CS1632, Lecture 22: 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
  - Security (soon!)

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

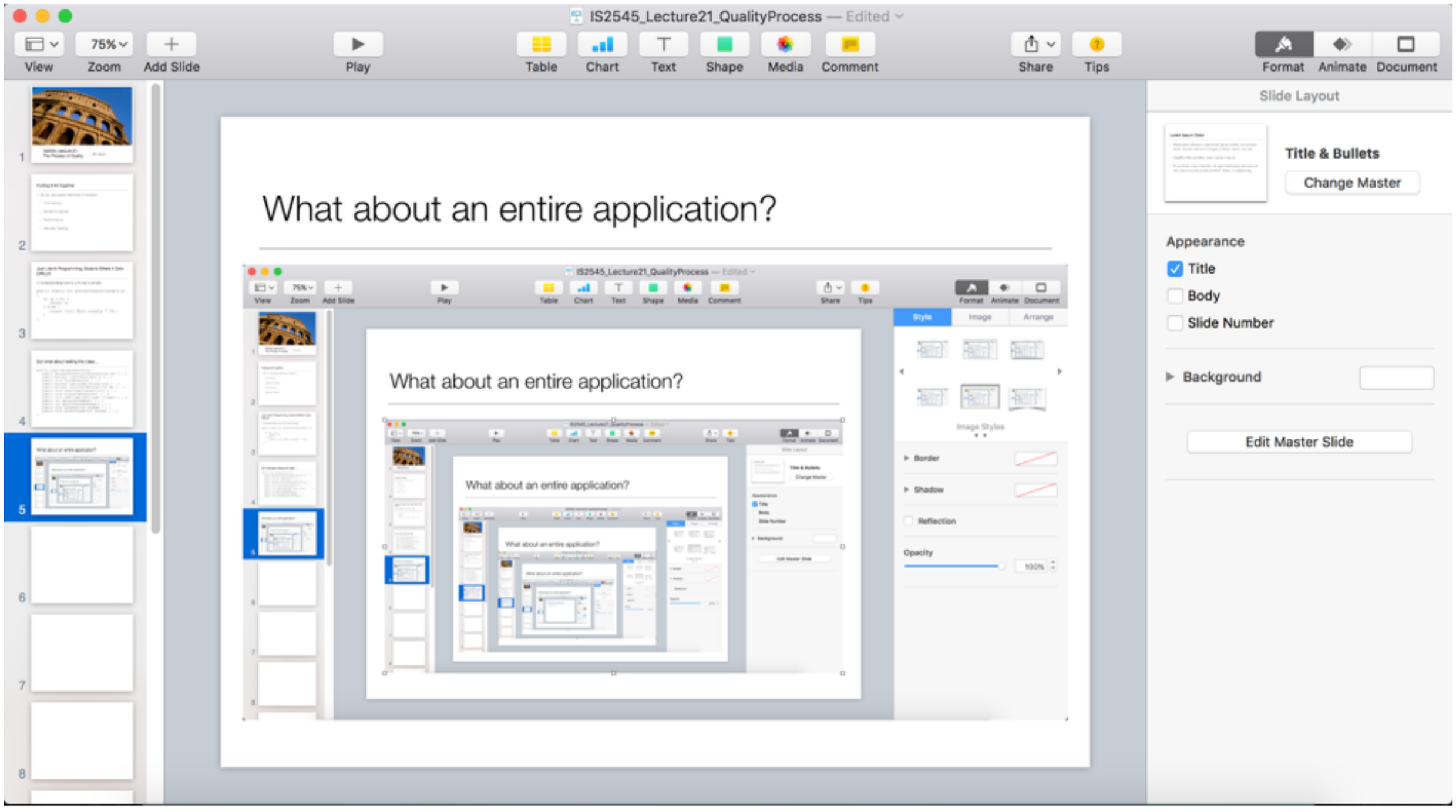
// Understanding how to unit test is simple:

```
public static int poundsToOunces(double p)
{
    if (p < 0) {
        return 0;
    } else {
        return (int) Math.round(p * 16);
    }
}
```

