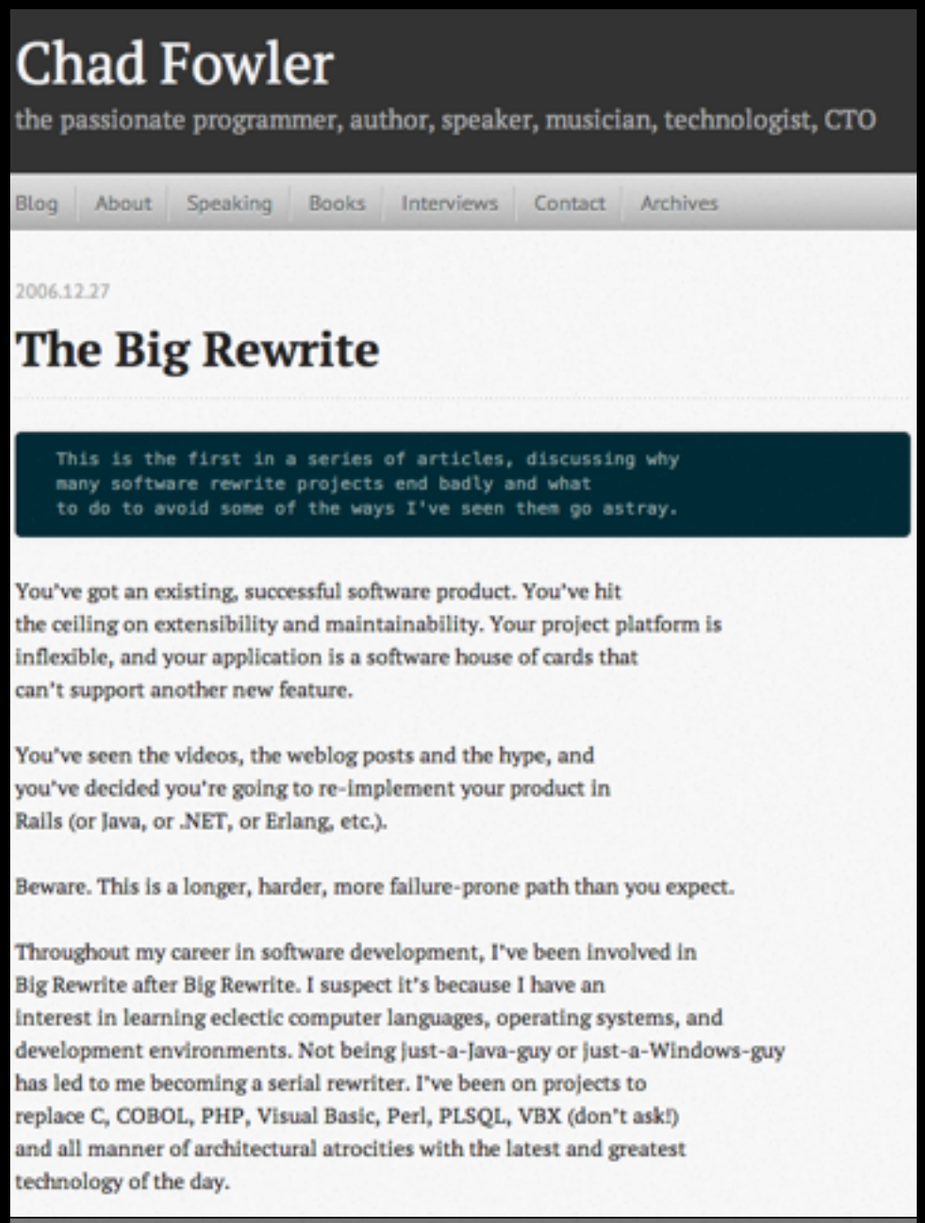
Legacy

@chadfowler

Systems Euthanizer



Chad Fowler

the passionate programmer, author, speaker, musician, technologist, CTO

Blog | About | Speaking | Books | Interviews | Contact | Archives

2006.12.27

The Big Rewrite

This is the first in a series of articles, discussing why many software rewrite projects end badly and what to do to avoid some of the ways I've seen them go astray.

You've got an existing, successful software product. You've hit the ceiling on extensibility and maintainability. Your project platform is inflexible, and your application is a software house of cards that can't support another new feature.

You've seen the videos, the weblog posts and the hype, and you've decided you're going to re-implement your product in Rails (or Java, or .NET, or Erlang, etc.).

Beware. This is a longer, harder, more failure-prone path than you expect.

