Approach to Automation

By Nikhil Sobti

About myself

- I have 10 years of both manual and automation testing experience.
- I worked both at Public and Private sector, at Glencore, DCSI, Petrosys Pty Ltd, SAFECOM, SA Water, SAPOL.
- My experience includes performing System/Integration Testing, UAT Testing, Automation Testing using tools like Froglogic Squish, UFT, Selenium, etc.
- You can connect with me on LinkedIn

Agenda

- Background – Why test automation, common views about test automation?
- Define Problems – Issues most organizations face with test automation.
- Documentation to support automation.
- Fixing the above issues with architecture that has the most ROI
- Review

Why Test Automation

Dev Team's Perspective

- Increase effectiveness, efficiency and coverage of software testing.
- For regression testing.
- Improve quality.

Business Perspective

- Reduce operational cost.
- Increase revenue.

What comes to your mind when you hear the term “We have Automated Tests”?

Replicating Manual Testing – GUI Testing Functional Test

Take the weight off of the manual testers

Focused on GUI testing

- Hire consultants onshore/offshore with a mantra “Automate everything”.
- QTP, Selenium Webdriver, Test Complete, etc. use tools specific to GUI tests.
- Every single small scenario gets added to GUI automation.

Tests start failing randomly

When you need the test to run – It never works.

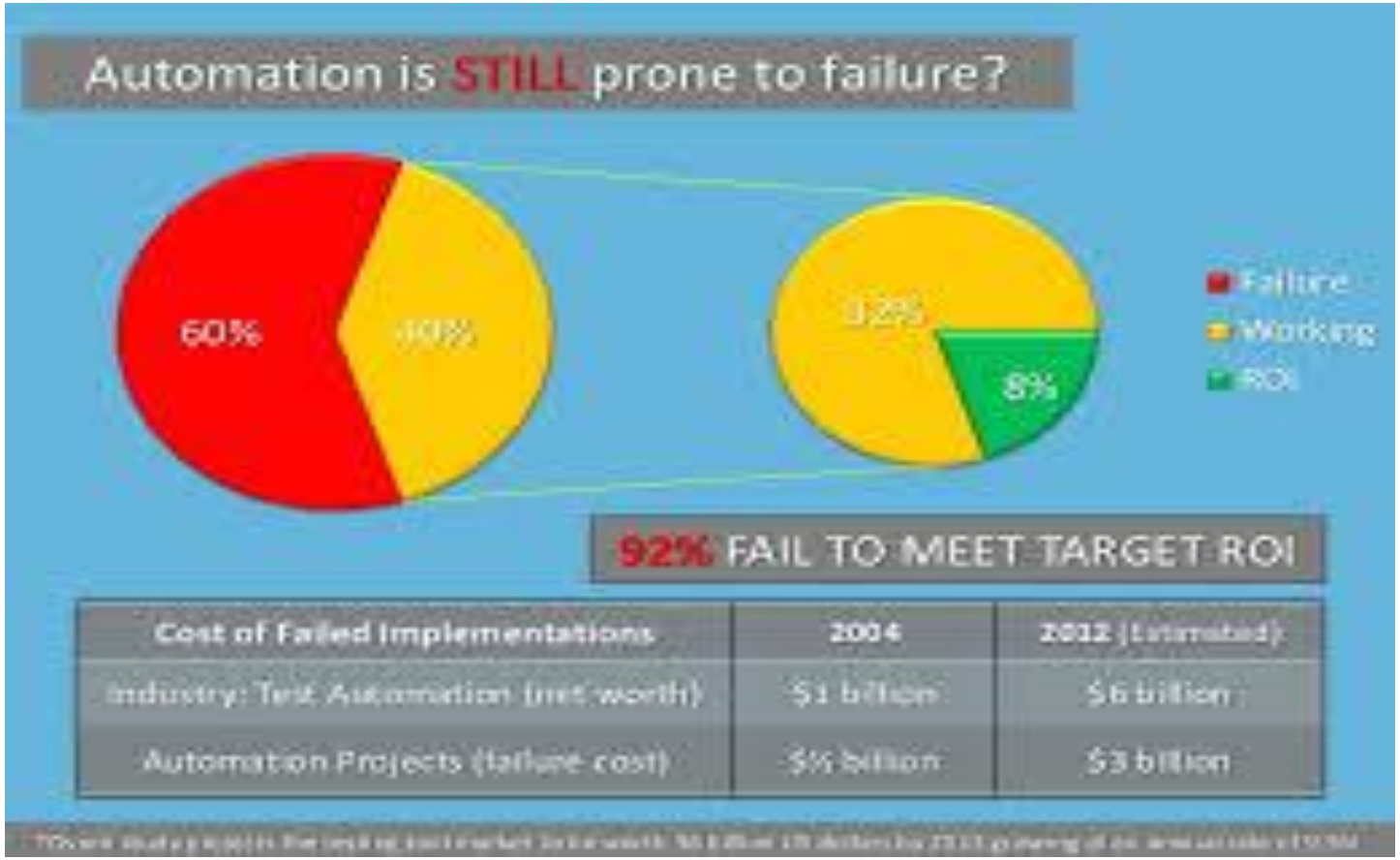
Please work!

