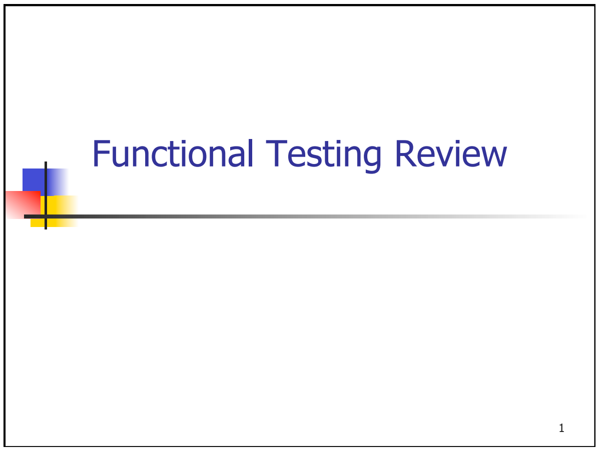
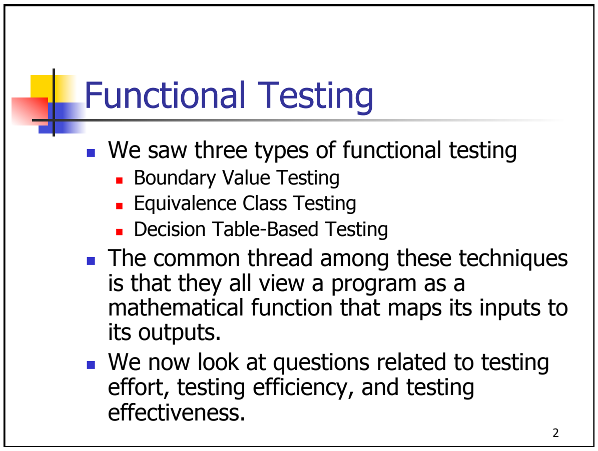
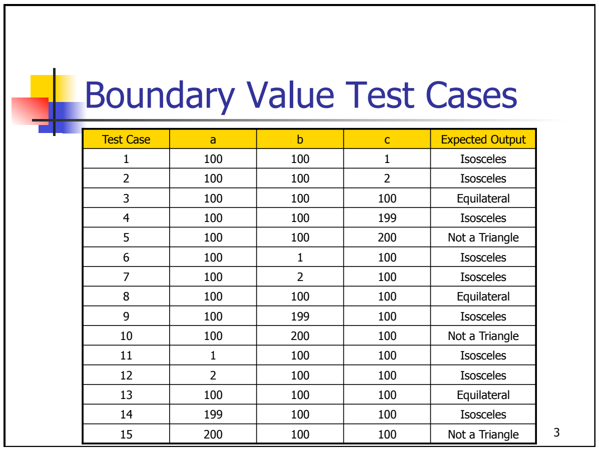


Functional Testing Review



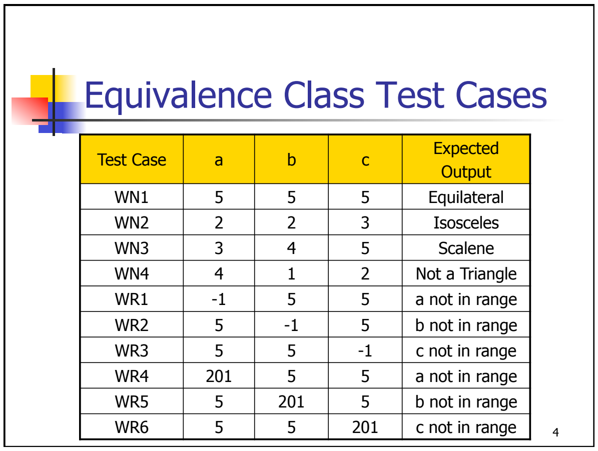
Functional Testing

- We saw three types of functional testing
  - Boundary Value Testing
  - Equivalence Class Testing
  - Decision Table-Based Testing
- The common thread among these techniques is that they all view a program as a mathematical function that maps its inputs to its outputs.
- We now look at questions related to testing effort, testing efficiency, and testing effectiveness.