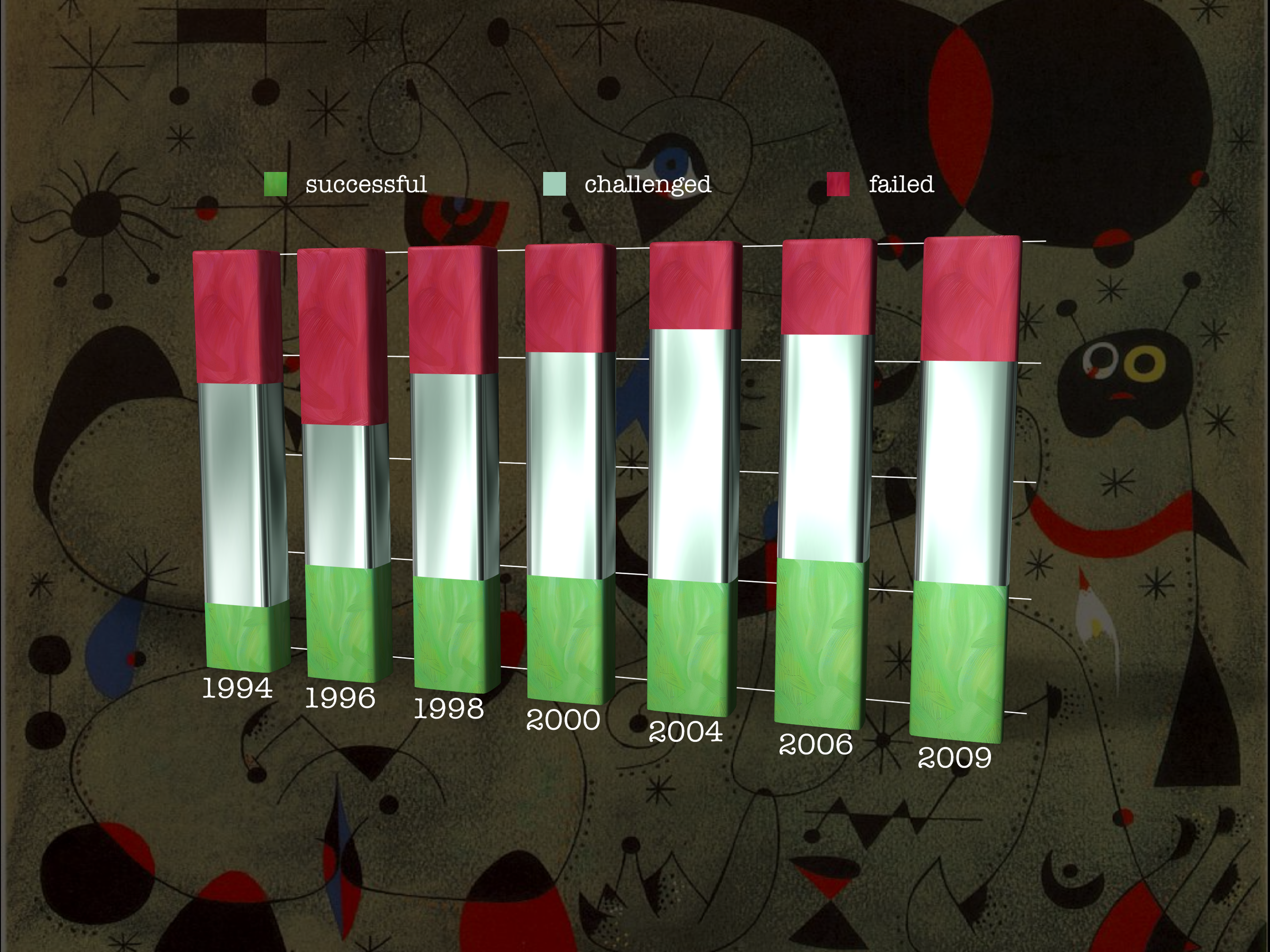
Throughout my career in software development, I've been involved in Big Rewrite after Big Rewrite. I suspect it's because I have an interest in learning eclectic computer languages, operating systems, and development environments. Not being just-a-Java-guy or just-a-Windows-guy has led to me becoming a serial rewriter. I've been on projects to replace C, COBOL, PHP, Visual Basic, Perl, PLSQL, VBX (don't ask!) and all manner of architectural atrocities with the latest and greatest technology of the day.

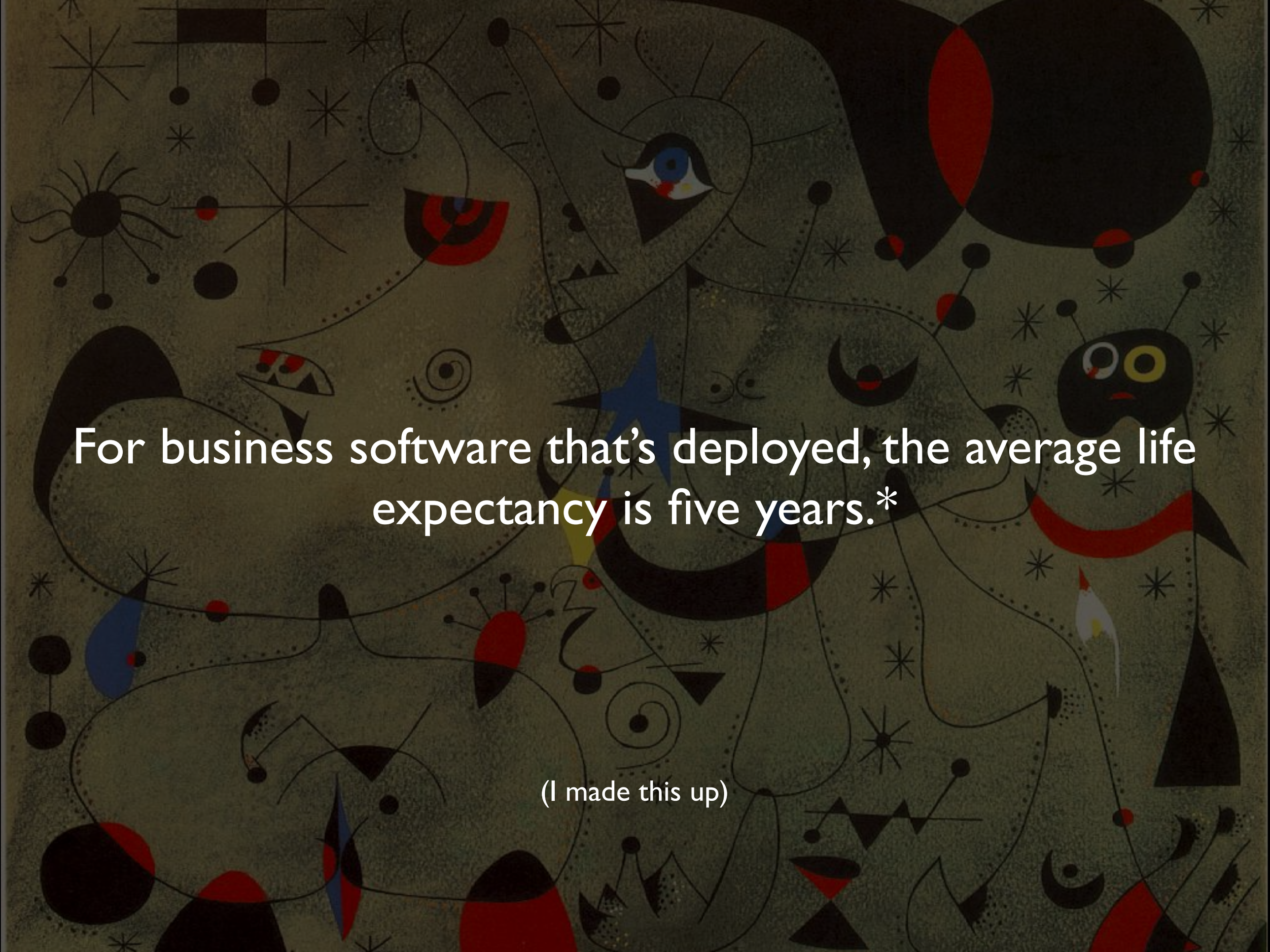
"legacy"

1 *a legacy from a great aunt*: BEQUEST, inheritance, heritage, endowment, gift, patrimony, settlement, birthright; formal benefaction.

