

# An Intergated Approach to Dependability Management

R. Ben Ayed, Tunis, A. Mili, NJIT, F.T.  
Sheldon, ORNL, M. Shereshevsky, WVU  
Monday, May 16, 2005

## Introduction

- A priviledge to speak at this symposium
- A pleasure to acknowledge and honor the broad, far-reaching contributions of Dr Basili.
- work presented here is based on relational interpretation of programs and specifications
- is an outgrowth of functional/ relational approach advocated by Vic and his group
- Relations vs pre/post-conditions: algebraic structure; body of programming laws (relational algebra) that precedes programming.

## Outline

- Dependability: A Multi-dimensional attribute
- An Integrated Approach to Reliability
- A Uniform Representation for Dependability
- Inference Rules
- Applications

preliminary work; raise questions rather than provide answers. Partially funded by ORNL; interest in security.

## Dependability: A Multi Dimensional Attribute

Four Dimensions to Dependability:

- *Availability*: Probability of providing services when needed.
- *Reliability*: Probability of Failure Free Operation.
- *Safety*: Probability of Disaster Free Operation.
- *Security*: Probability of Interference Free Operation (exposure, intrusion, damage).

*Conceptually orthogonal, actually interdependent.*

## *Availability*

Depends on:

- Reliability.
- Repair Time.

Dependent on Reliability.

Related to Security: DoS affects availability.

## *Reliability*

Basic Concepts:

- *Fault*. System feature that may cause system to misbehave.
- *Error*. Impact of fault on system state. Early warning of misbehavior.
- *Failure*. Impact of fault on system (mis) behavior. Observable misbehavior.

System feature; State feature; Output feature.

## *Reliability, II*

Basic Means:

- *Fault Avoidance.* Fault Free Design.
- *Fault Removal.* Debugging/ Testing.
- *Fault Tolerance.* Error detection and recovery.

*Three successive, increasingly desperate, lines of defense.*

## *Safety*

Key Concepts:

- *Hazard.* System feature that makes accidents possible.
- *Mishap.* Operational conditions that makes accident imminent.
- *Accident.* Unplanned, undesirable event.
- *Damage.* Loss that results from an accident.

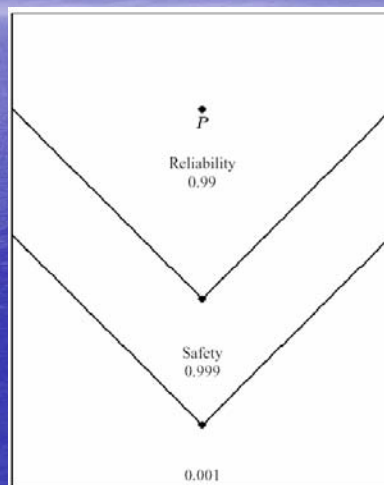
## *Safety, II*

Key Measures:

- *Hazard Avoidance.* Hazard Free design.
- *Hazard Removal.* Intervene before hazards cause accidents.
- *Damage Limitation.* Limit the impact of an accident.

*Three successive lines of defense.*

## *Reliability vs. Safety*



## *Security*

Key Concepts:

- *Vulnerability.* System feature that makes an attack possible.
- *Threat.* Situational condition that makes an attack possible.
- *Exposure/ Attack.* Deliberate or accidental loss of data and/or resources.

## *Security, II*

Key Measures:

- *Vulnerability Avoidance.* Vulnerability Free design.
- *Attack Detection and Neutralization.*  
Intervene before Attacks cause loss/ damage.
- *Exposure Limitation.* Limit the impact of attacks/ exposure. Intrusion-tolerance.  
*Three successive lines of defense.*

## *Special Role of Security*

- *Without security, there can be no reliability nor safety.* All claims about reliability and safety become void if the system's data and programs can be altered.
- *Without reliability, there can be no security.* Security measures can be viewed as part of the functional requirements of the system.

## Security: treated differently

- Mills and Dyer: variance of 1 to 50 in impact of faults on failures
- remove 60% of the faults, improve reliability by a dismal ... 3%.
- Adams, study of IBM software products: some faults cause failure after hundreds of thousands of months of product usage

## Analytical argument shows why

- faults do not necessarily cause errors.
- errors may be maskable.
- unmaskable errors may be recoverable.
- unrecoverable errors may produce outputs within tolerance of specification.

