Continuous Improvement and Constructive Criticism

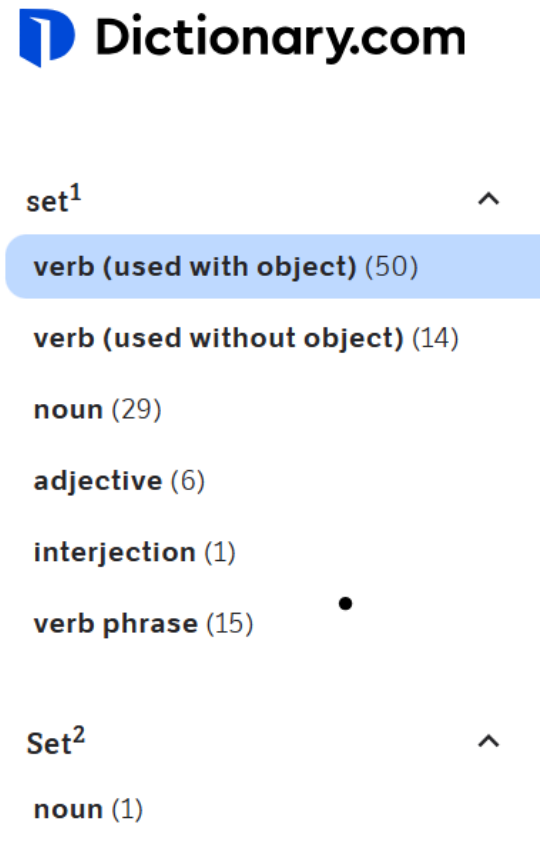
For Software Professionals

Disclaimer

- This is based on my understanding of the subject
- Some of the content might be obvious to some
- There will be some oversimplifications involved
- Happy to receive questions and feedback during the talk

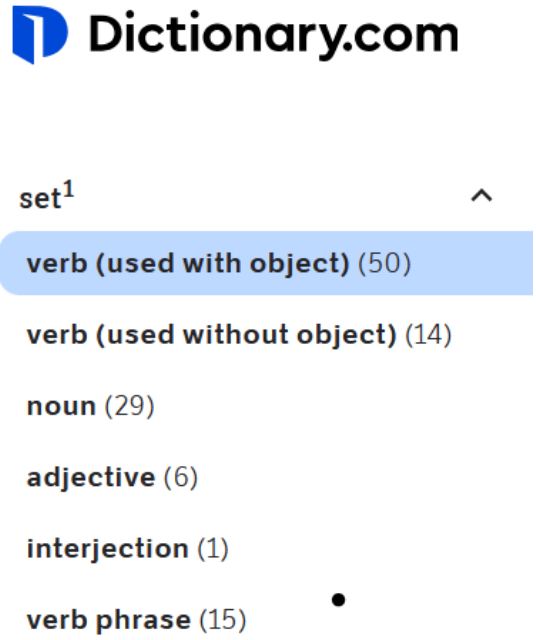
Natural language is awful to communicate

Natural language is awful to communicate



It’s ambiguous and dependent on the context

Natural language is awful to communicate

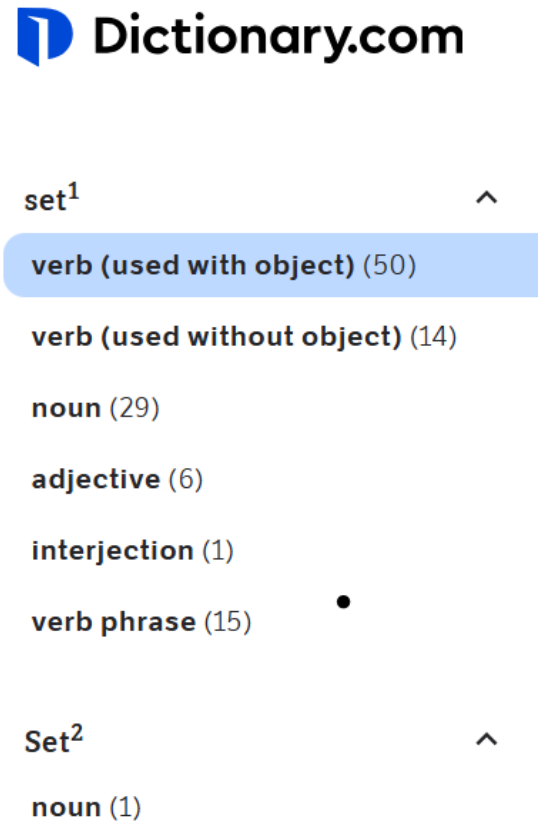


- **Awful** – Literally "full of awe", originally meant "inspiring wonder (or fear)", hence "impressive". In contemporary usage, the word means "extremely bad".

