Xest: An Automated Framework for Regression Testing of Embedded Software

Matt Netkow[1] Dennis Brylow[2]

[1]Solution Architecture Team
The SAVO Group

[2]Department of Mathematics, Statistics and Computer Science
Marquette University

Workshop on Embedded Systems Education
October 28, 2010

Xinu External Suite Tester (XEST)

- Automated, parallelized regression testing
- Embedded software kernels running directly on real embedded hardware

Outline

Project Nexos

Original project supported in part by:
NSF DUE CCLI #0737476 - Marquette University (MU)
NSF DUE CCLI #0737243 - University of Buffalo (UB)

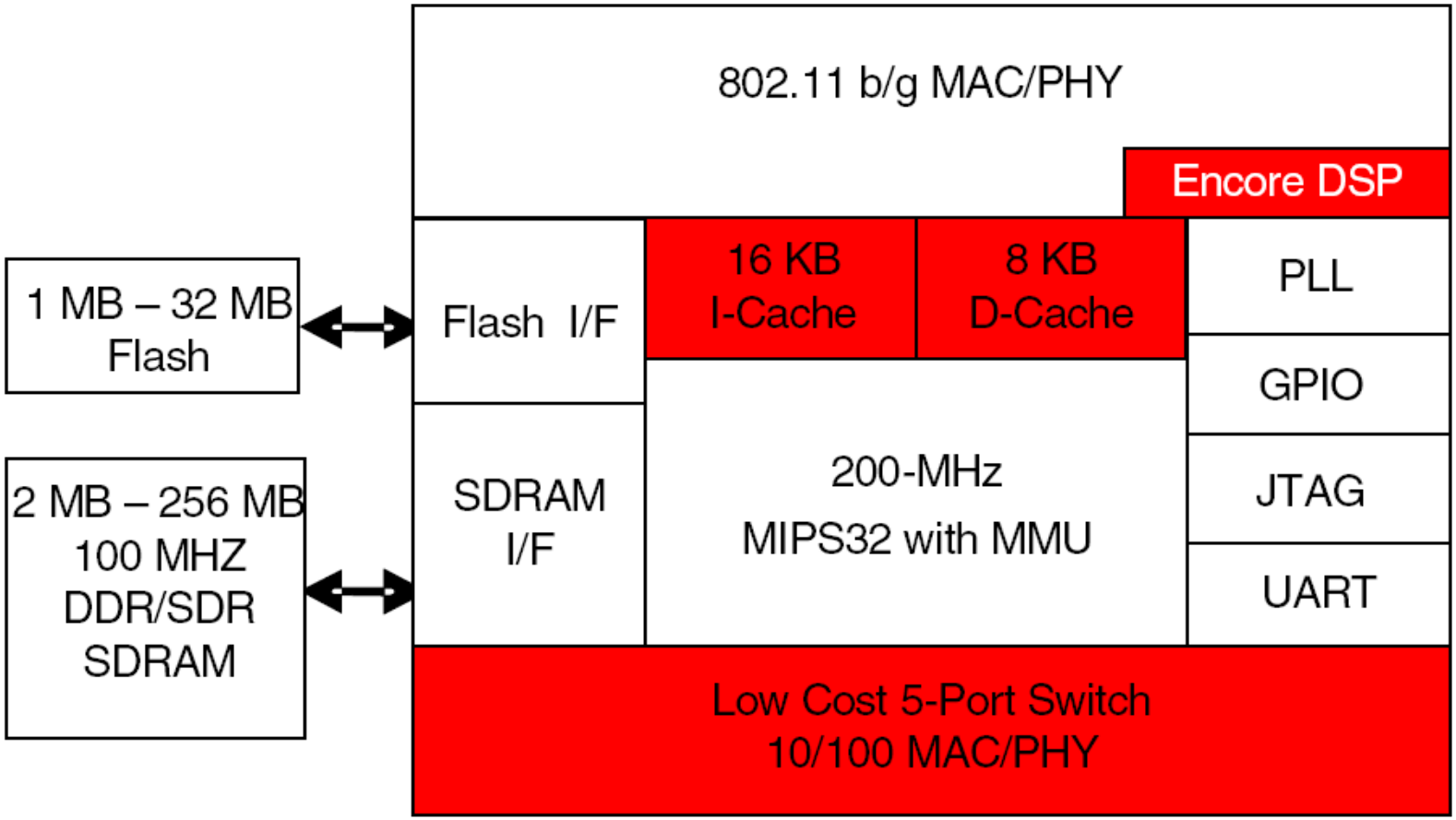
Goals

- Curriculum materials, supporting experimental laboratory for hands-on projects in embedded systems courses,
- Inexpensive, flexible, commodity embedded hardware,
- Freely available development, testing and debugging tools,
- Applicable and duplicable in a wide variety of settings.

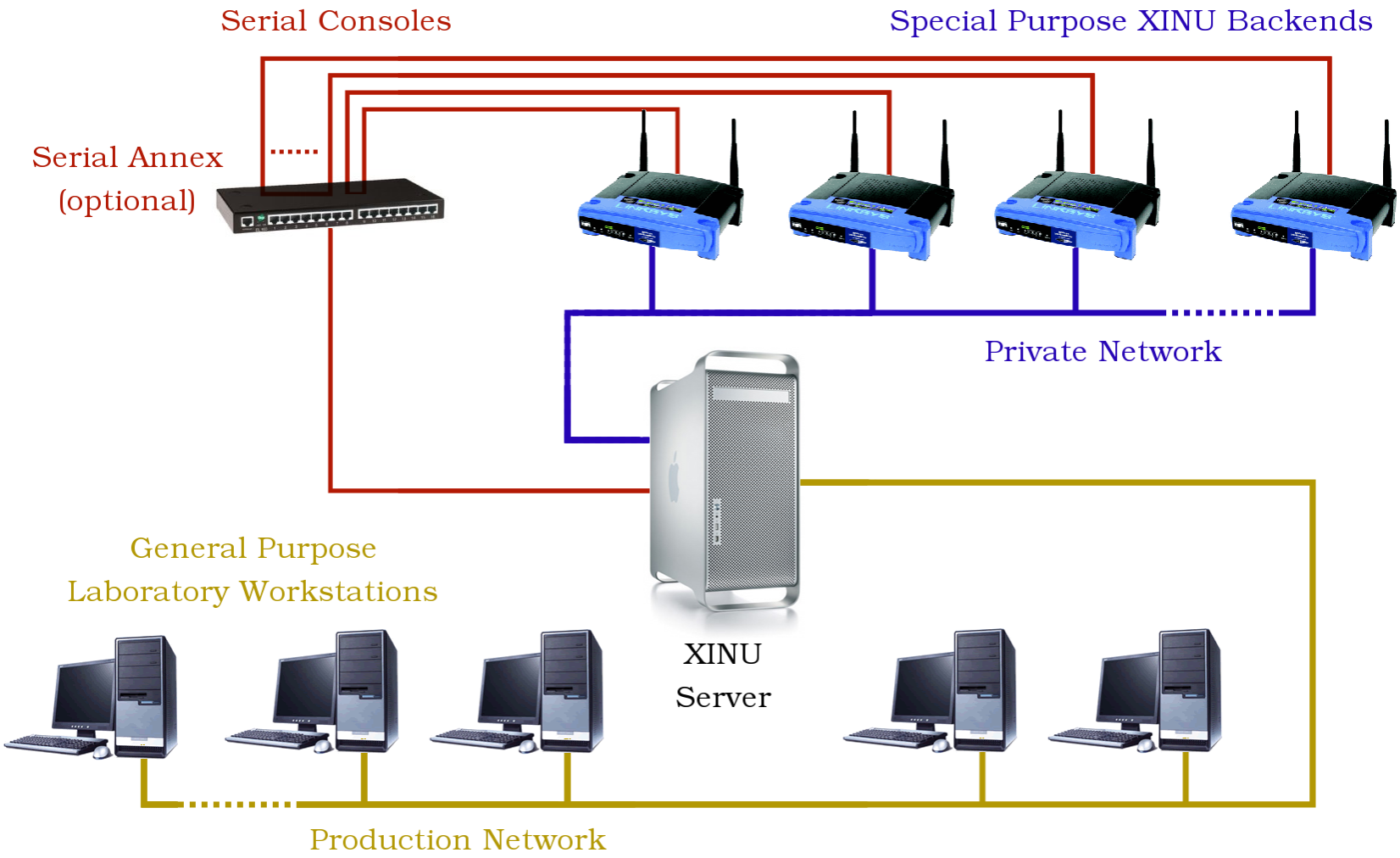
Project continues in cooperation with University of Mississippi, Purdue University, and several other test sites.

