Software Fault Injection - Industry Experience

Lakshmankumar Mukkavilli
Cisco Systems, Inc.

Topics

- Acronyms
- Background
- Goal
- Tool for Software Fault Injection
- Usage – In Regression Testing
- Results
- Usage – In Unit Testing
- Results
- Conclusions

Acronyms

- SA – Static Analysis
- MIF – Middleware for Injection of Faults (an internal tool)
- CUT – Component Under Test
- CFD – Customer Found Defect
- MBT – Model Based Testing
- UT – Unit Testing
- ROI – Return on Investment

Background

- Very large embedded software
- About 10-30% of the code is error handling code
- Typically this code is not touched by tests
- Developed a tool called MIF (Middleware for Injection of Faults)

Background (Continued)

- We have a large repository of functionality tests that are used in regression testing. These tests cover non-error handling code.
- All these tests are executed using our test automation system under different configurations
- All the code has gone through SA. SA detects instances where return values are ignored. Users are forced to write error handling code.

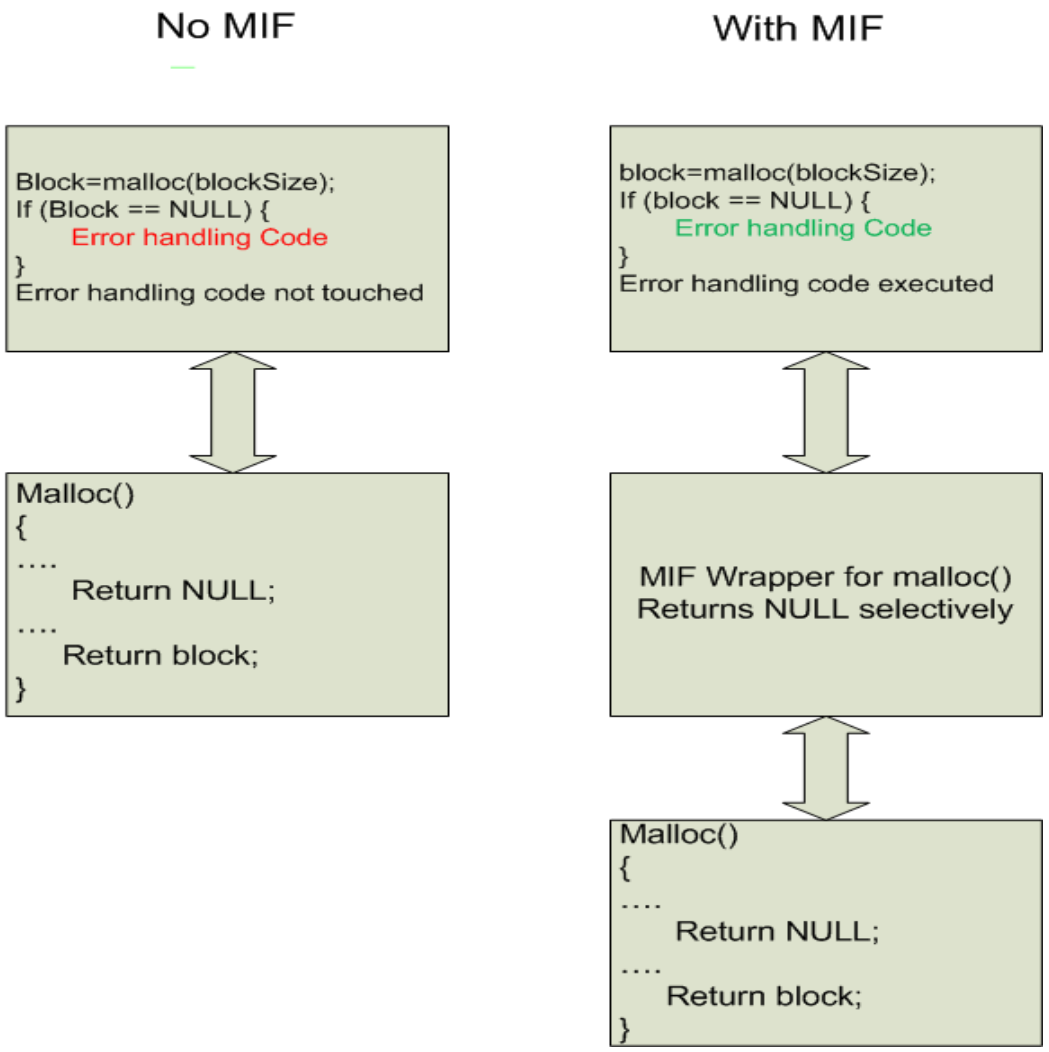
Goal

To ensure that all the error handling code is touched.

Tool for Software Fault Injection

- An internal tool (MIF) is used for software fault injection. Very easy to use.
