Size metrics

Designing and Maintaining Software (DAMS)

Louis Rose

Habitable Software

Leaner

Less **Complex**

Loosely **Coupled**

More **Cohesive**

Avoids **Duplication**

Clearer

More **Extensible**

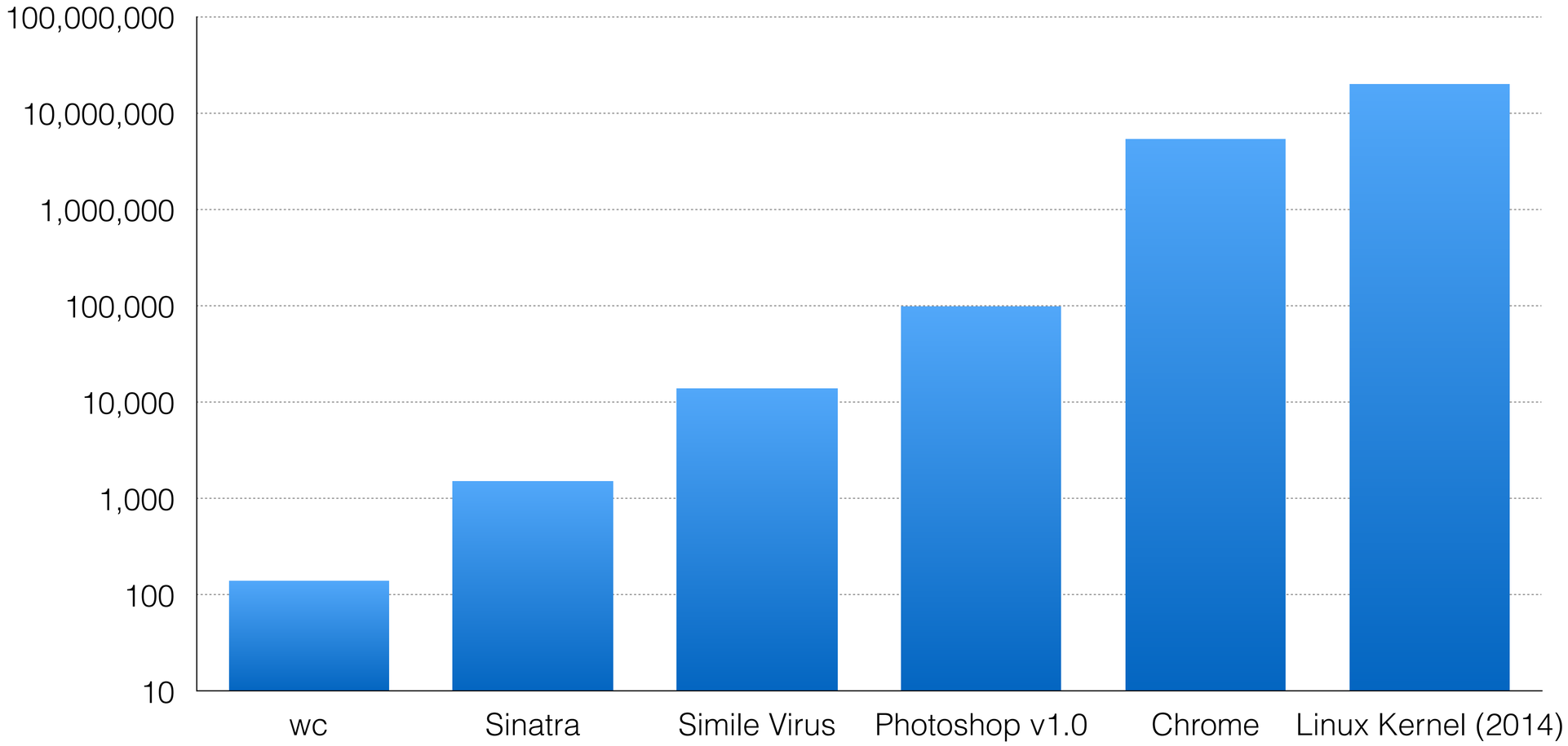
???

Lines of Code (LOC)

LOC:
Total number of (source) lines in a program

Excluding
whitespace
comments
dependencies
tests, build scripts, etc.

Lines of Code (LOC)



Problem: golfing?

```
%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}}}
```

https://github.com/rkh/almost-sinatra

Problem: waffling?

“One of my biggest irritations are studies of productivity based on lines of code… Any good developer knows that they can code the same stuff with huge variations in lines of code. ”

- Martin Fowler

http://martinfowler.com/bliki/CannotMeasureProductivity.html

Solution

Treat metrics as a tool for developers,
not as an instrument for managers.

Intra-Class Metrics

Which are the largest files in our program?

Which classes have the most attributes / methods?

Which classes have unusual attribute:method ratios?

God Class

What is it?
A very large class (relatively speaking)

Why is it problematic?
Often indicates missing abstractions

When does it arise?
👎🏻 “Junk drawer” mentality
👎🏻 Enforced / encouraged by framework
👍🏻 Temporary home

Lazy Class

What is it?
A very small class (relatively speaking)

Why is it problematic?
Every class (strictly, abstraction) incurs overhead

When does it arise?
👎🏻 Speculative abstraction
👎🏻 Downsized during refactoring
👍🏻 Domain objects

Inter-Class Metrics

Which are the longest methods in a class?

Which are the methods with the most parameters?

God Method

What is it?
A very large method (relatively speaking)

Why is it problematic?
Inhibits OO benefits: explanation, sharing, choosing

When does it arise?
👎🏻 Poor grasp of OO programming
👍🏻 ?

Kitchen Sink Method

What is it?
A method that takes a lot of parameters
(relatively speaking)

Why is it problematic?
Increases complexity of using a method
Likely to change often

When does it arise?
👎🏻 Poor separation of concerns
👎🏻 Coddling (doing the method's work for it)
👍🏻 Unpacking objects to break dependency chains

Summary

Use LOCs only to gauge rough size of project

Use (size) metrics as a design tool,
not as management instrument

Size metrics can act as an indicator
for many common issues