maintainable ?
acceptable
functional
correct
dependable

Reliability: *does* what is expected

Unreliability: *doesn't* do what is expected


unavailable
incorrect
compromised
unsafe

Reliability: *measured in…*

…success/failure **probability**

…**M**ean **T**ime **T**o **F**ailure

measured in …

Mean **T**ime **T**o **R**epair

risk exposure

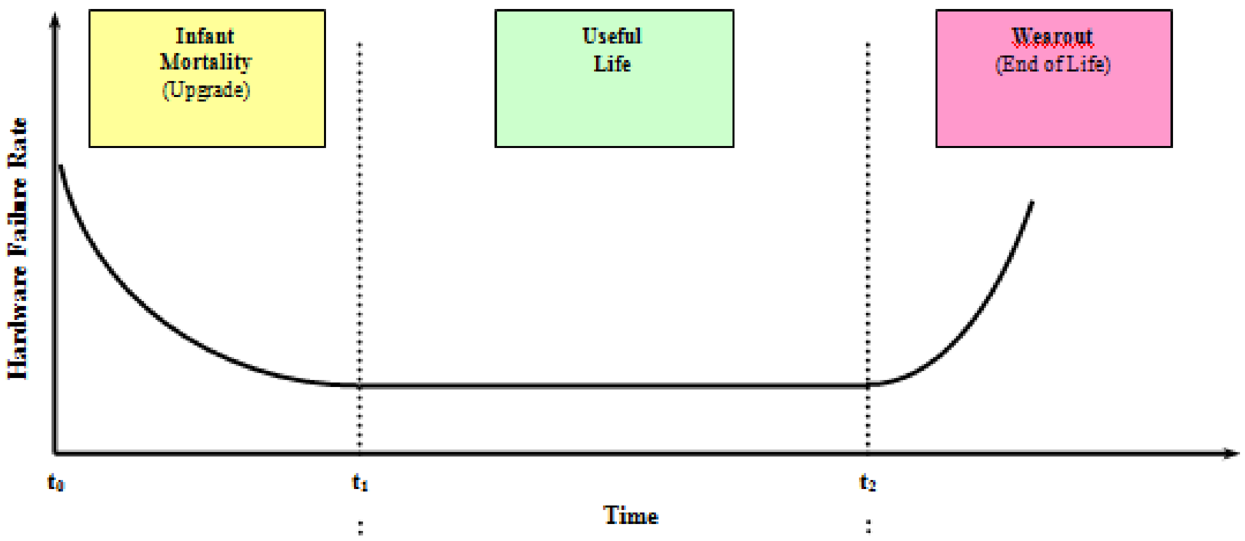
mission failure

$ loss

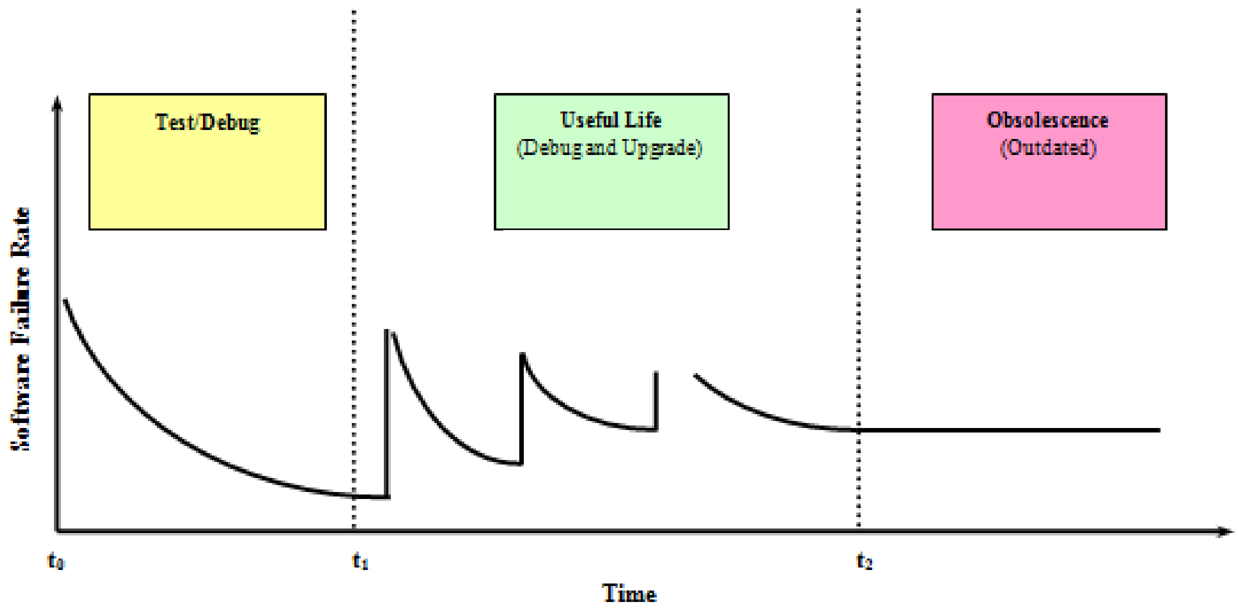
	Hardware	Software
Causes of Failures	Some defects from errors in specification and design Many defects arise from production variability Wear-out causes failures after extended use	All defects from errors during development No productionvariability: all copies identical No wear-out

	Hardware	Software
Causes of Failures	Some defects from errors in specification and design Many defects arise from production variability Wear-out causes failures after extended use	All defects from errors during development No productionvariability: all copies identical No wear-out

	Hardware	**Software**
Causes of Failures	Some defects from errors in specification and design Many defects arise from production variability Wear-out causes failures after extended use	All defects from errors during development No production variability: all copies identical No wear-out



	Hardware	Software
Causes of Failures	Some defects from errors in specification and design Many defects arise from production variability Wear-out causes failures after extended use	All defects from errors during development No productionvariability: all copies identical No wear-out



	Hardware	Software
Reliability Issues	Historical failure data available for components Warnings (precursors to failure) often occur Redundancy does improve reliability	Typically no component history available Failures usually occur without warning Redundancy (with identical copies) does not improve reliability

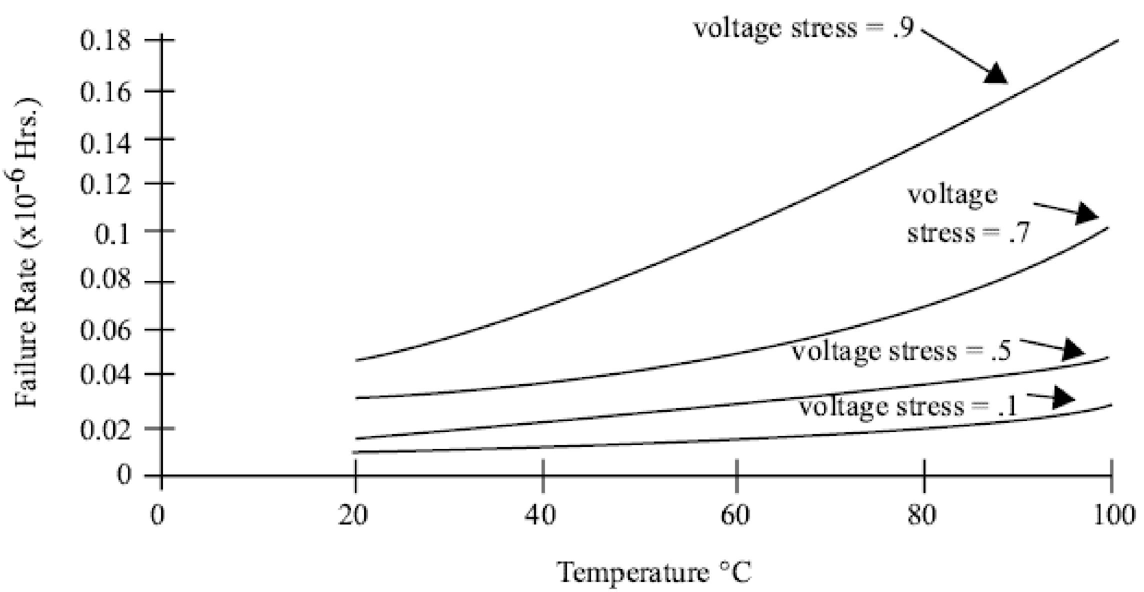


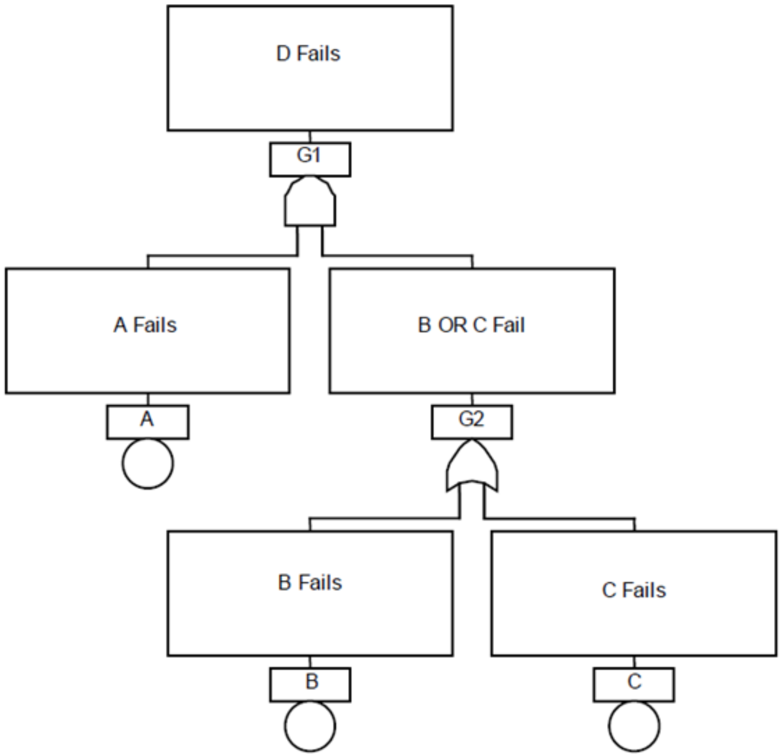
Figure 2. Trimmer Ceramic Capacitor Failure Rates/Stress Plot from MIL-HDBK-217

