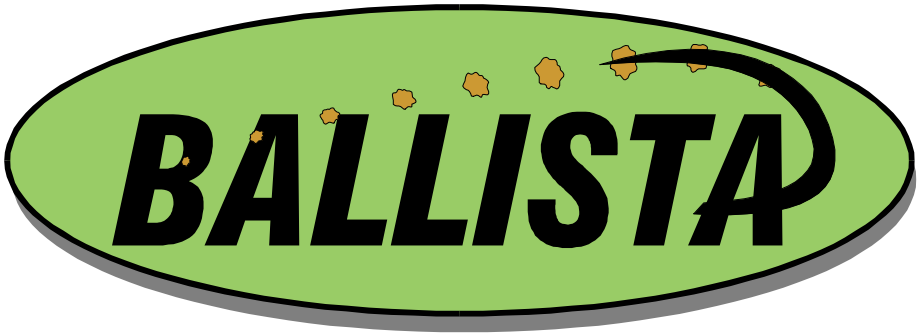
Software Robustness Evaluation



http://www.ices.cmu.edu/ballista

Prof. Philip Koopman

koopman@cmu.edu - (412) 268-5225 - http://www.ices.cmu.edu/koopman

(and more than a dozen other contributors)

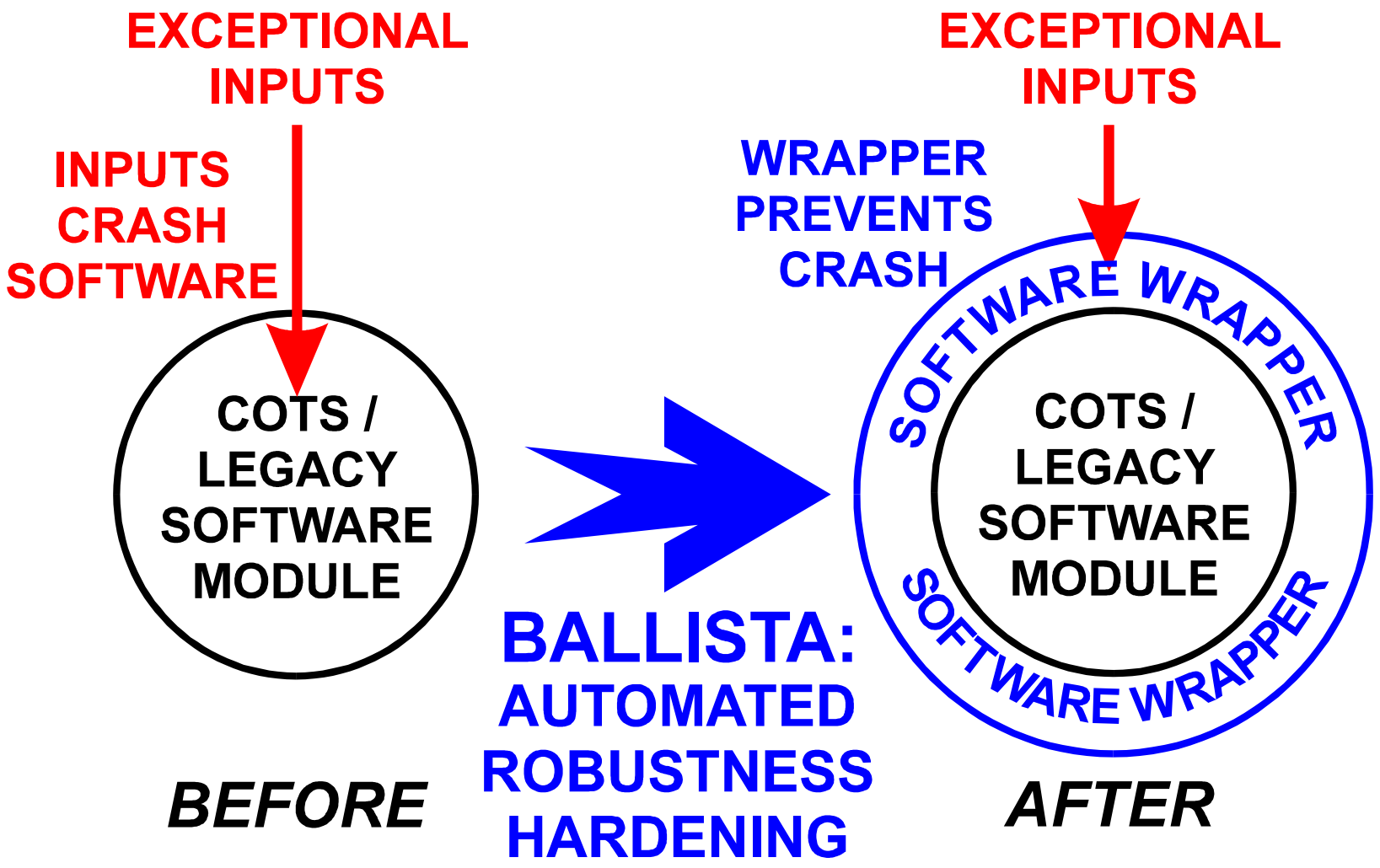
Institute for Complex Engineered Systems