The Linksys WRT54GL series wireless router

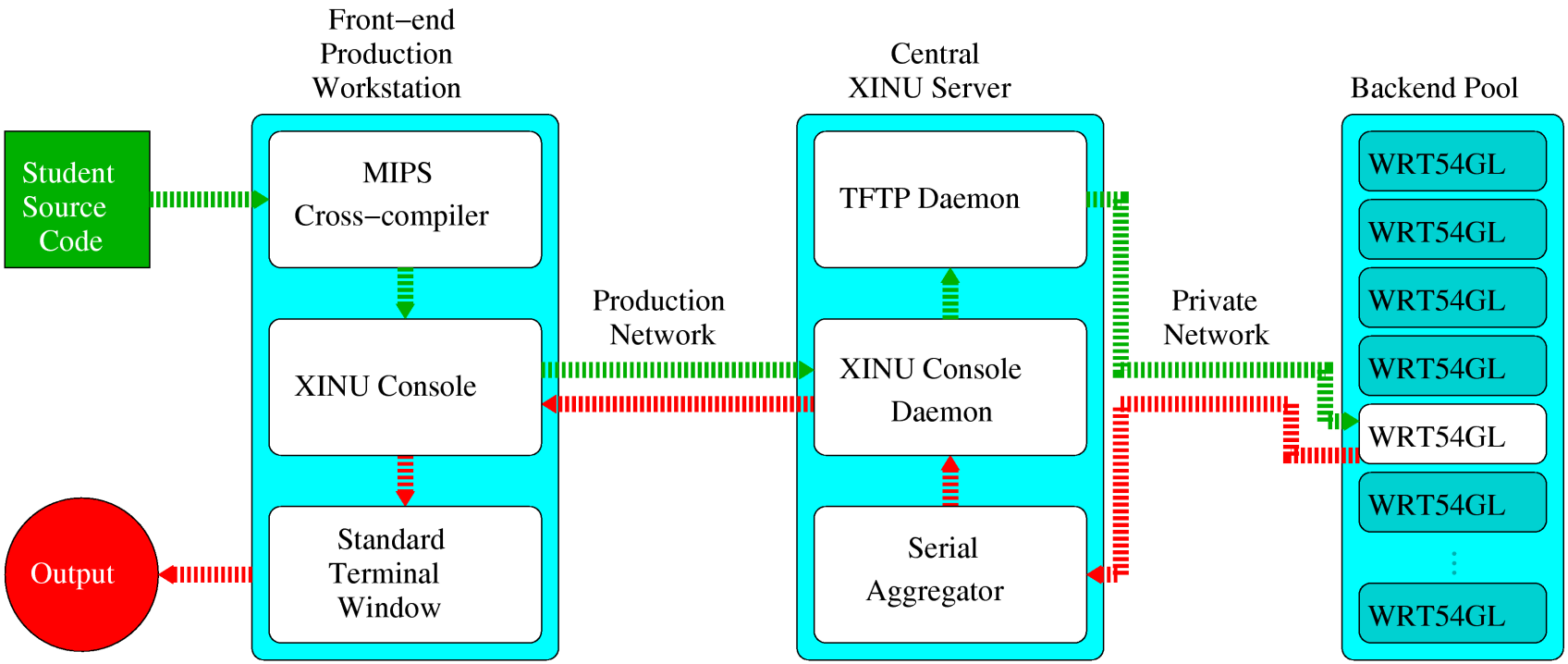
The Broadcom BCM53xx series System on a Chip (SoC)



