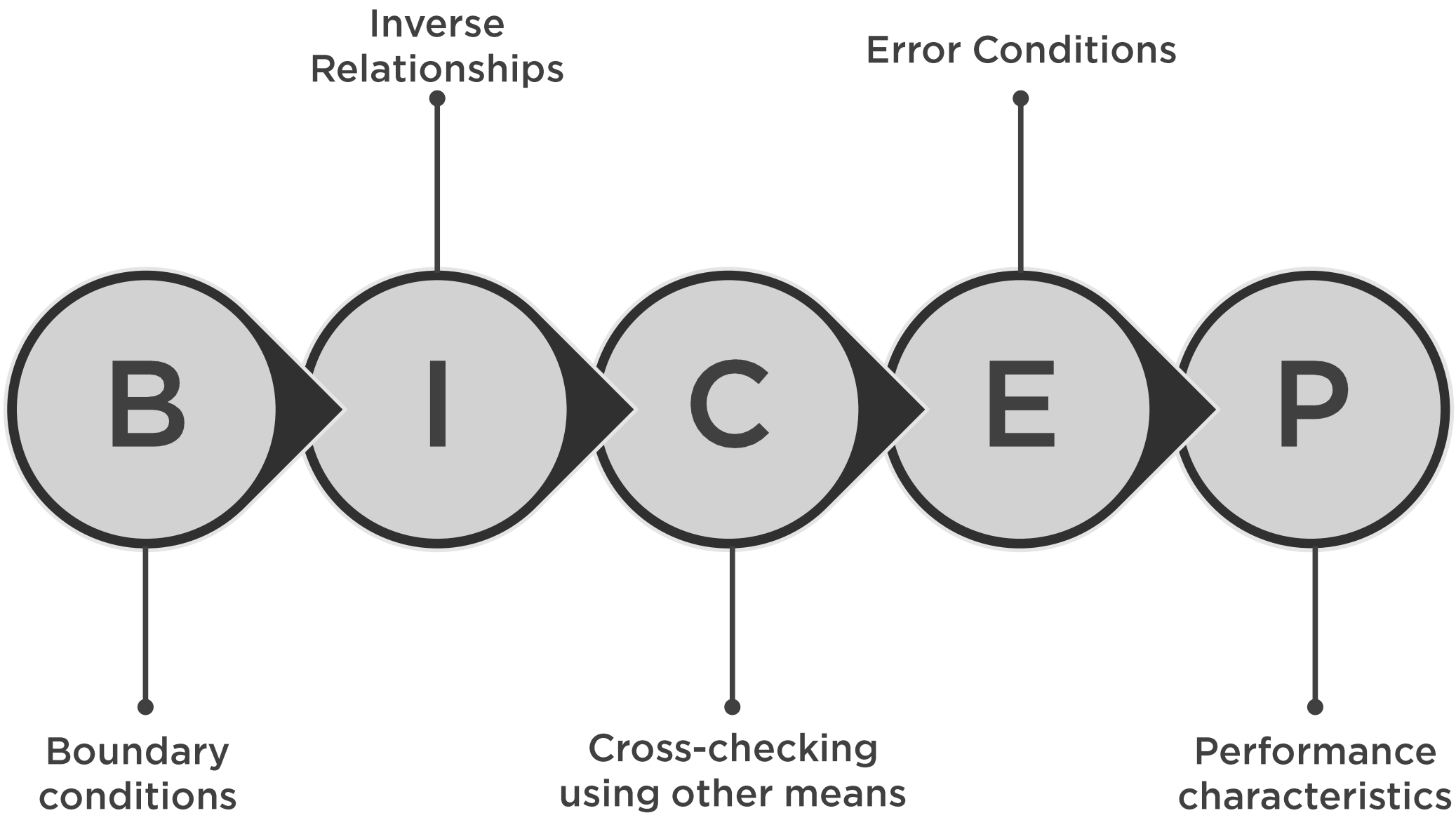
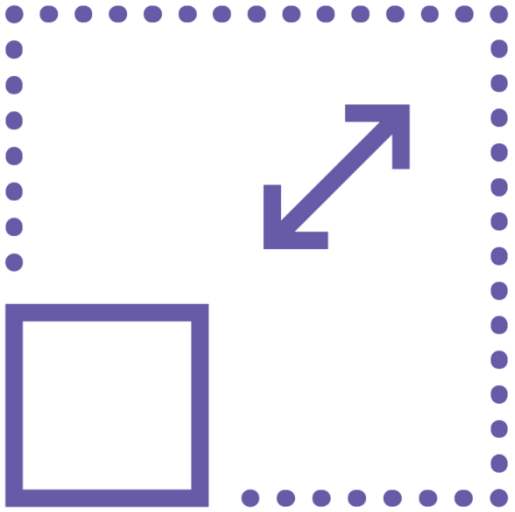
Leveraging BICEP Principles

Andrejs Doronins
TEST AUTOMATION ENGINEER


Inverse Relationships
Error Conditions
B
I
C
E
P
Boundary conditions
Cross-checking using other means
Performance characteristics