It’s ambiguous and dependent on the context

The meaning changes with time

Natural language is awful to communicate



•**Awful** – Literally "full of awe", originally meant "inspiring wonder (or fear)", hence "impressive". In contemporary usage, the word means "extremely bad".

"Snout-fair", for example, means "having a fair countenance; fair-faced, comely, handsome", while "sillytonian" refers to "a silly or gullible person, esp one considered as belonging to a notional sect of such people".

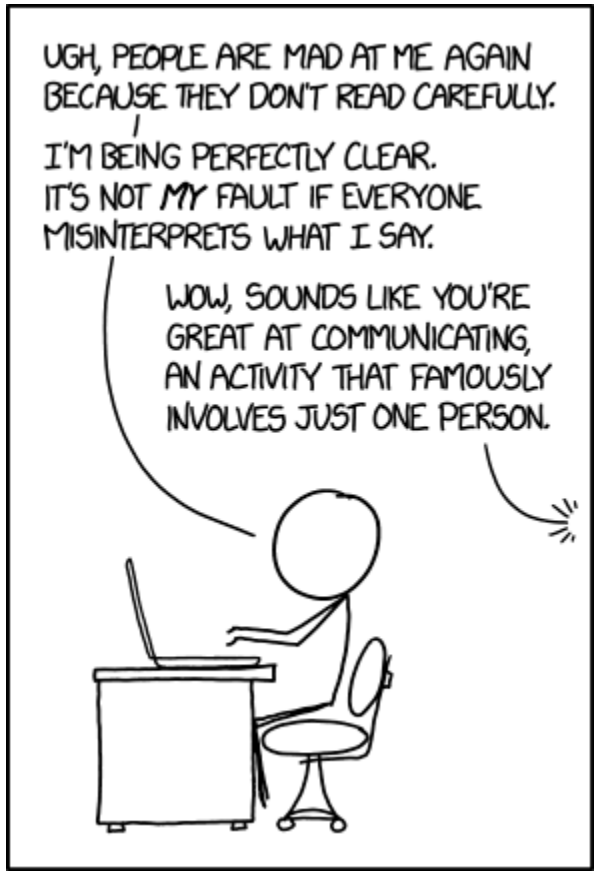
https://www.bbc.co.uk/news/uk-41266000

It's ambiguous and dependent on the context

The meaning changes with time

Some words get lost with time

Natural language is awful to communicate



But computers do exactly what you tell them to!

But computers do exactly what you tell them to!

Do they?

A simple program

```
import time
import custom

def Dosomething(thing):
    x = time.clock()
    y = time.clock()
    return (y-x)/x + custom.scaling * thing
```

A simple program

```
import time
import custom

def Dosomething(thing):
    x = time.clock()
    y = time.clock()
    return (y-x)/x + custom.scaling * thing
```

This is a user defined module – could be different for each system

A simple program

```
import time
import custom

def Dosomething(thing):
    x = time.clock()
    y = time.clock()
    return (y-x)/x + custom.scaling * thing
```

This is a dependency – functions defined here could change with newer versions

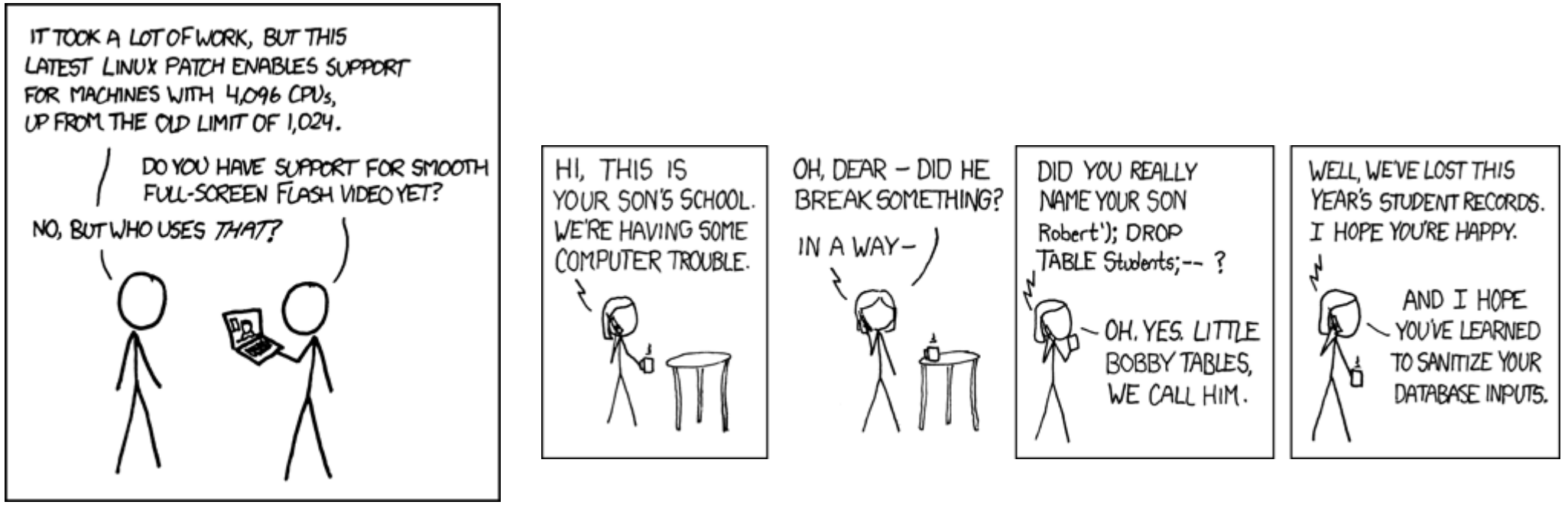
A simple program

```
import time
import custom

def Dosomething(thing):
    x = time.clock()
    y = time.clock()
    return (y-x)/x + custom.scaling * thing
```

