Debugging with the Scientific Method

@stuarthalloway

The One Where the Database Broke the Queue

system developed against H2

production against Cassandra

testing started late

on switch to Cassandra, app hangs with HornetQ error

hair on fire

Why Debugging?

because bugs!

look smart to your friends

debugging happens everywhen

write better bugs

Why Me?

Clojure screening

Datomic support

gray hair

lazy

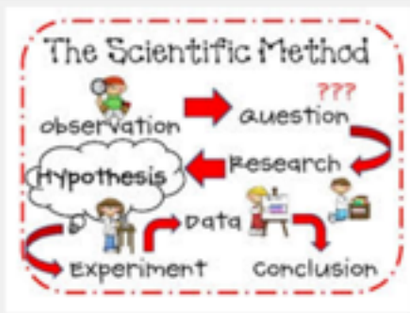
Tools are Great

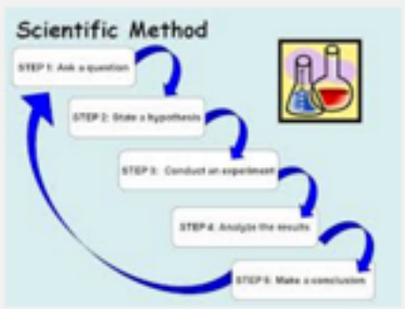
Cursive

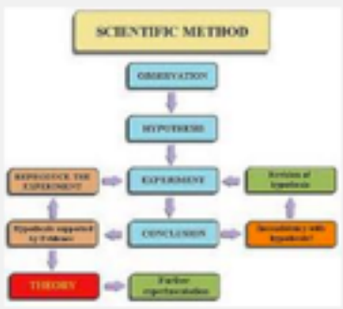
(cider[ ])

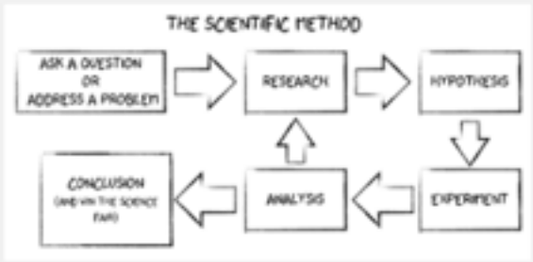
Why Scientific Method?

