CS1632, Lecture 14: Testing Strategy and the Process of Quality

Bill Laboon

Putting It All Together

- So far, discussed elements in isolation
    - Unit testing
    - Systems testing
    - Performance and Non-Functional Testing
    - Combinatorial Testing
    - Property-based Testing

Test Strategy != Test Plan

- Testing strategy covers all aspects of testing the software, which may or may not include test plans!
  - Roles / responsibilities
  - Defect reporting mechanisms
  - Test tracking mechanisms
  - Testing priorities and schedules
  - Testing enviroment(s)

Evaluating Risk

- Risk (from a QA perspective) is the probability that a software defect will cause a negative impact on the software's stakeholders

- When testing, generally good to take a risk-based approach: how can we find as many important, high-stakes defects as possible before it's released?

- From an overall quality assurance perspective - how can we develop software in a way that these kinds of defects are minimized?

Risk Mitigation != Risk Elimination

- Risk can be mitigated and ameliorated, but all software development has risks
- By determining key aspects of the software and where the software is likely to fail, you can test smarter and not harder

Tools Besides Testing

- Testing is an important but not the only part of QA
- Continuous Integration
- Pair programming
- Code review and inspection
- “User testing” (technically a kind of testing)

Continuous Integration

- Avoid Big Bang integration
- Run tests automatically before merging
  - Or perhaps every time you commit/push
- Finds bugs earlier
- Ensures no failing tests in the master branch

Continuous Integration

- Avoid “Big Bang” integration - merge continuously to mainline
- Run tests automatically before merging
  - Or perhaps every time you commit/push
- Finds bugs earlier
- Ensures no failing tests in the master branch

Pair Programming

- Program with two people using the same computer
- Usually one person “drives” (i.e. types) and the other is “shotgun” (i.e. looks over the other’s shoulders)
- Talking is encouraged! Constructive criticism is encouraged!
- Often one person thinks of edge cases or factors that the other did not
- Research shows that this tends to be slower but produces much higher-quality code with fewer defects

Code Inspections / Reviews

- Looking over others' code to determine if it is optimal or room for improvement (almost always the latter)

- Almost always used in non-trivial size engineering teams

- Poor algorithms, bad naming choices, not using built-in libraries, etc.

User Testing

- Allowing users to use the product and give feedback
- A great way to course-correct
- Avoids incorrect assumptions, poor usability, allows you to find difficult areas
- Provides a truly independent perspective on the software

Just Like In Programming, Scale Is Where It Gets Difficult!

