

Building Better Tools

Jan Schaumann

jschauma@netmeister.org

136D 027F DC29 8402 7B42 47D6 7C5B 64AF AF22 6A4C

whoami

```
$ groups  
netbsd yahoo  
$
```



whoami

```
$ groups  
netbsd yahoo  
$
```

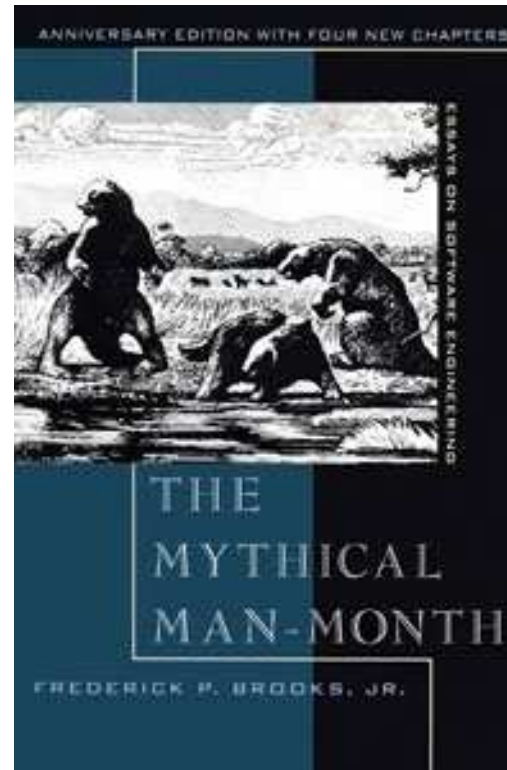


whoami

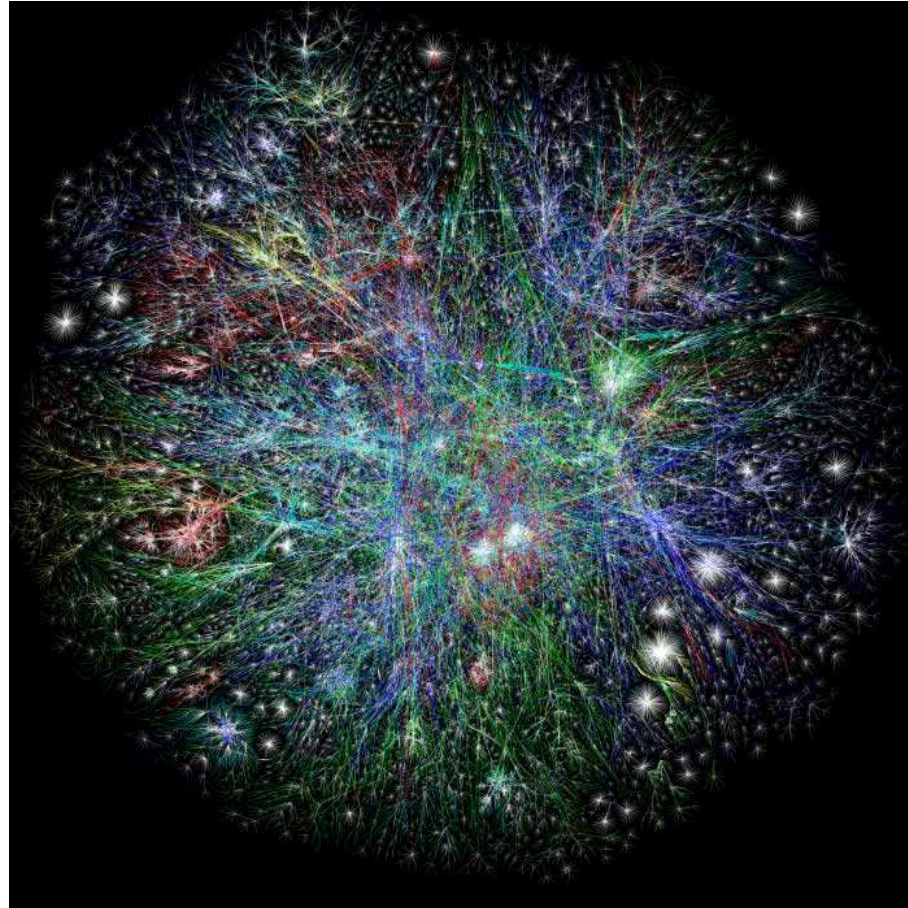
\$ groups
netbsd yahoo
\$



A much better way to spend your time.



