eclipseCON™ 2008

Code Quality Analysis Toolset for Embedded Systems

Kouhei NADEHARA

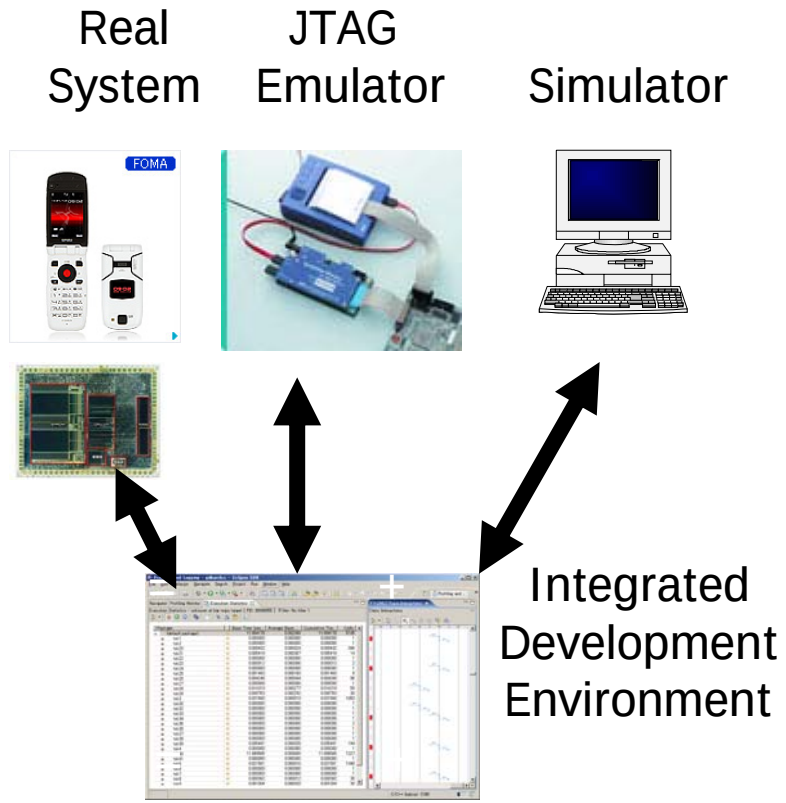
Yuichi NAKAMURA

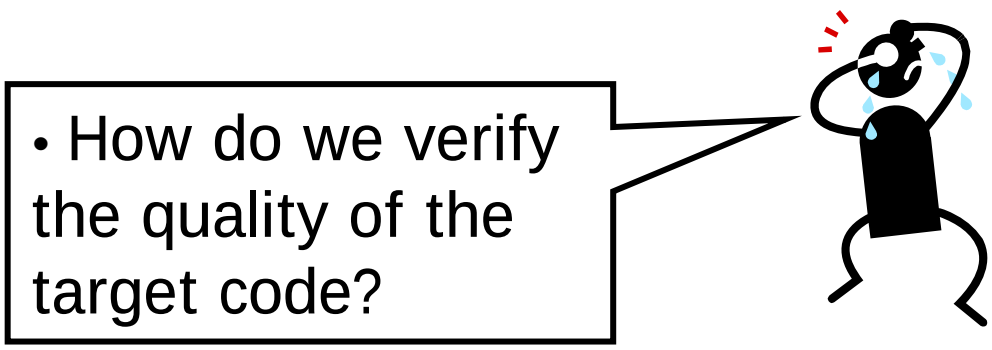
System IP Core Res. Labs., NEC Corporation

 Mar 19, 2008

Embedded System Development with Eclipse

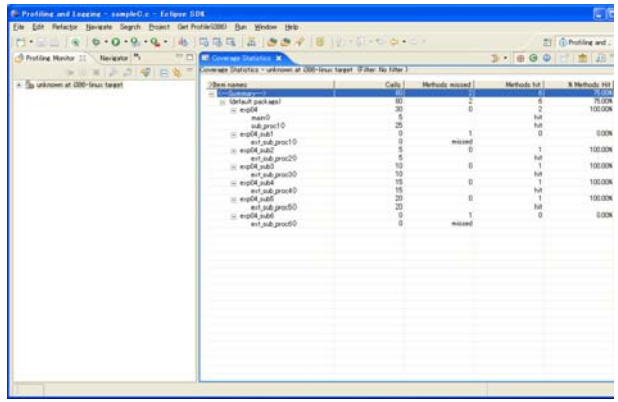
- Deals with various “remote” target systems
  - Software simulator
  - Emulator via JTAG or USB
  - Real system via Ethernet connection
- Remote connection framework is required
  - Employ various existing tools
    - Open-source, GNU, or commercial tools
    - For performance and quality analysis

