

# Marine Corps Operational Test and Evaluation Activity



Interpreting Reliability and Availability  
Requirements for Network-Centric Systems



# What MCOTEA Does

Planning

Testing

Reporting

**Infantry Automatic Rifle Reliability**  
System Assessment Plan

---

**Logistics Vehicle System Replacement–Tractor Variant**  
System Evaluation Plan



June 2010  
Marine Corps Operational Test and Evaluation Activity  
3020 Barnett Avenue  
Quantico, VA 22134-5014

*Approved:*  *Thomas G. McQueen*  
Deputy Director, MCOTEA  
By direction

Expeditionary, Naval, and Amphibious Systems


C4ISR & IT/Business Systems




**Infantry Automatic Rifle Reliability**  
System Assessment Report

---

**Logistics Vehicle System Replacement–Tractor Variant**  
Operational Test Agency Evaluation Report



December 2010  
Marine Corps Operational Test and Evaluation Activity  
3020 Barnett Avenue  
Quantico, VA 22134-5014

*Approved:*  *D.L. Reeves*  
Colonel USMC  
Director, MCOTEA

12-23-10  
Date

FOR OFFICIAL USE ONLY

Evaluation Plans  
Assessment Plans  
Test Plans  
Observation Plans

Combat Service Support Systems

Initial Operational Test  
Follow-on Operational Test  
Multi-service Test  
Quick Reaction Test  
Test Observations

Ground Combat Systems

Evaluation Reports  
Assessment Reports  
Test Data Reports  
Observation Reports



# Purpose

---

- To engage test community in a discussion about methods in testing and evaluating RAM for software-intensive systems



# Software-intensive systems

---

- U.S. military one of the largest users of information technology and software in the world <sup>[1]</sup>
- Dependence on these types of systems is increasing
- Software failures have had disastrous consequences

Therefore, software must be highly reliable and available to support mission success



# Interpreting Requirements

---

## Excerpts from capabilities documents for software intensive systems:

### Availability

“The system is capable of achieving a threshold operational availability of 95% with an objective of 98%”

“Operationally Available in its intended operating environment with at least a 0.90 probability”

### Reliability

“Average duration of 716 hours without experiencing an operational mission fault”

“Mission duration of 24 hours”

“Completion of its mission in its intended operating environment with at least a 0.90 probability”



# Defining Reliability & Availability

---

What do we mean reliability and availability for software intensive systems?

- One consideration: unlike traditional hardware systems, a highly reliable and maintainable system will not necessarily be highly available
  - Highly recoverable systems can be less available
  - A system that restarts quickly after failures can be highly available, but not necessarily reliable
- Risk in inflating availability and underestimating reliability if traditional equations are used.
- Perhaps “dependability” is a better term [2]



# Interpreting requirements

## What do we mean by failures?

- One interpretation: flaws in code or software “bugs”
- Challenges:
  - Impractical to test all possible code combinations in CT, DT or OT
  - A relatively simple system with 20 inputs, each with 10 possible values =  $10^{20}$  possible combinations of settings [3]

