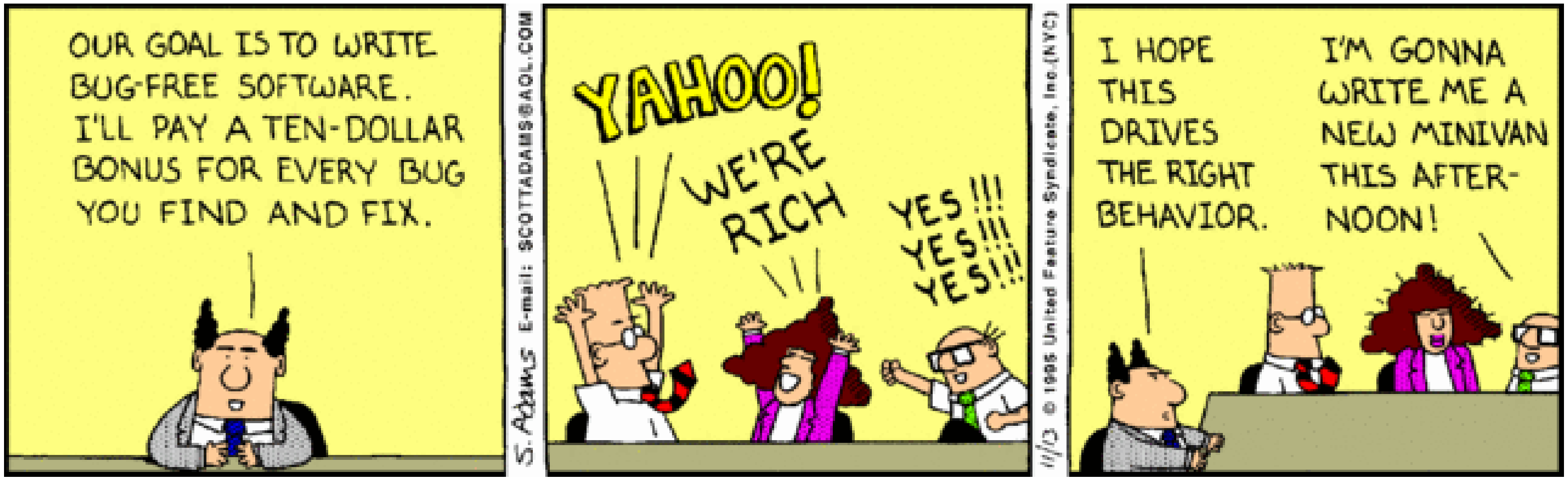
Selecting Structural Test

Get the Right Incentives


OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!
S. Adams E-mail: SCOTTADAMS@AOL.COM
11/13 © 1995 United Feature Syndicate, Inc.(NYC)

Objectives

- To understand program flow graphs
- Present some additional white box selection selection approaches
- To practice white box test case selection

Binary Search (C++)

- Replace with portrait slide

Control and Data-driven Programs

```
case A is
    when “One” => i := 1 ;
    when “Two” => i := 2 ;
    when “Three” => i := 3 ;
    when “Four” => i := 4 ;
    when “Five” => i := 5 ;
end case ;
```

