Software Bugs

Sandro Morasca and Dario Bertolino

Università degli Studi dell'Insubria

Dipartimento di Scienze Teoriche e Applicate

Via Ottorino Rossi 9 – Padiglione Rossi

I-21100 Varese, Italy

{sandro.morasca,dario.bertolino}@uninsubria.it

Why Testing and Analysis

- Software is never correct no matter the development technique used
- Any software must be verified
- Software testing and analysis are
  - important to control the quality of the product (and of the process)
  - very (often too) expensive
  - difficult and stimulating

- Project Mercury was the first human spaceflight program of the United States, running from 1958 through 1963. An early highlight of the Space Race, its goal was to put a man into Earth orbit and return him safely, ideally before the Soviet Union.
- Nasa, summer 1963 discover bug before flight
- Syntax error in FORTRAN code
  - DO I = 1.10 … DO I = 1,10
- Luckily detected before the flights to the Moon

Ariane V

- ESA (European Space Agency), 1996
- Missile failure: the launcher veered off its flight path, broke up, and exploded
  - 37 seconds after take-off
  - onboard computer made an abrupt correction
  - on a number that was … an error message
  - due to a problem when storing a 64-bit number in a 16-bit field
- ESA took 10 years to develop Ariane V
- Budget: $ 7 Billion
- Payload: 4 expensive and uninsured scientific satellites

Patriot Missile Misses

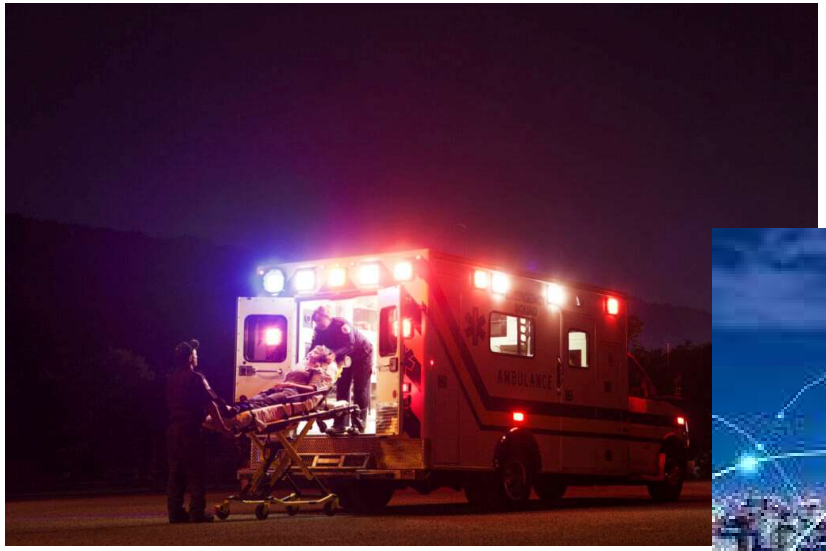
- US Military, 1991
- US Patriot missiles were designed to head off Iraqi Scud missiles during the 1991 Gulf war
- The system failed to track several incoming Scud missiles
  - one killed 28 soldiers
- Software defect put the tracking system off by 0.34 seconds
- The system was supposed to be operated only for 14 hours at a time
- The missile battery was on for 100 hours and the deviations accumulated

Therac-25 Radiation Therapy

- X-Ray machine used to treat cancer patients
- Six patients were subject to massive overdoses
  - three patients died
- Problem coud not be reproduced for years
  - vendor claimed overdosing was impossible
- Bug occurred only several steps including doing some settings, going back to correct a mistake, and finally starting treatment within short period of time
- The bug could actually be prevented, as some formal methods studies showed

Emergency System Problem

- San Francisco, 1989
- Death of a 5 year old
- Emergency system truncated the last digit in the street number
  - ambulance arrived too late



Iran Air Flight 655

- US Military, 1987
- A US battleship shot down an Iranian Passenger Flight
- 290 people died
- Cause: mistaking Airbus plane for an enemy military plane
- One of the problems was actually due to the user interface (!)

