

Text-based Acceptance testing using TextTest

Geoffrey Bache, ScanDev 2010



Contentions

1. Acceptance testing based on xUnit's approach isn't always the best choice.
2. The text-based approach is a useful alternative.
3. Such tests can be used effectively by both developers and testers.
4. Text-based test-driven development can be even more agile than ordinary TDD.

A typical unit test

```
def testAdd(self):  
    x = Number(1)  
    y = Number(1)  
    result = Number(2)  
    self.assertEqual(x.add(y), result)
```

- As seen in xUnit tools (here PyUnit)
- We assume an API to the Number object
- We assert that it returns certain values hardcoded in the test

Acceptance test tools have the same approach

Fixtures.Addition		
x	y	add()
1	1	2
1	2	3

- Business people write tests in tabular format
- Developers write “fixtures” to make them execute.
- Essentially xUnit with variable data and a nice interface.

Example situations where this paradigm is less than ideal

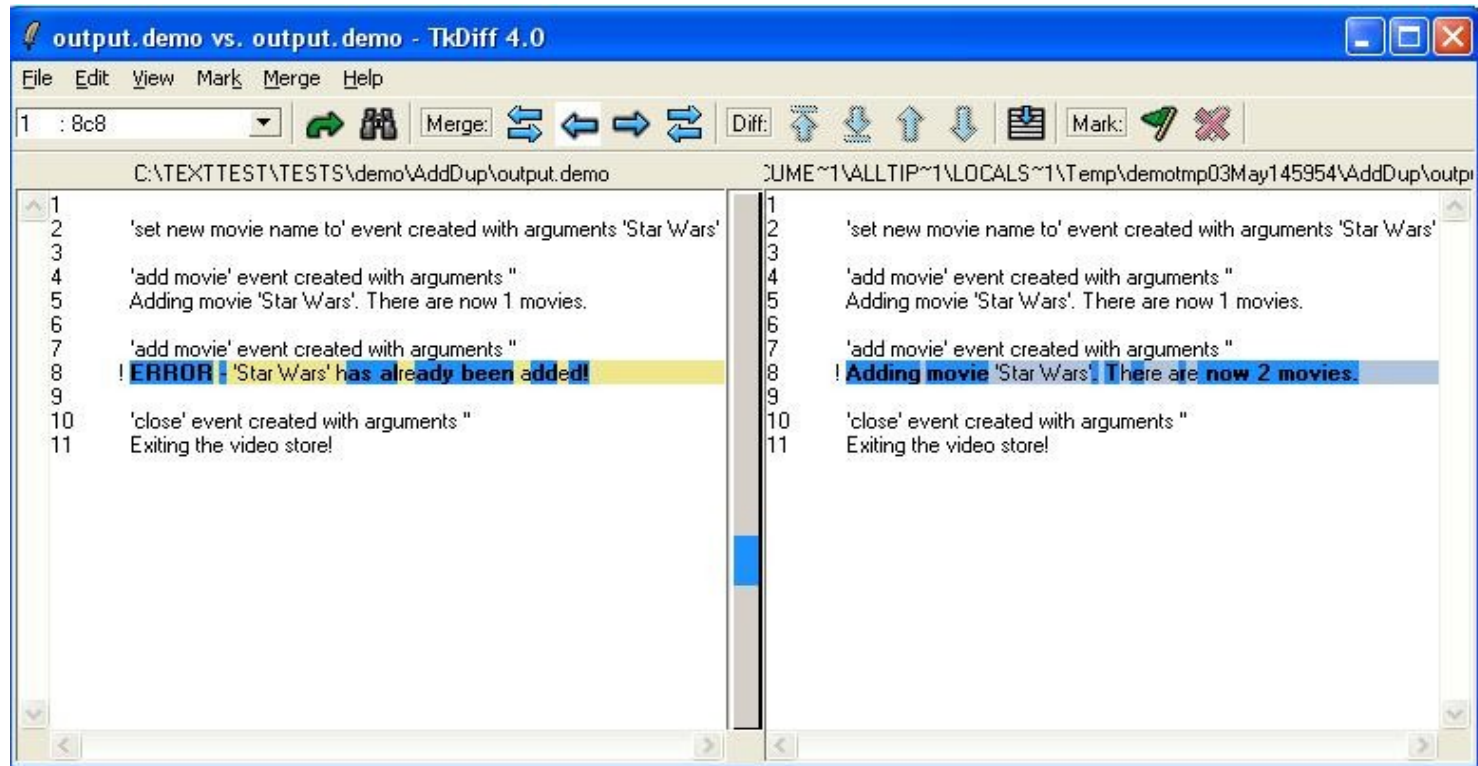
- × **Non-mainstream languages**
 - Need for porting
- × **UNIX-style command-line scripts**
 - Bypass command-line + output
 - Test code is overkill
- × **Legacy code**
 - Retrofitting test APIs difficult
 - Correct behaviour unknown
- × **System testing**
 - Behaviour subjective and volatile