2 *a legacy of the wars*: CONSEQUENCE, effect, upshot, spin-off, repercussion, aftermath, by-product, result.


C. Provença
257 / 292
La Pedrera


successful
challenged
failed
1994
1996
1998
2000
2004
2006
2009



For business software that's deployed, the average life expectancy is five years.*

(I made this up)

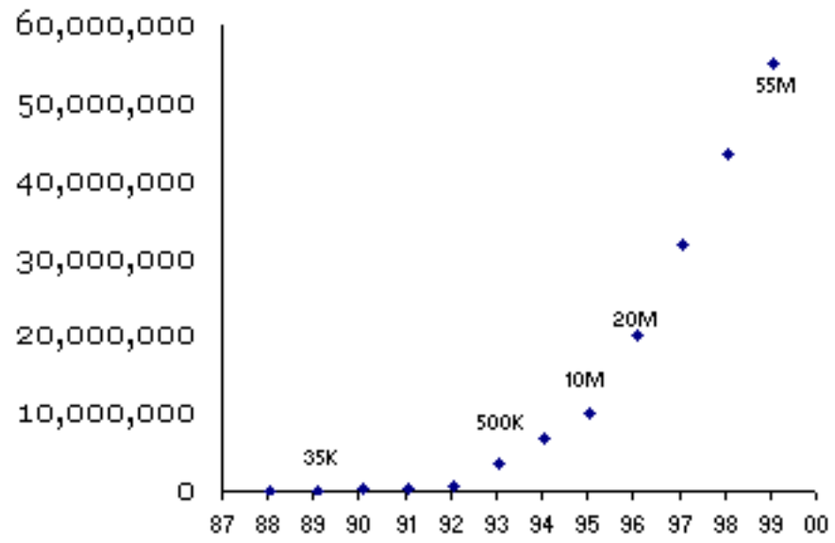
Joel on Software

Good Software Takes Ten Years. Get Used To it.

by Joel Spolsky

Saturday, July 21, 2001

Have a look at this little chart:



Nobody

remember

will

work

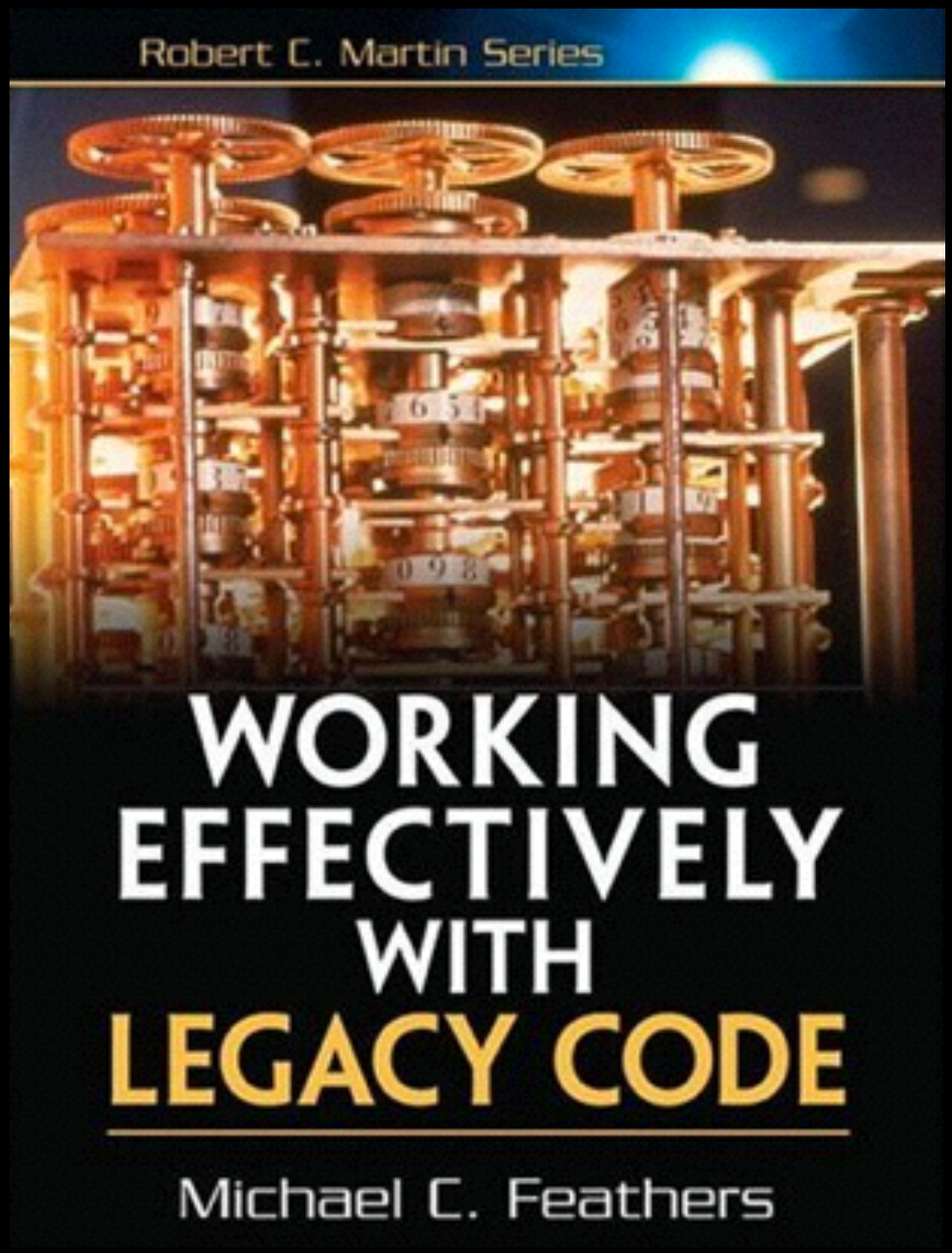
die

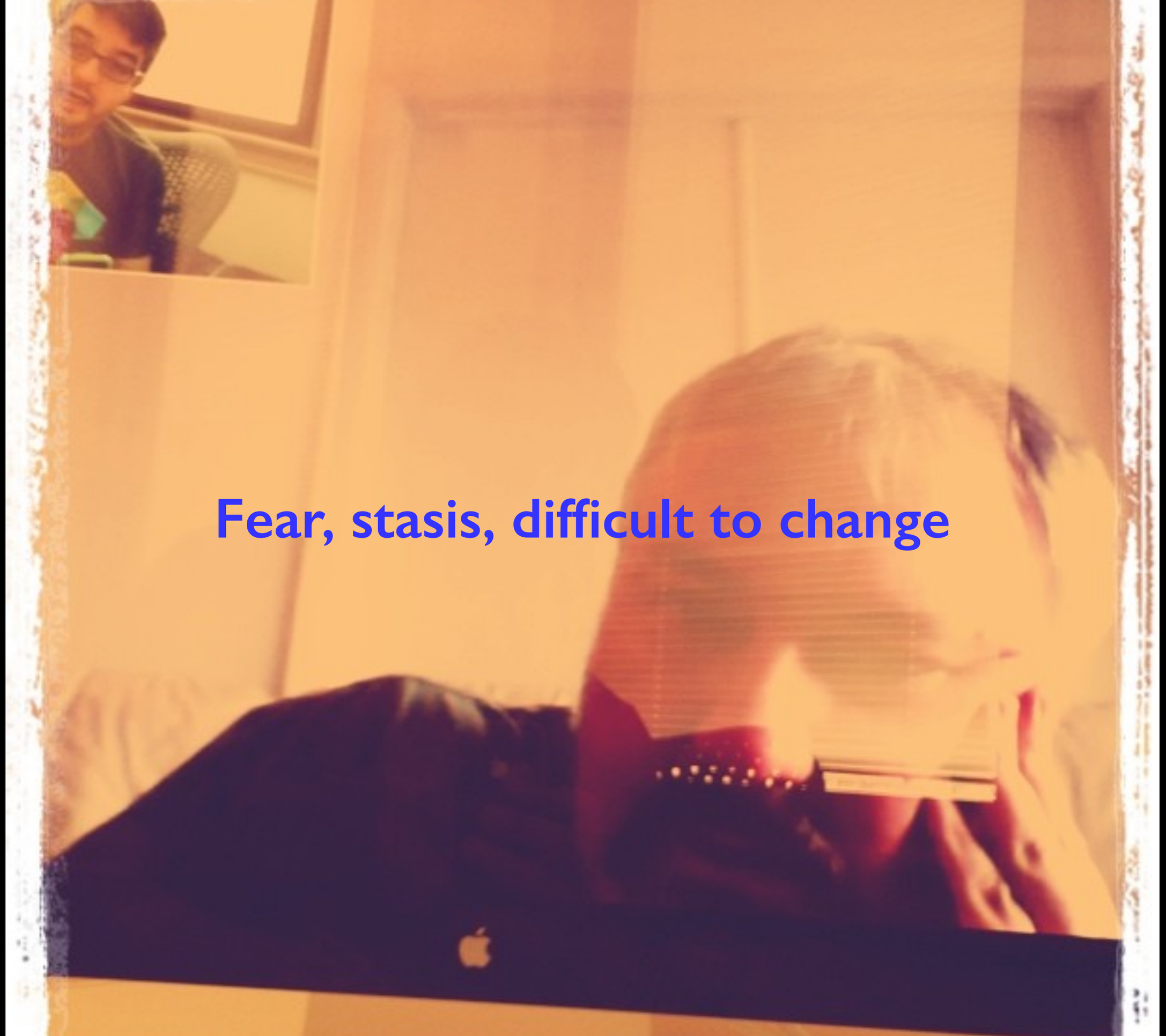
your

you

when

How do you CREATE Legacy software?


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers



Fear, stasis, difficult to change

Software developers, what are the characteristics (internal & external) that make rare OLD still-in-use software survive?

Posted at 2011-06-15 05:23:42
via Echofon
From: Washington DC, USA

negative bias

Friend

thomasfuchs
10,754 followers

fear of awesome

bokmann
274 followers

sunk cost fallacy.

heavysixer
159 followers

OLD still-in-use managers.

but also:

sujayghosh
107 followers

What keeps old software alive is a strng roadmap and value addition.

1 day ago - Reply Bangalore,India

jcrossley3
224 followers

It works.

shilesh_kumar
8 followers

Stability

TechScruggs
403 followers

one's that adhere to the unix philosophy: do one thing and do one thing well.

http://37signals.com/svn/archives2/predicting_06_enterprise_is_the_new_legacy.php

Predicting '06: Enterprise is the new legacy David 27 Dec 2005

53 comments Latest by Tim

In the face of the new year, here's a single 37signals' prediction for 20

"Careful. "Legacy" isn't a bad word. "Legacy" usually means tried, true, and of enough value that it lasted long enough to be old and outdated.