- It is integrated into our test automation system. Execution, logging and analysis are automated.
- Capable of
  - Interception of function calls and return error conditions. This is the focus in regression testing.
  - Replace functions
  - Various policies for fault injection activity are supported
  - Simulation of exceptions, delays
- Users specify the components and functions of interest. Tool does the rest of work.

MIF


No MIF
Block=malloc(blockSize);
If (Block == NULL) {
Error handling Code
}
Error handling code not touched
Malloc()
{
....
Return NULL;
....
Return block;
}
With MIF
block=malloc(blockSize);
If (block == NULL) {
Error handling Code
}
Error handling code executed
MIF Wrapper for malloc()
Returns NULL selectively
Malloc()
{
....
Return NULL;
....
Return block;
}

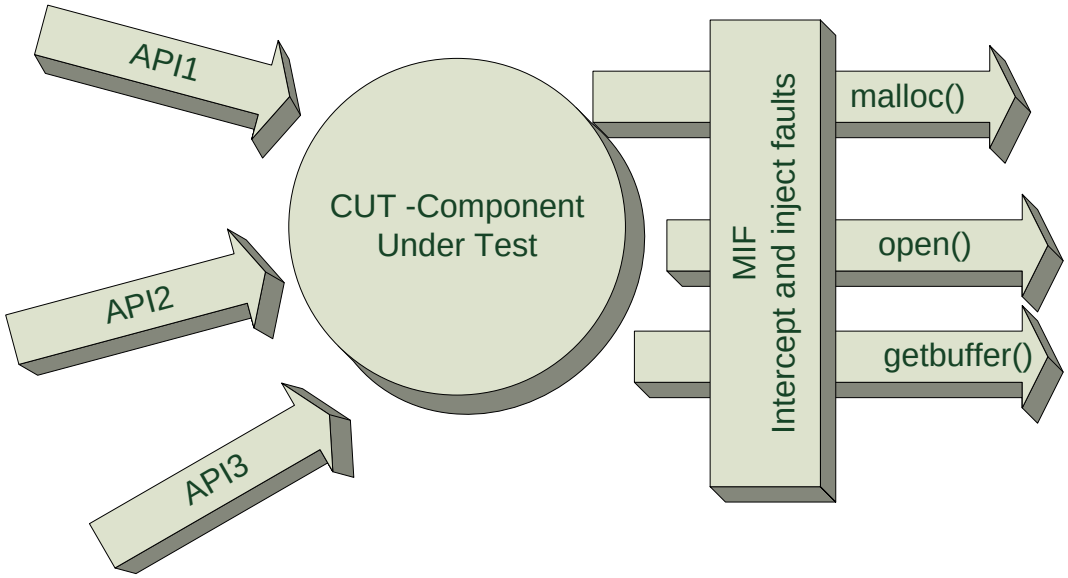
Usage

Two modes of use

- Regression Testing
- Unit Testing

Testing using MIF

Faults injected on outgoing interfaces



Incoming Interfaces
Tests invoke these calls

Outgoing Interfaces
Intercepted and faults injected by MIF

Fault Injection in Regression Testing

- Faults are injected on all the calls to the standard functions.
- Existing tests are used
- Mainly look for crashes/hangs

