Trusting Software

It is a people, product, process and project thing

Larry Bernstein
lbernstein@ieee.org

What is Trustworthy Software?

Trustworthy software is secure, safe and reliable

The Airbus A320

Come Fly with Me

Background

- First civilian fly-by-wire computer system so advanced can land plane virtually unassisted
- No instrument dials – 6 CRTs

Crash June 26, 1988

- Mulhouse-Habsheim test airfield in Alsace, France
- The airplane software interpreted the low altitude/downed gear as "We're about to land“; would not allow the pilot to control the throttle.

Crash February 1990

- Indian Airlines A320 during final approach
- Speed drops to dangerously low level causing rapid descent
- A320 slams into golf course just short of runway

Crash January 1992

- Airbus A320 plows into pine forest near Mont Sainte-OdilMinimum approach altitude reads 4700 feet on instruments
- Height at impact: about 2500 feet

What’s the Problem?

- In all three crashes, the pilot claimed the plane was higher than indicated.
- Altitude read 67ft before the wheels had even left the ground!
- The fly-by-wire system could ignore pilot actions.

Poor Designs in A320

- Programmed landing maneuvers with bug in altitude calculation
- Warning system alerts only seconds before accident; no time to react
- Flight path angle and vertical speed indicator have the same display format; confuses pilots.

More Blame on Software

- Pilot is either extremely busy or extremely bored. During flight, they get a false sense of security.
- Error and warning messages during data entry are often indecipherable, so pilots ignore them.

