# Testing in Layers

http://www.aleax.it/pyconit17_en.pdf

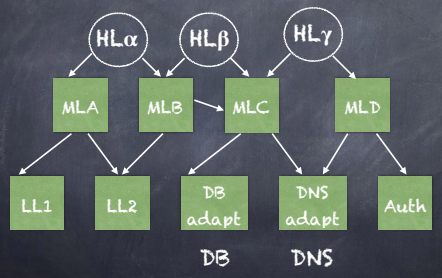
Google





1

# A typical SW system





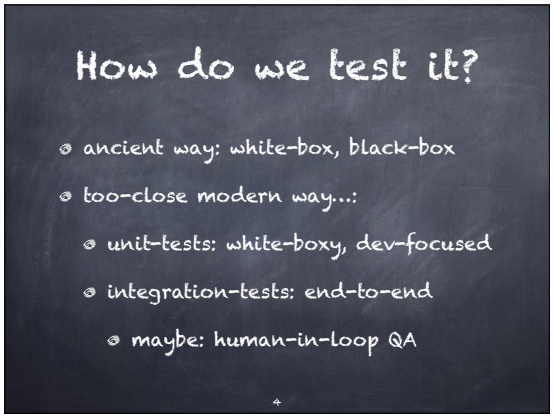

2

# Why test?

- if you need me to explain this...
  - ...then you need my OTHER talks on testing!-) [other people's too]
  - i.e, I'm not covering that today!-)

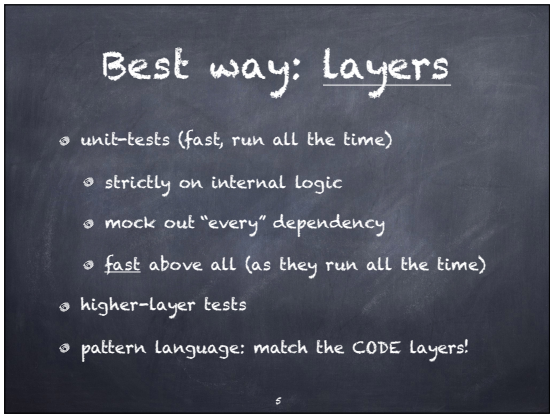


3



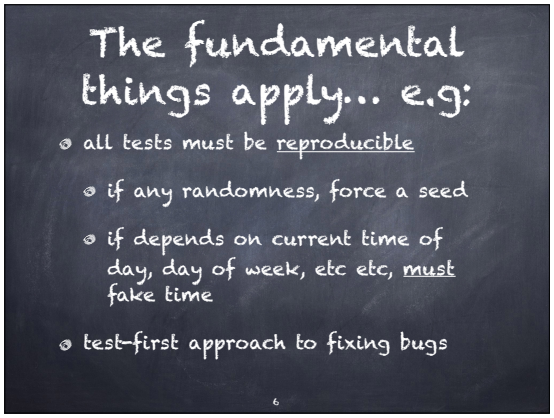

4

QA = Quality Assurance (use a different term than "test", it's TOTALLY different!!!)



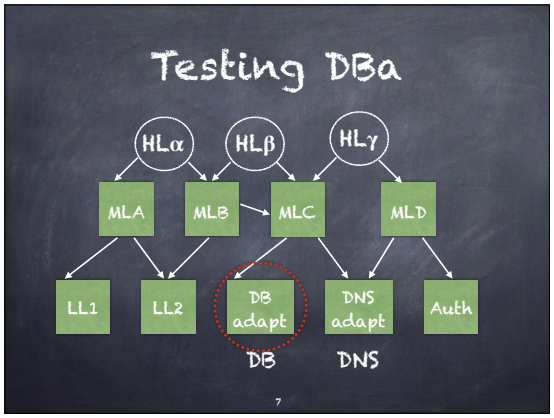

5

If running automatically in the background, fast allows lower latency for feedback;
if not, fast means you'll run tests more often; either way -> higher productivity



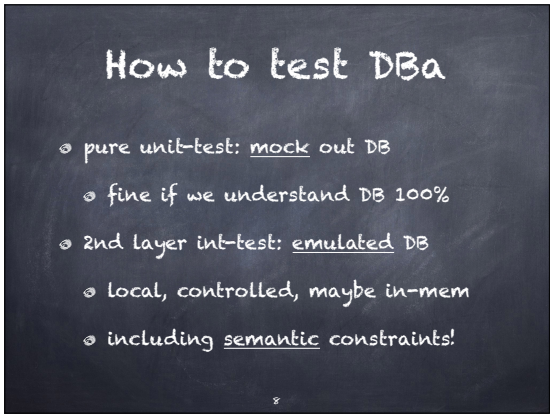

6

…and many other excellent, necessary qualities of automated tests, layered or not.




