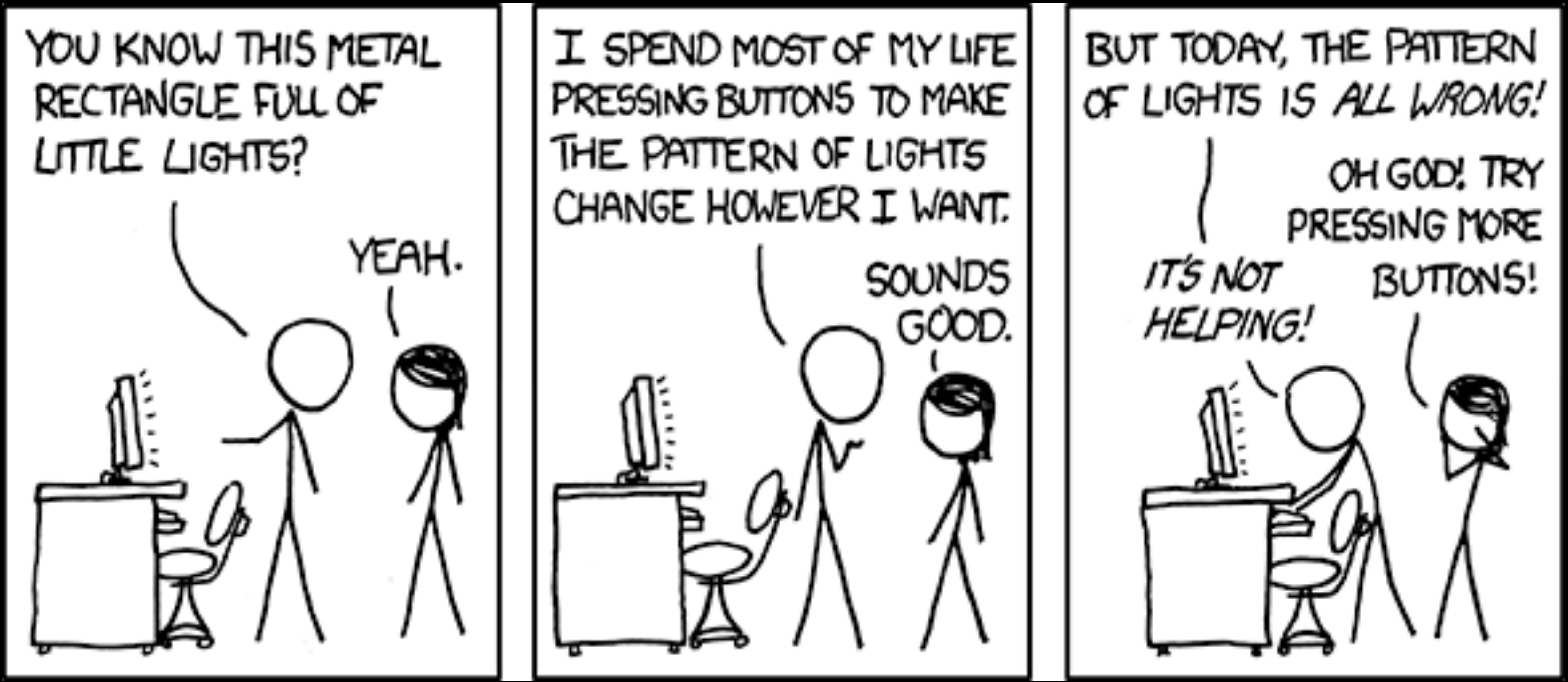