- The system let users buy lottery tickets and choose their numbers
  - after the drawing!
- Six people won that way

Toys 'R' Us

- Christmas shopping, 1999-2000
- $2 M net loss due to performance problems
- $4 M spent to increase performance
- $1.5 M in civil penalties
  - shipping occurred after Christmas
- $9 M net loss due to decline in on-line sales

eBay

- In 1999, there was a 21 hour outage
- $5 M net loss
- 11% drop in share price
- 1.2 M daily website visitors lost bids
- Outage occurred as eBay was getting ready for an open house for Wall Street analysts

Victoria's Secret

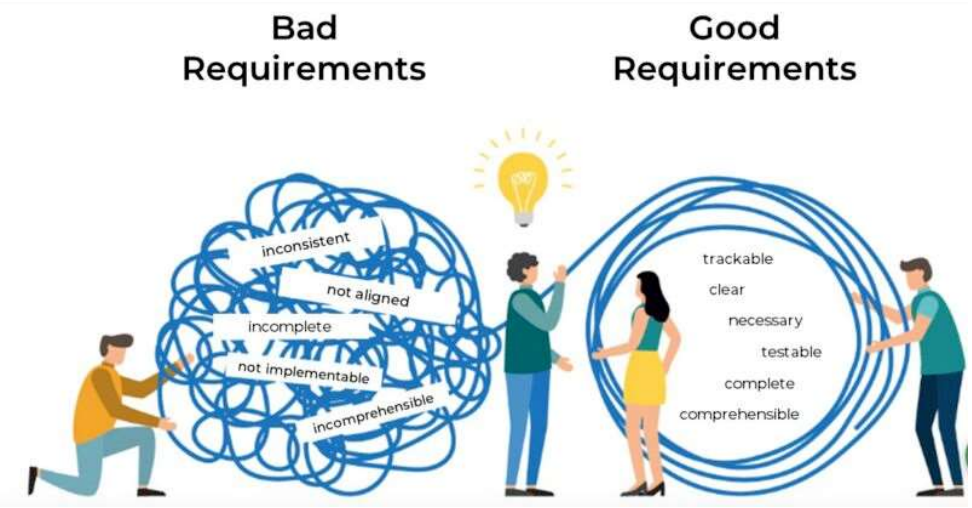
- Annual Spring Fashion show in 1999 on the internet crashed as millions of viewers tried to log on
- Advertised during Super Bowl XXXIII to attract the maximum number of viewers
  - Super Bowl ad cost for 30 seconds: $1.6 M
- The computers could handle at most a load of 500k simultaneous viewers
  - 5 M viewers tried to log on

Related Costs

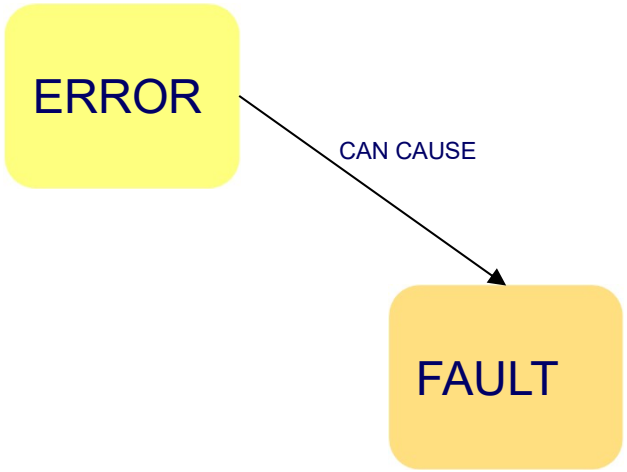
- The US National Institute of Standards and Technology found that software bugs cost the US economy about
  - $2.4 Trillion only in 2022
- More than a third of that cost could be removed by improving testing