People think the internet looks like this.



In reality...



Tools



The right tool?



The right tool?



The right tool?



The right tool?



Internet Reality



Sturgeon's Revelation

90% of everything is crud.

A better hammer!



Tools Development

Three basic categories:

- scripting
- programming
- software development

Tools Development

Three basic categories:

- scripting



Tools Development

Three basic categories:

- scripting
 - automating *very* simple tasks

Tools Development

Three basic categories:

- scripting
 - automating *very* simple tasks
 - customization of user environment

Tools Development

Three basic categories:

- scripting
 - automating *very* simple tasks
 - customization of user environment
 - often only suitable for one individual user

Tools Development

Three basic categories:

- scripting
 - automating *very* simple tasks
 - customization of user environment
 - often only suitable for one individual user
 - usually eventually evolves into larger programs

Tools Development

Three basic categories:

- programming



Tools Development

Three basic categories:

- programming



Tools Development

Three basic categories:

- programming
 - suitable for simple to moderately complex tasks

Tools Development

Three basic categories:

- programming
 - suitable for simple to moderately complex tasks
 - results frequently used by a small base of users

Tools Development

Three basic categories:

- programming
 - suitable for simple to moderately complex tasks
 - results frequently used by a small base of users
 - uses basic framework or common toolkits

Tools Development

Three basic categories:

- programming
 - suitable for simple to moderately complex tasks
 - results frequently used by a small base of users
 - uses basic framework or common toolkits
 - provides consistent interface

Tools Development

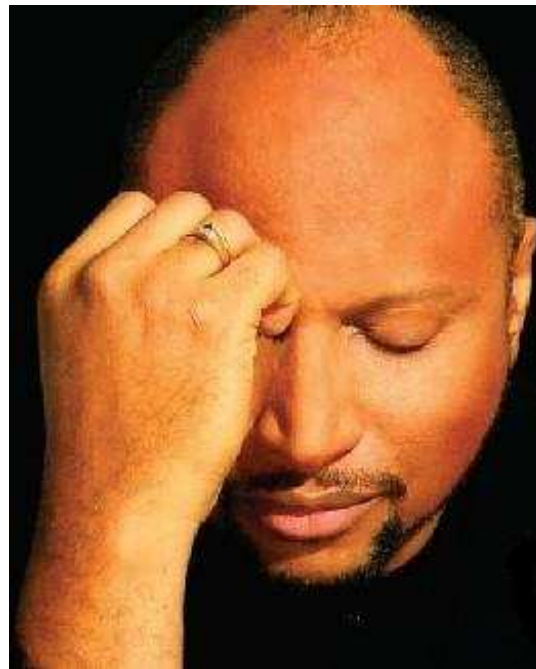
Three basic categories:

- programming
 - suitable for simple to moderately complex tasks
 - results frequently used by a small base of users
 - uses basic framework or common toolkits
 - provides consistent interface
 - may evolve into full product

Tools Development

Three basic categories:

- software development



Felton Pilate, *MC Hammer's producer*

Tools Development

Three basic categories:

- software development



Tools Development

Three basic categories:

- software development
 - required for any reasonably complex task

Tools Development

Three basic categories:

- software development
 - required for any reasonably complex task
 - uses formal software engineering approach (measurable goals, requirements, specifications, ...)