But what about testing this class…

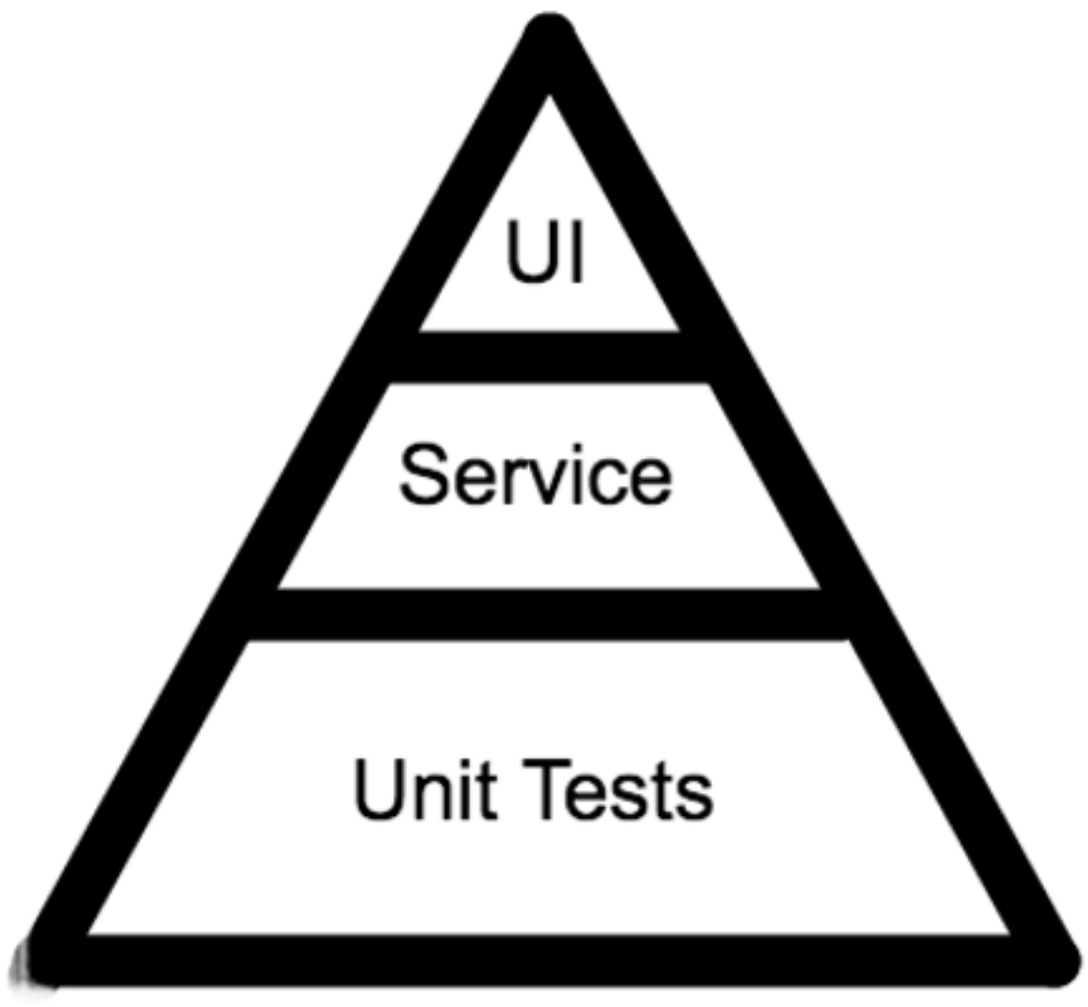
```
public class DatabaseConvertor {
    public DatabaseConvertor(DbConnection db) { … }
    public boolean insertData(Data d) { … }
    public void forceReshard() { … }
    public boolean executeSql(String sql) { … }
    public boolean selectDb(DbConnection db) { … }
    public void revertLastTransaction() { … }
    public void unrevertReversion() { … }
    public void addTrigger(DbTrigger trigger) { … }
    public int getCurrentDbNum() { … }
    public int getCurrentDbThreads() { … }
    public void setDbNum(int newNum) { … }
    public void setDbThreads(int newNum) { … }
}
```

What about an entire application?



