Unit Testing Framework for Tcl

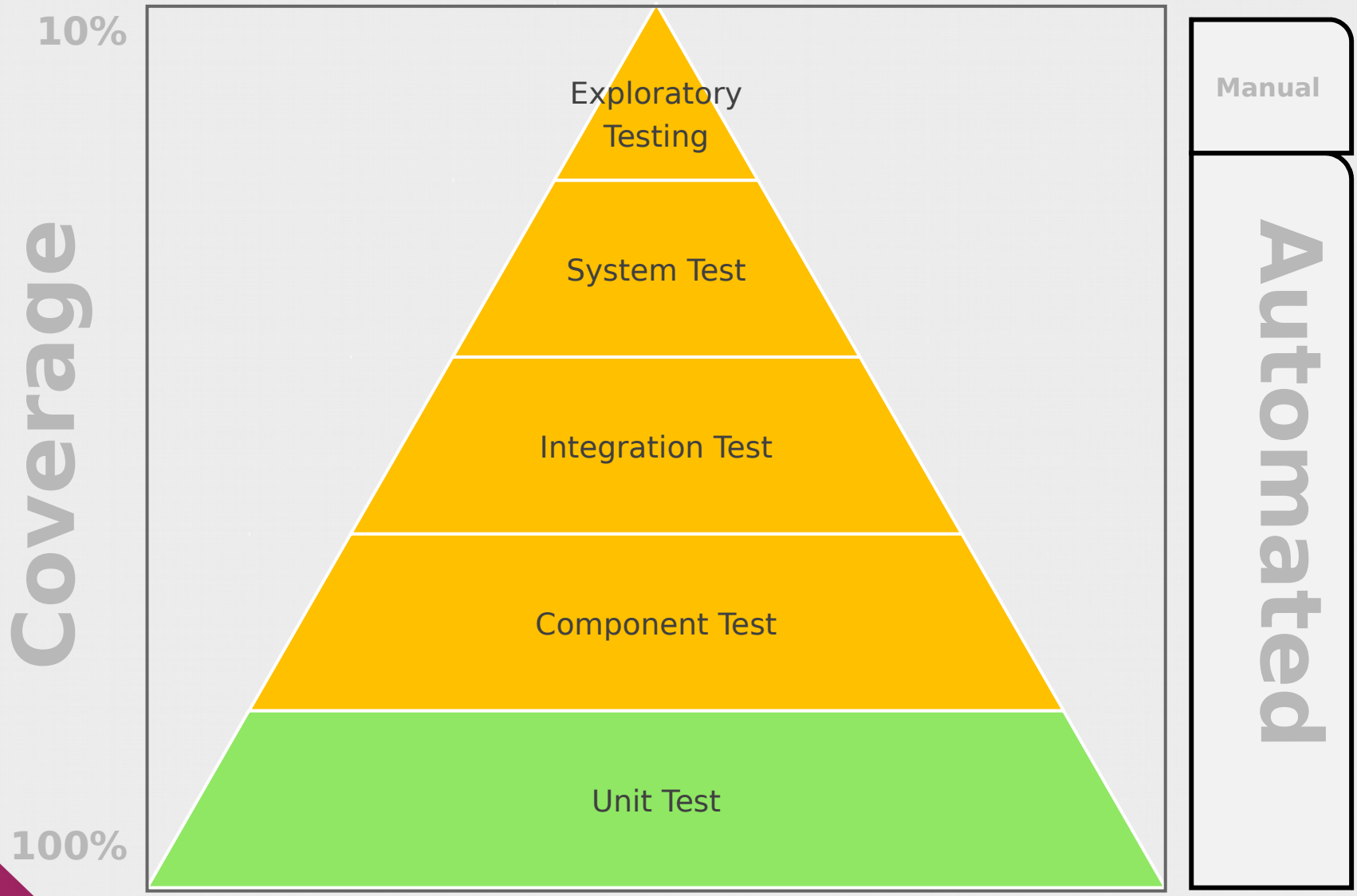
Unit Testing Framework for Tcl

- What is Unit Testing?
- Why Unit Testing?
- Why do we need/want a framework?
- The TIP?
- Where to from here?