Embedded Xinu Laboratory Structure



Programming Enviroment



Tools

Compiler

- GCC cross-compiler for 32-bit mipsel
- Standard ELF format linking

Lab Tools

- Xinu Console Daemon tools manage backend pool
- Standard DHCP and TFTP daemons provide network support
- Power control daemon

Embedded Xinu

- ANSI standard C, standard GNU make
- Freely available under BSD-style license

Requirements

Xinu Lab Infrastructure

- Serial console
- Firmware that can upload a fresh O/S kernel over an I/O device at boot time

Supported Platforms

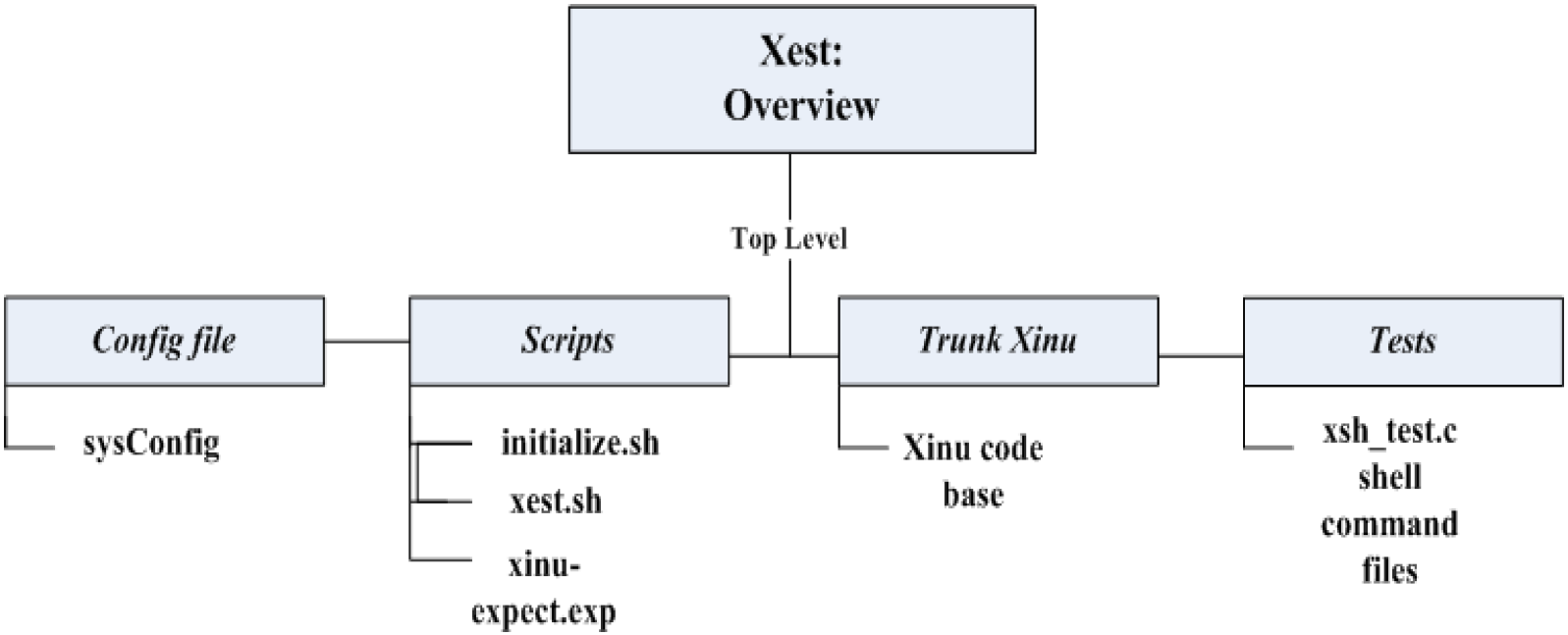
- Broadcom 32-bit MIPS (little endian) SoC
- Atheros 32-bit MIPS (big endian) SoC
- Arduino 8-bit ATmega1280
- Motorola 16-bit 68HC11 and 68HC12 dev boards
- Pentium 64-bit dual-core workstations with PXE boot capabilities

Requirements

Xest Requirements

- Embedded platforms that can be remotely power-cycled into a "known good" state,
- Provisions for unattended upload of software image to embedded device, and
- At least one stable I/O communication channel with a "fail safe" device driver.
- Multiple, concurrently accessible target platforms.