Carnegie Mellon

Electrical & Computer ENGINEERING

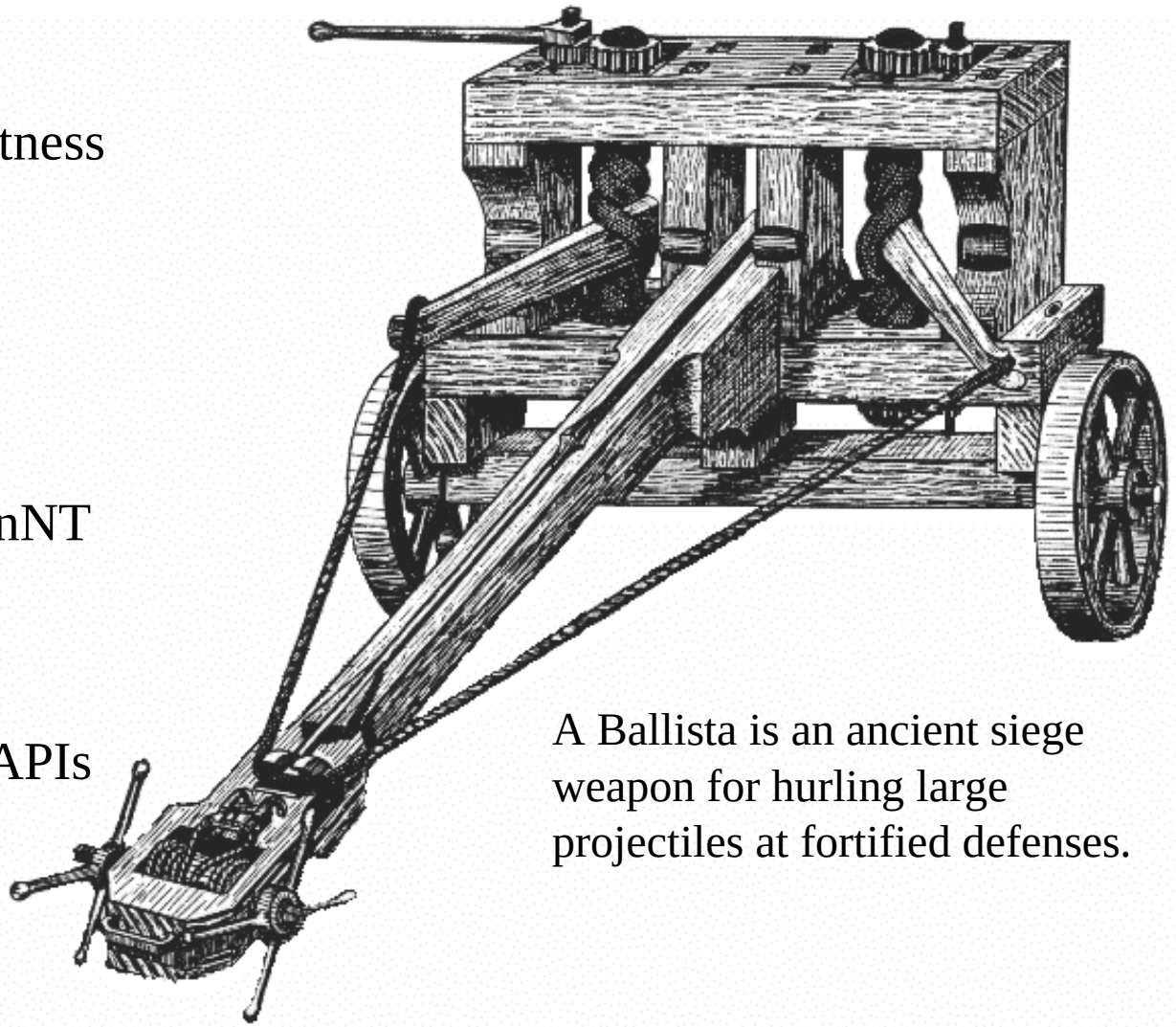
Where We Started: Component Wrapping

- **Improve Commercial Off-The-Shelf (COTS) software robustness**



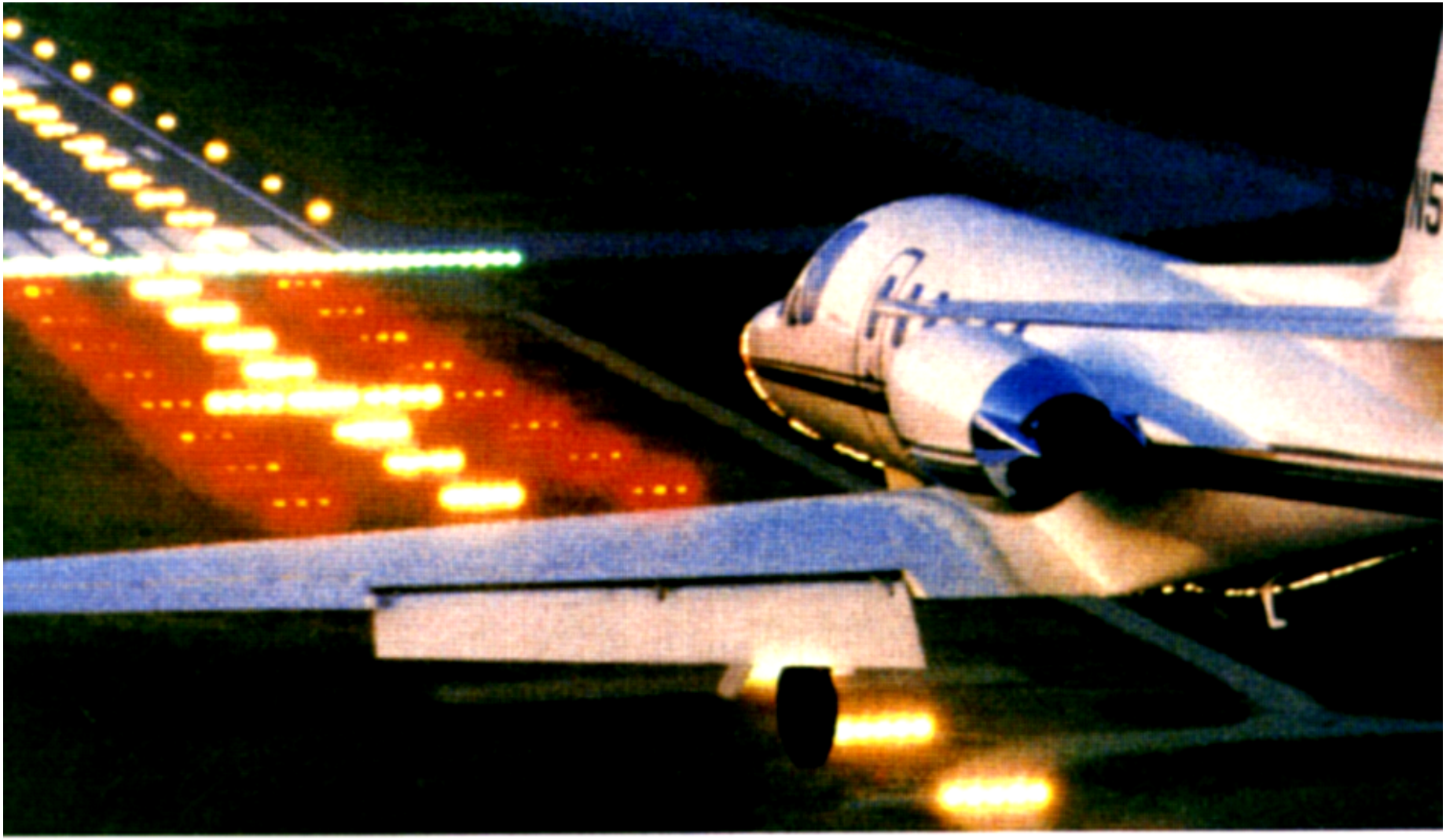
Overview: Automated Robustness Testing

- **System Robustness**
  - Motivation
  - Ballista automatic robustness testing tool
- **OS Robustness Testing**
  - Unix
  - Windows
  - Comparing Linux to WinNT
- **Testing Service**
  - Technology Transfer
  - Application to Non OS APIs
- **Conclusions**



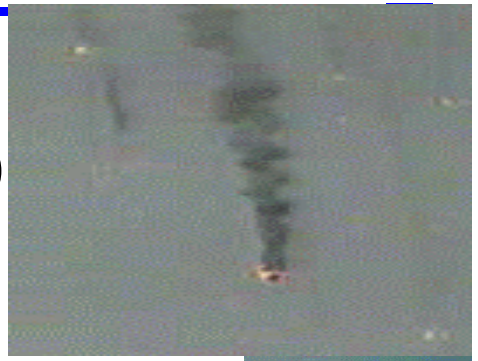
A Ballista is an ancient siege weapon for hurling large projectiles at fortified defenses.

BALLISTA



THIS IS A BAD PLACE TO DISCOVER YOUR RTOS IS ONLY 83.3% ROBUST.

Ariane 5 Flight 501 Robustness Failure



- **June, 1996 loss of inaugural flight**
  - Lost $400 million scientific payload (the rocket was extra)
- **Efforts to reduce system costs led to the failure**
  - Re-use of Ariane 4 Inertial Reference System software
  - Improperly handled exception caused by variable overflow during new flight profile (that wasn't simulated because of cost/schedule)
    - 64-bit float converted to 16-bit int *assumed* not to overflow
    - Exception caused dual hardware shutdown (because it was assumed software doesn't fail)
- **What really happened here?**
  - **The narrow view:** it was a software bug -- fix it
    - Things like this have been happening for decades -- Apollo 11 LEM computer crashed during lunar descent
  - **The broad view:** the loss was caused by a lack of system robustness in an exceptional (unanticipated) situation
- **Our research goal: *improved system robustness***

Good Exception Handling Improves Robustness

"If builders built buildings they way computer programmers write programs, the first woodpecker that came along would have destroyed all civilization" -- Gerald Weinberg

- **Exception handling is an important *part* of dependable systems**
  - Responding to unexpected operating conditions
  - Tolerating activation of latent design defects
  - (Even if your software is "perfect," what about other people's software?)

- **Robustness testing can help evaluate software dependability**
  - Reaction to exceptional situations (current results)
  - Reaction to overloads and software "aging" (future results)
  - First big objective: measure exception handling robustness
    - Apply to operating systems
    - Apply to other applications

- **It's difficult to improve something you can't measure … so let's figure out how to measure robustness!**

BALLISTA

Joe reboots his PC every day.

That's a fact.

Conventional OS Architecture
The monolithic OS on Joe's machine clumps all OS components into a single address space. One subtle programming error in just one driver, and ***whoomp!****, Joe has to reboot – again.*

Dave hasn't since 1994.

Measurement Part 1: Software Testing

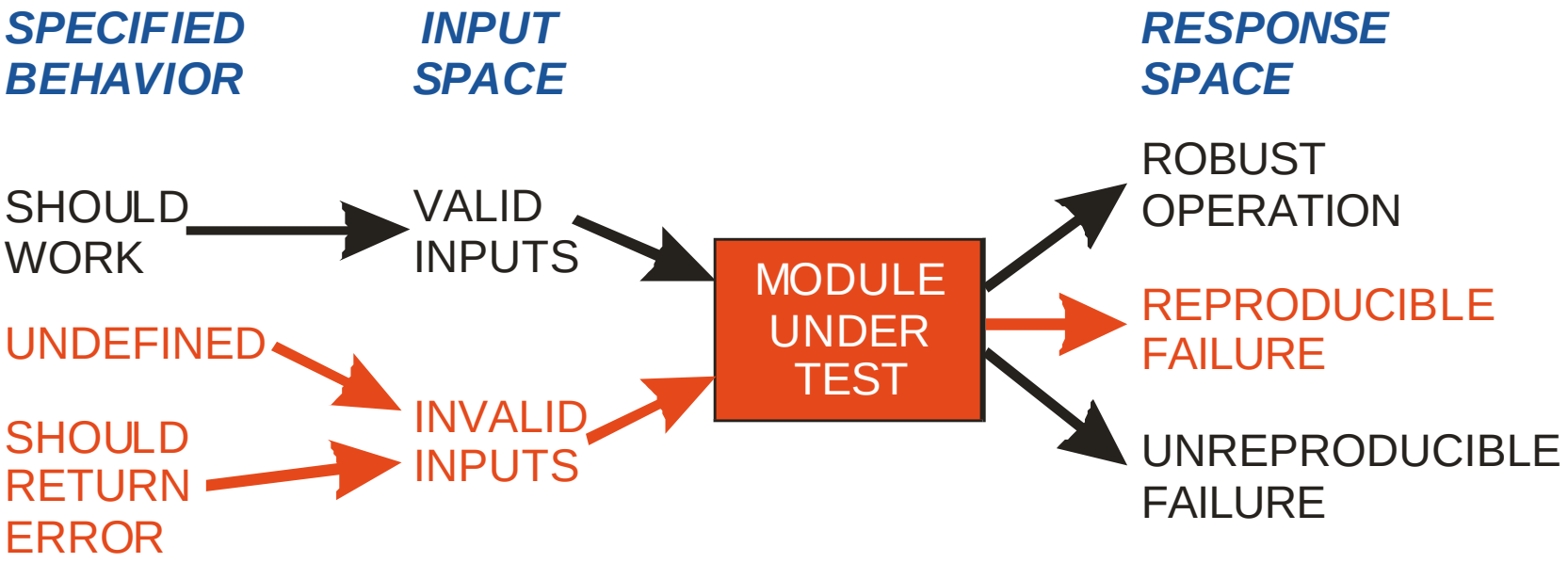
- **SW Testing requires:**
  - Test case
  - Module under test
  - *Oracle* (a "specification")

Ballista uses:

"Bad" value combinations

Module under Test

Watchdog timer/core dumps



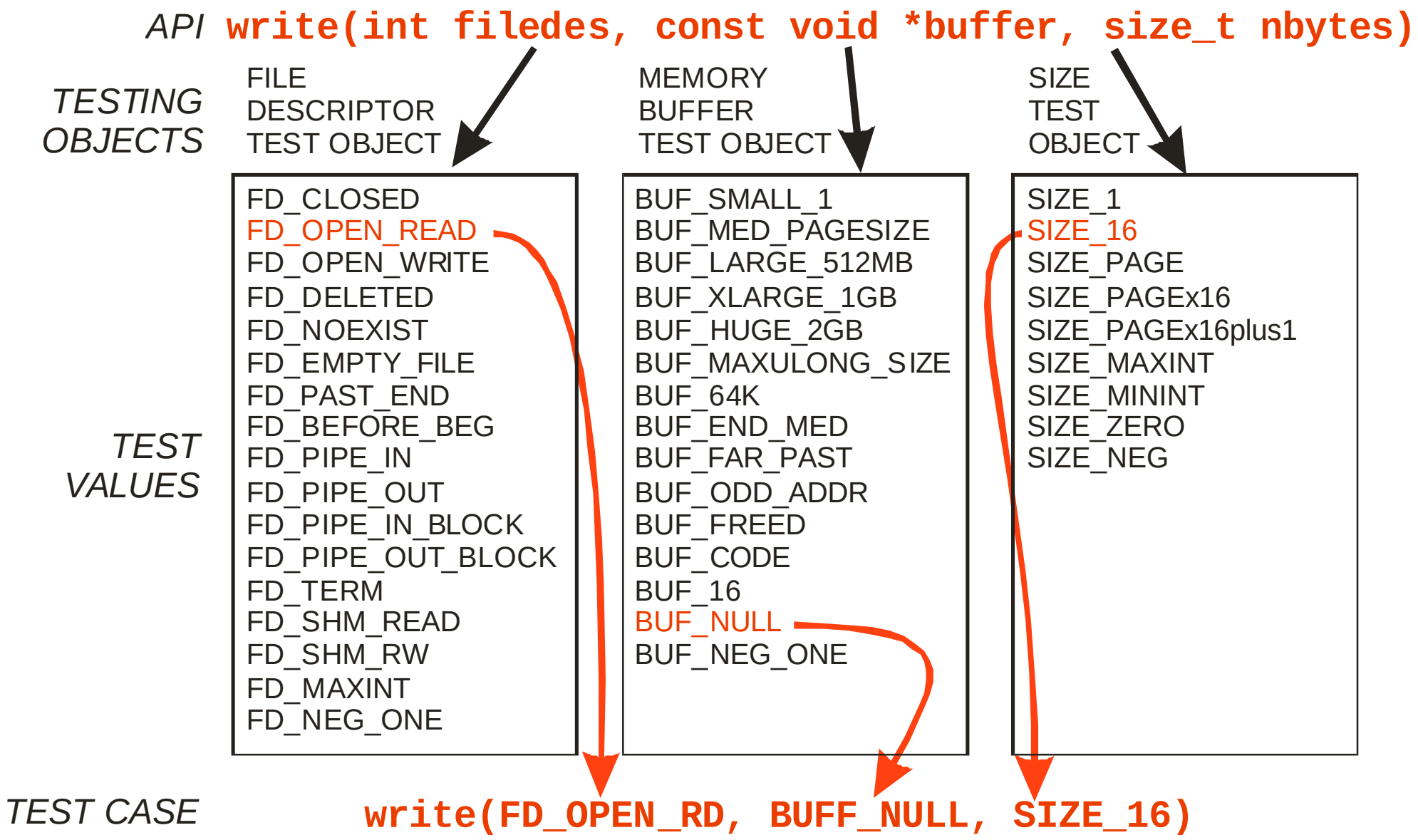
- **But, software testing is expensive**
  - Key idea: *use a very simple oracle*!

Measurement Part 2: Fault Injection

- Use repeatable, high level fault injection for inexpensive testing

Name	Method	Level	Repeatability
FIAT	Binary Image Changes	Low	High
FERRARI	Software Traps	Low	High
Crashme	Jump to Random Data	Low	Low
FTAPE	Memory/Register Alteration	Low	Medium
FAUST	Source Code Alteration	Middle	High
CMU-Crashme	Random Calls and Random Parameters	High	Low
Fuzz	Middleware/Drivers	High	Medium
Ballista	Specific Calls with Specific Parameters	High	High

Ballista: Scalable Test Generation



- **Ballista combines test values to generate test cases**

Ballista: "High Level" + "Repeatable"

- **High level testing is done using API to perform fault injection**
  - Send exceptional values into a system through the API
    - Requires no modification to code -- only linkable object files needed
    - Can be used with any function that takes a parameter list
  - Direct testing instead of middleware injection simplifies usage

- **Each test is a specific function call with a specific set of parameters**
  - System state initialized & cleaned up for each single-call test
  - Combinations of valid and invalid parameters tried in turn
  - A "simplistic" model, but it does in fact work...

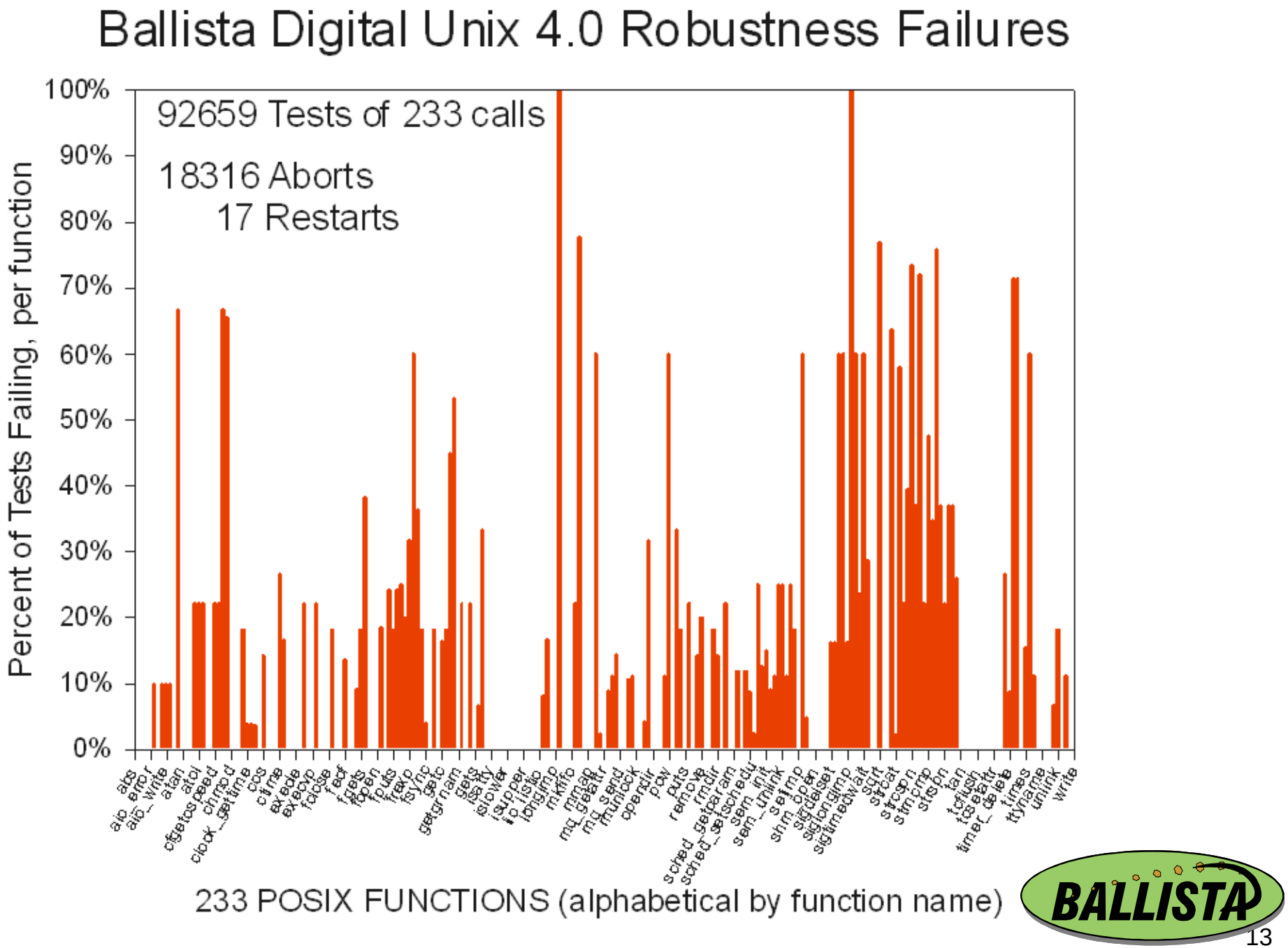
- **Early results were encouraging:**
  - Found a significant percentage of functions with robustness failures
  - Crashed systems from user mode