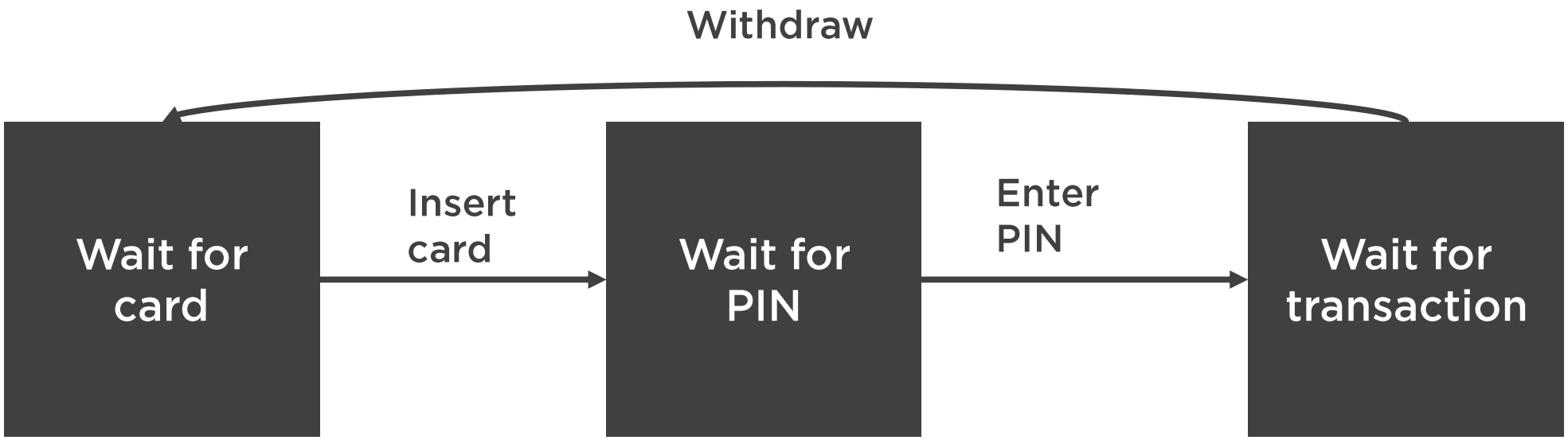
Boundary Conditions

Happy Path

Edge Cases

Happy Path

A default scenario featuring no exceptional or error conditions. It's a scenario when the user or other systems do everything as you expect them to.



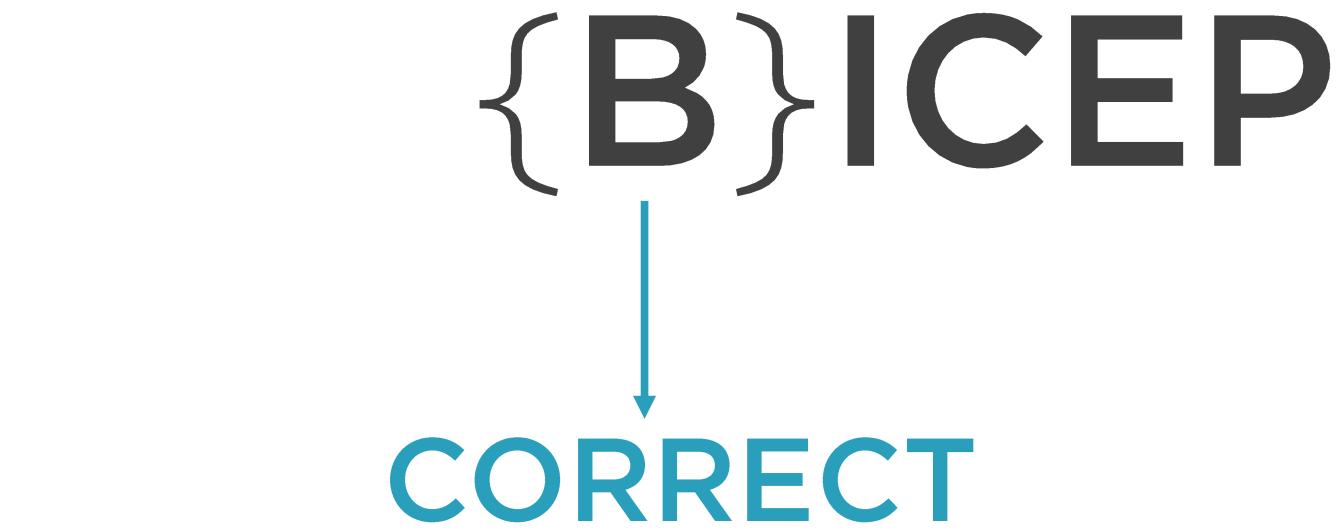
What if... ?

- Wrong PIN? Three times?
- Force pull the card out?
- Try to withdraw 0?
- Try to withdraw too much?
- Take the cash, but forget the card?