(Re-)Introducing The text-based paradigm

```
$ expr 1 + 1
2
$ cat config.expr
executable:expr
$ cat Test/options.expr
1 + 1
$ cat Test/output.expr
2
```

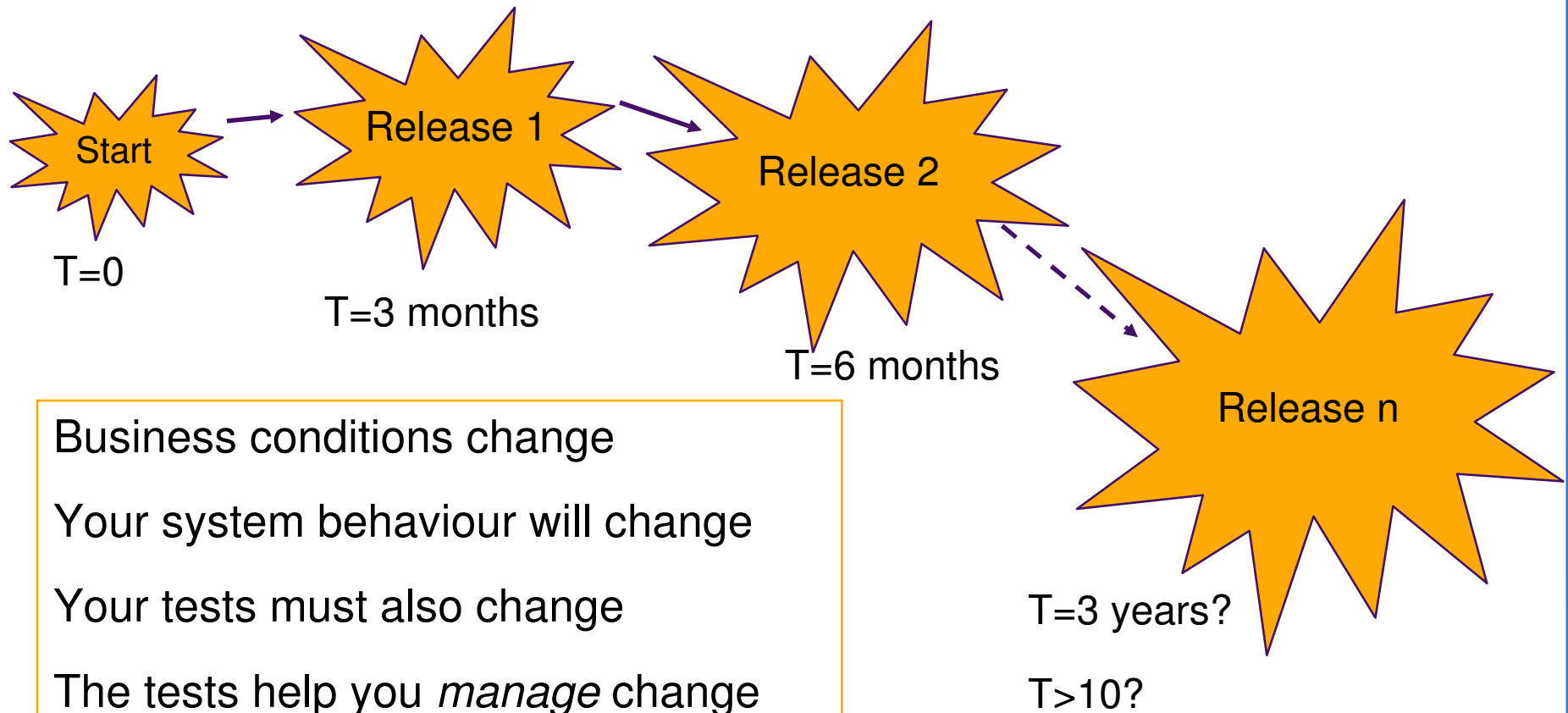
- Define tests in terms of different command lines (and data files)
- Compare textual output files with previously saved versions
- “Save” when appropriate

Behaviour Change Management by Comparing Plain Text



- Use “result files” and internal logs as a measure of system behaviour.
- Invest in them so they are easy to read and have the right level of detail.

