CrossTest, Inc.

# *White Box Test Automation Framework for C/C++*

*plinares@crosstest.com*

*650 248 1480*

# A few facts/ Food for thought

**Agile development** adoption accelerates, however  testing not fully adopted as a central component of the methodology

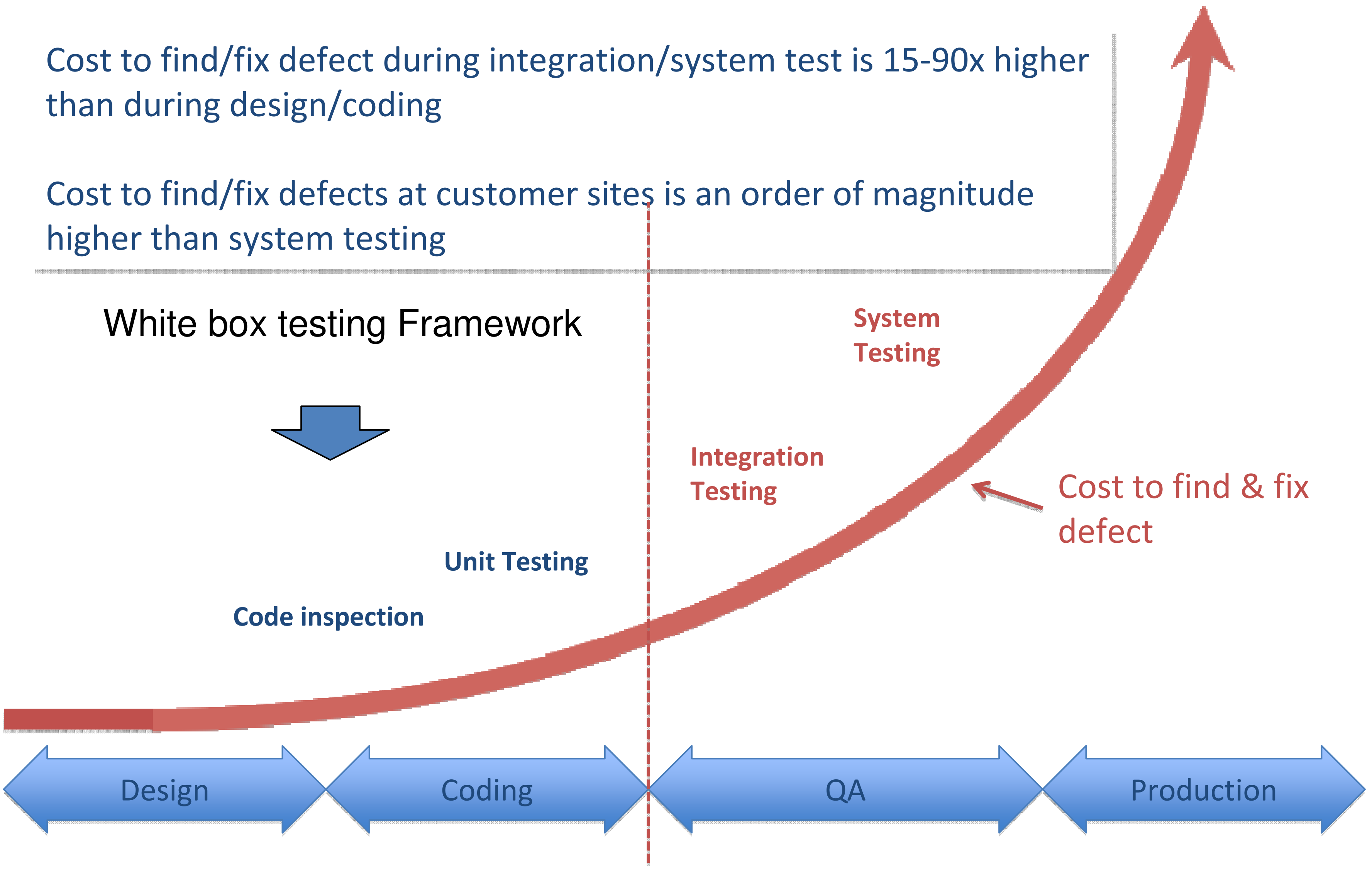
Majority of software engineering (50%+) follow the traditional waterfall model: **tests come last**

Analysis on embedded systems outlined 40% of the bugs found in the field originated from **untested “error” paths**

# Cost of Traditional Testing Approach

Cost to find/fix defect during integration/system test is 15-90x higher than during design/coding