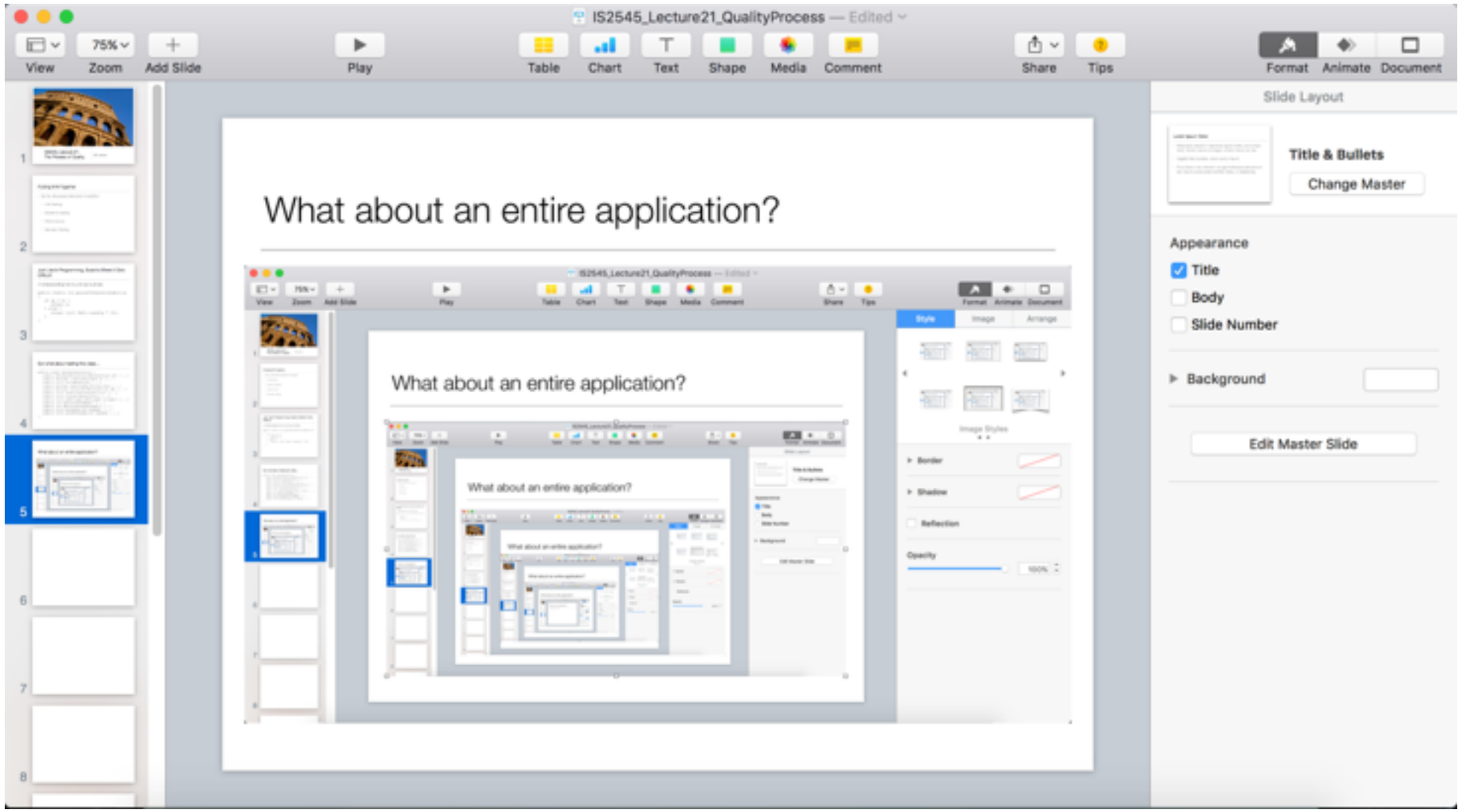
// Understanding how to unit test is simple:

```
def pounds_to_ounces(p)
    if (p < 0)
       0
    else
       (p * 16).round
    end
end
```

But what about testing this class…

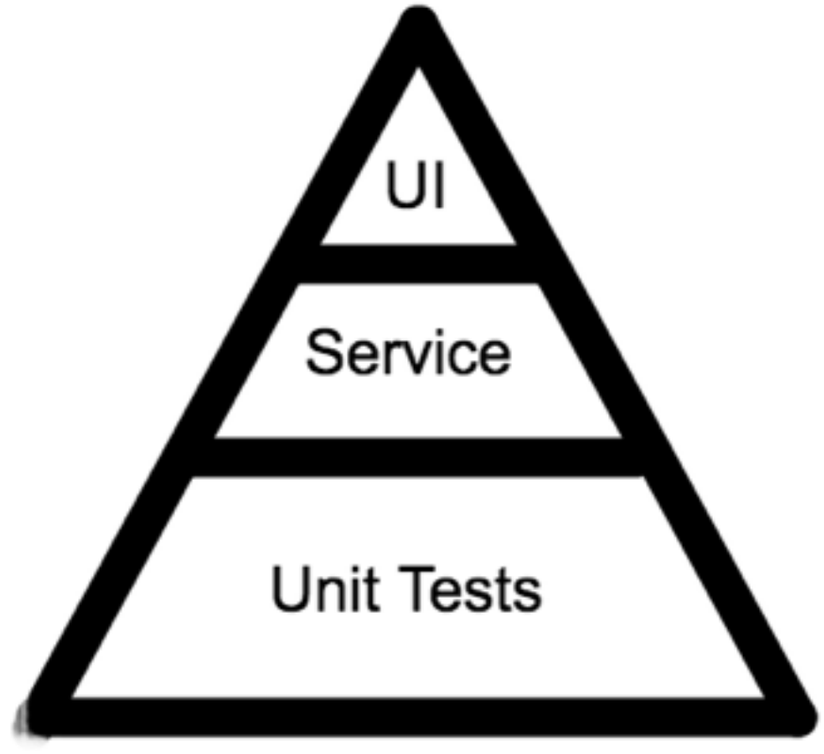
```
class DatabaseConvertor
   def initialize(db_connection) . . .
   def insert_data(data) ...
   def force_reshard  . . .
   def execute_sql(sql_string) . . .
   def select_db(db_connection) . . .
   def revert_last_transaction . . .
   def unrevert_reversion . . .
   def add_trigger(table, trigger) . . .
   def current_db_num . . .
   def current_db_threads() . . .
   def set_db_num(new_num) . . .
   def set_db_threads(new_num_threads) . . .
end
```

What about an entire application?