	Hardware	Software
Reliability Issues	Historical failure data available for components Warnings (precursors to failure) often occur Redundancy does improve reliability	Typically no component history available Failures usually occur without warning Redundancy (with identical copies) does not improve reliability

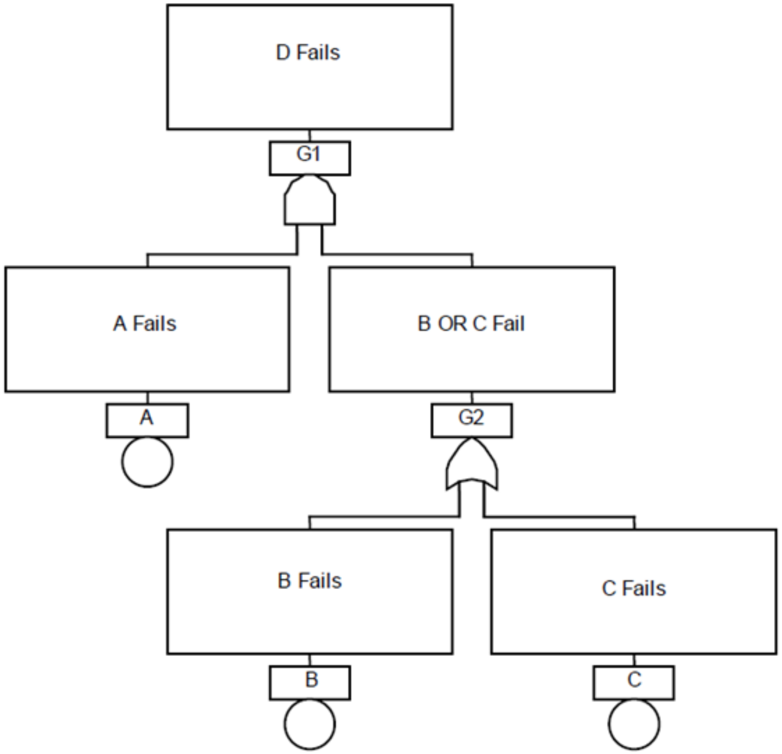
Table 5: U.S. Averages for Number of Defects Per FP [Jones, 2008]

Form of Software	Size 100	In FP 1000	10000	100000	Average
End-User	1.05	-	-	-	1.05
Web	0.52	0.60	1.01		0.71
MIS	0.32	0.75	1.14	2.54	1.19
U.S. Outsource	0.19	0.59	0.90	1.76	0.86
Offshore Outsource	0.41	0.81	1.13	2.22	1.14
Commercial	0.24	0.40	0.64	0.92	0.55
Systems	0.15	0.24	0.35	0.56	0.33
Military	0.22	0.47	0.62	0.77	0.52
Average	**0.39**	**0.55**	**0.83**	**1.46**	**0.81**

	Hardware	**Software**
Reliability Issues	Historical failure data available for components Warnings (precursors to failure) often occur Redundancy does improve reliability	Typically no component history available Failures usually occur without warning Redundancy (with identical copies) does not improve reliability



	Hardware	Software
Reliability Issues	Historical failure data available for components Warnings (precursors to failure) often occur Redundancy does improve reliability	Typically no component history available Failures usually occur without warning Redundancy (with identical copies) does not improve reliability