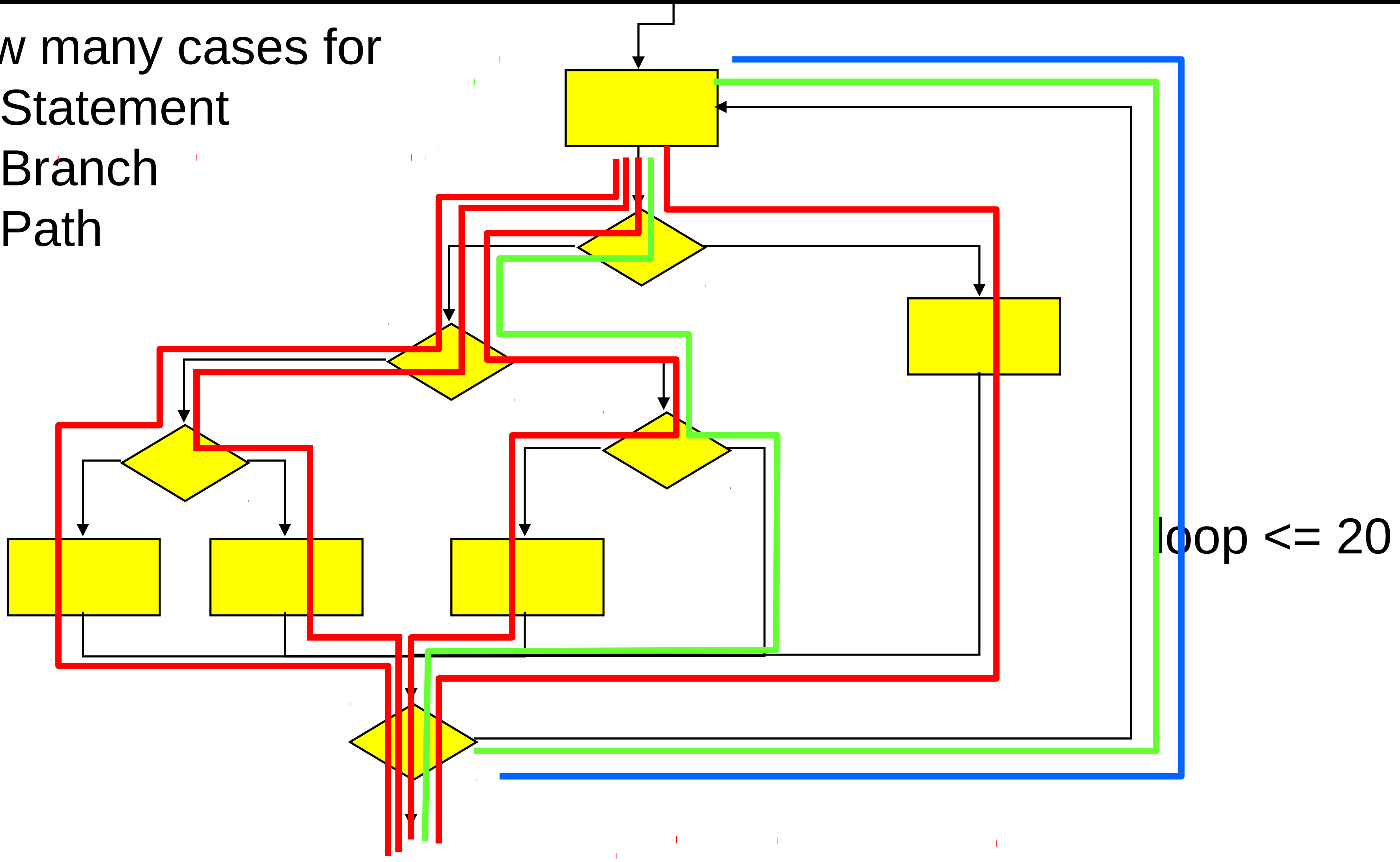
```
Strings: array (1..5) of STRING
:=
    (“One”, “Two”, “Three”,
“Four”, “Five”);
 i := 1 ;
loop
   exit when Strings (i) = A ;
   i := i + 1 ;
end loop ;
```

DOGBERT THE CONSULTANT
A GOOD WAY TO JUDGE CORPORATE HEALTH IS TO LOOK AT YOUR EMPLOYEE TURNOVER RATE.
OUR TURNOVER RATE IS VERY LOW. WE ONLY HIRE PEOPLE WHO AREN'T SKILLED ENOUGH TO WORK ANYPLACE ELSE.
MAYBE METRICS AREN'T THE WAY TO GO HERE.
NO METRIC HAS BEATEN ME YET!!
S. Adams
E-Mail: SCOTTADAMS@AOL.COM
© 1996 United Feature Syndicate, Inc.