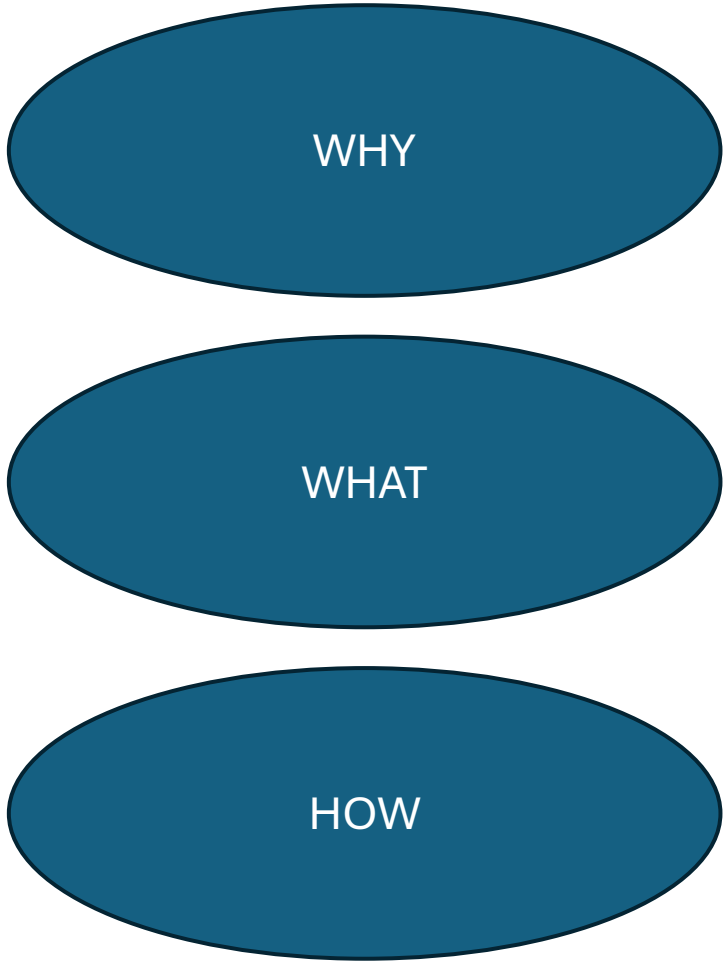
In fact, this function has been deprecated since python 3.3 and was removed in 3.8

And this is without getting into kernel/OS limit overrides, hardware interactions, poor documentation or user inputs...

