Control Flow Testing

Anuruddha Jayasekara Pathiranage

03/05/2015

Control Flow Testing

In structural testing, the software is viewed as a white box and test cases are determined from the implementation of the software. Control-flow testing is a structural testing strategy that uses the program's control flow as a model and this can use the control structure of a program to develop the test cases for the program. The control structure of a program can be represented by the control flow graph of the program. Flow graphs consist of three primitives.

01. A decision is a program point at which the control can diverge.
02. A junction is a program point where the control flow can merge.
03. A process block is a sequence of program statements uninterrupted by either decisions or junctions (Naik & Tripathy, 2008).

There are three testing criteria for control flow testing.

- Path Testing: Execute all possible control flow paths through the program.
- Statement Testing: Execute all statements in a program at least once under some test.
- Branch Testing: Execute enough tests to assure that every branch alternative has been exercised at least once under some test (Ostrand, 2002).

Example of Control Flow Testing

The AbValue program function returns the absolute value of the integer passed to the function as a parameter.

Input: An integer

Output: The absolute value if the input is integer.

Code Line number	Program
1	int AbValue(int y)
2	{

3	If (y < 0)
4	y = - y ;
5	return y;
6	}

Table 01: Program function

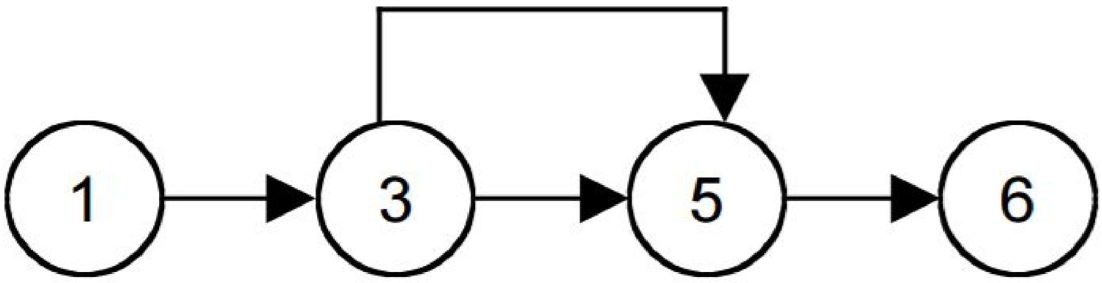

Figure 01: The Flow graph for AbValue

01. Test cases to satisfy path coverage for AbValue function

AbValue takes as its input any integer. There are many integers that could be input to AbValue making it impractical to test all possible inputs to AbValue.

02. Test cases to satisfy statement testing coverage for AbValue function

Paths		Process links				Test cases	
		P	Q	R	S	Input	Output
	PQR	x	x	x		A Negative Integer, y	-y

