EUROMNIS 2024

TEST IT!

WELCOME

- Alex Clay
- CEO at Suran Systems, Inc.
- Software for faith-based and nonprofit organizations, and more…
- Started in 1985 with Omnis 3
- 2600+ clients
- Today we use Studio Now (11.2) and variety of other technologies
- PostgreSQL backend since 2010
- 7 full-time developers + manager (say hi to Sirena), QA, support, sales, etc.

ABOUT YOU

AGENDA

- What is automated testing?
- Why test?
- Best practices
- OmnisTAP
- pgTAP

WHAT IS AUTOMATED TESTING?

AUTOMATED TESTING

- Code that runs your code
- Based on assertions (true/false)
- Focuses on ensuring expected behavior

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”)
```

WHAT CAN YOU TEST?

- Software logic (backend, middleware, application)
- User interaction
- Deployment (e.g. installers)
- Server configuration
- More!

ENSURING EXPECTED BEHAVIOR

X INPUT YIELDS Y OUTPUT

WHO DECIDES EXPECTED BEHAVIOR?

- All stakeholders
  - Developers
  - Product owners
  - **Users (most important)**
    - Convention
    - Past experience, including in your product
    - Product signals/sign posts

USER INTERACTION LEVEL

- When I click this button, I see a confirmation dialog
- When I double-click this list item, a details window opens
- When I enter a record and click save, the database records the information

CODE LEVEL

- When I call a database function, the right records are updated
- When I pass certain values to a name formatting function, I get the right computed value
- When I ask for the tax amount on a tax-free item, I get $0.00
- When I print a statement for a customer:
  - If their account is behind on payment they get a past due notice
  - If their account is caught up they get a paid in full notice
  - If notices are suppressed they get no notice regardless of balance

CONFIGURATION LEVEL

- My database configuration has the right settings
- My tables have the right ownership
- My application installs files with the correct permissions
- My firewall is configured with the right rules

ENVIRONMENTAL LEVEL

- When the network is down the application displays a message and prevents updating
- When the local language is changed we render the correct string tables
- When the disk is full we throw a helpful error and don’t save partial data
- When the OS is too old we prevent installation

WHY TEST?

FOR THE USER: ENSURE EXPECTED BEHAVIOR

- Being user friendly
- Ensuring quality
- Ease of documentation and training
- Cleaner path for when expectations shift

FOR THE DEVELOPER: LEVEL UP YOUR CODE

- Safety harness for code improvements
- Reduces the cost of hiring new developers
- Promotes cleaner, more maintainable code
- Reduce redundant, manual work

FIXING BUGS!!!

DEFINITION OF A BUG

ACTUAL BEHAVIOR <> EXPECTED BEHAVIOR

FIXING BUGS WITH TESTING

- Reproduce the problem
- Write a test that does this automatically with code and watch it fail
- Fix the bug and watch the test repeat
- Run that test before you ship updates
- **NOTE: There is more than one way to produce a bug, so you may need additional tests**

BEST PRACTICES

SOME ADVICE

- Start by testing the most critical components
- Always a test for every bug
- Use tests for unclear logic (domain-specific behavior for the user)
- Write testable code
  - Operation/Access/Decision
- AUTOMATE AUTOMATE AUTOMATE

OMNISTAP

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

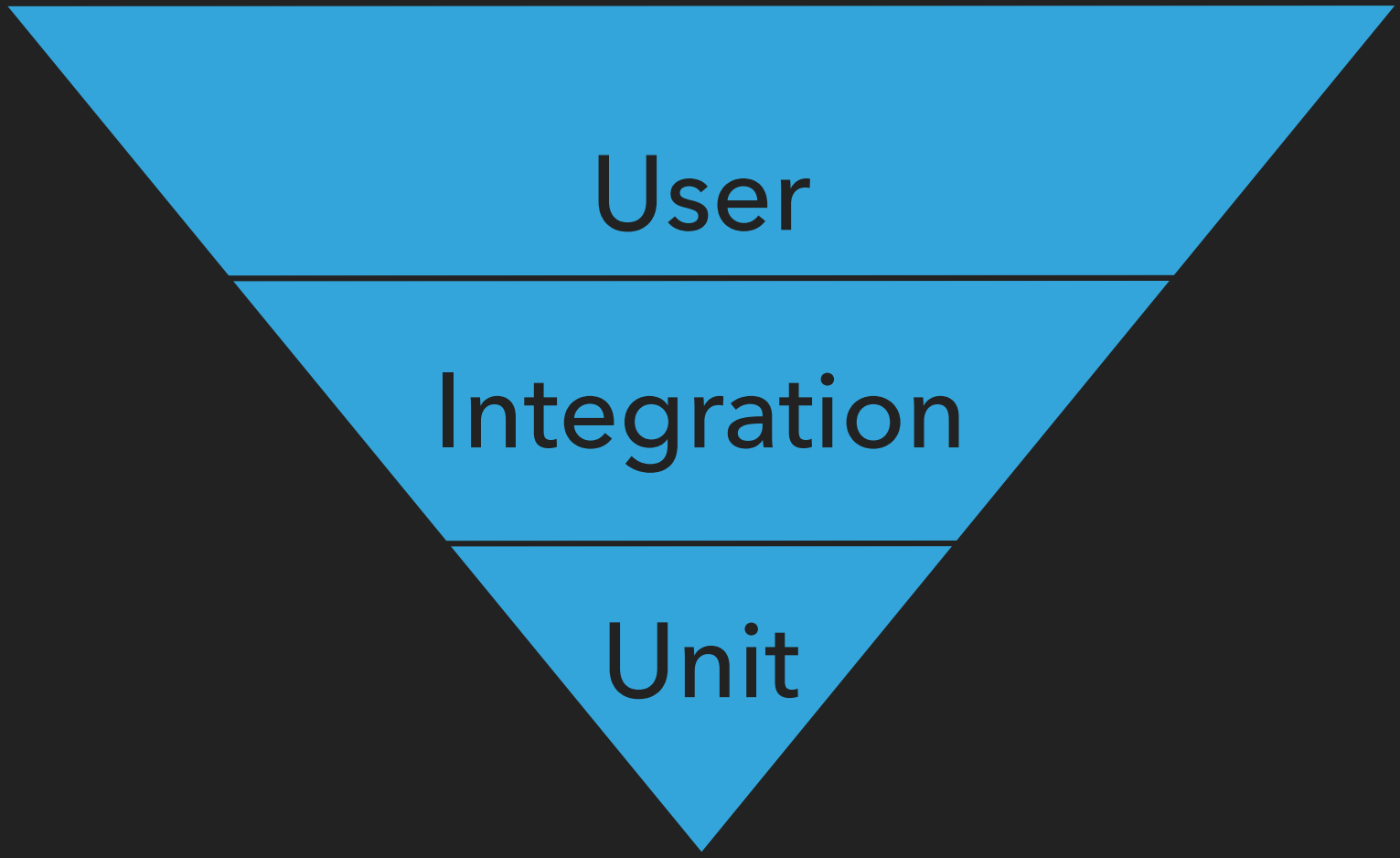
pass (ok)	✓
fail (not ok)	✓
diagnostic (#)	✓
Plans (1..N)	✘
Bail	✘
Skips	✘
To Do	✘

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

DEMO

PGTAP

PGTAP

- https://pgtap.org/
- Created by David Wheeler
- Suran has a mocking tool and some IDE/build automation tooling

PGTAP USE CASES

- Database functions (we have nearly 5,000)
- Database schema (tables, columns, views)
- Roles and permissions
- Extensions and server configuration

DEMO