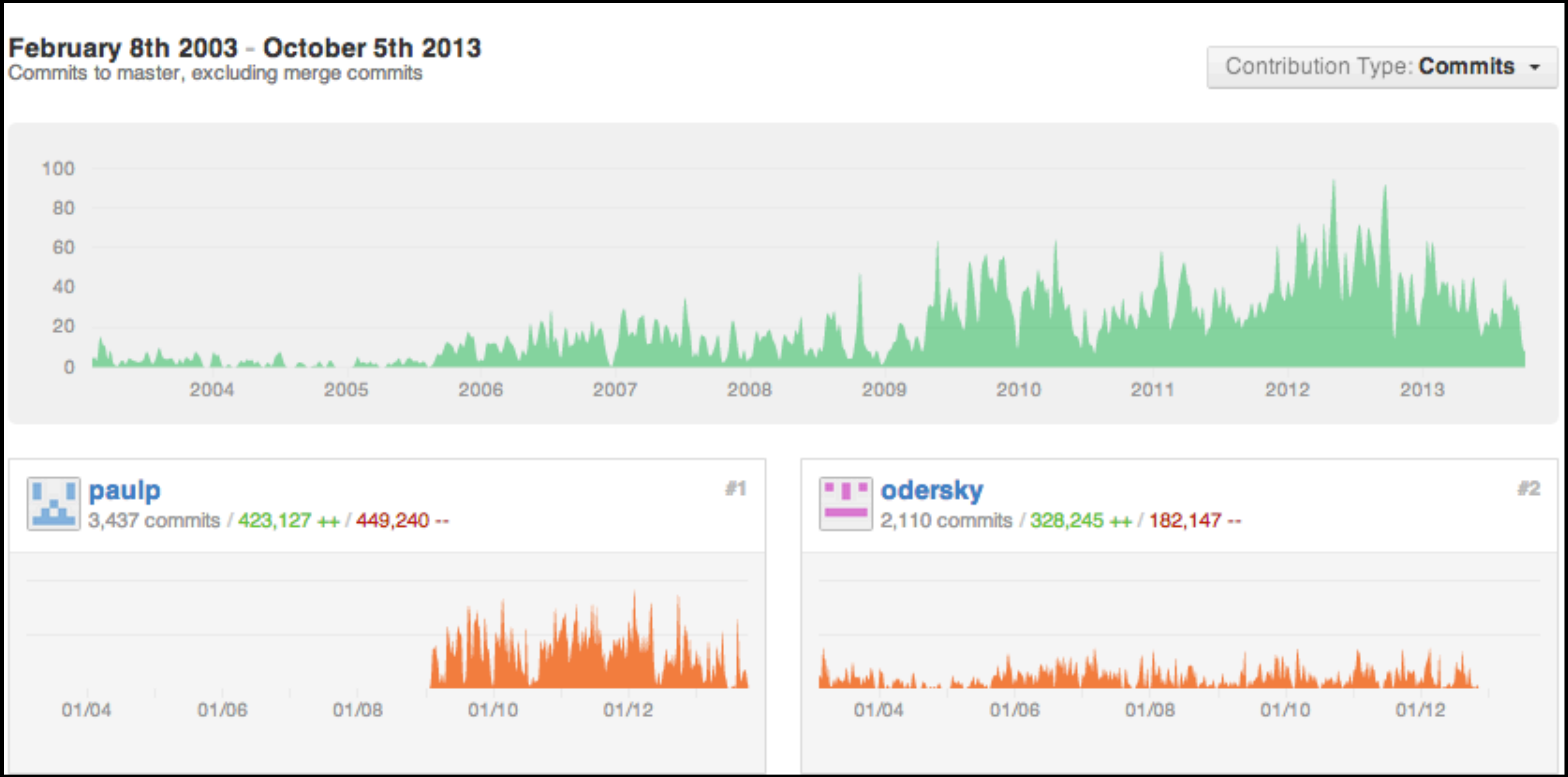
We're Doing it All Wrong