Path Testing

How many cases for

- Statement
- Branch
- Path



Path Testing

- Path coverage requires:
  - **3,656,158,440,062,976** test cases

- If you run 1000 tests per second, this will take **116,000 years**.

How About Loops?

- Simple loops
  - Skip loop entirely
  - Only one pass through the loop
  - Two passes through the loop
  - m passes where $m < n$
  - $(n-1)$, n, and $(n+1)$ passes

  - Where n is the max allowed passes through the loop

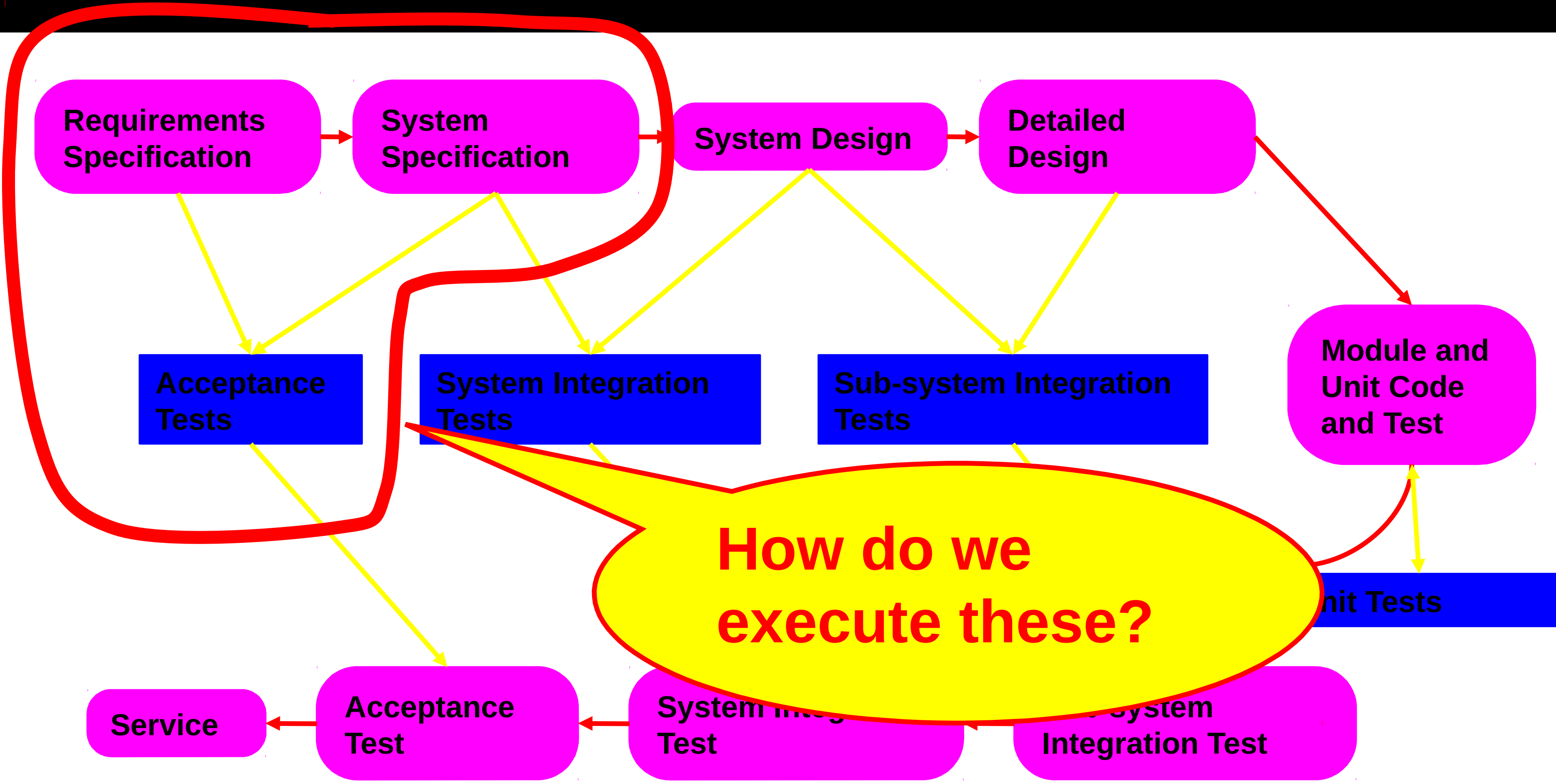
Nested and Concatenated Loops

- Nested
  - Test innermost loop first with all outer loops at the minimum value
  - Move one loop out, keep the inner loop at “typical” values, and test it as the previous step
  - Continue until outermost loop tested
- Concatenated loops
  - Independent and can be tested independently
  - Most of the time……