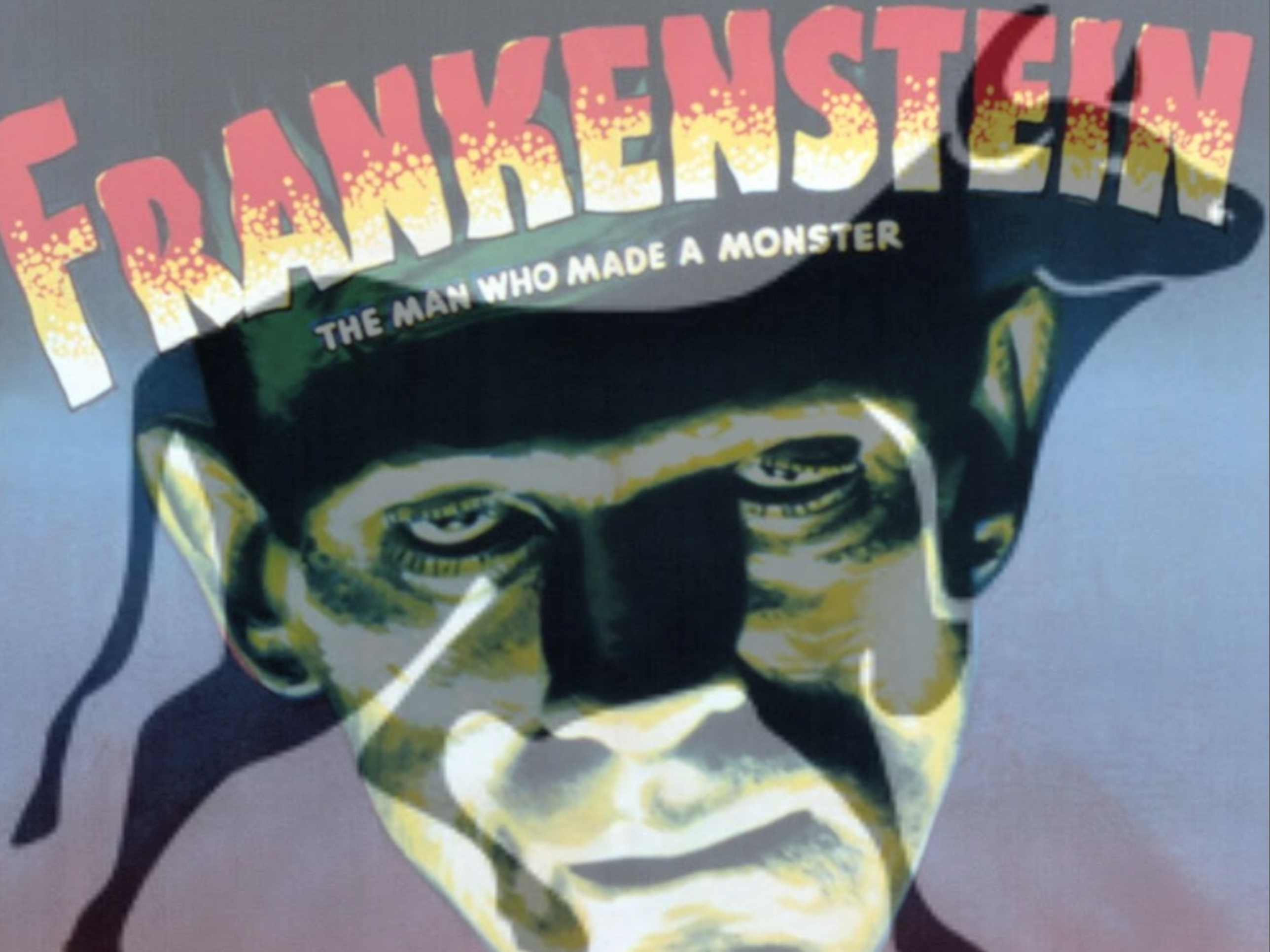
To mock "Legacy" is to look at the successes of the past and to declare that they aren't to be revered or respected. Most of what runs our economies is "Legacy".

In the future, I hope that the software I'm creating now was highly regarded enough that it's still around and being referred to as "Legacy".

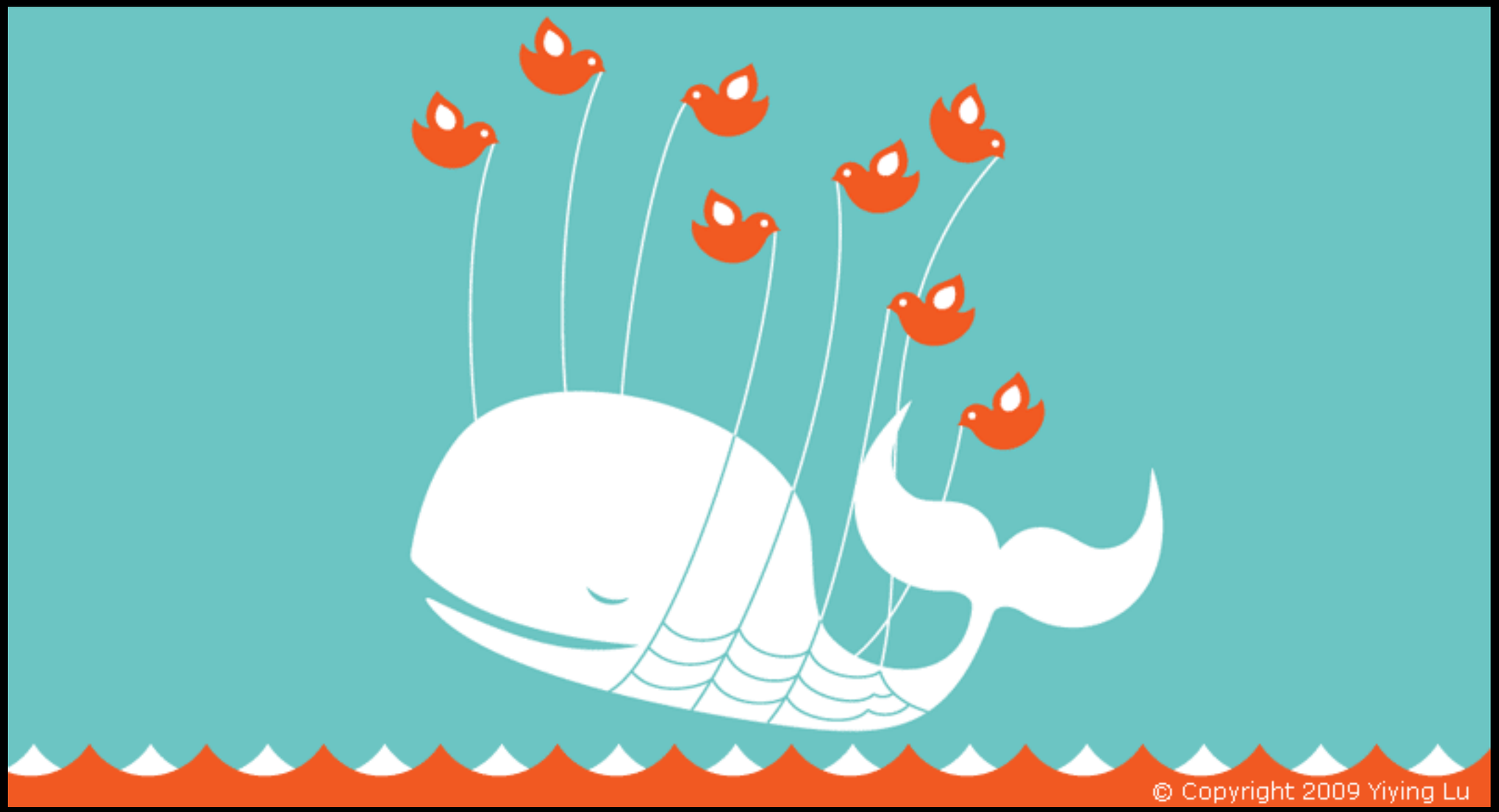
it wasn’t :(

FRANKENSTEIN

THE MAN WHO MADE A MONSTER



How do you create systems that survive?