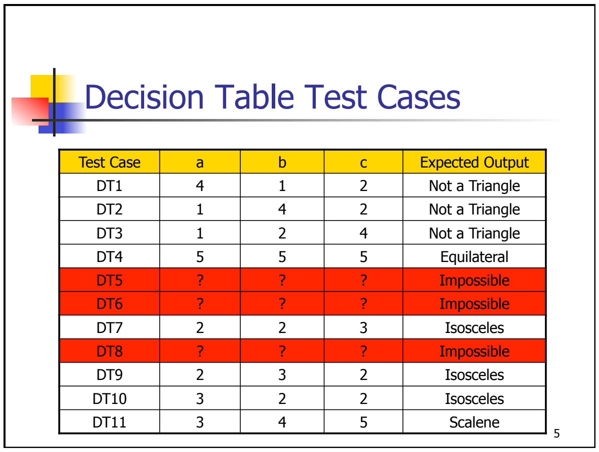
Boundary Value Test Cases

Test Case	a	b	c	Expected Output
1	100	100	1	Isosceles
2	100	100	2	Isosceles
3	100	100	100	Equilateral
4	100	100	199	Isosceles
5	100	100	200	Not a Triangle
6	100	1	100	Isosceles
7	100	2	100	Isosceles
8	100	100	100	Equilateral
9	100	199	100	Isosceles
10	100	200	100	Not a Triangle
11	1	100	100	Isosceles
12	2	100	100	Isosceles
13	100	100	100	Equilateral
14	199	100	100	Isosceles
15	200	100	100	Not a Triangle



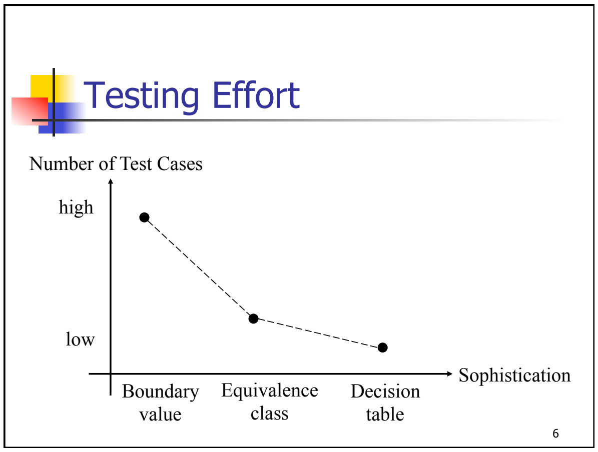
Equivalence Class Test Cases

Test Case	a	b	c	Expected Output
WN1	5	5	5	Equilateral
WN2	2	2	3	Isosceles
WN3	3	4	5	Scalene
WN4	4	1	2	Not a Triangle
WR1	-1	5	5	a not in range
WR2	5	-1	5	b not in range
WR3	5	5	-1	c not in range
WR4	201	5	5	a not in range
WR5	5	201	5	b not in range
WR6	5	5	201	c not in range

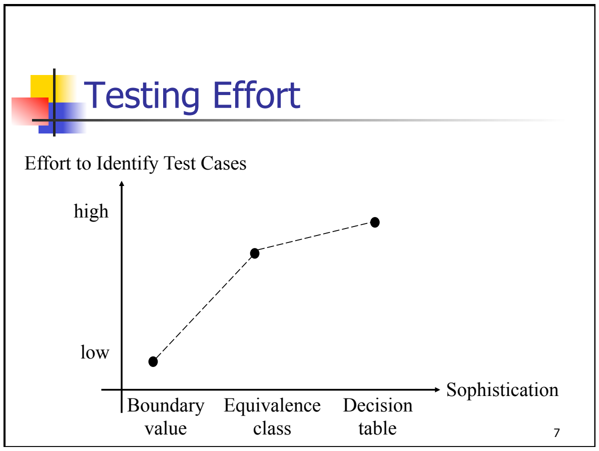


Decision Table Test Cases

Test Case	a	b	c	Expected Output
DT1	4	1	2	Not a Triangle
DT2	1	4	2	Not a Triangle
DT3	1	2	4	Not a Triangle
DT4	5	5	5	Equilateral
DT5	?	?	?	Impossible
DT6	?	?	?	Impossible
DT7	2	2	3	Isosceles
DT8	?	?	?	Impossible
DT9	2	3	2	Isosceles
DT10	3	2	2	Isosceles
DT11	3	4	5	Scalene