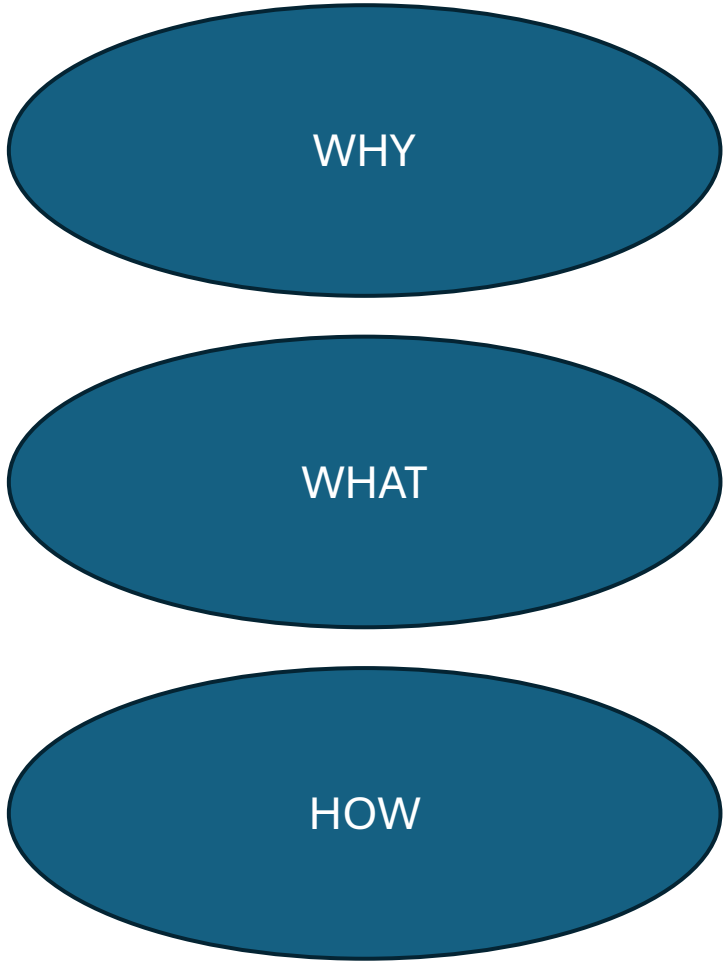


So what can we do?

I find it useful to break it down into 3:

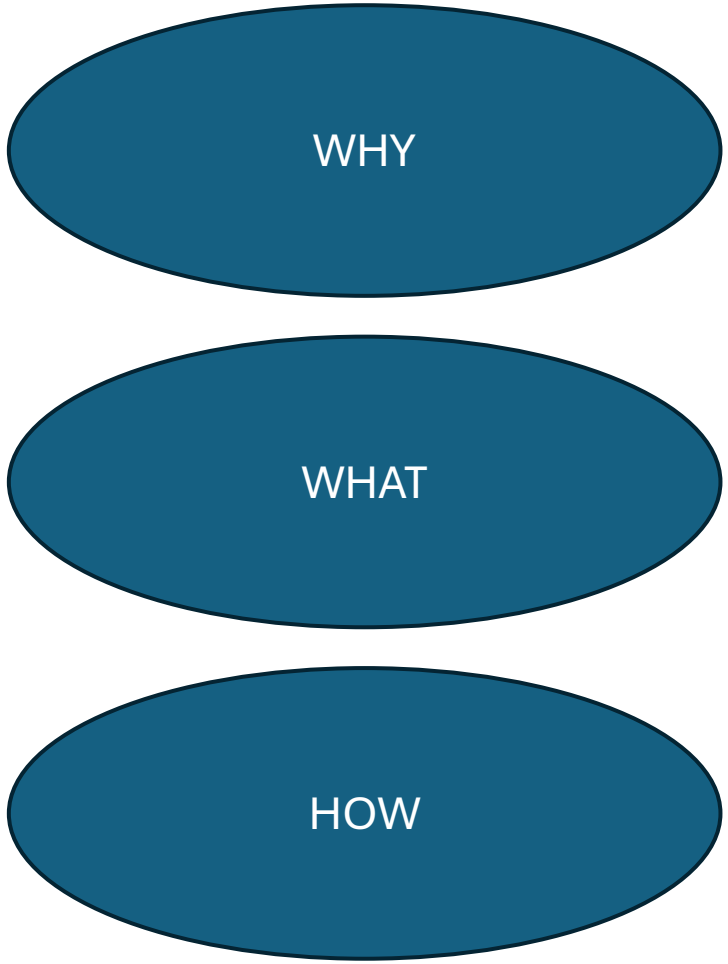


I find it useful to break it down into 3:



Why was this made? Who does this affect?
What are the current circumstances?

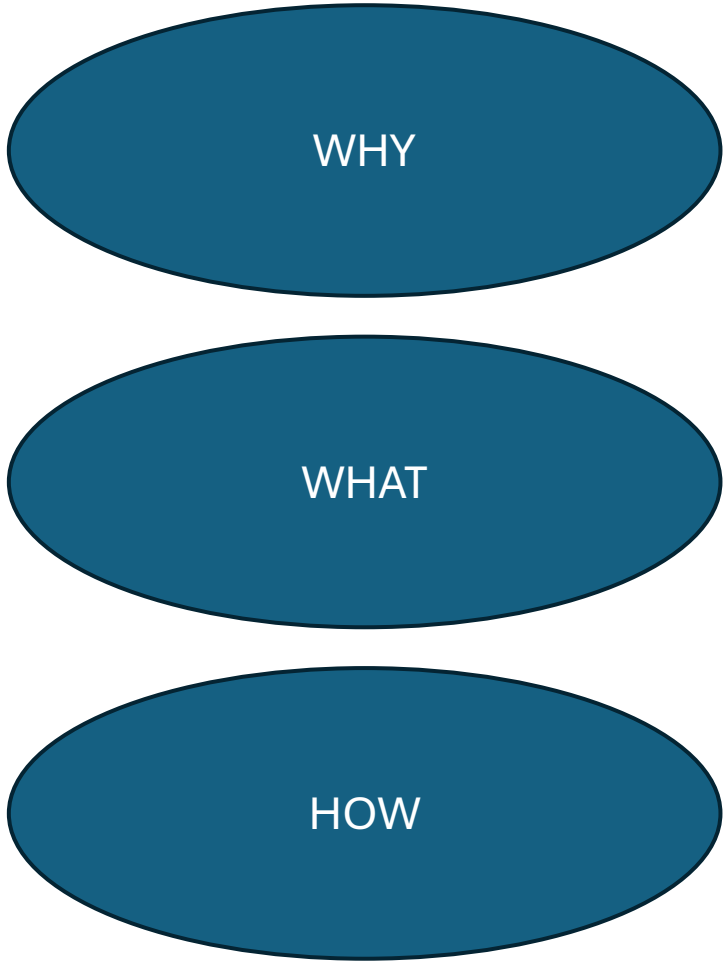
I find it useful to break it down into 3:



Why was this made? Who does this affect?
What are the current circumstances?

What's the core task this is doing?

I find it useful to break it down into 3:



Why was this made? Who does this affect?
What are the current circumstances?

What's the core task this is doing?

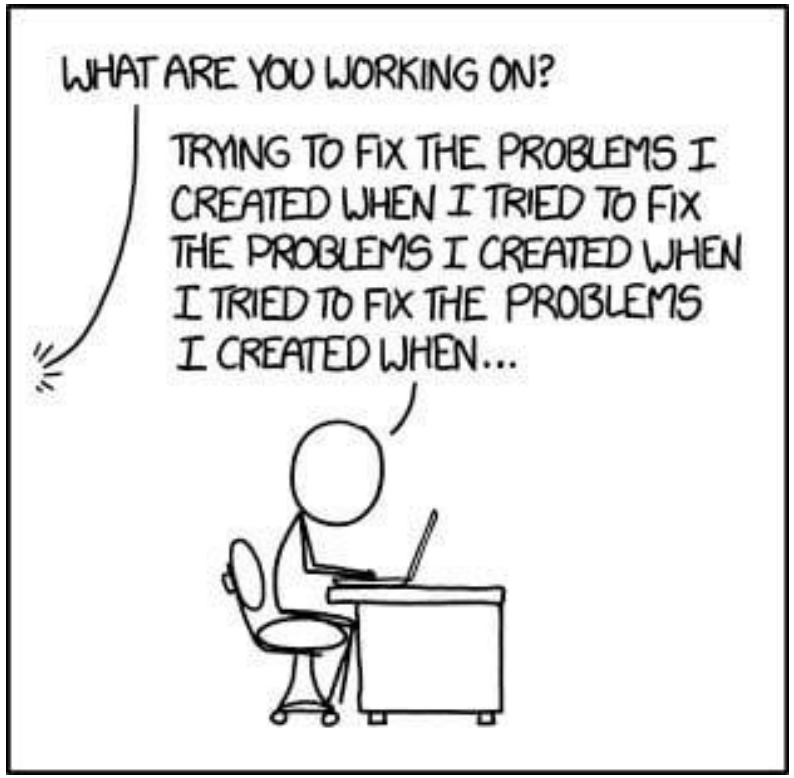
What's the specific implementation of it?

WHY – why was this made?

The motivation for this software or section of software

Is there some historic/political/legal background

What is it trying to do/fix?



WHY – who does this affect?

EVERY CHANGE BREAKS SOMEONE'S WORKFLOW.

stakeholders include:

- users
- admins
- contributors

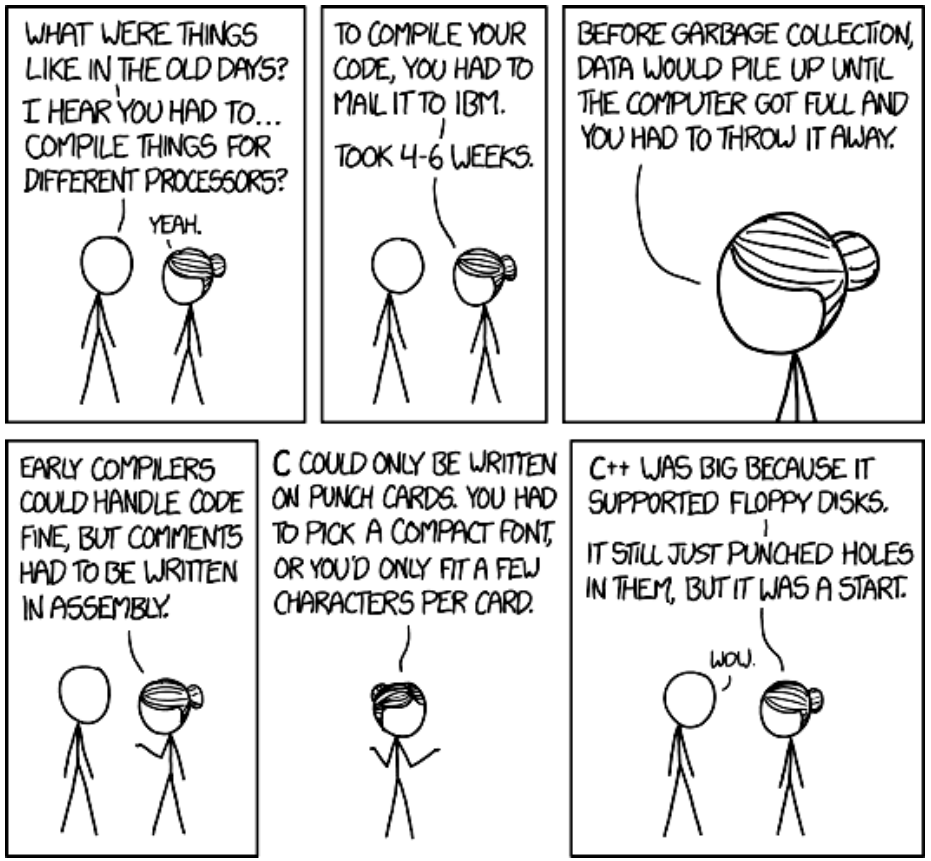
of this software.