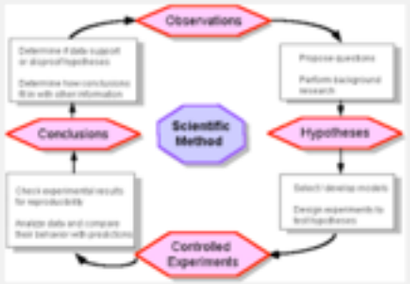


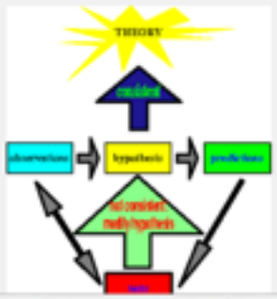


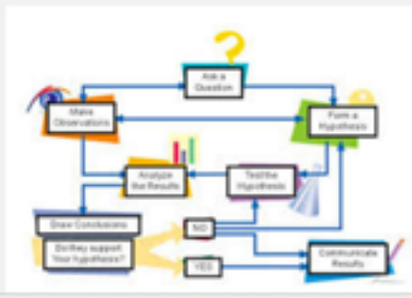


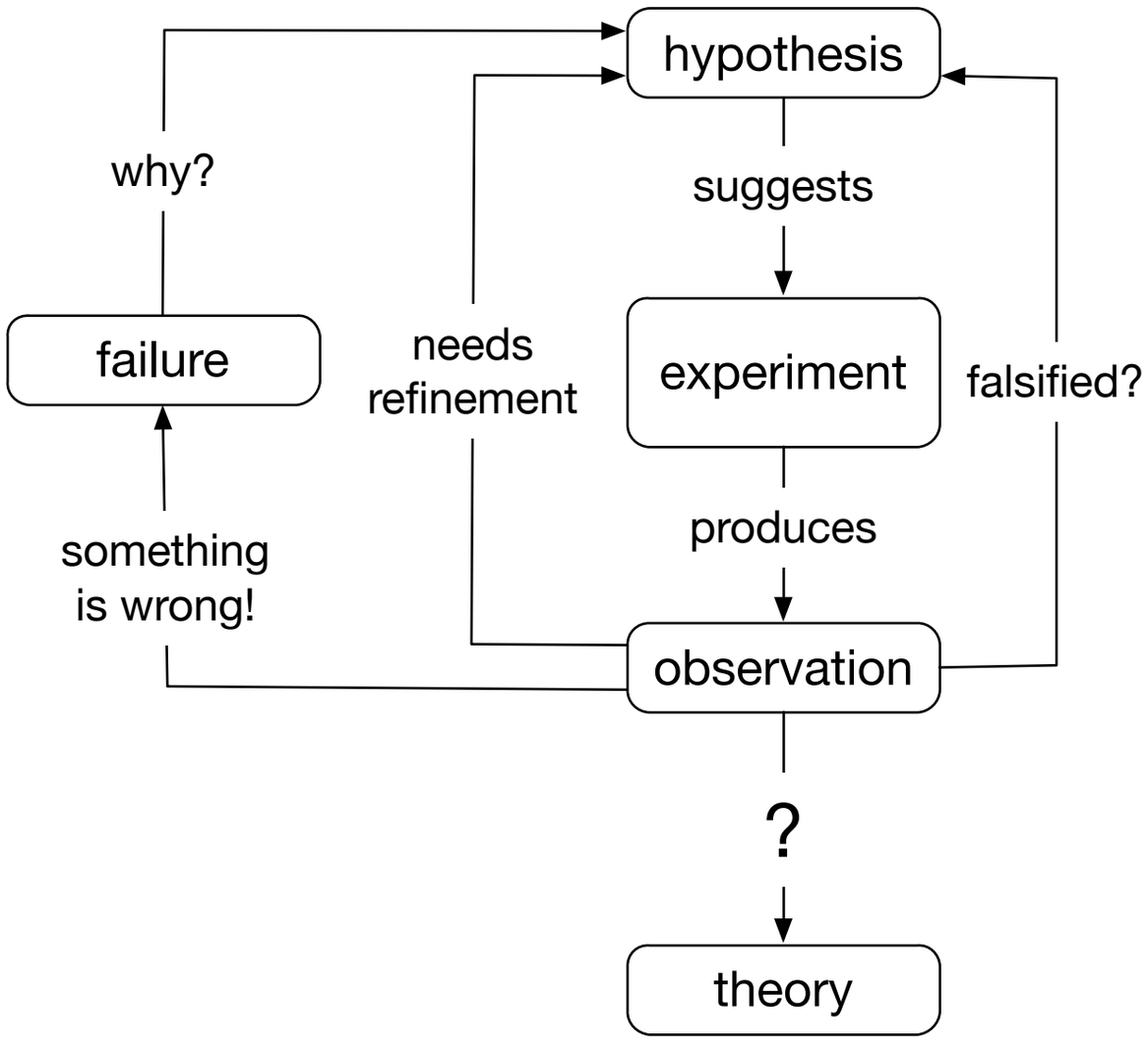




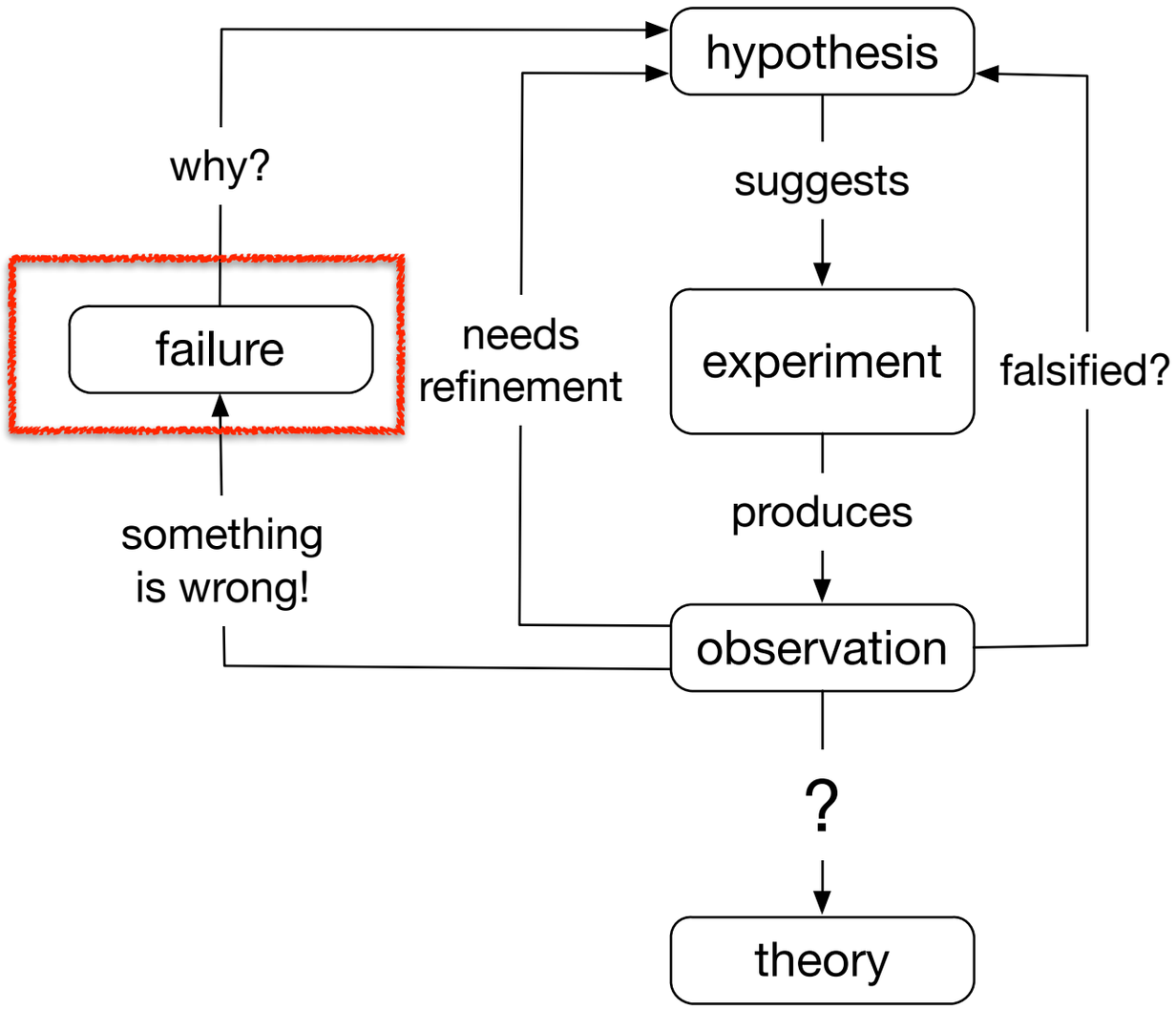




SM for Debugging



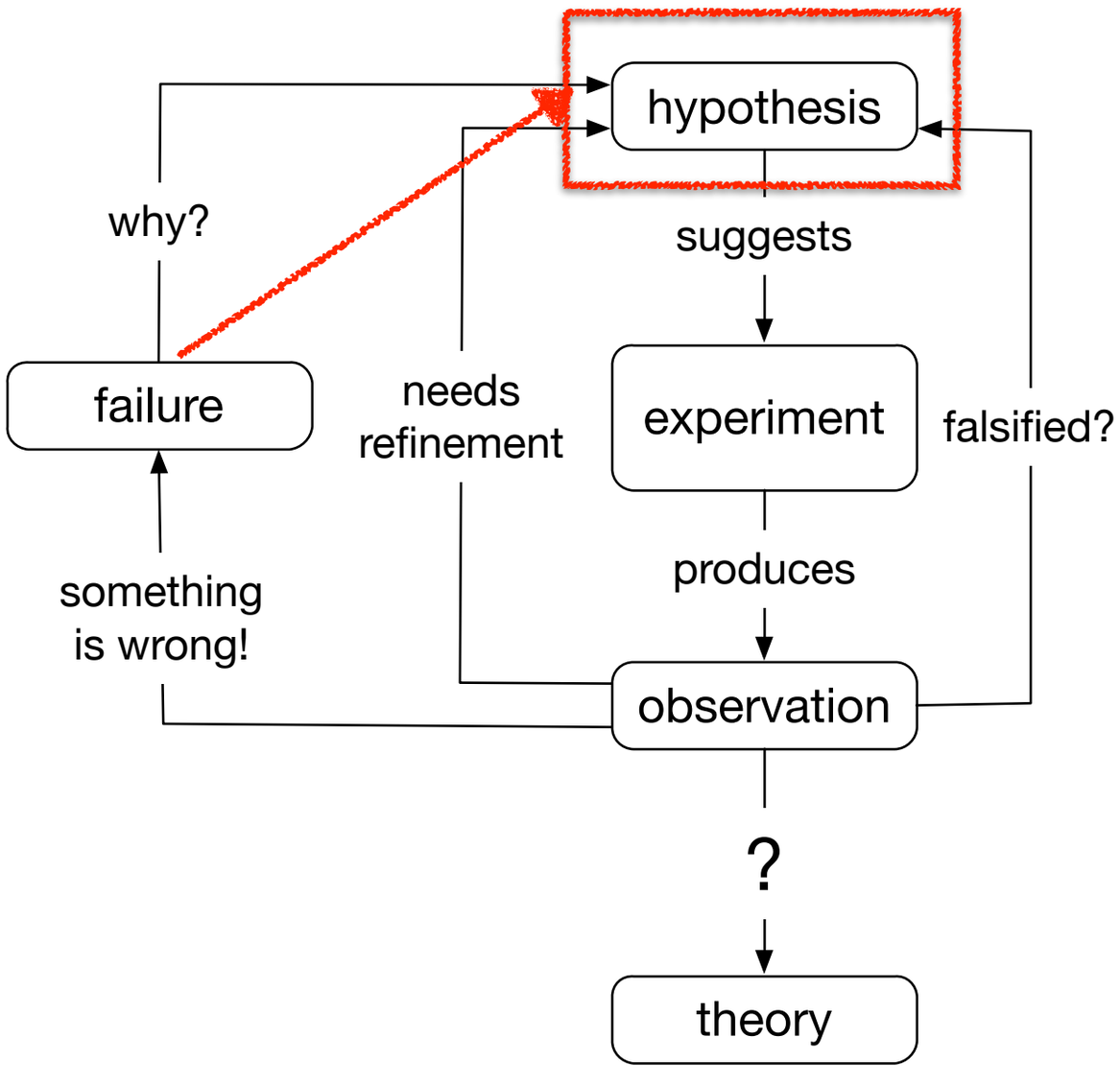