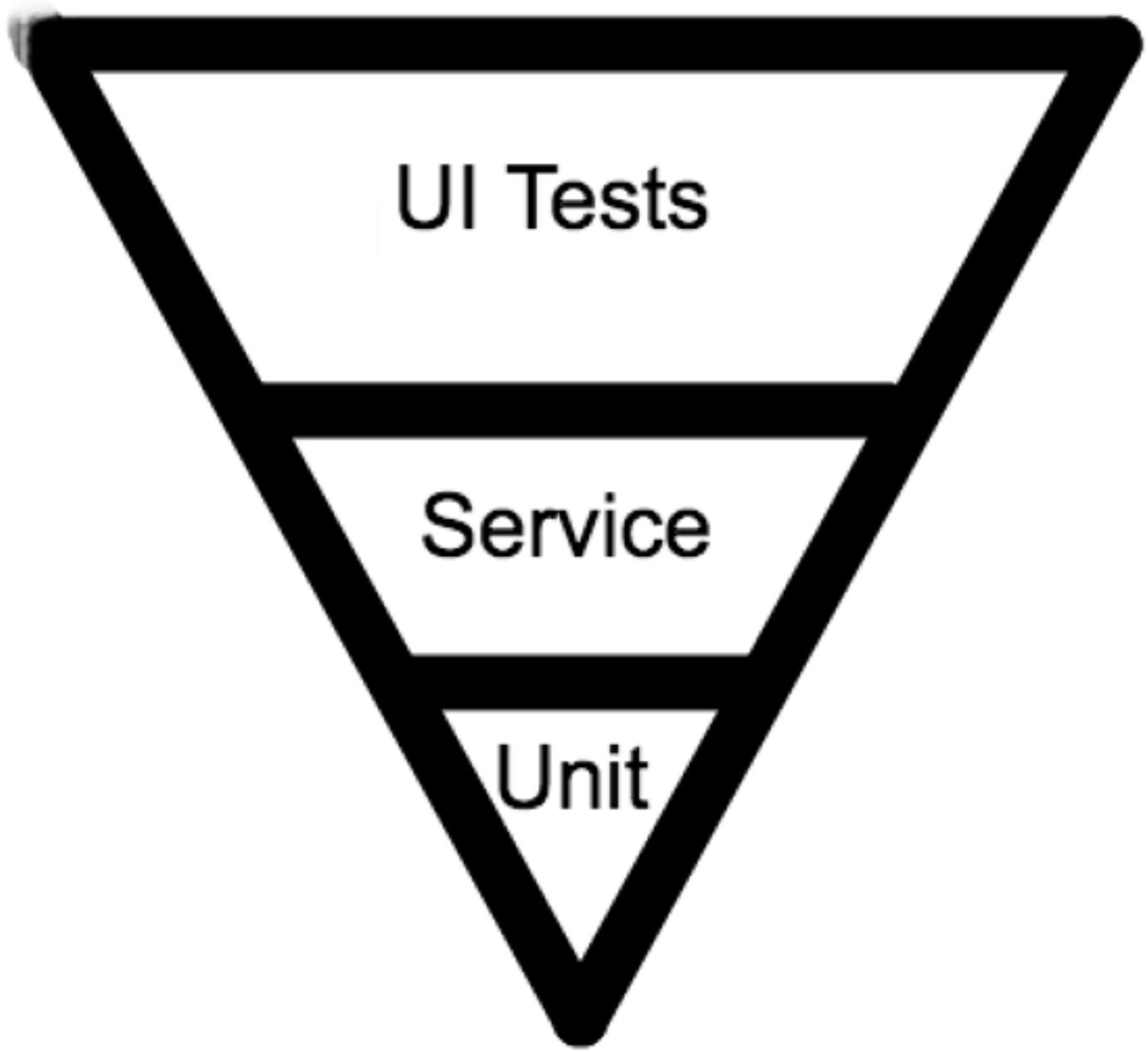
The Testing Pyramid

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



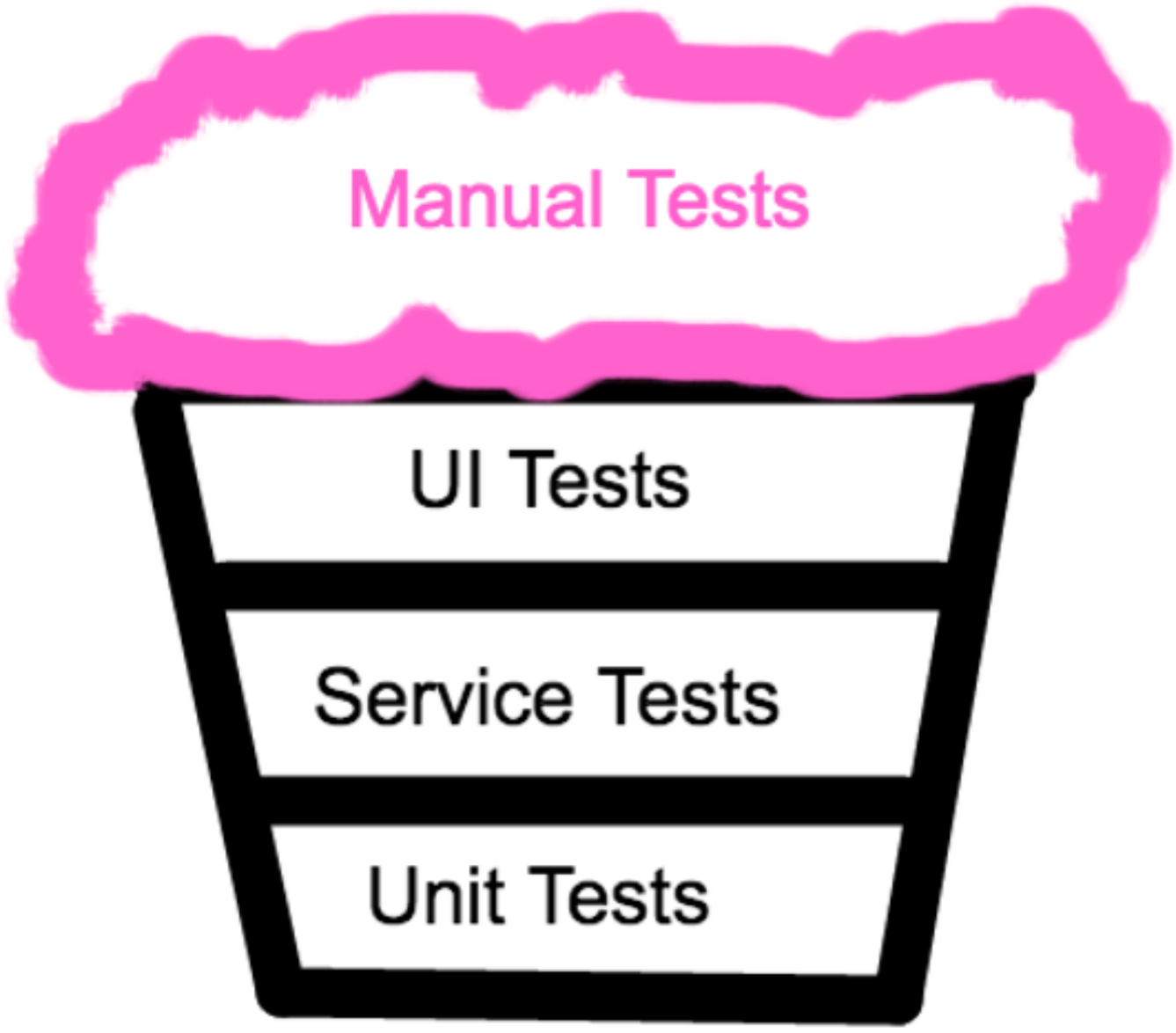
Ice Cream Cone Anti-Pattern

