



A Dimensionality Model Approach to Testing and Improving Software Robustness

Jiantao Pan
ECE Department
Carnegie Mellon University
jpan@cmu.edu



Institute
for Complex
Engineered
Systems

Carnegie Mellon



Electrical & Computer
ENGINEERING



Software Robustness

◆ Definition of Software Robustness

- *The robustness of a software component is the degree to which it can function correctly in the presence of exceptional inputs or stressful environmental conditions. (IEEE standard)*

◆ Software Robustness failure

- Occurs if the program is not able to function correctly when presented with exceptional or unspecified conditions
- Current work does not address stressful environmental conditions

Software Robustness Failures — Desktop

- ◆ **What should your word processor do when you try to open a corrupt file?**
 - a. Reboot the machine
 - b. Lock keyboard, refusing any future input
 - c. Exit to operating system
 - d. Use corrupt data as if it were valid
 - e. Prompt you with an error message, saying registration has expired
-
- f. Prompt you with an error message, saying file is corrupt

Robustness Failures — Critical Application

- ◆ **Stakes are higher in critical applications than in a word processor**

- Aerospace flight control
 - Ariane 5 rocket explosion
- Radiation therapy machine
 - Therac 25 overdose accident
- Nuclear power plant control system
- Airplanes
- Automotive
- Weapons, warheads
- Chemical processing plant
- Telephone systems
- Banks
-



Ballista Overview

◆ **The Ballista project**

- Testing user software modules for robustness failures
 - API level testing and hardening
 - No source code needed
 - Repeatable results
 - Generate standalone C code to reproduce the failure

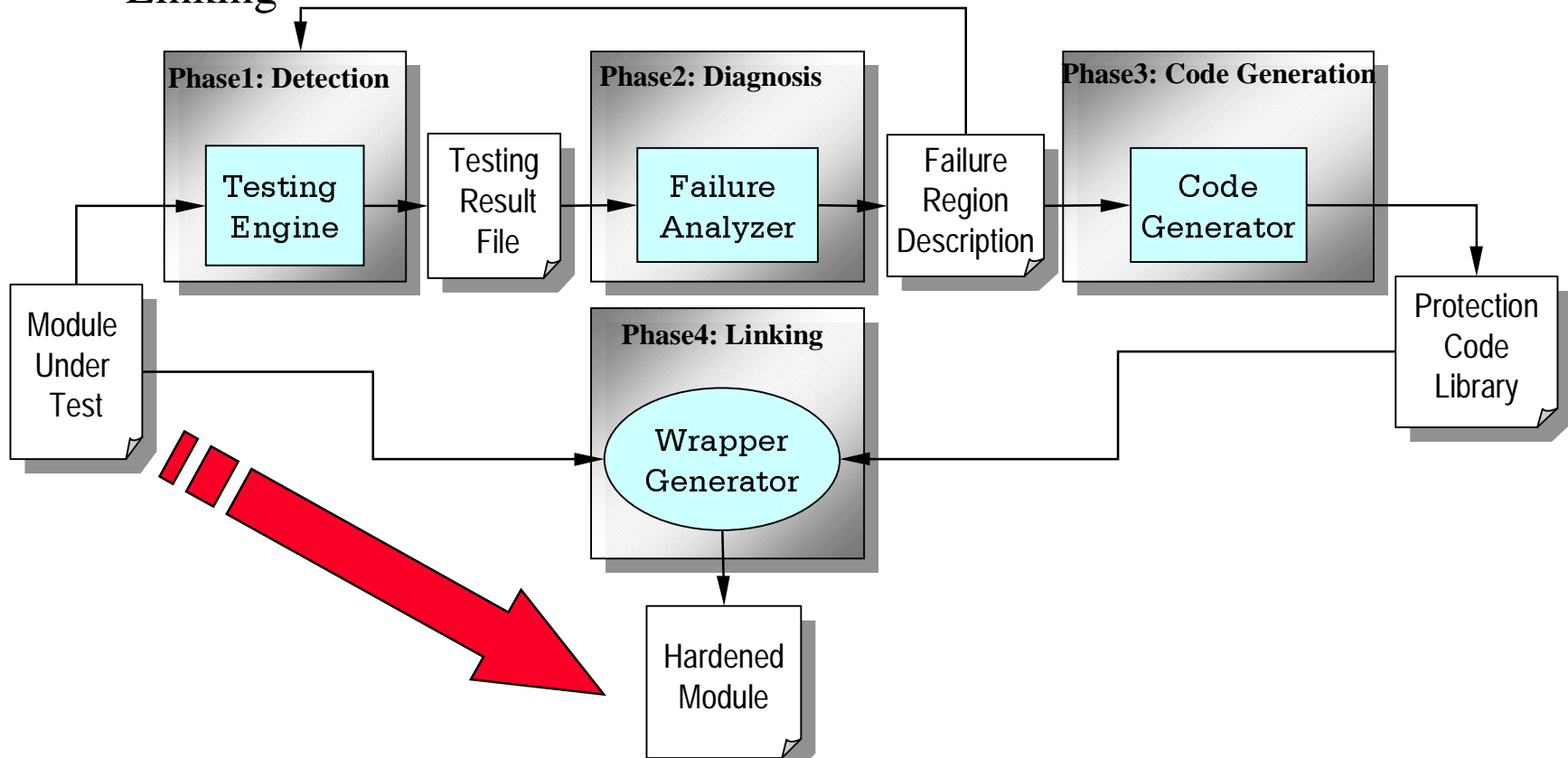
◆ **Ballista public www testing server**