Overview



Bash + Expect + Xinu == Xest

Implementation

System Initialization

- Xinu Code
- E-Mail results

Test Creation

- Test Cases
- xsh_test.c files
- Print statements are test markers:

```
printf("===TEST BEGIN===");
code to test
printf("===TEST END===");
```

Implementation

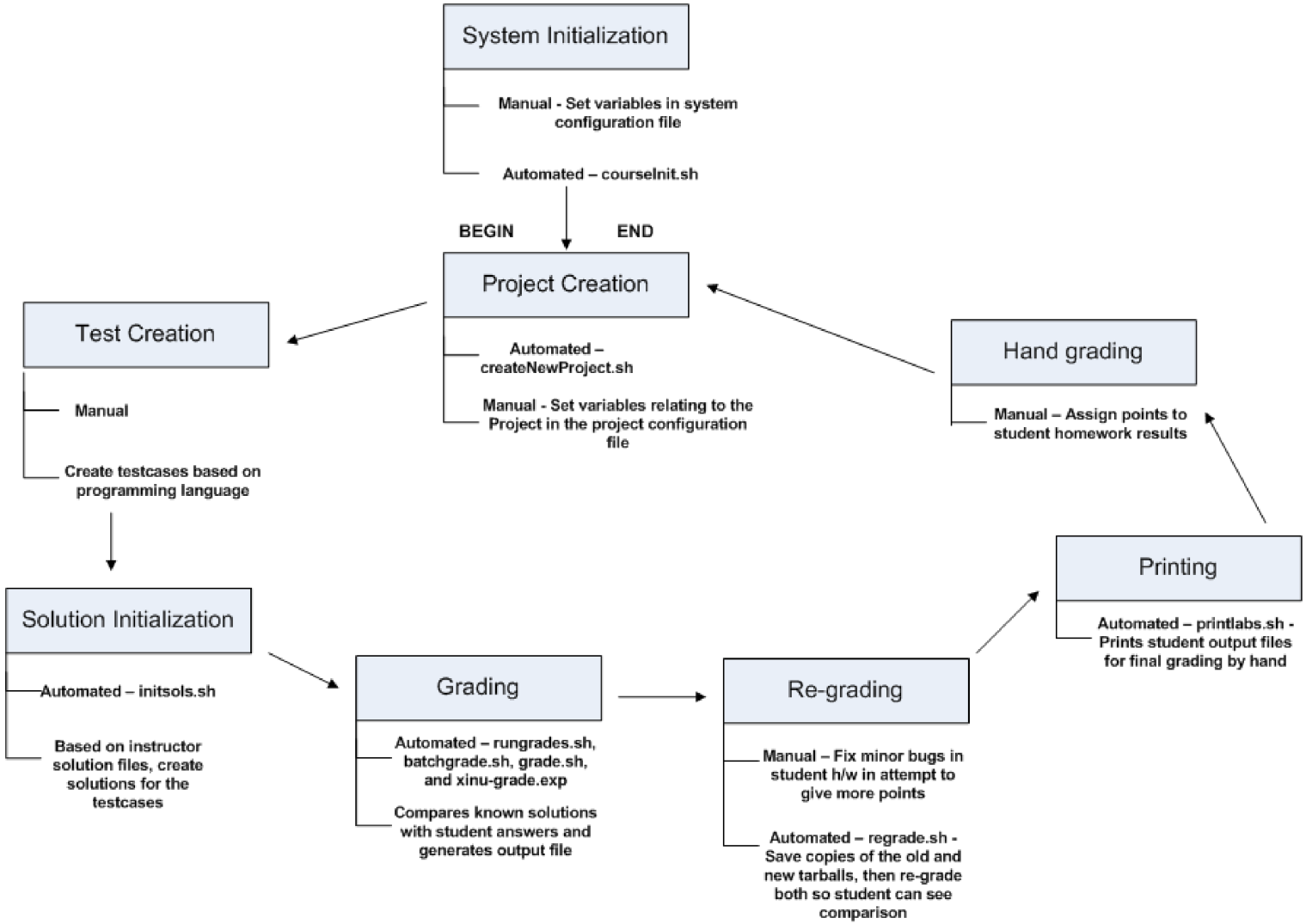
Solution Initialization

- Initialization bash script
- Each testcase copied into test-hook location in source
- Kernel re-compiled with each testcase – minimize overhead
- `test` command sent over serial console
- Expect script looks for special test marker and saves output to a file the "solution"
- Repeated for entire suite

Xest as Grading Tool

- Regression tool to student assessment tool?
- Support for multiple courses?
  - Hardware systems, O/S, Networking, Compilers, and Intro to Java.
  - Grading scripts, tests, solution files
- Error checking?
  - Check existence of files
  - Log errors into separate file
  - Recover from student h/w submission errors

Evaluation of Student Submissions



Capabilities

Testing Categories

- Application-specific faults
  (Failure to anticipate realistic input combinations)
- Implementation-specific faults
  (Implementation fails to meet the specification)

Targetted Areas

- Serial Console input and output
- Network interface input and output
- Time-aware abstractions under consideration

Xest as Student Tool

Test Suite Directory