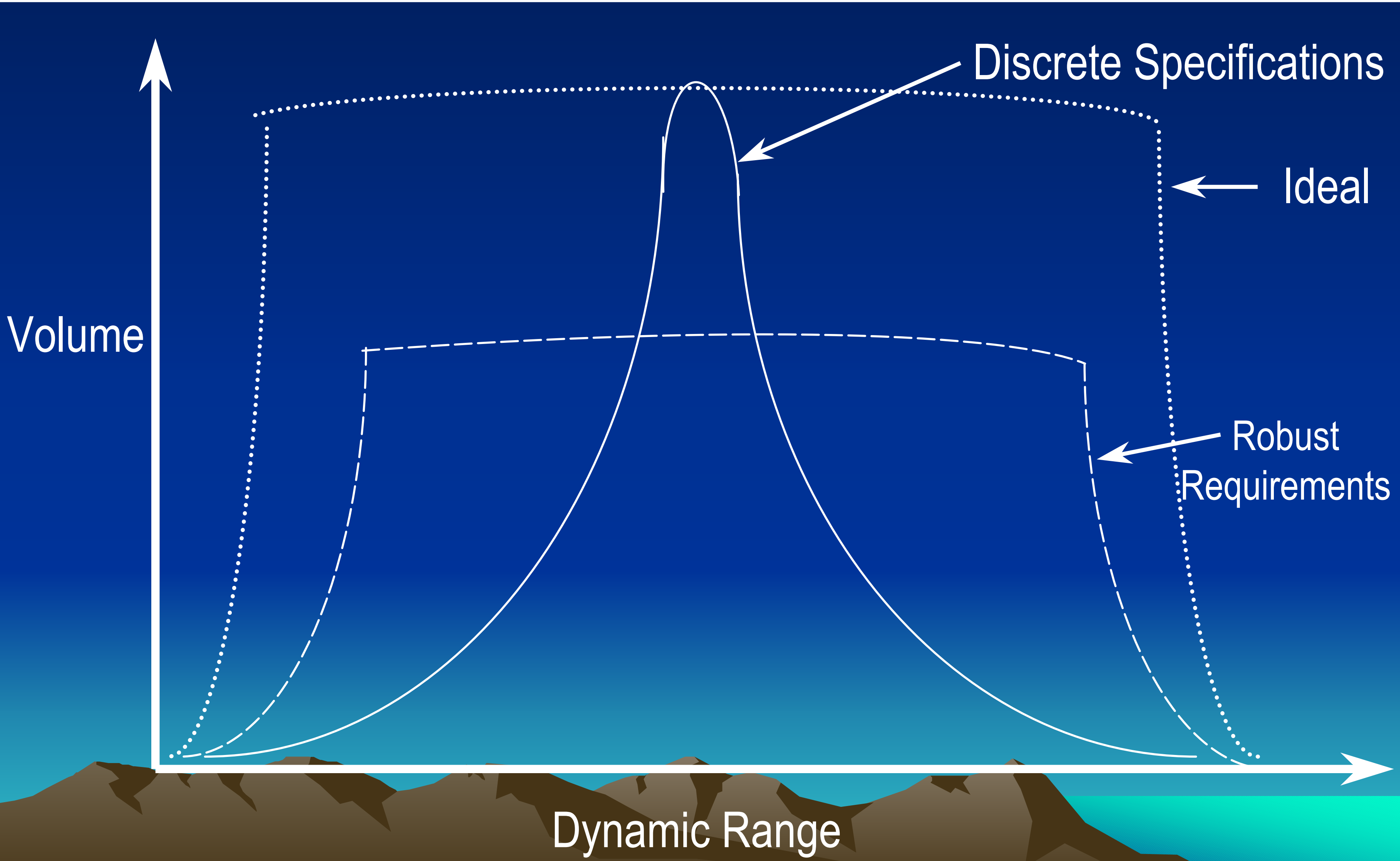
London Ambulance Dispatch Failure 1992

The major objective of the London Ambulance Service Computer Aided Dispatch project was to automate many of the human-intensive processes of manual dispatch systems associated with ambulance services in the UK.

Characteristics

- Designed for Trustworthiness
- Defined Development Process
- Bounded Execution Domains
- Certified against Requirements
- Certified against Problem
- Reliability Tested
- Stress Tested
- Diabolically Tested

System Performance Resulting from Robust Requirements vs. Discrete Specifications



Universal Software Engineering Equation

For constant error rate:

Reliability (t) = exp (-k λt)