Acceptance testing as Behaviour Change Management – not Correctness Assertion



What is TextTest?



- A tool for automated text-based testing
- Free, open source and cross-platform
- Continually developed and improved by Jeppesen with help from open source contributors.
- Short demo follows...

My tests will just fail constantly...

TextTest dynamic GUI : testing TextTest under /users/geoff/work/master/Testing/TextTest (started at 03Mar13:43:05)

File View Actions Bazaar Help

Quit Kill Save All 1406 tests completed at 03Mar13:50:10

Tests: 0/1406 selected, 895 hidden

Test	Status
CreateDataDirectory	gui_log different
RecreateDataDirectory	gui_log different
ChooseDataSource	gui_log different
RemoveDataDirectory	gui_log different
OverwriteFileRootSuite	gui_log different
ReservedNames	gui_log different
ViewEnvironmentData	gui_log different
NamesOverlap	gui_log different
Checkouts	
GUI	
BadAbsolutePath	gui_log different
BatchTests	
GUI	
SaveVersionMultiApps	dynamic_gui_log different
ReconnectTests	
GUI	
RecomputeFileFilters	dynamic_gui_log different(+)
RunAndReconnect	dynamic_gui_log different(+)
NoTestDir	dynamic_gui_log different
RecomputeLater	dynamic_gui_log different
RecomputeLaterDiffChanged	dynamic_gui_log different
DirectoryOptions	gui_log different
RememberOptions	gui_log different
DiagnosticHandling	
GUI	
DiagFailure	dynamic_gui_log different
DiagOnPermanently	dynamic_gui_log different
EnableDiagnostics	gui_log different
RecreateDiagnostics	gui_log different
TempDiagOnPermanent	dynamic_gui_log different
CopyTests	gui_log different
DiagNoFile	dynamic_gui_log different
CompleteAfterRemovedDiags	dynamic_gui_log different
RecomputeRemovedDiags	dynamic_gui_log different

Status	Number	Visib
Final filter	0	<input type="checkbox"/>
Succeeded	895	<input type="checkbox"/>
Failed	511	<input checked="" type="checkbox"/>
Differences	511	<input checked="" type="checkbox"/>
dynamic_gui_log different	233	<input checked="" type="checkbox"/>
Group 1	129	<input checked="" type="checkbox"/>
Group 10	2	<input checked="" type="checkbox"/>
Group 11	2	<input checked="" type="checkbox"/>
Group 14	2	<input checked="" type="checkbox"/>
Group 15	2	<input checked="" type="checkbox"/>
Group 2	38	<input checked="" type="checkbox"/>
Group 22	3	<input checked="" type="checkbox"/>
Group 3	21	<input checked="" type="checkbox"/>
Group 4	4	<input checked="" type="checkbox"/>
Group 5	10	<input checked="" type="checkbox"/>
Group 6	2	<input checked="" type="checkbox"/>
Group 7	5	<input checked="" type="checkbox"/>
Group 9	2	<input checked="" type="checkbox"/>
gui_log different	332	<input checked="" type="checkbox"/>
Group 1	2	<input checked="" type="checkbox"/>
Group 10	13	<input checked="" type="checkbox"/>
Group 11	16	<input checked="" type="checkbox"/>
Group 14	6	<input checked="" type="checkbox"/>
Group 15	2	<input checked="" type="checkbox"/>
Group 17	4	<input checked="" type="checkbox"/>
Group 18	2	<input checked="" type="checkbox"/>
Group 2	94	<input checked="" type="checkbox"/>
Group 20	3	<input checked="" type="checkbox"/>
Group 27	2	<input checked="" type="checkbox"/>
Group 29	2	<input checked="" type="checkbox"/>
Group 3	62	<input checked="" type="checkbox"/>
Group 30	3	<input checked="" type="checkbox"/>

Shortcuts: qsys New

TextTest started at 03Mar13:43:13.