IEEE

IEEE Recommended Practice on Software Reliability

IEEE Reliability Society

Sponsored by the
Standards Committee

1633™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

27 June 2008

IEEE Std 1633™-2008

Contents

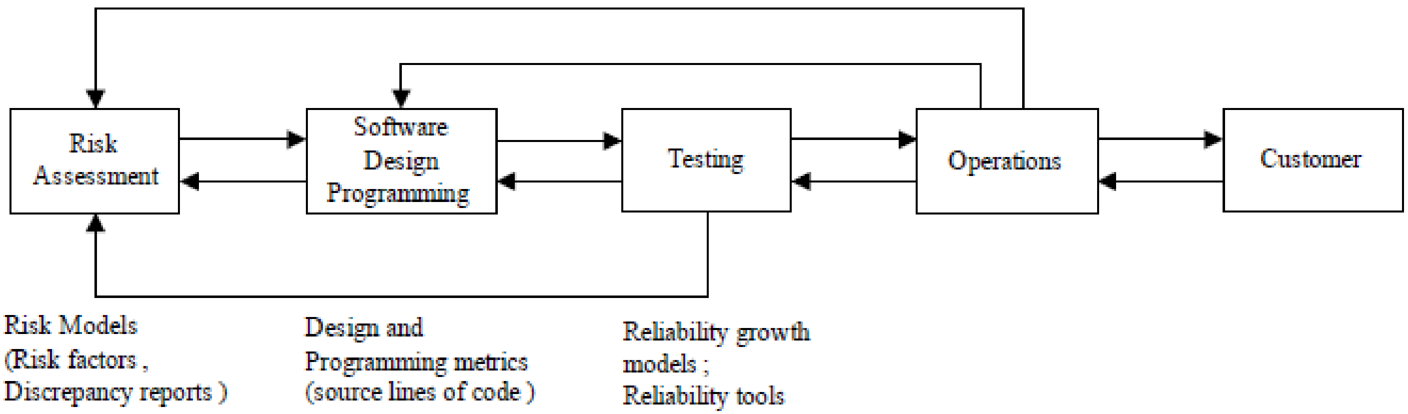


Figure a—SRE process

IEEE Std 1633-2008
IEEE Recommended Practice in Software Reliability

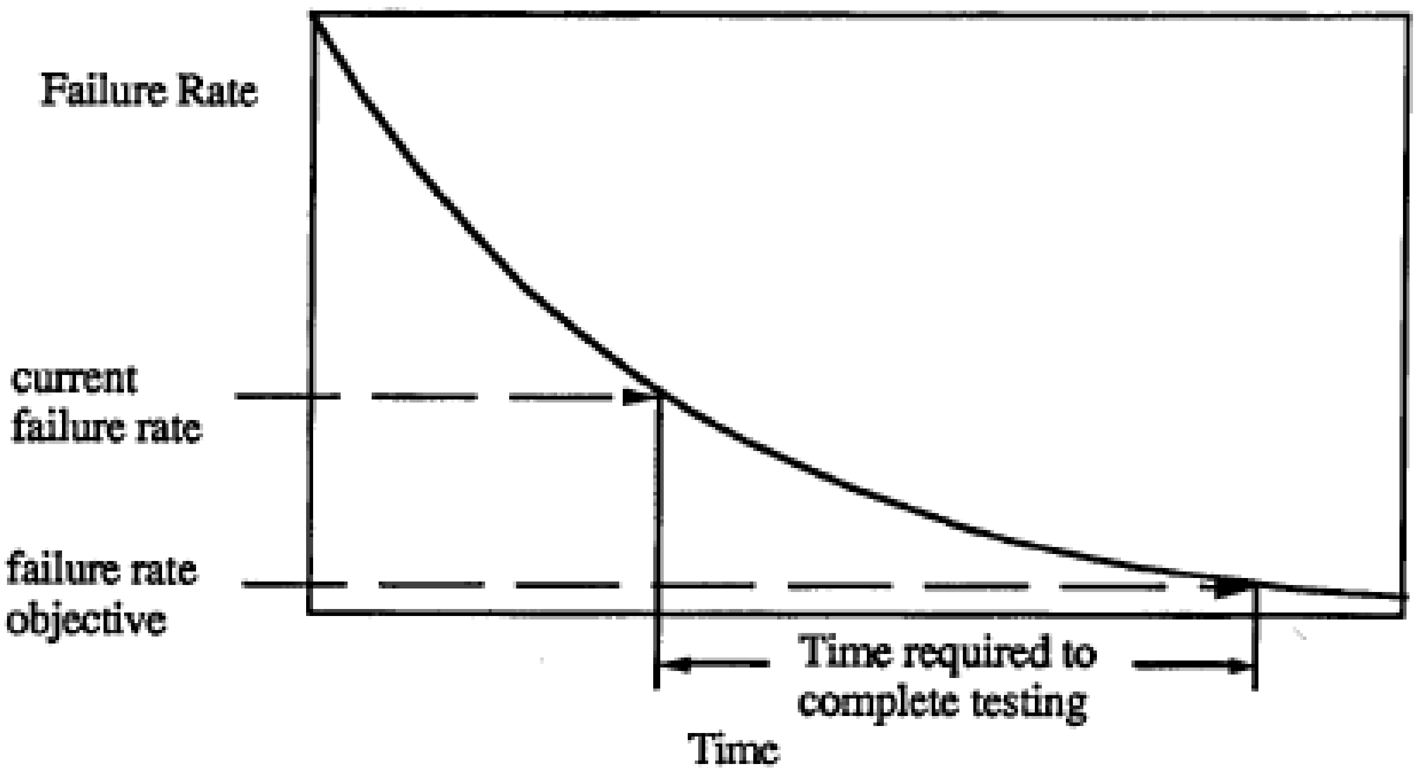


Figure 3—Example SR measurement application

IEEE Std 1633-2008
IEEE Recommended Practice in Software Reliability

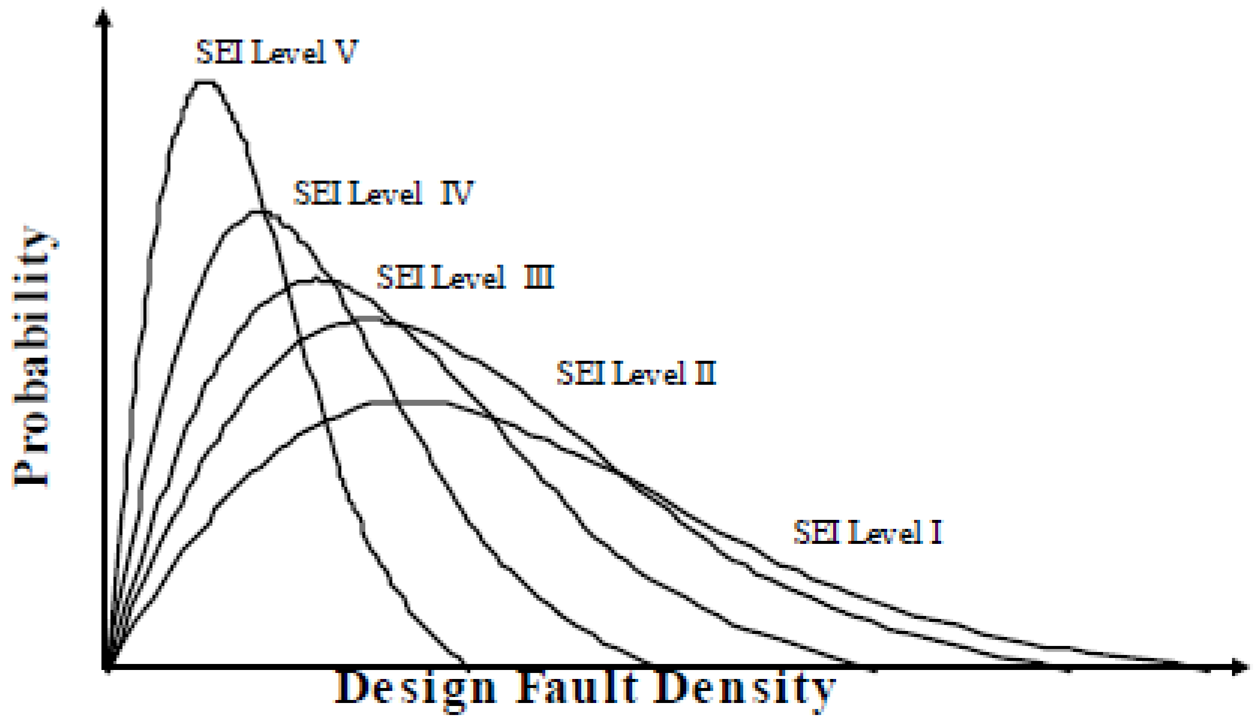


Figure F.2—Illustrating projected design defect density as a function of the development organization's design capability, as measured in terms of CMM capability

Progressive Software Reliability Prediction

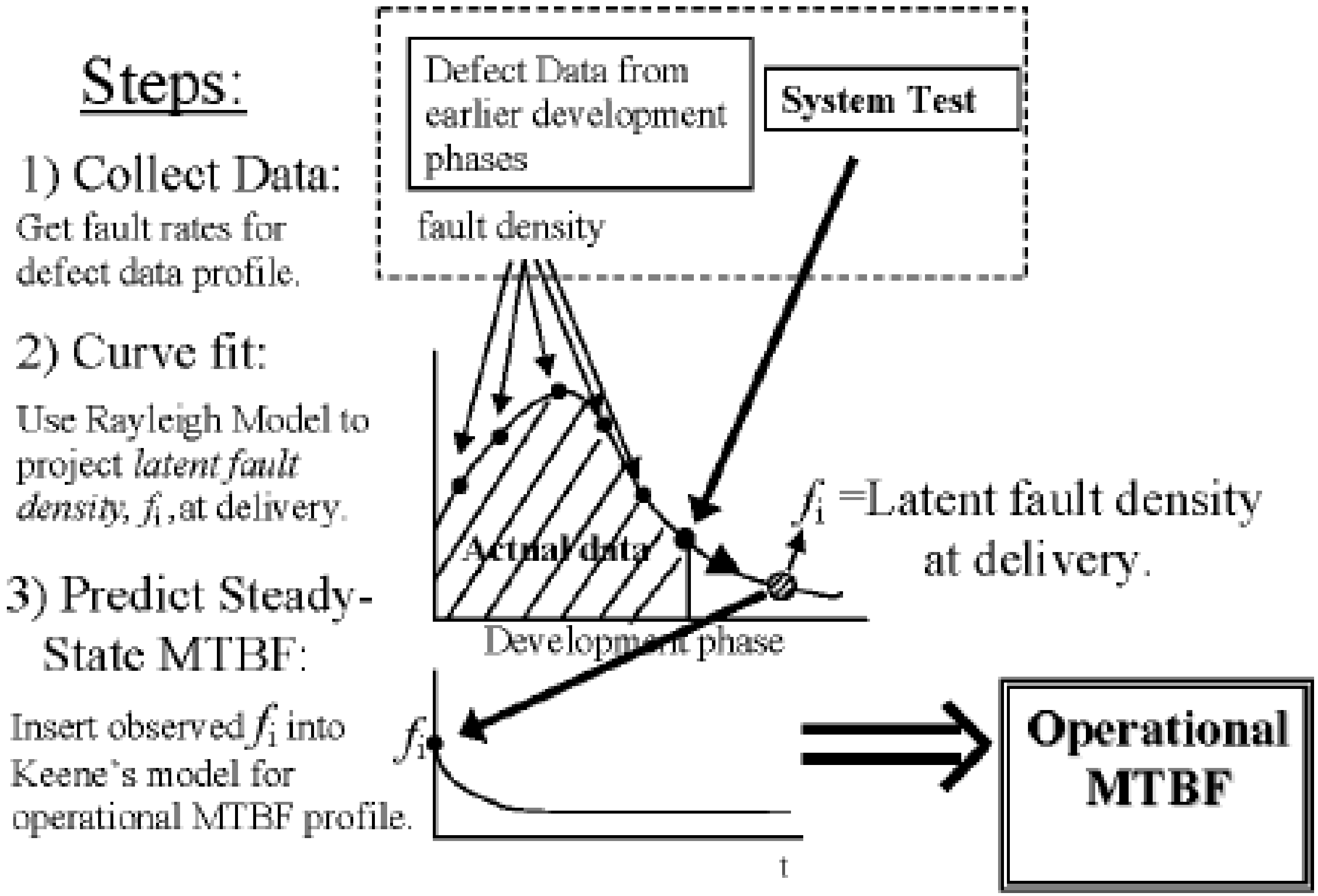
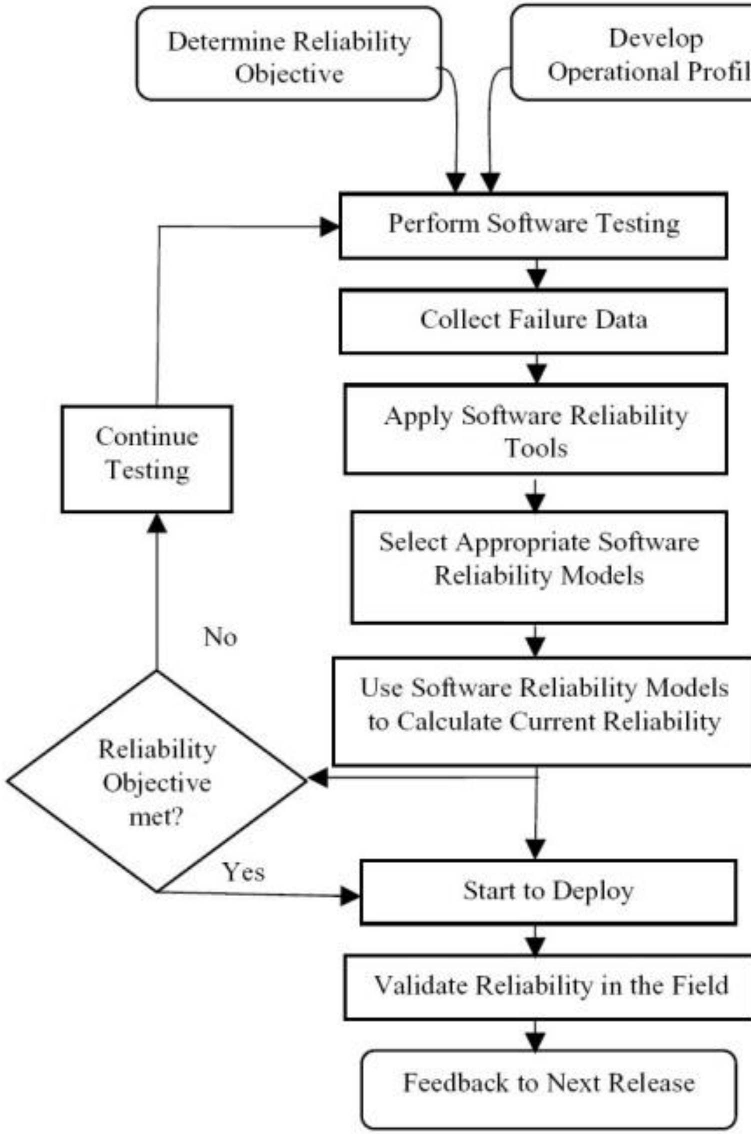
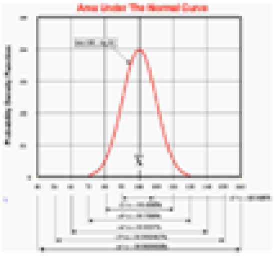