What’s the problem eh?

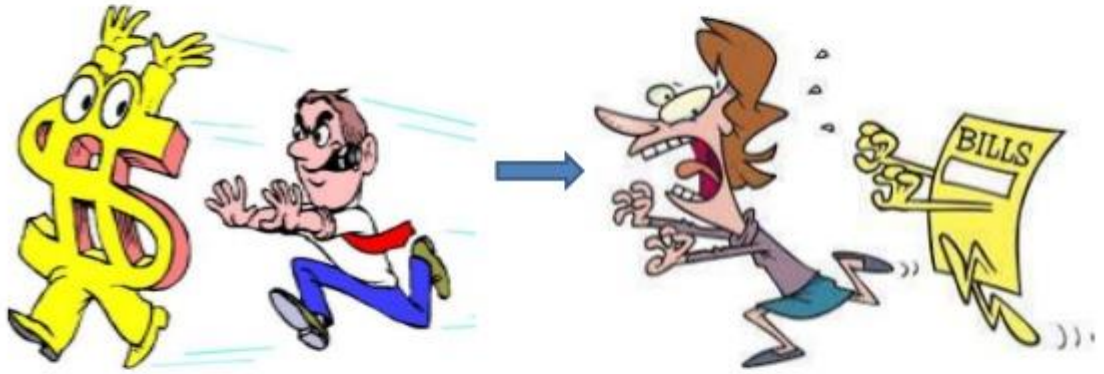
- Test execution is slow.
- Tests are fragile.
- Tests are not reusable.
- Tests need more infrastructure.
- Tests never works on my PC and fails on daily run.

Leads to

- More time spent on figuring why the tests are failing.
- Resource tied up just reviewing failures.
- Maintenance takes way too much time.


Automation is STILL prone to failure?
60%
40%
32%
8%
Failure
Working
ROI
92% FAIL TO MEET TARGET ROI
Cost of Failed Implementations
2004
2012 (Estimated)
Industry: Test Automation (net worth)
$1 billion
$6 billion
Automation Projects (failure cost)
$½ billion
$3 billion