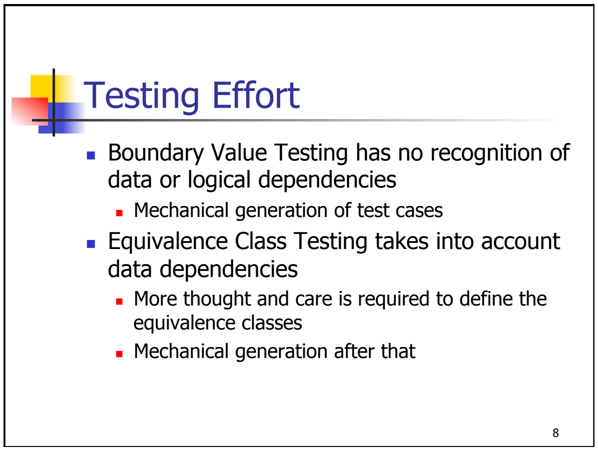


Testing Effort



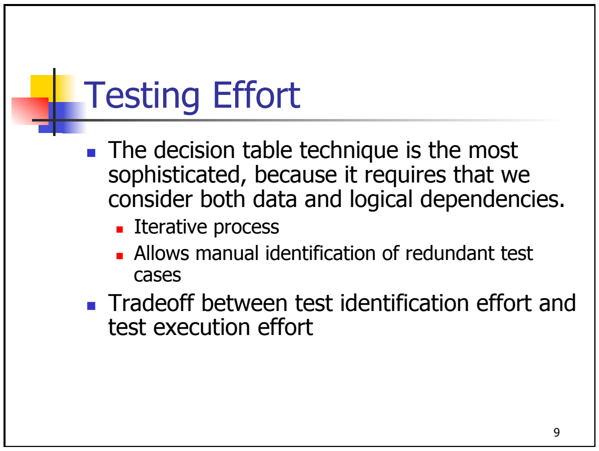
Testing Effort

Effort to Identify Test Cases



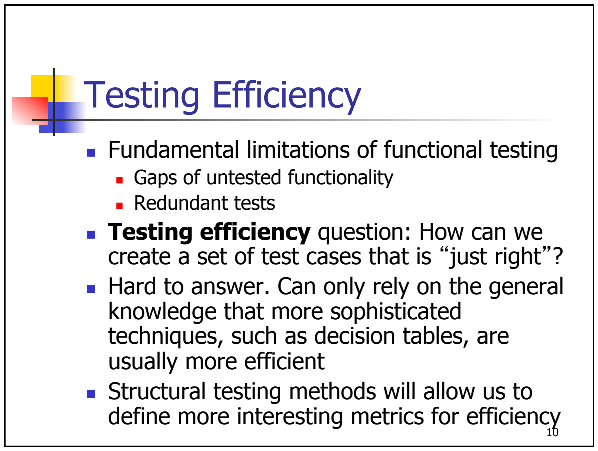
Testing Effort

- Boundary Value Testing has no recognition of data or logical dependencies
  - Mechanical generation of test cases
- Equivalence Class Testing takes into account data dependencies
  - More thought and care is required to define the equivalence classes
  - Mechanical generation after that



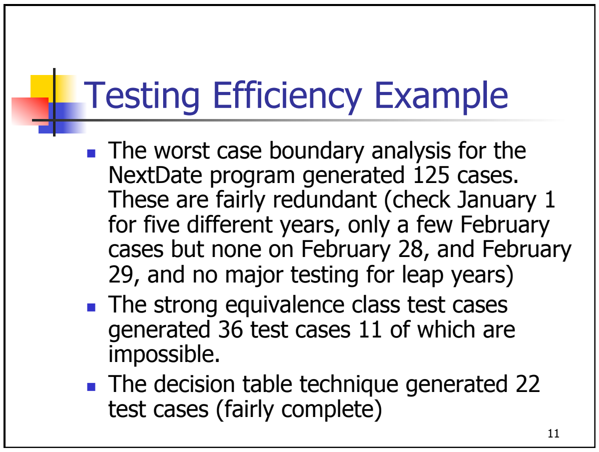
Testing Effort

- The decision table technique is the most sophisticated, because it requires that we consider both data and logical dependencies.
  - Iterative process
  - Allows manual identification of redundant test cases
- Tradeoff between test identification effort and test execution effort