## Analogy

- it is conceivable that vulnerabilities have as tenuous a relation to reliability as faults have to reliability
- one concession: faults are sensitized at random, vulnerabilities are not.
- little acknowledgement in security literature.

- Stop equating security with freedom from vulnerabilities.
- Stop chasing vulnerabilities.
- Stop measuring security by (hypothesized) number of vulnerabilities (faults per KLOC long since discredited as a measure of reliability).
- define an independent characterization of security, that focuses on external /visible / measurable security properties.
- Focus security improvement on measures that lead us to the highest impact vulnerabilities first.

## A Logic for System Security

- Security specification notation
- Security abstraction notation
- Security certification formula

## *Integrated Approach to Reliability*

- Three Broad families of methods: Fault avoidance, fault removal, fault tolerance.
- *Which works best?* Spirited debate, religious arguments, dogmatic positions, jokes, etc.

*Pragmatic position:* use all three in concert, whenever possible/ warranted.

## *Rationale for Eclectic Approach*

- *The Law of Diminishing Returns.*
- *Method effectiveness varies greatly according to specification.*
- *Refinement calculus allows us to compose verification efforts/ claims.*
- *Refinement calculus allows us to decompose verification goals.*

## Pragmatic Rationale

- tested program against some test data, using some oracle.
  - inspected the program for some key property.
  - added some assert statements to improve error detection/ fault tolerance.
- ... so what?

## *Mapping Methods to Specifications*

- ***Proving:*** Reflexive, transitive relations.
- ***Testing:*** Relations that are good candidates for oracles (reliably implemented).
- ***Fault Tolerance:*** Unary relations that are reliably and efficiently implemented.

## Composing Verification Effort

- All methods must be cast in a common logical framework.
- Refinement calculus (based on relations) offers such a common framework.
- Specifications and programs are relations; refinement ordering between relations; lattice properties.

## *Modeling Verification Methods*

- *Proving*: Proving that  $P$  is correct with respect to specification  $V$ :

$$P \supseteq V.$$

- *Testing*: Certification testing, Oracle  $\Omega$ , test data  $D'$ , successful test on  $D$ .

$$P \supseteq T,$$

where  $T =_{D'} \Omega$ .

## *Modeling Verification Methods, II*

- *Fault Tolerance*: Upon each recovery block, check condition  $C$ , invoke recovery routine  $R$ . Because we do not know which outcome we have each time, all we can claim is:

$$P \supseteq F,$$

where  $F = C \cap R$ .

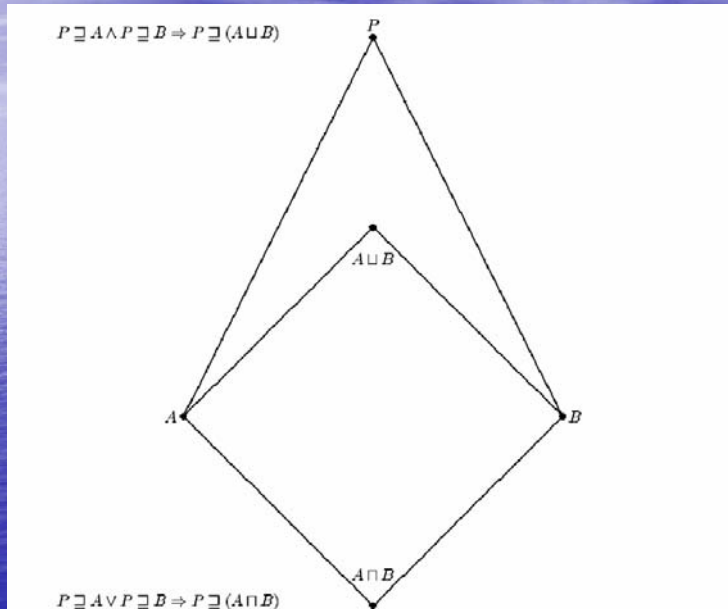
## *Cumulating Results*

- Proving:  $P \supseteq V$ .
- Testing:  $P \supseteq T$ .
- *Fault Tolerance*:  $P \supseteq F$ .
- *Lattice Identity*:

$$P \supseteq (V \cup T \cup F).$$

Cumulating verification results into a comprehensive correctness claim.