Failure



lack of success

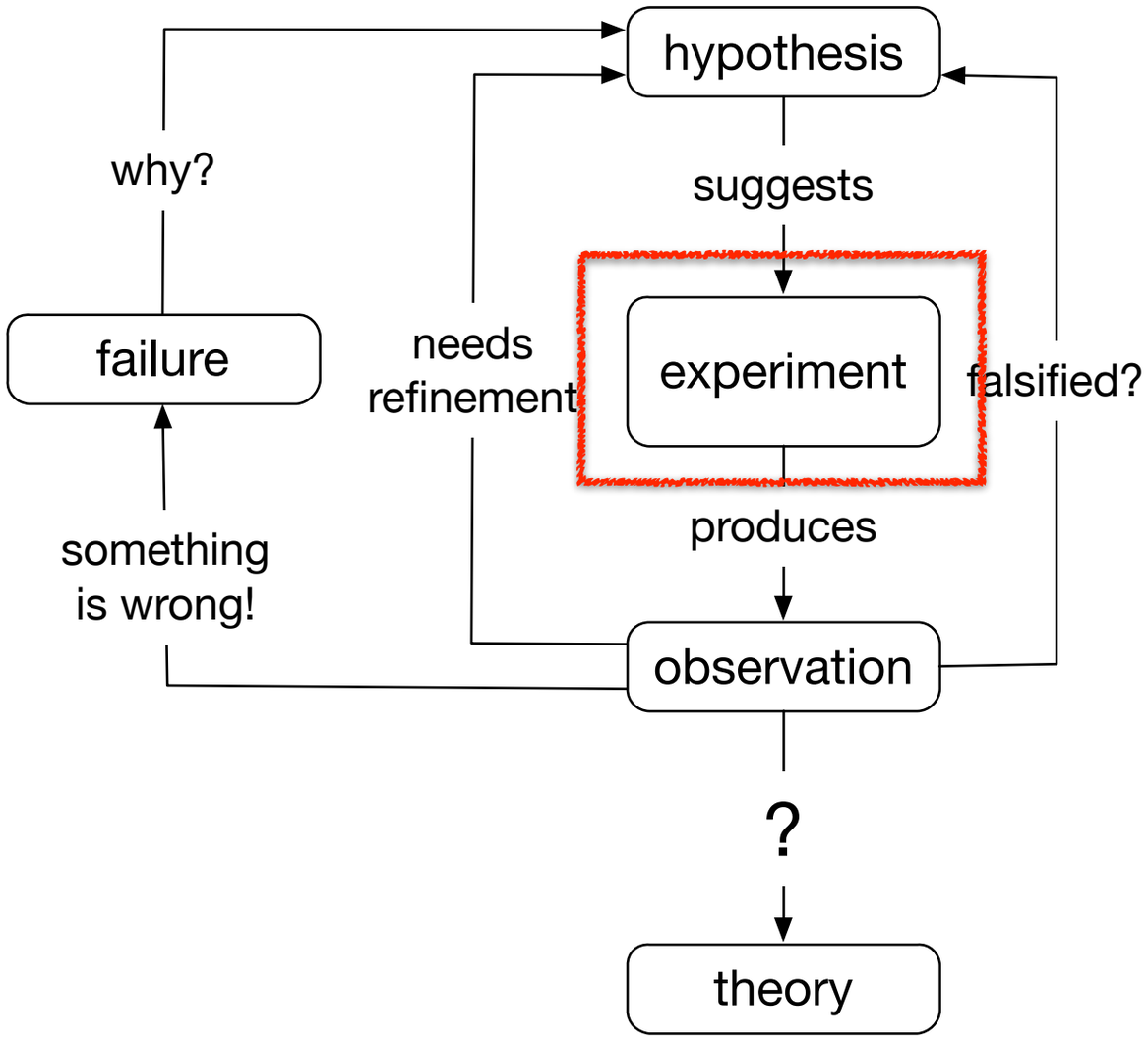
omission of
expected action

Hypothesis



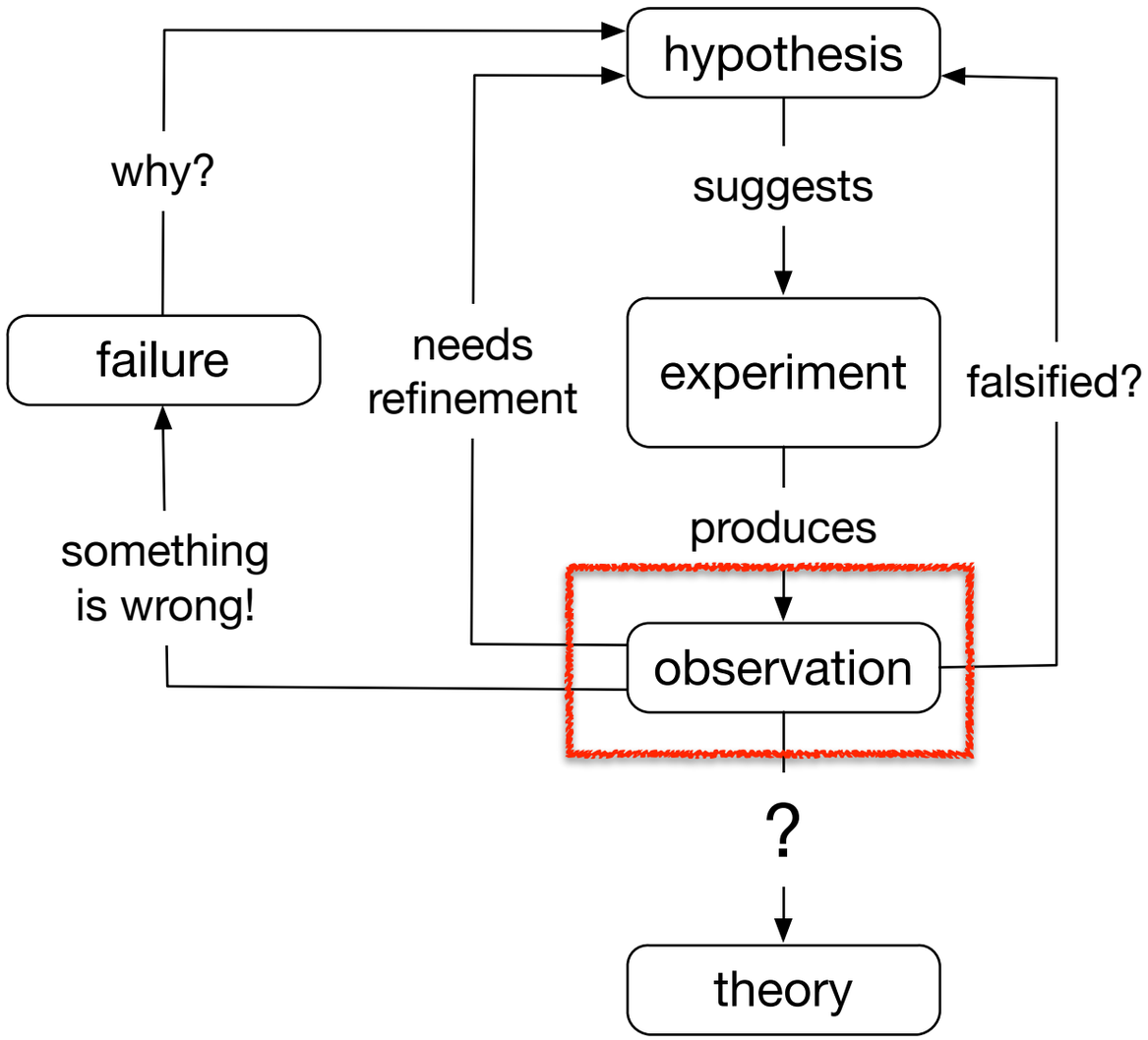
a proposed explanation made on the basis of limited evidence as a starting point for further investigation