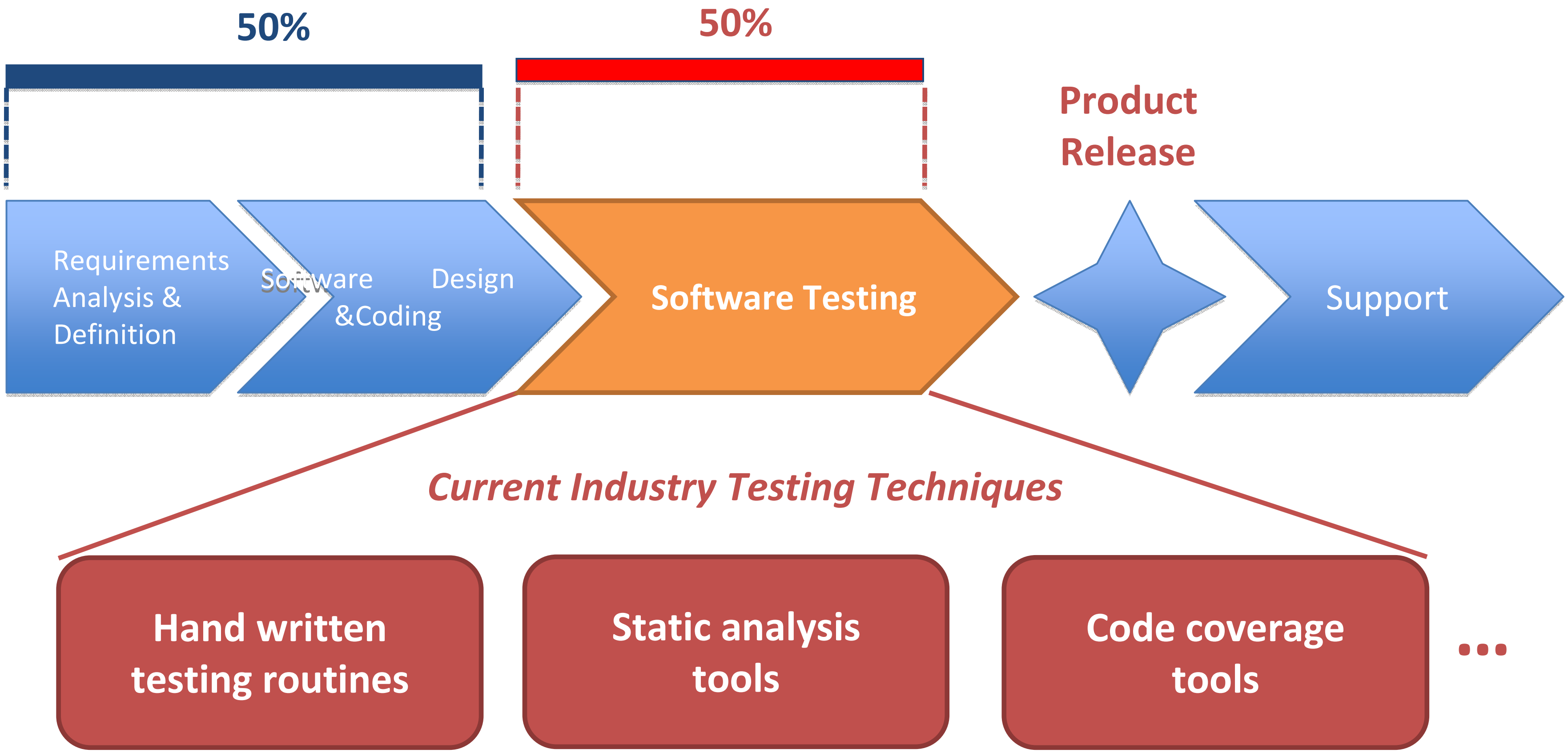
Cost to find/fix defects at customer sites is an order of magnitude higher than system testing



White box testing Framework

*Source: IDC*

# Device Software Development Time & Costs



50%

50%

Product Release

Requirements Analysis & Definition

Software Design &Coding

Software Testing

Support

*Current Industry Testing Techniques*

Hand written testing routines

Static analysis tools

Code coverage tools

…

***"Software testing remains a major bottleneck for the release of new devices"***

# How to change the rules of the game for testing

## Processes

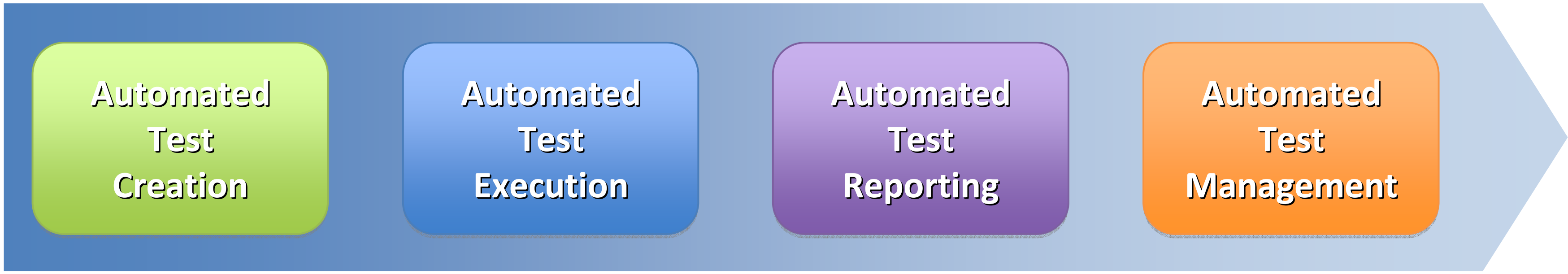
- Introduce testing earlier in the development cycle through white-box testing
- Focus on dynamic testing
- Find the best trade-off between exploratory and regression testing
- Run tests on host, simulator and target devices

## Product

- Make testing possible when code is in an “incomplete” stage
- Easy to use with very fast ramp up
- Adapt to software lifecycle ( automated test updates when new releases are delivered)
- Offer a complete environment for automated testing