- **The testing object-based approach *scales!***

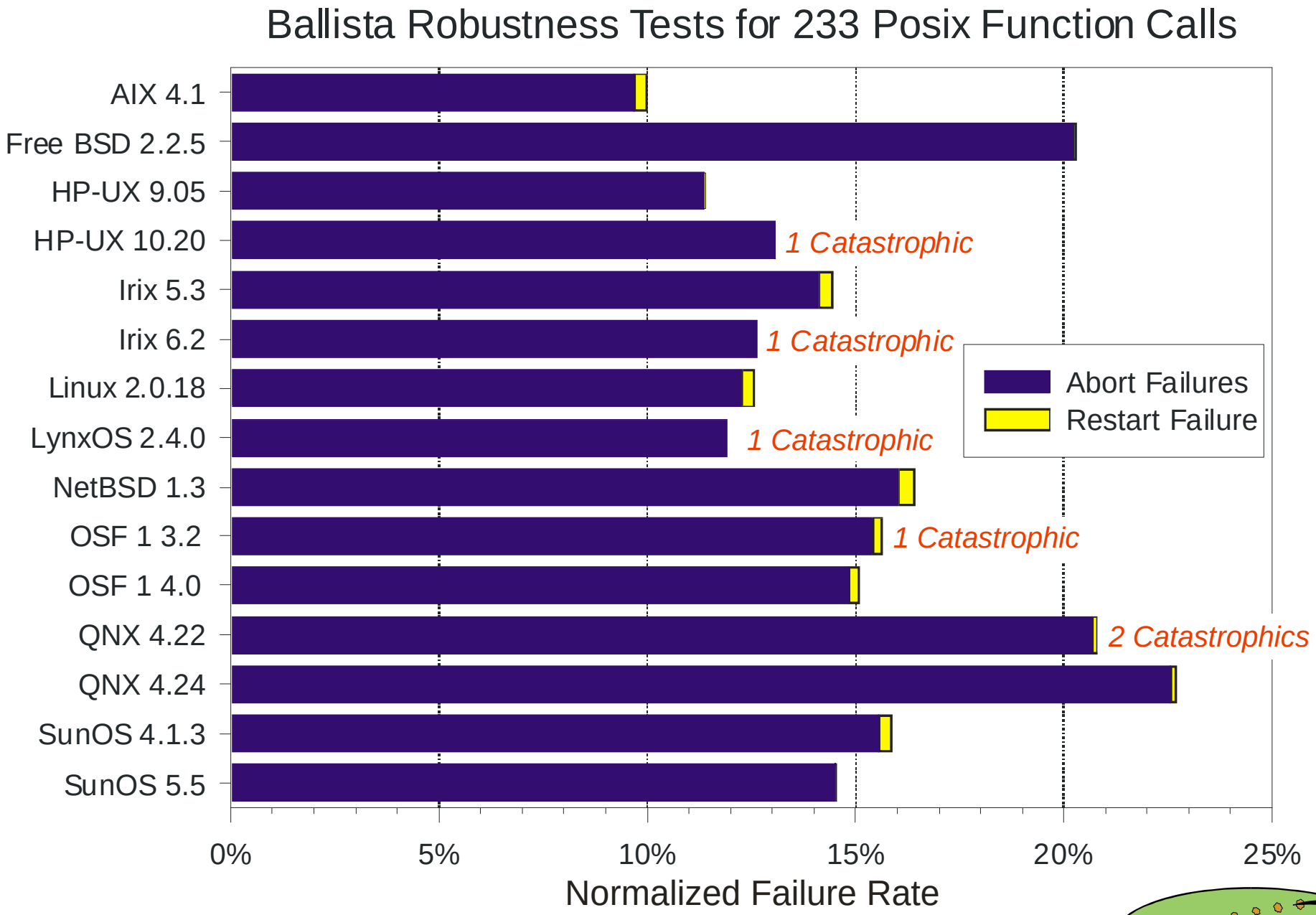
CRASH Robustness Testing Result Categories

- **Catastrophic**
  - Computer crashes/panics, requiring a reboot
  - *e.g.*, Irix 6.2: `munmap(malloc((1<<30)+1), ((1<<31)-1)) );`
  - *e.g.*, DUNIX 4.0D: `mprotect(malloc((1 << 29)+1), 65537, 0);`
- **Restart**
  - Benchmark process hangs, requiring restart
- **Abort**
  - Benchmark process aborts (*e.g.*, "core dump")
- **Silent**
  - No error code generated, when one should have been (*e.g.*, de-referencing null pointer produces no error)
- **Hindering**
  - Incorrect error code generated
  - Found via by-hand examinations, not automated yet