	PSR	x		x	x	A Positive Integer, y	y

Table 02: Statement testing

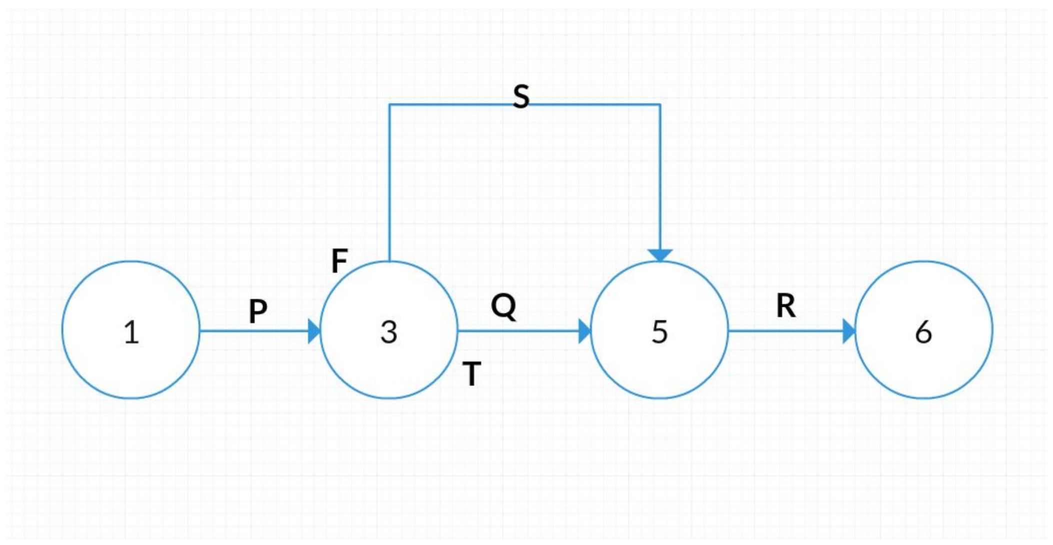

Figure 02: The Flow graph with path

03. Test cases to satisfy Branch Testing coverage for AbValue function

Paths	Decisions	Test Cases	
		Input	Output
PQR	T	A Negative integer, y	-y
PSR	F	A Positive integer, y	y

Table 03: Branch testing

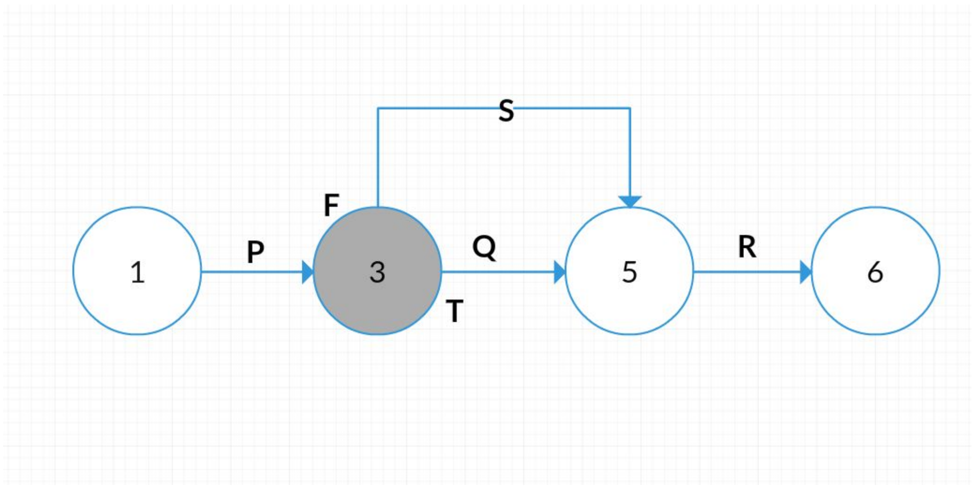

Figure 03: The Flow graph for branch testing (Elodie, 2011).

When Control Flow Testing should be used

Control-flow testing is most applicable to new software for unit testing.

Types of software testing problems that control flow testing addresses

Control-flow testing is more effective for unstructured code than for code that follows structured programming. About 65% of all bugs can be caught in unit testing. Unit testing is dominated by control-flow testing methods.

Key assumptions of control flow testing

01. Specifications are correct

02. Data is defined and accessed properly

03. There are no bugs other than those that affect control flow

Limitations of control flow testing

Control-flow testing as a sole testing technique is limited:

– Interface mismatches and mistakes are not caught.

– Not all initialization mistakes are caught by control-flow testing.

– Specification mistakes are not caught.

References

Naik, K., & Tripathy, P. (2008). Control Flow Testing. In *Software Testing and Quality Assurance* (pp. 88–111). John Wiley & Sons, Inc. https://doi.org/10.1002/9780470382844.ch4

Ostrand, T. (2002). White-Box Testing. In *Encyclopedia of Software Engineering*. John Wiley & Sons, Inc. https://doi.org/10.1002/0471028959.sof378

Elodie, V. (2011). *White Box Coverage and Control Flow Graphs*. June. Retrieved from http://fmt.cs.utwente.nl/~timmer/teaching/WhiteBoxCoverageCFG.pdf