Tools Development

Three basic categories:

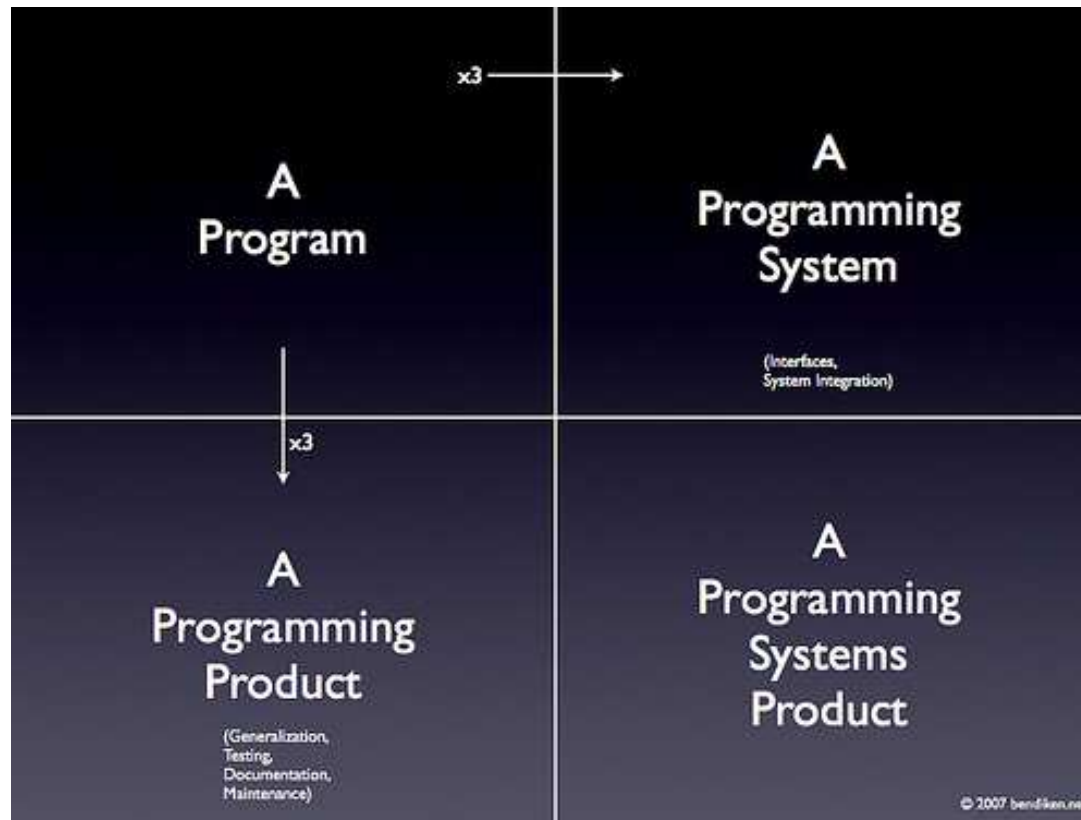
- software development
 - required for any reasonably complex task
 - uses formal software engineering approach (measurable goals, requirements, specifications, ...)
 - may evolve from previous prototypes

Tools Development

Three basic categories:

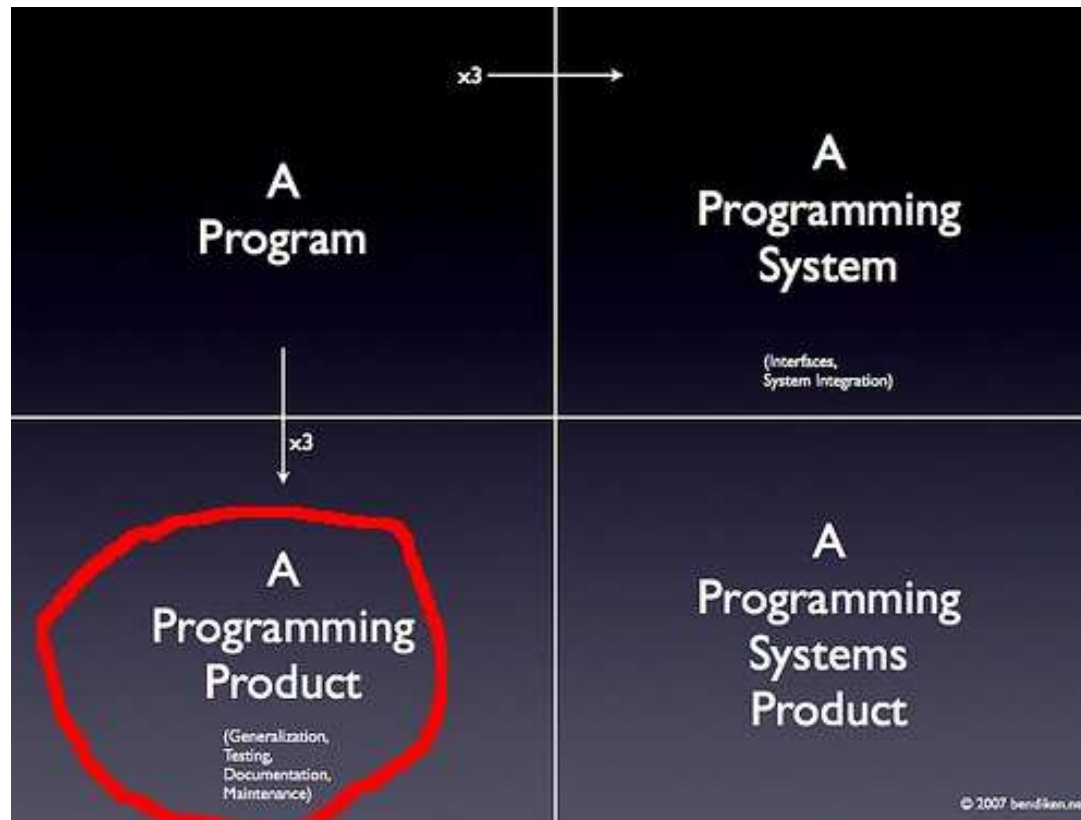
- software development
 - required for any reasonably complex task
 - uses formal software engineering approach (measurable goals, requirements, specifications, ...)
 - may evolve from previous prototypes
 - requires ongoing continuous maintenance / development efforts

Evolution of the programming systems product



F. Brooks, "The Mythical Man Month"

Evolution of the programming systems product

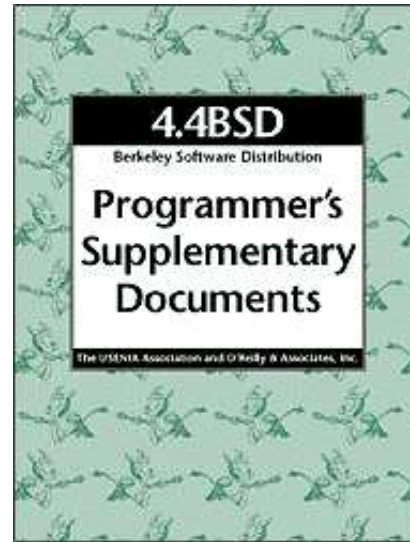
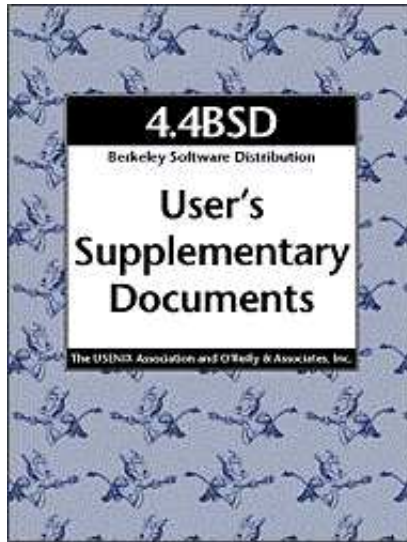


F. Brooks, "The Mythical Man Month"

“Trust me, I know what I’m doing.”