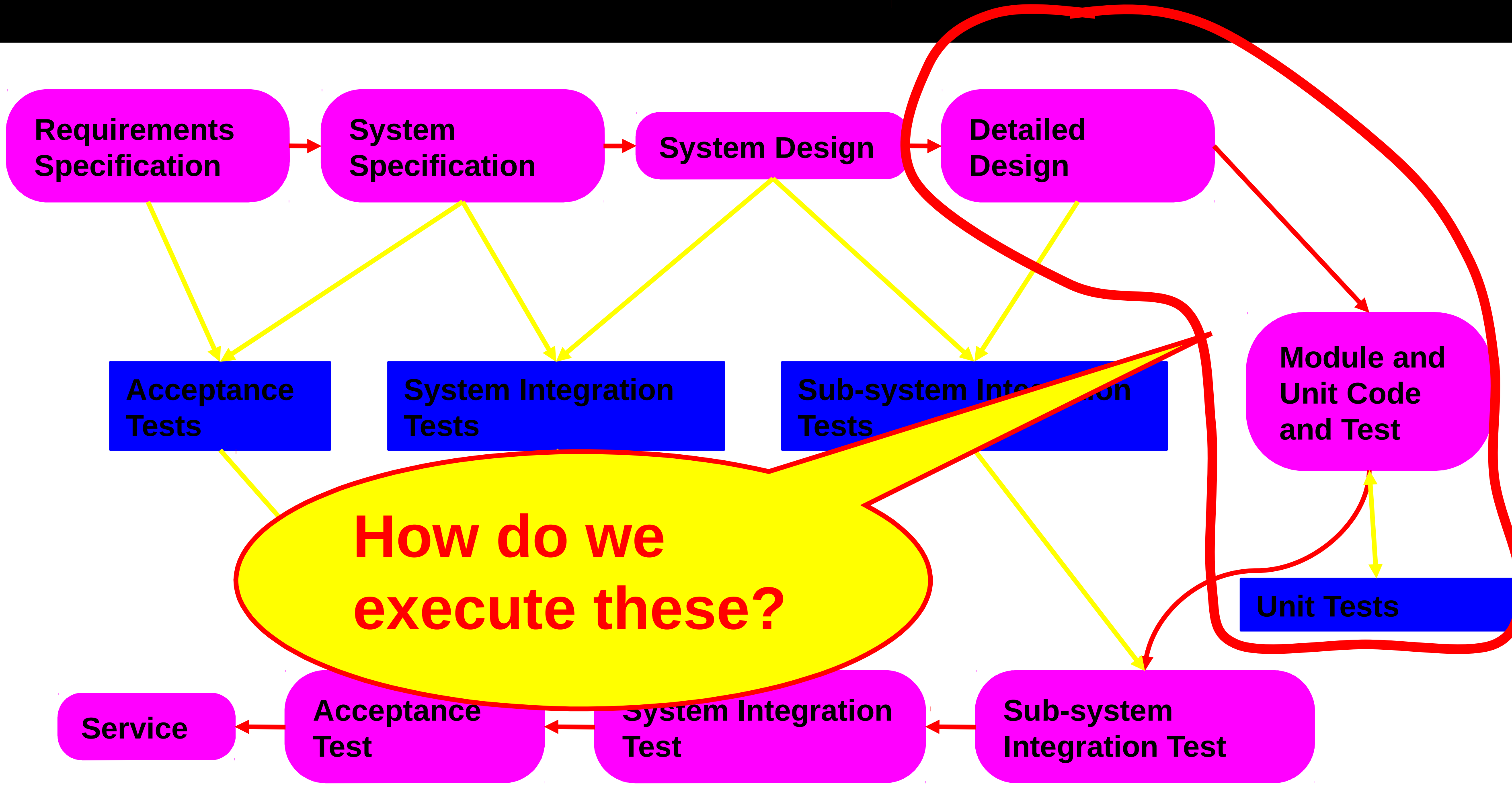
Interface Testing

- Takes place when modules or sub-systems are integrated to create larger systems
- Objectives are to detect faults due to interface errors or invalid assumptions about interfaces
- Particularly important for object-oriented development as objects are defined by their interfaces