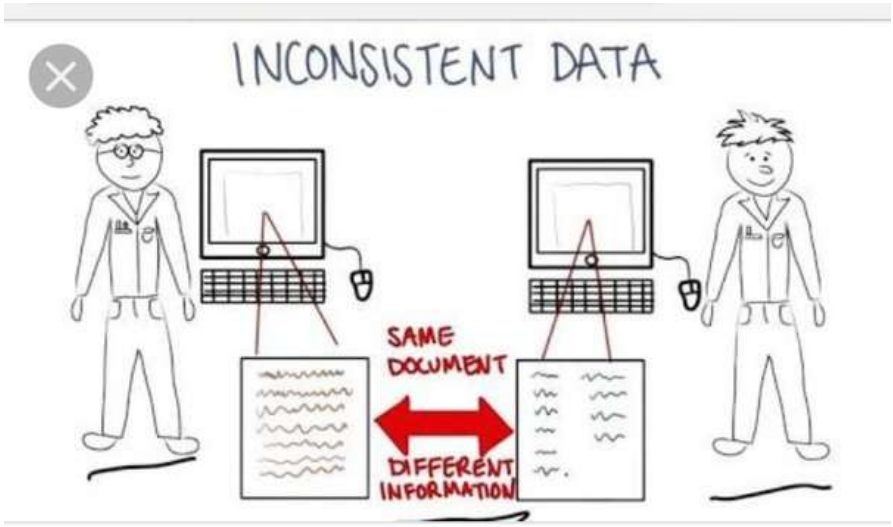
- **Error** (**mistake**): a human action that produces an incorrect result, e.g.,
  - logical error in software code, architectural misconception, oversight, poor communication

ERROR

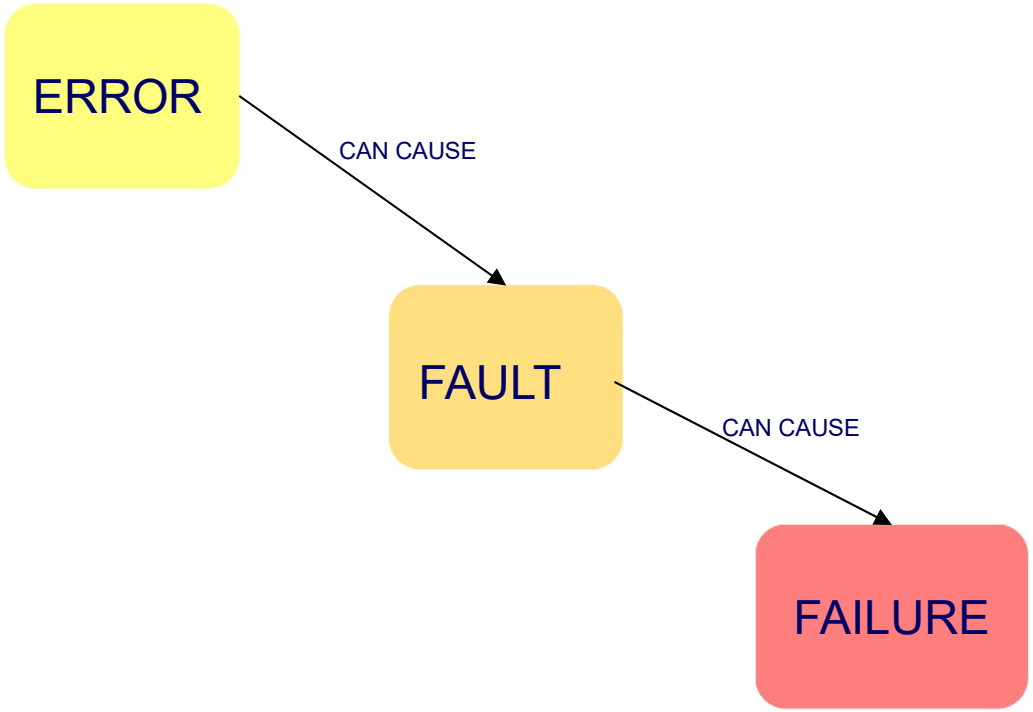


- **Fault** (defect): an incorrect step, process, or data definition in a component or system that can cause the component or system to fail to performed as specified.