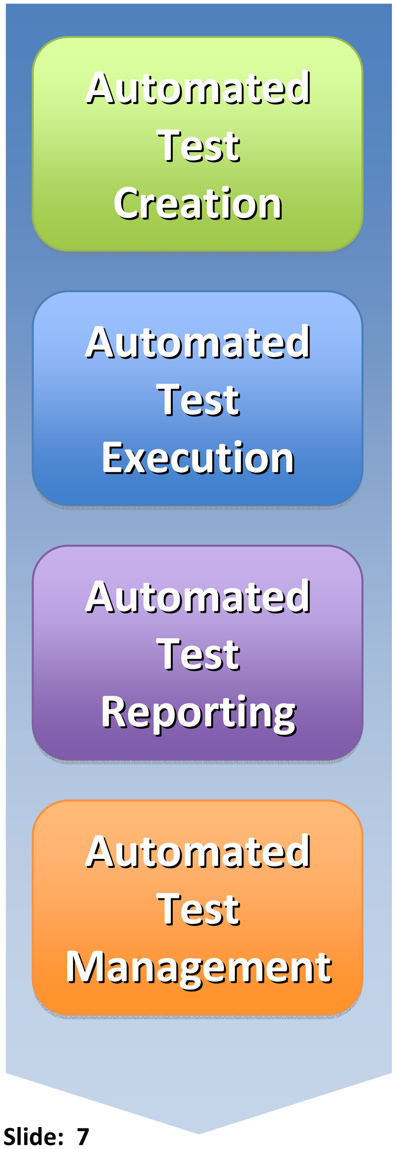
# Fully Integrated Testing Platform

- Key components:



- Unique features:
  - Source code ingestion and analysis
  - Modularity of the “perimeter” under test
  - Permanent synchronization between tests and source code
- Target users:
  - Developers starting very early in the testing of the code
  - Testers for unit, integration and regression testing

# Highly Automated Testing Platform



Automated Test Creation

Automated Test Execution

Automated Test Reporting

Automated Test Management

- Test cases are automatically generated
- No test code needs to be hand-written

- Build, download, and run process is fully automated
- Tests can be run on virtual platform (simulation), then on target device (real deployment)

- Test results are produced and checked
- HTML reports (pass/fail,  code coverage, % at function or line of code level, etc)

- Exported XML test files can be shared between users

- Tests are kept in sync w/ source code changes
- Tests impacted by code modifications are flagged and rerun

# Dynamic Testing

- **The possibilities:**
  - Source code is executed on host system or target device
  - Large coarse or very granular settings for tests run
  - Command line or GUI based
  - Single test or batteries of regression suites
  - Exports/Imports of test suites in XML format for easy re-usability
  - "Snapshot" and "stubbing" as extremely powerful for code coverage and in-depth insight of the code behavior
- **Testing is performing all of the following:**
  - Providing software with inputs
  - Executing a sub-routine or the entire application
  - Monitoring software state and/or outputs for expected properties, such as conformance to requirements, match to output values, lack of system crashes, preservation of invariants

# White Box Testing

## System Level Testing

(Black Box Testing)

- Goal is to test what software does, not how it is implemented
- Emphasis is on meeting system requirements and overall behavior

### Black Box Testing Is Valuable but

### NOT SUFFICIENT:

- Does not cover all portions of software implementation
- Very efficient in repetitive and systematic regression testing
- Usually adopted in a waterfall development cycle

## White box Framework Approach

(White Box Testing)

- Goal is to exercise code based on analysis of code structure
- Tests are designed to ensure that code behaves as expected

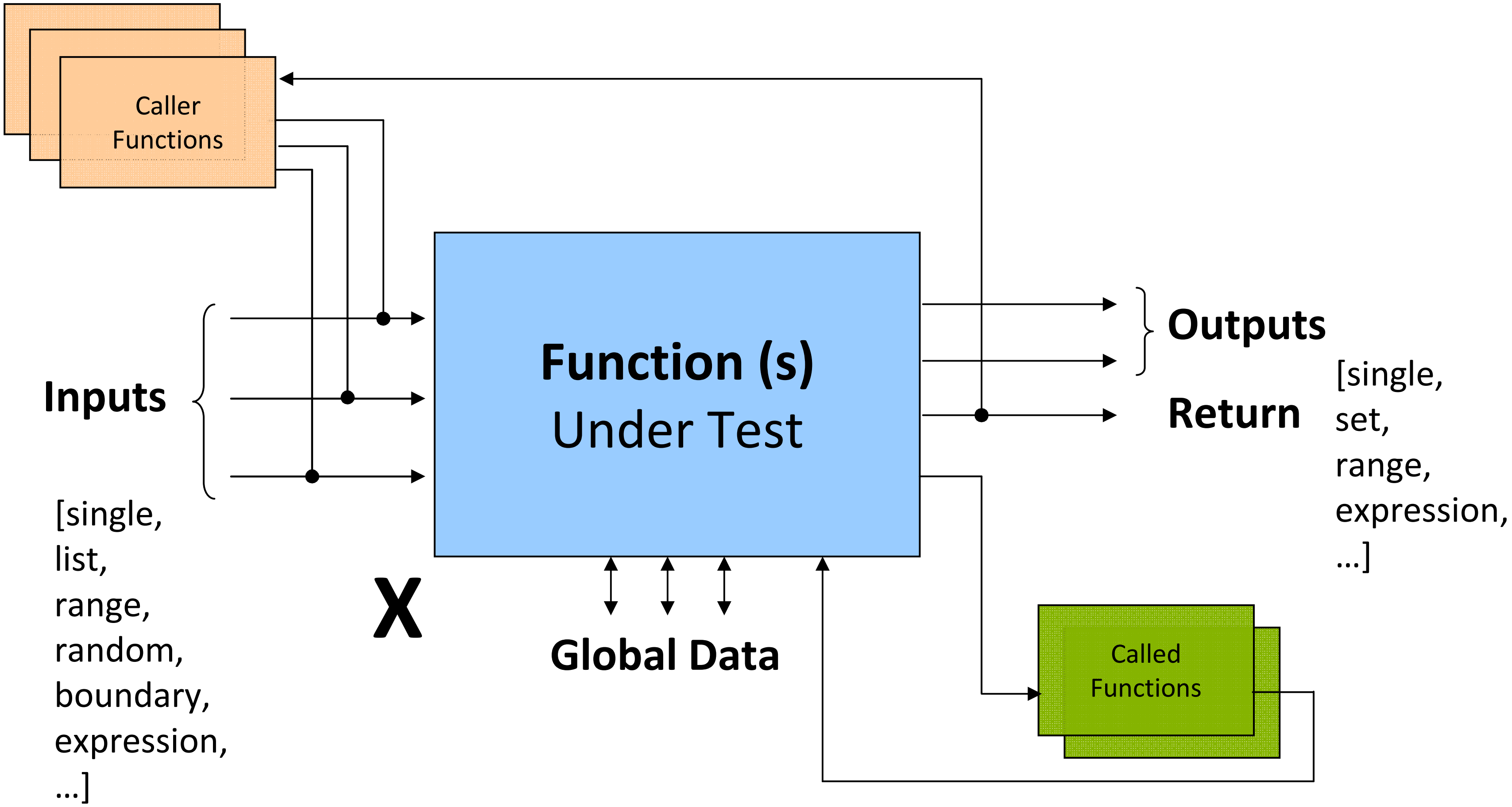
### White Box Testing Is NEEDED:

- Enables testing of program logic and structure
- Achieves superior code coverage (% of statements and branches tested)
- Digs deeper at algorithmic discontinuities & branches (boundary cases and special values get tested)

***"White Box testing presents a required complementary tool to traditional Black Box testing approaches"***

# Functional Verification

*A white box framework must enable rigorous testing of every function of the software*



Caller Functions

Inputs

[single,
list,
range,
random,
boundary,
expression,
…]

X

**Function (s)**
Under Test

Global Data

Outputs

Return

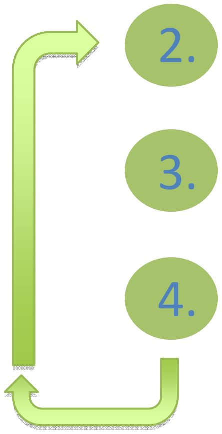
[single,
set,
range,
expression,
…]

Called Functions

# Coverage-Driven Verification

***A white box framework must provide a step-by-step approach towards full functional and statement coverage:***

1. Generate **robustness tests** based on:
   - Inputs, outputs, global variables and returns' **data types**
   - **Permutations, random values, ranges, boundary values…**



2. Run generated tests against the **user's code**
3. Measure **functional and statement coverage**
4. Add **functional tests** to fill the coverage holes based on:
   - **User-provided** values
   - Outputs of **actual test run**
   - **Function Stubs**
   - **Function Monitors** (snapshots)

# Metrics-Based Methodology

## *Continuous tracking of the software verification process:*

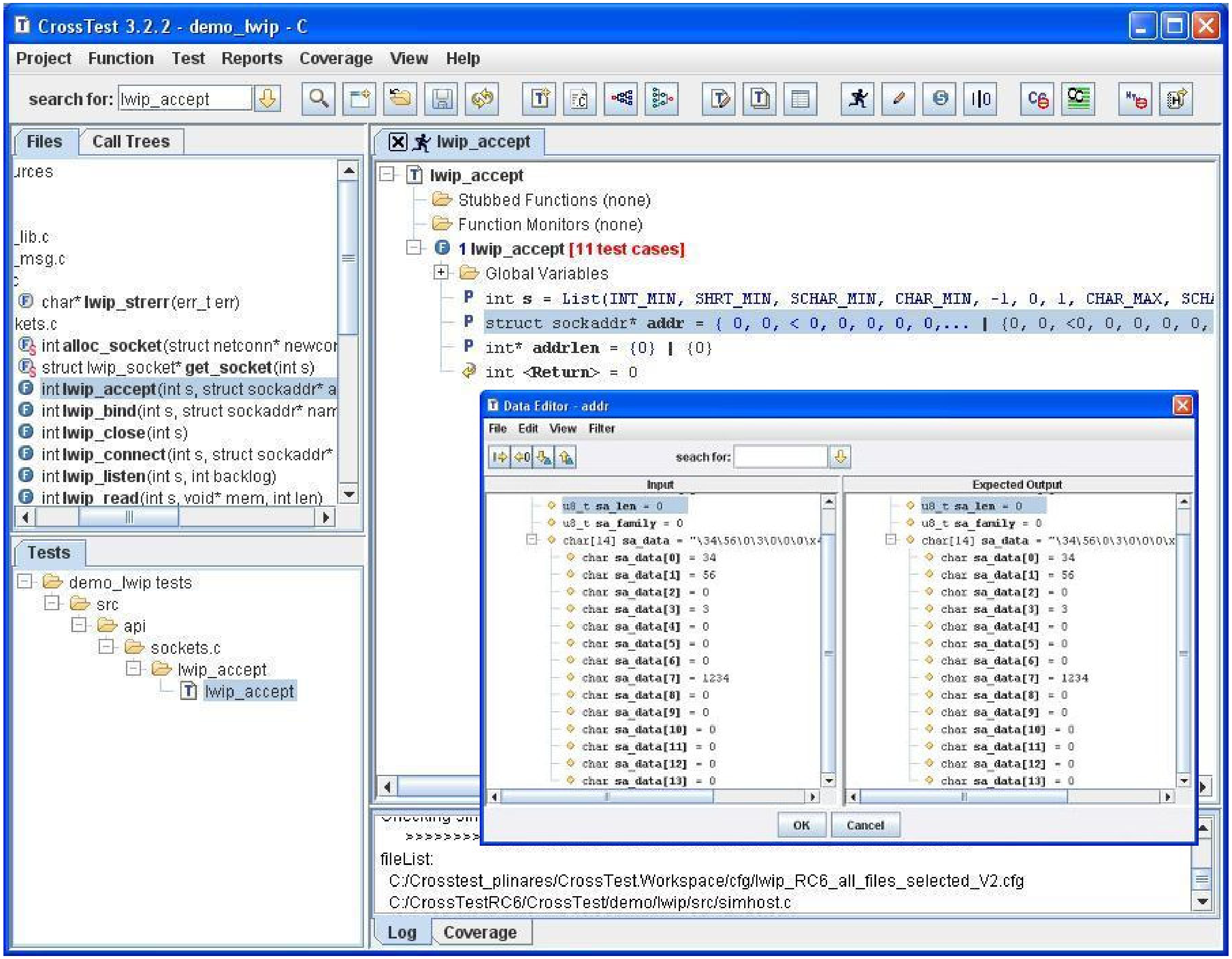
- Testing metrics are collected at all levels of abstraction: **function, file and application** levels