- Test Case subdirectories
  - Input to be sent over serial console. (Trigger)
  - Expected output. (Response)
    Optionally, open-ended
  - Test-hook code – compiled into embedded system.
    (More advanced feature)

Execute `xest-test`

- Runs test suite.
- Produces itemized PASS/FAIL by test case.
- Output recorded for open-ended tests.
- Diff output available for failed tests.

Example

```
case 'h':
    /* Single sleepy process test. */
    ready(apid, RESCHED_YES);
    printq(sleepq);

    kprintf("proc %d is ", apid);
    if (proctab[apid].state == PRSLEEP)
    { kprintf("PRSLEEP\r\n"); }
    else
    { kprintf("NOT asleep\r\n"); }
    break;
```

Example

Testcase Trigger

```
h
```

Expected Response

```
Enter test case:
[ 1]
proc 1 is PRSLEEP
All user processes have completed.
```

Running an entire suite of tests

```
Test Suite  1: Argument Passing     [PASS]
Test Suite  2: Priority Scheduling  [PASS]
Test Suite  3: Thread Preemption    [PASS]
Test Suite  4: Recursion            [PASS]
Test Suite  5: Single Semaphore     [PASS]
Test Suite  6: Multiple Semaphores  [PASS]
Test Suite  7: Counting Semaphores  [PASS]
Test Suite  8: Killing Semaphores   [PASS]
Test Suite  9: Process Queues       [PASS]
Test Suite 10: Delta Queues         [PASS]
Test Suite 11: Standard I/O         [PASS]
Test Suite 12: TTY Driver           [PASS]
Test Suite 13: Character Types      [PASS]
...
```