- Easy to filter “run-dependent text” e.g. times, addresses
- Need to change mindset from xUnit : Behaviour Change Management
- Important to invest effort in improving logs
- Good logging is a skill

The 'Expert Reads Output' Antipattern



- "Output reading is error-prone. We'll end up proving the wrong thing!"
- Configure TextTest to find error messages etc. automatically.
- TextTest will group similar changes.
- Behaviour Change Management : changes are more important than contents.
- Care and discipline still needed

Tests are independent of code structure

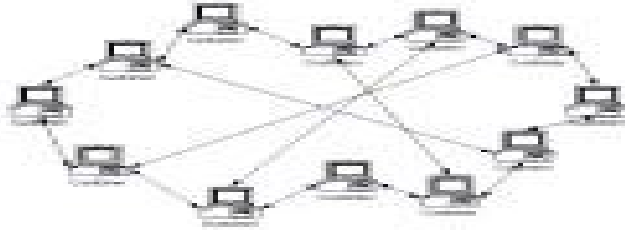


- Ideal for regression testing legacy code
 - Extracting an API to call is risky
 - Inserting log statements much safer
- Refactoring never breaks your tests
 - Makes large scale refactoring practical
- But tests do not drive design in any way

Little or no coding per test

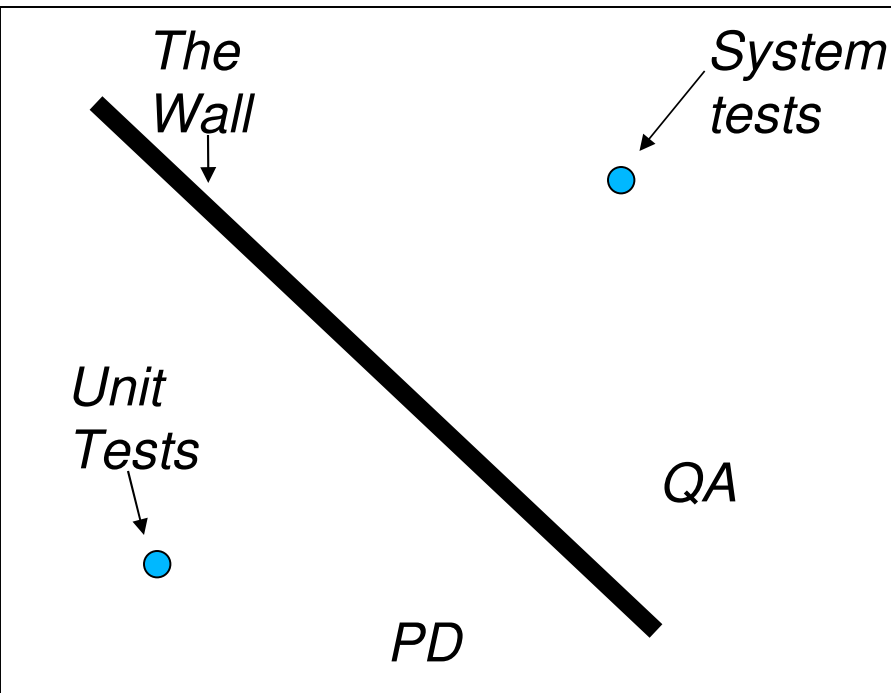
- Writing a test does not usually mean writing code
 - Log statements replace assertions
 - Apply to many tests
 - Live with the code and naturally evolve with it
- Having a lot of test code is *not* a good thing
 - Shuts non-coders out of the tests
 - Has to be maintained
 - Hinders mobility of the code

One test tool for the whole system



- Tests independent of programming language
 - Good for less mainstream languages
 - Good for polyglot systems
- Can capture interactions between components
 - Records to a file what the real component does
 - Can then replay it without that component
 - Command line / synchronous messages / Python modules

