Compare unit test to an integration test.

c:\Users\md663b\Documents\eet\_bert3\test\tests\ATX\test\_atxEscalate\_Integration\_dev.t

c:\Users\md663b\Documents\eet\_bert3\test\tests\ATX\test\_atxEscalate.t

1. Unit 1600 lines
   1. Begin
   2. Use
   3. Mock object
   4. 10 subtests – that is, 10 different subroutines are checked here. Why these 10?
      1. Many unit tests have one 1 subtest
      2. each with 5-6 checks; 5-6 ways something could go wrong with this sub
2. Each subtest
   1. Subtest defined with title => sub {
   2. Define fake inputs for this subtest
   3. Check 1
      1. Do you need to create a mock db?
         1. Mockdb
         2. Add resultset
         3. Associated it with dbh property of object being passed into call
      2. $result – call sub
      3. Define expected data
      4. Assert $result vs. expected.

**Integration test**

1. Begin
   1. Use lib
   2. Use test::MockModule
   3. Pass in all modules inot new() of mockmodule.
2. Use – call in reamining modules.
3. Prepare tests – that is,
   1. create all the mocks – here 10 mocks
      1. define the return for either a call to the module or to a sub inside the module
   2. create objects for real modules used
   3. define any datainput/outputs.
4. Instead of “subtests” there is a series of checks – 16 of them
5. Destruct Tests section
6. Helper functions
   1. Un/seedDB
   2. mockSOAP
   3. un/seedSpecificEscalateTest
   4. un/seedUpdateOrder
   5. setEFMSForUpdate, for closed,
7. Finish Testing

Donetesting

dbConnectBert