- Publicly accessible robustness testing service
 - Giving away accounts for free now
- Web interface and command line interface available
- Client-server architecture for portability
- Now support Digital Unix, HPUX, Solaris, Linux RedHat 5.0
- User can design their own test cases or use preexisting ones

Automatic Robustness Hardening - A Strategy

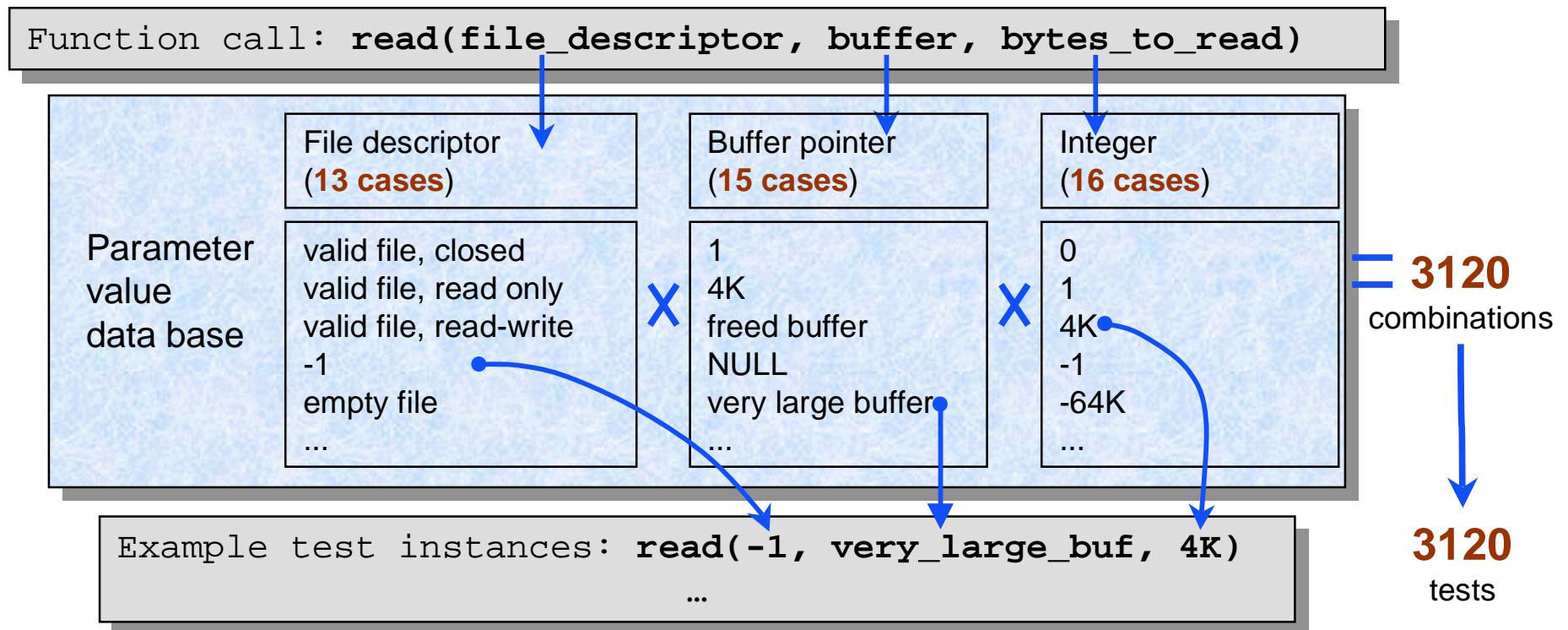
◆ 4 phases all need to be automated

- Robustness testing