Experiment



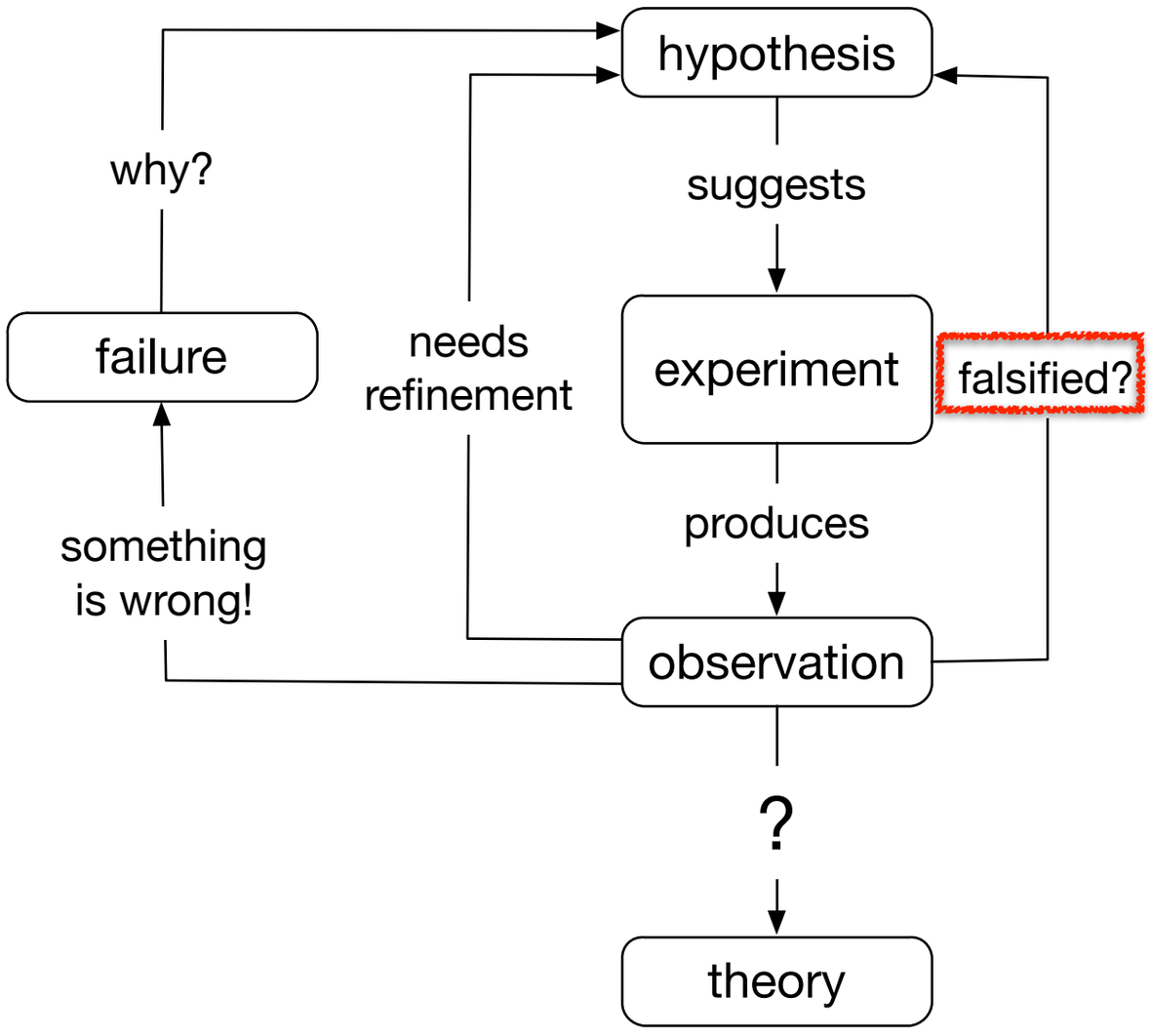
a test, trial, or tentative procedure

Observation



active acquisition of information from a primary source

Falsification



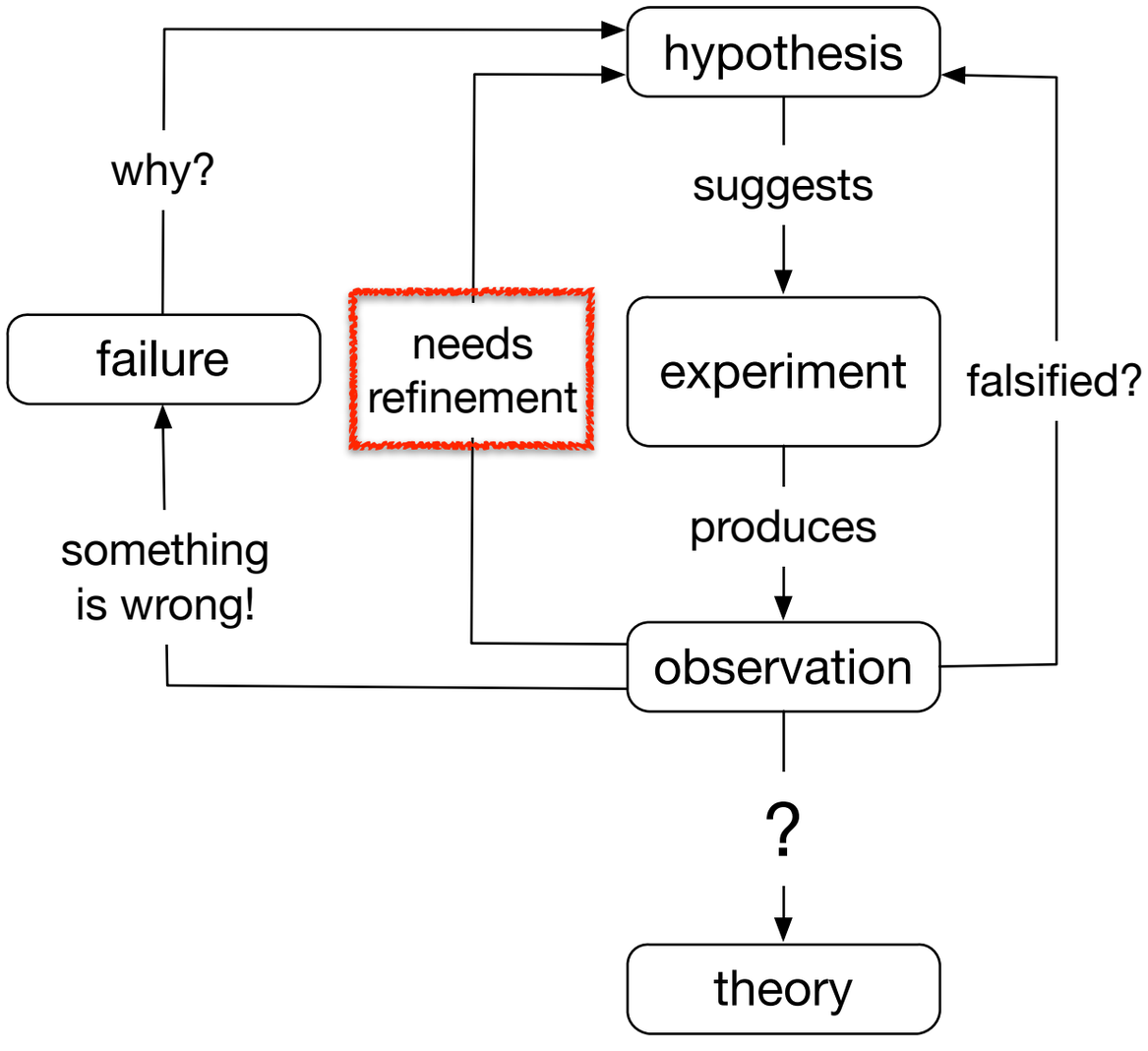
deductive process
using modus tollens:

$$H \rightarrow \neg O$$

$$O$$

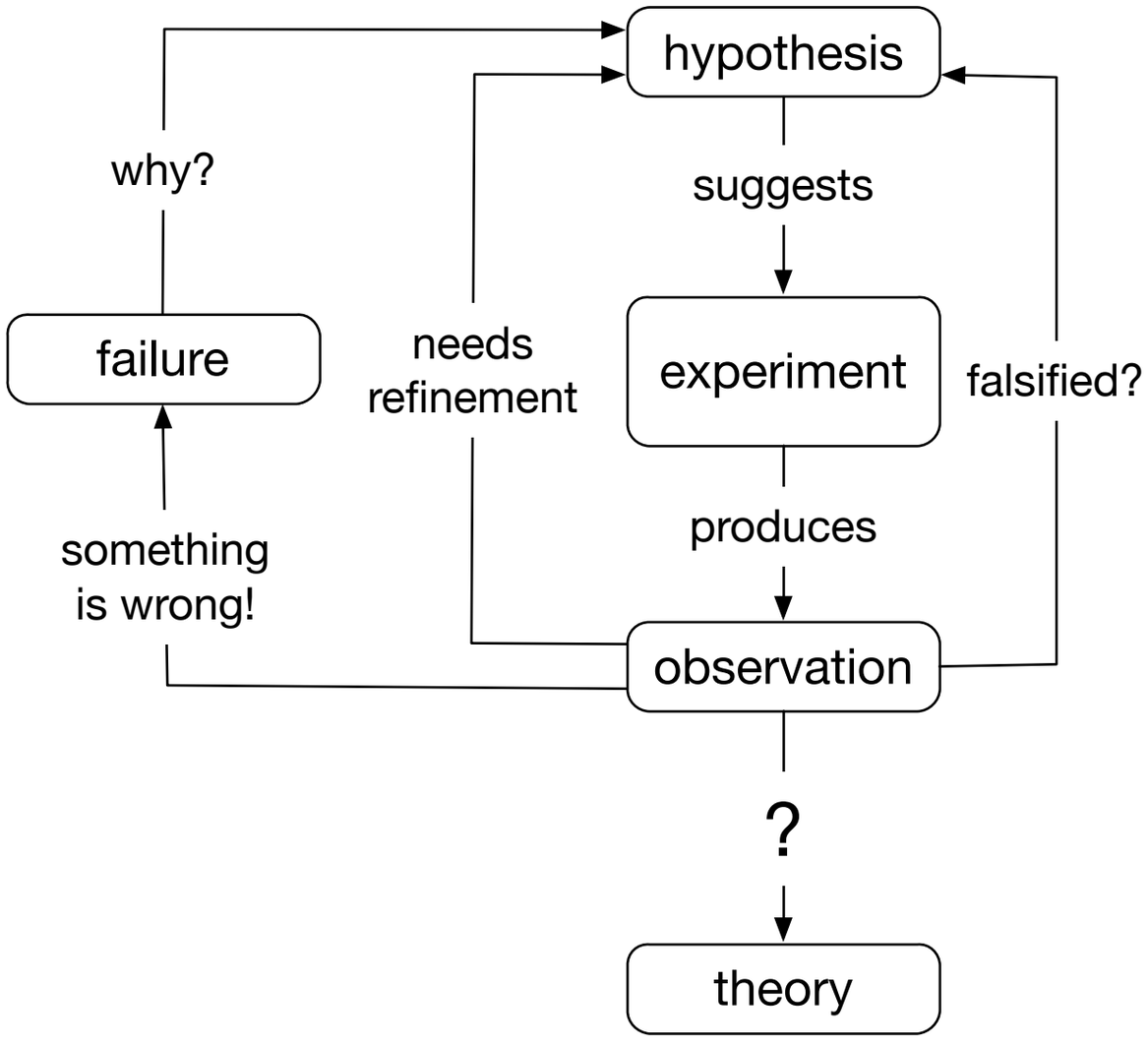
$$\neg H$$

Refinement

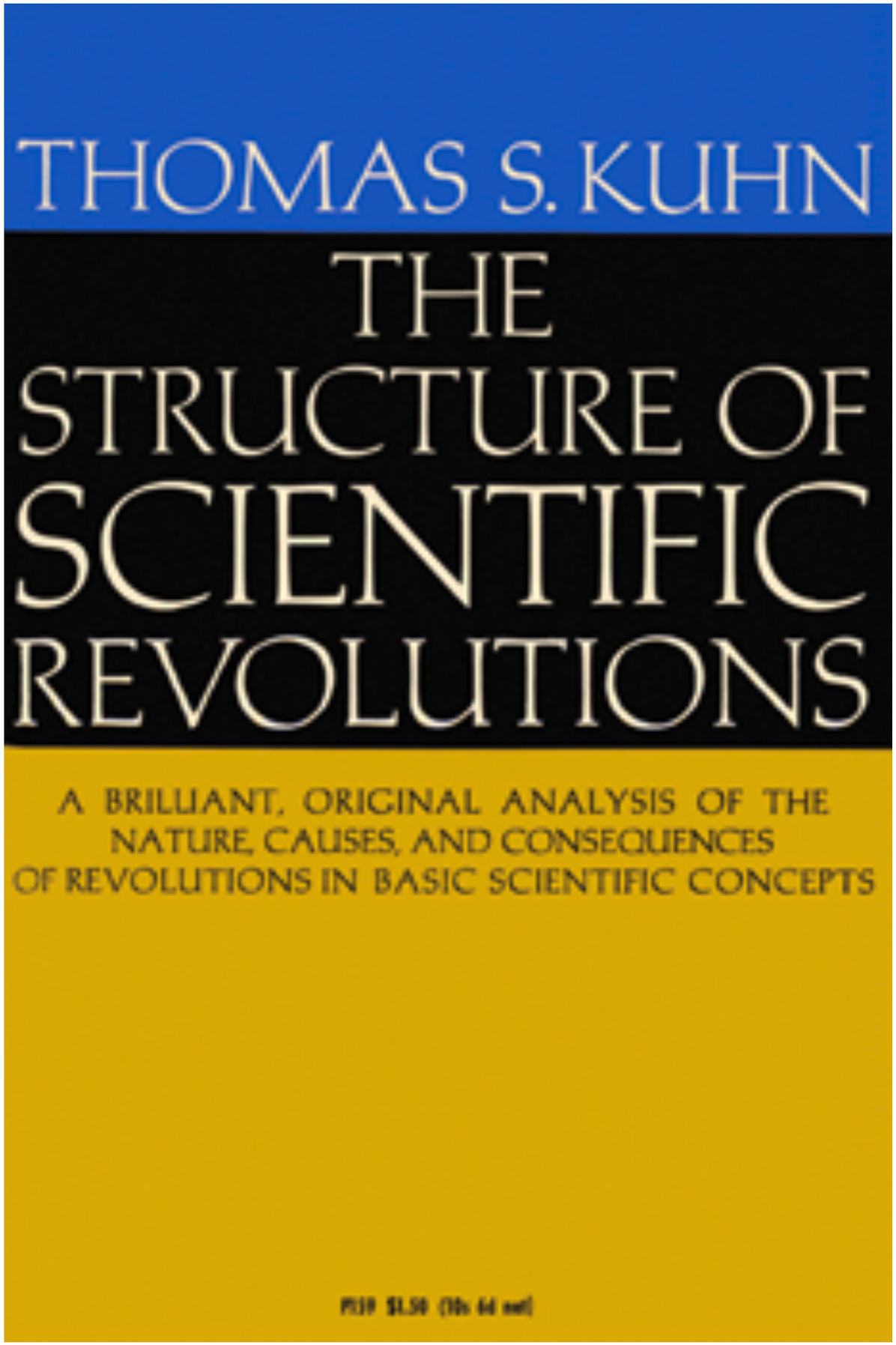


the process of
removing impurities
or unwanted elements
from a substance

Theory



a hypothesis offering valid predictions that can be observed



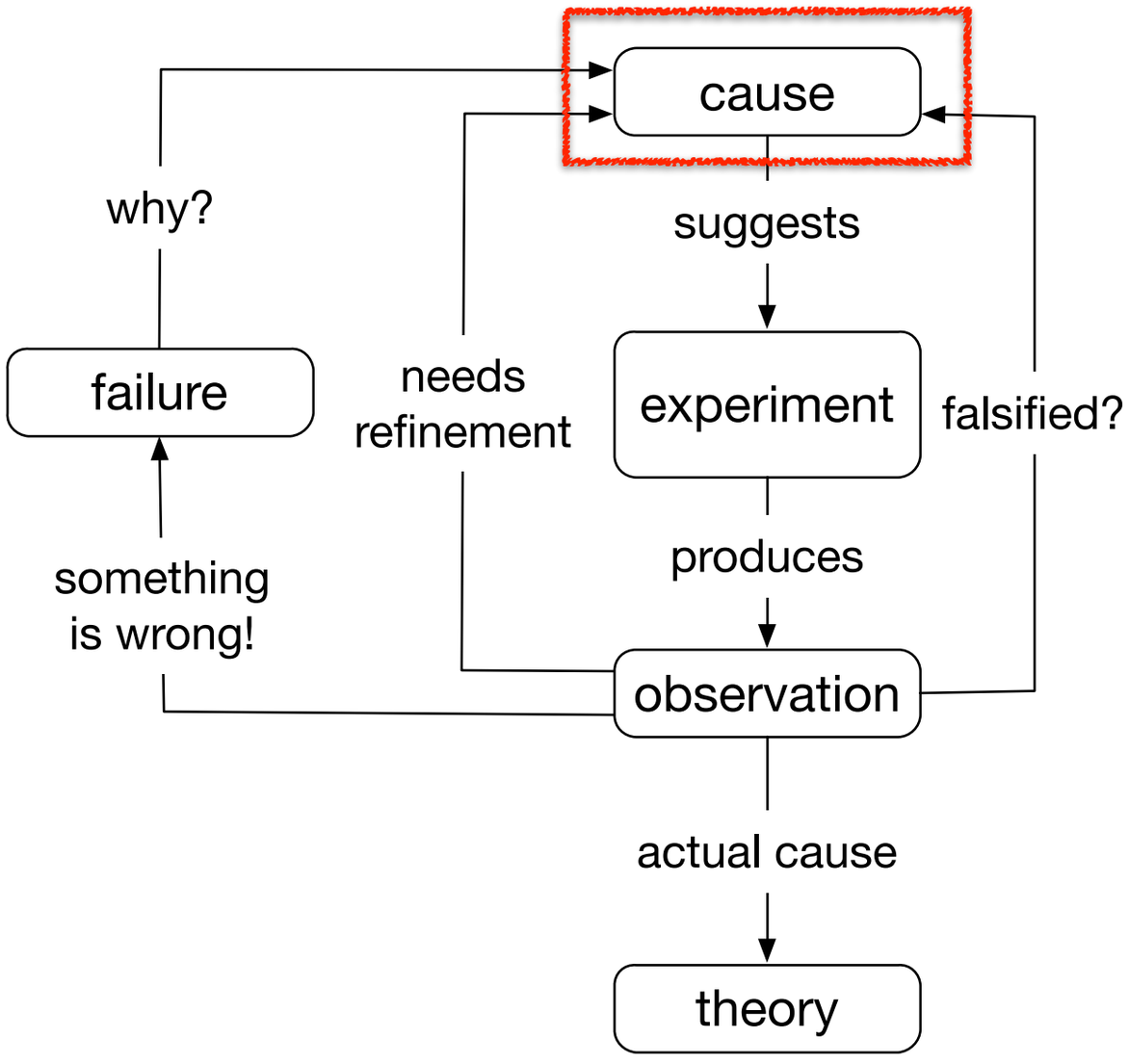
epistemological challenges

Debugging:
Most “Scientific” Thing Ever!