Step 1: Has to be born

http://tinyurl.com/codealive

UR SOFTWARE IS ALIVE!!!!!

It all comes back to one thing: code survives by providing value and by being difficult to replace.

Value > Difficulty

The primordial soup is chunky with SQL, ant scripts, and old servlet carcasses. Time goes on, and complexity builds.

http://tinyurl.com/ulsgabriel

richard p. gabriel

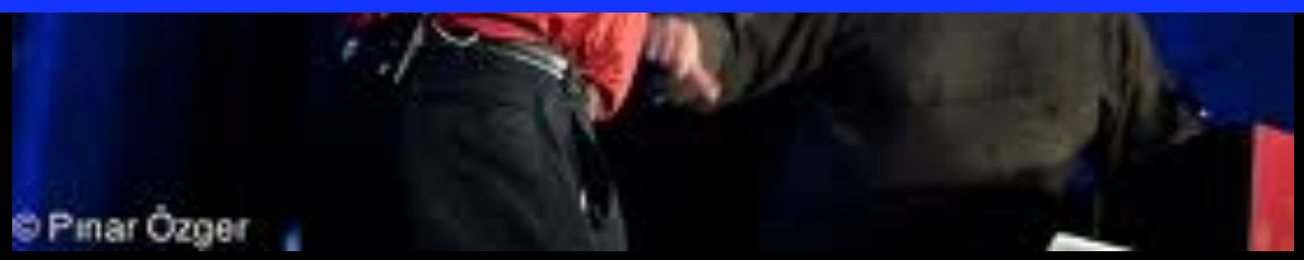
In this presentation I talk about trillions of lines of code in order to emphasize a scale way beyond what we think of as remotely feasible today. This is an exaggeration because Grady Booch has estimated that collectively, humankind has produced a total of about a trillion lines of code since programming as we know it began in 1945

http://tinyurl.com/ulsgabriel

richard p. gabriel

Biological

systems are very much larger than anything (coherent) that people have built.



How do we create systems that outlast us?

homeostasis

Homeostasis

Definition

noun

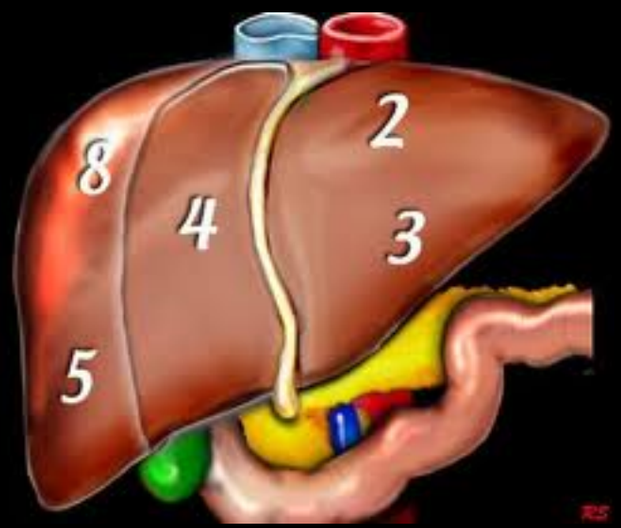
(Science: Biology)

(1) The tendency of an organism or a cell to regulate its internal conditions, usually by a system of feedback controls, so as to stabilize health and functioning, regardless of the outside changing conditions

(2) The ability of the body or a cell to seek and maintain a condition of equilibrium or stability within its internal environment when dealing with external changes

brain

liver



Metabolize toxic substances

kidney

Blood water level, re-absorption of substances into blood, excretion

“An inability to maintain homeostasis may lead to death or a disease, a condition known as ***homeostatic imbalance****.”*

You are dying right now!

50 trillion cells in your body
3 million die per second

* this is a guess

Friend

glv

1,589 followers

We've learned that software should start small and grow; challenging to replace an existing system that way.

What are the oldest surviving software systems you regularly use? GNU Linux comes to mind. What else?

emacs

“UNIX”

BSD

C-language toolchain

grep

Apache

X-Windows System

make

Small components

Systems

http://www.flickr.com/photos/joeshlabotnik/4991688551/sizes/o/in/photostream/

What is a cell?

What is a system?

When do you build a system vs. a cell?

Are you building the right one now?

tiny components

```
%w.rack tilt date INT TERM.map{|l|trap(l){$r.stop}rescue require l};$u=Date;$z=($u.new.year + 145).abs;puts "== Almost Sinatra/No Version has taken the stage on #$z for development with backup from Webrick
$n=Module.new{extend Rack;a,D,S,q=Rack::Builder.new,Object.method(:define_method),/@@ *([^\n]+)\n(((?!@@)[^\n]*\n)*)/m
  %w[get post put delete].map{|m|D.(m){|u,&b|a.map(u){run->(e){[200,{"Content-Type"=>"text/html"},[a.instance_eval(&b)]]}}}}
  Tilt.mappings.map{|k,v|D.(k){|n,*o|$t||=(h=$u._jisx0301("hash, please");File.read(caller[0][/^[^:]+/]).scan(S){|a,b|h[a]=b};h);v[0].new(*o){n=="#{n}"?n:$t[n.to_s]}.render(a,o[0].try(:[],:locals)||{})}}
  %w[set enable disable configure helpers use register].map{|m|D.(m){|*_,&b|b.try :[]}};END{Rack::Handler.get("webrick").run(a,Port:$z){|s|$r=s}}
  %w[params session].map{|m|D.(m){q.send m}};a.use Rack::Session::Cookie;a.use Rack::Lock;D.(:before){|&b|a.use Rack::Config,&b};before{|e|q=Rack::Request.new e;q.params.dup.map{|k,v|params[k.to_sym]=v}}}
```

Code is “this big”