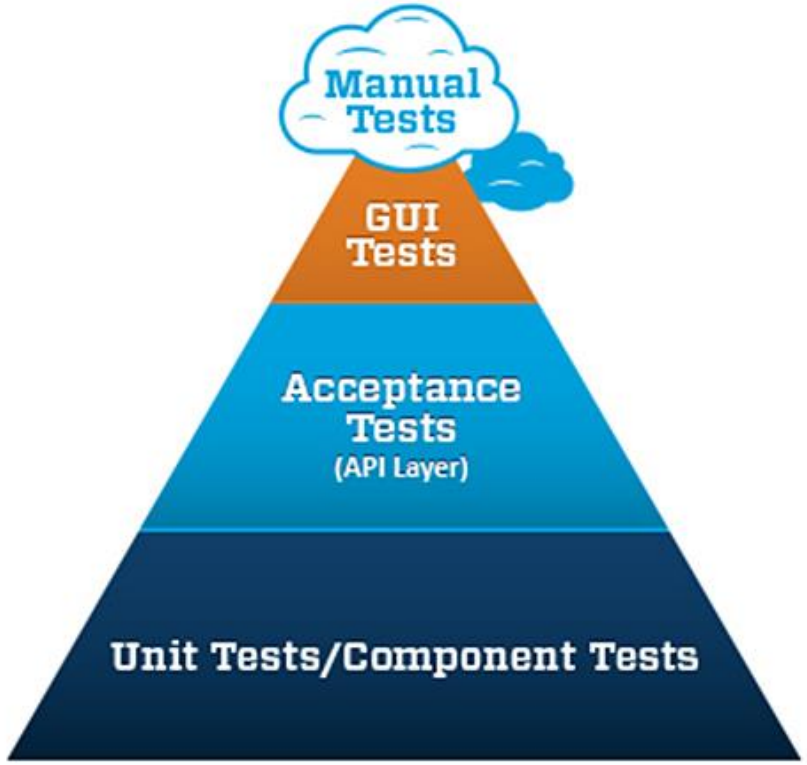
Goal of Automation → Ends up being



Why do we end up in this situation?

Ignoring the fundamentals.

Golden Rule


Manual Tests
GUI Tests
Acceptance Tests
(API Layer)
Unit Tests/Component Tests

GUI Layer Test

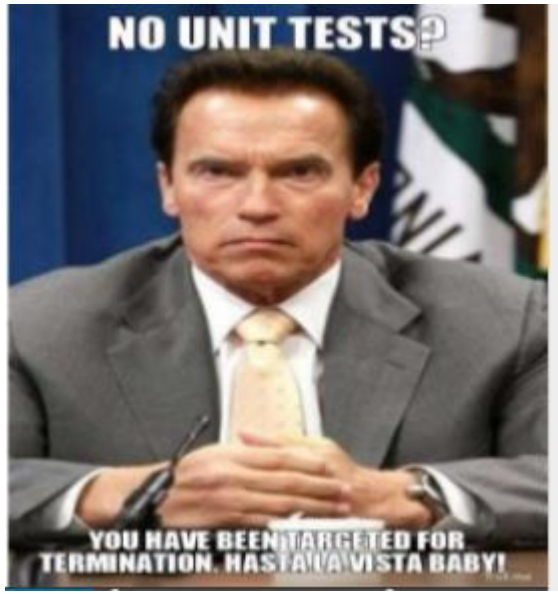
- Test the whole application end to end.
- Sensitive to UI changes.
- Does not require good code.
- Works for Legacy system and with newer software architecture like SOA as it is independent of backend technology.
- Eg. Open a browser, fill in the form and submit the data.