## Lattice Properties



## Decomposing Verification Goals

Premises:

- A Complex Specification can be decomposed into simpler sub-specifications in a refinement-compatible manner:

$$S = S_1 \cup S_2 \cup \dots \cup S_N.$$

- We can consider each  $S_i$  in turn, mapping it to the method that is most efficient for it.

## *Mapping Specifications to Methods*

Features Methods	Arity		Reflexivity and Transitivity		Coding Complexity		Execution Time		Inductive Reasoning	
	1	2	Y	N	L	H	L	H	possible	impossible
Proving	-1	1	1	-1	0	0	0	0	1	-1
Testing	0	0	0	0	1	-1	0	0	0	1
Tolerance	1	-1	-1	1	1	-1	1	-1	1	-1

	<i>SamSol</i>	<i>TriAng</i>	<i>Prec</i>
Proving	3	-2	-3
Testing	-1	2	3
Tolerance	-3	5	2

## *A Uniform Representation for Dependability Measures*

Logical representation of verification results unrealistic:

- Most verification results are best viewed as probabilistic, not logical, statements.
- Most verification results are conditional, contingent upon different conditions.
- Many verification results can be interpreted in more than one way.

## *Probabilistic (vs Logical) Claims*

- No absolute certainty.
- Even highly dependable, totally formal, verification systems may fail.
- We want to quantify level of confidence.

## *Verification Results are Conditional*

- *Proving*: Conditional on verification rules being consistent with actual compiler/ being borne out by runtime environment.
- *Testing*: Conditional on testing environment being identical to/ more stringent than operating environment.
- *Fault Tolerance*: Conditional on system preserving recoverability.

## *Multiple Interpretations*

- *Testing, first interpretation:*  $P \supseteq_{D, \Omega}$ , with probability 1.0.
- *Testing, second interpretation:*  $P \supseteq \Omega$ , with probability  $p < 1.0$ , conditional on D being representative.

*Which interpretation do we choose?* We do not have to choose, in fact. We can keep both, and learn to add/ cumulate them.

## *Characterizing Verification Claims*

- *Property.* Correctness preservation, recoverability preservation, security property, operational attribute.
- *Reference.* Functional Specification, Safety Requirement, Security Requirement, etc.
- *Assumption.* Implicit conditions in each method.
- *Certainty.* Quantified by probability.
- *Stake/ failure cost.* Cost of failure to satisfy a property with respect to a reference.
- *Penalty/ verification cost.* The cost of performing a verification task (for a given property/ reference/ assumption, etc).

## Uniform Representation

- Conditional probability:

$$\Pi(S \supseteq R \mid A) = P.$$

Two additional cost functions:

- Verification cost:

$$\text{Prop} \times \text{Ref} \times \text{Meth} \times \text{Assum} \rightarrow \text{Cost.}$$

- Failure cost:

$$\text{Prop} \times \text{Ref} \rightarrow \text{Cost.}$$

## Modeling goals

- Reliability: S refines R, contingent upon assumption VA (verification rules borne out by runtime environment)
- Safety: Same as reliability, much higher failure cost, hence higher probability of failure free operation.

## Security (vs intrusion)

- Specification: relation from event traces to outputs and responsiveness.
- Abstraction: relation from event traces to capability matrix.
- Certification: that for each event trace, behavior is within functional/ operational tolerances.

conflict between security (reduce capabilities) and reliability (give enough capabilities to get the job done).

## Modeling Methods

- Correctness verification, testing, fault tolerance: refinement property.
- Recoverability verification: recoverability preservation.
- Security measures: certification property.
- Model checking: functional/ operational property.
- Verification of Adaptive Systems: partial functional properties.

## *Sample Representations*

Certification Testing:

- First:  $\Pi(S \supseteq_D \Omega \mid TA) = 1.0$ ,  
where D is successful test data, TA is condition about testing environment.
- Second:  $\Pi(S \supseteq \Omega \mid TA \ \& \ R) = 0.7$ ,  
where R is representativity of D.

## *Sample Representations*

Formal Verification:

- $\Pi(S \supseteq R \mid VA) = 0.99$ ,  
where R is system specification, VA is condition that the verification method borne out by the operational environment.

## *Inference Rules*

- Collecting claims is insufficient.
- Cumulating/Synthesizing claims (as we did with logical results) is impractical.
- ⇒ Build inference mechanisms that can infer conclusions from a set of claims.