Paul Phillips
paulp@improving.org


YOU KNOW THIS METAL RECTANGLE FULL OF LITTLE LIGHTS?
YEAH.
I SPEND MOST OF MY LIFE PRESSING BUTTONS TO MAKE THE PATTERN OF LIGHTS CHANGE HOWEVER I WANT.
SOUNDS GOOD.
BUT TODAY, THE PATTERN OF LIGHTS IS ALL WRONG!
OH GOD! TRY PRESSING MORE BUTTONS!
IT'S NOT HELPING!

Source: as every single one of you already knows, xkcd.

Credentials



February 8th 2003 - October 5th 2013
Commits to master, excluding merge commits

Contribution Type: **Commits**

paulp #1
3,437 commits / 423,127 ++ / 449,240 --

odersky #2
2,110 commits / 328,245 ++ / 182,147 --

The Winding Stairway

- Five years on scala
- Rooting for scala/typesafe
- But I quit a dream job...
- ...because I lost faith

Um, PNW *Scala*

- I don't say scala is a bad choice relative to today's plausible alternatives
- I do say that scala has issues which leave me in doubt as to its future
- Some of them cannot be addressed
- And others will not be addressed

This Talk in a Nutshell

The nutshell I sought	The nutshell I found

"We're Doing it All Wrong"

- I thought I meant it hyperbolically
- I found it was almost literally true
- So this talk is hardly a taste
- It's a thin line between vision and megalomania: you decide

A Token Nutshell

"Plans are useless, but planning is indispensible."
-- Dwight D. Eisenhower

"Modification is undesirable, but modifiability is paramount."
-- me

If it isn't straightforward to modify, *it will never be any good*.
It will never be fast.
It will never be correct.

And it will eventually be replaced by something modifiable.
...after consuming as many hours as you feed it.

Billions of Bugs

- Many, perhaps most, bugs arise from entanglements with state and time
- Most of what we seek to express has no need for either of them
- Why then do we trundle these deathtraps around with us, everywhere we go?

A Big Sack for 3 Potatoes

- def compare(x: T, y: T): Int
- This relationship admits exactly 3 outcomes
- Represented as a type with 4294967296 states
- A type which is itself overloaded across thousands of distinct and incompatible uses, all of which are silently interchangeable

Pure of heart?

- Maybe you think the (T, T)Int example is unfair, because the signature is for "compatibility" and/or "performance"
- False idols upon whose altars we spill the blood of correctness
- Your false gods do not demand that you compromise the model or the semantics. That we do to ourselves.

"Compatibility means deliberately repeating other people's mistakes." -- Wheeler

SCALA (*sullenly*): It was Java's idea.

SCALA'S MOM (*exasperated*): If Java cut off its nose, would you.... aaagh, hypothetical!

[Next: @ TOM'S RHINOPLASTY]

Self-Inflicted Wounds

- def equals(x: Any): Boolean
- Inflicting this abominable signature on every object in the language is indefensible
- Universal equality is so privileged in scala (pattern matcher, collections, etc.) it's impossible to avoid
- Scala: a capricious mistress

The Biggest Mistake in scala.collection...

- Is: trying to abstract over mutability through inheritance
- An inherited implementation is ALWAYS wrong somewhere!
- Example: how do you write "drop" so it's reusable?
- In a mutable class, drop MUST NOT share
- In an immutable class, drop MUST share
- So half the overrides in collections are there simply to stave off the incorrectness which threatens from above. This is nuts.

Wag or be wagged?

- Compatibility is Tail. Performance is Tail.
- Correctness is Dog.
- "Those who would give up correctness for a little temporary performance deserve neither correctness nor performance."

If Nothing is Forbidden...

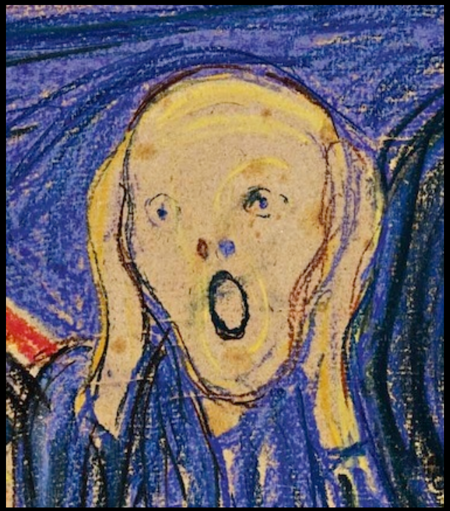
- (1 to 100000000).toList filter (_ => true)
- Oops, we have to make a whole new list because maybe they test if (x1 eq x2)
- We must assume the worst...
- ...so the worst must be less worst.
- Reference equality: inexpressible by default

What's Int.MaxValue between friends?

```
scala> val x1: Float = Long.MaxValue
x1: Float = 9.223372E18

scala> val x2: Float = Long.MaxValue - Int.MaxValue
x2: Float = 9.223372E18

scala> println(x1 == x2)
true
```



NO WONDER NOTHING WORKS

On Performance

- "The greatest performance improvement of all is when a system goes from not-working to working." -- Osterhaut
- "The key to performance is elegance, not battalions of special cases." -- Bentley
- "If you want fast, start with comprehensible." -- me

Contemplate Madness

- The point is not any example in isolation.
- The point is that we are not even trying.
- How do we expect the hard parts to work if we don't even try to get the easy parts right?
- Do you think bridges, airplanes, and nuclear reactors are designed so cavalierly? (If they are, please don't tell me.)