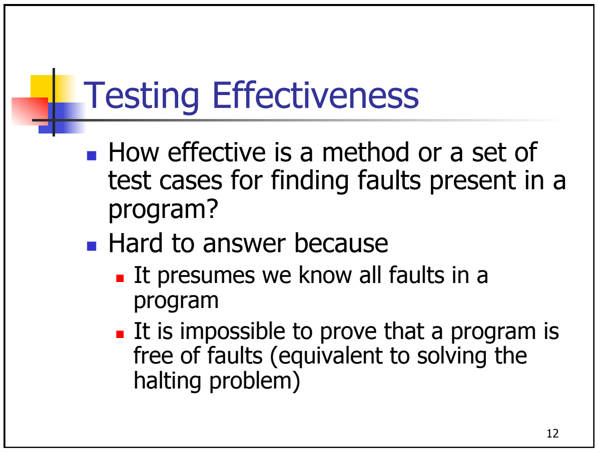
Testing Efficiency

- Fundamental limitations of functional testing
  - Gaps of untested functionality
  - Redundant tests
- **Testing efficiency** question: How can we create a set of test cases that is “just right”?
- Hard to answer. Can only rely on the general knowledge that more sophisticated techniques, such as decision tables, are usually more efficient
- Structural testing methods will allow us to define more interesting metrics for efficiency



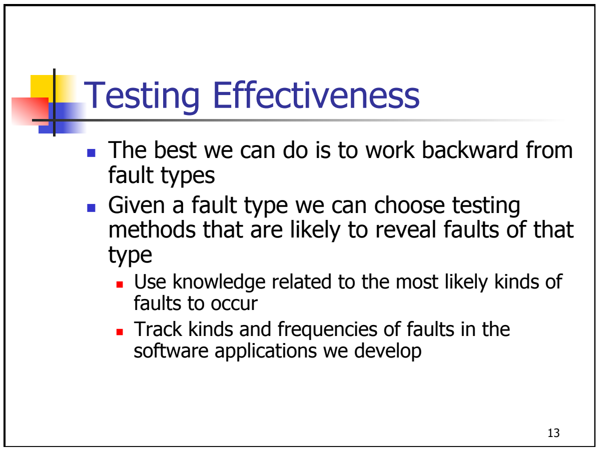
Testing Efficiency Example

- The worst case boundary analysis for the NextDate program generated 125 cases. These are fairly redundant (check January 1 for five different years, only a few February cases but none on February 28, and February 29, and no major testing for leap years)
- The strong equivalence class test cases generated 36 test cases 11 of which are impossible.
- The decision table technique generated 22 test cases (fairly complete)



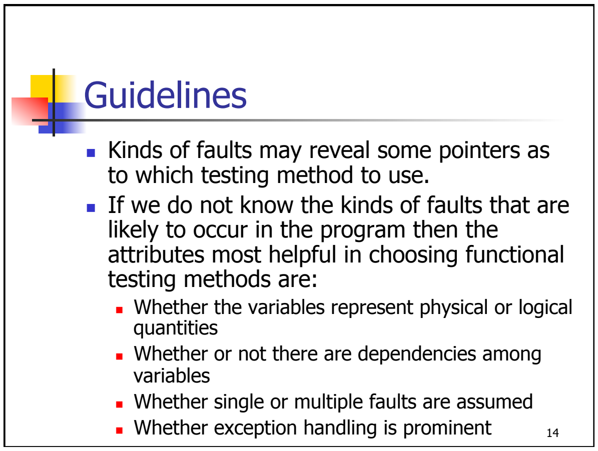
Testing Effectiveness

- How effective is a method or a set of test cases for finding faults present in a program?
- Hard to answer because
  - It presumes we know all faults in a program
  - It is impossible to prove that a program is free of faults (equivalent to solving the halting problem)