- Metrics include **test success and failure** rates, **functional and statement coverage**

- Results are rolled up into a **dashboard** to track individual and team progress towards verification closure

# CrossTest – The User Interface

XCrossTest, Inc.



```
urces
_lib.c
_msg.c
c
char* lwip_strerr(err_t err)
kets.c
int alloc_socket(struct netconn* newcon
struct lwip_socket* get_socket(int s)
int lwip_accept(int s, struct sockaddr* a
int lwip_bind(int s, struct sockaddr* nam
int lwip_close(int s)
int lwip_connect(int s, struct sockaddr*
int lwip_listen(int s, int backlog)
int lwip_read(int s, void* mem, int len)
```

```
demo_lwip tests
  src
    api
      sockets.c
        lwip_accept
          lwip_accept
```

```
lwip_accept
  Stubbed Functions (none)
  Function Monitors (none)
  1 lwip_accept [11 test cases]
    Global Variables
    int s = List(INT_MIN, SHRT_MIN, SCHAR_MIN, CHAR_MIN, -1, 0, 1, CHAR_MAX, SCHA
    struct sockaddr* addr = { 0, 0, < 0, 0, 0, 0, 0,... | {0, 0, <0, 0, 0, 0, 0,
    int* addrlen = {0} | {0}
    int <Return> = 0
```

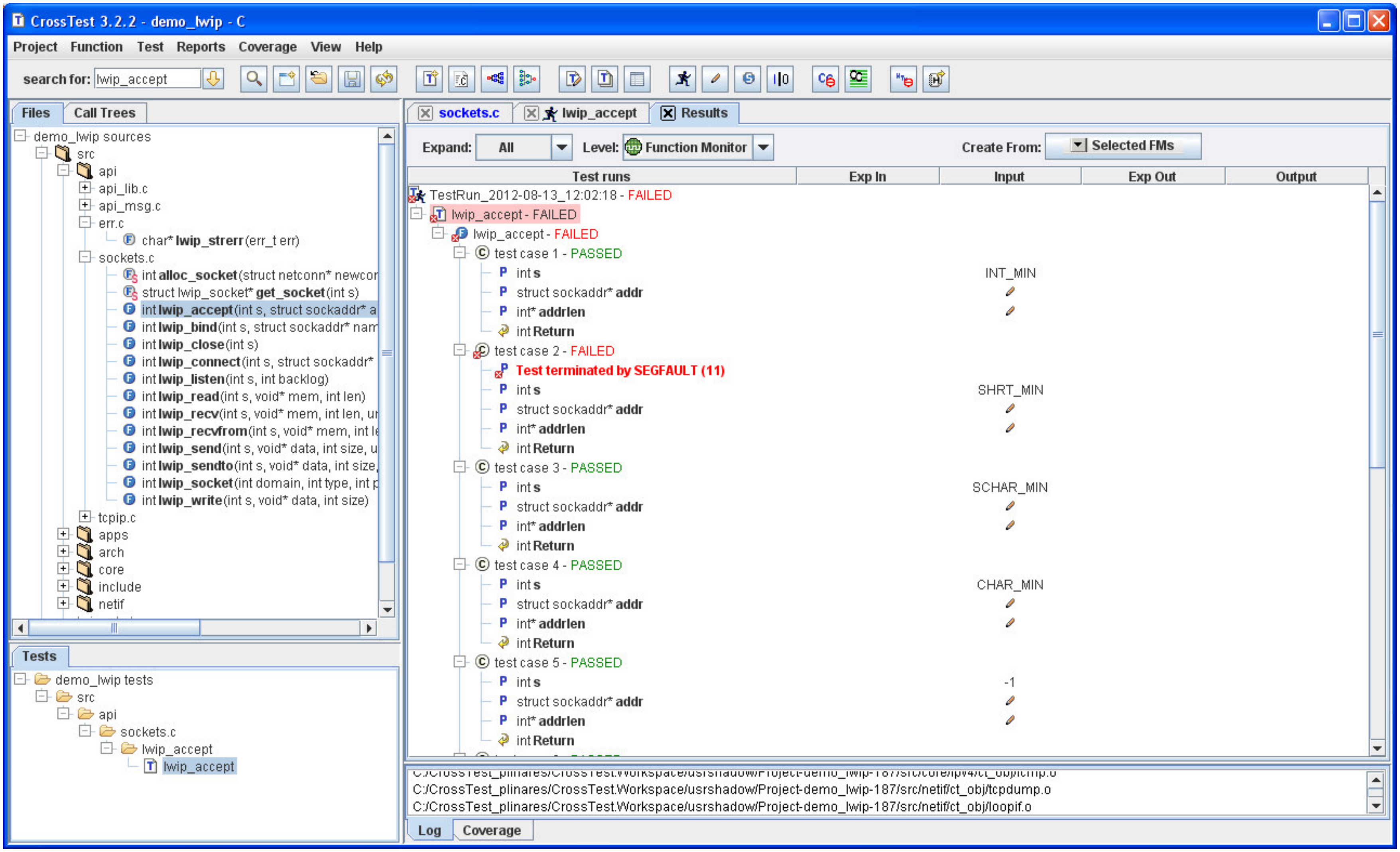
| Input | Expected Output |
|---|---|
| u8_t sa_len = 0 | u8_t sa_len = 0 |
| u8_t sa_family = 0 | u8_t sa_family = 0 |
| char[14] sa_data = "\34\56\0\3\0\0\0\x4 | char[14] sa_data = "\34\56\0\3\0\0\0\x |
| char sa_data[0] = 34 | char sa_data[0] = 34 |
| char sa_data[1] = 56 | char sa_data[1] = 56 |
| char sa_data[2] = 0 | char sa_data[2] = 0 |
| char sa_data[3] = 3 | char sa_data[3] = 3 |
| char sa_data[4] = 0 | char sa_data[4] = 0 |
| char sa_data[5] = 0 | char sa_data[5] = 0 |
| char sa_data[6] = 0 | char sa_data[6] = 0 |
| char sa_data[7] = 1234 | char sa_data[7] = 1234 |
| char sa_data[8] = 0 | char sa_data[8] = 0 |
| char sa_data[9] = 0 | char sa_data[9] = 0 |
| char sa_data[10] = 0 | char sa_data[10] = 0 |
| char sa_data[11] = 0 | char sa_data[11] = 0 |
| char sa_data[12] = 0 | char sa_data[12] = 0 |
| char sa_data[13] = 0 | char sa_data[13] = 0 |

```
Checking sim
  >>>>>>>>
fileList:
  C:/Crosstest_plinares/CrossTest.Workspace/cfg/lwip_RC6_all_files_selected_V2.cfg
  C:/CrossTestRC6/CrossTest/demo/lwip/src/simhost.c
```

