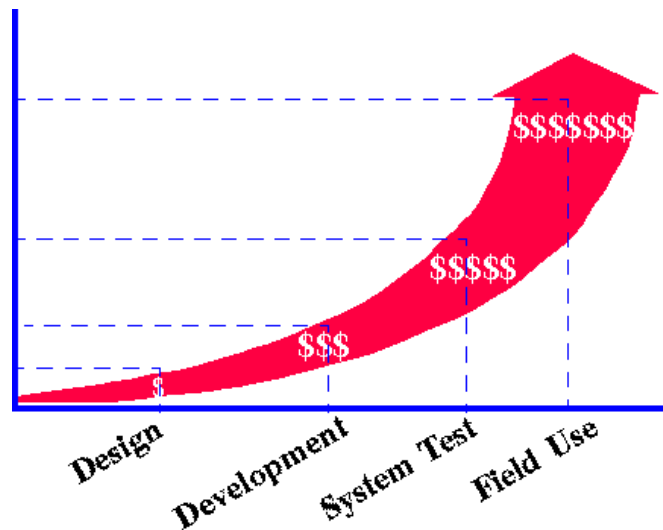


Code Coverage Testing & Tool Support

W. Eric Wong
Department of Computer Science
The University of Texas at Dallas
ewong@utdallas.edu
<http://www.utdallas.edu/~ewong>

Importance of Code Coverage Testing

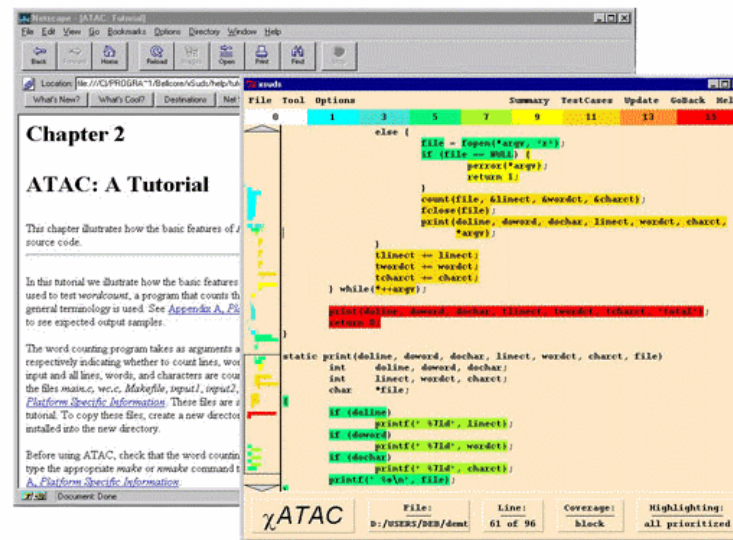
- In general, a piece of code must be executed before a fault in it can be exposed
- *Help early fault detection*
 - Are system testers finding faults that should have been found and fixed by developers?
 - Relative cost of fixing a software fault



χSuds Home Page



Telcordia Software Visualization and Analysis Toolsuite



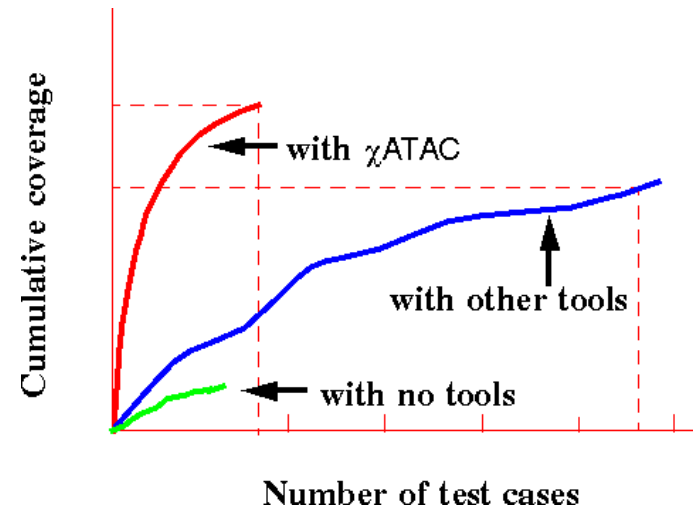
<http://xsuds.arggreenhouse.com>

The χ Suds Tool Suite

- Telcordia Technologies (formerly Bellcore or Bell Communications Research)
 - χ Suds (Software understanding and diagnosis systems): a set of software testing, analysis, and understanding tools for C and C++ programs
 - χ ATAC
 - χ Slice
 - χ Regress
 - χ Vue
 - χ Prof

Efficient Coverage Testing (1)

- How much code is currently tested?
What is missing?
 - Which statements were exercised?
 - Which paths were traversed?
 - Which def-use associations were exercised?
 - Which functions got invoked from where?
- Need help in creating tests?
 - Which statement should I try to cover next?



Analyzing the controlflow graph of the program to find the dominant blocks, decisions, and def-use pairs.

For example, when a test covers highly dominant blocks it will cover many other blocks.

Efficient Coverage Testing (2)

- Use *prioritization* and *visualization* to provide hot spots that give the most value in coverage.
- Each color represents a different weight determined by a control flow analysis using the concept of superblocks and dominators.