Ideally if a change does not affect a group, it should be invisible to them, as this mean

WHY – what are the current circumstances?

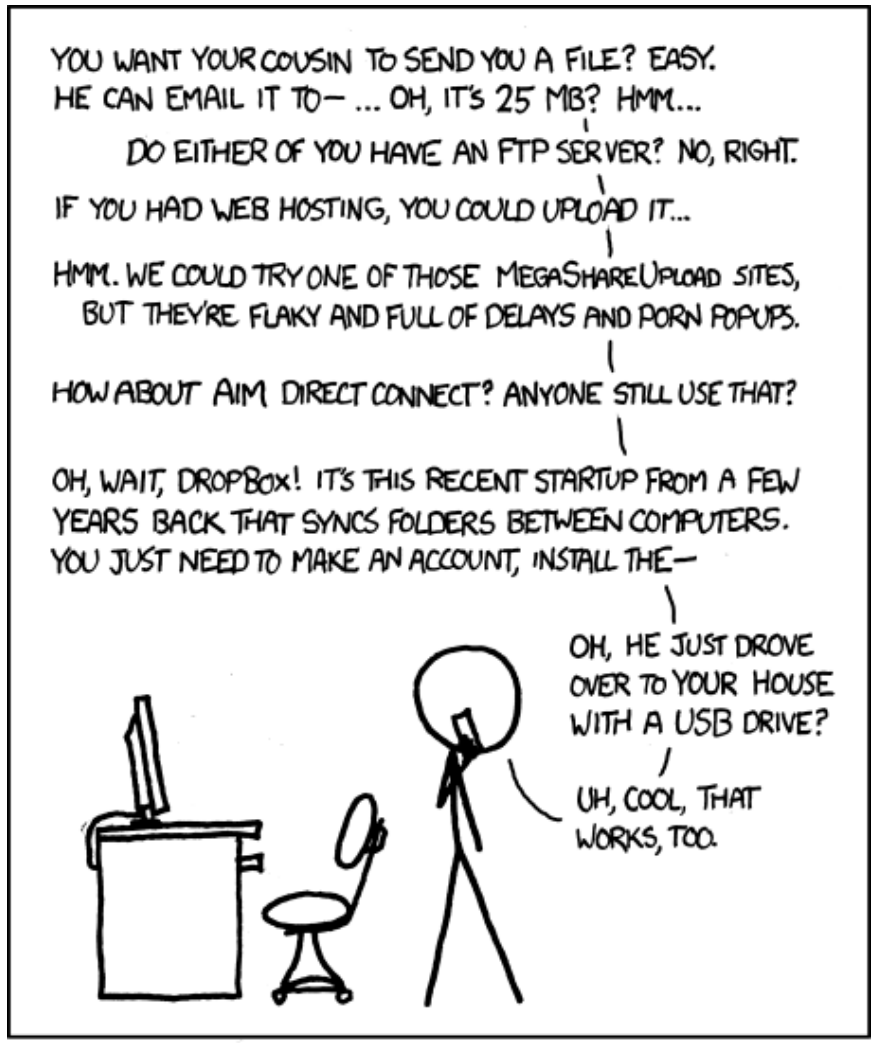
Any external factors affecting this software, e.g.