more constrained than science

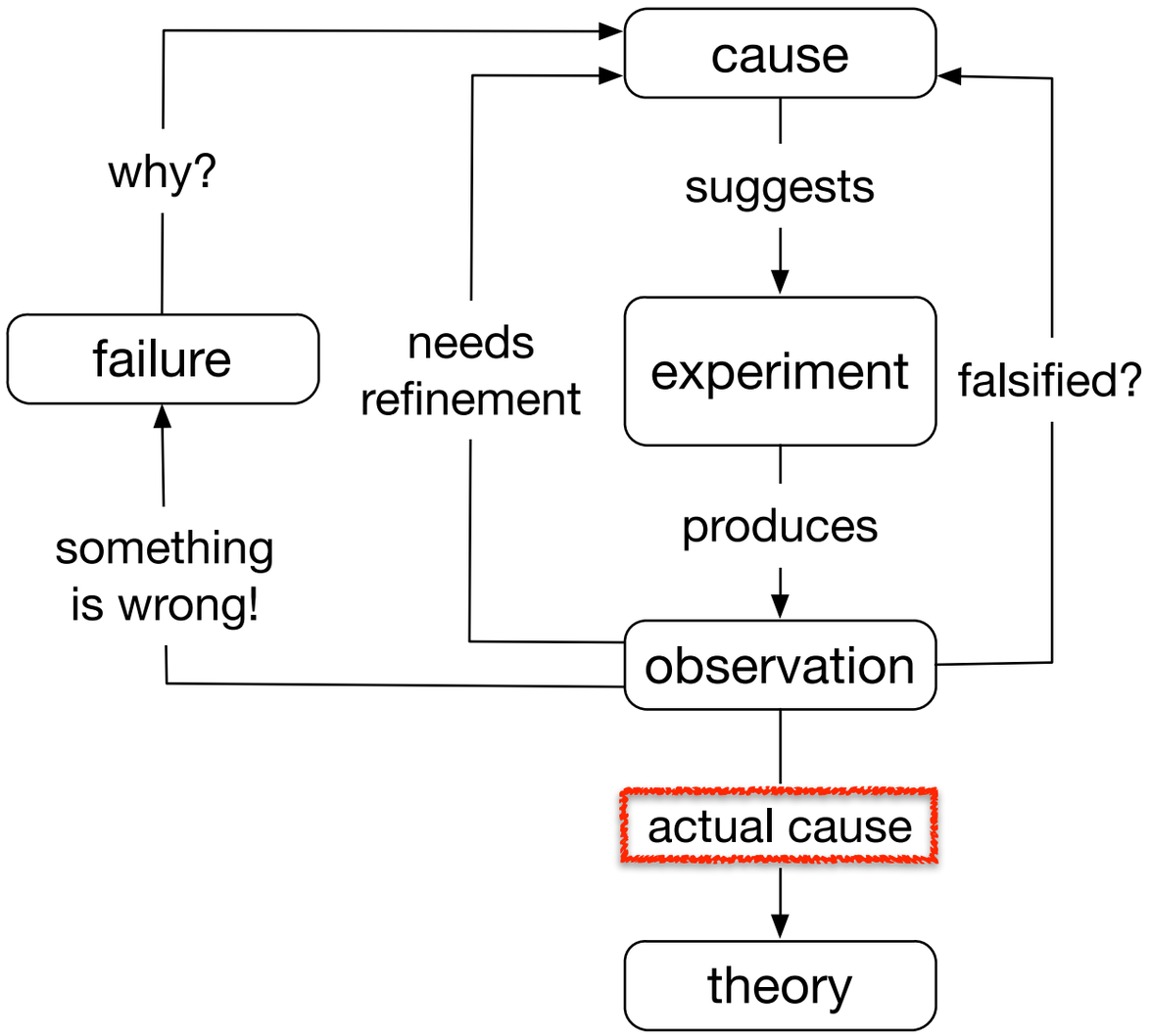
deductive, not inductive

Cause



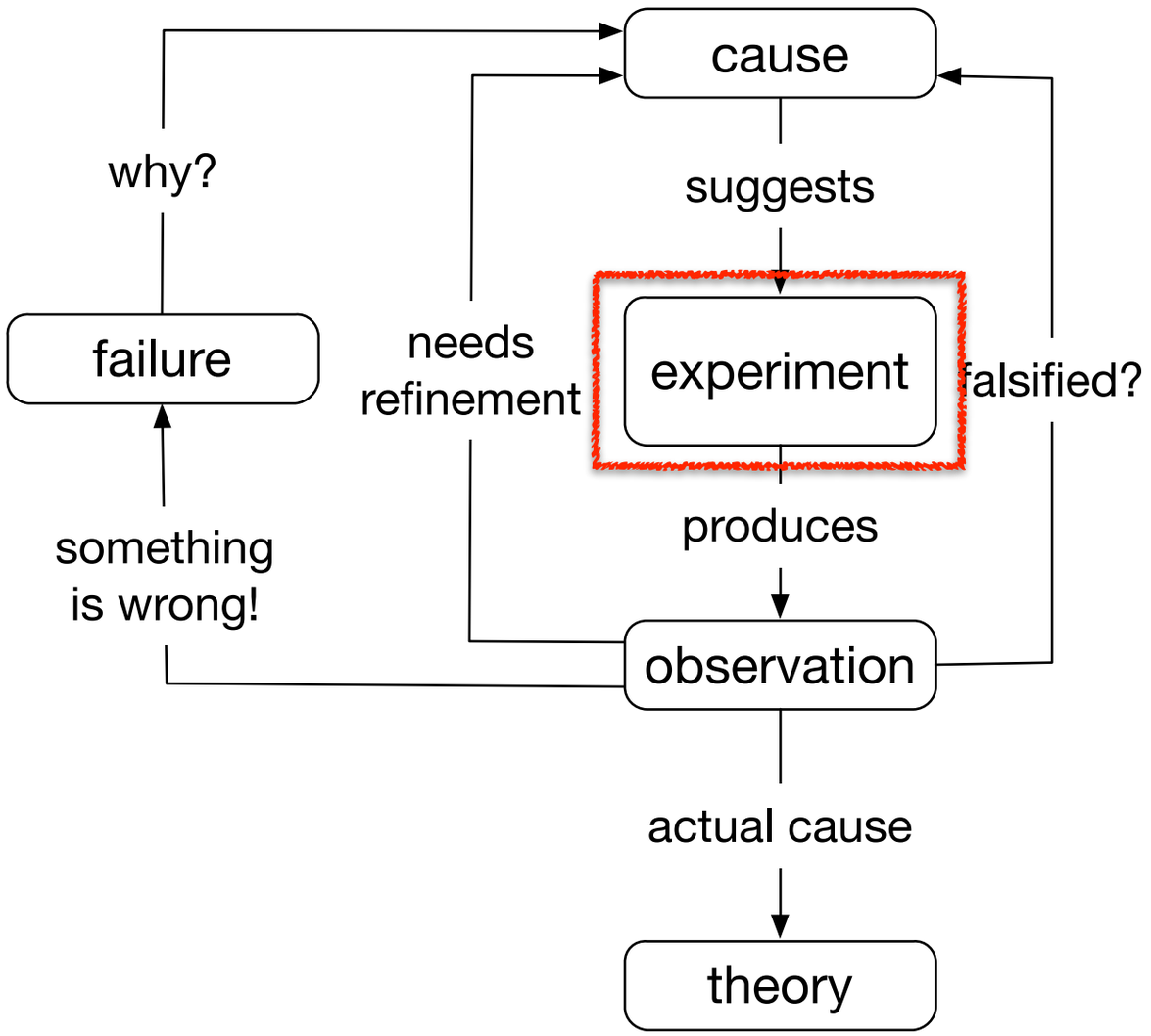
an event preceding an effect without which the effect would not have occurred

Actual Cause



difference between the actual world and the closest possible world in which the effect does not occur

Fix



an experiment that establishes an actual cause

Doing Science Well

clear problem statement

efficient hypotheses

good experiments

useful observations

read the entire f-ing manual

write everything down

What People Actually Do

encounter a failure

jump to conclusions

change multiple things

while juggling chainsaws

give up and call for help

Clear Problem Statements

steps you took

what you expected

what actually happened

Efficient Hypotheses

divide and conquer

decrease and conquer

bisection

interval halving

proportional reduction

http://oceans11.wikia.com/wiki/File:Vlcsnap-2013-08-28-00h56m04s3.png

Search Automation

http://boundlessgeo.com/2013/07/finding-a-performance-regression-with-git-bisect/

Quick! Where's the Bug?

Your App	
Clojure Lib	
Clojure Lang	Popular Java Lib
JVM	
OS	
hardware	
physics	

Good Experiments

reproducible

driven by hypothesis

small

change only one thing

Which of these should be in your repro case?

test framework

build system

Docker

Kubernetes

database

application

web browser

debugger

IDE

web framework

The One With Java 11

Datomic uses Janino to compile transaction functions

cascading forced upgrades for Java 11 compatibility

“it didn’t work”

https://github.com/Datomic/janino-repro-3

Lots Of Noise

database

peer and transactor processes

multiple languages (Java and Clojure)

stateful (code installed in the database)

After Many Reductions

Starting with 3.0.7* (commit 63fe4d054?*),
Janino cannot find a class in a JAR*
if another class with a common package
prefix* occurs earlier in the classpath*.

*separate experiments

The Last Experiment

```java
package datomic;

import org.codehaus.commons.compiler.CompileException;
import org.codehaus.commons.compiler.jdk.ScriptEvaluator;

public class JaninoRepro3 {
    public static void main(String[] args) throws CompileException {
        System.out.println("Class location:");
        System.out.println(ClassLoader.getSystemClassLoader().getResource("junit/framework/TestCase.class"));

        System.out.println("Classpath:");
        System.out.println(System.getProperty("java.class.path"));

        System.out.println("Run trivial script:");
        ScriptEvaluator ev = new ScriptEvaluator();
        ev.setDefaultImports(new String[]{"junit.framework.TestCase"});
        Runnable r = (Runnable) ev.createFastEvaluator("System.out.println(123);", Runnable.class, new String[]{});
        r.run();
    }
}
```

Making Observations

understand all the outputs

suspect correlations

use good tools

The One With Logging

cryptic subsystem failure…

…and also logging misbehaving

```
10:40:20,444 |-WARN

Resource [logback.xml] occurs multiple
times on the classpath.
```

The Bug

library appeared on classpath twice

dragged in two copies of SLF4J, causing the warning

semantic versioning is broken

exports can have the same name but incompatible semantics

idiomatic JAR loading will commingle two versions of lib

RTEFM

a bug starts as an “unknown unknown”

so that means read the ***entire*** manual

good docs (for debugging) are

short

specifications

The One Where the Load Balancer Didn’t (?)

three services: two HTTP (Jetty) and one custom socket

all three gracefully retry/failover in isolated tests

one talks to wrong machine during rolling deploy

"if one or more target groups does not have a healthy target in an Availability Zone, we *remove the IP address for the corresponding subnet from DNS*, but the load balancer nodes in the other Availability Zones are still available to route traffic"

https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

Write Things Down

problem statement

hypotheses

what experiment should show

why experiment even makes sense

observations

YOUR MIND PALACE SUCKS

https://medium.com/on-breaking-the-mold/the-mind-palace-a-place-for-everything-753688facc6d

https://en.wikipedia.org/wiki/Mastermind_(board_game)

https://en.wikipedia.org/wiki/Mastermind_(board_game)

The One Where the Database Broke the Queue

system developed against H2

production against Cassandra

testing started late

on switch to Cassandra, app hangs with HornetQ error

hair on fire

What Else Do We Know?