7

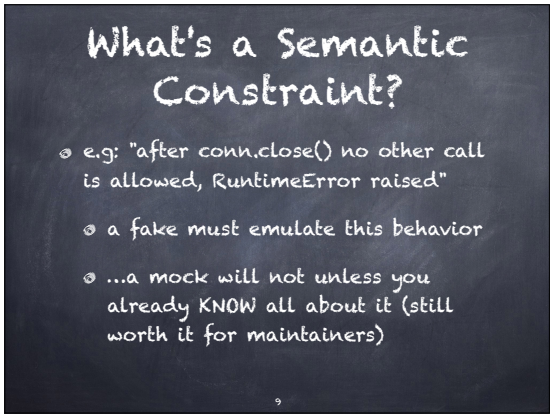
---




8

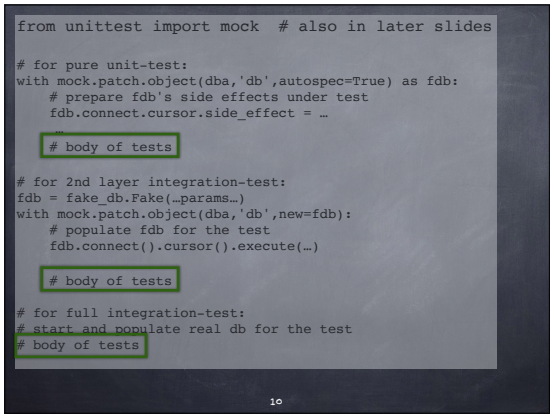
DB = database

---

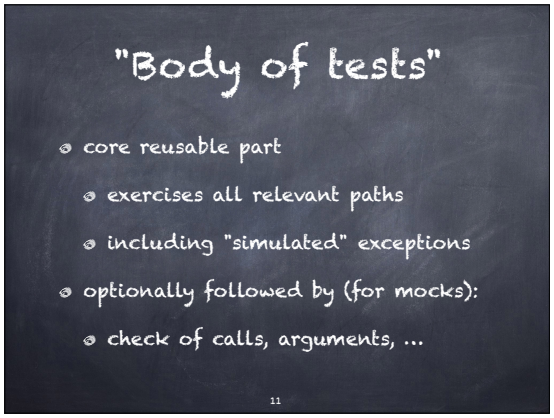



9

A fake may also add constraints such as "DB size < 23 MB" — for test uses, should be OK.




10




11

"Body of tests" constant and reusable; difference among test layers is in the preparation for running the "body of tests" (optionally also in extra checks afterwards for mocks).


12

## Mocks vs Fakes

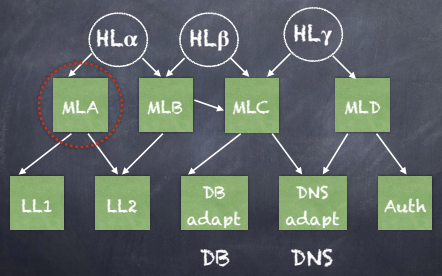
- mock: flexibly emulates anything
  - depending on tests' calls on it
  - lets you check calls, args, ...
- fake: fast, limited simulation
  - of a specific component/module

13

13

Both should also, on request, emulate error situations (e.g. "CPU on fire":-) to check your code handles such disasters gracefully (almost impossible to check w/o simulation!).

## Testing MLA




14

14

## How to test MLA

- pure unit-test: mock out LL1, LL2
  - fastest, mostly fine (team ownership)
- 2nd layer int-test: actual LL1, LL2
  - if fast enough (check w/timeit)...
  - don't need pure unit-test (less work)

15

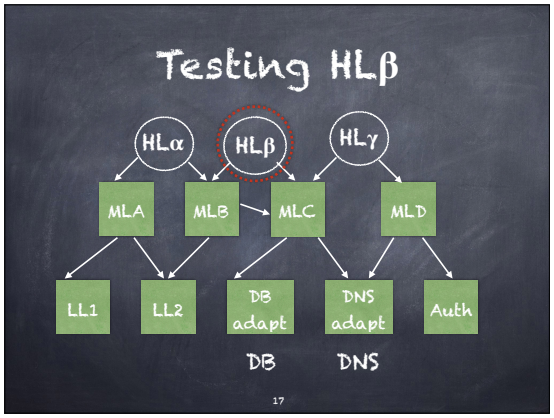
15

If uncertain, try both ways — timeit!