## Test Creation

# CrossTest – The User Interface

XCrossTest, Inc.



```
demo_lwip sources
  src
    api
      api_lib.c
      api_msg.c
      err.c
        char* lwip_strerr(err_t err)
      sockets.c
        int alloc_socket(struct netconn* newcon
        struct lwip_socket* get_socket(int s)
        int lwip_accept(int s, struct sockaddr* a
        int lwip_bind(int s, struct sockaddr* nam
        int lwip_close(int s)
        int lwip_connect(int s, struct sockaddr*
        int lwip_listen(int s, int backlog)
        int lwip_read(int s, void* mem, int len)
        int lwip_recv(int s, void* mem, int len, ur
        int lwip_recvfrom(int s, void* mem, int le
        int lwip_send(int s, void* data, int size, u
        int lwip_sendto(int s, void* data, int size,
        int lwip_socket(int domain, int type, int p
        int lwip_write(int s, void* data, int size)
      tcpip.c
    apps
    arch
    core
    include
    netif
```

```
demo_lwip tests
  src
    api
      sockets.c
        lwip_accept
          lwip_accept
```

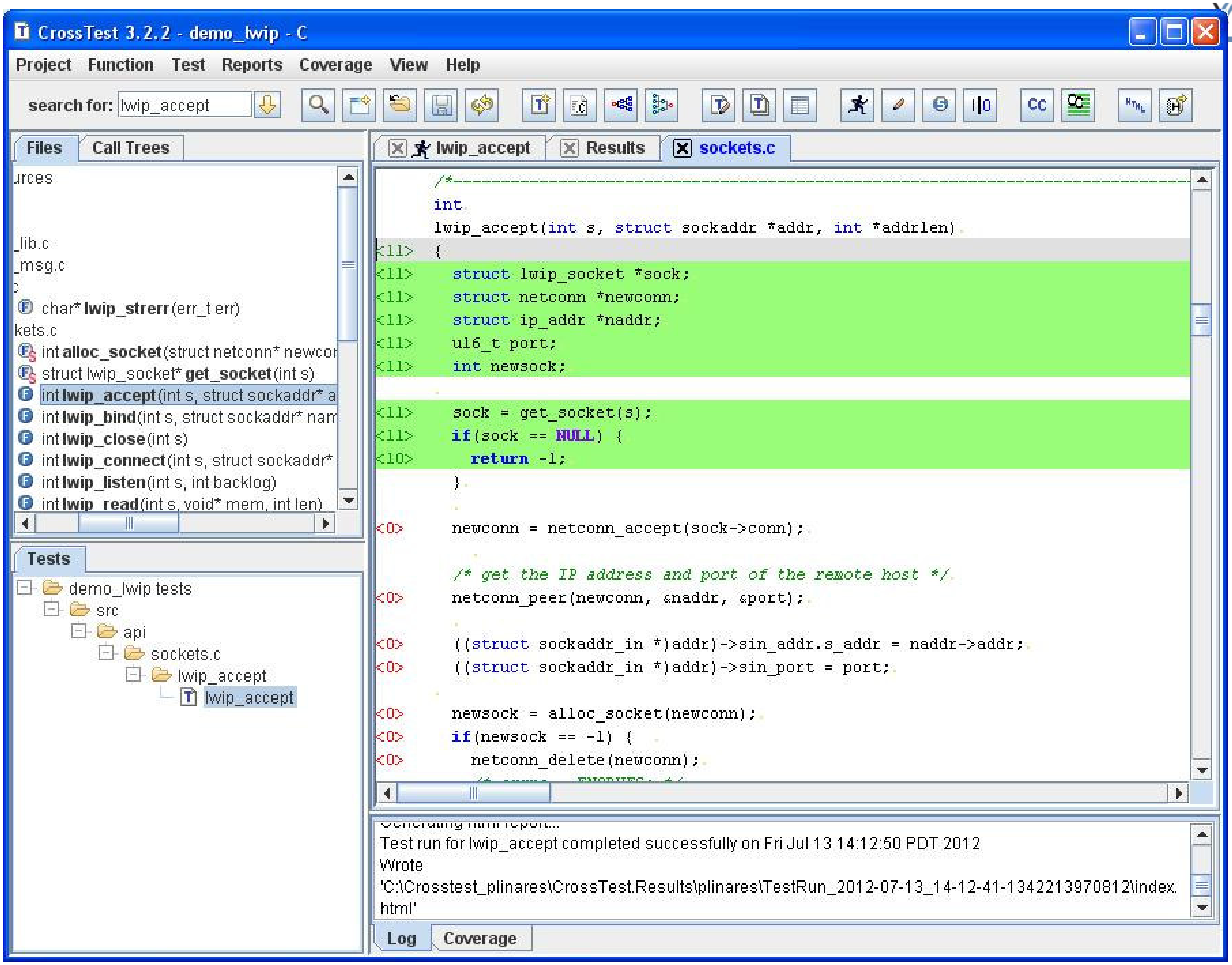
```
Test runs                                                  Exp In   Input       Exp Out   Output
TestRun_2012-08-13_12:02:18 - FAILED
  lwip_accept - FAILED
    lwip_accept - FAILED
      test case 1 - PASSED
        P int s                                                     INT_MIN
        P struct sockaddr* addr
        P int* addrlen
          int Return
      test case 2 - FAILED
        P Test terminated by SEGFAULT (11)
        P int s                                                     SHRT_MIN
        P struct sockaddr* addr
        P int* addrlen
          int Return
      test case 3 - PASSED
        P int s                                                     SCHAR_MIN
        P struct sockaddr* addr
        P int* addrlen
          int Return
      test case 4 - PASSED
        P int s                                                     CHAR_MIN
        P struct sockaddr* addr
        P int* addrlen
          int Return
      test case 5 - PASSED
        P int s                                                     -1
        P struct sockaddr* addr
        P int* addrlen
          int Return
```

```
C:/CrossTest_plinares/CrossTest.Workspace/usrshadow/Project-demo_lwip-187/src/core/ipv4/ct_obj/icmp.o
C:/CrossTest_plinares/CrossTest.Workspace/usrshadow/Project-demo_lwip-187/src/netif/ct_obj/tcpdump.o
C:/CrossTest_plinares/CrossTest.Workspace/usrshadow/Project-demo_lwip-187/src/netif/ct_obj/loopif.o
```

## Test Execution

# CrossTest – The User Interface

CrossTest, Inc.



```
urces

_lib.c
_msg.c
c
char* lwip_strerr(err_t err)
kets.c
int alloc_socket(struct netconn* newcon
struct lwip_socket* get_socket(int s)
int lwip_accept(int s, struct sockaddr* a
int lwip_bind(int s, struct sockaddr* nam
int lwip_close(int s)
int lwip_connect(int s, struct sockaddr*
int lwip_listen(int s, int backlog)
int lwip_read(int s, void* mem, int len)
```