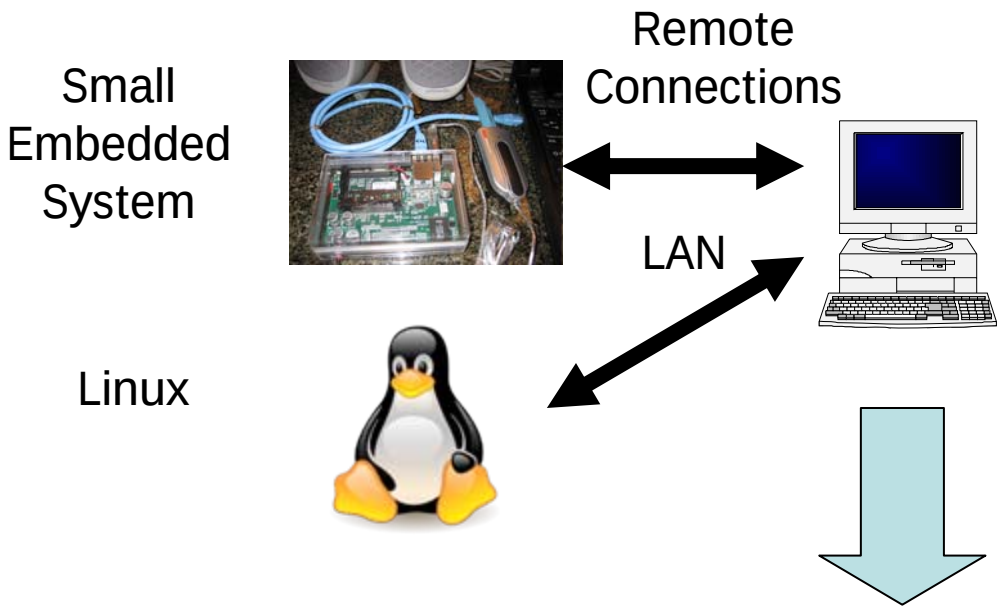




Our Environment

Demonstration at EclipseCON 2007*

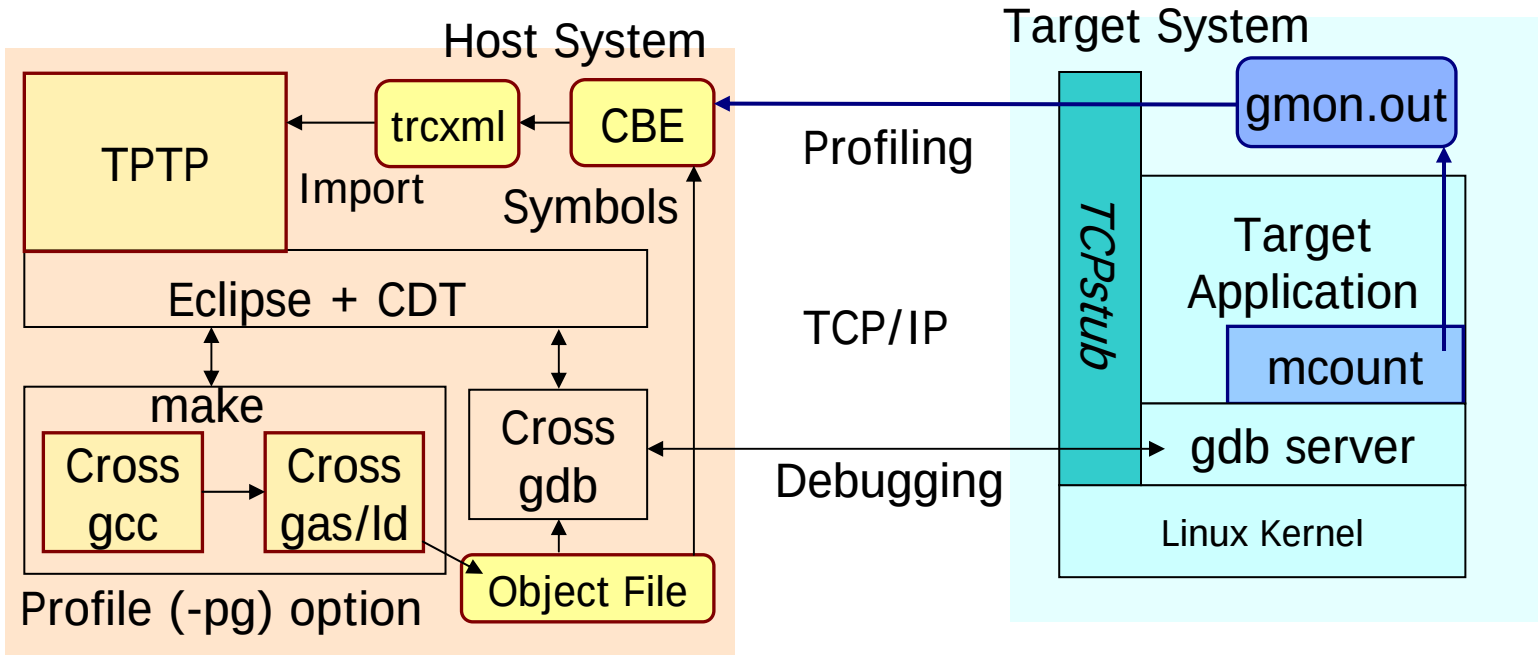




- New in 2008: code quality analysis