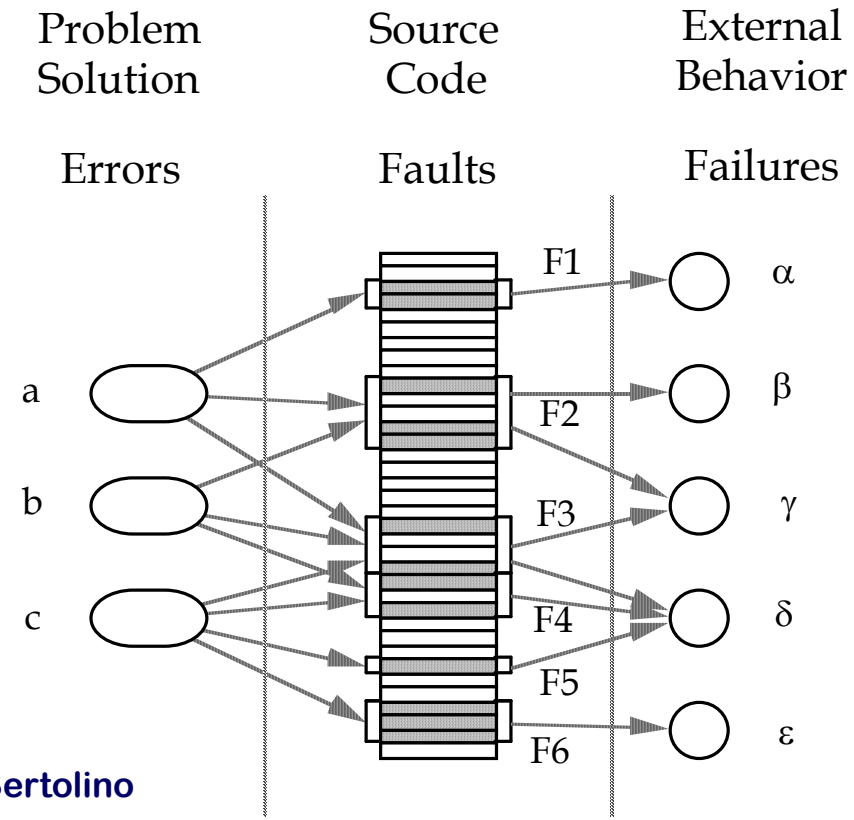


- **Failure**: Deviation of the component or system from its requirements/specifications. A fault, if encountered during execution, may cause a failure of the component or system.
  - e.g., system crash, incorrect result, insufficient performance, inability to satisfy real-time constraints, security hole ...



Standard Definitions

- One error may cause more than one fault
  - but one fault can be caused by more than one error
- One defect can cause more than one failure
  - but one failure can be caused by more than one fault