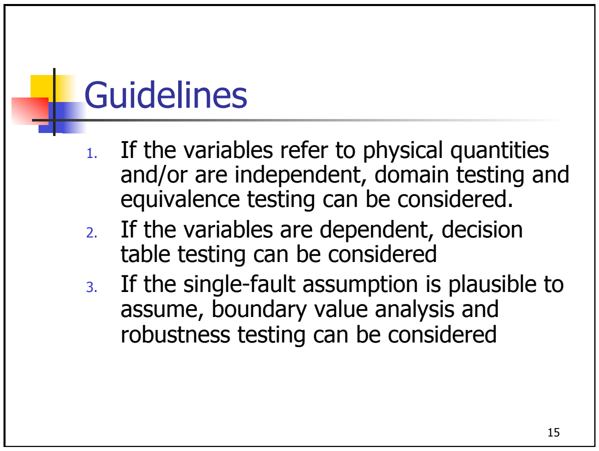
Testing Effectiveness

- The best we can do is to work backward from fault types
- Given a fault type we can choose testing methods that are likely to reveal faults of that type
  - Use knowledge related to the most likely kinds of faults to occur
  - Track kinds and frequencies of faults in the software applications we develop



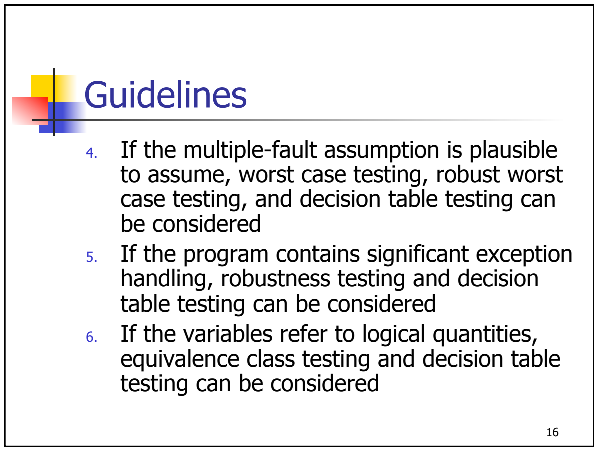
Guidelines

- Kinds of faults may reveal some pointers as to which testing method to use.
- If we do not know the kinds of faults that are likely to occur in the program then the attributes most helpful in choosing functional testing methods are:
  - Whether the variables represent physical or logical quantities
  - Whether or not there are dependencies among variables
  - Whether single or multiple faults are assumed
  - Whether exception handling is prominent



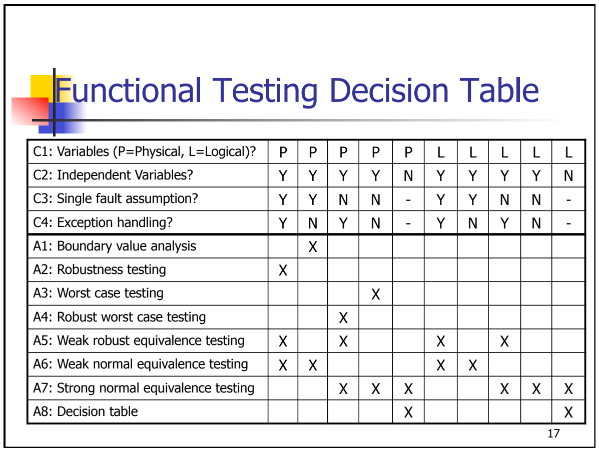
Guidelines

1. If the variables refer to physical quantities and/or are independent, domain testing and equivalence testing can be considered.
2. If the variables are dependent, decision table testing can be considered
3. If the single-fault assumption is plausible to assume, boundary value analysis and robustness testing can be considered



Guidelines

4. If the multiple-fault assumption is plausible to assume, worst case testing, robust worst case testing, and decision table testing can be considered
5. If the program contains significant exception handling, robustness testing and decision table testing can be considered
6. If the variables refer to logical quantities, equivalence class testing and decision table testing can be considered



Functional Testing Decision Table

C1: Variables (P=Physical, L=Logical)?	P	P	P	P	P	L	L	L	L	L
C2: Independent Variables?	Y	Y	Y	Y	N	Y	Y	Y	Y	N
C3: Single fault assumption?	Y	Y	N	N	-	Y	Y	N	N	-
C4: Exception handling?	Y	N	Y	N	-	Y	N	Y	N	-
A1: Boundary value analysis		X								
A2: Robustness testing	X									
A3: Worst case testing				X						
A4: Robust worst case testing			X							
A5: Weak robust equivalence testing	X		X			X		X		
A6: Weak normal equivalence testing	X	X				X	X			
A7: Strong normal equivalence testing			X	X	X			X	X	X
A8: Decision table					X					X