- Hardware capability
- Usage intensity
- Operational costs
- Available effort
- Running costs



WHAT – what's the core task this is addressing

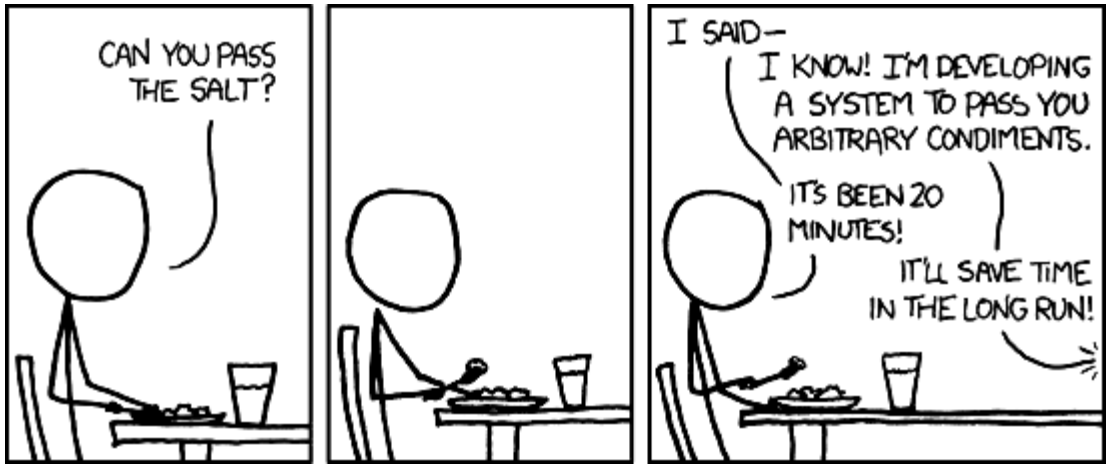
- Try to keep it as detached to the implementation / technical details as possible. This should be understandable by all stakeholders.
- E.g. a way to transfer data between two endpoints
- Sometimes a WHEN, WHERE or WHO might be part of this, e.g. if the task is time critical or needs to fit a pre-existing specification.
  - E.g. a way to sustain 100Gb/s of data between all sites part of the CERN collaboration



I LIKE HOW WE'VE HAD THE INTERNET FOR DECADES, YET "SENDING FILES" IS SOMETHING EARLY ADOPTERS ARE STILL FIGURING OUT HOW TO DO.

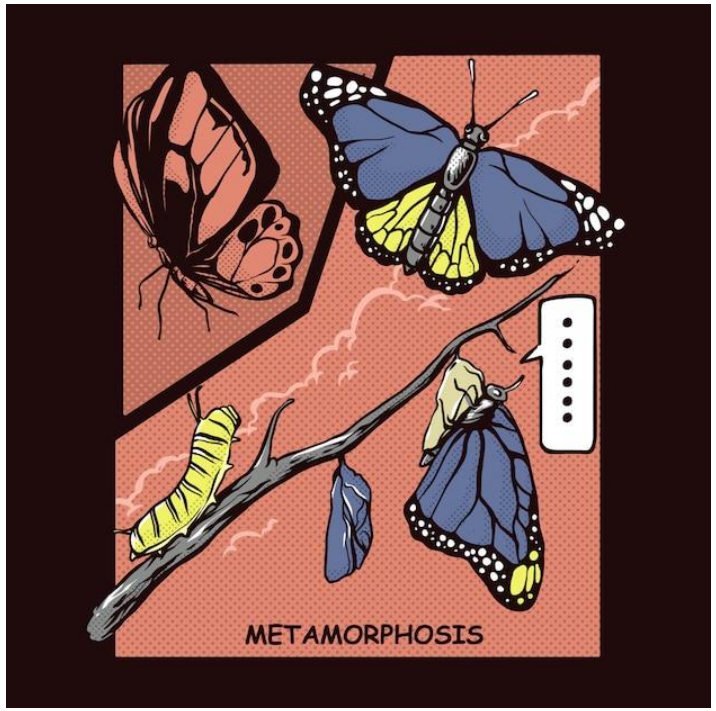
HOW – what's the specific implementation of this task

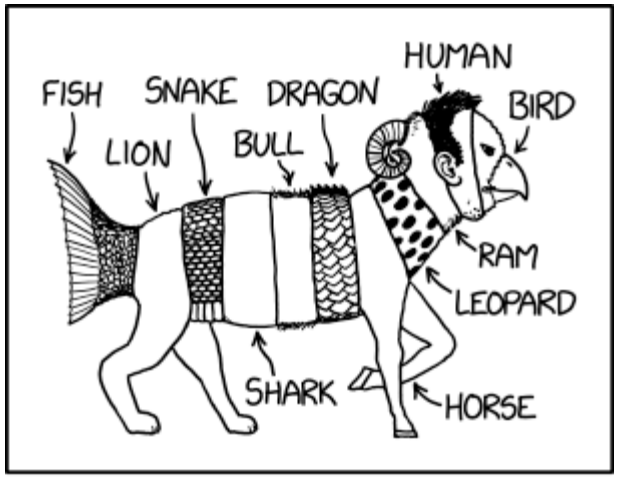
- The implementation doesn't have to be technical
- Depending on the situation, this could be addressed as:
  - A technical problem, e.g. by implementing interfaces to all supported technologies involved,
  - A policy issue, e.g. by agreeing on all sites using the same technology stack
  - Or a combination of both
- There may be valid reasons to pursue either approach
- This WILL change with time, either trough improvements or new feature, or by obsolescence of programming languages or dependencies