What is Unit Testing?

- Also know as “White Box Testing”
  - Test the smallest units of the software.
    - Procs and Methods
  - Test all paths through the unit.
  - Test in isolation:
    - Assumes no known set of data in a database.
    - Known state.
    - Controlled reaction.

Why Unit Test?



Why Unit Test

- Customer Satisfaction
  - Customers do not like bugs!
    - Will pay more for a product with less bugs.
    - Costs money when functions are not available.
    - Could lead to audits and/or government fines to the customer.
- Total Lifecycle Cost
  - Bugs cost money
    - To find and repair – and have negative impact on sales and/or the cost of sales.
    - The later bugs are found, the more costly to find and fix.
    - Cuts into funds available for ...

The TIP - #452

- This TIP proposes an enhancement to the tcltest package to add support for easy creation of test stubs, mocks and seams.
  - The tcltest package allows for automated testing of Tcl code. However, doing proper automated unit testing requires that the unit under test (i.e., the method or procedure) not invoke the actual implementation of other units, but rather should invoke stub or mock units that are under the control of the test being performed as to the results they return and any exceptions they raise.
  - This TIP adds support for building these mechanisms, making it significantly easier to create isolated unit tests of Tcl code.

The TIP - #452

- The following commands are added:
  - ::tcltest::TestSetup - Defines which procedures/commands are stubbed out and how they should behave for each invocation.
    - This should only be called once per test.
  - ::tcltest::SaveVars - Saves the values of variables to be restored later.
    - This should only be called once per test.
  - ::tcltest::AddStub - Adds a procedures/commands to the list that are stubbed out.
  - ::tcltest::AddVars - Add a variable to the list of variables to be restored later.

The TIP - #452

- When defining a stub it takes
  - procName procData
    - procData consit of duples:
      - invocationNumber behaviorDict
        - The invocationNumber is a positive number, defining the behaviour of a given invocation, or an asterisk ("*") defining the behavior for an invocation

The TIP - #452

- behaviorDict may have any of the following key value pairs:
  - use - this specifies what routine is to handle this invocation.
    - Defaults to "standard". It must be one of the following:
      - standard - the standard stub processing is to be done.
      - actual - use the actual implementation.
      - prefix - use the prefix specified as a value.
  - returns - the value of which is returned from the stub, defaults to {}.
  - code - the value given to the -code option on return, defaults to "ok".
  - errorcode - the value given to the -errorcode option if the code is not "ok".
  - set - a list of triplets specifing variables to be modified and their values.
    - This is only done for a code of "ok". The triplets are as follows: varName varType value
      - Where:
        - varName - the name of the variable
        - varType - an "S" for a scalar variable and an "A" for an array
        - value - For a scalar, the value. For an array, a list suitable to be used by [array set]

The TIP - #452

- The following commands are added:
  - ::tcltest::CallCount - Returns a dictionary sorted list of the stubbed out procedures and how many times they were called.
  - ::tcltest::TestCleanup - Restores saved variables and stubbed out procedures.
  - ::tcltest::SortedArrayData - Return the values of an array as a list of key-value pairs sorted by the keys.

The TIP - #452

- The following commands are added:
  - ::tcltest::CallProc - Call the real implementation of a stubbed out procedure.
  - ::tcltest::Seam - Test seam definition and injection (aka enabling).
    - This command is available without requiring the tcltest package – but only *define*.
      - Actions:
        - define seamName body
        - activiate seamName body
        - deactivate seamName

The TIP - #452

- The following test are added:
  - Test of http package utilizing stubs.

Where to from here?

- Is this the correct direction?
- What should we do for TclOO?