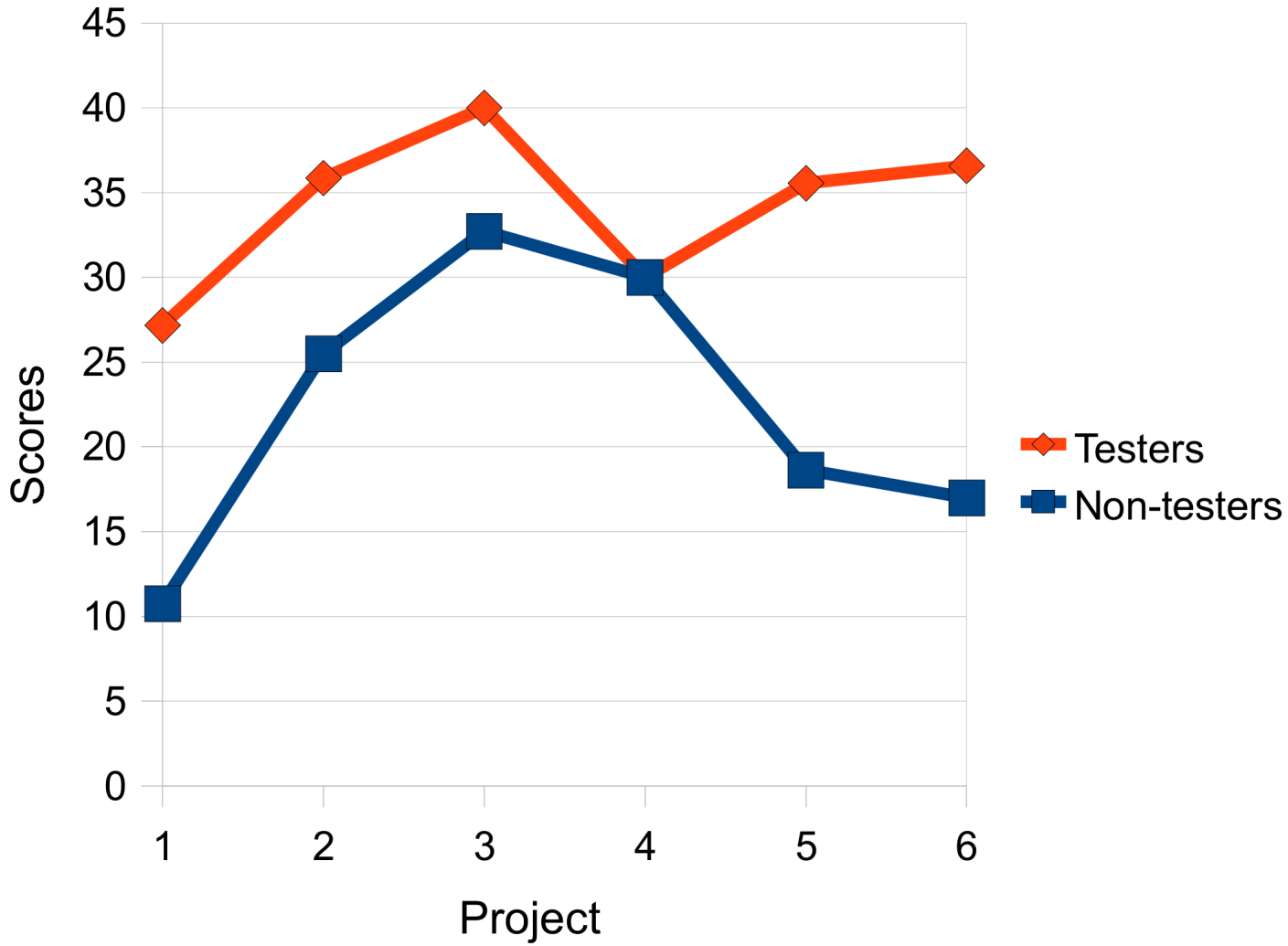
Evaluation

Used to evaluate student submissions in several courses for 4 years. Capabilities refined, generalized.