Results

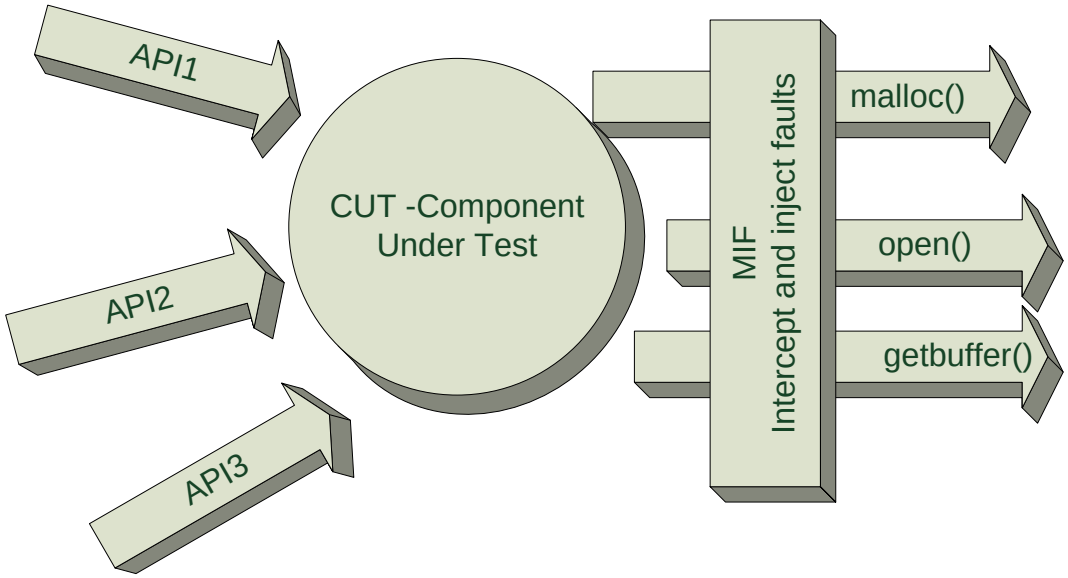
- For first three years - 4% of the regression resources were used. They found 25% of the defects.
- Many of defects were related to CFD's
- Almost in all instances the error handling code was being executed for the first time in our testing.
- The tool provides good information to facilitate reproduction of the defect. The problems were quickly resolved.

Fault Injection in Unit Testing

- Faults are injected on the services used by the unit.
- Sometimes faults are injected on intra-unit calls
- MIF is integrated into our unit testing tool.
- Very popular with development engineers.
- Incoming interfaces are called in three ways
  - Fully automated - Function calls are generated by the API robustness test generator. This is the asic robustness testing of a component
  - Semi-automatic – API dependencies are modeled using our lightweight MBT. API's are invoked based on the test sequences generated from the models.
  - Manual – Users create the tests where various functions/API's are called.
- See the companion paper on "Introduction of Developer Testing in an Embedded Environment"

Robustness Testing using MIF

Faults injected on outgoing interfaces



Incoming Interfaces
API test generator invokes these calls

Outgoing Interfaces
Intercepted and faults injected by MIF

Results

- Please see the companion paper on "Introduction of Developer Testing in an Embedded Environment"
- Defects found using MIF constitute a significant % of the defects found in UT.
- Development engineers are very creative and varied in using the tool.
- Fewer development escapes.

Conclusions

- A good tool for fault injection is the key to success
  - Important factors are – Ease of use, Support for automation, support for a variety of fault injection techniques, good examples and training materials
- Software fault injection is a valuable bug finding technique.
- Very good ROI
- Now Software Fault Injection is a mainstream testing technique.