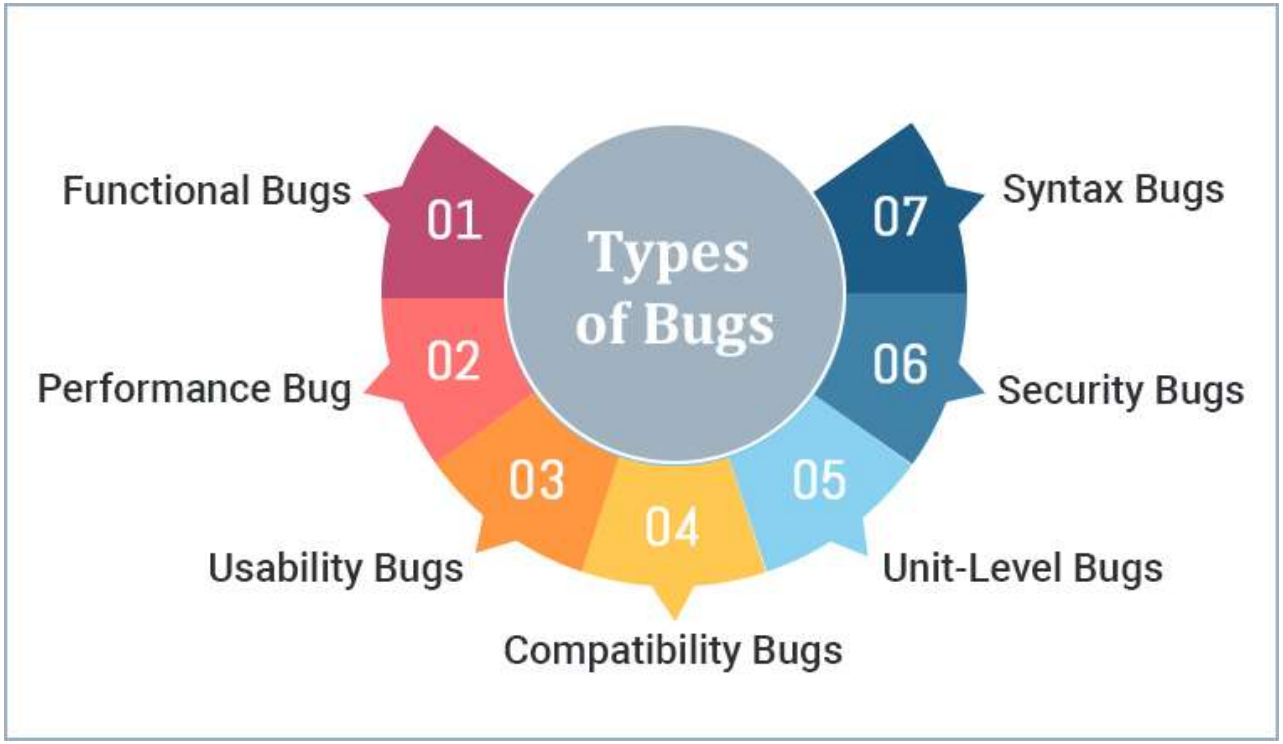


Good to Know

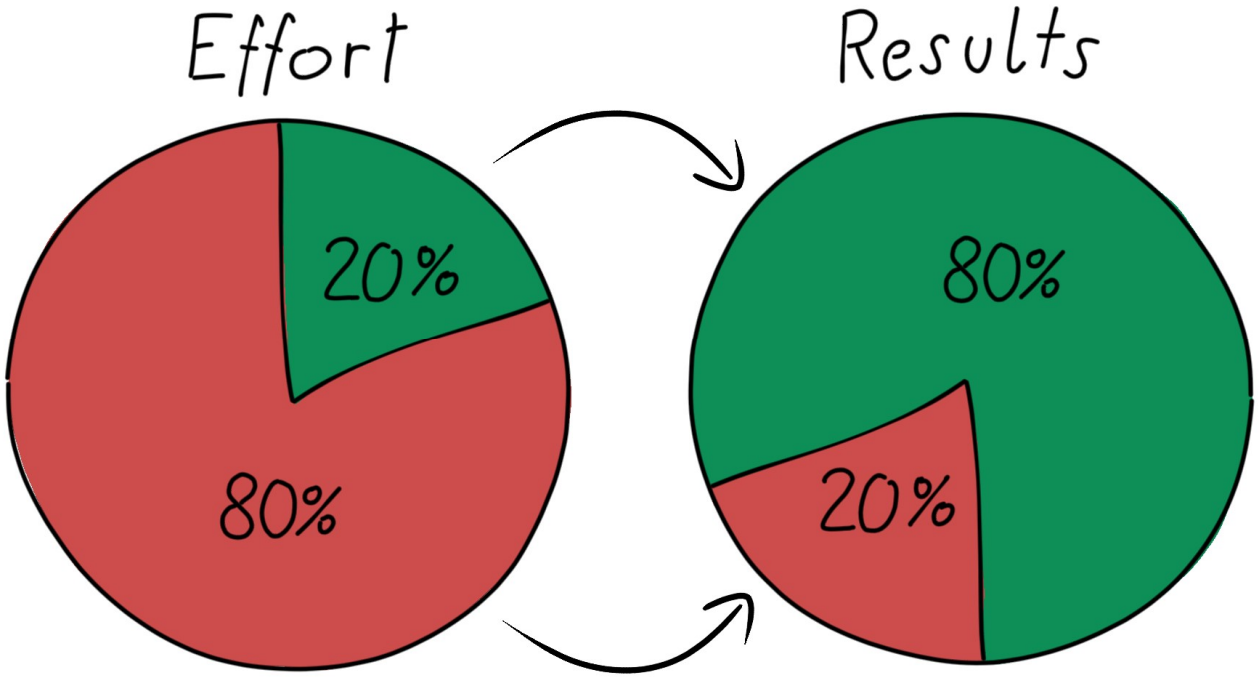
- Problems with failure reproduction
  - especially with concurrent software code
- Errors are usually difficult to identify
- Defects need to be defined precisely
  - e.g., if I have consistently written “j” instead of “i” in a Java method, is it one defect or several?
- Faults and Failures often go unnoticed,
  - at least for a while
  - Monitoring tools are important