Figure F.4—Progressive SR prediction


Determine Reliability Objective
Develop Operational Profile
Perform Software Testing
Collect Failure Data
Apply Software Reliability Tools
Select Appropriate Software Reliability Models
Use Software Reliability Models to Calculate Current Reliability
Continue Testing
No
Reliability Objective met?
Yes
Start to Deploy
Validate Reliability in the Field
Feedback to Next Release

Software Reliability Engineering

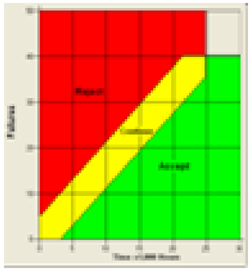


Establish quantitative reliability targets

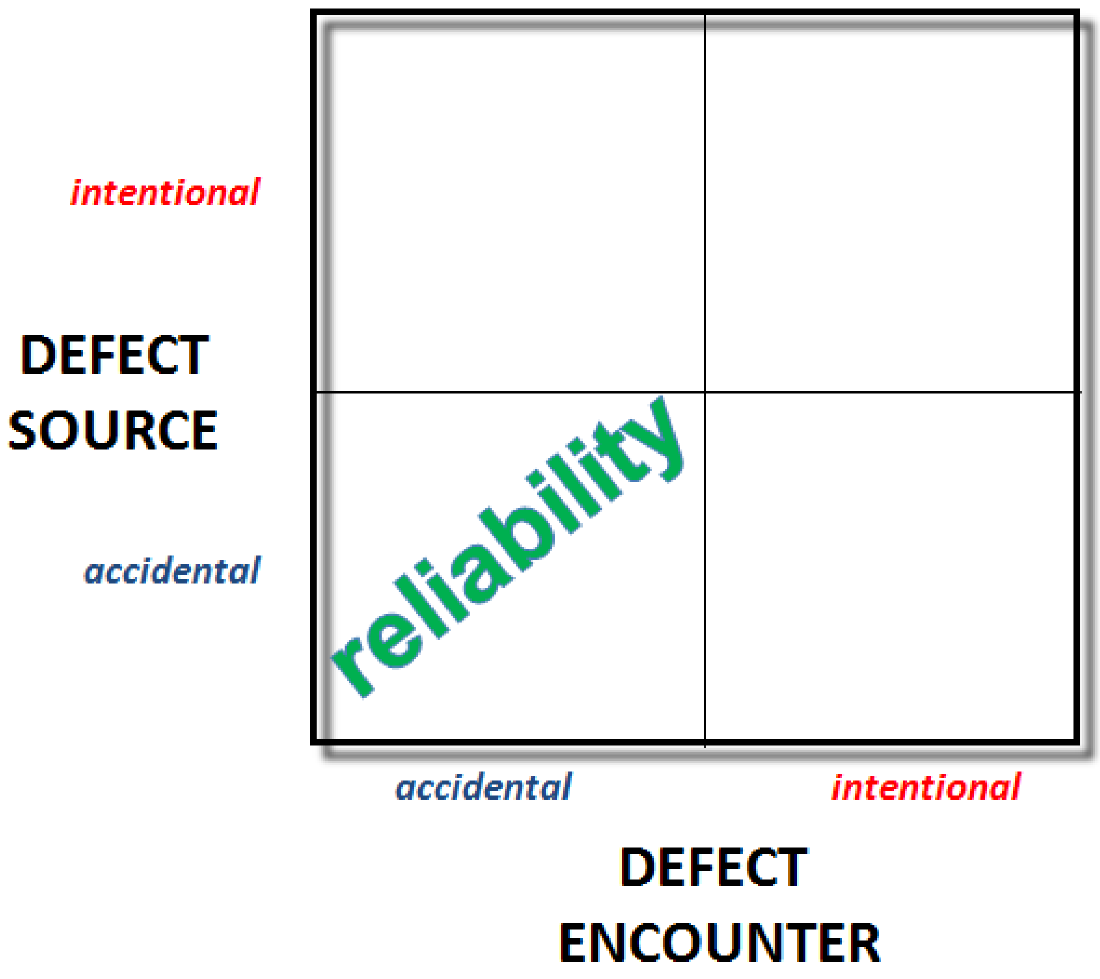
Construct usage profiles of operational system

Operation	Occurrence probability	Initial test cases
Enter card	.332	66
Verify PIN	.332	66
Withdraw checking	.199	40
Withdraw savings	.066	13
Deposit checking	.040	8
Deposit savings	.020	4
Query status	.00664	1
Test terminal	.00332	1
Input to stolen card list	0.00058	0
Backup files	0.000023	0
Total	1	199

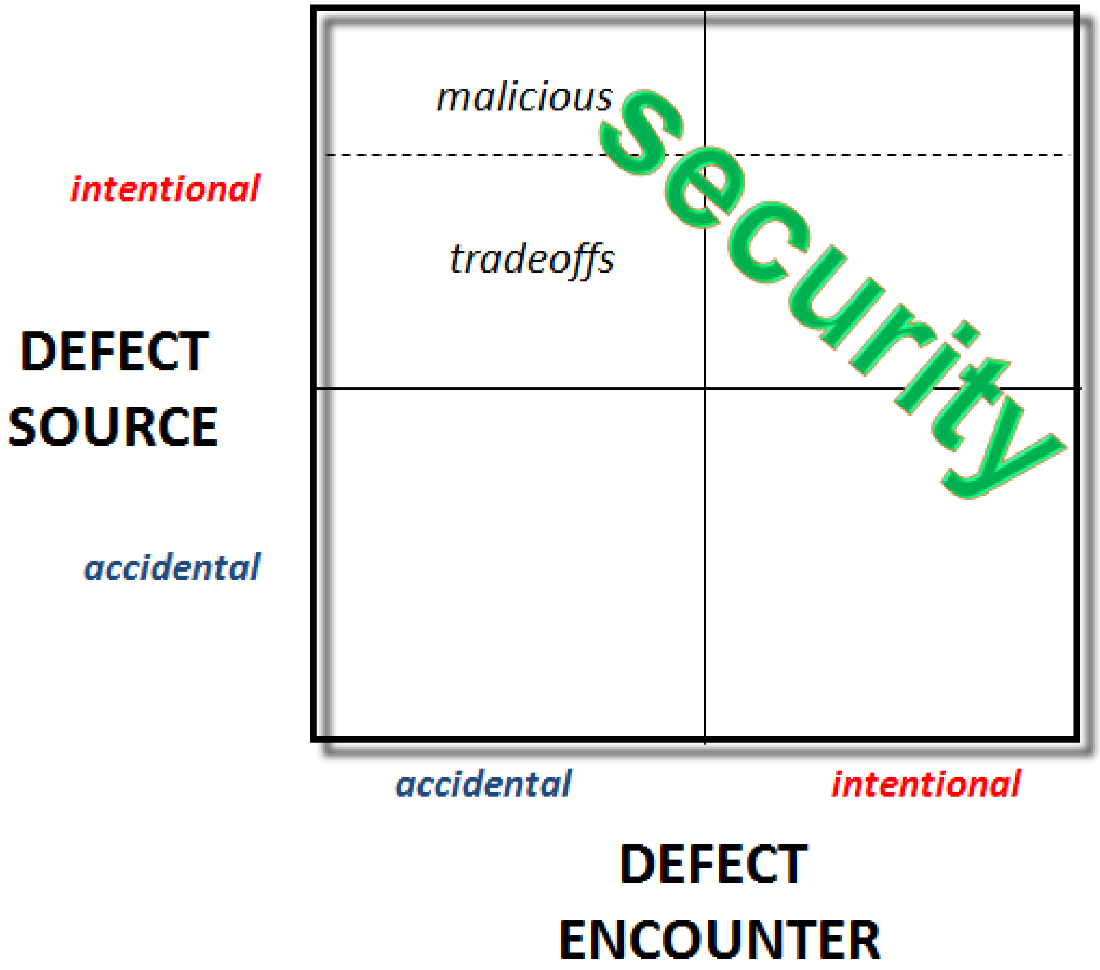
Test statistically to predict system reliability



Software Unreliability



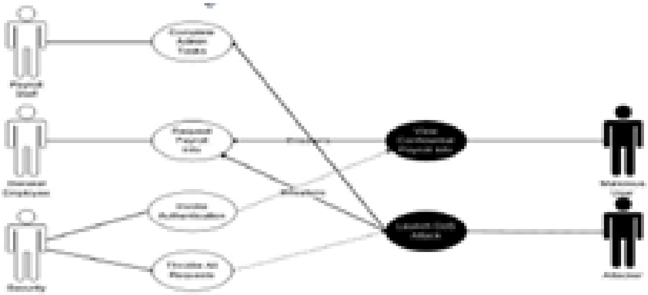
Software Insecurity



Software Security Engineering

Establish multiple quantitative targets

Use threat modeling to identify abuse cases



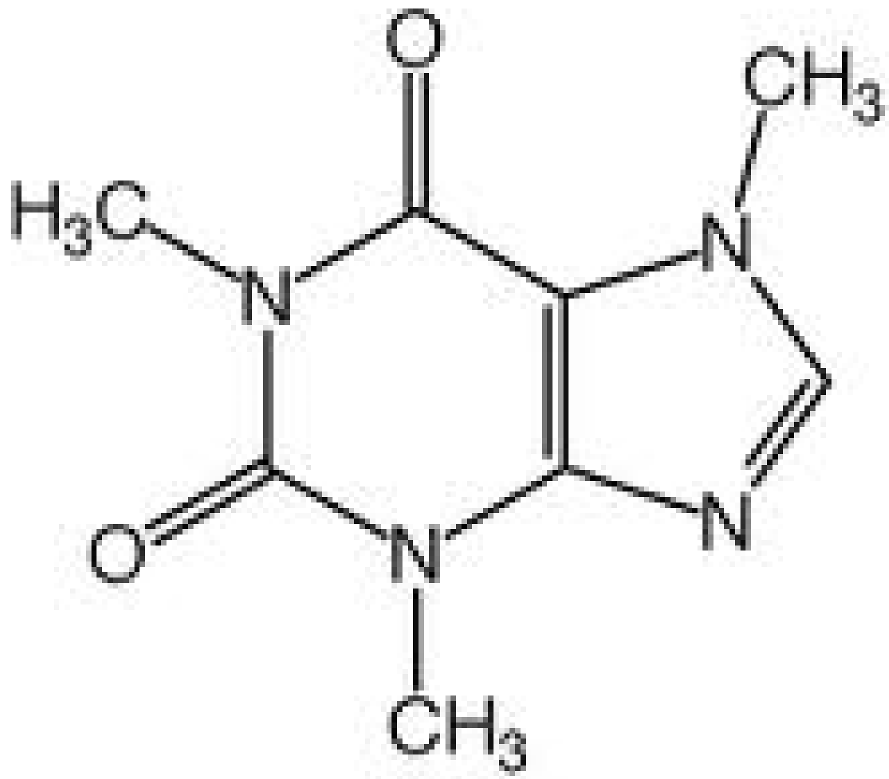
Rethink software reliability growth modeling