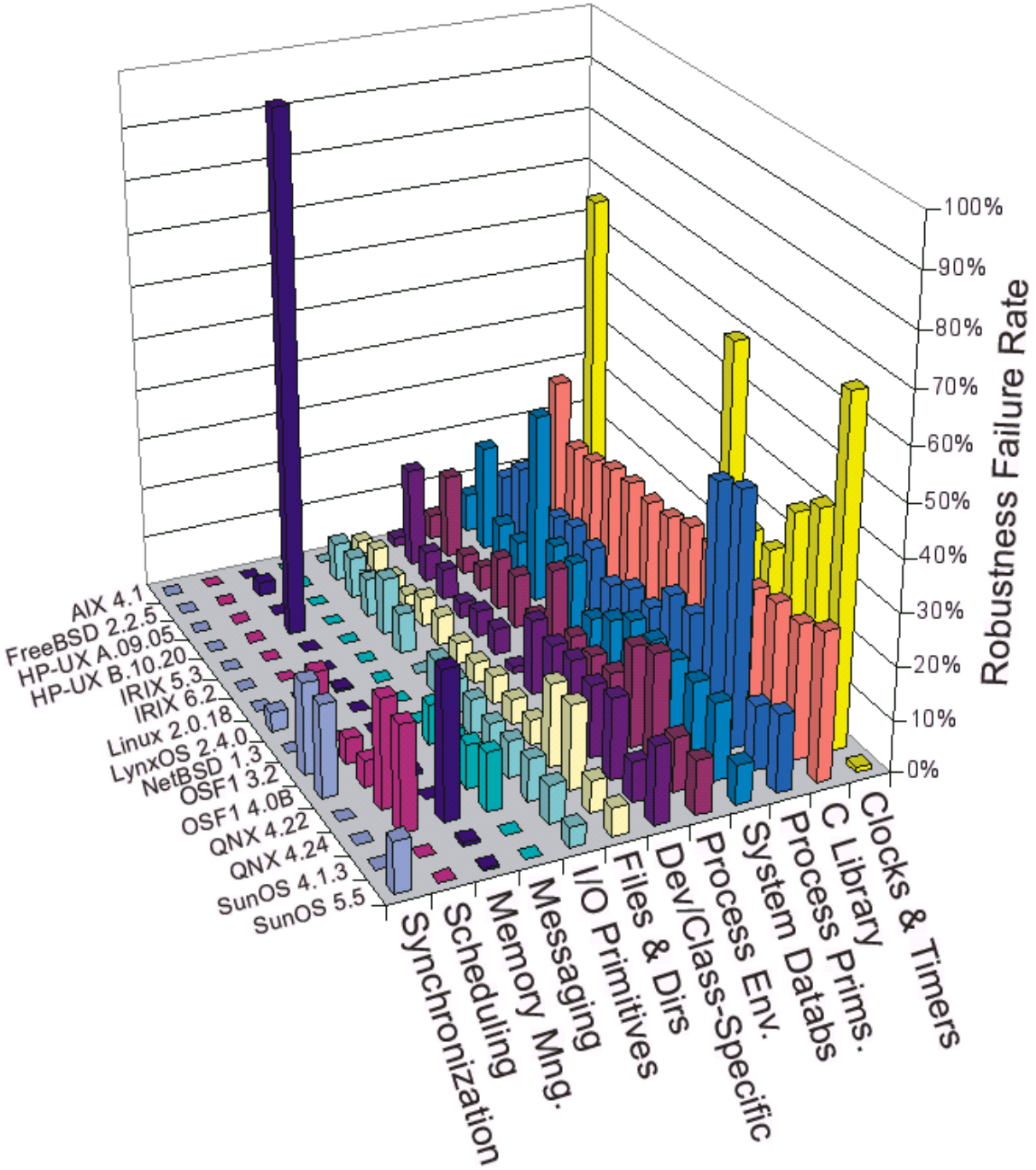
Digital Unix 4.0 Results



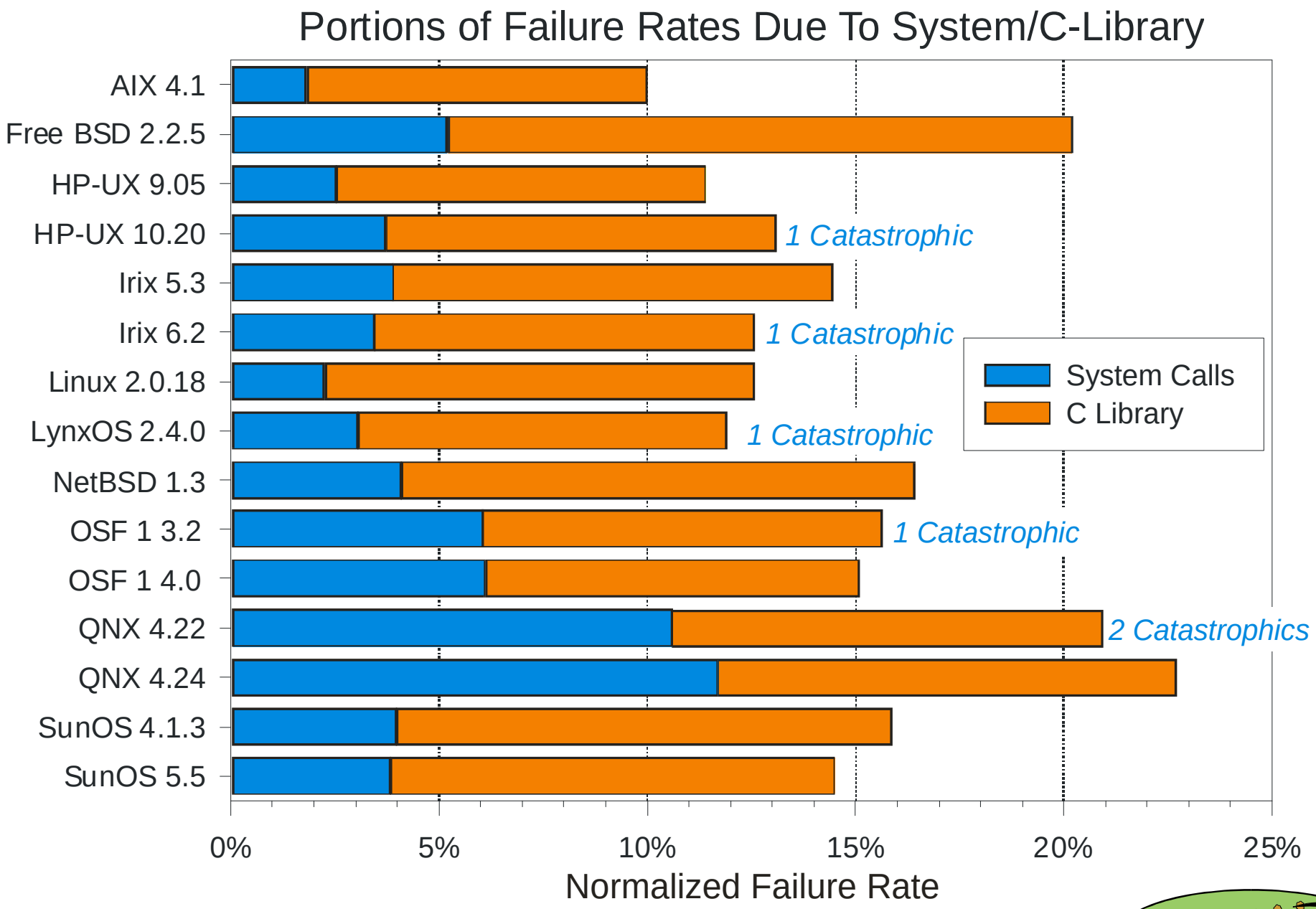
Comparing Fifteen POSIX Operating Systems



Failure Rates By POSIX Fn/Call Category



C Library Is A Potential Robustness Bottleneck



Common Failure Sources

- **Based on correlation of failures to data values, not traced to causality in code**

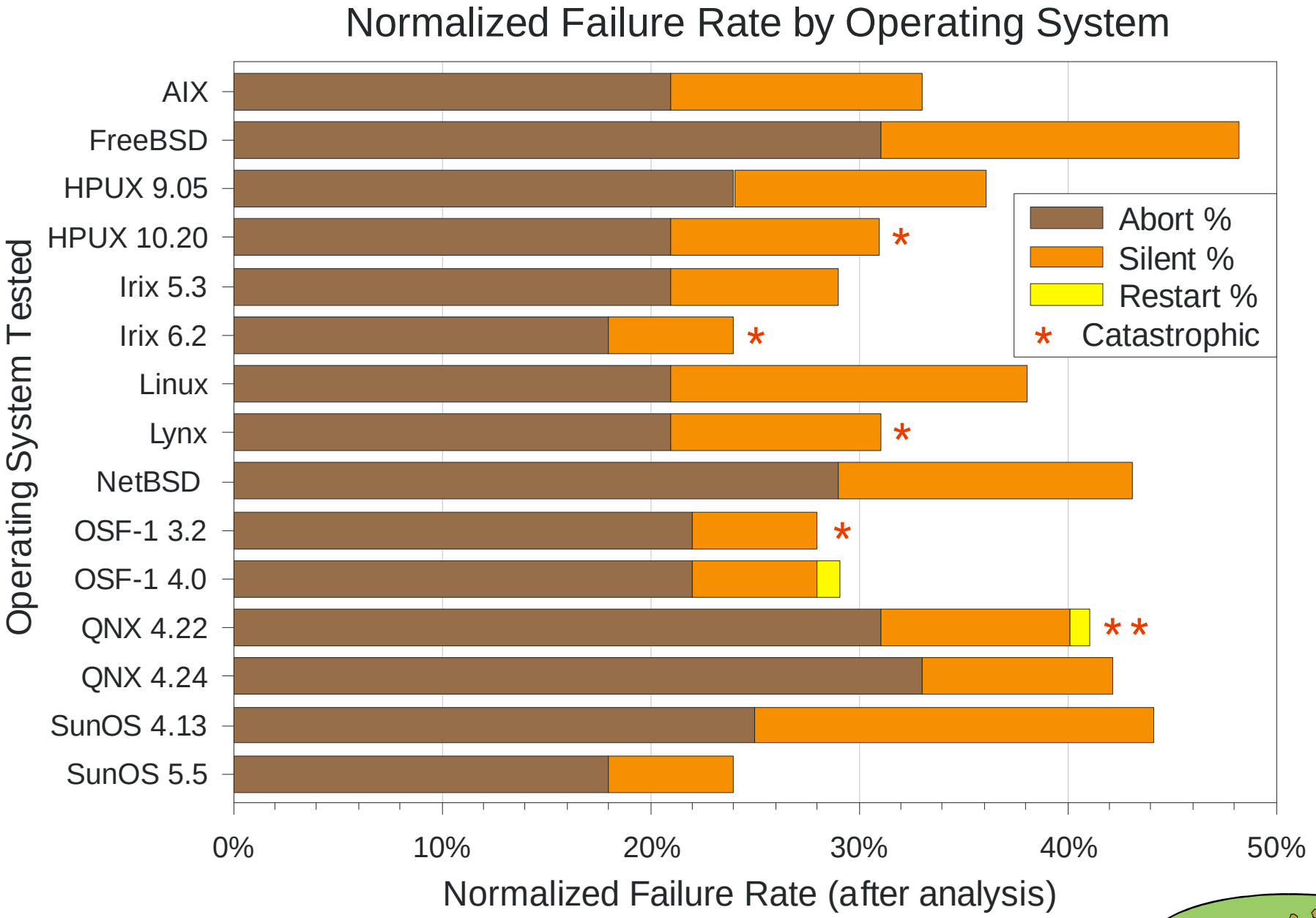
- **Associated with a robustness failure were:**
  - 94.0% of invalid file pointers (excluding NULL)
  - 82.5% of NULL file pointers
  - 49.8% of invalid buffer pointers (excluding NULL)
  - 46.0% of NULL buffer pointers
  - 44.3% of MININT integer values
  - 36.3% of MAXINT integer values

Data Analysis Using N-Version Comparisons

- **Use N-version software voting to refine data (and use manual sampling to check effectiveness)**
  - Eliminate non-exceptional tests -- 12% of data; method ~100% accurate
    - *e.g.*, reading from read-only file
  - Identify Silent failures

- **Silent failures -- 6% to 17% additional robustness failure rate**
  - 80% accurate when one OS reports "OK" while at least one other OS reports an error code
    - ~2% were bugs involving failure to write past end of file
    - 28% of remainder due when POSIX permits either case
    - 30% of remainder due to false alarm error codes (many in QNX)
    - ~40% of remainder just out of scope of POSIX standard
  - 50% accurate when one OS reports "OK" but another OS dumps core
    - Half of remainder due to order in which parameters are checked
    - Half of remainder due to FreeBSD floating point library Abort failures (*e.g.*, fabs(DBL_MAX))

Estimated N-Version Comparison Results



Is Dumping Core The "Right Thing?"

- **AIX has only 10% raw Abort failure rate -- on purpose**
  - Wish to avoid Abort failures in production code
  - Ignores some NULL pointer reads by setting page 0 to read permission
  - *BUT -- 21% adjusted Abort failure rate*; 12% Silent failure rate

- **FreeBSD has 20% raw Abort failure rate -- on purpose**
  - Intentionally aborts to flag bugs during development cycle
  - 31% adjusted Abort failure rate; *BUT -- 17% adjusted Silent failure rate*

- **Future challenges:**
  - Flag defects during development
    - Boundschecker-like systems need a workload to find problems
  - And still tolerate robustness problems once system is fielded
    - Truly Portable exception handling for POSIX API
    - Perhaps wrappers to manage complexity of exception handling (*e.g.*, Bell Labs XEPT work)

But What About Windows?