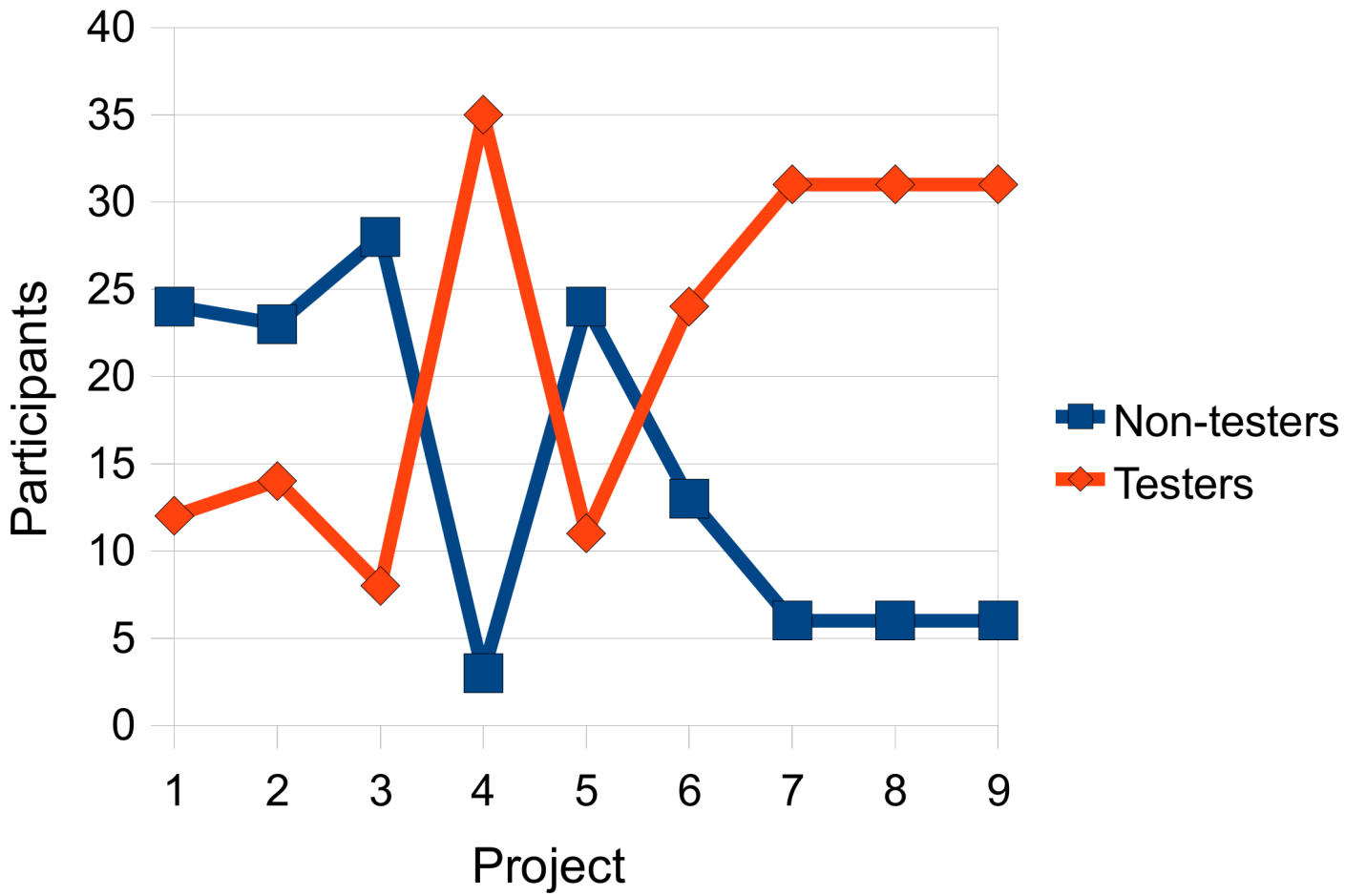
Spring 2010 COSC 3250 / COEN 4820 – Embedded O/S

- Introduced as student testing tool in Lab #1.
- 38 students: Computer Science, Computer Engineering, Biocomputing Engineering.
- CS students have prior testing background (JUnit).
- Engineering students may or may not have prior software testing experience.

Testers vs. Non-Testers – Performance



Testers vs. Non-Testers – Choice



Summary

Embedded Xinu Laboratory

- Flexible, inexpensive, ubiquitous consumer platform,
- Open-source tools that are widely available,
- Curriculum materials, hardware and software tools, community web portal to support adoption,
- More than half a dozen schools have built or are constructing an Embedded Xinu Laboratory.

Summary

Student Usage

- Testers consistently outscored non-testers.
- By end of term, nearly all students opted to embrace some test-driven development.
- Initially only a third were interested in this approach, but vast majority came to adopt early testing when they perceived that their peers were having greater success using this methodology.

Future Work

- Expanded support for time-sensitive testing.
  What abstractions should a testing framework for students provide?
- Continued tracking of students.
  Does a turn toward test-driven development in this course continue in subsequent courses?
- Polish Xest codebase for release.

http://xinu.mscs.mu.edu/