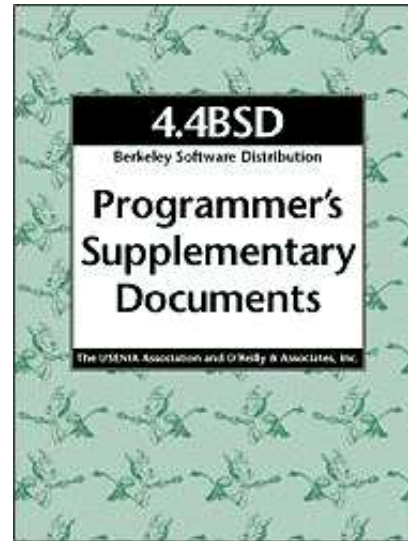
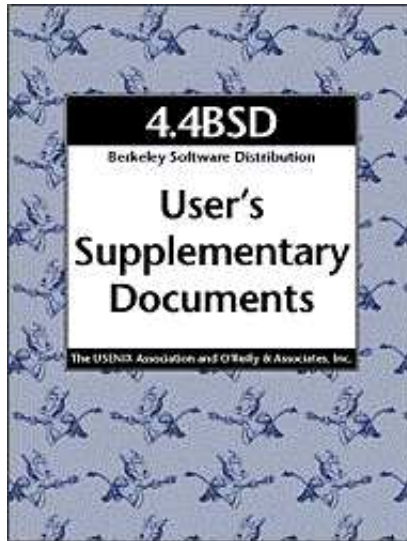


Documentation



“RTFM!”

Documentation



WTFM

User Interface



Unix Philosophy



Unix Philosophy

Do one thing and do it well.

The KISS Principle

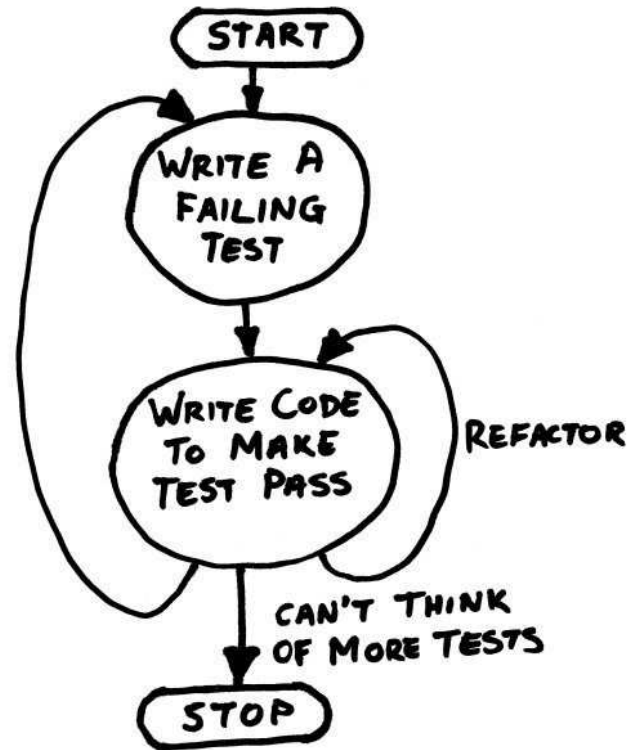


POLA

Principle of Least Astonishment



Test Driven



The Zen of Python

Beautiful is better than ugly.

The Zen of Python

Explicit is better than implicit.

Explicit is better than implicit.