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



Cupcake Anti-Pattern

- Lots of Manual Tests On Top



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

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?

Path Testing

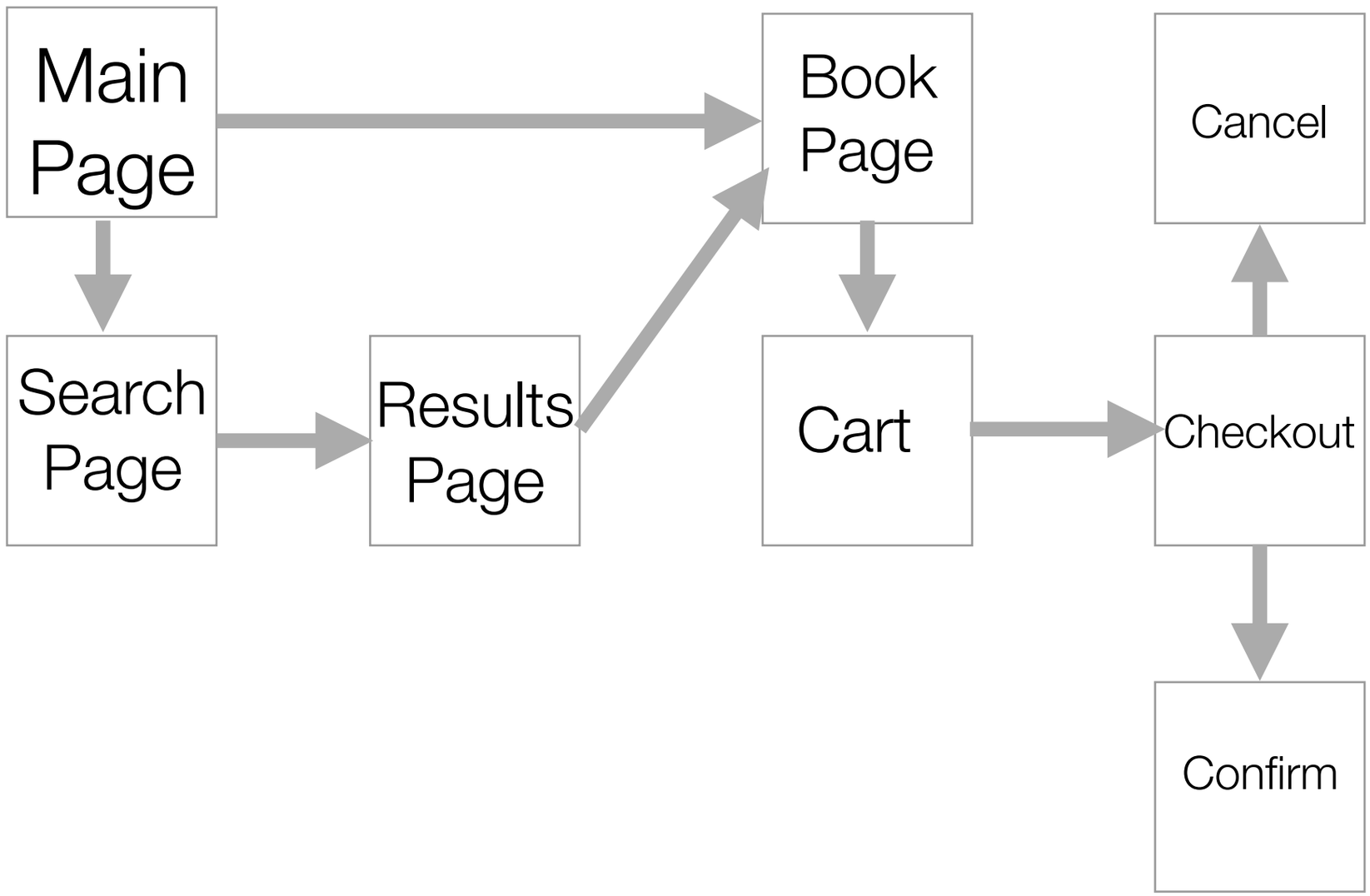
- Test that all possible paths through the program are executed