Generality is Not OK

- We try to use a "general purpose" language for everything (or near enough)
- There is no such thing as a general purpose!
- So we are, always and forever, using the wrong tool for the job
- Does i = i + 1 really belong in the same expression domain as type A <: B?

Power: Obvious Benefits, Hidden Costs

- Third law of motion: "To every action there is always an equal and opposite reaction"
- Every increase in expressiveness brings an increased burden on all who care to understand the message
- Unnecessary expressiveness is the enemy
- It racks up a huge body count, but its victims have no voice - it is "silent evidence"

True Power is in Restriction

- Power's less appreciated twin brother
- "The limits of my language are the limits of my world" -- Wittgenstein
- Inexpressibility confers immunity
- Enforce separation of concerns by construction

Good Fences Make Good Neighbors

- Fight impedance mismatches at well defined border layers
- Defend the layer's defining assumptions with your life. This is the Alamo!
- To have to defend against all outcomes at all times is to abandon any hope of progress.

Leaky Abstraction is Not Abstraction

- Opacity is the hallmark of reusability
- The 90% solution is frequently worse than no solution at all
- The 90% solution fails to meet many needs, yet is deployed with the express intention of preventing access to that which underlies it - now it exists only to get in our way!

Not Catchy Enough

~~Good Fences Make Good Neighbors~~

~~True Power is in Restriction~~

~~Leaky Abstraction is Not Abstraction~~

Catchy Enough

- War is Peace
- Freedom is Slavery
- Ignorance is Strength

Too Seldom Seen

But it was then that I saw that such a sketch of a program still to be made, could be regarded as an abstract version of the final program, or even, that it could be regarded as "the program" if such a machine were available; usually it is not available, and actions and data types assumed have to be further detailed in the next levels of refinement. It is the function of these next levels to build the machine that has been assumed to be available at the top level, i.e. at the highest level of abstraction.

-- Dijkstra, July 1970

What I'm after

- I don't need a programming language
- I need a coherent set of tools for creating software. A "language" is incidental.
- The IRs to actually be specified, and implemented via loosely coupled, independent components
- Why is the scala parser entangled with desugaring entangled with type inference entangled with the optimizer entangled with the code generator?
- Why is the canonical code representation a STRING?

What I'm after, cont.

- Two IRs at least: one purely syntactic and carrying types. These can be submitted directly.
- A virtual machine within the virtual machine: designed in concert with the IRs, and designed to interpret them directly.
- Such a VM would free us from the many restrictions imposed by the JVM during development.
- Why do we need JVMTI, JRebel, and huge complexity for live feedback? We don't have to base our development lives on java bytecode!

What I'm after, cont.

- Change as a first class entity. The compiler is pervasively designed around inputs which are Diff[T]s against an in-memory understanding of the last successful compilation. Cold start is the uncommon case of diff against empty.
- Such pervasive and instantaneous feedback to my code changes that both "repl" and "debugger" are superfluous as separate tools.

What I'm after, cont.

- A language where unknowns are assumed in the most restrictive way, so accidental behaviors don't paralyze optimization
- If a feature is ill-advised, why do we knock ourselves out to make it easy to use? Reference equality is a bad idea. Type tests are a bad idea. State is a bad (if sometimes unavoidable) idea.
- You can opt into these things, so we don't ALL have to pay the price.

What I'm after, cont.

- An editor which understands the meaning of the code I am manipulating at a deep level, and which offers many layers of information
- Integration with revision control at the lowest level. My entire life is spent working with sublime and git: how is it that my editor, my DVCS, and my compiler are not in constant communication?
- Whatever you're picturing, it doesn't go as far as I mean for it to go.

