TESTING TOOL CLASSIFICATION

Requirements testing tools

Static analysis tools

Test design tools

Test data preparation tools

Test running tools - character-based, GUI

Comparison tools

Test harnesses and drivers

Performance test tools

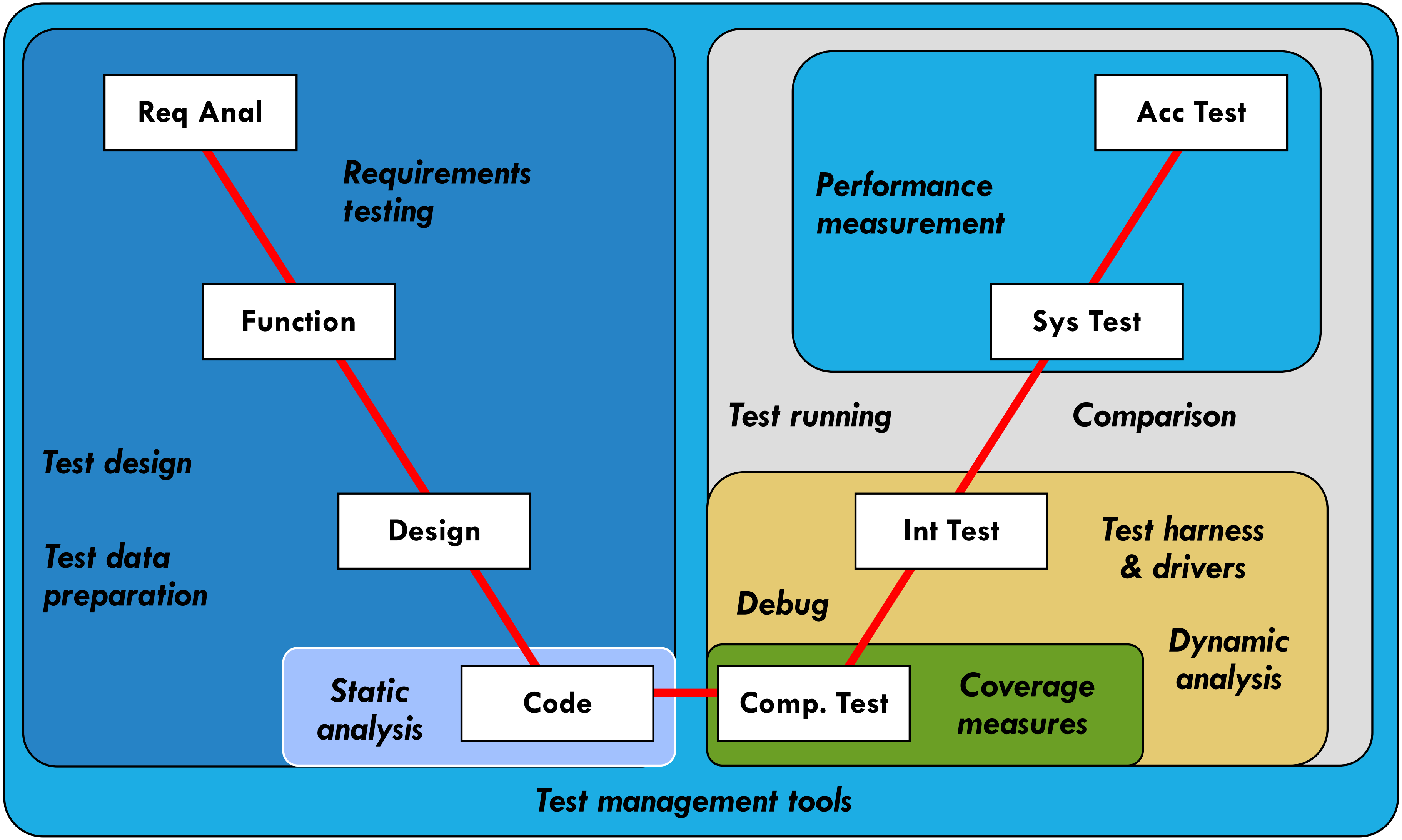
Dynamic analysis tools

Debugging tools

Test management tools

Coverage measurement

WHERE TOOLS FIT



REQUIREMENTS TESTING TOOLS

Automated support for verification and validation of requirements models

- consistency checking
- animation

STATIC ANALYSIS TOOLS

Provide information about the quality of software

Code is examined, not executed

Objective measures

- cyclomatic complexity
- others: nesting levels, size

TEST DESIGN TOOLS

Generate test inputs

- from a formal specification or CASE repository
- from code (e.g. code not covered yet)

TEST DATA PREPARATION TOOLS

Data manipulation

- selected from existing databases or files
- created according to some rules
- edited from other sources

TEST RUNNING TOOLS 1

Interface to the software being tested

Run tests as though run by a human tester

Test scripts in a programmable language

Data, test inputs and expected results held in test repositories

Most often used to automate regression testing

TEST RUNNING TOOLS 2

Character-based

- simulates user interaction from dumb terminals
- capture keystrokes and screen responses

GUI (Graphical User Interface)

- simulates user interaction for WIMP applications (Windows, Icons, Mouse, Pointer)
- capture mouse movement, button clicks, and keyboard inputs
- capture screens, bitmaps, characters, object states

COMPARISON TOOLS

Detect differences between actual test results and expected results

- screens, characters, bitmaps
- masking and filtering

Test running tools normally include comparison capability

Stand-alone comparison tools for files or databases

TEST HARNESSES AND DRIVERS

Used to exercise software which does not have a user interface (yet)

Used to run groups of automated tests or comparisons

Often custom-build

Simulators (where testing in real environment would be too costly or dangerous)

PERFORMANCE TESTING TOOLS

Load generation

- drive application via user interface or test harness
- simulates realistic load on the system & logs the number of transactions

Transaction measurement

- response times for selected transactions via user interface

Reports based on logs, graphs of load versus response times