Software Security Engineering

confidentiality

integrity

accessibility

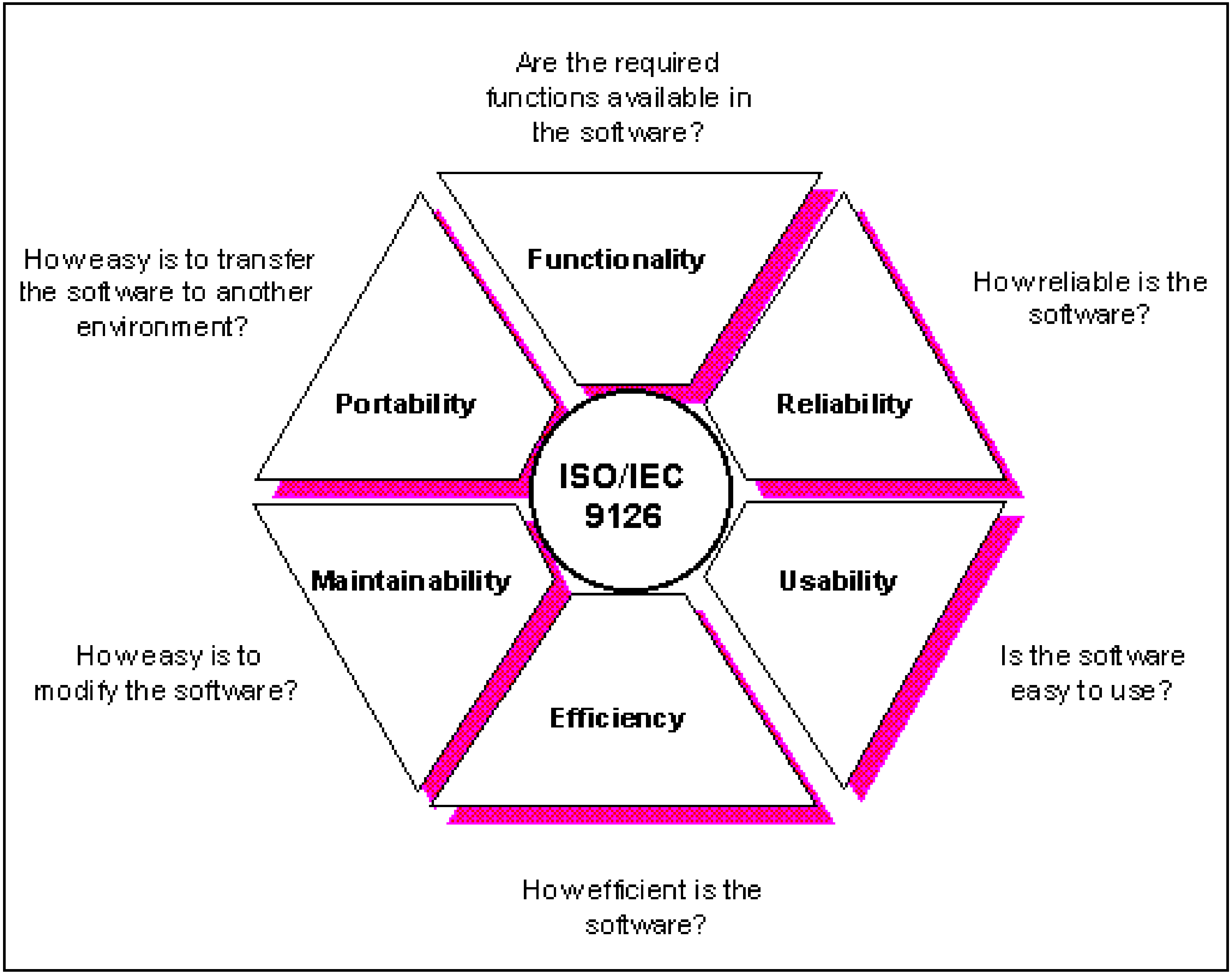

O
CH3
H3C
N
N
O
N
N
CH3

10:15 - 11:00 am	Designing-In Reliability
11:00 - 11:45 am	Building-In Reliability