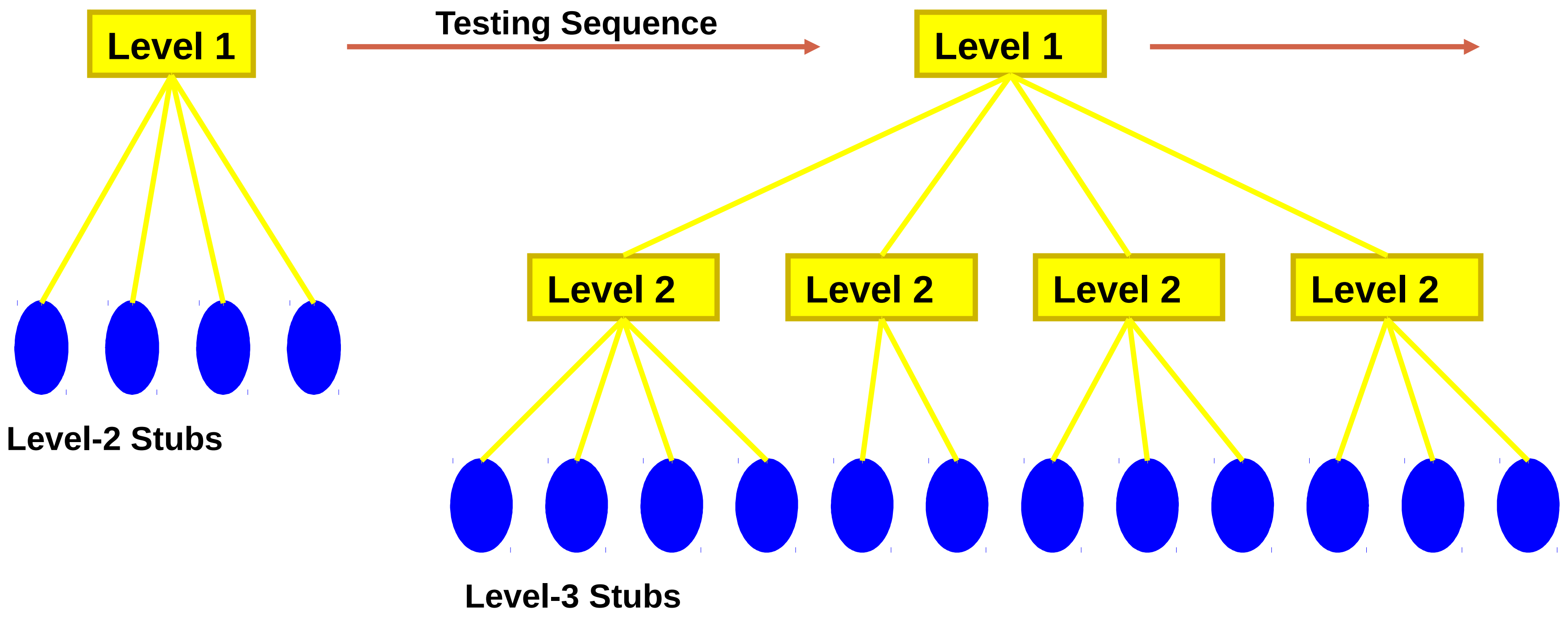
The V-Model of Development



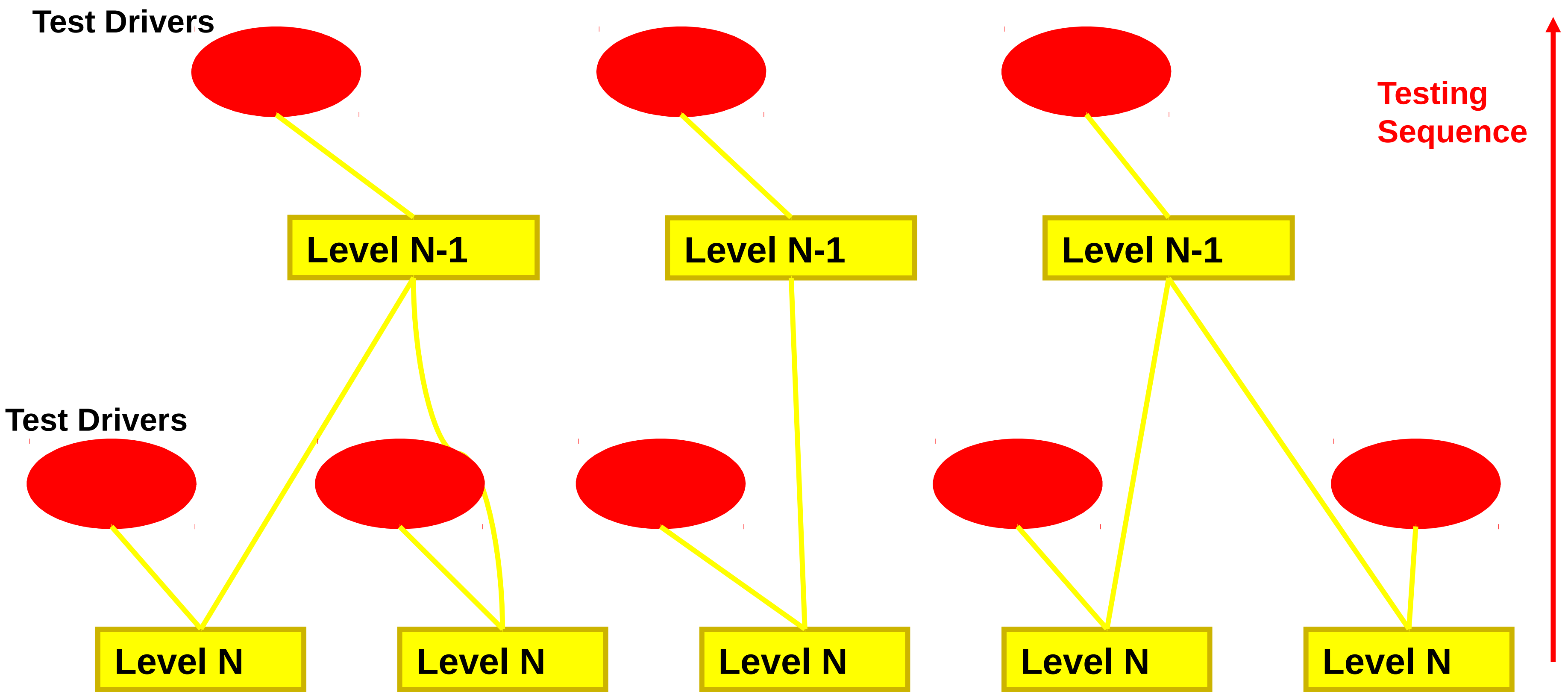
The V-Model of Development



Top-down testing



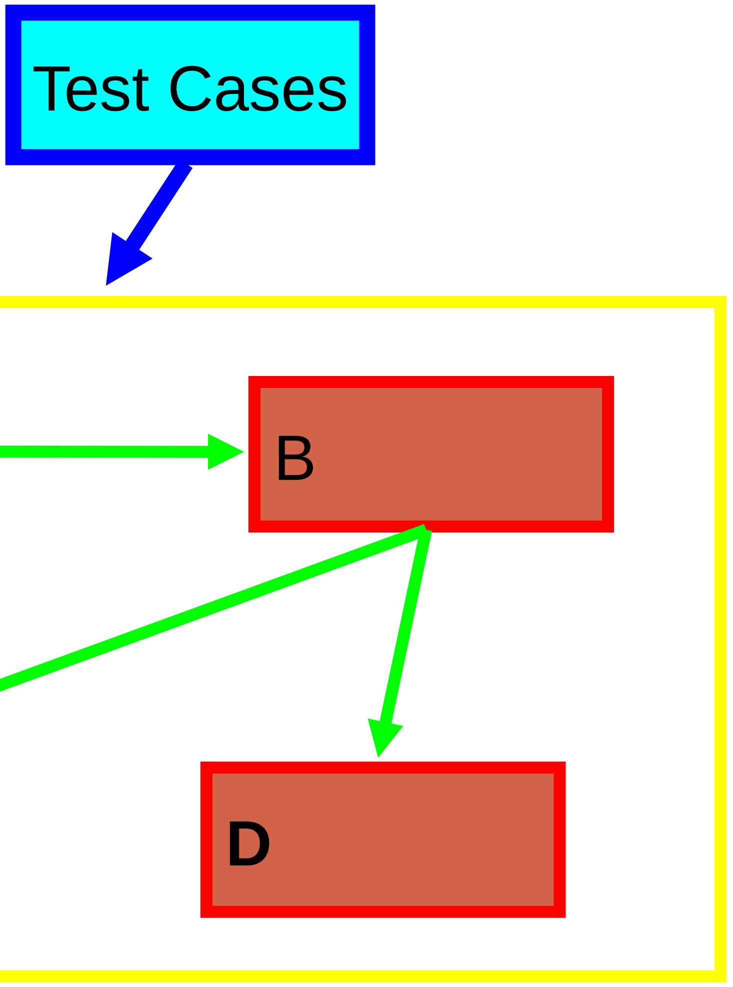
Bottom-Up Testing



Interfaces Types

- Parameter interfaces
  - Data passed from one procedure to another
- Shared memory interfaces
  - Block of memory is shared between procedures
- Procedural interfaces
  - Sub-system encapsulates a set of procedures to be called by other sub-systems
- Message passing interfaces
  - Sub-systems request services from other sub-systems

Interface Testing



Interface Errors

- Interface misuse
  - A calling component calls another component and makes an error in its use of its interface
  - e.g., parameters in the wrong order
- Interface misunderstanding
  - A calling component embeds assumptions about the behavior of the called component that are incorrect
- Timing errors
  - The called and the calling component operate at different speeds and out-of-date information is accessed

Interface Testing Guidelines

- Design tests so that parameters to a called procedure are at the extreme ends of their ranges
- Always test pointer parameters with null pointers
- Design tests which cause the component to fail
- Use stress testing in message passing systems
- In shared memory systems, vary the order in which components are activated

We Have Learned

- Test Coverage Measures
  - Statement, branch, and path coverage
  - Condition coverage (basic, compound)
  - Data flow coverage
- Test coverage measures ensure that statements have been executed to some level
  - However, it is not possible to exercise all combinations