- Failure region diagnosis
 - **The Dimensionality Model**: for finding failure patterns
- Protection code generation
- Linking



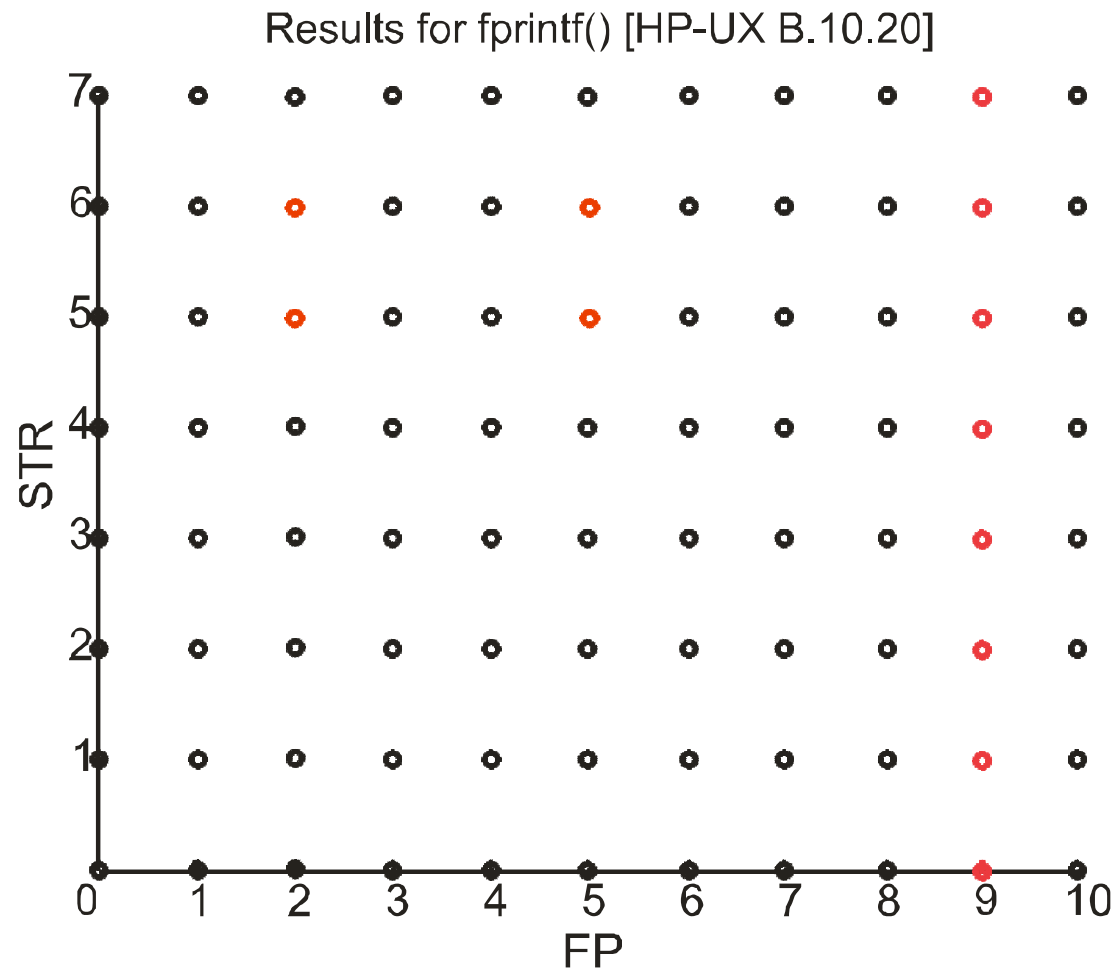
Phase1: Automated Robustness Testing

- ◆ **Combines testing and fault injection ideas**
 - **C-R-A-S-H** scale metric to classify robustness failures
- ◆ **Combinations of valid and invalid inputs**
 - Single call per test for simplicity, ignore interactions and timing