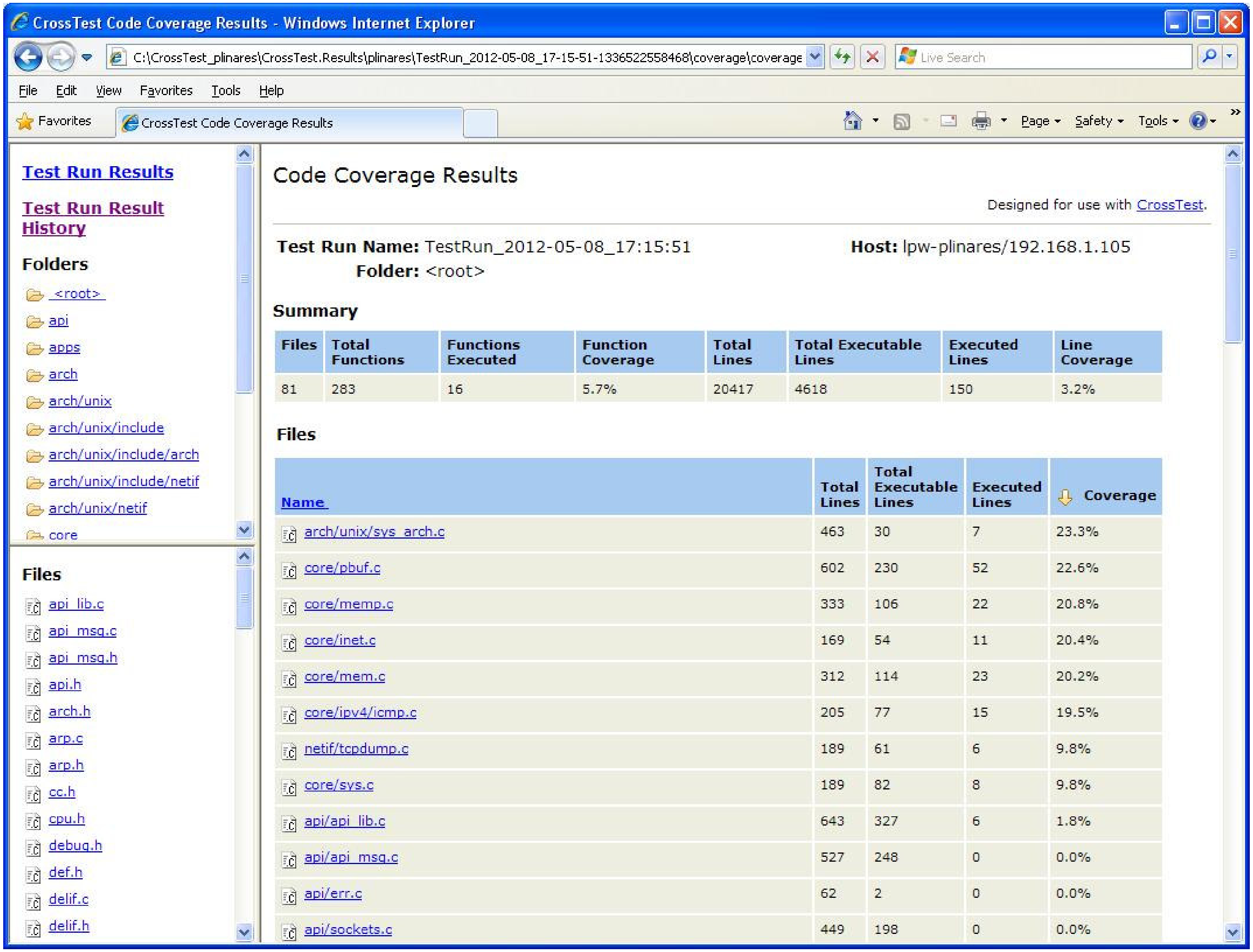
```
demo_lwip tests
  src
    api
      sockets.c
        lwip_accept
          lwip_accept
```

```
      /*----------------------------------------------------------------------------
      int
      lwip_accept(int s, struct sockaddr *addr, int *addrlen)
<11>  {
<11>    struct lwip_socket *sock;
<11>    struct netconn *newconn;
<11>    struct ip_addr *naddr;
<11>    u16_t port;
<11>    int newsock;

<11>    sock = get_socket(s);
<11>    if(sock == NULL) {
<10>      return -1;
        }

<0>     newconn = netconn_accept(sock->conn);

        /* get the IP address and port of the remote host */
<0>     netconn_peer(newconn, &naddr, &port);

<0>     ((struct sockaddr_in *)addr)->sin_addr.s_addr = naddr->addr;
<0>     ((struct sockaddr_in *)addr)->sin_port = port;

<0>     newsock = alloc_socket(newconn);
<0>     if(newsock == -1) {
<0>       netconn_delete(newconn);
```

```
Generating html report...
Test run for lwip_accept completed successfully on Fri Jul 13 14:12:50 PDT 2012
Wrote
'C:\Crosstest_plinares\CrossTest.Results\plinares\TestRun_2012-07-13_14-12-41-1342213970812\index.
html'
```

## Code coverage

# CrossTest – The Dashboard


CrossTest Code Coverage Results - Windows Internet Explorer
C:\CrossTest_plinares\CrossTest.Results\plinares\TestRun_2012-05-08_17-15-51-1336522558468\coverage\coverage
Live Search
File Edit View Favorites Tools Help
Favorites
CrossTest Code Coverage Results
Page Safety Tools

Test Run Results

Test Run Result History

Folders

- <root>
- api
- apps
- arch
- arch/unix
- arch/unix/include
- arch/unix/include/arch
- arch/unix/include/netif
- arch/unix/netif
- core

Files

- api_lib.c