## Examples of code failures:

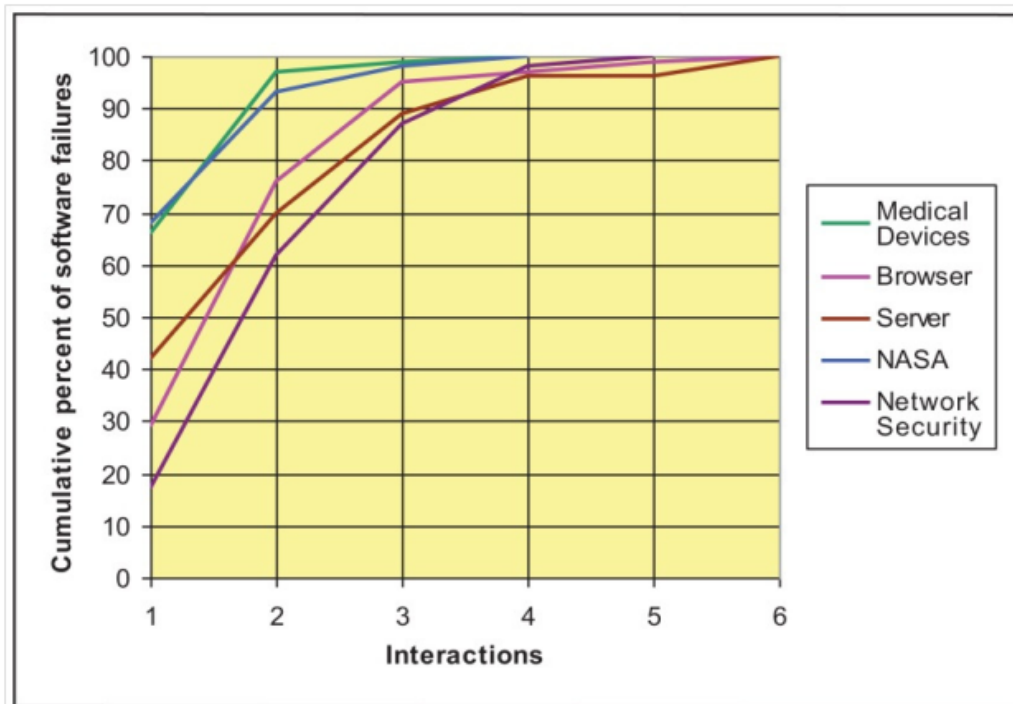
- Mars Pathfinder
- USS Yorktown
- Ariane 5 rocket (European Space Agency)



# Code failures

Since code failures are deterministic

- Pseudo-exhaustive testing
- $C(20,4) = 20!/4!(20-4)! = 4,845$
- $C(20,6) = 38,760$
- Combinatorial testing
  - E.g., ACTS from NIST
  - Algorithms to produce smaller test sets







# Stochastic failures

---

- Bandwidth variability
- User in the loop
  - Operator should be viewed as system components
  - Training
  - Crew-caused errors





# Failure to interpret requirements

## Number one cause of software failures [1]

- Inadequate specifications
- Misconceptions about requirements
- Serious usability flaws
- These flaws are often overlooked, and users are unfairly blamed [1]

Examples of lethal consequences from poor designs:

- Precision Lightweight GPS Receiver (PLGR)
- Therac-25 radiotherapy machines



# Solutions



# Equations

## Measure and report reliability and availability from different perspectives

### Availability:

- **Mission availability (availability over time)**

$$A_m = \frac{\mu}{\lambda + \mu} - \frac{\lambda}{(\lambda + \mu)^2 T} \exp[-(\lambda + \mu)T]$$