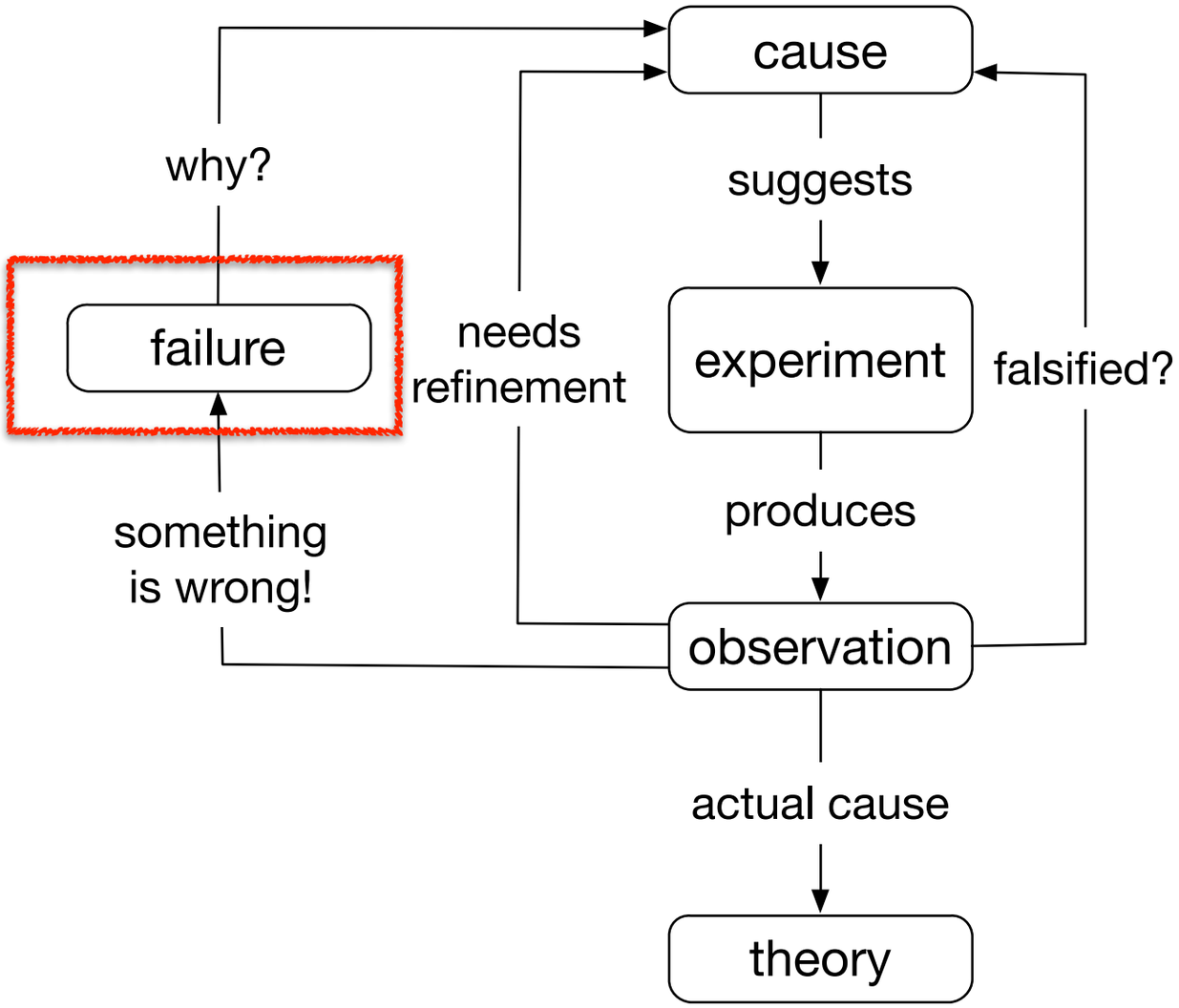
works with H2 and not with Cassandra

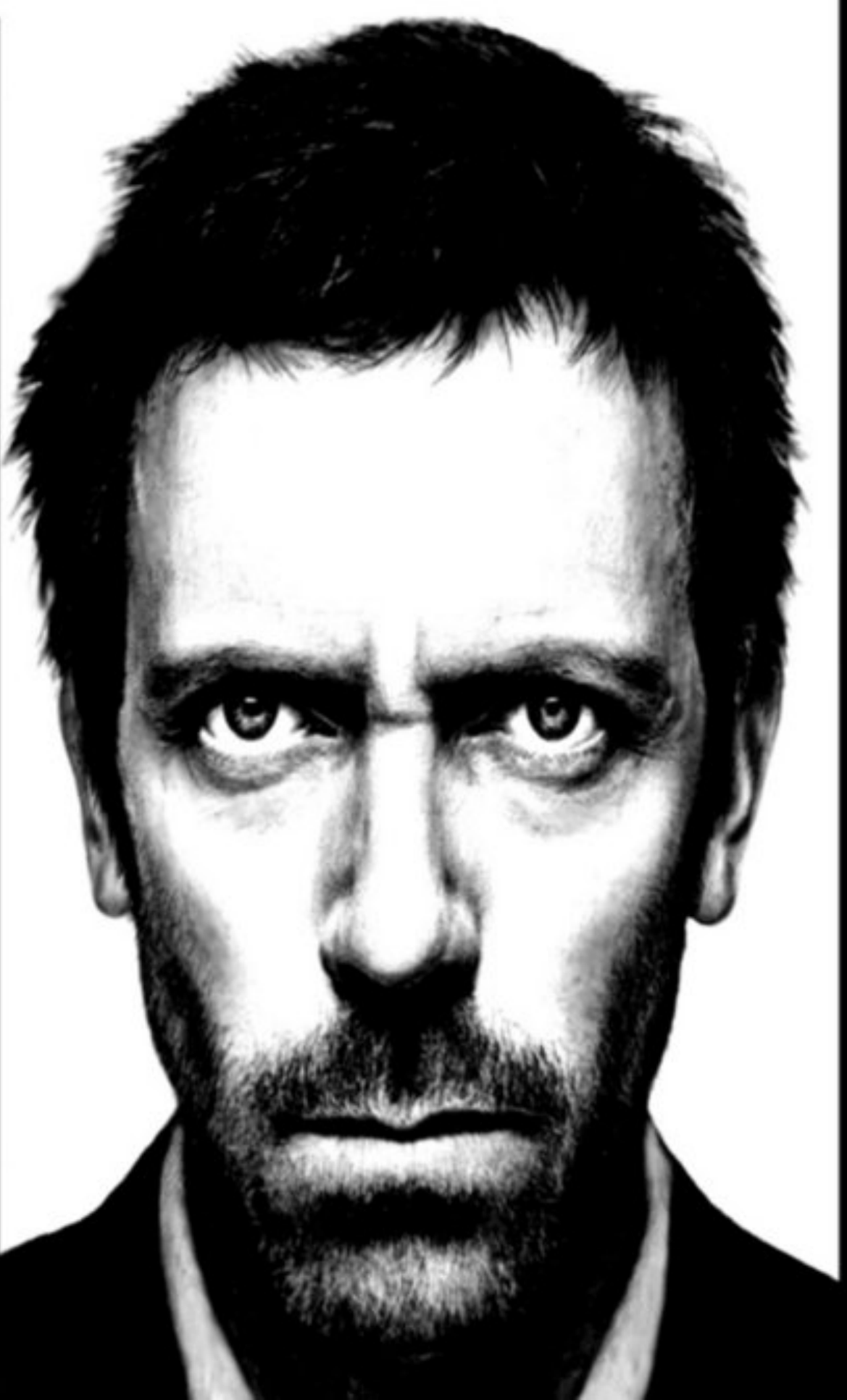
failure shows up in HornetQ

program is performing a very large query

CPU is pegged just before failure

The Failure is not the Defect





IT'S NEVER LUPUS

http://ambrohexal.deviantart.com/art/Its-never-Lupus-186073495

It's *Always* GC

OOM is typically unexpected

OOM can happen anywhere

OOM can appear as almost any other exception

near-OOM dramatically impacts scheduling

OOM related problems cascade

Hypothesis: Large Query

works with H2 and not with Cassandra

failure shows up in HornetQ

program is performing a very large query

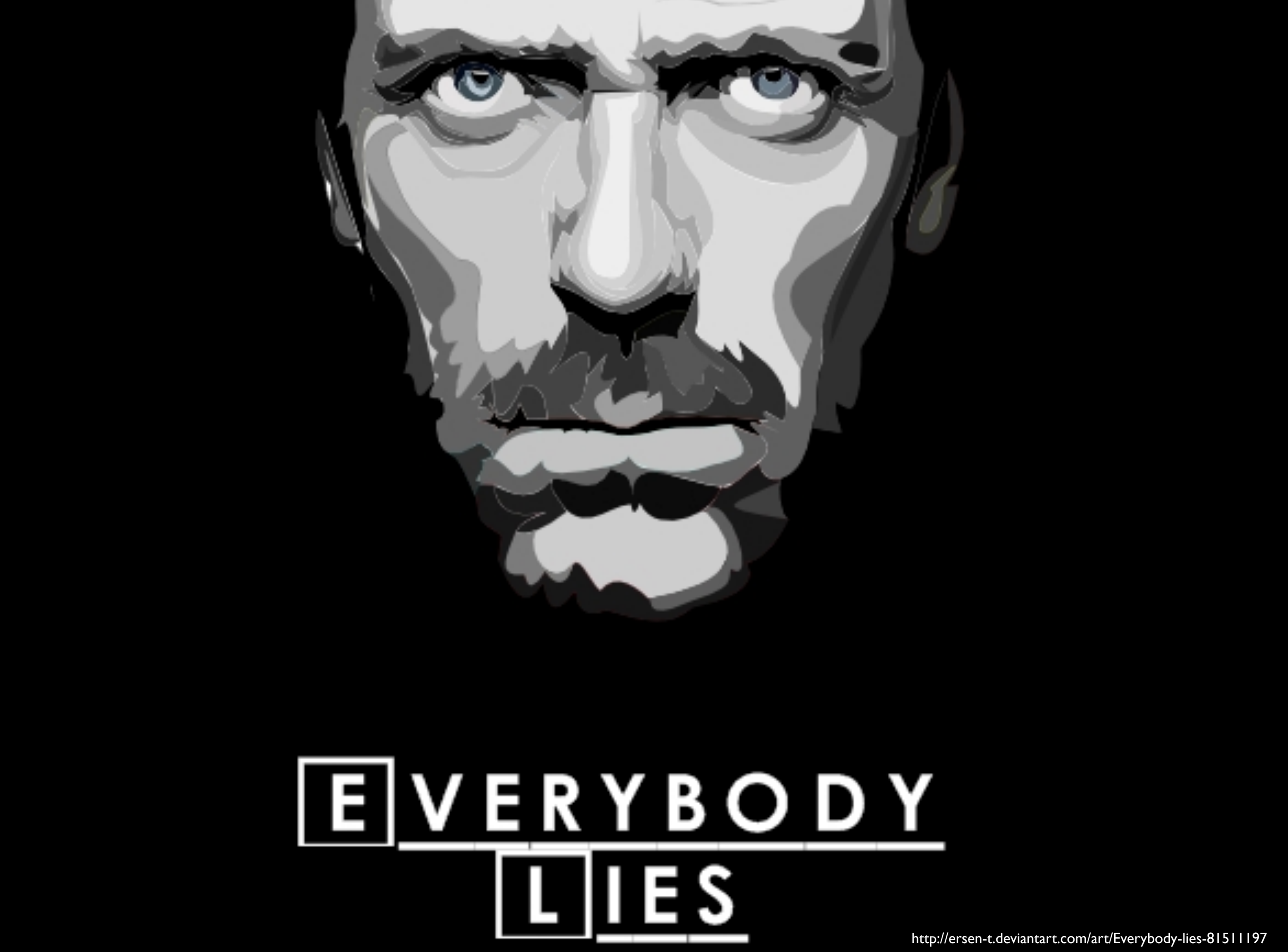