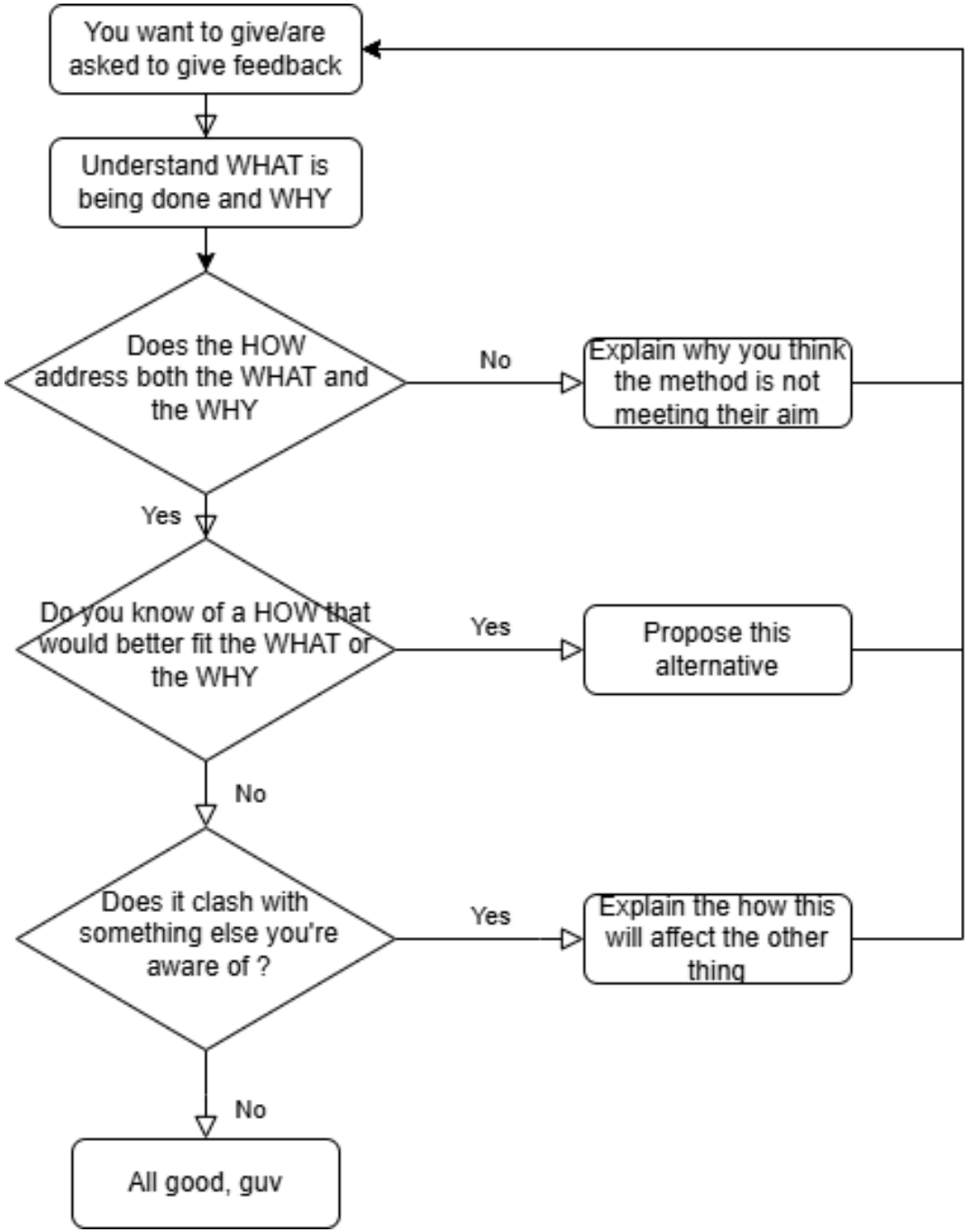
Sometimes, the WHAT and the WHY change…

- New communities join in
- A software finds an unexpected use in a different field
- Scope creep…





THE OMNITAUR



Constructive Criticism

- This is not about micro-optimization
- Often the best solution is not feasible in the circumstances
  - E.g. technical debt mean the effort involved is higher than can be given for this priority
- If you have a method you think would work, you can propose it, but it's up to the actioner if they want to do it that way or a different way.
- This can be used to have a critical look at your own work too

Continuous Improvement

Continuous Improvement

Is a series of implemented Constructive Criticisms

Thank You

Any feedback/questions?