DYNAMIC ANALYSIS TOOLS

Provide run-time information on software (while tests are run)

- allocation, use and de-allocation of resources, e.g. memory leaks
- flag unassigned pointers or pointer arithmetic faults

DEBUGGING TOOLS

Used by programmers when investigating, fixing and testing faults

Used to reproduce faults and examine program execution in detail

- single-stepping
- breakpoints or watchpoints at any statement
- examine contents of variables and other data

TEST MANAGEMENT TOOLS

Management of testware: test plans, specifications, results

Project management of the test process, e.g. estimation, schedule tests, log results

Incident management tools (may include workflow facilities to track allocation, correction and retesting)

Traceability (of tests to requirements, designs)

COVERAGE MEASUREMENT TOOLS

Objective measure of what parts of the software structure was executed by tests

Code is instrumented in a static analysis pass

Tests are run through the instrumented code

Tool reports what has and has not been covered by those tests, line by line and summary statistics

Different types of coverage: statement, branch, condition, LCSAJ, et al

ADVANTAGES OF RECORDING MANUAL TESTS

documents what the tester actually did

- useful for capturing ad hoc tests (e.g. end users)
- may enable software failures to be reproduced

produces a detailed “script”

- records actual inputs
- can be used by a technical person to implement a more maintainable automated test

ideal for one-off tasks

- such as long or complicated data entry

CAPTURED TEST SCRIPTS

will not be very understandable

- it is a programming language after all!
- during maintenance will need to know more than can ever be ‘automatically commented’

will not be resilient to many software changes

- a simple interface change can impact many scripts

do not include verification

- may be easy to add a few simple screen based comparisons

AUTOMATED VERIFICATION

there are many choices to be made

- dynamic / post execution, compare lots / compare little, resilience to change / bug finding effective

scripts can soon become very complex

- more susceptible to change, harder to maintain

there is a lot of work involved

- speed and accuracy of tool use is very important

usually there is more verification that can
(and perhaps should) be done

- automation can lead to better testing (not guaranteed!)