We will explore applications of this capability, subsequently.

## *Inference Rules*

Derived from refinement calculus (tentative; to be double checked):

- $\Pi(S \supseteq (R_1 \cup R_2) \mid A)$   
 $\geq \Pi(S \supseteq R_1 \mid A) \times \Pi(S \supseteq R_2 \mid A).$
- $\Pi(S \supseteq (R_1 \cap R_2) \mid A)$   
 $\leq \Pi(S \supseteq R_1 \mid A) + \Pi(S \supseteq R_2 \mid A).$
- $\Pi(S \supseteq (R_1 \cap R_2) \mid A)$   
 $\geq \max(\Pi(S \supseteq R_1 \mid A), \Pi(S \supseteq R_2 \mid A)).$

## *Inference Rules*

Derived from Probability Calculus (tentative):

- $R_1 \supseteq R_2 \Rightarrow \Pi(S \supseteq R_1 | A) \leq \Pi(S \supseteq R_2 | A)$ .
- $(A_1 \Rightarrow A_2) \Rightarrow \Pi(S \supseteq R | A_1) \leq \Pi(S \supseteq R | A_2)$ .
- $\Pi(S \supseteq R | A \wedge B) =$   
 $\Pi(S \supseteq R | A) \times \Pi(S \supseteq R | B) / \Pi(S \supseteq R)$ .
- Bayes' Theorem.

## *Inference Rules*

Derived from Cost functions (tentative/ illustrative):

- $R_1 \supseteq R_2 \Rightarrow VC(\supseteq, R_1, M, A) \geq VC(\supseteq, R_2, M, A)$ .
- $(A_1 \Rightarrow A_2) \Rightarrow$   
 $VC(\supseteq, R, M, A_2) \geq VC(\supseteq, R, M, A_1)$ .
- $R_1 \supseteq R_2 \Rightarrow FC(\supseteq, R_1) \geq FC(\supseteq, R_2)$ .

## Inference Rules

Derived from Relations Between properties:

- Highlights relations between various properties.
- Preserves correctness: Preserves recoverability.
- Preserves recoverability wrt R: preserves correctness wrt  $F(R)$ .

## *General Applications*

Providing dependability:

- Deploy eclectic approaches.
- Dispatch goals to methods to control verification costs.
- Dispatch verification effort to verification goals to control failure costs.
- Budget verification cost.
- Minimize / assess failure risk (probability, severity of failure).

## *General Applications*

Assessing dependability:

- Deploy eclectic/ diverse approaches.
- Record all measures in knowledge base.
- Updating knowledge base as system evolves.
- Maximize coverage.
- Minimize overlap.
- Budget verification cost.
- Minimize / assess failure risk (probability, severity of failure).

## *General Applications*

Certifying dependability:

- Deploy eclectic/ diverse approaches.
- Budget certification cost.
- Target certification effort (certification goal inferred from claims established by certification activity).

## Sample Queries

- *Given a set of claims, with what probability can we claim that  $S$  refines  $R$  subject to  $A$ ? Greatest lower bound for  $\Pi(S \supseteq R | A)$ .*
- *Given a set of claims, and  $R, A, p$ , can we claim that  $S$  refines  $R$  under assumption  $A$  with probability at least  $p$ ? Theorem  $\Pi(S \supseteq R | A) \geq p$ .*
- *Given a set of claims, provide a weighted average of failure costs.* Mean of a random variable.
- *Given a set of claims, identify the specification, property and method maximize impact.* Maximizing derivative  $d FC / d VC$ .

## Conclusion

- A Uniform representation of dependability claims.
- A discipline for composing dependability claims, decomposing dependability goals.
- Means to highlight dependencies between dimensions of dependability.
- Integrated approach to dependability management: minimizing overall failure cost.
- Means to identify next steps in V&V.

## *Prospects*

Eclectic, yet Integrated, Approach.

- Allows us to model diverse approaches, and combine their results.
- Allows us to measure claims.
- Allows us to budget cost, risks.
- Allows us to model multiple stakeholders: multiple failure cost functions.

## *Relevant Wisdom*

*Une Science a l'age de ses instruments de mesure.*

Louis Pasteur.

*A Science is as advanced as its measurement tools.*