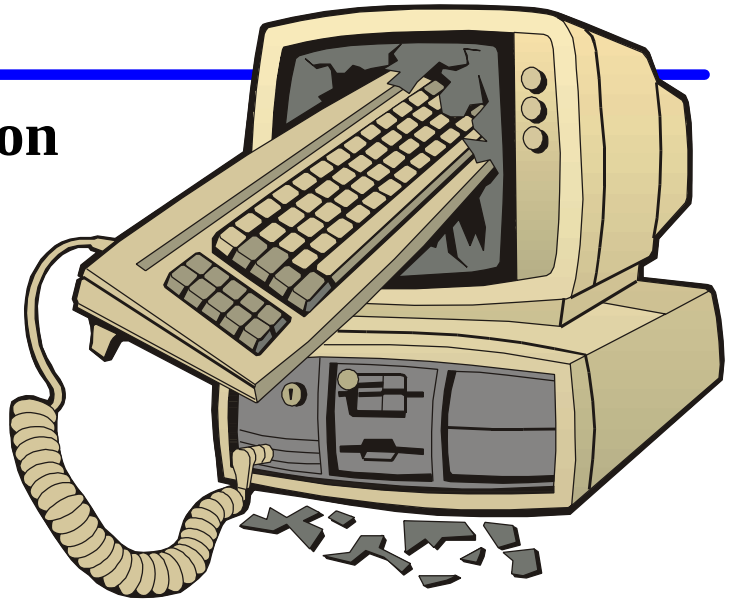
Widows Systems Tested

- **One major new datatype needed: HANDLE**
  - Also ported testing client to Win32 API

- **Desktop Windows versions on Pentium PC**
  - Windows 95 revision B
  - Windows 98 with Service Pack 1 installed
  - Windows 98 Second Edition (SE) with Service Pack 1 installed
  - Windows NT 4.0 with Service Pack 5 installed
  - Windows 2000 Beta 3 Pre-release (Build 2031)
  - 143 Win32 API calls + 94 C library functions tested

- **Windows CE**
  - Windows CE 2.11 running on a Hewlett Packard Jornada 820 Handheld PC
  - 69 Win32 API calls + 82 C library functions tested

Windows Robustness Testing



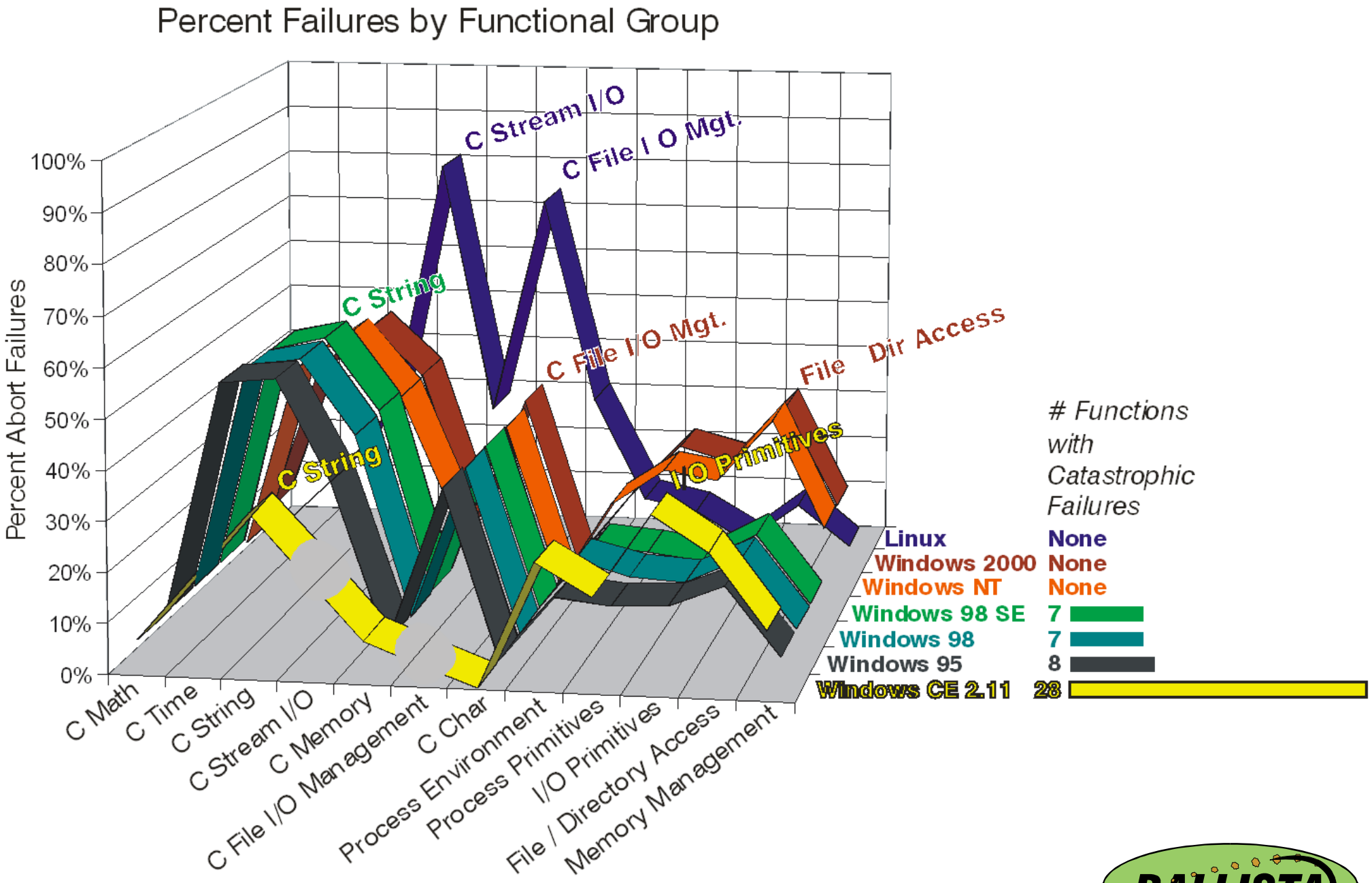
- **Several calls cause complete system crashes on Win 95/98/98 SE and Win CE**
  - Windows 95: 8 calls
  - Windows 98: 7 calls
  - Windows 98 SE: 7 calls
  - Windows CE: 28 calls
  - Windows 98 and Windows CE example:
    ```
    GetThreadContext(GetCurrentThread(), NULL);
    ```

- **Windows results compared to Linux**
  - Test groups of comparable calls by functionality
  - No tests caused system crashes on Windows NT/2000 nor on Linux
    - They're not "crashproof" – they're just not quite so easy to crash

- **Linux and Windows NT/2000 both "generally robust"**
  - Linux has lower Abort rate on system calls; higher on `glibc`
  - Have to dig deeper for the complete story, of course

BALLISTA

Failure Rates by Function Group



Technology Transfer

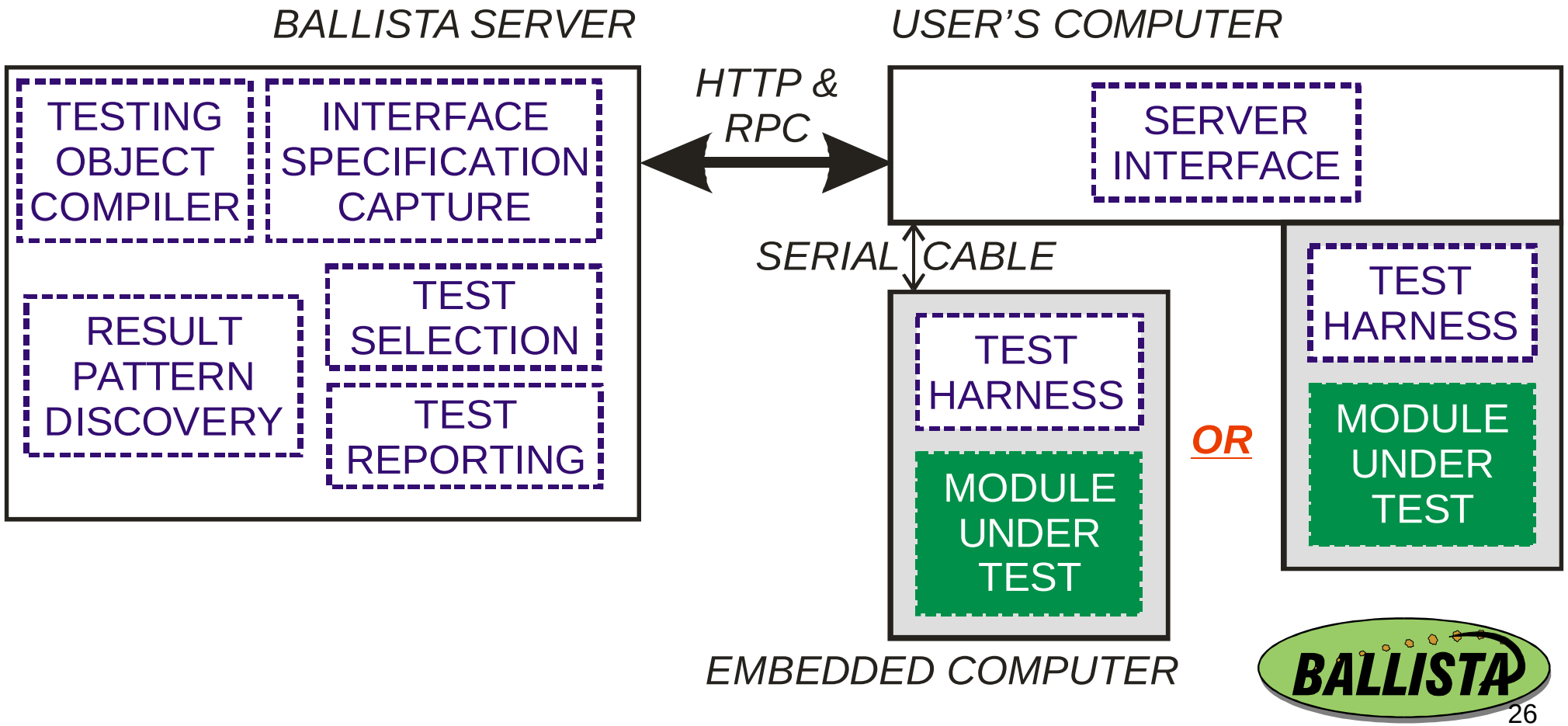
- **Original project sponsor was DARPA**
  - Sponsored technology transfer projects for:
    - Trident Submarine navigation system (U.S. Navy)
    - Defense Modeling & Simulation Office HLA system

- **Industrial sponsors are continuing the work**
  - Cisco – Network switching infrastructure
  - ABB – Industrial automation framework
  - Emerson – Windows CE testing
  - AT&T – CORBA testing
  - ADtranz – (defining project)
  - Microsoft – Windows 2000 testing

- **Other users include**
  - Rockwell, Motorola
  - And, potentially, some POSIX OS developers
  - HP-UX

Public Robustness Testing Service

- **Ballista Server**
  - Selects tests
  - Performs pattern Analysis
  - Generates “bug reports”
  - Never sees user’s code
- **Ballista Client**
  - Links to user’s SW under test
  - Can “teach” new data types to server (definition language)



Specifying A Test (web/demo interface)

- **Simple demo interface; real interface has a few more steps...**

As an example, test the `fopen()` function with:

```
fopen ( fname, str, --None--, --None--, --None-- )
```

fopen (fname , ints , --None-- , --None-- , --None-- , --None--)

Submit Reset

aiocb
buf
dir
fd
float
fmode
fname
fp
int
intp
ints
mode
msgq
oflags
pid
sem
sigset
size
str
timeout

When you click on **Submit** there will b hile the server performs the requested test; then you will see a page containing the test cases that cor robustness faults within Digital Unix 4.0. You can read the notes section to learn a bit more about ng on. After you have tried `strcpy()` you can try several more examples, or just pick your favor call.

