EUROMNIS 2023

OMNISTAP IN DEPTH

OMNISTAP ON GITHUB

- https://github.com/suransys/omnistap
- Installation guide
- Wiki
- Requirements
  - Studio 8.1
  - macOS or Windows
  - Un-tested (ha ha) on Linux, but it ought to work

WHAT IS TAP

- Test Anything Protocol: https://testanything.org
- Started as a protocol for perl tests in 1987 (not as old as Omnis, but not far off)
- Any TAP consumer can read output from any TAP producer
- OmnisTAP is a TAP producer

SAMPLE OMNIS TAP CODE

```
Do ioTAP.$ok(1=1,"1 equals 1”)


Do ioTAP.$is_char(low("FOO"),"foo","low() works”)


Do ioTAP.$isnotclear($libs.$findname("omnistap_example"),"Our
  library is open”)
```

SAMPLE TAP OUTPUT

```
1..3
ok 1 1 equals 1
ok 2 low() works
ok 3 Our library is open
# 17 ms
```

OMNISTAP SUPPORT FOR THE TAP SPECIFICATION

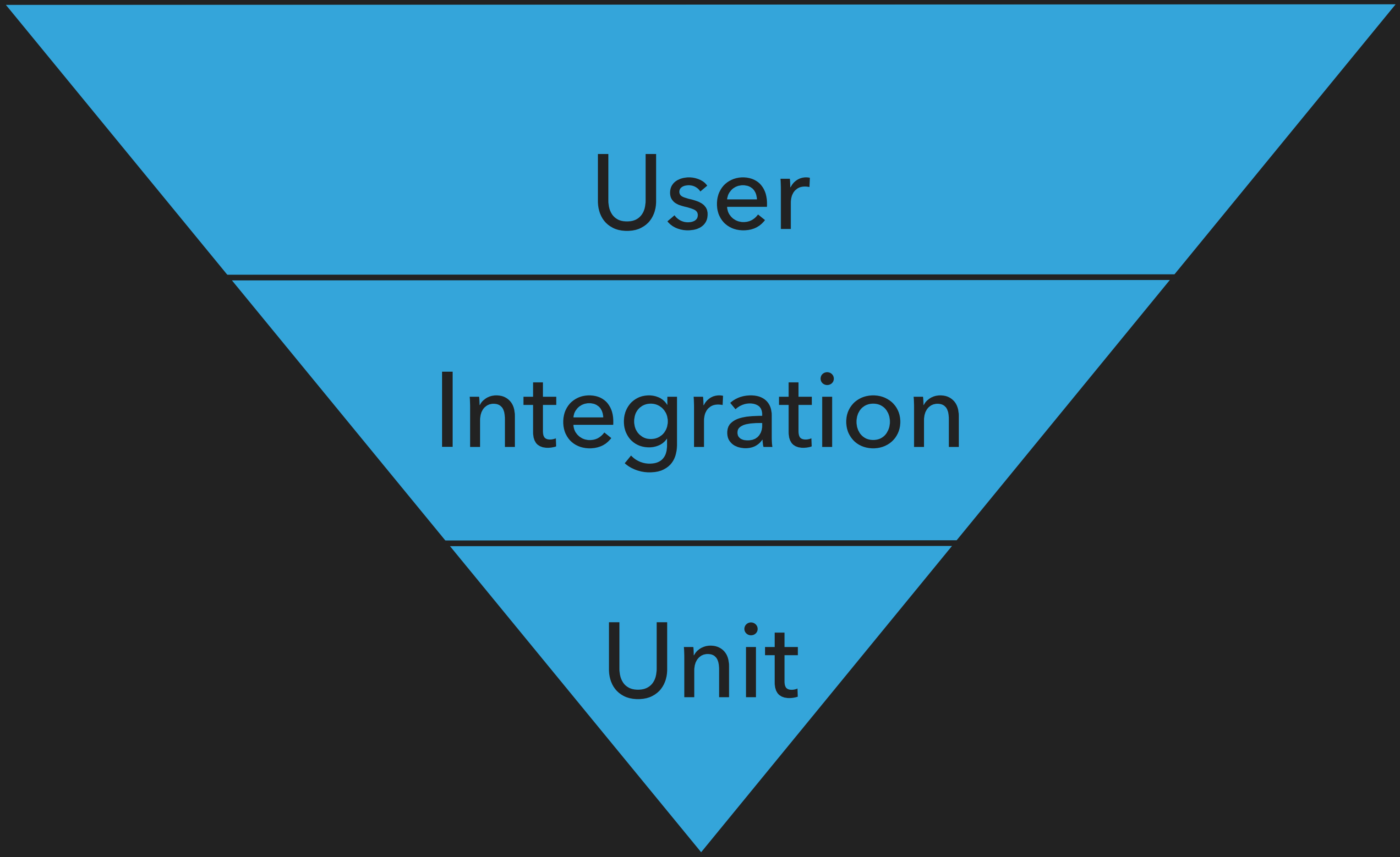
pass (ok)	✓
fail (not ok)	✓
diagnostic (#)	✓
Plans (1..N)	✗
Bail	✗
Skips	✗
To Do	✗

WHAT CAN YOU TEST WITH OMNIS TAP

- Pretty much anything in Omnis!
- Generally break tests down into two categories:
  - Unit tests
  - Integration tests