EFFORT TO AUTOMATE

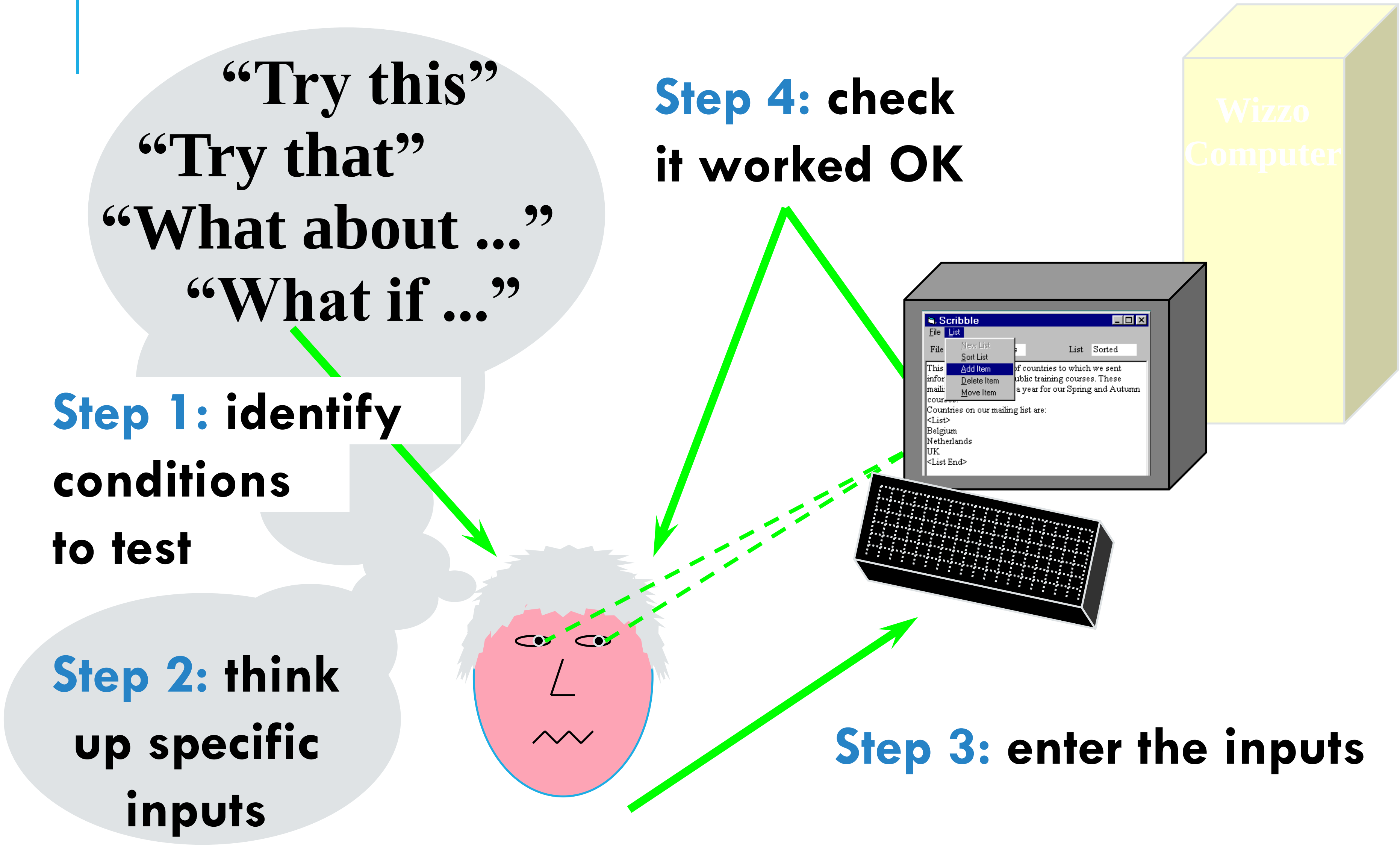
The effort required to automate any one test varies greatly

- typically between 2 and 10 times the manual test effort

and depends on:

- tool, skills, environment and software under test
- existing manual test process which may be:
  - unscripted manual testing
  - scripted (vague) manual testing
  - scripted (detailed) manual testing

UNSCRIPTED MANUAL TESTING



"Try this"
"Try that"
"What about ..."
"What if ..."