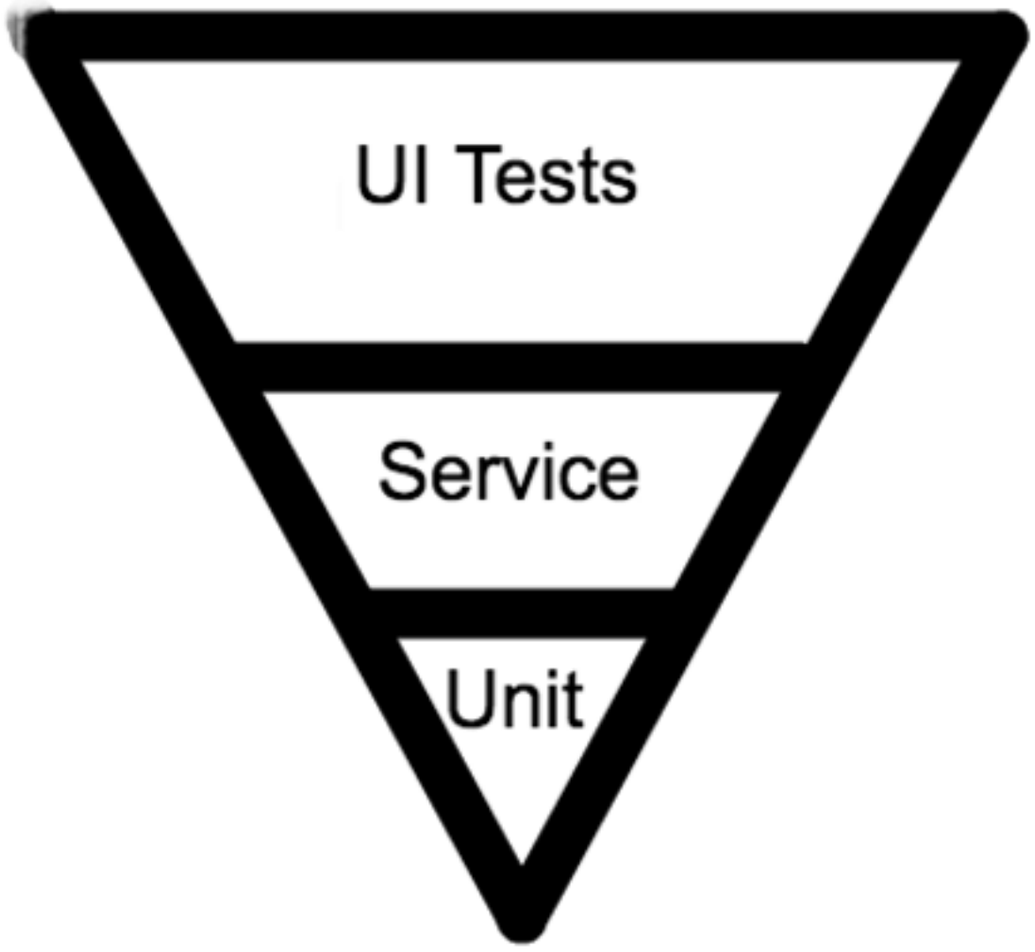
The Testing Pyramid

- 10% UI Tests
    - Tests which check the whole system end-to-end
- 20% Service (Integration) Tests
- 70% Unit Tests