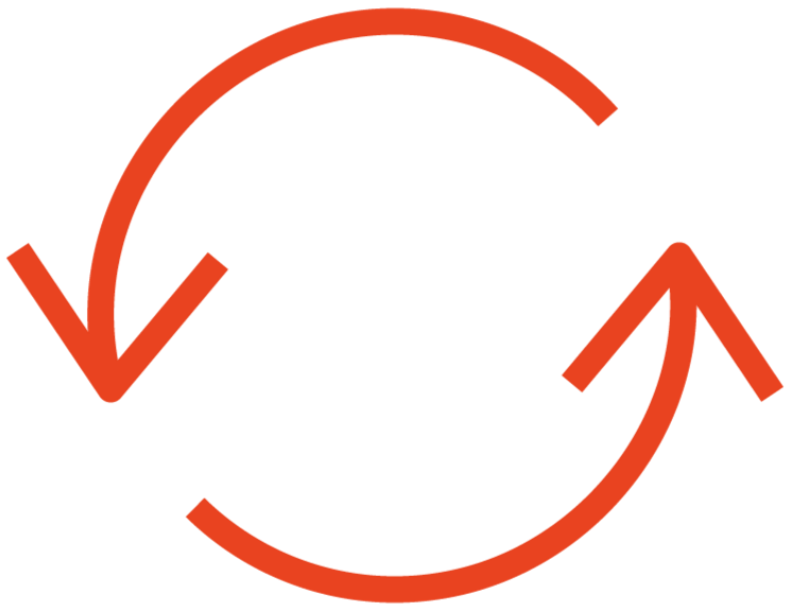
Better Boundary Scenarios

Deep understanding of the domain and SUT

Universal and common scenarios


{B}ICEP
CORRECT

Checking Inverse Relationships

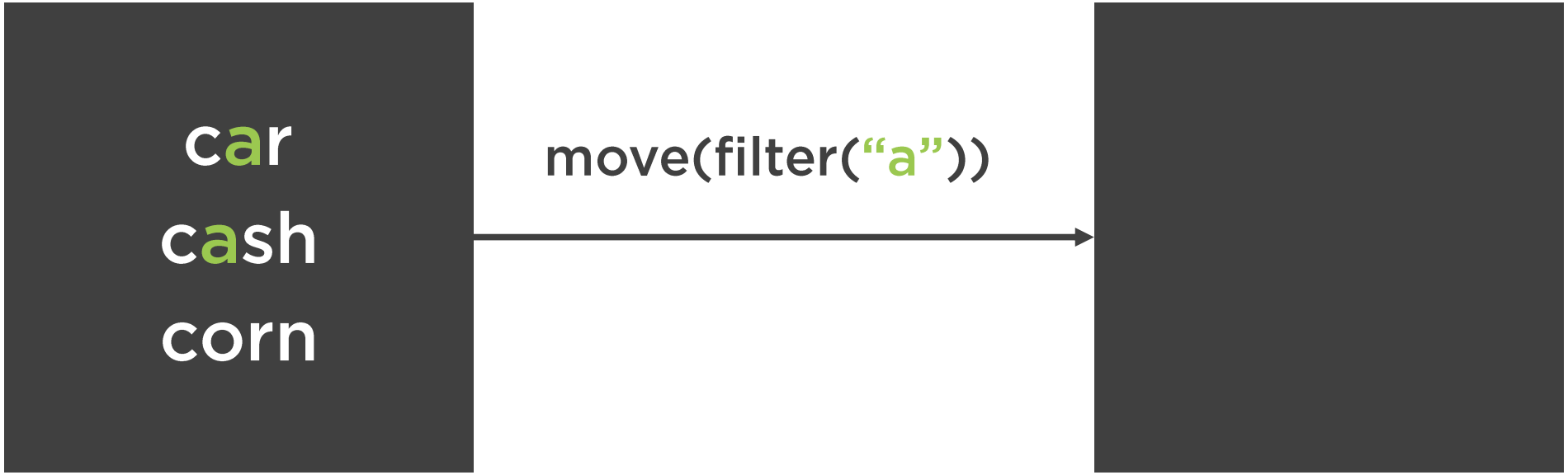


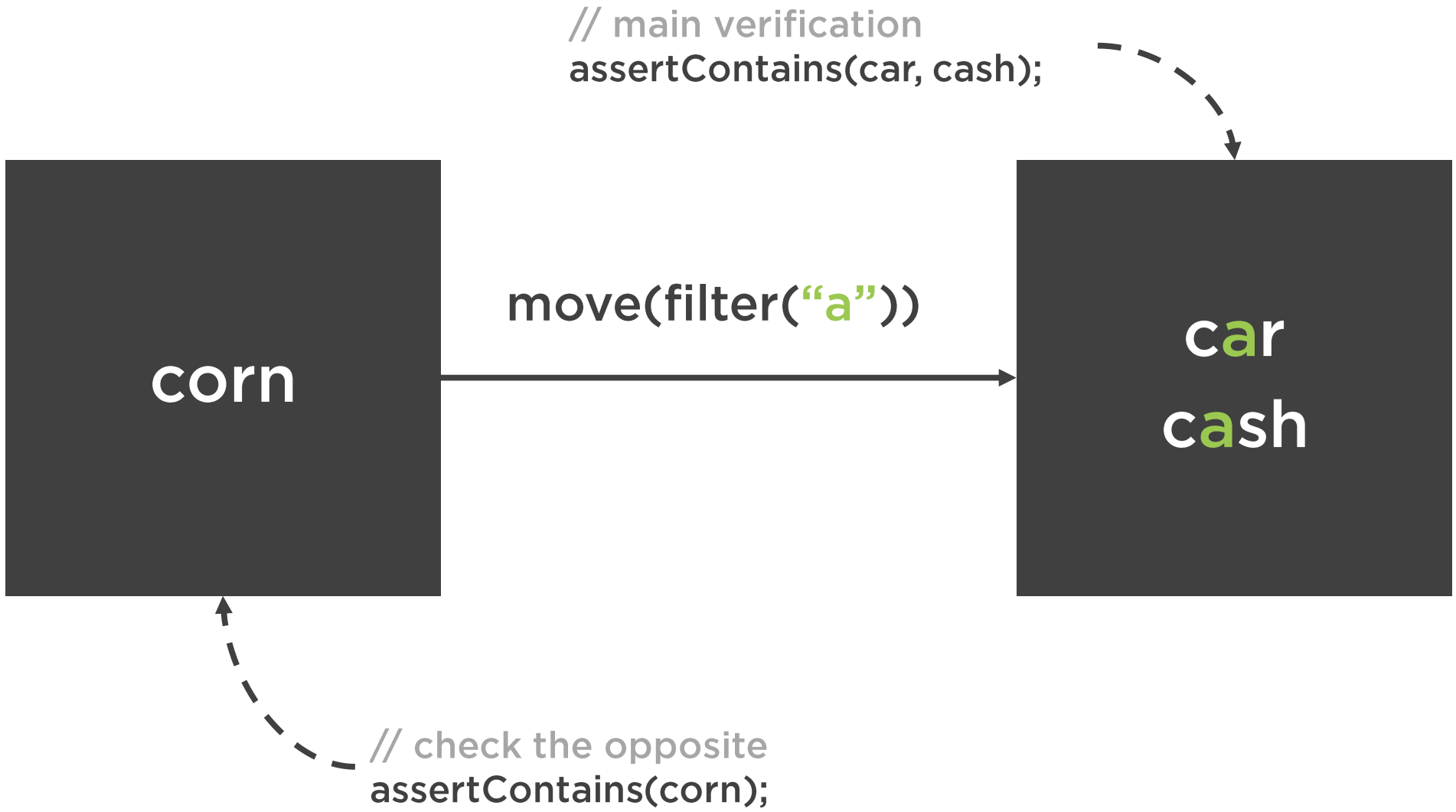
Math:

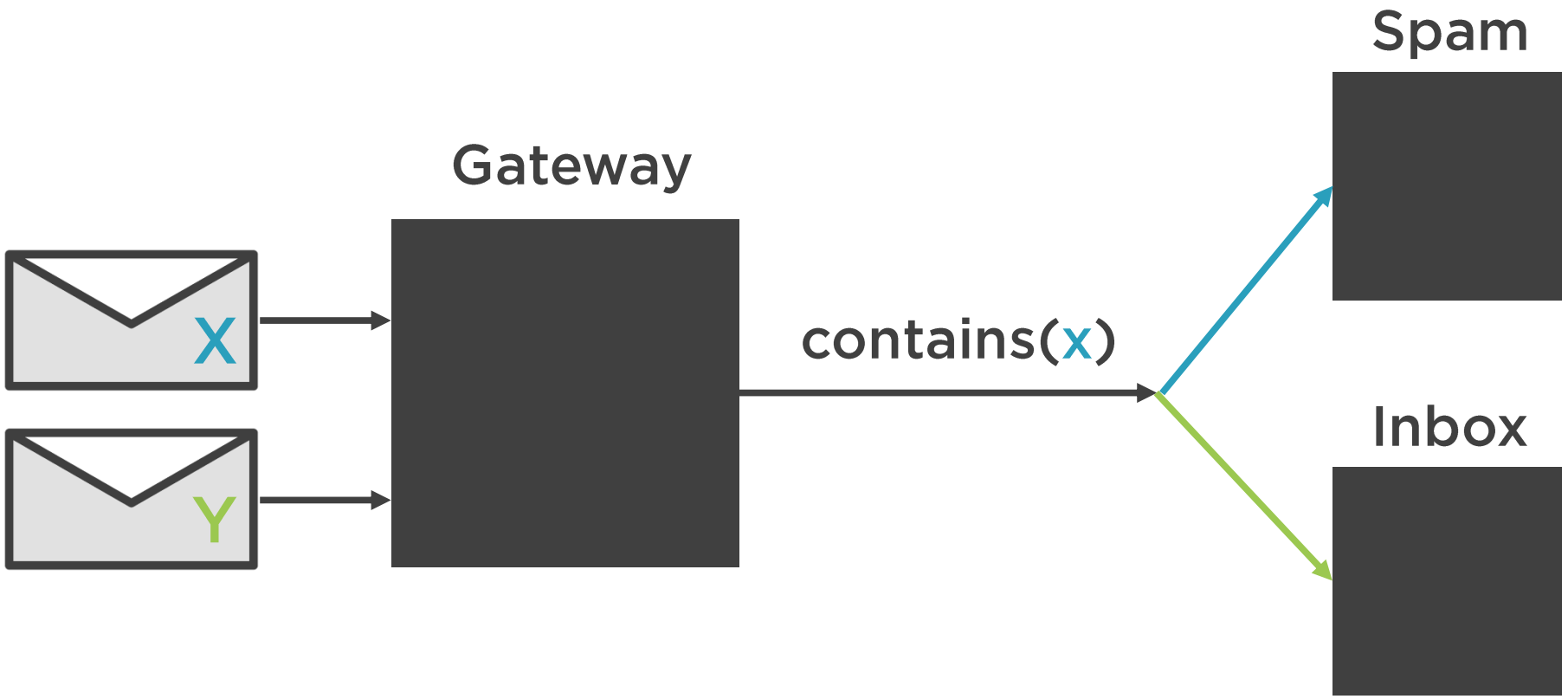
```
int result = 2 + 2;
int result2 = 4 - 2;
assertEquals(result, result2);
```

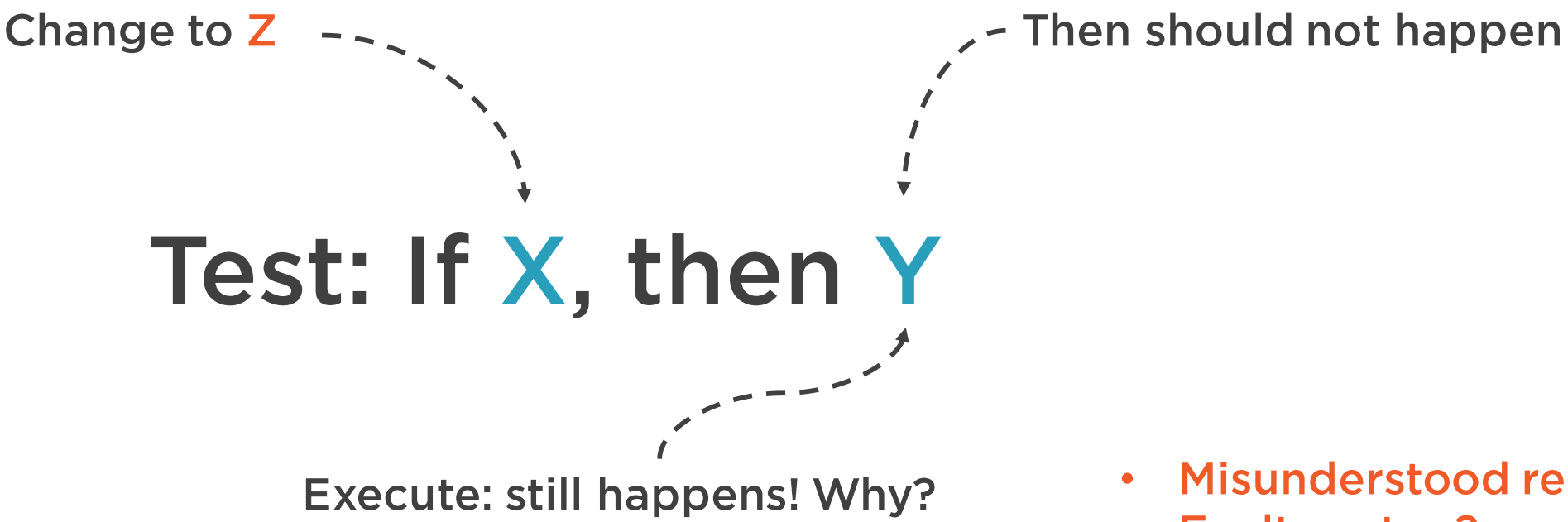
Negative Testing (the opposite):