The screenshot displays the xATAC code coverage tool interface. At the top, a menu bar includes 'File', 'Tool', 'Options', 'Summary', 'TestCases', 'Update', 'GoBack', and 'Help'. Below the menu is a toolbar with 9 numbered buttons (0-8) color-coded from blue to red. The main area shows a C program with code blocks highlighted in various colors (white, yellow, green, cyan, red) to represent different coverage weights. A sidebar on the left shows a tree view of the code structure. Two callout boxes provide context: a purple box points to white and yellow blocks, stating 'Code in white has already been covered by a test case and covering it again will not add new coverage'; a red box points to a red block, stating 'Covering this red block guarantees the execution of at least 8 additional blocks.' The bottom status bar shows 'xATAC', 'File: cmp2.c', 'Line: 121 of 151', 'Coverage: block', and 'Highlighting: all prioritized'.

```
0 1 2 3 4 5 6 7 8
do ++p; while (isalpha(*p));
break;
case 'o':
    month = 11;
    do ++p; while (isalpha(*p));
    break;
default:
    return -1;
}

value = 0; nDigits = 0;
while (isdigit(*p)) {
    ++nDigits;
    value = value * 10 + *p++ - '0';
}

if (delim == '-') {
    if (value < 1 || value > 31 || nDigits > 2)
        return -1;
    day = value;
} else {
    if (nDigits == 2)
        year = 1900 + value;
    else if (nDigits == 4)
        year = value;
    else return -1;
}

return year * 10000 + month * 100 + day;
}
```

Code in white has already been covered by a test case and covering it again will not add new coverage

Covering this red block guarantees the execution of at least 8 additional blocks.

xATAC File: cmp2.c Line: 121 of 151 Coverage: block Highlighting: all prioritized

Efficient Coverage Testing (3)

The screenshot displays the XATAC tool interface for code coverage testing. The main window shows a C program with several decision points highlighted. A red box with an arrow pointing to the menu bar (specifically to the '8' tab) contains the text: "Covering either *true* or *false* branch guarantees the execution of at least another 8 branches."

```
File Tool Options Summary TestCases Update GoBack Help
0 1 2 3 4 5 6 7 8
do ++p; while (isalpha(*p));
break;
case 'o':
month = 11;
do ++p; while (isalpha(*n));
break;
default:
return -1;
}

value = 0; nDigits = 0;
while (isdigit(*p)) {
++nDigits;
value = value * 10 + *p++ - '0';
}

if (delim == '-' || ...) {
if (value < ... || nDigits > 31 || nDigits < 0) {
return
day = value
} else {
if (nDigits ... ) {
year = 1900 + value;
} else if (nDigits == 4) {
year = value;
} else return -1;
}

return year * 10000 + ... + day;
}
```

Decision points shown in the flow:

- Decision 1: `isalpha(*p)` (true/false/dismiss)
- Decision 2: `isalpha(*n)` (true/false/dismiss)
- Decision 3: `delim == '-' || ...` (true/false/dismiss)
- Decision 4: `value < ... || nDigits > 31 || nDigits < 0` (true/false/dismiss)
- Decision 5: `nDigits ...` (true/false/dismiss)
- Decision 6: `nDigits == 4` (true/false/dismiss)

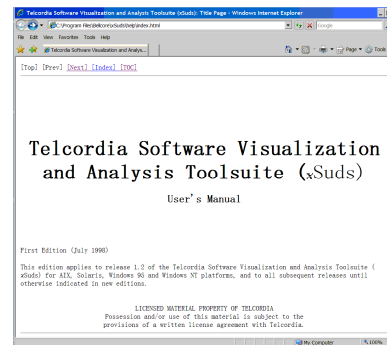
Tool status bar:

- XATAC
- File: cmp2.c
- Line: 121 of 151
- Coverage: decision
- Highlighting: all prioritized

χ ATAC Demo: Coverage Testing of C Code

Compile code with χ ATAC

Initial display



χ Suds User's Manual

Source display after executing wordcount.1

Source display after executing wordcount.2

Block	Coverage
Block 1	35 of 51 (68.6%)
Block 2	8 of 51 (15.7%)
total	41 of 51 (80.4%)

100 % block coverage after executing wordcount.5

File	Decisions	Coverage
*xSuds-Tutorial/wc.c	12 of 12	100%
*xSuds-Tutorial/main.c	19 of 23	82.6%
total	31 of 35	88.6%

Source display after executing wordcount.6

File	Decisions	Coverage
*xSuds-Tutorial/wc.c	12 of 12	100%
*xSuds-Tutorial/main.c	20 of 23	86.9%
total	32 of 35	91.4%

100 % block & decision coverage after executing wordcount.9

Coverage Testing Tools for Java Code

- eXVantage (eXtreme Visual-aid novel testing and generation)
 - A tool suite for code coverage prioritization, test generation, test execution, debugging, and performance profiling of Java, C, and C++ programs
 - Based on the JBT (Java Bytecode Testing) tool suite developed at UTD since 2002
- Clover
- Cobertura
- etc.