Notes:

What's going on with this demo?

This is a second-generation operating s suite (you can read about the first-generation test suite in a conference paper preprint). It takes the me and parameter data types that you enter and composes a set of operating system robustness tests on on our Alphastation web server.

Viewing Results

- **Each robustness failure is one test case (one set of parameters)**

Test Results

Back to OS Test Page
Ballista Home Page

`fopen(fname,str)`

Results for Alpha OSF 4.0 : Out of 100 tests run, 68 passed and **32 failed**.

A list of failures follows. Click on a line to view source code that should reproduce the failure.

A `result` of 'Abort' indicates that the function being tested generated an exception. `Return value` is the value returned by the system call. `Parameters` are the specific parameter values generated by Ballista for that test case. Complete results for both pass and failure cases are also available.

Result	Return value	Parameters	
Abort	-1	FNAME_NOEXIST	STR_RAND
Abort	-1	FNAME_NOEXIST	STR_NEG
Abort	-1	FNAME_EMBED_SPC	STR_RAND
Abort	-1	FNAME_EMBED_SPC	STR_NEG
Abort	-1	FNAME_LONG	STR_RAND
Abort	-1	FNAME_LONG	STR_NEG
Abort	-1	FNAME_CLOSED	STR_RAND
Abort	-1	FNAME_CLOSED	STR_NEG
Abort	-1	FNAME_OPEN_RD	STR_RAND
Abort	-1	FNAME_OPEN_RD	STR_NEG
Abort	-1	FNAME_OPEN_WR	STR_RAND

“Bug Report” program creation

- Reproduces failure in isolation (>99% effective in practice)

```
/* Ballista single test case Sun Jun 13 14:11:06 1999
 * fopen(FNAME_NEG, STR_EMPTY) */
...
  const char *str_empty = "";
...
  param0 = (char *)  -1;

  str_ptr = (char *) malloc (strlen (str_empty) + 1);
  strcpy (str_ptr, str_empty);
  param1 = str_ptr;
...
  fopen (param0, param1);
```

Application to Non-OS system HLA-RTI

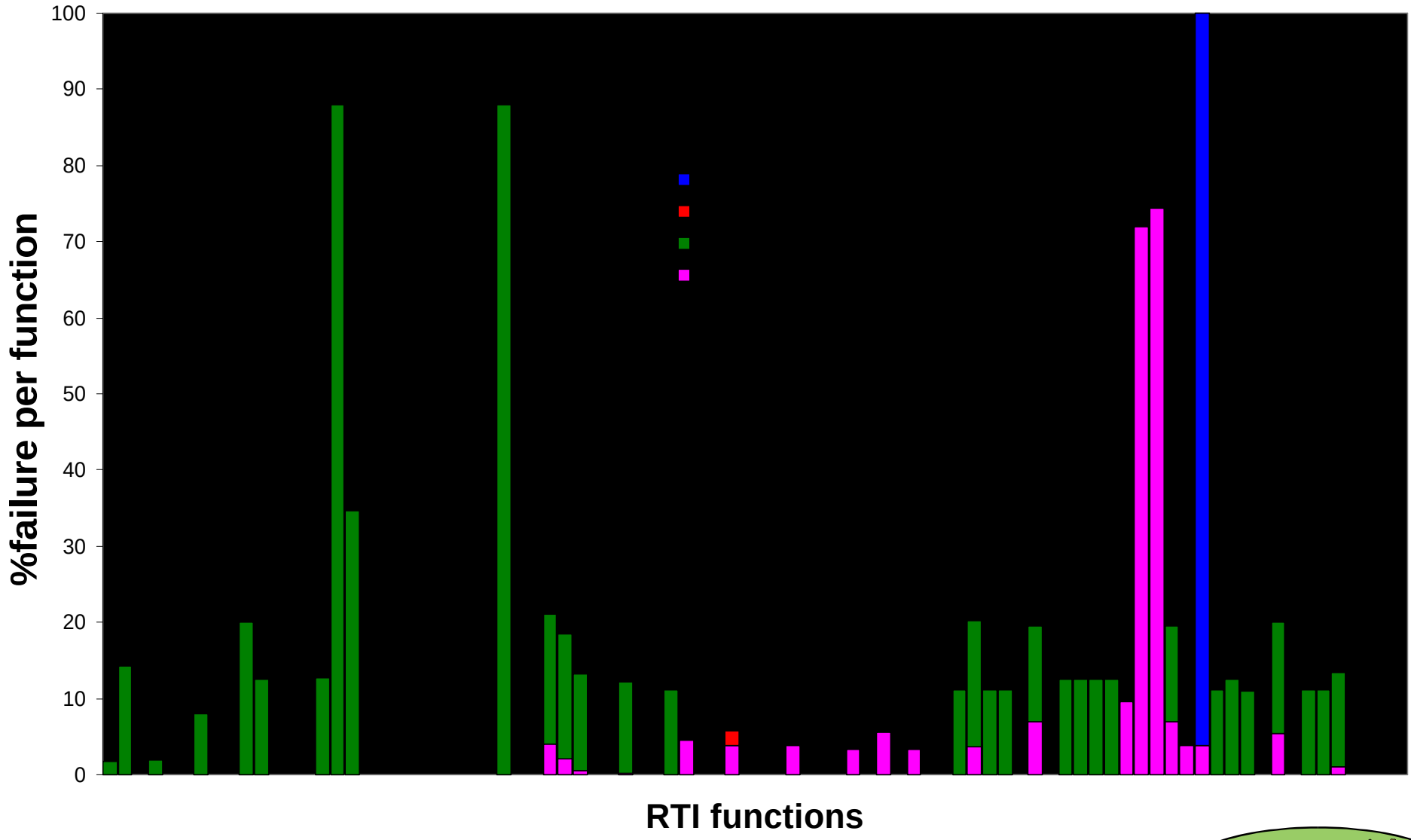
- **DOD HLA-RTI - High Level Architecture Run Time Infrastructure**
  - DOD simulation framework to support massive distributed simulations of large scale combat scenarios

- **Specifically designed for robust exception handling**

- **Their goal is that every exception condition should be handled, with sufficient information returned to the process such that the exception can be handled in a graceful manner**
  - No generic exceptions
  - No default OS actions (*i.e.* abnormal process termination via a signal)

RTI-HLA Digital Unix 10.2 % Failure Rate

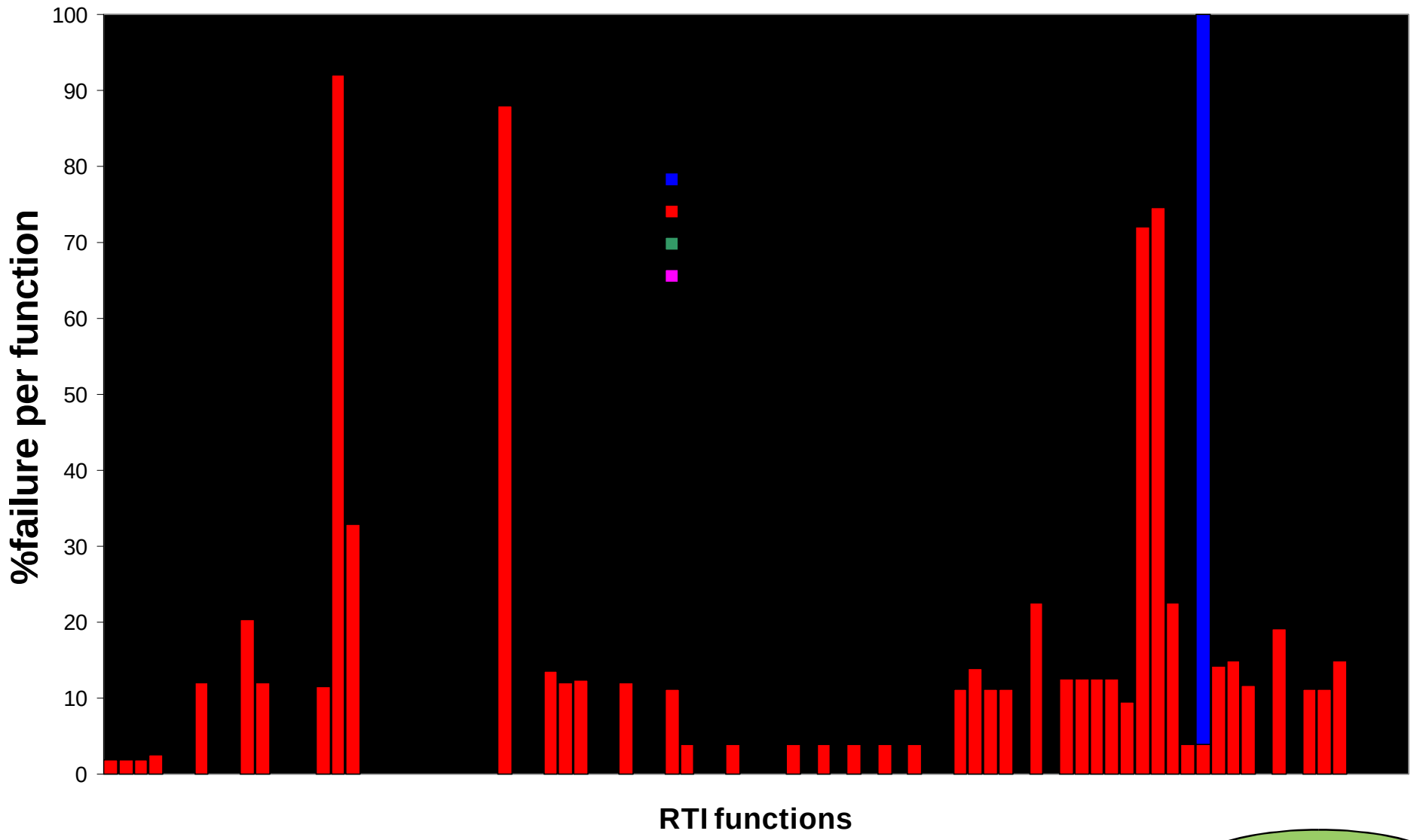
Robustness Failures of RTI 1.3.5 for Digital Unix 4.0