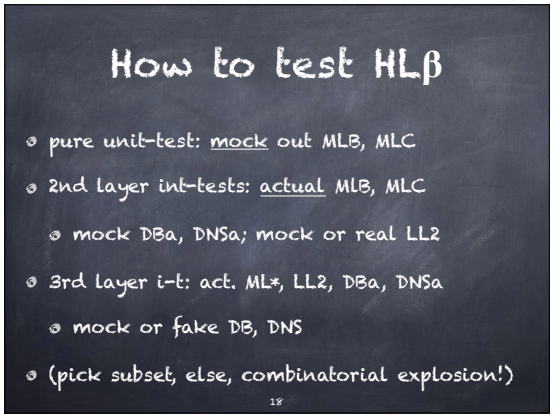
```
# for pure unit-test:
with mock.patch.object(mla,'lll',autospec=True) as f1,\
     mock.patch.object(mla,'ll2',autospec=True) as f2:
    # prepare f1's & f2's side effects under test
    …
    # body of tests

# for 2nd layer integration-test:
# prepare lll's and ll2's for the test
# body of tests

# no further integration-tests in this specific case
```

16

16

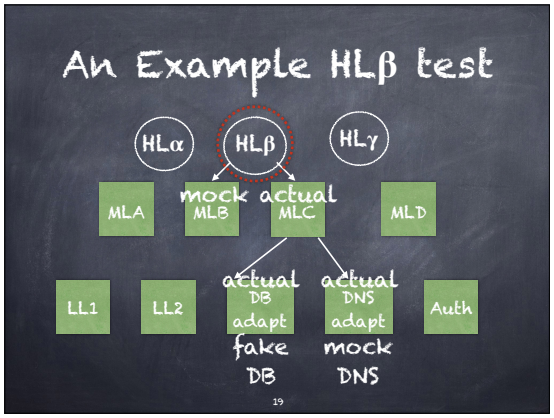



17

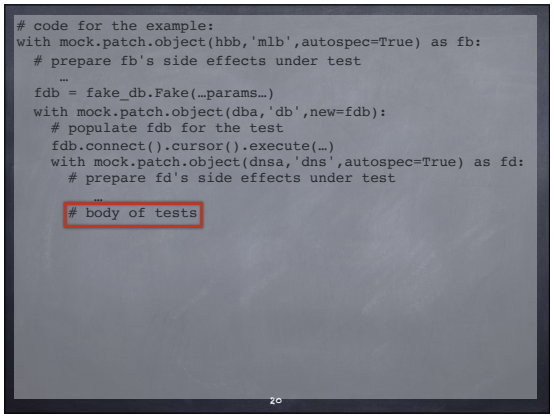



18

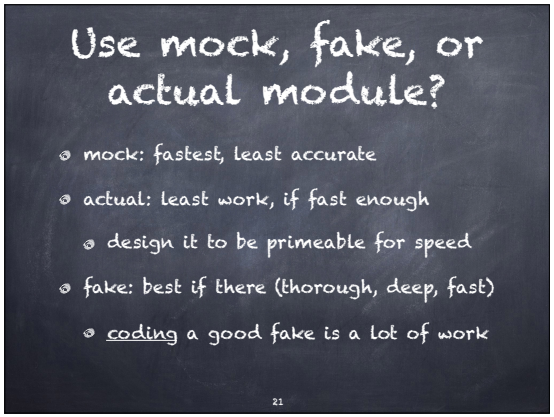
tradeoffs: mocks may be faster; mostly: ownership of test-double, thus, how much detailed/precise understanding of corner cases is needed


An Example HLβ test
HLα
HLβ
HLγ
mock actual
MLA
MLB
MLC
MLD
actual DB adapt
actual DNS adapt
LL1
LL2
Auth
fake DB
mock DNS
19