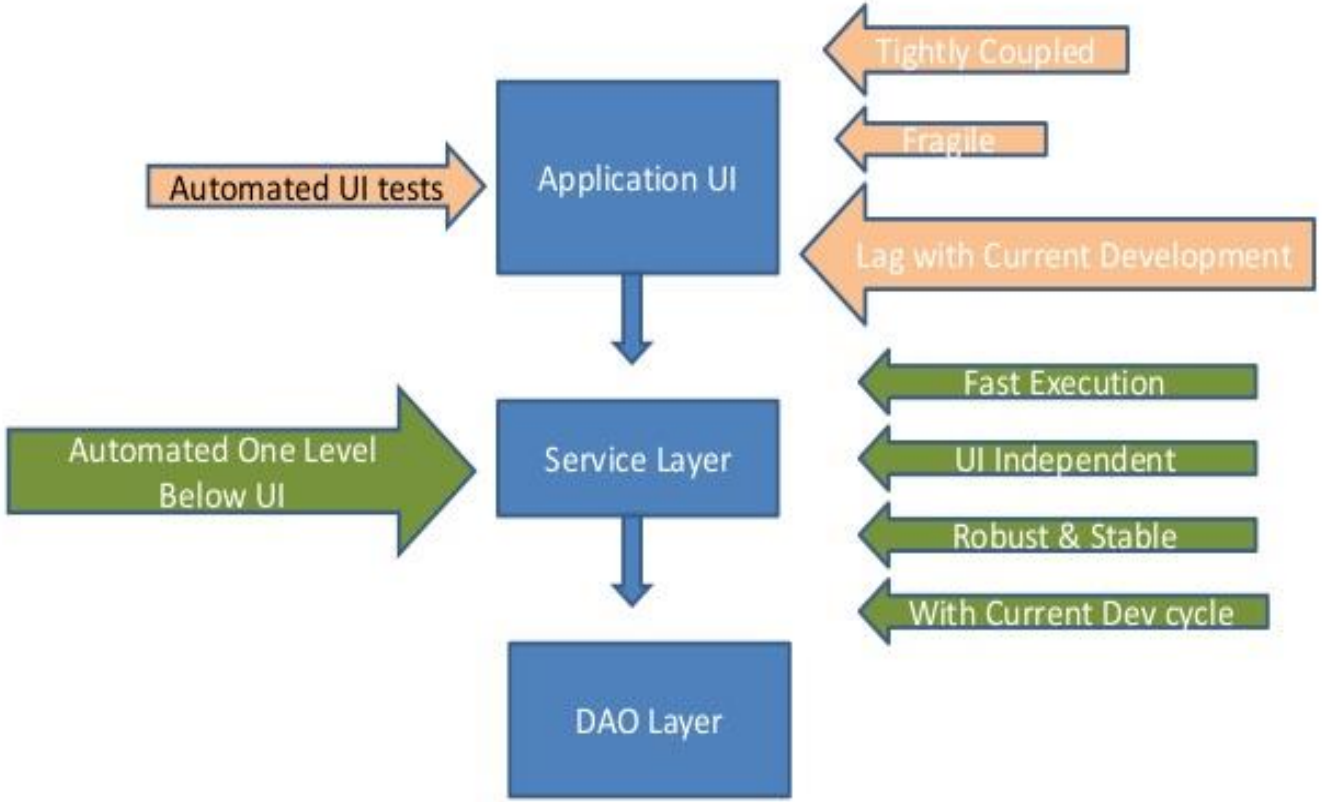
API/Service Layer test

- Test the application logic by making service calls.
- Faster than UI testing.
- Requires service to exist.
- Eg. Instantiate the calculator service and get it to add two numbers.