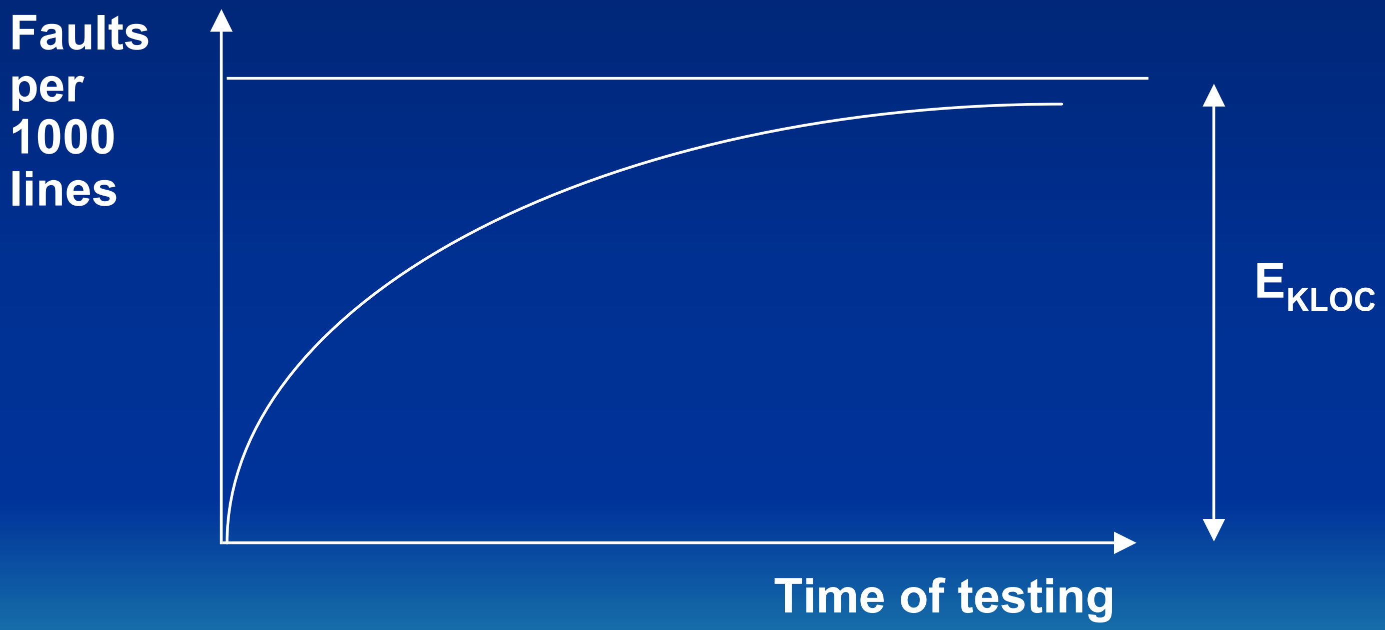
where k is a normalizing constant,

λ = Complexity/ effectiveness x staffing

For linear increasing error rate:

Reliability (t) = ?

Fault density is a function of time



Conditions That Cause Instability

- Poor Algorithms
- Missing Deadlines
- Roundoff Error Build Up
- Memory Leaks
- Broken Pointers
- Register Misuse

People

Software Trustworthiness depends on people:

I propose that customers insist that software products identify a Software Architect and Software Project Manager in their contracts

Software Architect:

- Affirms that the software product solves the customer’s problem
- Affirms that the software product is suitably reliable, easy-to-use, extendible, not harmful and robust. That it is trustworthy.
- Affirms that the requirements are valid.

Software Project Manager:

- Affirms that the software was successfully tested against the requirements.
- Affirms and identifies the good software engineering processes were used in the software development and integration.
- Affirms that the project is within budget, on-time and performs satisfactorily.

Trustworthy Software is:

- Safe: Does no harm
- Reliable: No crash or hang.
- Secure: No Hacking Possible

Who does it?

Designed and Implemented by Professionals committed to IEEE/ACM ethical code

ACM/IEEE Ethics

1. PUBLIC - Software engineers shall act consistently with the public interest.

2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.

ACM/IEEE Ethics

5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Introduction to Deadlocks

- Formal definition :
  A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause
- None of the processes can …
  - run
  - release resources
  - be awakened
- Number of processes and resources is unimportant

Four Conditions for Deadlock

1. Mutual exclusion condition
    - each resource assigned to 1 process or is available
2. Hold and wait condition
    - process holding resources can request more
3. No preemption condition
    - previously granted resources cannot forcibly taken away
4. Circular wait condition
    - must be a circular chain of 2 or more processes
    - each is waiting for resource held by next member of the chain

STEVENS
Institute of Technology

Master of Science Quantitative Software Engineering

Offered by
Arthur Imperatore School of
Sciences and Arts
2003

STEVENS
Institute of Technology

Master of Science in Quantitative Software Engineering

:

- 7 Required Courses:
  - CS 540 -- Fundamentals of Quantitative Software Engineering
  - CS 533 -- Cost Estimation & Metrics
  - CS 564 -- Software Requirements Acquisition and Analysis
  - CS 565 -- Software Architecture and Component based Design
  - CS 567 -- Software Testing, Quality Assurance, and Maintenance
  - CS 568 Software Project I / CS 687 Large Software Systems
  - CS 569 Software Project II / CS 689 Software Reliability
- 3 Elective Courses: Any 3 CS Courses. Sample Choices:
  - CS 573 -- Fundamentals of Cyber Security
  - CS 668: Foundations of Cryptography
  - CS 693: Cryptographic Protocols
  - CS 694: E-business Security and Information Assurance

Vision

- Reliable Software Products produced within budget and on-time.
- Metrics based Software Management
- Quantitative Software Design
- Understanding Technical Foundations
- Repeatable Processes
- State-of-the-Art

Innovations

- Design Simplification
- Live-Thru Case Histories
- Reliability Based Software Engineering
- Universal Equation used for tradeoff analysis
- Model Driven Design
- COTS Acquisition

Trustworthy Systems for Today and Tomorrow

STEVENS
Institute of Technology