  - Memory management with Valgrind (leak, cache statistics)
  - Code coverage testing with Gcov
  - TPTP-based and our original GUIs

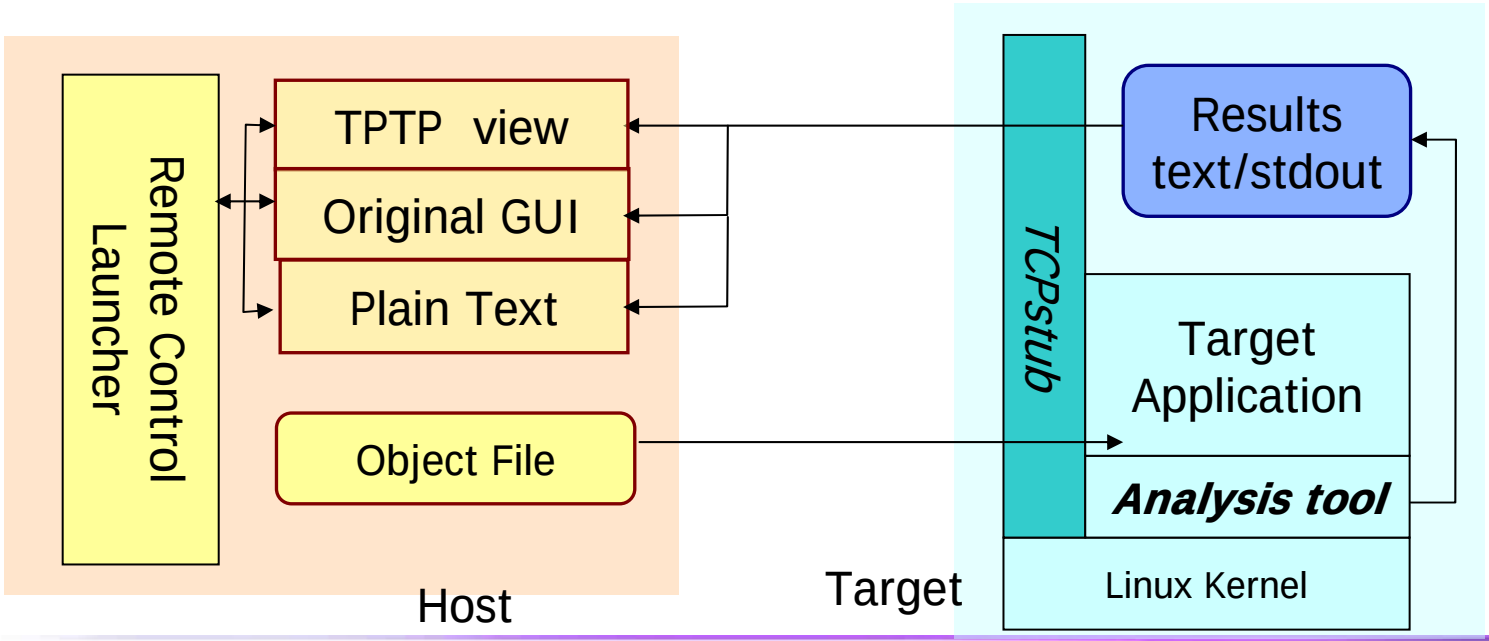
Base System: Remote Debugging/Profiling Architecture