Building Security In Maturity Model

Secure Software Development Life Cycle Processes

Systems Security Engineering

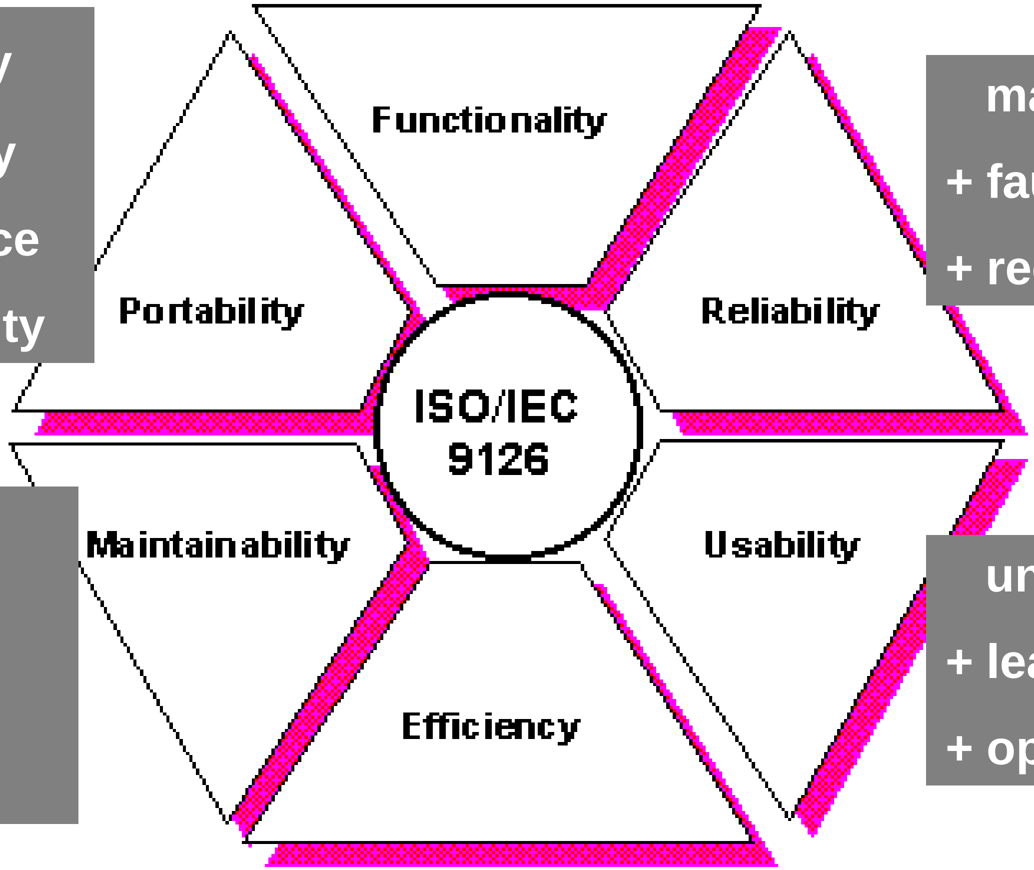


ISO/IEC 9126-1:2001

suitability + accurateness + interoperability + compliance + security

adaptability
+ installability
+ conformance
+ replaceability

maturity
+ fault tolearnce
+ recoverability



analyzability
+ changeability
+ stability
+ testability

understandability
+ learnability
+ operability

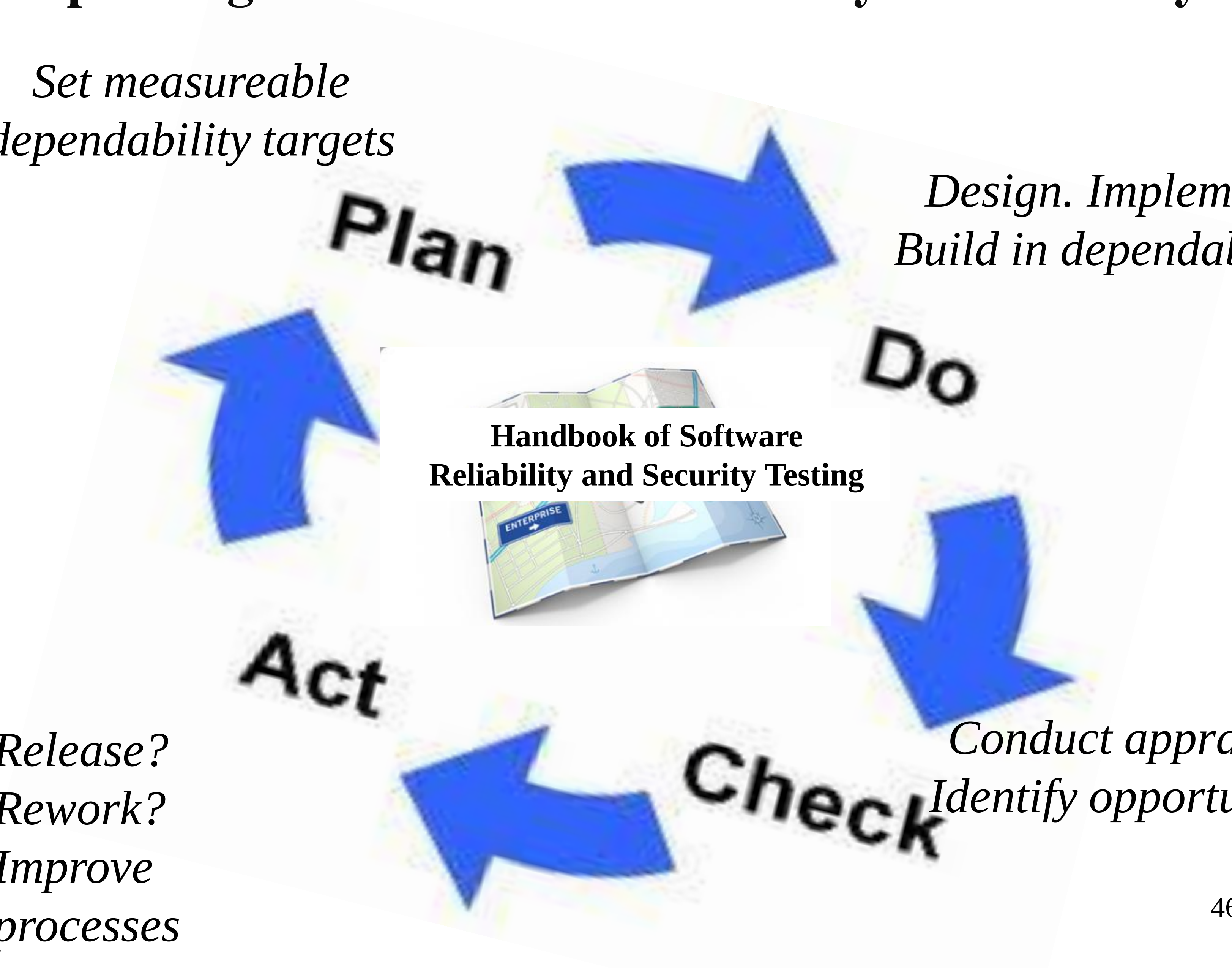
time behavior + resource behavior

ISO/IEC 9126-1:2001

"Improving Your Software Reliability and Security"

Set measureable dependability targets

Design. Implement. Build in dependability.



Handbook of Software Reliability and Security Testing

Conduct appraisals. Identify opportunities.

Release? Rework? Improve processes

Photo # NH 96566-KN First Computer "Bug", 1945

Statistical
Modeling and
Estimation of
Reliability
Functions for
Software

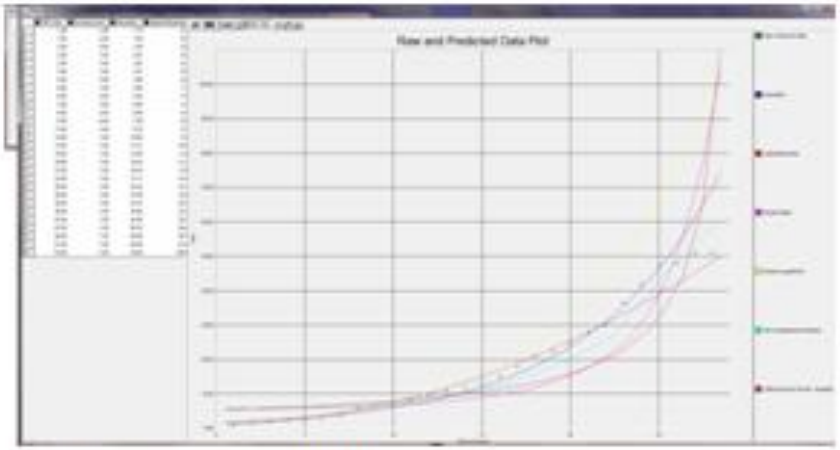


Figure 3. Typical SMERFS Output Curves

Figure 4. Typical SMERFS Output Calculations

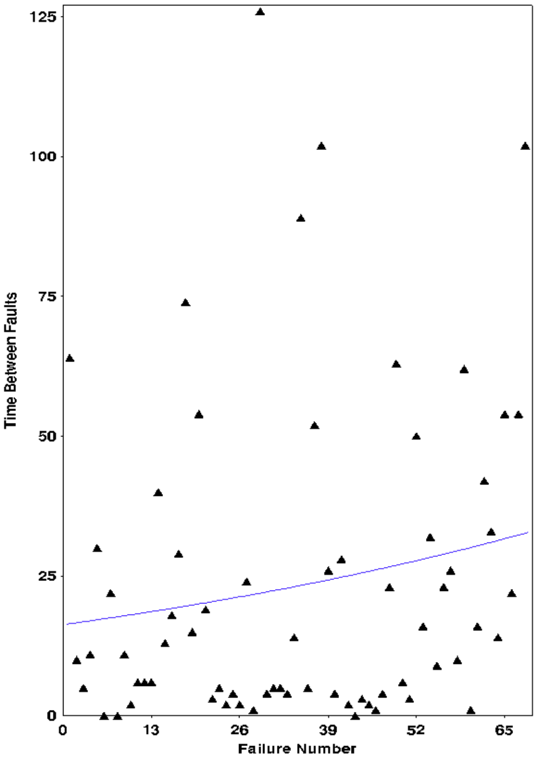
Ozmet (2005) analyzed OpenBSD 2.2 data

79 vulnerabilities discovered 1998-2002

Applied reliability growth models in SMERFS

Found best fit from
Musa logarithmic model

Acceptable results also from
other models



"Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models." Andy Ozment, University of Cambridge. *First Workshop on Quality of Protection*, Milan, Italy, September 15, 2005.

[Berkeley Software Distribution = Unix-derived operating system]

Shin and Williams (2013) analyzed Firefox web browser

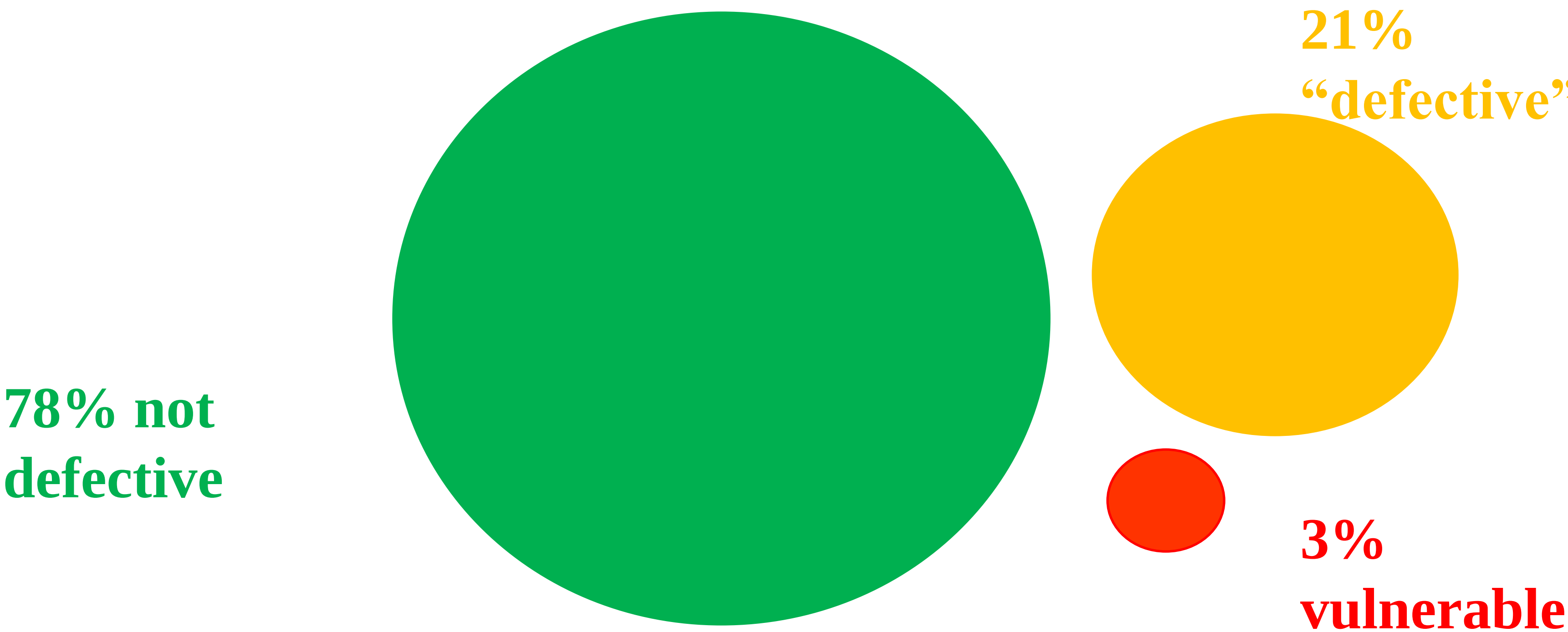
Used fault prediction models based on traditional metrics

Found valid to predict vulnerabilities, although with high rate of false positives

“Can traditional fault prediction models be used for vulnerability prediction?” Yonghee Shin (DePaul University) and Laurie Williams (North Carolina State University). *Empirical Software Engineering* (2013) 18:25-59.