Step 1: identify conditions to test

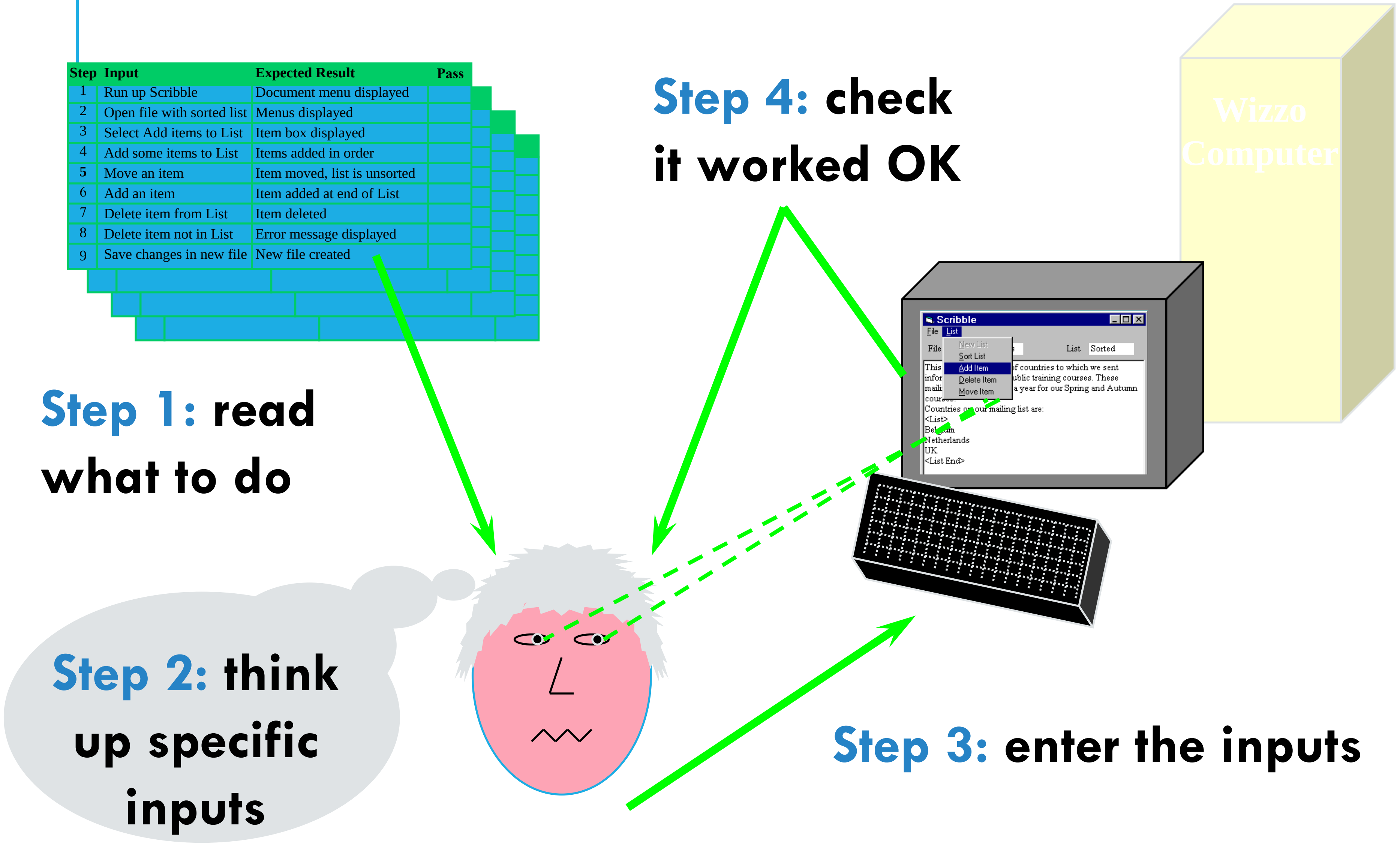
Step 2: think up specific inputs

Step 3: enter the inputs

Step 4: check it worked OK

This of countries to which we sent
infor ublic training courses. These
maili a year for our Spring and Autumn
cour
Countries on our mailing list are:
<List>
Belgium
Netherlands
UK
<List End>

SCRIPTED (VAGUE) MANUAL TESTING



Step	Input	Expected Result	Pass
1	Run up Scribble	Document menu displayed	
2	Open file with sorted list	Menus displayed	
3	Select Add items to List	Item box displayed	
4	Add some items to List	Items added in order	
5	Move an item	Item moved, list is unsorted	
6	Add an item	Item added at end of List	
7	Delete item from List	Item deleted	
8	Delete item not in List	Error message displayed	
9	Save changes in new file	New file created	