- uncover bugs
- uncover faulty or useless tests


car
cash
corn
move(filter("a"))


// main verification
assertContains(car, cash);
corn
move(filter("a"))
car
cash
// check the opposite
assertContains(corn);


Spam
Gateway
X
contains(x)
Y
Inbox



- Misunderstood requirements?
- Faulty setup?
- A test that always passes proves nothing!

See the test both pass and fail

(automatically adhered to with TDD)

Checking Inverse Relationships

If I...

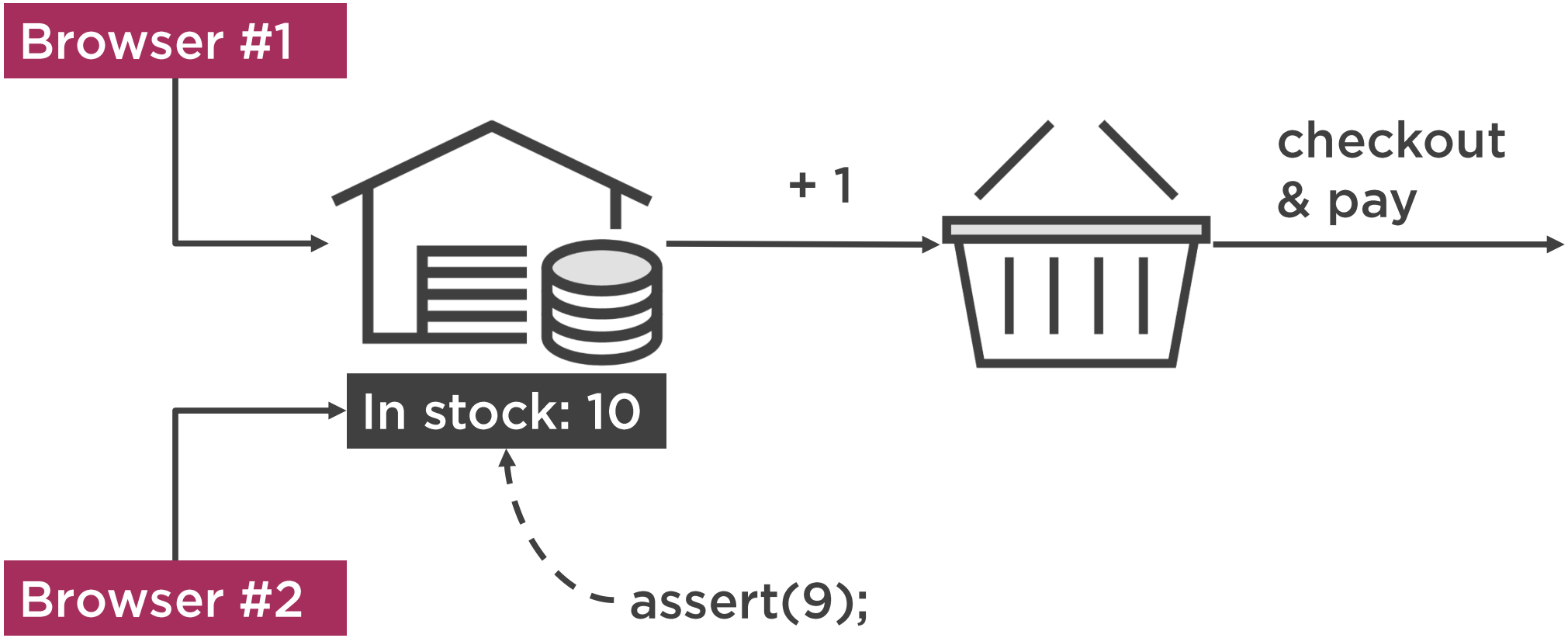
- ... reverse or undo the operation - will it bring me back to the exact original state without side-effects?
- ... execute an operation on A and B, will C and D remain unaffected?