- **IT availability (time independent)**

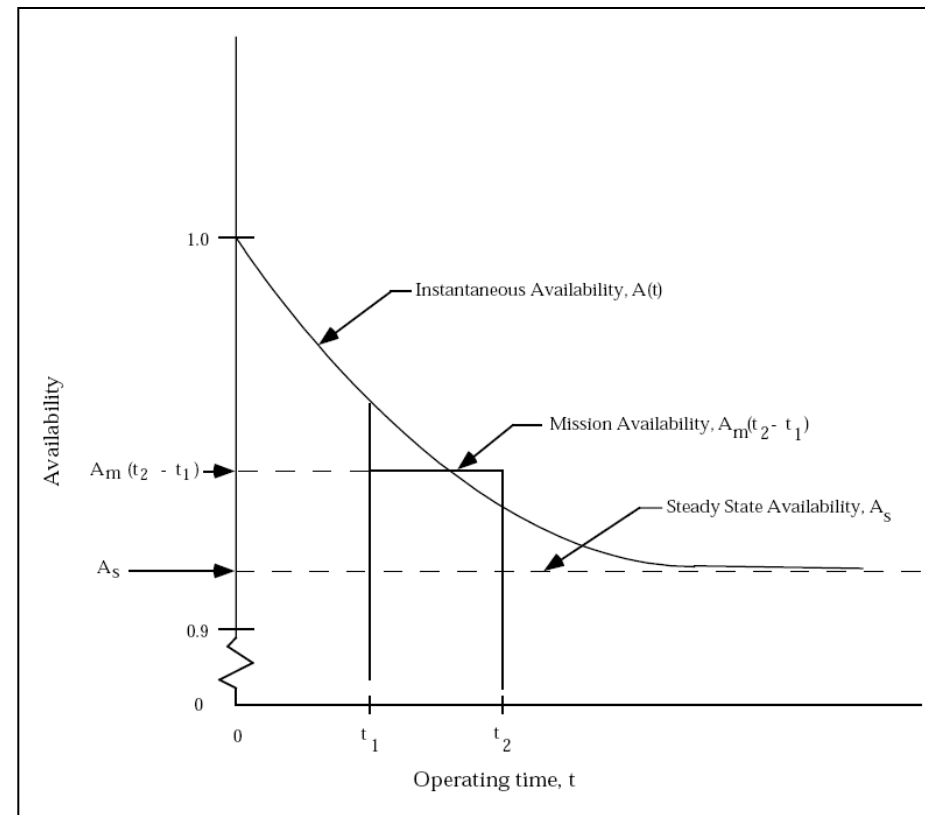
$$A_{IT} = \frac{\text{\# successful attempts to access system}}{\text{\# total attempts}}$$

### Reliability:

- $R_{IT} = 1 - (\text{Sum}_j f_j/n_j * P(Z_j))$  [5]
- $R(t) = e^{-\lambda t}$

### Dependability:

- $D_s(t; \tau) = A_s(t) * R_s(t, \tau)$  [2]



Availability as a function of operating time [6]



# Measure at different nodes

---

- User level
- Server level
- Unit level
- Report on each, plus:
  - Level of degraded service
  - May have to “script” some tests to ensure breadth of mission scenarios covered

Microsoft uses similar methods to test expected operations over a randomized schedule over a smaller interval of time [6].  
(i.e., Accelerated Life Testing)



# Integrated Testing

---

- Partner with developmental testers
  - Ensure detection of code failures
  - Early user evaluations
- Early dialogue with requirements documents generators
  - Critical to ensure systems are built right for intended missions
  - Solicit clear guidance on definitions of reliability, availability and dependability



# References

---

1. National Academy of Sciences. 2007. Software for Dependable Systems: Sufficient Evidence? Jackson, D. Thomas, M. and Millett, L.I., Editors, Committee on Certifiably Dependable Software Systems, National Research Council.
2. Heddaya and Helal, 1996. Reliability, Availability and Dependability: A user-centered view. Boston University White Paper BU-CS-97-011
3. Kuhn, D.R., Kacker, R.N., and Lei Y. 2010. Advanced Combinatorial Test Methods for System Reliability. Reliability Society 2010 Annual Technical Report.
4. Kuhn, D.R., D.R. Wallace and A.M. Gallo, Jr., 2004. Software Fault Interactions and Implications for Software Testing, IEEE Transaction on Software Engineering. Vol 30, No. 6.
5. ATEC. 2004. Reliability, Availability, Maintainability (RAM) Concept Information Technology. July 14, 2004 brief. 5. Donald MacKenzie, 2001, *Mechanizing Proof: Computing, Risk, and Trust*, MIT Press, Cambridge, Mass., Chapter 9.
6. Nicholson, C.D., Tsang, M.Y., Zhao, Z., Zhou, H. 2003. System and Method for Testing Software Reliability over Extended Time. United States Patent No.: US 6,557,120 B1.



# Questions

---

Jeanne Hartzell

Jeanne.hartzell@usmc.mil

Visit our website at [www.mcotea.marines.mil](http://www.mcotea.marines.mil)