Phase 1: Testing Result Pattern

- ◆ `fprintf(File_Pointer, STRing)` in a POSIX compliant OS



- Pass (Success, Error code)
- Robustness Failure (Catastrophic, Restart, Abort)

Phase 2: The Dimensionality Model

◆ Definitions:

- Parameter dimensionality: number of arguments accepted by a software module
- Robustness failure dimensionality: number of parameters contributing to the failure

◆ Examples

• Function with parameter dimensionality of 3:

– `read(file_descriptor, buffer, bytes_to_read)`

• 1-D failure: `read(NULL, —, —)`

– NULL `file_descriptor`

• 2-D failure: `read(—, 16K, 64K)`

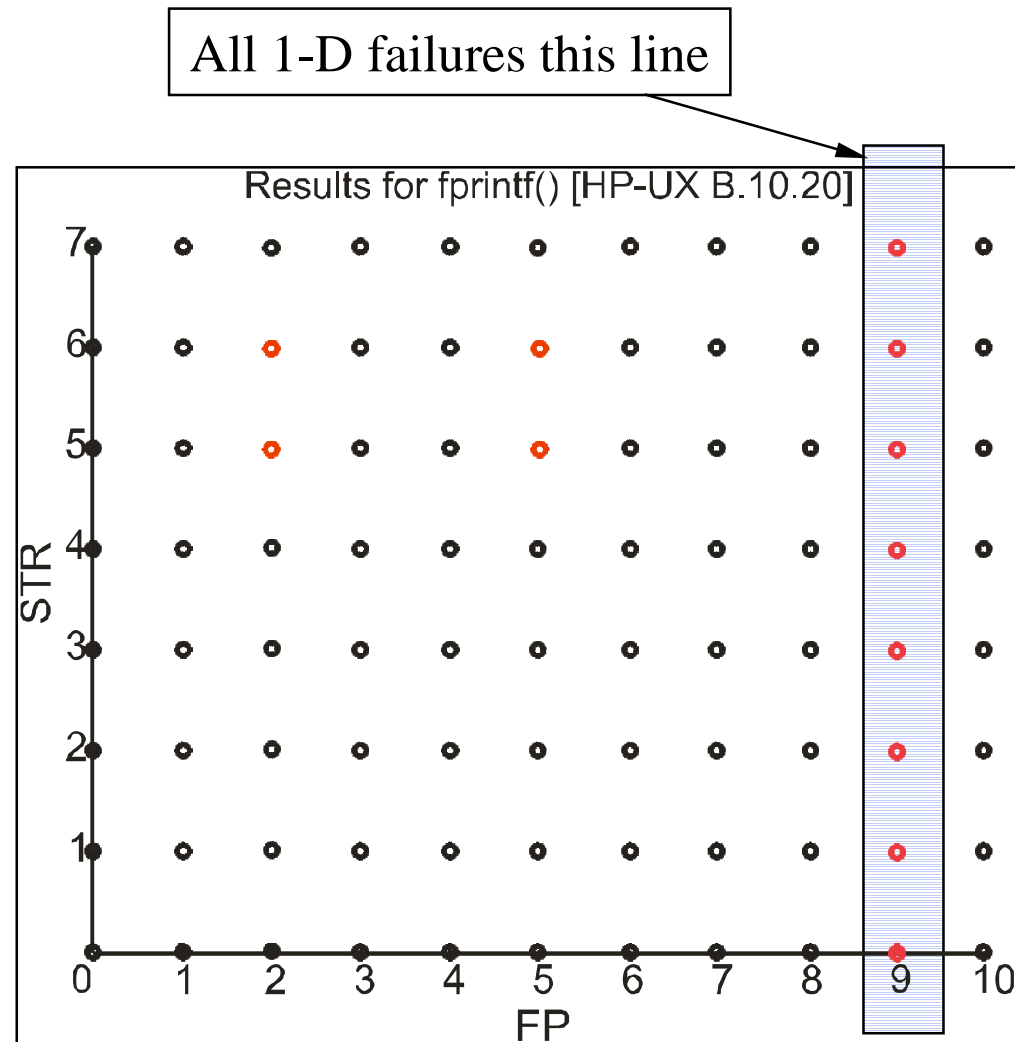
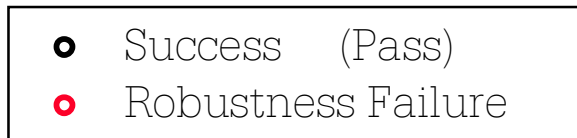
– `buffer` smaller than `bytes_to_read`