Categorizing Faults

- There is no “right” categorization
  - may depend on design style, implementation language, process and documents, ...
  - should probably be revised occasionally



80/20 Rule (a.k.a. Pareto analysis)



80/20 Rule (a.k.a. Pareto analysis)

- Goal is enough precision for “Pareto” analysis (80/20 rule) considering severity and cost
  - categorization need not be perfect or painful, but keeping records is essential
- Identify one or two “dominant” fault categories
  - Considering severity, cost, and frequency
  - Further problem analysis is limited to these
- Categories may “level” over time
  - A good time for rethinking the categories

Types of Bugs: Kaner

- Error handling
  - prevention
  - detection
  - recovery
- Boundary conditions
  - numeric boundaries
  - boundaries in space, time
  - boundaries in loops
- Hardware
  - device unavailable
  - unexpected end of file

Types of Bugs: Kaner

- Data handling or interpreting
  - data type errors
  - parameter list variables out of the right order or missing
  - outdated copies of data
  - wrong value from a table
  - wrong mask in bit field
- Documentation
  - none
- Load conditions
  - required resource not available
  - doesn’t return unused memory

Types of Bugs: Kaner

- Source and version control
  - old bugs mysteriously reappear
  - source code doesn't match binary
- User Interface
  - functionality
  - communication
  - command structure
  - missing commands
  - performance
  - output

Types of Bugs: Kaner

- Control flow
  - program runs
  - program stops
  - loops
  - if then else or maybe not
- Initial and later states
  - failure to set a data item to 0
  - failure to initialize a loop control variable
  - failure to clear a string
  - failure to reinitialize

Types of Bugs: Kaner

- Calculation
  - outdated constants
  - calculation errors
  - wrong operation order
  - overflow and underflow
- Race conditions
  - assuming one event occurs before another
  - assuming that input will not occure in a specific interval
  - task starts before its prerequisites are met

Failure Severity: Beizer's Classification

- **Mild**: The symptoms of the bug offend us aesthetically; a misspelled output or misaligned command.
- **Moderate**: Outputs are misleading or redundant. The bug impacts system performance.
- **Annoying**: The system's behavior, because of the bug, is dehumanizing. Names are truncated or arbitrarily modified. Bills for $0.00 are sent. Operators must use unnatural command sequences and must trick the system into a proper response for unusual bug-related cases.
- **Disturbing**: It refuses to handle legitimate transactions. The ATM won't give you money. A credit card is declared invalid.
- **Serious**: It loses track of transactions: not just the transaction itself (your paycheck), but the fact that the transaction occurred. Accountability is lost.