Shin and Williams (2013) … Firefox web browser

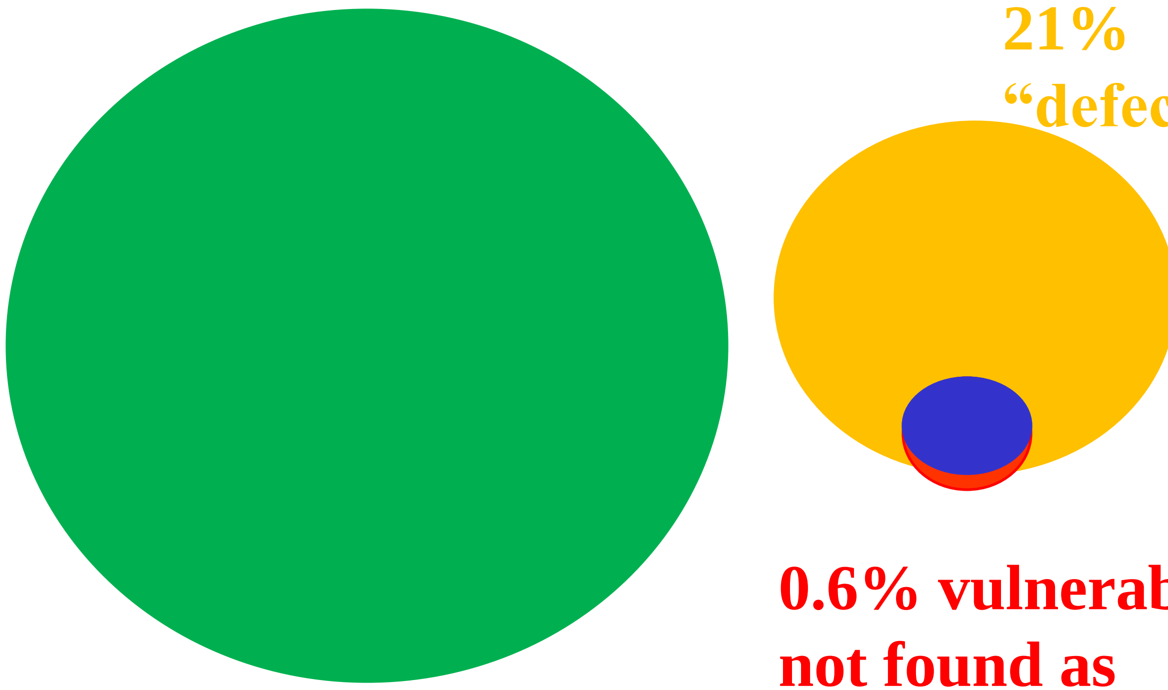
11,259 total files



Shin and Williams (2013) … Firefox web browser

11,259 total files

21% “defective”



78% not defective

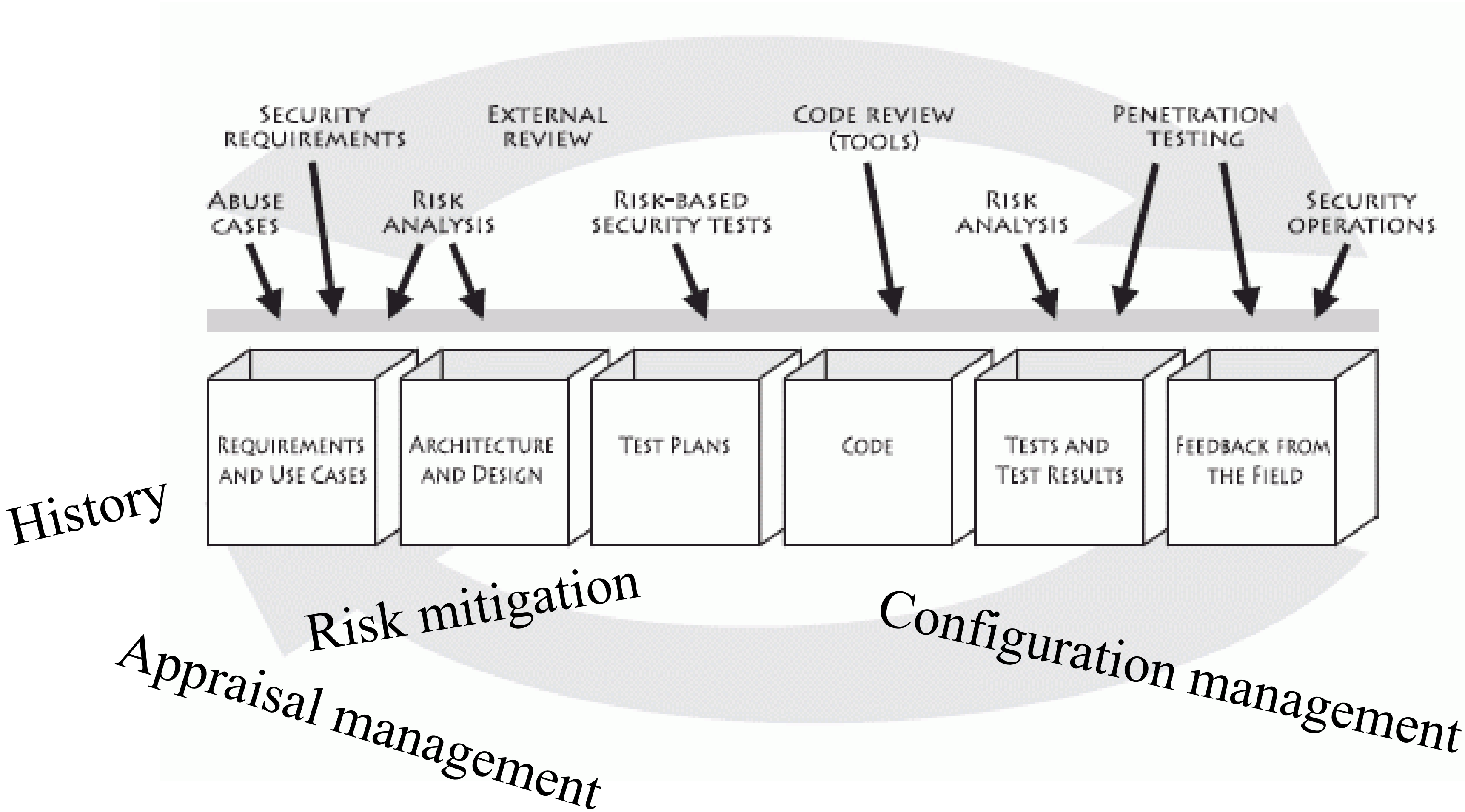
0.6% vulnerable but not found as “defective”

Traceability Matrix

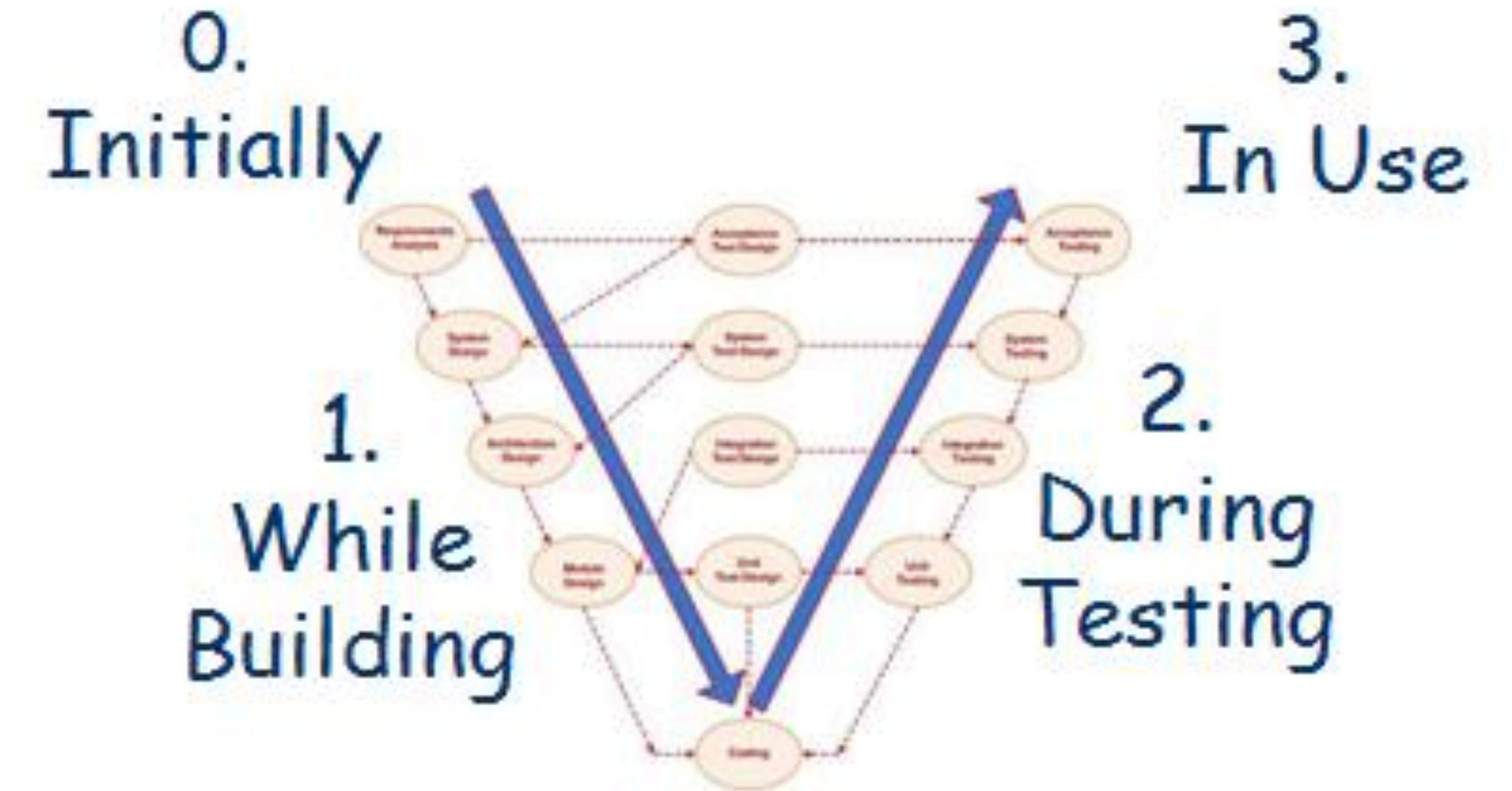
Software Requirements Specification	Design	Code	Test Plan
[Enter SRS ID here.]	[Enter design element ID here.]	[Enter code location or ID here.]	[Enter test case number here.]