- api_msg.c
- api_msg.h
- api.h
- arch.h
- arp.c
- arp.h
- cc.h
- cpu.h
- debug.h
- def.h
- delif.c
- delif.h

## Code Coverage Results

Designed for use with CrossTest.

**Test Run Name:** TestRun_2012-05-08_17:15:51 **Host:** lpw-plinares/192.168.1.105

**Folder:** <root>

### Summary

| Files | Total Functions | Functions Executed | Function Coverage | Total Lines | Total Executable Lines | Executed Lines | Line Coverage |
|---|---|---|---|---|---|---|---|
| 81 | 283 | 16 | 5.7% | 20417 | 4618 | 150 | 3.2% |

### Files

| Name | Total Lines | Total Executable Lines | Executed Lines | Coverage |
|---|---|---|---|---|
| arch/unix/sys_arch.c | 463 | 30 | 7 | 23.3% |
| core/pbuf.c | 602 | 230 | 52 | 22.6% |
| core/memp.c | 333 | 106 | 22 | 20.8% |
| core/inet.c | 169 | 54 | 11 | 20.4% |
| core/mem.c | 312 | 114 | 23 | 20.2% |
| core/ipv4/icmp.c | 205 | 77 | 15 | 19.5% |
| netif/tcpdump.c | 189 | 61 | 6 | 9.8% |
| core/sys.c | 189 | 82 | 8 | 9.8% |
| api/api_lib.c | 643 | 327 | 6 | 1.8% |
| api/api_msg.c | 527 | 248 | 0 | 0.0% |
| api/err.c | 62 | 2 | 0 | 0.0% |
| api/sockets.c | 449 | 198 | 0 | 0.0% |

# Benefits

- Improve phase containment and reduce integration and system test cycle duration
- Test before the entire SW is complete
- Find defects before integration and system test – reduce time on bug fixing & mgmt = more time coding
- Improve test coverage & code quality
- Collaboration – sharing of tests and data

# Conclusion

CrossTest as an emerging technology bridging the software testing gap between early stage development and integration testing

Full fledged automated test framework

Renewed interest with agile development and exploratory testing