- Communication managed by the “TCPstub”
- Remote debugging with gdb server and CDT
- Remote profiling with GNU profiler (gprof) and TPTP



Code Quality Analysis with “TCPstub”

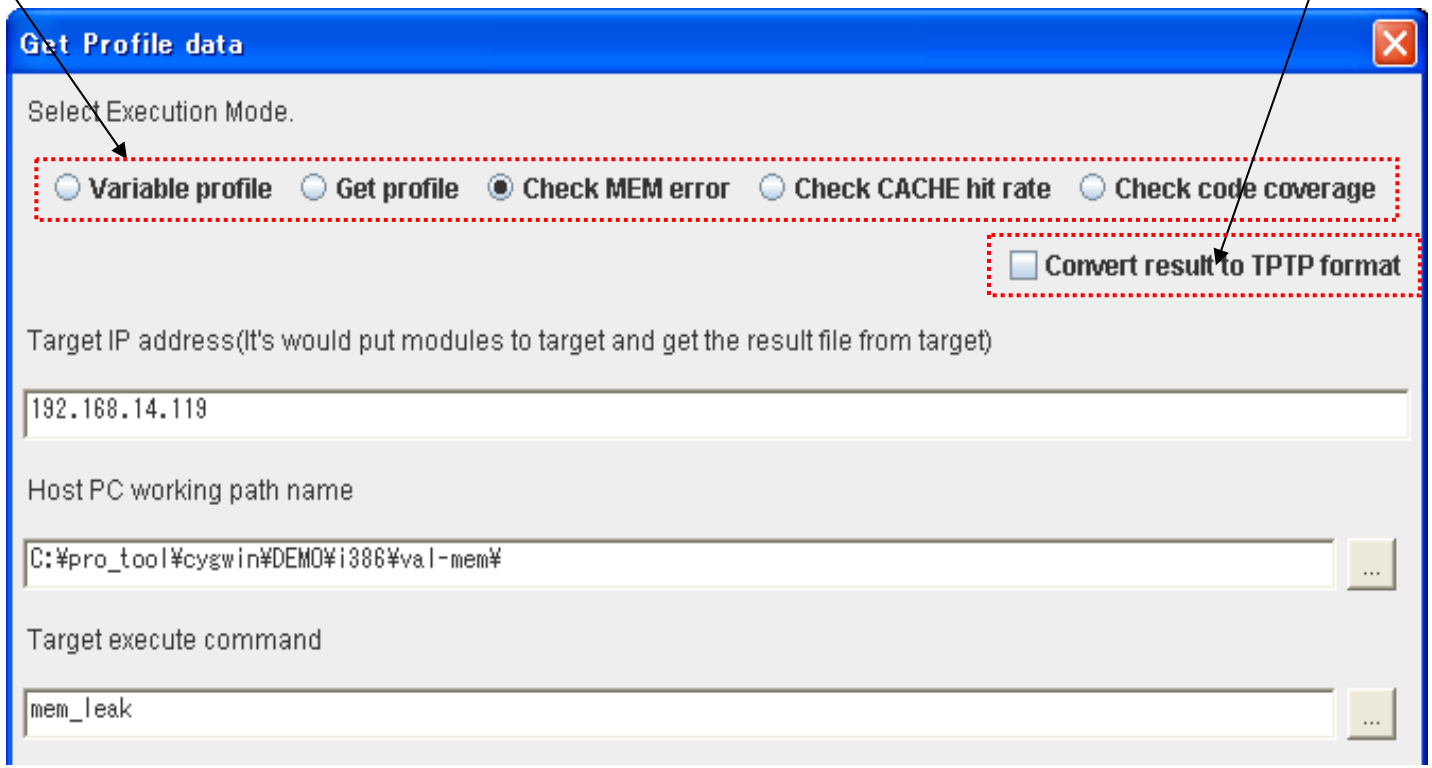
- Network agent “TCPstub” manages communication
  - Invokes the application under an ***Analysis tool*** on the target
  - Retreive its results to the host
- Results shown by TPTP view, or our original GUI