◆ 1-D failures: ~80% of all failures found

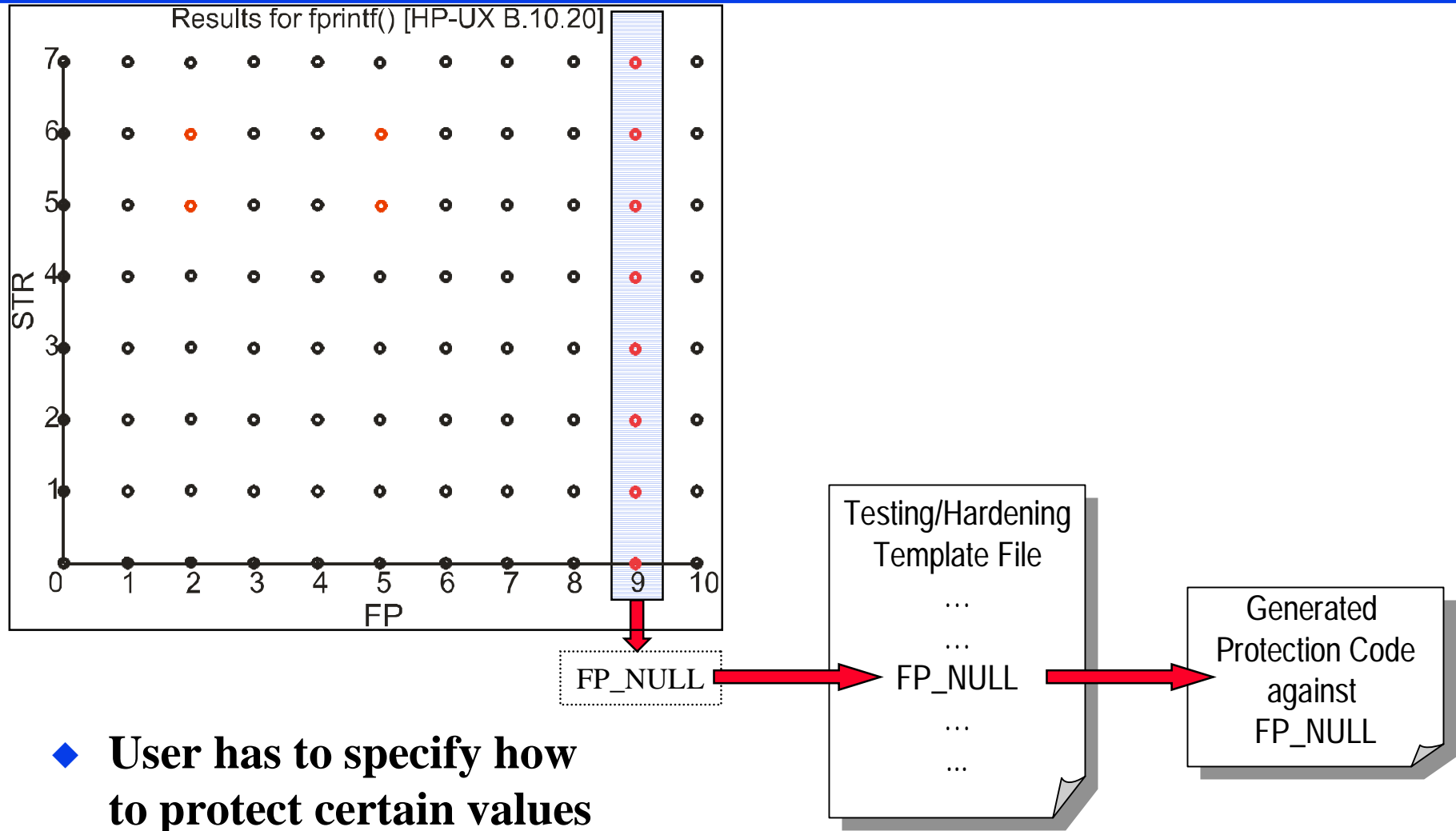
- From previous Ballista testing result on 15 POSIX OS