- Can be done at method or code level, or at a high level

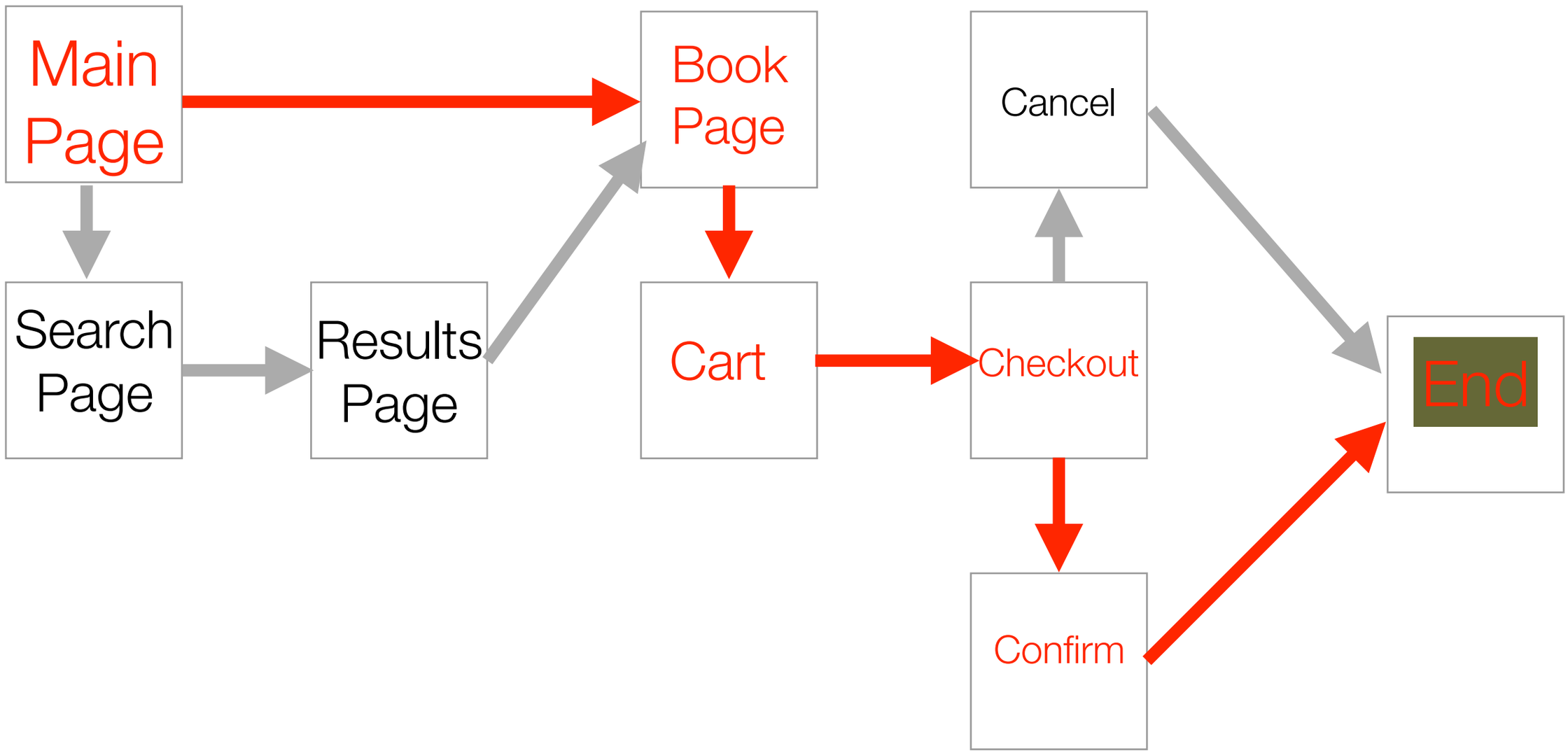
- McCabe cyclomatic complexity tells you how many paths there are (E - N + 2p)