Kill and replace cells regularly

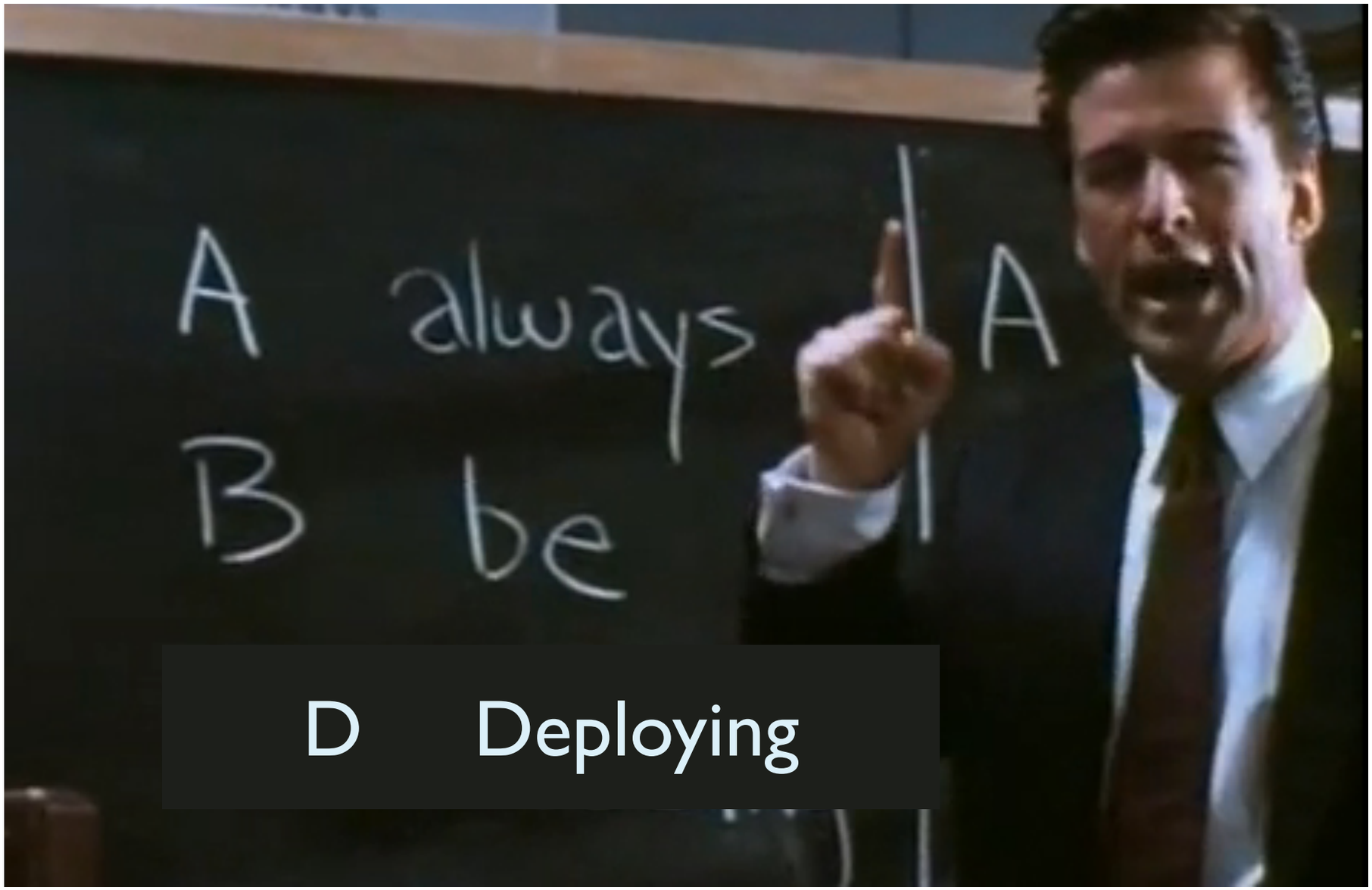
forces you to work with small components

"When a cell is not healthy, an outside
cell that's part of the immune system can command the cell to destroy itself
without spreading toxins."

Nodes are Disposable

Immutable Deployments

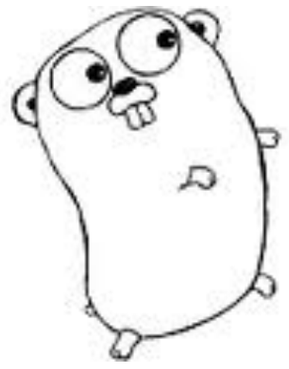
Never Upgrade Software on an Existing Node