Remote Debugger Launcher

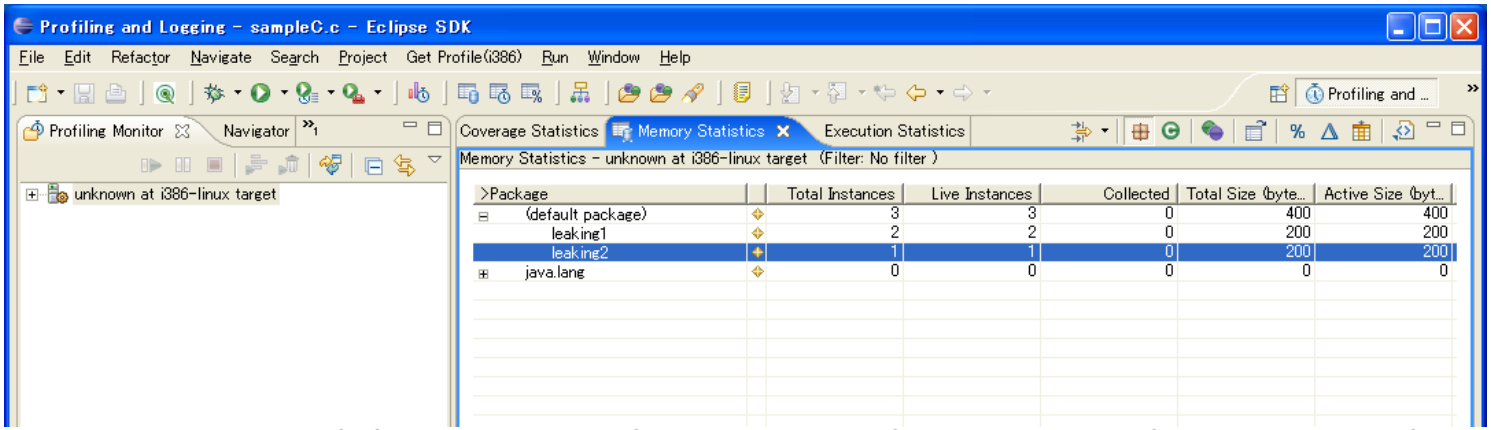
Select analysis tool

Select GUI



Memory Management with Valgrind

- Results shown by the TPTP memory statistics view

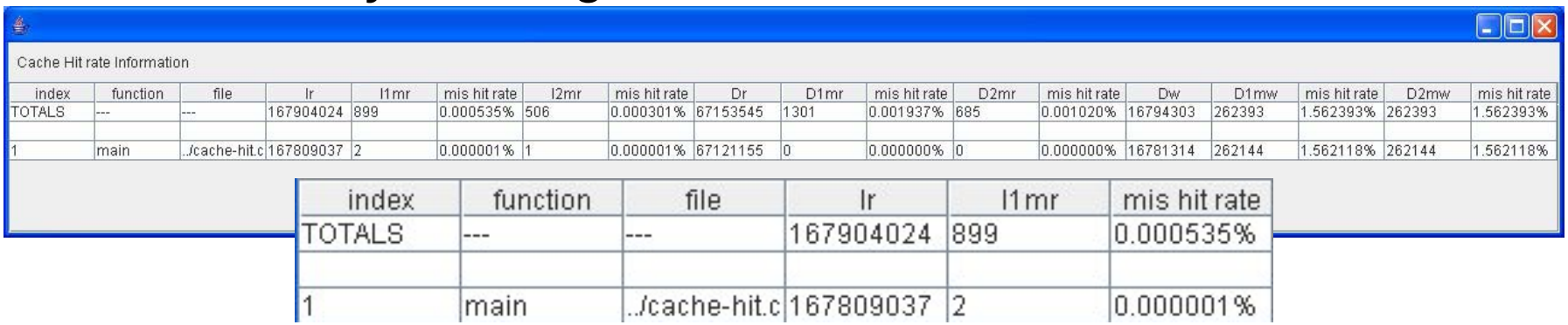


>Package		Total Instances	Live Instances	Collected	Total Size (byte...	Active Size (byt...
(default package)		3	3	0	400	400
leaking1		2	2	0	200	200
leaking2		1	1	0	200	200
java.lang		0	0	0	0	0

Because of the capability of the TPTP memory statistics view, results shown is memory leaks only.