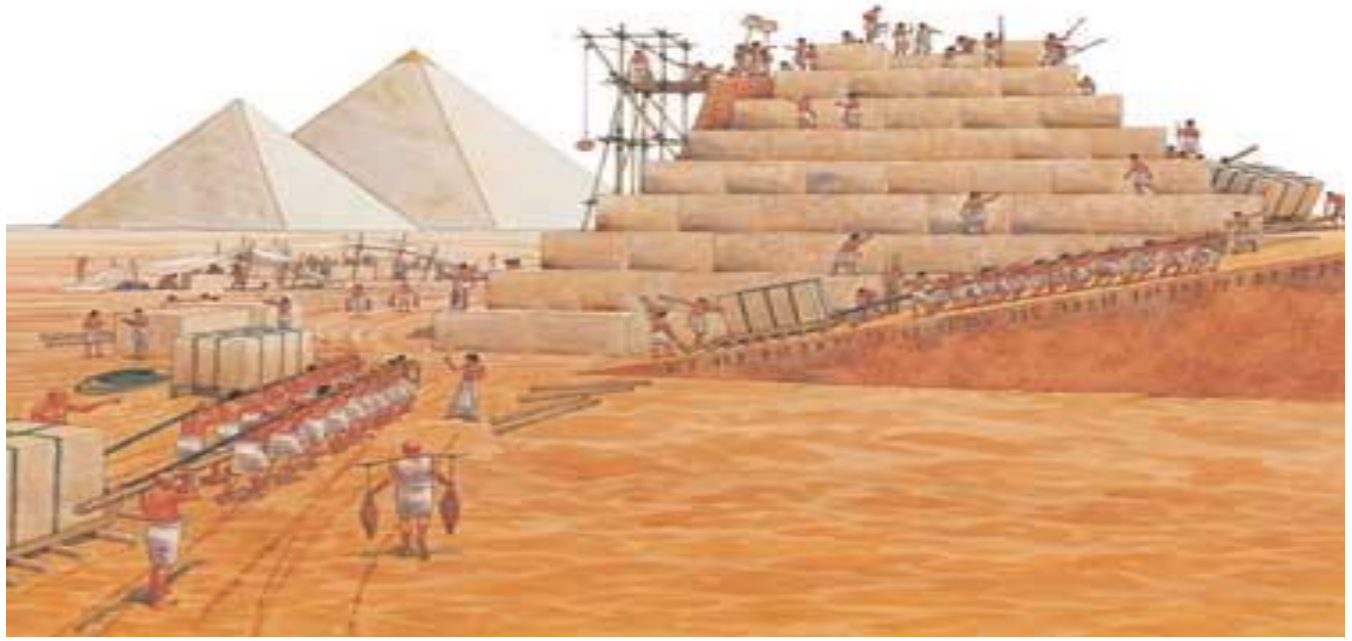
Unit level test

- Test individual classes.
- Must faster than service level testing.
- Requires good design.



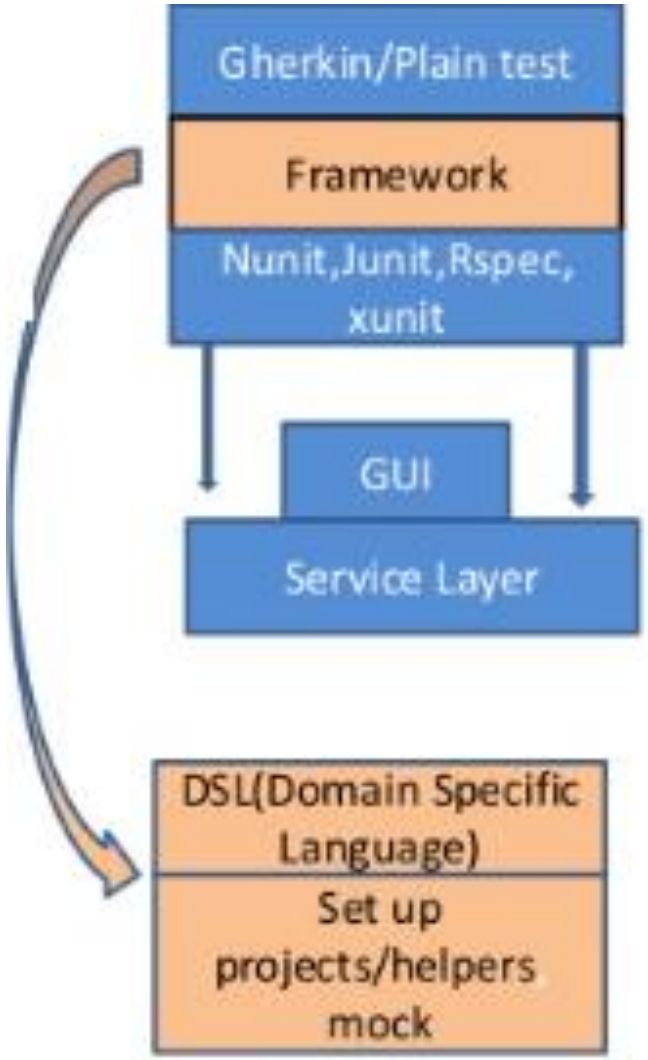

Tightly Coupled
Fragile
Automated UI tests
Application UI
Lag with Current Development
Fast Execution
Automated One Level Below UI
Service Layer
UI Independent
Robust & Stable
With Current Dev cycle
DAO Layer

So how to build your pyramid?