Autovivification is Evil!

```
$oink = { 'foo' => { 'x' => 1, },  
        'bar' => { 'y' => 2, },  
        } ;  
if (exists($oink->{'baz'}{'z'})) {  
    print "oink->baz->z exists\n";  
}  
if (exists($oink->{'baz'})) {  
    print "oink->baz exists\n";  
}
```

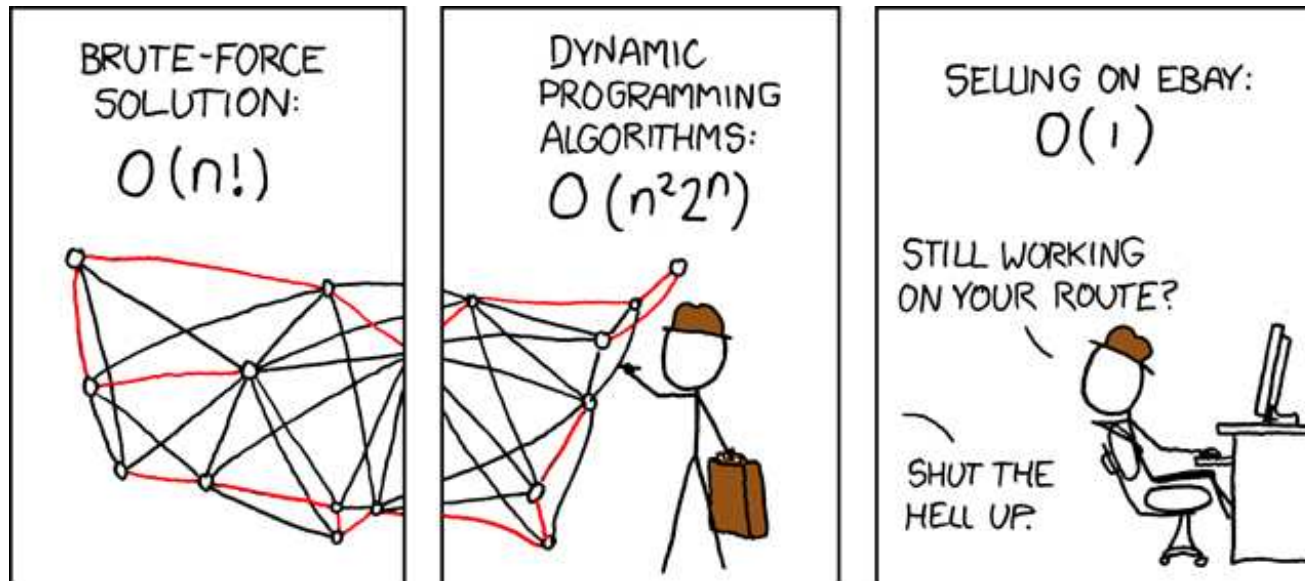
Explicit is better than implicit.

For \$Deity's sake, return *something*.

```
sub func {
  my ($ref, $ref_or_is_it_something_else) = @_;
  for my $a (@{$ref}) {
    if (condition) {
      return 1;
    }
  }
}
```

The Zen of Python

Simple is better than complex.



<http://xkcd.com/399/>

The Zen of Python

Complex is better than complicated.

The Zen of Python

Flat is better than nested.

The Zen of Python

Sparse is better than dense.

The Zen of Python

Readability counts.

The Zen of Python

Special cases aren't special enough to break the rules.

The Zen of Python

Special cases aren't special enough to break the rules.

Although practicality beats purity.

The Zen of Python

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Sturgeon's Law: *Nothing is always absolutely so.*

The Zen of Python

Errors should never pass silently.

The Zen of Python

Errors should never pass silently.

(That would be implicitly accepted failure.)

The Zen of Python

Errors should never pass silently.

(That would be implicitly accepted failure.)

(You know what would be better than something *implicit*?)

The Zen of Python

Errors should never pass silently.

(That would be implicitly accepted failure.)

(You know what would be better than something *implicit*?)

(Why, of course, something *explicit*!)

The Zen of Python

Errors should never pass silently.

Unless explicitly silenced.