A always
A
be
D Deploying

Simple Interfaces

UNIX pipes
Bull RPC

Heterogenous By Default



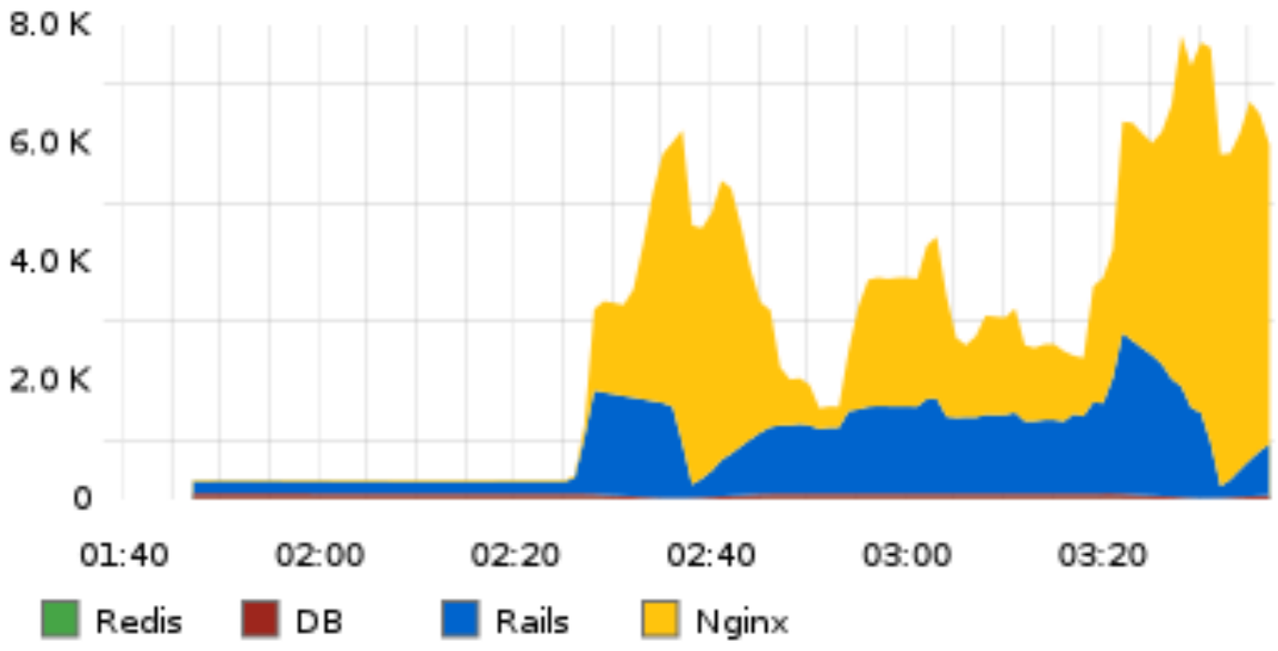
Assume Failure

MTBF vs MTTR

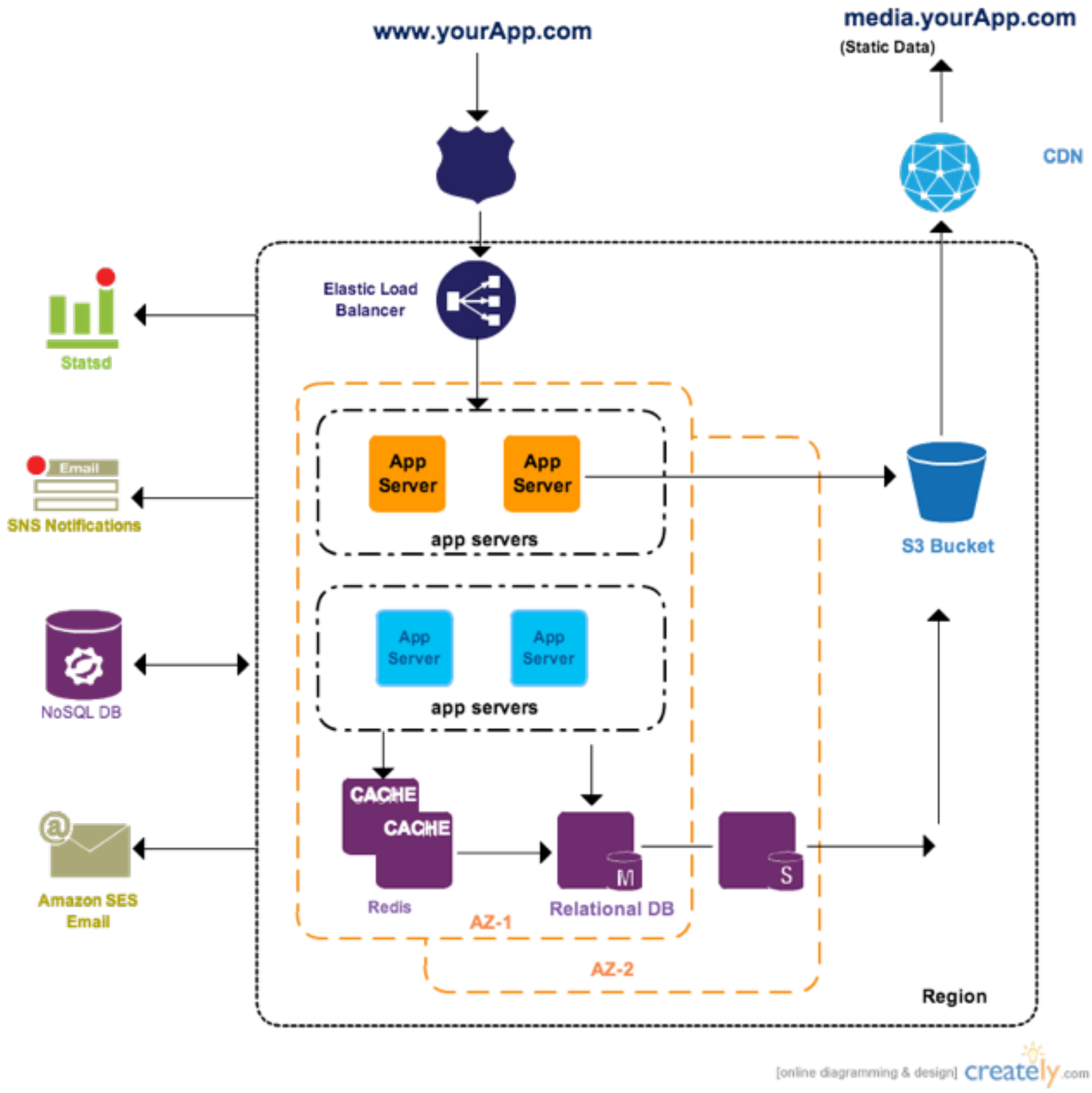
Monitor Everything

Favor measurement over testing

Experience the Worst Case Scenario so You Don't Have to Fear It



Homeostatic Regulation



Homeostasis

Services own and encapsulate data

tiny data

hardware limitations

sorry :(

i don’t know how to do it

Chad Fowler

the passionate programmer, author, speaker, musician, technologist, CTO

2006.12.27

The Big Rewrite

> This is the first in a series of articles, di ing w
> many software rewrite projects end badly a
> to do to avoid some of the ys I've see em go a y.

You've got an existing, successful softw duct 've hit the ceiling on extens bility and maintain Your p latform is inflexible, and your applicatio ftwa ho f cards that can't suppor another new featu

You' seen the videos, the weblog po nd the hype, an ve've decide re to re-implement your pro uct in Rails (or Jav .NET, or ng, etc.).

Bewa This longer, h r, more fail e-prone path than you expect.

ghou career in software evelopment, I've been involved in Rew Big Rewrite. I suspect it's because I have an i st in learning eclecti computer languages, operating systems, and deve ment environ ents. Not being just-a-Java-guy or just-a-Windows-guy h led to me bec ing a serial rewriter. I've been on projects to repl e C, CO L, PHP, Visual Basic, Perl, PLSQL, VBX (don't ask!) and all er of architectural atrocities with the latest and greatest technology of the day.

"By believing passionately in something that does not yet exist, we create it. The nonexistent is whatever we have not sufficiently desired."
- Nikos Kazantzakis