Failure Severity: Beizer's Classification

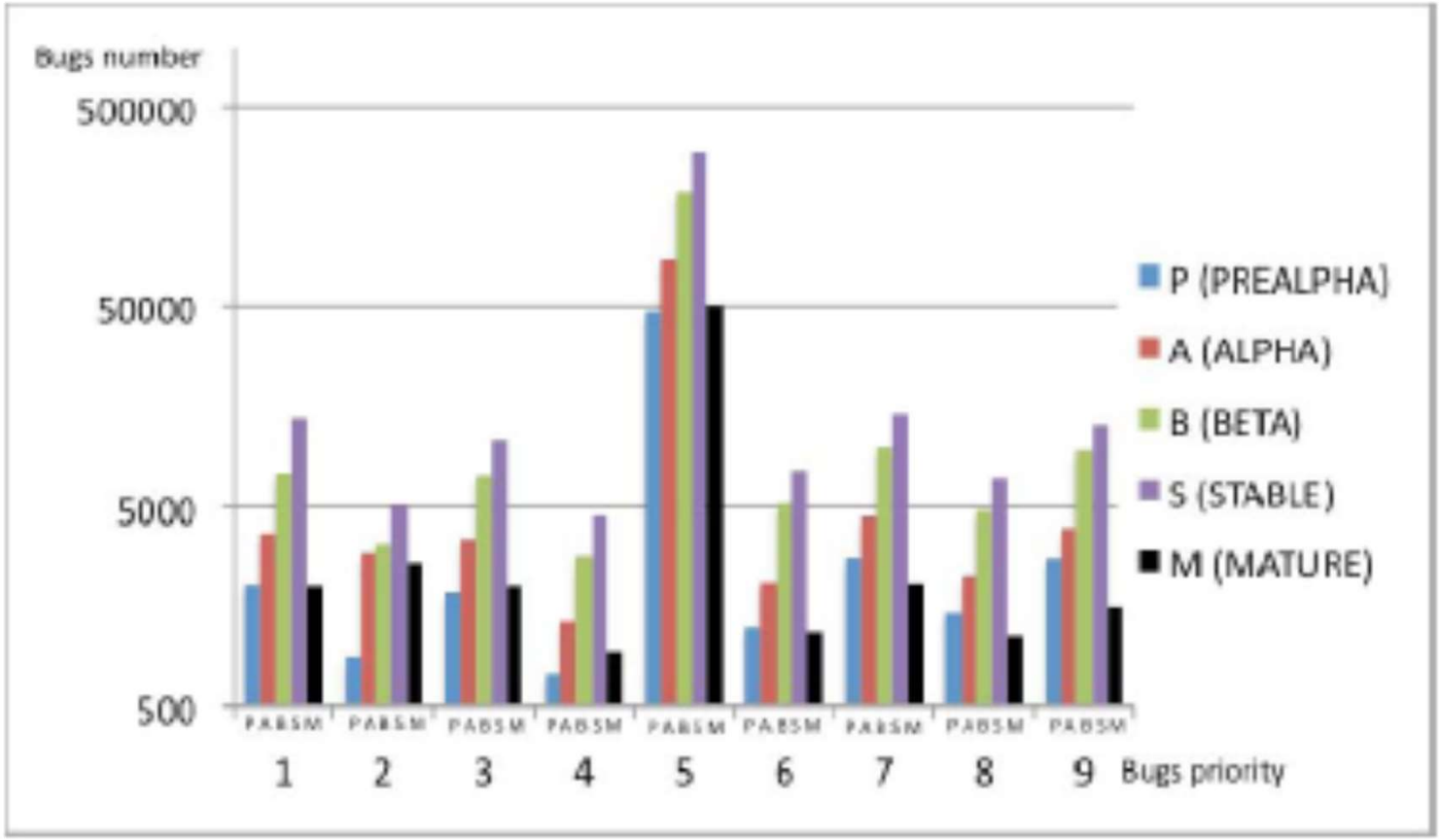
- **Very Serious**: Instead of losing your paycheck, the system credits to another account or converts deposits into withdrawals. The bug causes the system to do the wrong transaction.
- **Extreme**: The problems aren't limited to a few users or to a few transaction types. They are frequent and arbitrary, instead of sporadic or for unusual cases.
- **Intolerable**: Long-term, unrecoverable corruption of the database occurs and the corruption is not easily discovered. Serious consideration is given to shutting down the system.

Failure Severity: Beizer's Classification

- **Catastrophic**: The decision to shut down is taken out of our hands because the system fails.
- **Infectious**: What can be worse than a failed system? One that corrupts other systems or physical environment; whose influence, because of malfunction, is far greater than expected; a system that kills.

Failure Severity: Sourceforge Bug Distribution

- Logarithmic scale (!)
- 135834 projects
- Six months



Fixing Cost Based on Detection Time