Our GUI for Valgrind

Cache statistics and illegal memory accesses are shown by our original GUI



index	function	file	Ir	I1mr	mis hit rate	I2mr	mis hit rate	Dr	D1mr	mis hit rate	D2mr	mis hit rate	Dw	D1mw	mis hit rate	D2mw	mis hit rate
TOTALS	---	---	167904024	899	0.000535%	506	0.000301%	67153545	1301	0.001937%	685	0.001020%	16794303	262393	1.562393%	262393	1.562393%
1	main	../cache-hit.c	167809037	2	0.000001%	1	0.000001%	67121155	0	0.000000%	0	0.000000%	16781314	262144	1.562118%	262144	1.562118%

index	function	file	Ir	I1mr	mis hit rate
TOTALS	---	---	167904024	899	0.000535%
1	main	../cache-hit.c	167809037	2	0.000001%

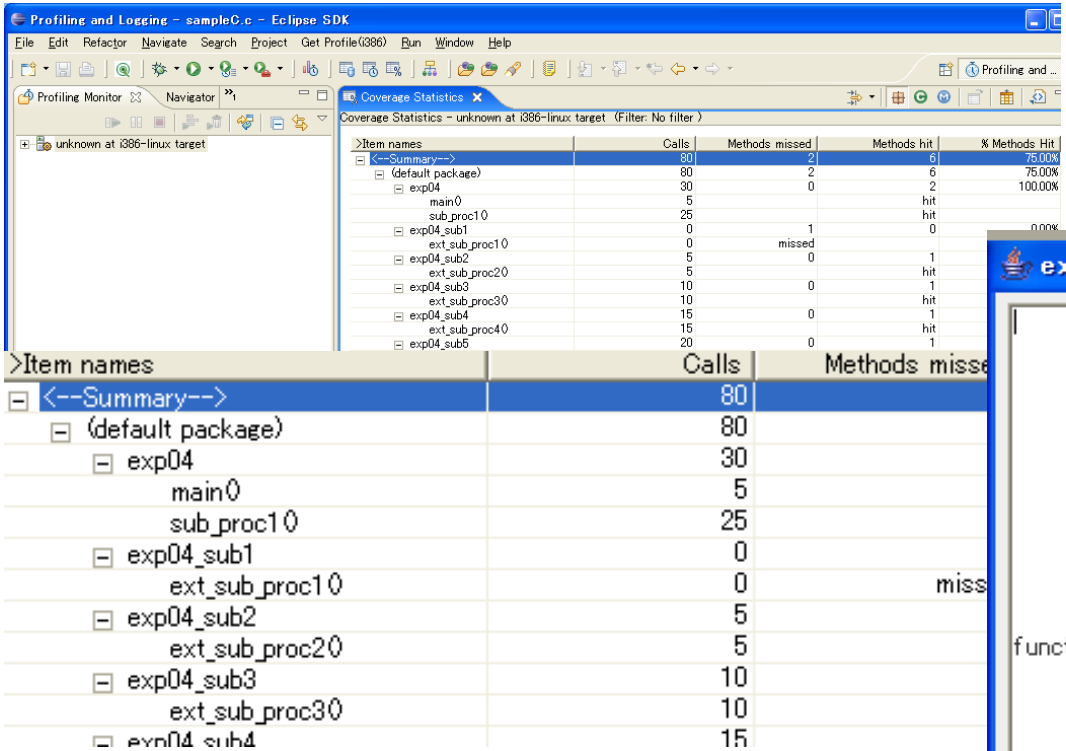
Ir	I1mr	mis hit rate	I2mr	mis hit rate	Dr	D1mr	mis hit rate	D2mr	mis hit rate	Dw	D1mw	mis hit rate	D2mw	mis hit rate
167904024	899	0.000535%	506	0.000301%	67153545	1301	0.001937%	685	0.001020%	16794303	262393	1.562393%	262393	1.562393%
167809037	2	0.000001%	1	0.000001%	67121155	0	0.000000%	0	0.000000%	16781314	262144	1.562118%	262144	1.562118%