Phase 2: 1-D Failure Patterns

- ◆ 1-D failures: form a line in a 2-D function (parameter dimensionality=2)
- ◆ They form a hyperplane in a n-D function



Phase 3: Automated Code Generation



Phase 4: Linking

◆ Purpose:

- Redirect the call to the hardened version

◆ Challenge:

- No source code or no permission to modify source code

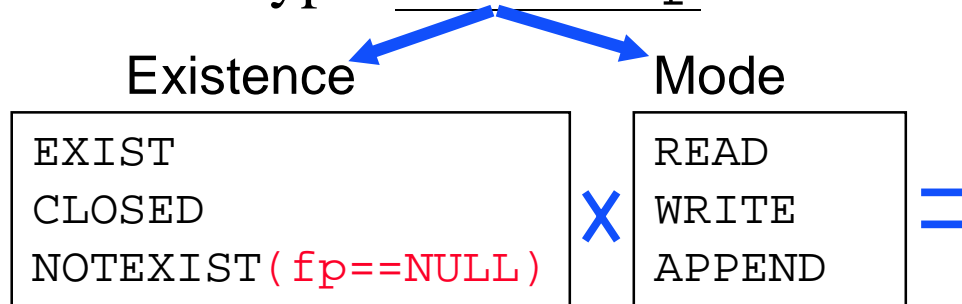
◆ Solution:

- Bell Labs researchers have the answer
 - Xmangler tool in Xept package
- My solution:
 - Borrow the Xept tools

Example

◆ Example POSIX function: `fgetc(FILE* fp)`

- data type: `FILE* fp`



| Result | Return | Parameters |
|--------|--------|---|
| Pass | 9 | <code>fp_EXIST</code> <code>fp_APPEND</code> |
| Pass | 9 | <code>fp_CLOSED</code> <code>fp_READ</code> |
| Abort | -1 | <code>fp_NOTEXIST</code> <code>fp_WRITE</code> |
| Pass | 0 | <code>fp_EXIST</code> <code>fp_READ</code> |
| Pass | 9 | <code>fp_CLOSED</code> <code>fp_APPEND</code> |
| Pass | 9 | <code>fp_CLOSED</code> <code>fp_WRITE</code> |
| Abort | -1 | <code>fp_NOTEXIST</code> <code>fp_APPEND</code> |
| Pass | 9 | <code>fp_EXIST</code> <code>fp_WRITE</code> |
| Abort | -1 | <code>fp_NOTEXIST</code> <code>fp_READ</code> |

- ◆ Three **Aborts** caused by `fp_NOTEXIST`, or `NULL` pointer value of `fp`

Results

- ◆ **Generate protection code against NULL file pointer**
- ◆ **Redirect call to `fgetc()` to the hardened version `h_fgetc()`**
- ◆ **Rerun the hardened program**
 - Robustness failure caused by NULL `fp` has been protected
- ◆ **Test the hardened version `h_fgetc()` using Ballista**
 - No failures found

Conclusions & Future Work

◆ **Conclusions:**

- Automatic protection code generation is feasible for some data types
 - Success for integers, floating point numbers, and NULL pointer values.
- The Dimensionality Model can be used to diagnose the trigger for a failure.
- May help reduce design cost & time while producing a robust system

◆ **Future work**

- See how far we can go for protection code generation
- Work on Windows software testing

Acknowledgements

- ◆ **Special thanks to Ballista sponsor: DARPA**



- ◆ **Thanks to my advisors Phil Koopman, Dan Siewiorek and my colleagues**
- ◆ **Thanks to the organizers of AUTOTESTCON conference**

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