Unit Test	Integration Test
Runs in < 5 seconds	Runs in < 90 seconds
Tests a single method	Tests how multiple methods work together
Should form the base of your test suite	Good way to get started and useful in the long run

TESTING PYRAMID



TYPES OF METHODS

- Operation
- Accessor
- Decision

SAMPLE OPERATION METHOD: $BUILDFULLNAME

```
Do loPerson.$buildFullName(
   pcTitle,
   pcFirst,
   pcMiddle,
   pcLast
 ) Returns lcFullName
```

FIXTURES

pcTitle	pcFirst	pcMiddle	pcLast	lcFullName
Captain	James	Tiberius	Kirk	Captain James Tiberius Kirk
	James	Tiberius	Kirk	James Tiberius Kirk
	James		Kirk	James Kirk
Captain			Kirk	Captain Kirk

TESTS USING FIXTURES

```
Do loPerson.$buildFullName("Captain","James","Tiberius","Kirk") Returns lcFullname
Do ioTAP.$is_char(lcFullname,"Captain James Tiberius Kirk")

Do loPerson.$buildFullName("","James","Tiberius","Kirk") Returns lcFullname
Do ioTAP.$is_char(lcFullname,"James Tiberius Kirk")

Do loPerson.$buildFullName("","James","","Kirk") Returns lcFullname
Do ioTAP.$is_char(lcFullname,"James Kirk")

Do loPerson.$buildFullName("Captain","","","Kirk") Returns lcFullname
Do ioTAP.$is_char(lcFullname,"Captain Kirk")
```

EXAMPLE ACCESSOR METHOD: $ISSUBSCRIBED

```
If $cinst.subscription_begin>=#D
  Quit method kTrue
End if


Quit method kFalse
```

TESTING AN ACCESSOR METHOD

```
Calculate lrPerson.subscription_begin as #D-2
Do lrPerson.$isSubscribed() Returns lbIsSubscribed
Do ioTAP.$is_boolean(lbIsSubscribed,kTrue,"The person is
  subscribed when their subscription has begun")


Calculate lrPerson.subscription_begin as #D+2
Do lrPerson.$isSubscribed() Returns lbIsSubscribed
Do ioTAP.$is_boolean(lbIsSubscribed,kFalse,"The person is
  not subscribed when the subscription has not begun”)
```

TESTING DECISION METHODS

- Decision methods call other methods based on some logic
- It’s easy to turn these tests into integration test
- Use mocking to test the decisions
- Use integration tests to test the whole block of code as a single unit

SAMPLE DECISION METHOD: $ADDPERSONTOEMAIL

```
If pfrPerson.$isSubscribed()
  Do $cinst.$_includePerson(pfrPerson)
Else
  Do $cinst.$_excludePerson(pfrPerson)
End if
```

INTEGRATION TEST ON $ADDPERSONTOEMAIL

```
Calculate lrNonSubscriber.subscription_date as #NULL
Calculate lrSubscriber.subscription_date as #D

Do llExcludeList.$add().$assignrow(lrNonSubscriber)
Do llIncludeList.$add().$assignrow(lrSubscriber)

Do loMailer.$addPersonToEmail(lrNonSubscriber)
Do loMailer.$addPersonToEmail(lrSubscriber)

Do ioTAP.$is_list(loMailer.ilExcludeList,llExcludeList,"We add the
  non-subscriber to the exclude list")
Do ioTAP.$is_list(loMailer.ilIncludeList,llIncludeList,"We add the
  subscriber to the include list")
```

UNIT TEST ON $ADDPERSONTOEMAIL

```
Do $cinst.$mock($tables.tPerson,lrNonSubscriber) ;; Get an instance of the person
  for mocking a non-subscriber
Do lrNonSubscriber.$mock(“$isSubscribed”).$return(kFalse) ;; Set the next call to
  $isSubscriber to return false
Do loMailer.$mock(“$_excludePerson").$expect(lrNonSubscriber) ;; Expect we'll add
  the person to the exclude list


Do $cinst.$mock($tables.tPerson,lrSubscriber) ;; Get an instance of the person for
  mocking a subscriber
Do lrAlivePerson.$mock("$isSubscribed").$return(kTrue) ;; Set the next call to
  $isSubscriber to return true
Do loMailer.$mock("$_includePerson").$expect(lrSubscriber) ;; Expect we'll add the
  person to the include list


Do loMailer.$addPersonToEmail(lrNonSubscriber) ;; Add the non-subscriber
Do loMailer.$addPersonToEmail(lrSubscriber) ;; Add the subscriber
Do $cinst.$assertMocks()
```

WORKING WITH LEGACY CODE

- Add integration tests to create a basic harness
- Break out individual chunks of a method to separate methods
- Add unit tests to those methods
- Eventually, convert your integration test to use the mocker
- Optionally keep the integration test if it's important

SPECIAL MOCKING CONSIDERATIONS

- Create seams for built-in methods and variables, like $cobj, Set Current Field, or #P
- Use $store to preserve task variables like sessions, utility objects, or environment variables
- Use protected methods ($_) instead of private ones
- Encapsulate user interactions, like Yes/No or Enter Data, with seams you can mock
- Mocking a class method like $open()
- RTFW for field references, table classes, objects, and other tricky spots

ACLAY@MAC.COM

AUDIENCE PARTICIPATION