Cache statistics

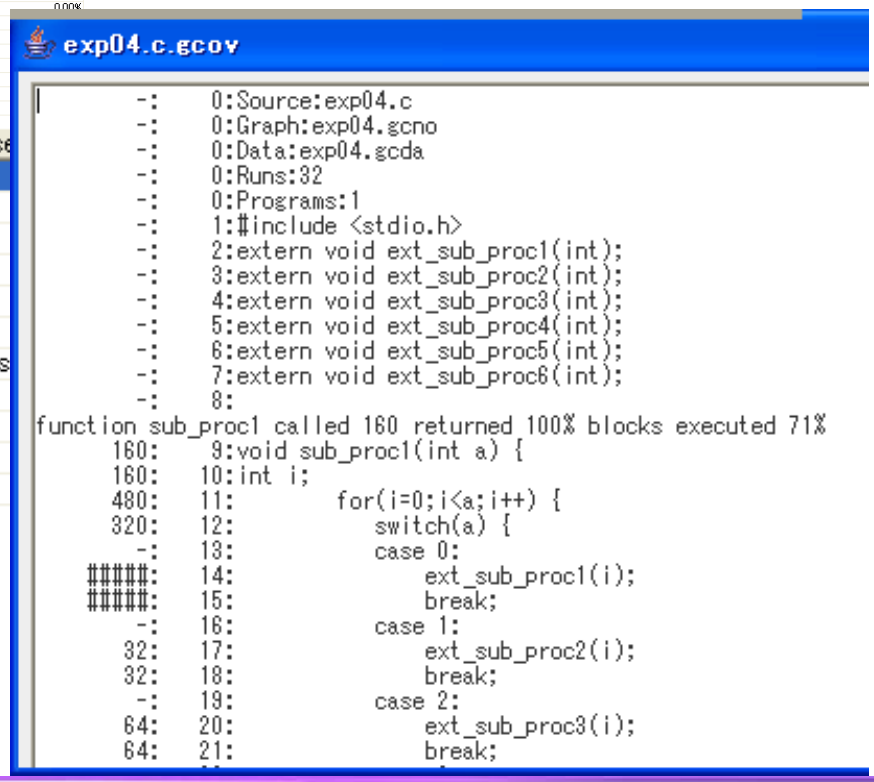
```
==19298== Invalid write of size 1
==19298==    at 0x8048564: leaking1 (mem_leak.c:8)
==19298==    by 0x80485B9: main (mem_leak.c:22)
==19298==  Address 0x402708D is 1 bytes after a block of size 100 alloc'd
==19298==    at 0x40046EA: malloc (vg_replace_malloc.c:149)
==19298==    by 0x804854B: leaking1 (mem_leak.c:6)
==19298==    by 0x80485B9: main (mem_leak.c:22)
==19298==
```