- Remember flowcharts? They're back!

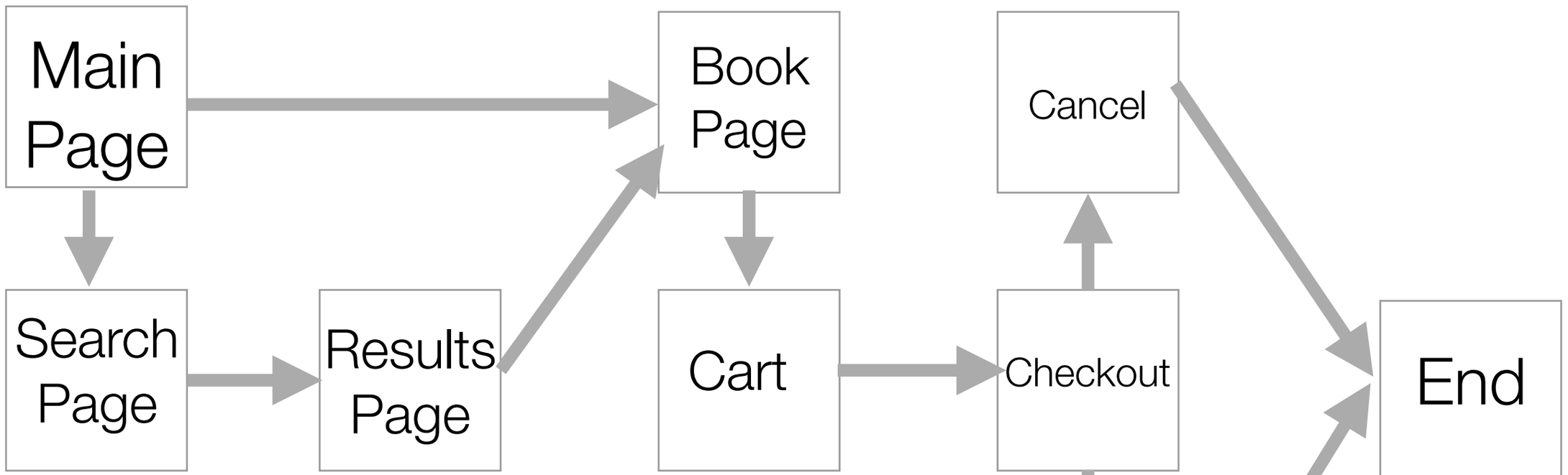
Path Testing



Path Testing



Path Testing



Four paths:

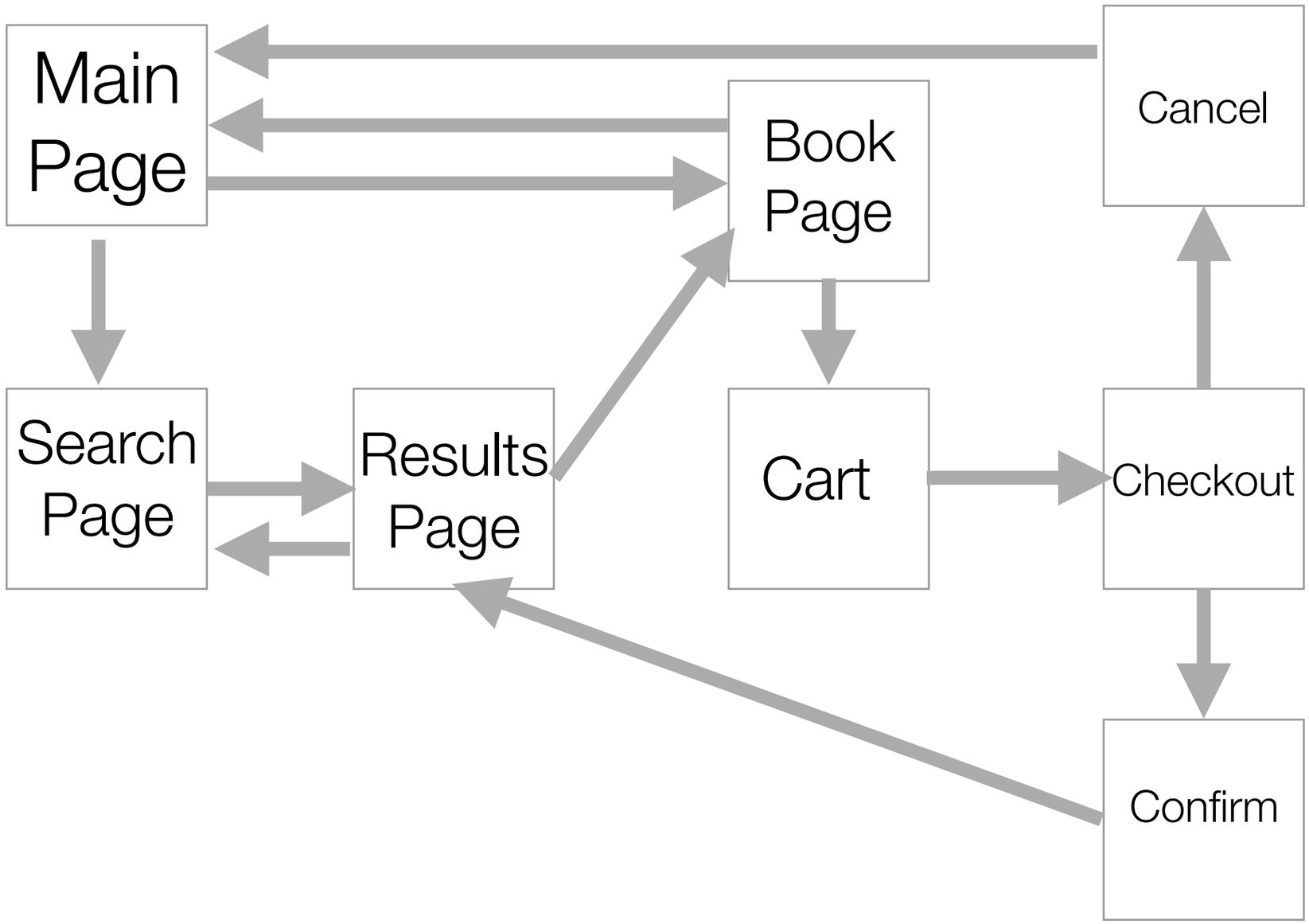
MP > BP > Cart > Ch > Ca > End

MP > SP > RP > BP > Cart > Ch > Ca > End

MP > BP > Cart > Ch > Co > End

MP > SP > RP > BP > Cart > Ch > Co > End

Path Testing Can Be Complicated / Infinite!



Case Study:
Rent-a-Cat, Inc.

- Phone-based app which brings cats for rental right to your door
- All payments handled in person by separate process
- No personal data on customers stored!
- Hoping for lots of people to use, potential for viral growth
- Running entirely in LOLCODE. Engineers are relatively new
- Small QA team - need big bang for the buck

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