BALLISTA

RTI-HLA Solaris 10.0 % Failure Rate

Robustness Failures of RTI 1.3.5 for Sun OS 5.6

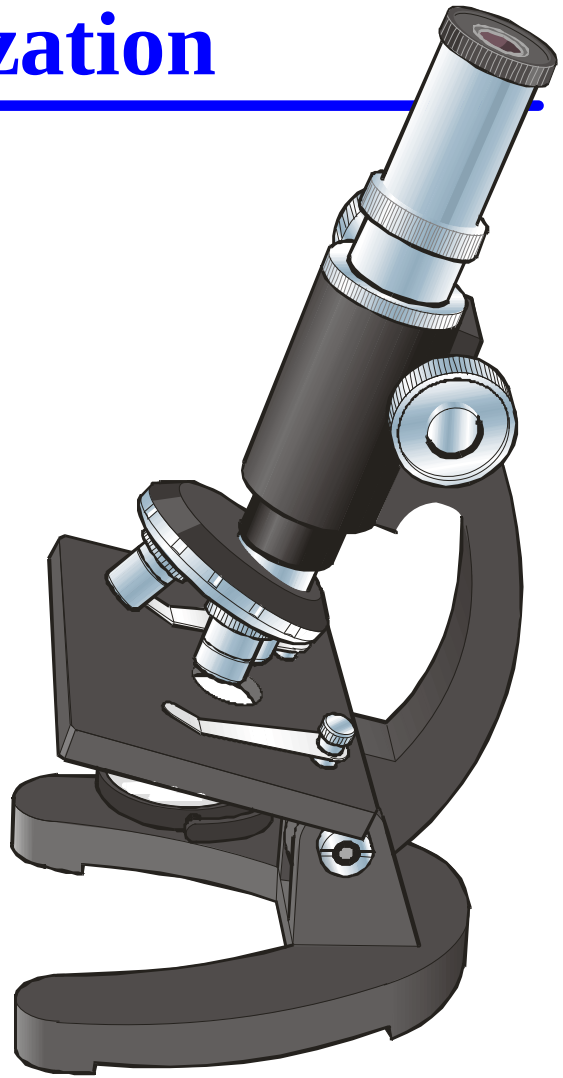


BALLISTA

Refinement: Fine-Grain Characterization

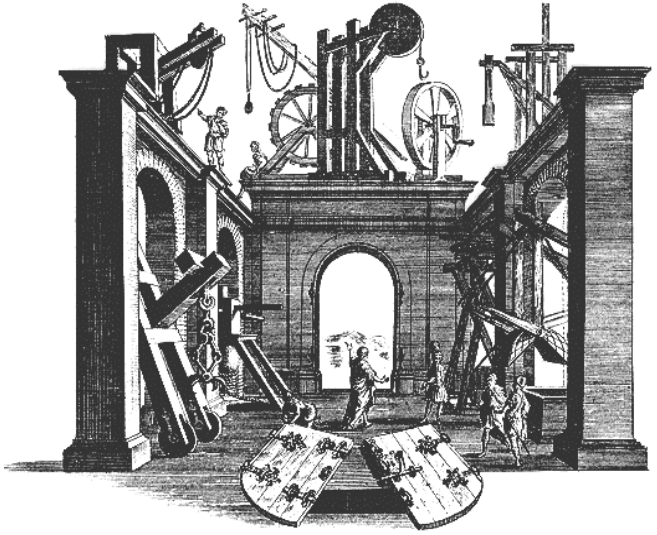
- **Problem: detailed coverage of rich data types (*e.g.*, file handle)**
  - Want tests with high degree of flexibility
  - Want useful notion of "adjacency" in test results

- **Solution: Logical Structs**
  - Decompose data type into *logical* struct of orthogonal sub-types
  - Example for file handle:
    1) File exists, does not exist, deleted after creation
    2) Open for: read, write, r/w, closed
    3) File system permissions for: read, write, r/w, none
    4) File positioned at: beginning, middle, end, past end
    5) ...



BALLISTA

Refinement: User Defined "Scaffolding"



- **Needed for state-intensive systems**
  - DMSO HLA simulation backplane
  - ABB framework software
- **Implemented via prologue/epilogue code per-function**
  - 10 "equivalence classes" for 86 functions in HLA
  - Needed for most functions in ABB framework
- **It works, but building scaffolding is time consuming**
  - Ballista only scales well when scaffolding can be shared among many functions
  - Doesn't look promising for database API testing

BALLISTA

Refinement: Set State Via "Phantom" Params

- **Don't really need separate scaffolding mechanism…**
  - Use "phantom" parameters:

    `func_name(+setup, param0, param1, …)`

    where "+" means execute constructor/destructor, but don't pass to function

- **Can also be used to accomplish some system-level state setting**
  - Test random number generator:

    `random(+random_seed, range)`

  - Test disk write with full/empty disk:

    `write(+set_disk_state, filedes, buffer, nbytes)`

Wrap-Up: What Ballista Does (and Doesn't Do)

- **Quantification of single-threaded exception handling robustness**
  - Scalable, inexpensive compared to traditional testing approaches
  - Really does find novel ways to crash commercial-grade software (in the future, will include heavy-load testing)

- **Evolving toward fine-grained testing**
  - Permits orthogonal attribute decomposition of data types
  - Will form the bases for future "smart" adaptive testing strategies

- **It's easy to test some system state**
  - Small amounts of system state in parameter-based tests
    - Larger system state possible using phantom parameters
  - But, large amounts of state are a problem on database-like systems

- **Testing turned out to be more successful than we thought**
  - And hardening is turning out to be very difficult

What Comes Next?

- **Do-it-yourself POSIX test kit**
  - Available for use with testing server this semester
- **"Bulletproof Linux"**
  - Reduce failure rates in glib
  - Survey of other issues in "bulletproofness"
- **Software aging/state-intensive testing**
  - Concurrency, resource exhaustion
- **Smart, pattern-based testing**
  - How do we cover huge search spaces?

Long Term Prospects

- **Technology Transfer**
  - Already taking place
- **Commercial product support**
  - Still trying to figure this one out

BALLISTA

Contributors

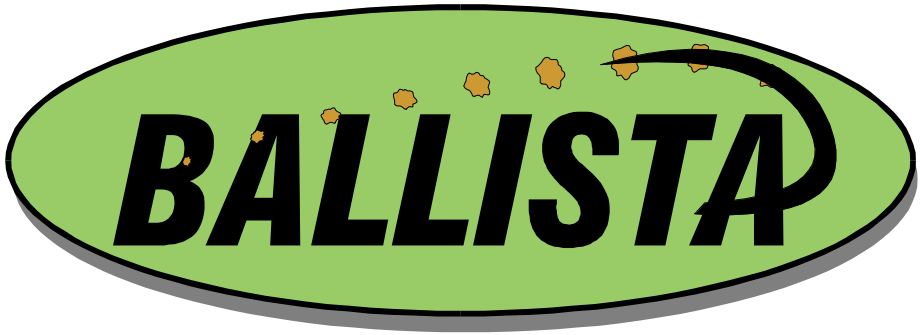
- **What does it take to do this sort of research?**
  - A legacy of 15 years of previous Carnegie Mellon work to build upon
    - But, sometimes it takes that long just to understand the real problems!
  - Ballista: 3.5 years and about $1.6 Million spent to date

Students:

- **Meredith Beveridge**
- **John Devale**
- **Kim Fernsler**
- **David Guttendorf**
- **Geoff Hendrey**
- **Nathan Kropp**
- **Jiantao Pan**
- **Charles Shelton**
- **Ying Shi**
- **Asad Zaidi**

Faculty & Staff:

- **Kobey DeVale**
- **Phil Koopman**
- **Roy Maxion**
- **Dan Siewiorek**



http://www.ices.cmu.edu/ballista