Cross-checking

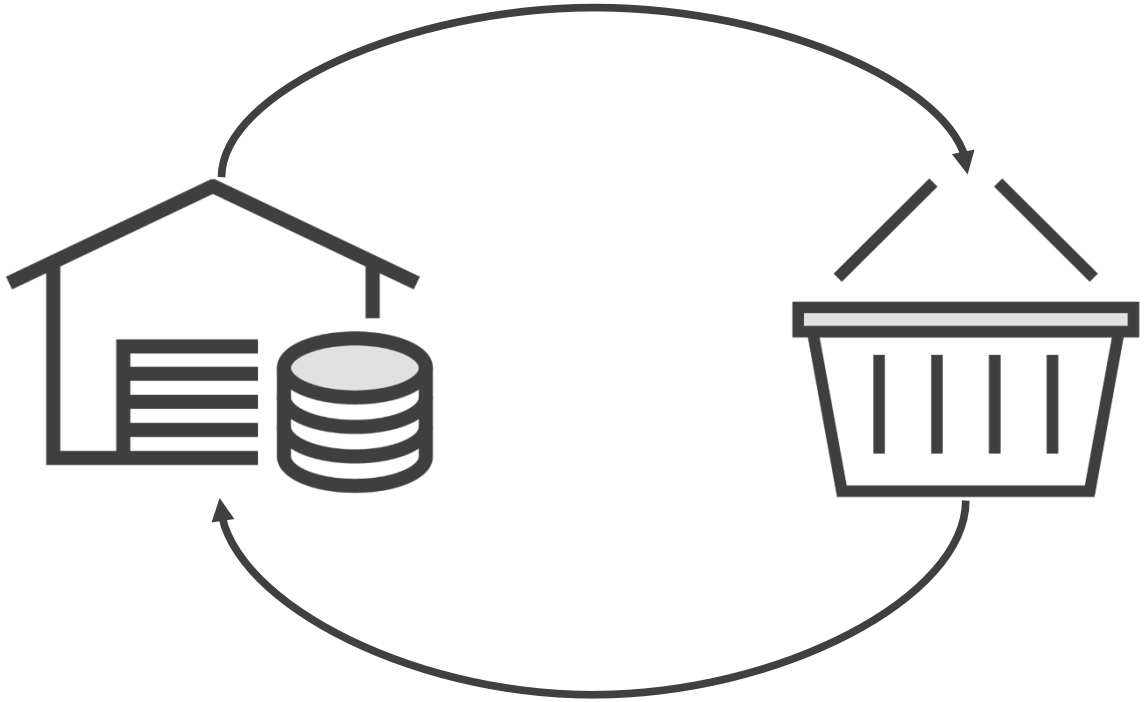
Ensuring that everything adds up and balances

- At the same layer
- Across layers

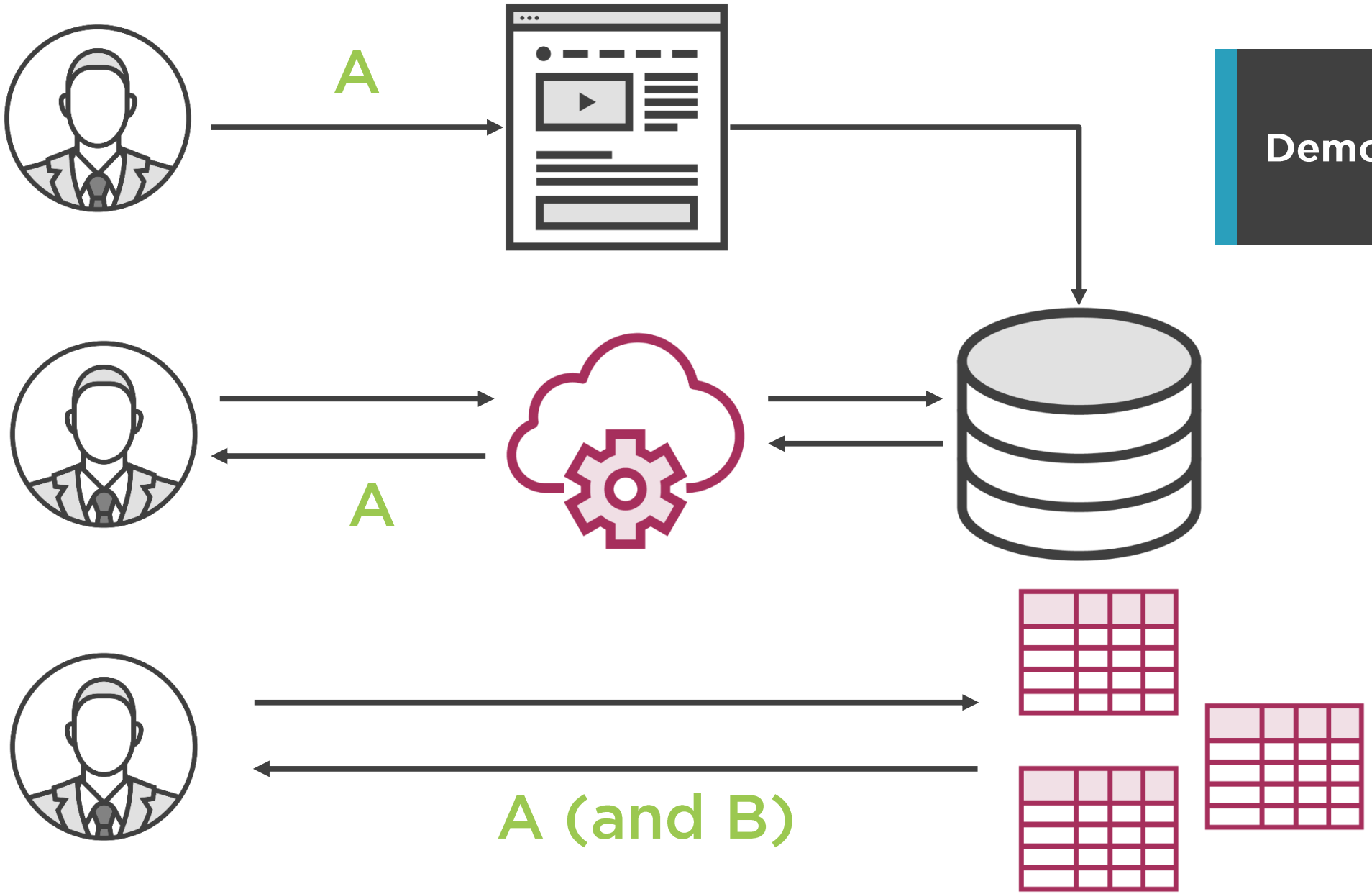
Just because you can cross-check, doesn't mean you should do it everywhere and all the time


Browser #1
+ 1
checkout
& pay
In stock: 10
Browser #2
assert(9);

It all must add up



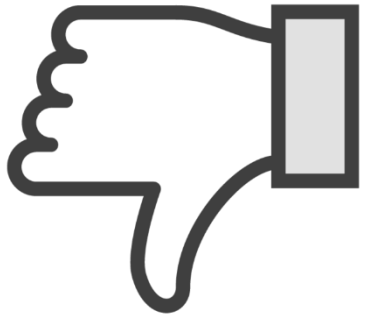
reserve
add
remove
add again
...