Step 1: read what to do

Step 2: think up specific inputs

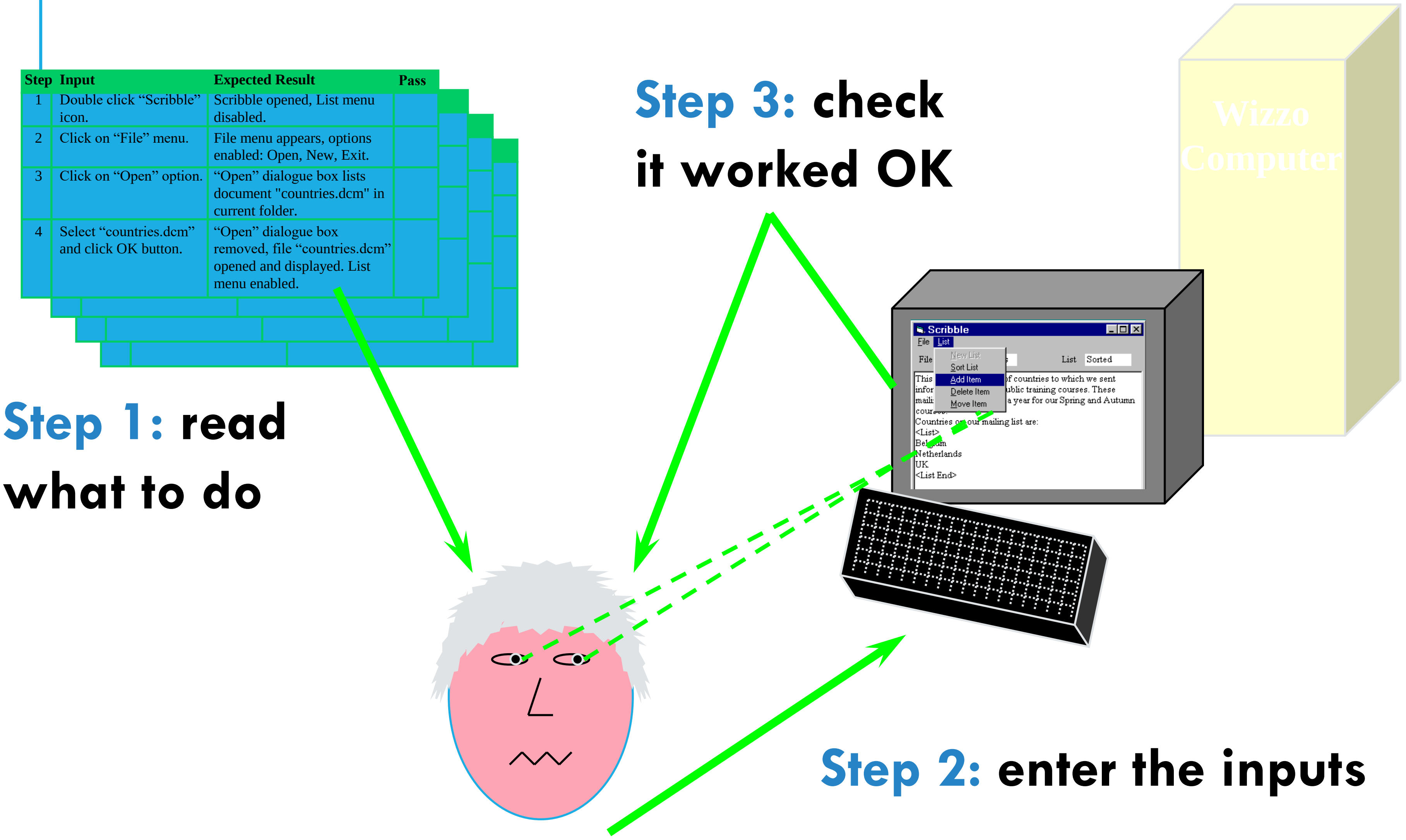
Step 3: enter the inputs

Step 4: check it worked OK

A VAGUE MANUAL TEST SCRIPT

Step	Input	Expected Result	Pass
1	Run up Scribble	Document menu displayed	
2	Open file with sorted list	Menus displayed	
3	Select Add items to List	Item box displayed	
4	Add two items to List	Items added in order	
5	Move an item	Item moved, list is unsorted	
6	Add an item	Item added at end of List	
7	Delete item from List	Item deleted	
8	Delete item not in List	Error message displayed	
9	Save changes in new file	New file created	

SCRIPTED (DETAILED) MANUAL TESTING



Step	Input	Expected Result	Pass
1	Double click “Scribble” icon.	Scribble opened, List menu disabled.	
2	Click on “File” menu.	File menu appears, options enabled: Open, New, Exit.	
3	Click on “Open” option.	“Open” dialogue box lists document "countries.dcm" in current folder.	
4	Select “countries.dcm” and click OK button.	“Open” dialogue box removed, file “countries.dcm” opened and displayed. List menu enabled.	

Step 1: read what to do

Step 2: enter the inputs

Step 3: check it worked OK

This of countries to which we sent
infor ublic training courses. These
maili a year for our Spring and Autumn
cour
Countries on our mailing list are:
<List>
Belgium
Netherlands
UK
<List End>

DON'T AUTOMATE TOO MUCH LONG TERM

as the test suite grows ever larger, so do the maintenance costs

- maintenance effort is cumulative, benefits are not

the test suite takes on a life of its own

- testers depart, others arrive, test suite grows larger nobody knows exactly what they all do … dare not throw away tests in case they're important

inappropriate tests are automated

- automation becomes an end in itself

MAINTAIN CONTROL

Best practice

keep pruning

- remove dead-wood: redundant, superceded, duplicated, worn-out
- challenge new additions (what's the benefit?)

measure costs & benefits

- maintenance costs
- time or effort saved, faults found?

INVEST

Best practice

commit and maintain resources

- “champion” to promote automation
- technical support
- consultancy/advice

scripting

- develop and maintain library
- data driven approach, lots of re-use

TESTS TO AUTOMATE

Best practice

run many times

- regression tests
- mundane

expensive to perform manually

- time consuming and necessary
- multi-user tests, endurance/reliability tests

difficult to perform manually

- timing critical
- complex / intricate

Automate

TESTS NOT TO AUTOMATE

Best practice

not run often

- if no need (rather than expensive to run manually)
- one off tests (unless several iterations likely and build cost can be minimised)

not important

- will not find serious problems

usability tests

- do the colours look nice?

some aspects of multi-media applications