Basics of Integration Testing

PA1417 Lecture Unit 4

Organizational

1. Lab 1 deadline by the end of this week
2. Lab 2 opens next week
3. 3-week roll call ends today, inactive students will be removed by EOD

   dasv22, liar19, oleg22 - are still looking for lab partners

Objective

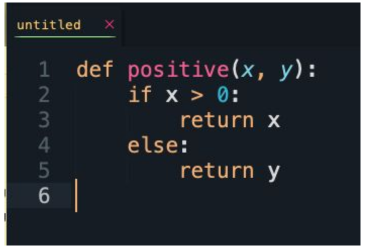
Today, in the first part, you will learn about integration testing and how to go about covering a system with low level tests.

In the second part, we will reiterate some key concepts from earlier lectures

Prepare your questions!

Integration testing

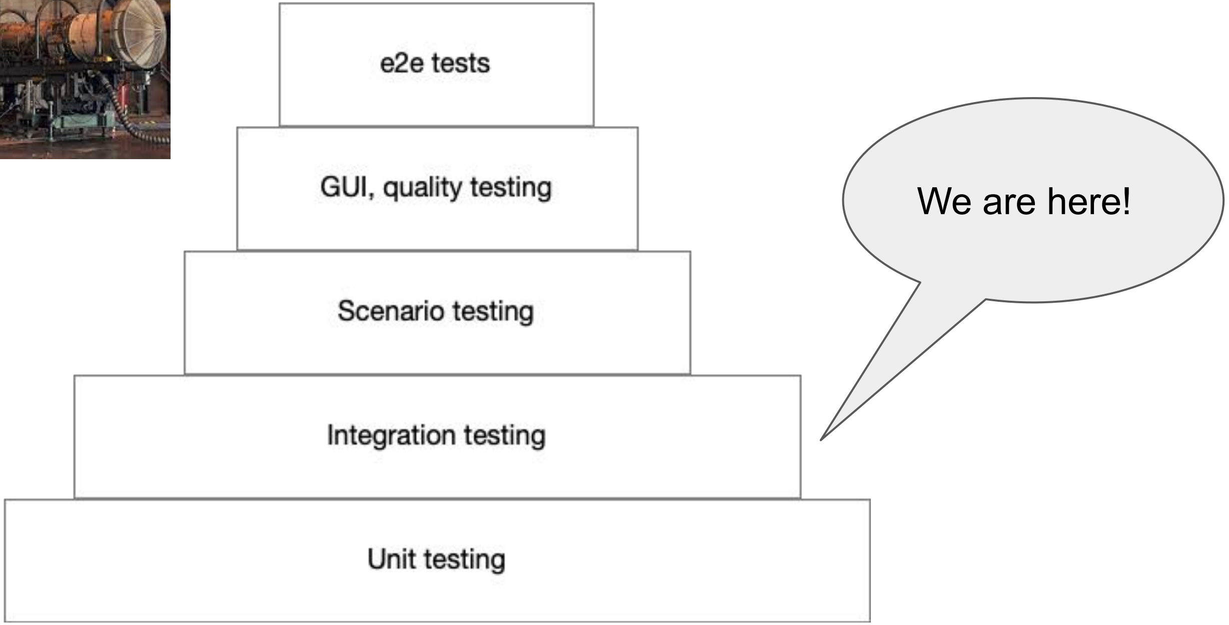
Integration testing is verifies whether individual software modules work together as a group.



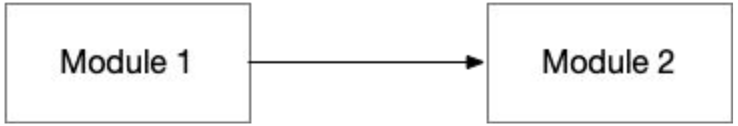
```
untitled

def positive(x, y):
    if x > 0:
        return x
    else:
        return y