Demo coming up

Test Suite

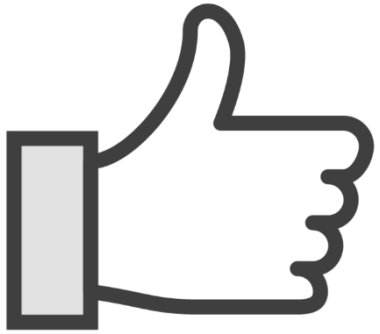
(all or most tests have cross-checks added)



Test Suite

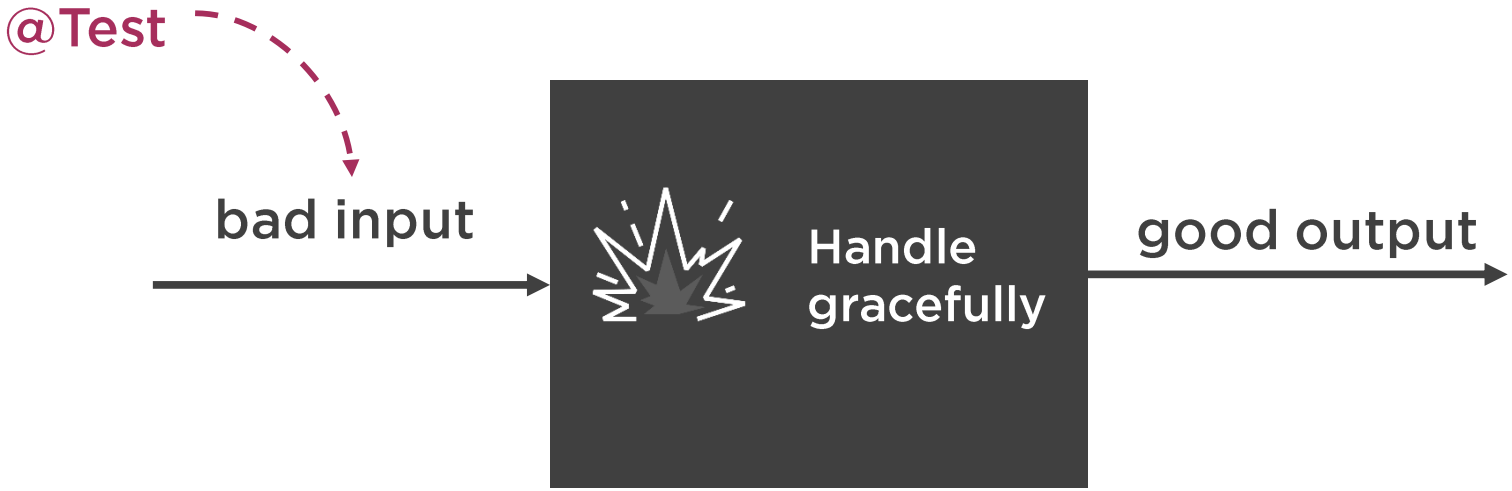
\+ Test with cross-check

\+ Test with cross-check



Demo

Cross-checking between UI and Web API


@Test
bad input
Handle
gracefully
good output

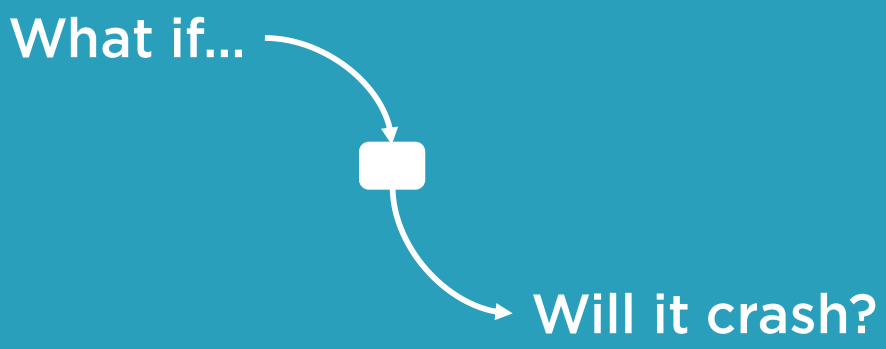
Error Conditions

Expected

Thrown by developers themselves, e.g. IllegalArgumentException

“Unexpected”

(i.e. typically less expected)



Error conditions are (sort of) a

sub-set of Boundary conditions

Typical “Unexpected” Error Conditions

Running out of memory

Running out of disk space

Network availability issues

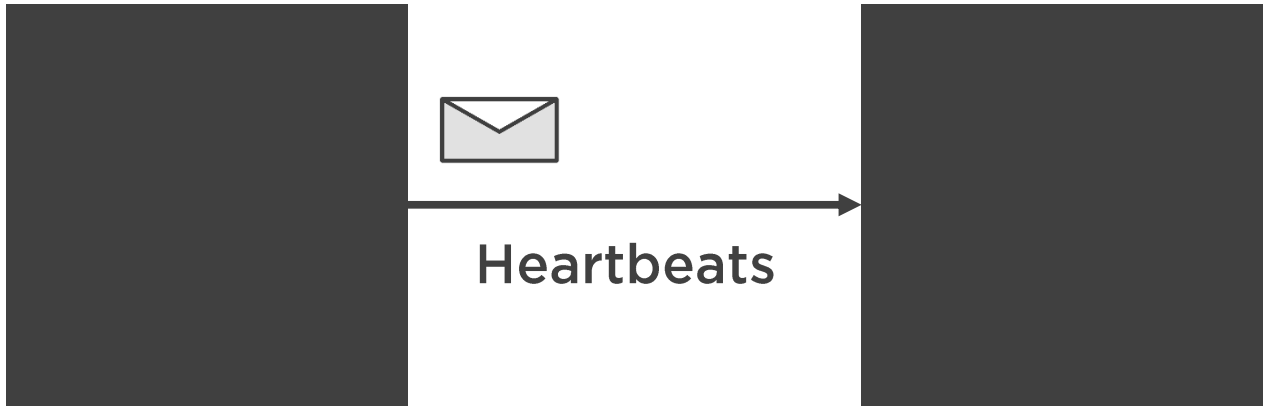
IO operations:

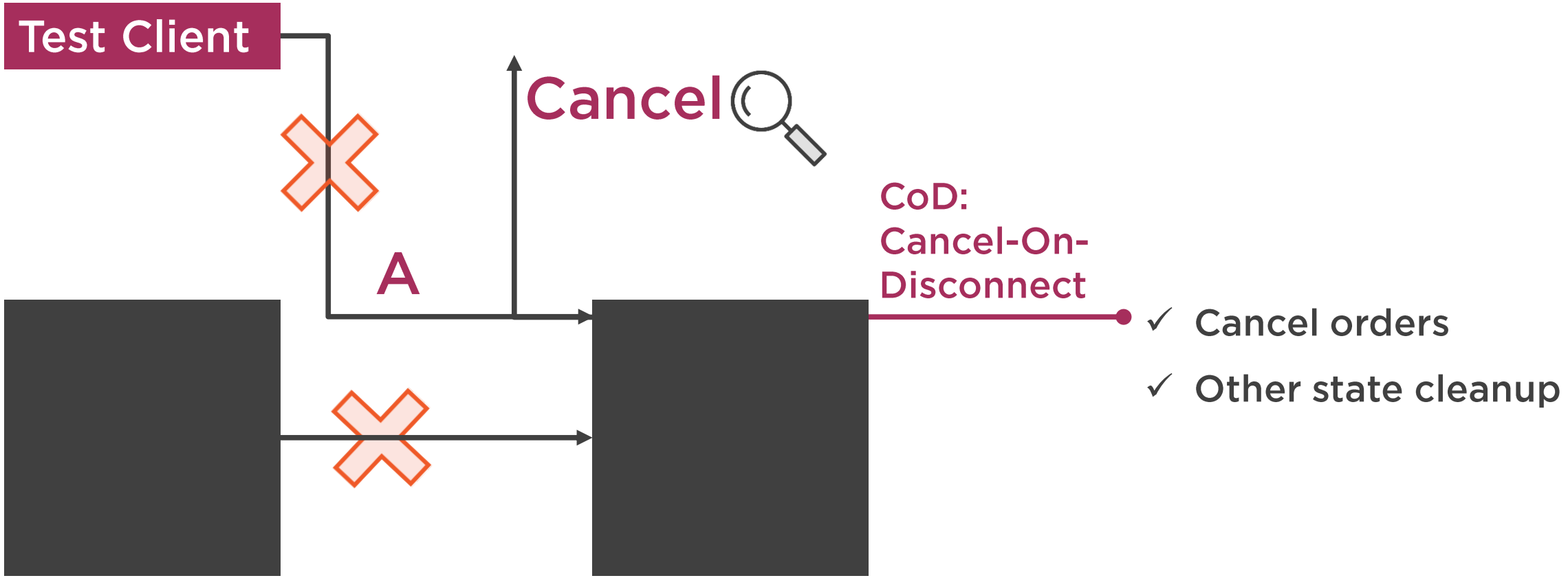
- Files doesn’t exist
- No read/write rights
- File is too big to load into memory
- etc.

Typical “Unexpected” Error Conditions

Running out of memory or space:

- Don’t catch Java Throwable or Error
- Do try to shutdown gracefully


Heartbeats


Test Client
Cancel
A
CoD:
Cancel-On-
Disconnect
✓ Cancel orders
✓ Other state cleanup

Heartbeat

A periodic signal to indicate normal operation.

```
boolean isAccessible =
                    isRegularFile(path) && isReadable(path) &&
                     isExecutable(path) &&  isWritable(path);


if (isAccessible) { /* ... */ }
```

Java IO

Handling files defensively

Demo

Forcing error conditions with bad input

Flight Search Service

From

INPUT

To

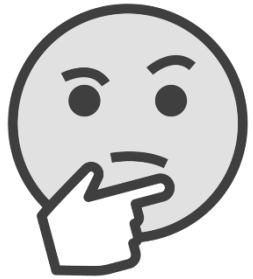
INPUT

Date

INPUT

Passengers #

INPUT



What if...

FIRST vs. BICEP

{F}IRST

Fast: refers to the speed of tests

BICE{P}

Performance: refers to the speed of the SUT

Resources on Performance Testing

Book:

- The Art of Application Performance Testing

Pluralsight Videos:

- Tracking Real World Web Performance
- JMeter: Getting Started

Performance Testing Tips

Don't try to optimize something that is working fine now

Don't try to guess where the problem is

- Optimizing one place or layer
- But the bottleneck might be somewhere else

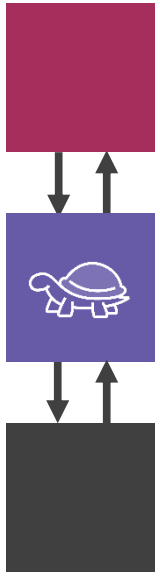
Investigating is a big part of performance testing

Typically carried out higher-than-unit level

“Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs [...] and that premature optimization is the root of all evil”

Donald Knuth

My performance unit tests pass!



Great, but the system as a whole is still slow!

Summary

Boundary – the “What if?” scenarios

Inverse relationship – consider when dealing with mathematical computations, counting or keeping track of stock

Cross-checking – single or cross-layer

Error conditions – “How can I make this system crash?”

Performance – typically done with higher level tests

Up Next: CORRECT