The Zen of Python

In the face of ambiguity, refuse the temptation to guess.

The Zen of Python

There should be one – and preferably only one – obvious way to do it.

The Zen of Python

There should be one – and preferably only one – obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.



The Zen of Python

Now is better than never.

The Zen of Python

Now is better than never.

Although never is often better than *right* now.

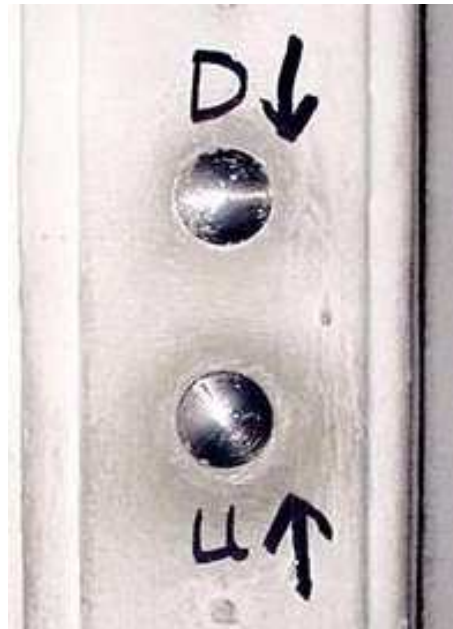
The Zen of Python

If the implementation is hard to explain, it's a bad idea.

The Zen of Python

If the implementation is easy to explain,
it *may* be a good idea.

A simple interface, easy to explain. Yet...



The Zen of Python

Namespaces are one honking great idea –
let's do more of those!

Consistency



Robustness Principle or Postel's Law

Be conservative in what you do; be liberal in what you accept from others.

Toss it!



Avoid the Quick Fix

There's nothing as permanent as a temporary solution.

Take a good look in the mirror!



Nice Ass!

Take a good look in the mirror!

Until you can *prove* otherwise,
assume that *you* are an Ass!

Don't try to be clever!

insert

clever sentence or image about how trying to be
clever increases complexity and leads to code that
is difficult to understand

here

Avoid the Project That Was Never Finished

Don't let the Perfect be the enemy of the Good.

Avoid Feature Creep



<http://www.feepingcreatures.com>

Release Early, Release Often

“More users find more bugs.”

F. Brooks, “The Mythical Man Month”

Increase the Bus Factor



"Just friends."

Fix Broken Windows



Program Maintenance

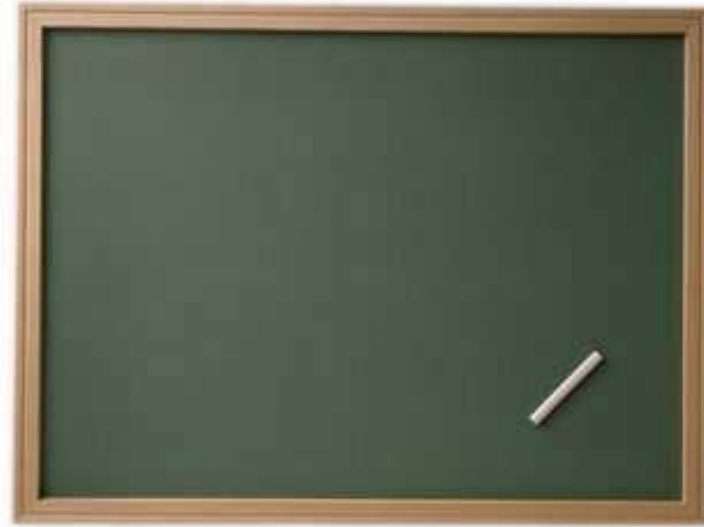
“... is an entropy-increasing process, and even its most skillful execution only delays the subsidence of the system into unfixable obsolescence.”

F. Brooks, “The Mythical Man Month”

Toss it!



Starting fresh



Ownership



Done!