```

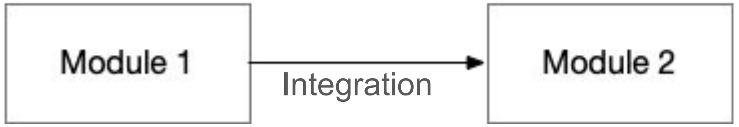


Examples



Two software modules with one-way dependency

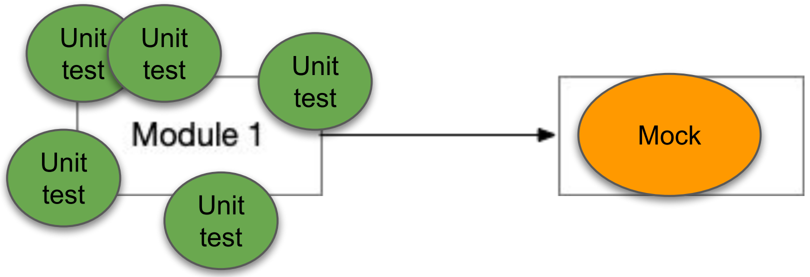
Examples of integration



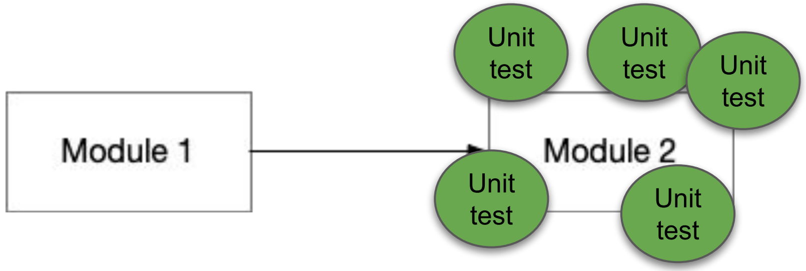
- Web backend & frontend
- Mobile app and backend API
- Reading/writing data to a database
- Reading file from a disk
- Calling Google Weather API
- One microservice calling another

Unit & integration testing

Testing Module 1



Testing Module 2

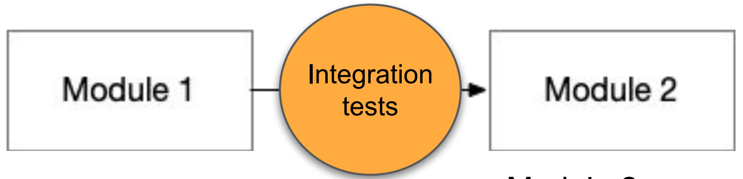


Unit testing verifies each module in isolation

Unit & integration testing

Make sure Module 1 can correctly
invoke Module 2

Module 1 calls Module 2



Module 2 exposes an integration interface (API):

- http endpoints
- sockets/ports
- interfaces
- and so on

Integration testing focus on making sure the two modules know how to talk to each other.

Integration test cases

- Similar to unit tests
- Focus on the communication between pairs of modules
- Mock everything else

Boundary value analysis and *equivalence partitioning* still applies.

Examples:

1. Save and retrieve valid user data from the DB
2. Save and retrieve invalid user data from the DB
3. Try saving/retrieving data when DB is not available

Back-box vs white-box testing

For integration testing you do not need to know how the other system is implemented.

- Focus on the API not the implementation
- Follow specifications

Example: To verify if your application can save results to the database you do not need to know how the DB engine is implemented and unit-test it.

- The DB server is a black-box, you assume it works.
- DB schema, SQL language, .. are your specifications

Causes of issues

Defects in the integration may arise from:

- Wrongly implemented API (client or server)
- Poor error handling
- Incompatible library versions
- Communication disruption
- Poorly communicated changes, one module updated without notifying another

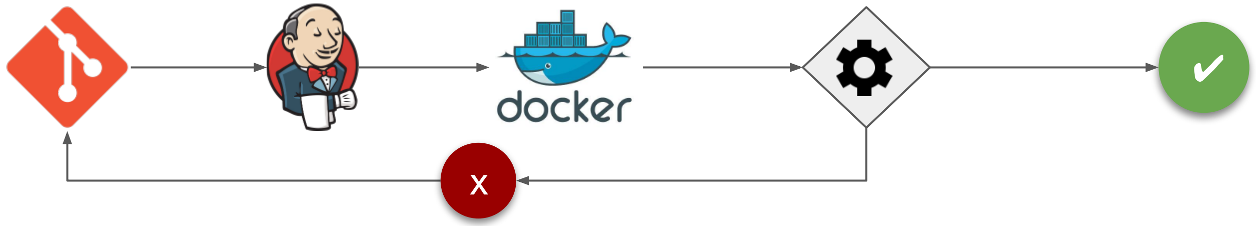
System robustness: Errors need to be handled gracefully! A defect in one module should not crash the whole system.

Continuous Integration

Unit and integration tests are often used in an automated CI/CD pipeline:

Typical process:

1. Developer pushes changes to a repository
2. CI (Jenkins, CircleCI, TeamCity) tool automatically takes the changes, creates test environment (Docker)
3. Builds everything, runs unit + integration tests
4. If successful, the new code is approved and merged to *master*
5. If unsuccessful, the merge is rejected and sent back to the developer along with the test report.



CI pipeline ensures that new changes does not break the system.

Chaos Monkeys

“Imagine a monkey entering a 'data center', hosting all the critical functions of our online activities.

The monkey randomly rips cables, destroys devices and returns everything that passes by the hand [i.e. flings excrement].

***The challenge for engineers** is to design the system so that it can work despite these monkeys, which no one ever knows when they arrive and what they will destroy.”*

Break

Next up: Tutorial 2