[Add rows until there is at least one row per requirement]

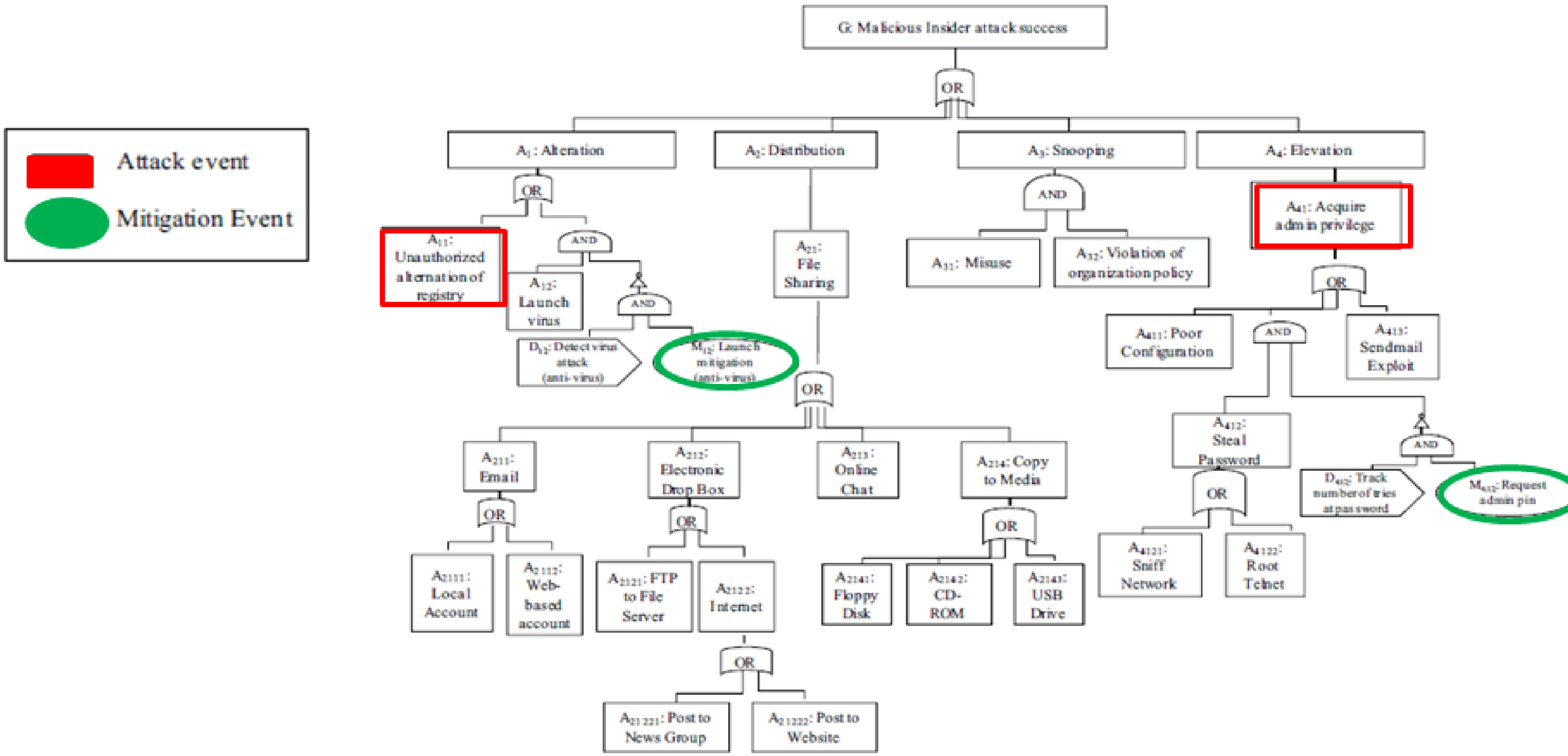
Software Security "Touchpoints"



Software Reliability Modeling



Software Security Modeling



Attack + **Countermeasure** Tree

12:45 - 1:30 pm	Watching As You Go … *Assessment and Mid-Course Corrections*
1:30 - 2:15 pm	Release Decision … *When to Let Go*

Security Assurance Maturity Model

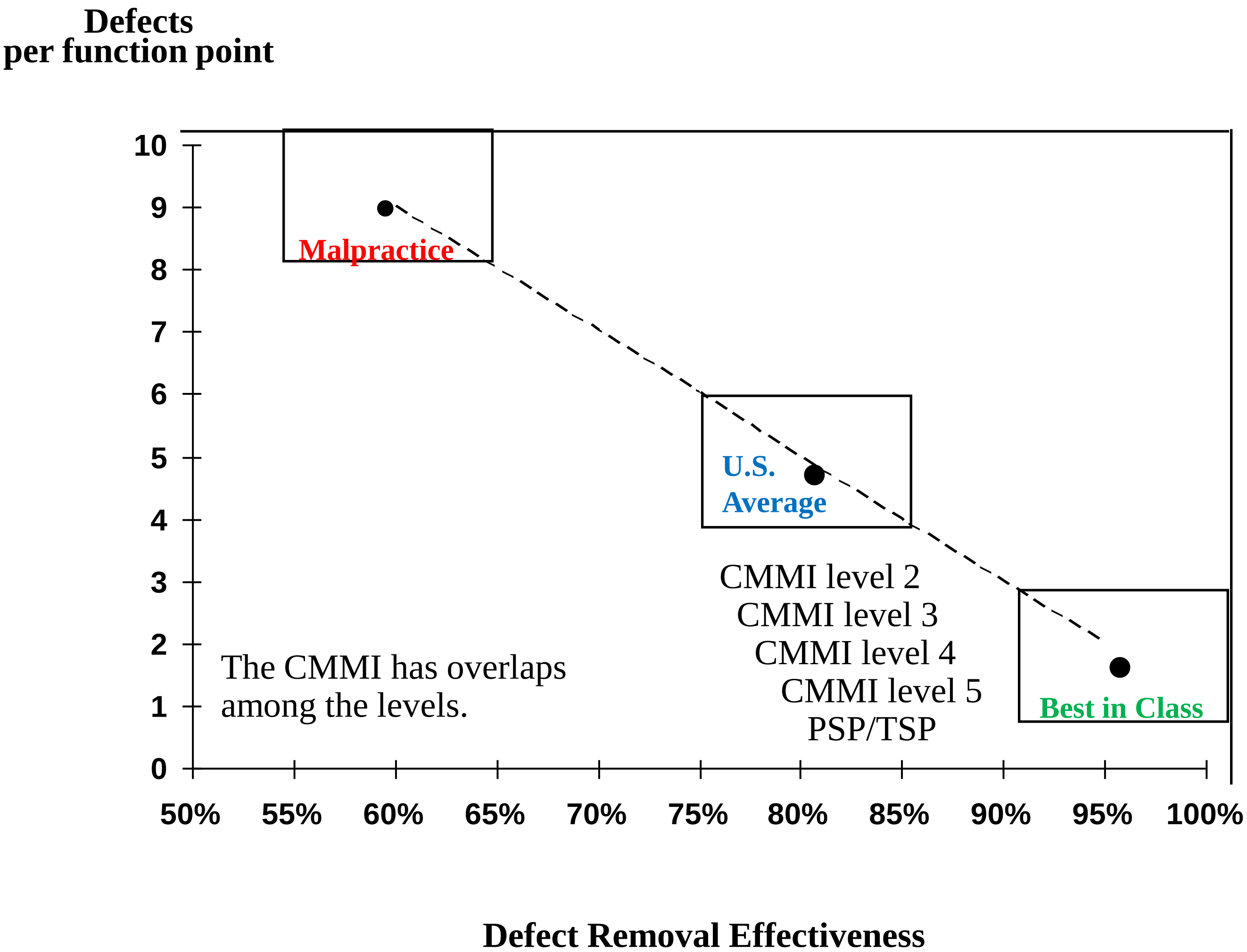
Technical Guide to Information Security Testing and Assessment

Open Source Security Testing Methodology Manual

Open Web Application Security Project Testing Guide

Handbook Topic 3.5.13: Optimal Release Time

MAJOR SOFTWARE QUALITY ZONES



[Capers Jones. unpublished communication]

RANGES OF DEFECT REMOVAL EFFECTIVENESS

	Lowest	Median	Highest
Requirements review	20%	30%	50%
Top-level design reviews	30%	40%	60%
Detailed functional design reviews	30%	45%	65%
Detailed logic design reviews	35%	55%	75%
Code inspections	35%	60%	85%
Unit tests	10%	25%	50%
New function tests	20%	35%	55%
Integration tests	25%	45%	60%
System test	25%	50%	65%
External beta tests	15%	40%	75%
CUMULATIVE EFFECTIVENESS	75%	97%	99.99%

[Capers Jones, unpublished communication]

Costs of meeting requirements

- **Prevention**
- **Appraisal**

Costs of *not* meeting requirements

- **Internal failures**
- **External failures**

COST OF QUALITY