Start with Unit Test
(Quality is not just QA's responsibility)

- Isolation.
- Test small piece of code – preferably separate test for each method.
- Biggest test should be one test for a class.
- Use mock services to remove dependency.
- Test only public endpoints.
- This forces to use good design.



Invest more in Service Layer Test

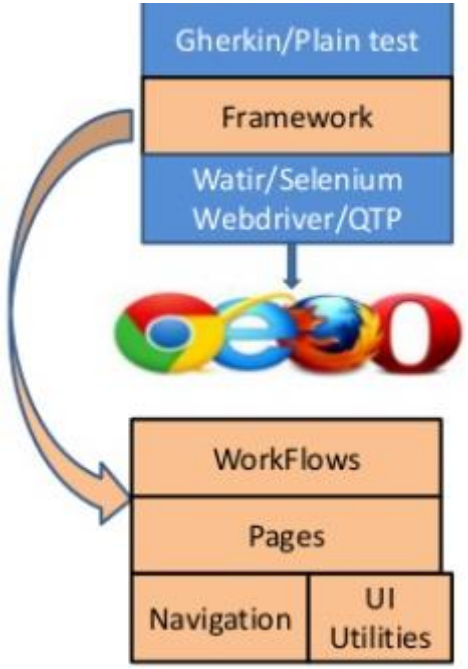
- Independent functional test.
- Use BDD tools like Cucumber, SpecFlow, Jbehave.
- Use mocks but also have specific tests to test third-party calls.
- Add data validation tests.
- Tests for boundary condition.
- Tests for error handling – wrong data type, missing data.
- Tests around calculations, business rules, etc.

Limit the Number of GUI test

- Ask if it can be tested in Service layer before starting.
- Start out with use and throw style test.
- Gradually build the framework.
- Use GUI test as an assistance for manual testing.
- Create fewer tests with broader end to end coverage.

Quick Guide to Building GUI Test Framework

- Do basic meta-programming and start automation. Don't design the entire framework right away.
- Re-factor often.
- Review test automation architecture regularly.
- Run daily.



Proof of concept (Select automation tool)

1. Decide test cases to be used in POC
2. Plan to show Manual Vs Automation in a way to show better quality outcomes.
3. Include test case that fails.
4. Use assertions & validation points where necessary.
5. Show areas that can/cannot be automated.
6. Pros/Cons of tools [Eg. Environment supported].
7. Can all features of application under test be automated using tool.
8. For web app, does it support cross-browser testing.
9. Reporting/Logging capability.
10. Additional features supported by tool.
11. Tool able to satisfy management objectives.
12. Give a live-demo to Management and seek approval

Documentation to support automation (Section-1)

1. Prepare Automation Test Plan

Objective, Test automation architecture & approach, What's In-Scope, Challenges faced during automation, Test automation cycle, Automation tool selection/Proof-of-Concept, Features of automation framework