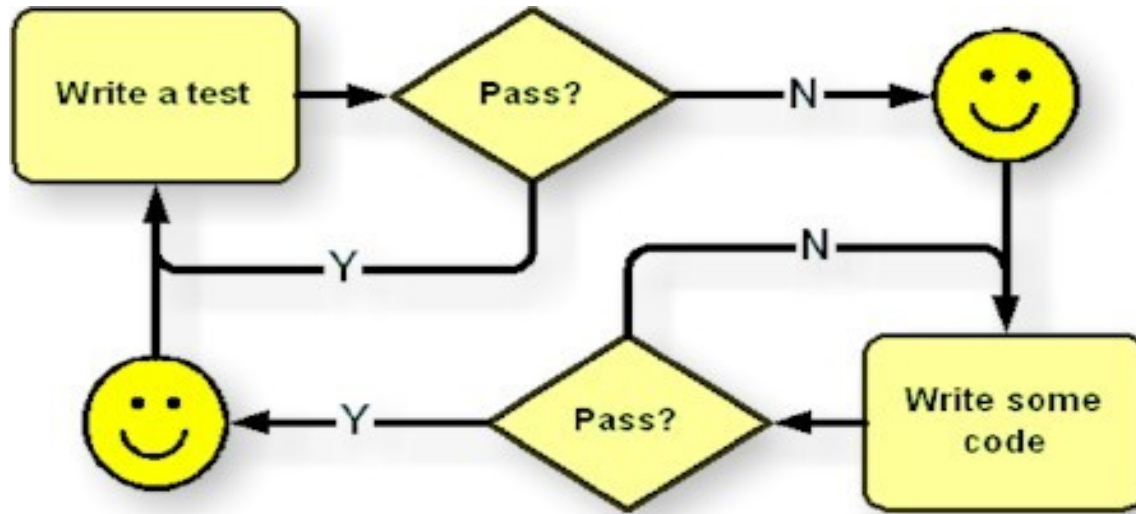
One test tool for the whole team



How automated tests are often organised.

- QA people can read and write tests – just edit plain text files.
- Developers can also create logs of lower level detail behaviour
- Normally disabled : enabled for debugging
- But checked into version control
- Gradually build a knowledge base, unlike using debuggers

Test-Driven Development



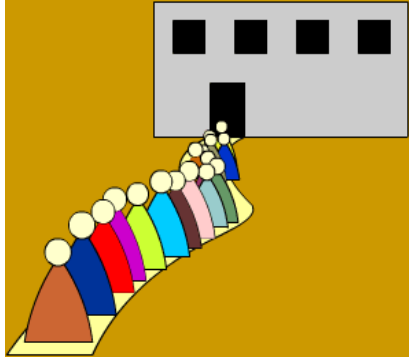
- We want to provide rapid feedback to the developer at every build
- Better feedback from system-level, business perspective tests
- Also becomes a communication channel between QA and PD

Text-based Test Driven Development



- Not “pure” test-first
 - Set up test input beforehand
 - Don’t usually specify behaviour up-front
 - “Improve” behaviour incrementally in debug logs
- Closely related to Behaviour Driven Development
 - Strong focus on behaviour over structure
 - “Outside-in” methodology enforced
- “Refactor” stage includes logs created
 - Tidy them up and check some of them in

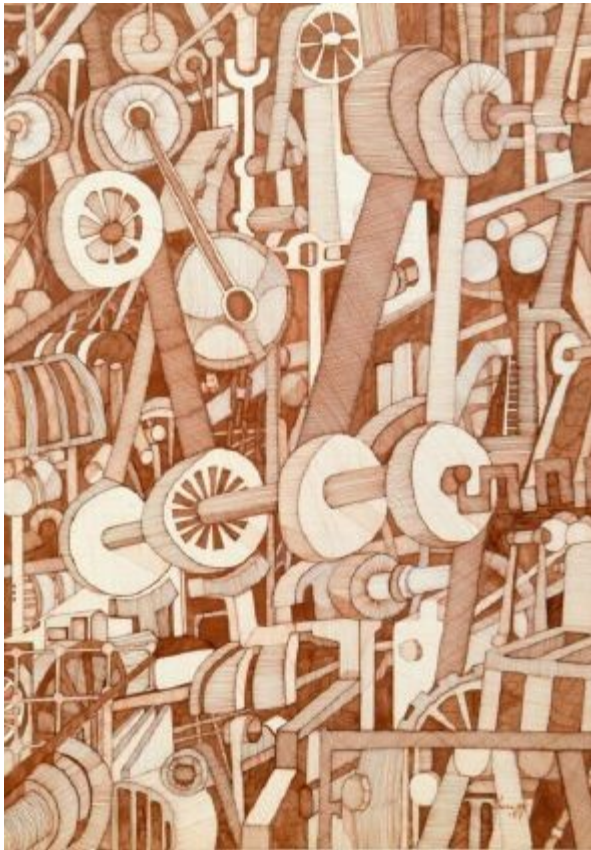
“But system tests will be way too slow to run at every build”



