The Epistemology of Software Engineering

Nathan Marz
@nathanmarz
My personal philosophies on software development
Agenda

1. Limits of human knowledge
2. Effect of the limits of knowledge on software development
3. Embracing those limits enables you to build better software
How do I know my software is correct?
How do I know a proposition is true?
Epistemology
How do I know my software is correct?
You don’t
Your code is wrong
How do I know a proposition is true?
You don’t
True knowledge is unattainable
But wait... philosophy?
Fallacies

- Strawman
- Appeal to authority
- Appeal to emotion
- Circular reasoning
- False dilemma
- Argument to moderation
- Moral highground
- Ad hominem attack
- Shotgun argumentation
- Correlation vs causation
- Equivocation
- Burden of proof

Fallacies
Your code is **wrong**
Your code is *literally* wrong
Your code is wrong
Why do you believe your code is correct?
Your code

- Dependency 1
- Dependency 2
- Dependency 3
Hardware → Dependency 3,000,000
Electronics
Chemistry
Atomic physics
Quantum mechanics
I think I can safely say that nobody understands quantum mechanics.

Richard Feynman
Your code is wrong
Infinite regress
Epistemological “solutions”

1. Infinitism

2. Foundationalism

3. Coherentism
Coherentism
Foundationalism

Axioms
René Descartes
Cogito ergo sum
I think, therefore I am
Codito ergo sum
I code, therefore I am.
Cartesian foundationalism

1. Limited axioms

2. Knowledge through deduction
Cartesian programming

1. Axioms = rules of programming language

2. Programs = deductions from those axioms
public int fib(int n) {
    if(n==0 || n==1) return 1;
    else return fib(n-1) + fib(n-2);
}
public BigInteger fib(BigInteger n) {
    if(n.equals(0) || n.equals(1))
        return BigInteger.ONE;
    else return fib(n.minus(1)) +
            fib(n.minus(2));
}
print "Hello world!"
print "Hello world!"
-> OutOfMemoryException
print "Hello world!"
-> Hallo welt!
All the software you’ve used has had bugs in it
Including the software you’ve written
Induction
f(0) and \((f(n) \rightarrow f(n+1))\) 
\[\Rightarrow\] 
\(\forall n \geq 0, \ f(n)\)
Induction
public boolean isSunRisingTomorrow(
    boolean sunAlwaysRisen) {
    if(sunAlwaysRisen) return true;
    else throw new RuntimeException("WTF??");
}
<sidenote>
David Hume
“Why is inductive reasoning valid?”
Skepticism
perfect code
value to users
“My software is correct”
“My software is sometimes correct”
How do you minimize imperfection?
Storm’s “reportError” method
Storm is a realtime computation system, like Hadoop but for realtime.
Storm architecture
Storm architecture

Master node (similar to Hadoop JobTracker)
Storm architecture

Used for cluster coordination
Storm architecture

Run worker processes
Storm’s “reportError” method
Used to show errors in the Storm UI
Error info is stored in Zookeeper
What happens when a user deploys code like this?

```java
try {
    methodThatReturnsNull().foo();
} catch(Exception e) {
    collector.reportError(e);
}
```
Denial-of-service on Zookeeper and cluster goes down
Robust!

Designed input space  
Irrelevant!

Robust!

Failures!  
Bad performance!  
Security holes!

Actual input space
Implement self-throttling to avoid overloading Zookeeper
Robust!

Designed input space  Actual input space
Robust!

Designed input space

Actual input space
Epistemology
Foundation of modern science
Newton’s laws of motion

1. When viewed in an inertial reference frame, an object either is at rest or moves at a constant velocity, unless acted upon by an external force.

2. The acceleration of a body is directly proportional to, and in the same direction as, the net force acting on the body, and inversely proportional to its mass. Thus, \( F = ma \), where \( F \) is the net force acting on the object, \( m \) is the mass of the object and \( a \) is the acceleration of the object.

3. When one body exerts a force on a second body, the second body simultaneously exerts a force equal in magnitude and opposite in direction to that of the first body.
Cambridge, we have a problem...
Einstein’s theory of relativity

Sorry, Newton, you’ve been PWNED:

\[ G_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu} \]
\[
\lim_{n \to \infty} \text{approximation}^n(\text{truth}) = \text{truth}
\]
Science algorithm

1. Make observations

2. Find theories consistent with those observations

3. Falsify theories by making more observations
Empiricism
John Locke
Occam’s Razor
Software gets messy
Refactoring
Robust!

Designed input space  Actual input space
Robust!

Designed input space  Actual input space
TESTING
assert(add(2,3) == 5)
assert(add(3,4) == 7)
assert(add(4,5) == 9)
assert(add(5,6) == 11)
Unit testing
Load testing
Stress testing
Fuzz testing
TDD?
Review

1. Cannot perfectly reason about software
   • Infinite regress problem
   • Deduction is fundamentally flawed
   • Evidence shows programmers are not good at deductive reasoning

2. Best you can do is minimize wrongness
   • Truth can only be approximate
   • Observe/theorize/falsify cycle minimizes wrongness over time
   • Testing = empiricism applied to software development
   • Make programs less wrong by testing more
Does any of this matter?
YES
Embrace “your code is wrong” to design better software
Redundancy
Fault-tolerance $>$ Perfection
An example
Learning from Hadoop

Jobtracker

Job

Job

Job
Learning from Hadoop

Jobtracker

Job

Job

Job
Learning from Hadoop

Jobtracker

Job

Job

Job
Your code is wrong
So your processes will crash
Storm’s daemons are process fault-tolerant
Storm

Nimbus

Topology

Topology

Topology
Storm

Nimbus

Topology

Topology

Topology
Storm

Nimbus

Topology

Topology

Topology
Storm

Nimbus

Topology

Topology

Topology
Storm

Nimbus

Topology

Topology

Topology
Robust!

Designed input space

Actual input space
Robust!

- Designed input space
- Actual input space
Reasoning is fundamentally hard
So program in ways that require less of it
public int foo(int a, Object b) {
    int c = this.bar.bar(a);
    if(c>10 && this.dug.helper(b)) {
        return c*2;
    } else {
        return c;
    }
}
public int fib(int n) {
    if (n == 0 || n == 1) return 1;
    else return fib(n-1) + fib(n-2);
}
Pure function
Mutability is hard to reason about
Minimize state mutation
Functional programming
Clojure
skepticism(skepticism)
perfect software
Thank you