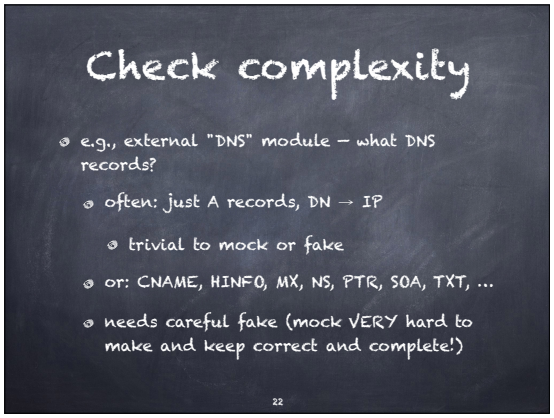
19


# code for the example:
with mock.patch.object(hbb,'mlb',autospec=True) as fb:
# prepare fb's side effects under test
…
fdb = fake_db.Fake(…params…)
with mock.patch.object(dba,'db',new=fdb):
# populate fdb for the test
fdb.connect().cursor().execute(…)
with mock.patch.object(dnsa,'dns',autospec=True) as fd:
# prepare fd's side effects under test
…
# body of tests
20

20

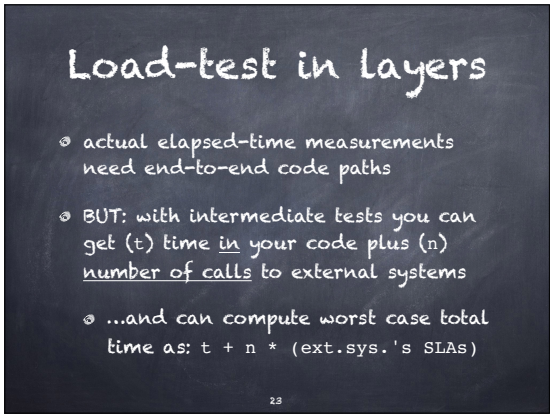

Use mock, fake, or actual module?
mock: fastest, least accurate
actual: least work, if fast enough
design it to be primeable for speed
fake: best if there (thorough, deep, fast)
coding a good fake is a lot of work
21

21



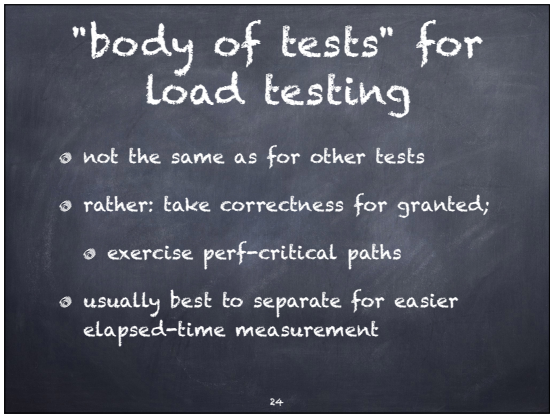

22

DNS = Domain Naming System; once mostly a simple "my.host.com" -> 22.33.44.55 mapping, now (over?)grown into much richer functionality (e.g., TXT records to validate ownership)

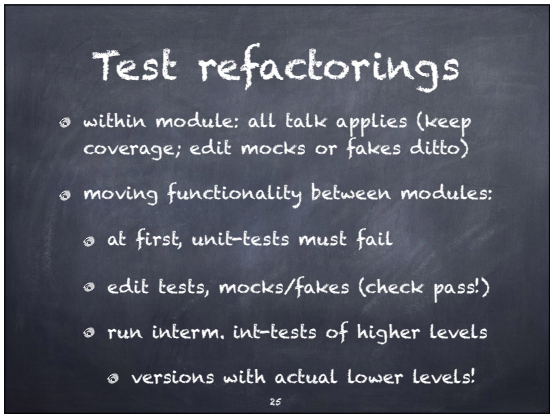



23

SLA = Service Level Agreement (e.g.: "90% of queries answered in < 33 msec")

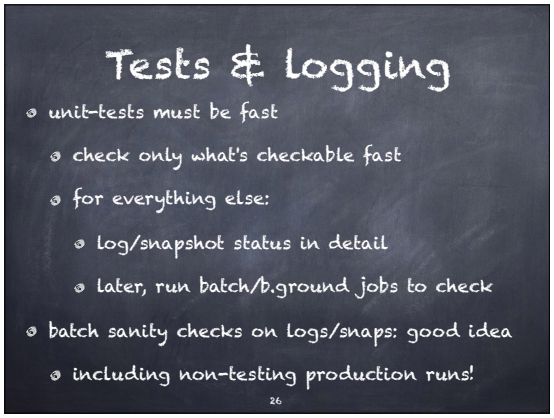



24




25

In a sense, it's automatically a test-driven situation (since tests must exist before refactoring).




26


27