2. Define Object Mapping Document

- *Best practices*
- *Issues & Solutions*

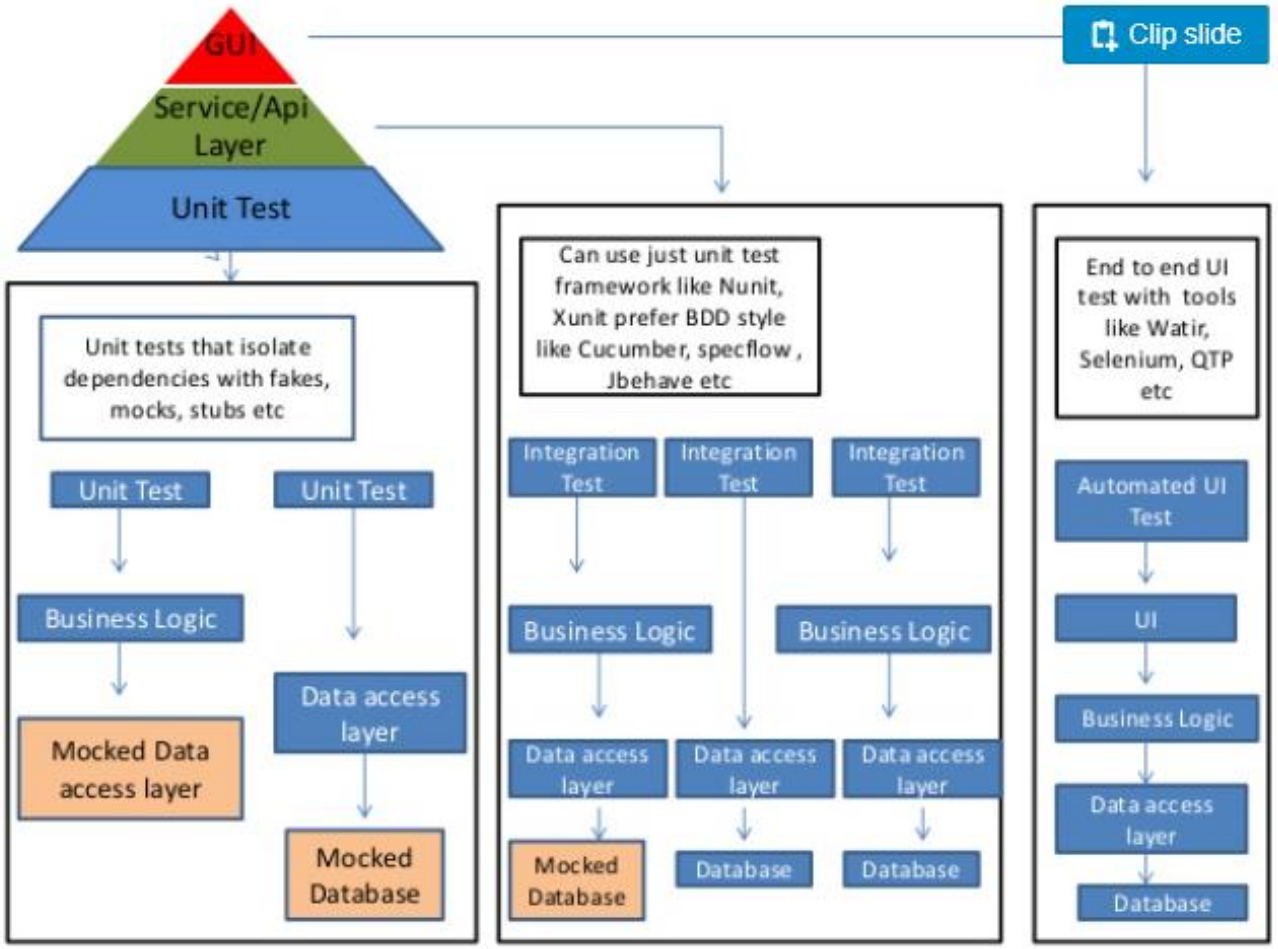
Documentation to support automation (Section-2)

3. Support Automation Transition Document

Things we need to know before execution, Scripting

4. Engage in Automation Framework

Scripting Standards [Eg. Naming conventions, Test-suite name, Object/Variable name]


GUI
Service/Api Layer
Unit Test
Clip slide
Unit tests that isolate dependencies with fakes, mocks, stubs etc
Unit Test
Unit Test
Business Logic
Mocked Data access layer
Data access layer
Mocked Database
Can use just unit test framework like Nunit, Xunit prefer BDD style like Cucumber, specflow , Jbehave etc
Integration Test
Integration Test
Integration Test
Business Logic
Business Logic
Data access layer
Data access layer
Data access layer
Mocked Database
Database
Database
End to end UI test with tools like Watir, Selenium, QTP etc
Automated UI Test
UI
Business Logic
Data access layer
Database

Once you get the pyramid built

Review

- Prioritize Unit and API/Service test automation.
- Limit the number of GUI tests, have a fixed number of stable end-to-end tests.
- It's ok to use multiple technologies.
- It's ok to use Record/Play tools like Selenium IDE for throwaway tests.
- It is better to have 50 stable test cases than 500 fragile ones that breaks regularly.