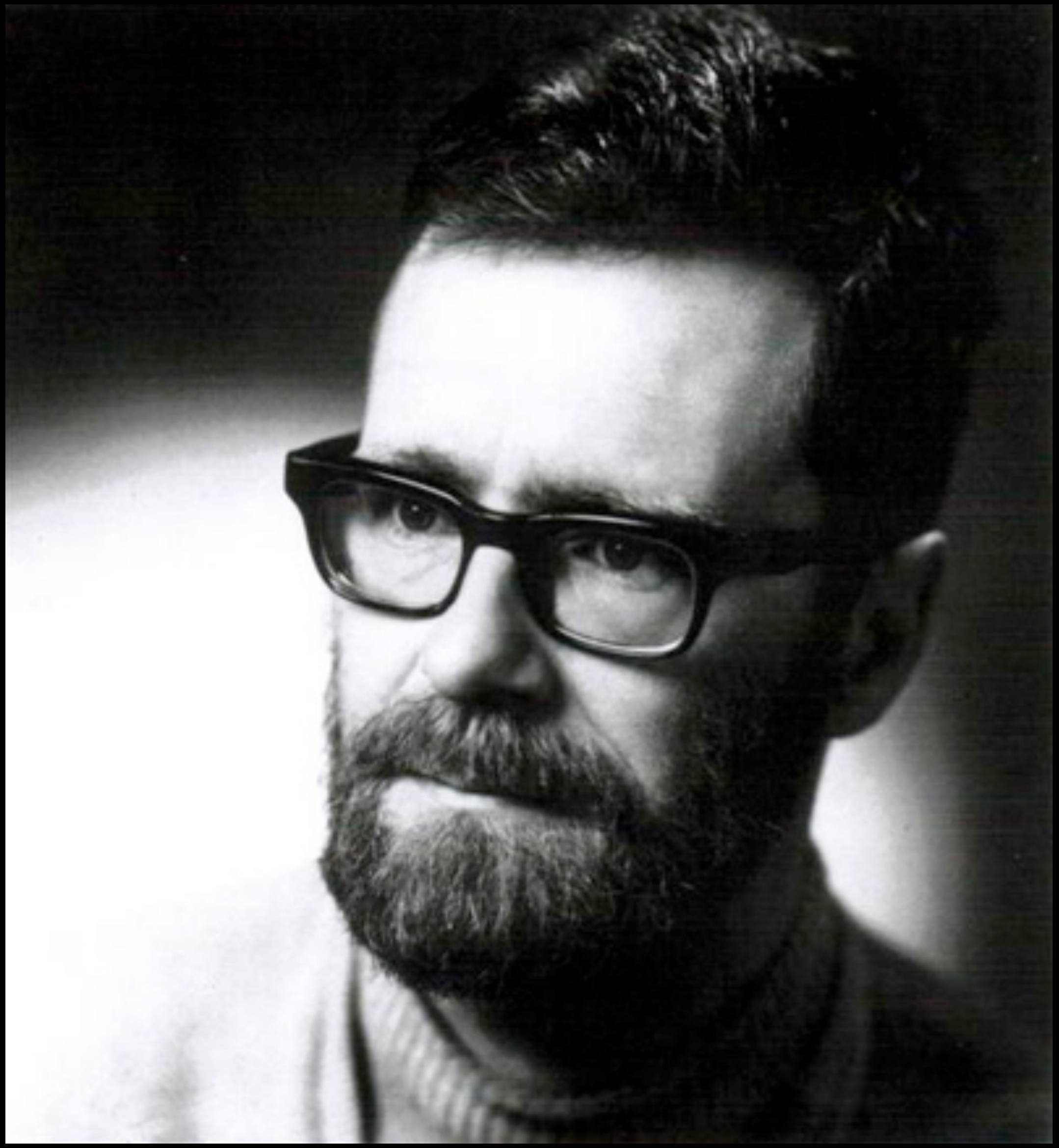
"And to a large extent, the people that we consider to be skilled software engineers are just those people that are really good at playing computer. But if we're writing our code on a computer... why are we simulating what a computer would do in our head? Why doesn't the computer just do it... and show us?"

-- Bret Victor

http://worrydream.com/

"For the present-day darkness in the programmer's world the programmers themselves are responsible and nobody else."

-- Dijkstra (pre-1970)



http://www.cs.utexas.edu/users/EWD/