CPU is pegged just before failure

Experiment 1

massively pare down the environment

- Test.java starts & performs problem query in a loop
- no app, no Clojure, no build tooling

compare runs with H2 and Cassandra



http://ersen-t.deviantart.com/art/Everybody-lies-81511197

Weak Science Beats Strong Tools

poor problem statement

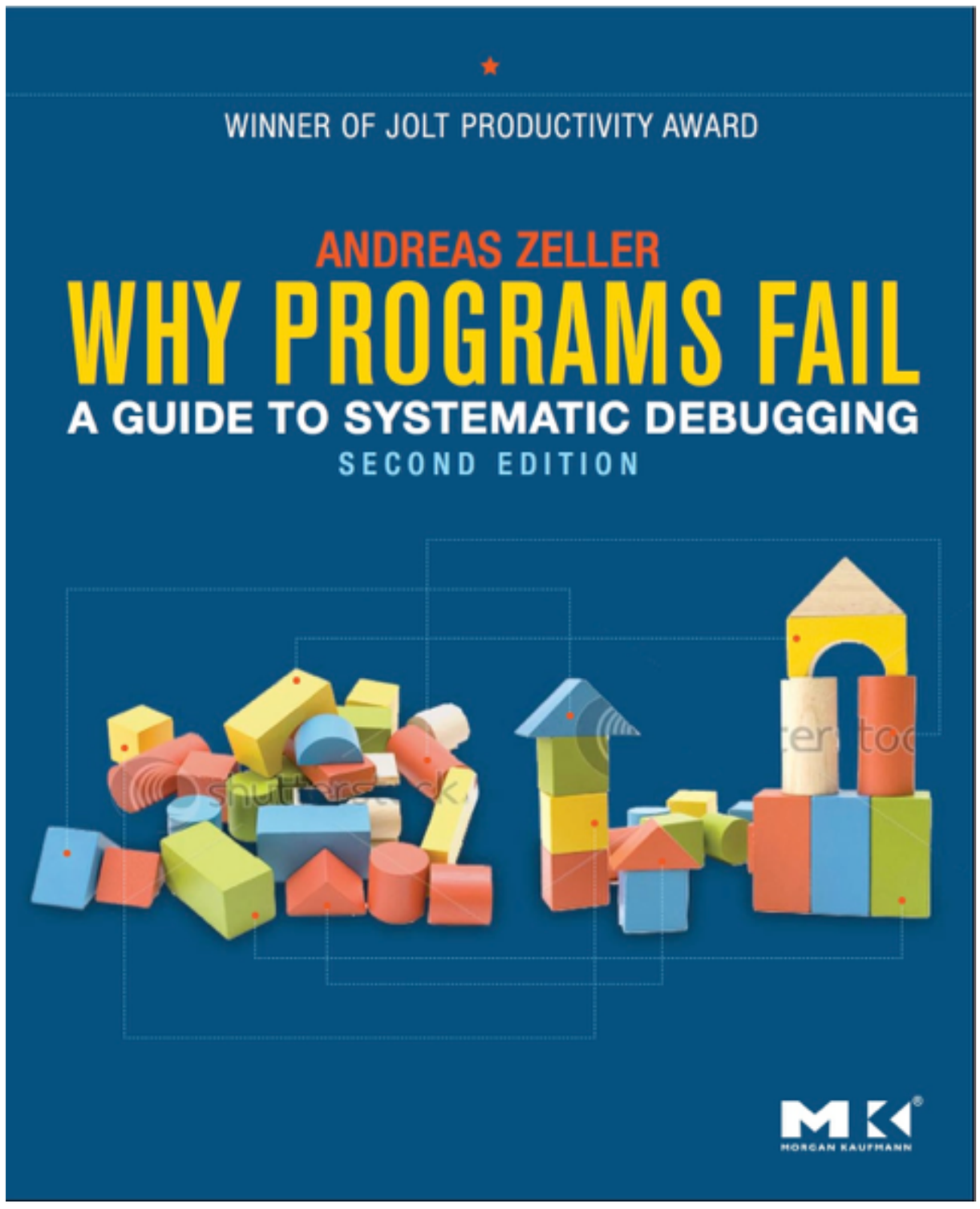
incomplete hypotheses

exploratory experiments

minimal domain knowledge

Call to Action

read chapters 5-7, 11-14



http://www.whyprogramsfail.com/

~~Debugging~~ ~~Science~~ Life Made Easy

make a plan

write it down

do one thing at a time

Debugging with the Scientific Method

@stuarthalloway