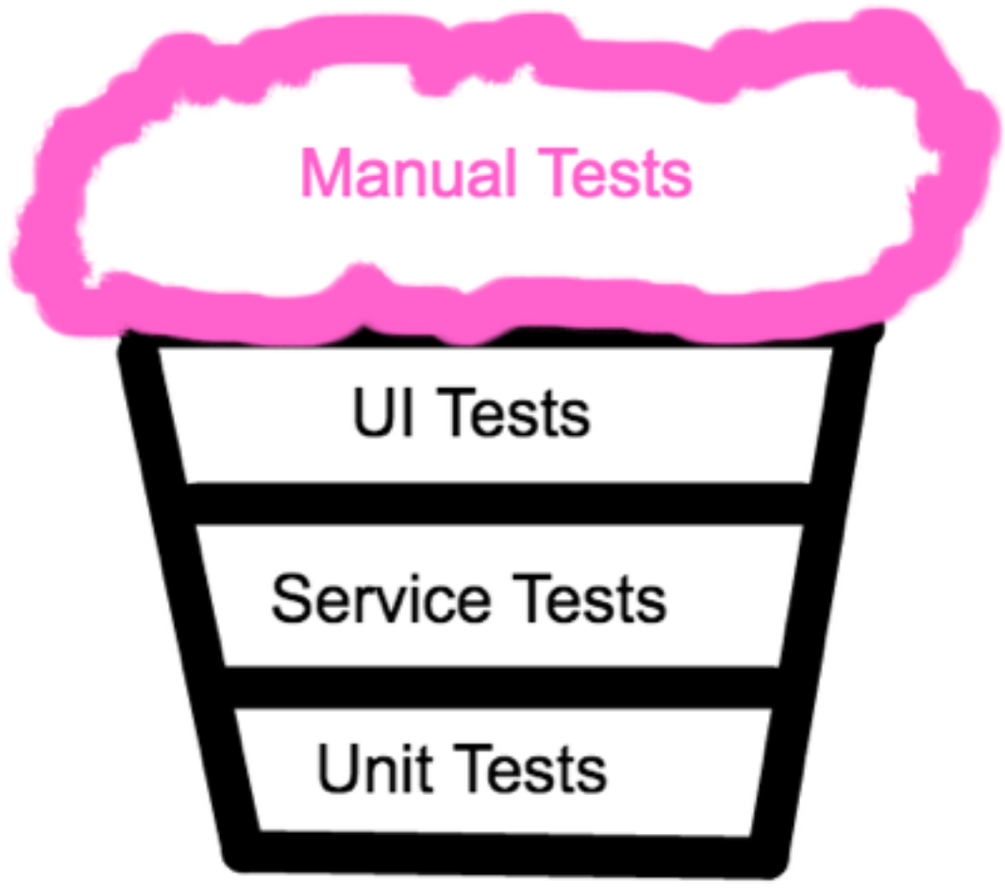
Ice Cream Cone Anti-Pattern

- Few Unit Tests, Some Service Tests, Many UI Tests



Cupcake Anti-Pattern

- Lots of Manual Tests On Top



What About Other Kinds of Tests?

- These will often fit in one of the kinds of test categories
- Other kinds of tests are done on as-needed basis and the amount necessary will vary by domain
- Test coverage and weighting will vary by domain
    - Other tests/quality processes might also be necessary! Usability, formal verification, code metrics, etc.

Example: Performance Testing

- Full Application performance testing: UI test
- Component testing: service test
- Performance of a function: unit test

Example: Security Testing

- Penetration testing: UI test
- Testing a subsystem for injection attacks: service test
- Checking against buffer overflow in a method: unit test

Example: Combinatorial Testing

- Checking OS / Browser / RAM combinations: UI tests

- Combinations of Microservices Loading: Service tests

- Checking for results of boolean args: Unit tests

Designing a Testing Strategy

- Think about what are the priorities of this application
- Where would defects be worst?
- What are likely issues?
- Which have been found by similar applications?
- What tools will be useful?
- What is our team like? Do we need outside help?

Case Study:
Rent-a-Cat, Inc.

- Phone-based app which brings cats for rental right to your door
- All payments handled by separate process
- Some personal data on customers stored
- Hoping for lots of people to use, potential for viral growth, but revenue-positive now (i.e., it's making money)
- Running entirely in LOLCODE. Engineers are relatively new
- Medium-size QA team

Case Study:
Big Bank, Inc.

Big Bank, Inc.

"Money is our business"

- Develop app to access and display bank account information
- Provides access to user accounts, along with all data on user
- People need to use their app - what are they going to do, use Bitcoin?
- Running in tried-and-true Java
- Very large QA team
- Lots of regulatory necessities

Case Study: facedinspace

- Brand new start-up focusing on creating a social network that's Facebook + MySpace + LinkedIn
- No monetary information given by users, but data sold to any advertiser who comes in off the street
- Key metric is user growth - burning through investor cash at over $1 million per day
- Developed with Ruby on Rails
- Very small QA team compared to number of developers
- Very few regulations, registered in Sealand