- Then run them in parallel!
 - Hardware is getting cheaper all the time
- Use a grid engine, e.g. Sun Grid or LSF
 - TextTest integrates directly with them
 - Sends all tests to a central pool
 - More tests faster => buy more hardware
- Use a cloud
 - Typical cost about 0.1 USD per instance-hour



“Surely you aren’t suggesting unit testing is now unnecessary?”



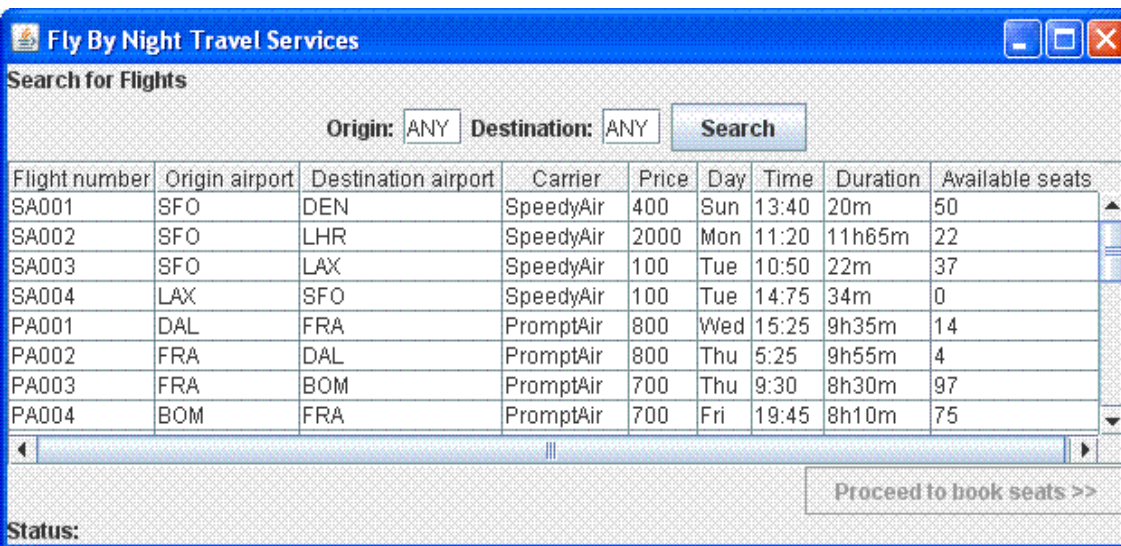
- Not needed for speed
 - Parallelised acceptance tests
- Not needed for error localisation
 - Debug-level logs in acceptance tests
- Not needed to drive design
 - interactive interpreters
 - “throw-away” unit tests
- Needed for simplicity and documentation
 - Where units are well defined /reusable

Conclusions



- The “xUnit paradigm” isn’t always ideal
- Text-based testing is particularly a good idea for
 - Legacy systems
 - Polyglot systems
 - Command-line scripts
 - System testing
- Change of mindset : “Behaviour Change Management”, not “Correctness Assertion”.
- Text-based can be more agile:
 - No coding per test
 - Technology and business perspectives in same test
 - Anyone can read plain text

Text-based Acceptance Testing and the GUI



- Use TextTest in conjunction with a pure simulation library (e.g. xUseCase tools)
- These are essentially domain-language record/playback.
- Also generate a log of visible UI changes.
- **Come to my talk tomorrow!**

```
wait for flight information to load
select flight SA004
proceed to book seats
accept error message
quit
```


TextTest Features



- Filters output to avoid false failure
- Manages test data and isolation from global effects
- Automatic organisation of test failures
- “Nightjob website” to get a view of test progress over time
- Performance testing
- Integrates with Sun Grid Engine for parallel testing (and LSF)
- Various “data mining” tools for automatic log interpretation
- Interception techniques to automatically “mock out” third-party components (command line/libraries/network traffic).
- Integrates with xUseCase tools for GUI testing
- Easy to customize with your own Python code