Illegal memory access

Gcov GUIs

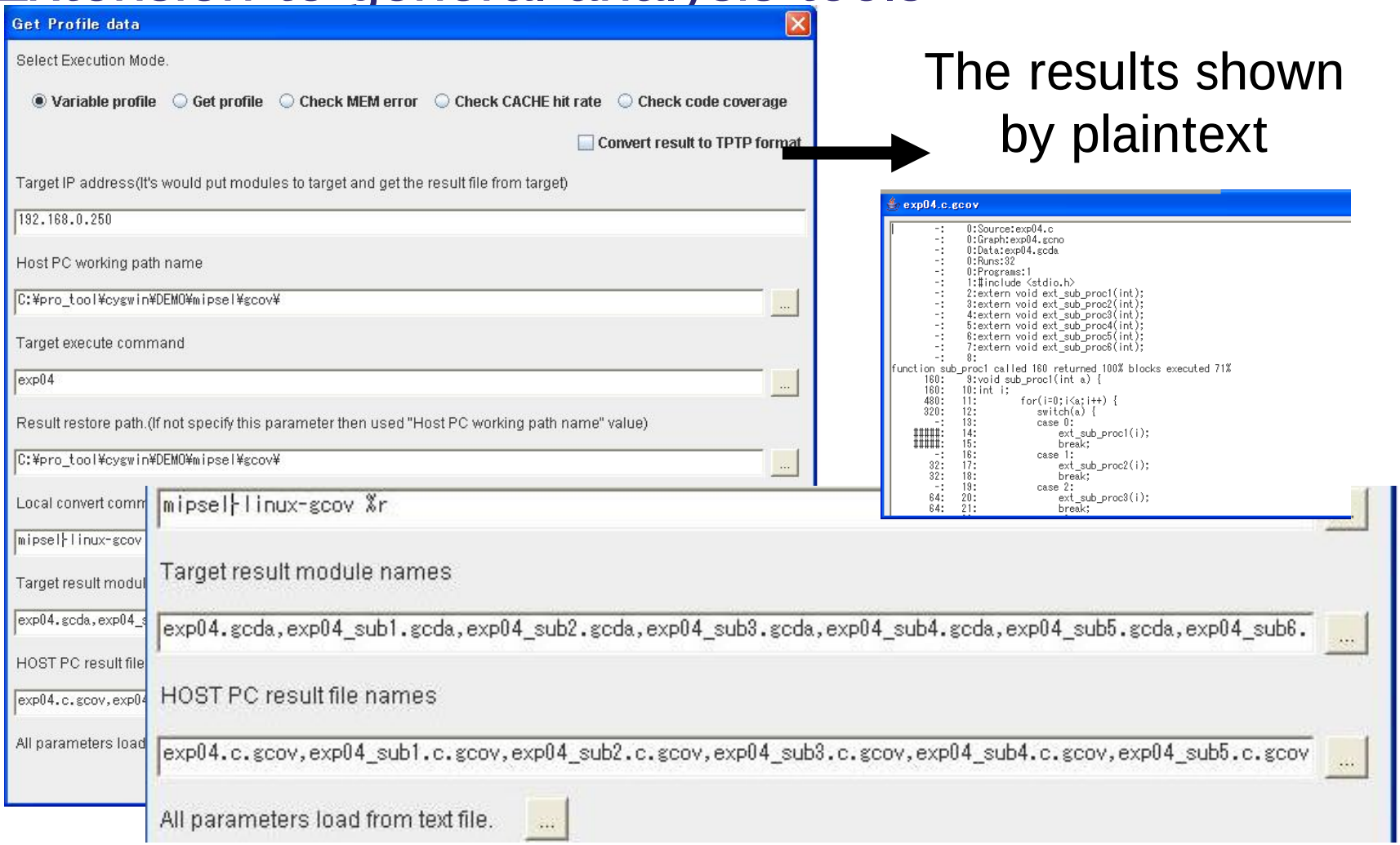


Function-level result shown by TPTP



Detailed line-level result shown by our GUI

Extension to general analysis tools



The results shown by plaintext

```
exp04.c.gcov
        -:    0:Source:exp04.c
        -:    0:Graph:exp04.gcno
        -:    0:Data:exp04.gcda
        -:    0:Runs:32
        -:    0:Programs:1
        -:    1:#include <stdio.h>
        -:    2:extern void ext_sub_proc1(int);
        -:    3:extern void ext_sub_proc2(int);
        -:    4:extern void ext_sub_proc3(int);
        -:    5:extern void ext_sub_proc4(int);
        -:    6:extern void ext_sub_proc5(int);
        -:    7:extern void ext_sub_proc6(int);
        -:    8:
function sub_proc1 called 160 returned 100% blocks executed 71%
      160:    9:void sub_proc1(int a) {
      160:   10:int i;
      480:   11:        for(i=0;i<a;i++) {
      320:   12:            switch(a) {
        -:   13:            case 0:
    #####:   14:                ext_sub_proc1(i);
    #####:   15:                break;
        -:   16:            case 1:
       32:   17:                ext_sub_proc2(i);
       32:   18:                break;
        -:   19:            case 2:
       64:   20:                ext_sub_proc3(i);
       64:   21:                break;
```

Setting for various tools

Conclusion

- Code quality analysis for embedded sytems
  - Memory management with Valgrind
  - Code coverage testing with Gcov
- Remote system analysis with network agent
  - By extending “TCPstub,” originally designed for debugging and profiling
- Seamlessly integrated to Eclipse
  - TPTP-based and our original GUIs
    - Because of the capability of the TPTP view

 |