the

Mythical

team-month
first things

FIRST
test double

http://test-double.com

@testdouble
this isn’t Management
Small teams go Faster
but claiming that Small teams go Faster requires definitions
but claiming that Small teams go Faster requires definitions
but claiming that active.

**Small teams go Faster** requires definitions.
let’s start with small
scrum says

7

+/-

2
amazon says
small is having nowhere to hide
If I can get away with not focusing not trying not caring
focusing trying caring
the team is not small
defining faster
is faster taking less time to write a line of code?
is faster taking less time to release a feature?
is faster taking less time to earn a dollar?
faster is taking less time to

Answer a Question
questions like,

Is this feature possible?
questions like,

Are we able to build it?
questions like,

Will customers use it?
questions like,

Will customers use it?

Pay For
questions like,

Can it handle 1 million visits each day?
Which design achieves a higher conversion rate?
questions like,

Why does adding a feature take so long?
The answers are feedback telling us where we are.
the answers are feedback suggesting where to go next
Small teams go Faster
let’s talk about

1. projects
2. teams
3. developers
let’s talk about

1. projects
2. teams
3. developers
25% of projects fail
25% of projects fail
ohnoes!
ohnoes!
that’s way too low!
would you wager

75% of projects are

Good Ideas?
plenty of ideas

Ought to Fail
so you may as well
Fail Quickly
but
but we’re conditioned to Avoid Failure
not-made-up headlines

Avoiding Project Failure: It's Not Rocket Science

Avoiding software development failure

Control Risk and Avoid Failure in Organisational Projects

How to Avoid Project Failure Through Project Planning and Effective Project Recovery

Avoid Failure – Facilitate Effective Communications

5 ways to prevent IT failure | ZDNet

Real-Life Project Management Strategies that Fail and How to Prevent Project Failure

To Avoid Failure, First Define Success CIO.com

Avoid these common causes for project failure | TechRepublic
In order to avoid failure, we’ll add more people.
In order to avoid failure, we’ll push back dates
In order to avoid failure, we’ll sacrifice scope.
In order to avoid failure, we’ll limp into production
In order to avoid failure, we’ll move the goalposts
In order to avoid failure, we’ll therefore, succeed!
minimizing failure is a Poor Optimization
what does project Success Mean?
ostensibly, success is Return on Investment
but hardly anyone measures ROI
instead, success is Delivered On Time
instead, success is

Delivered On Scope
instead, success is Delivered On Budget
but those are internal

Arbitrary Metrics
they presume we

Know what we Need
what if such a success were Never Purchased?
what if such a success were

Disliked by Users?
what if such a success were Costly to Maintain?
why not optimize for Faster Feedback?
incentivize projects to Fail Faster
Fail Faster
with fast failure we can

Try More Things
increasing our odds of Finding a Success
let’s talk about

1. projects
2. teams
3. developers
big teams
not all bad
successful big teams
GET THEIR START
as successful small teams
but why?
any new endeavor yields Lots of Questions
to answer those questions

You Need Feedback
a feedback loop

1. get idea
2. implement idea
3. give it to users
4. learn from usage
the faster the loop
the Sooner you Fail
the faster the loop
the Sooner you Succeed
mature, successful endeavors

Raise Fewer Questions
so they can survive with Slower Feedback
but regarding that New Venture of yours...
you should respect Communication Cost
2 people
1 Connection
3 people

3 Connections
4 people

6 Connections
5 people

10 Connections
6 people

15 Connections
7 people
21 Connections
8 people
28 Connections
9 people

36 Connections
10 people

45 Connections
12 people

66 Connections
13 people

78 Connections
16 people
120 Connections
that’s $O(n^2)$ (and it’s often called the handshake problem)
it slows down this loop

1. get idea
2. implement idea
3. give it to users
4. learn from usage
I believe agile exacerbates all this
Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage.

Ron Jeffries
<table>
<thead>
<tr>
<th>xp values</th>
<th>waterfall values</th>
</tr>
</thead>
<tbody>
<tr>
<td>communication</td>
<td>follow the plan</td>
</tr>
<tr>
<td>courage</td>
<td>follow the plan</td>
</tr>
<tr>
<td>feedback</td>
<td>follow the plan</td>
</tr>
<tr>
<td>simplicity</td>
<td>follow the plan</td>
</tr>
</tbody>
</table>
healthy agile teams run on consensus
but the handshake problem suggests consensus doesn't scale
consensus corrects for the team's needs
feedback corrects for the users' needs
sadly, time spent gaining consensus costs you in feedback
because Consensus and Feedback compete for the same Resources
communication + introspection + deliberation + correction
it’s easy to Confuse the Two
how can we build something THIS BIG with a small team?
is exactly the wrong question
how can we build something

This Small

and start failing or succeeding?
if Small Thing™ is wildly successful, then a team can grow organically.
if Small Thing™ is a spectacular failure, then at least it was a small thing
how do you find a Small Thing?
simplify the idea so much

one person can build it
any idea worth its salt can be simplified into one new thing
it took 8 years for the iPod to get an FM tuner
simplify the idea so much

one person can build it
because
great software is possible without a team
all solo acts

- Facebook for iPhone, Joe Hewitt
- DNSimple, Anthony Eden
- Instapaper, Marco Arment
- Delicious Library, Wil Shipley
- Redis, Salvatore Sanfilippo
- The Hit List, Andy Kim
- Alfred, Andrew Pepperrell
- nginx, Igor Sysoev
- Tweetie, Loren Brichter
building a small thing has never been easier
if you require a team
to build something small
you’ve got bigger problems
ideally the idea person and the developer are the same person
because

is faster than
so convince the developer to adopt the idea and then let him or her run with it
the trick is often
Finding a Developer
let’s talk about

1. projects
2. teams
3. developers
Study after study shows that the very best designers produce structures that are faster, smaller, simpler, clearer, and produced with less effort. The differences between the great and the average approach an order of magnitude.

Fred Brooks
well, why didn’t you say so!

just hire one of the great ones!
"I’d like one great developer, please!"
“Awesome, I know HTML and some ActionScript!”
if you can find one,
yay for you!
if you can find one, how are you sure you did?
validating someone is, in fact 10 times better seems hard
starting with just 1 developer can guide the search
that means
genralists > specialists
and
well-rounded
intelligent
look for developers who can succeed independently
frankly,
look for developers who can succeed without you
observable traits of great developers
empathetic

defends users by adopting their perspective
analytical breaks down large problems into smaller ones
visionary identifies a great idea and aggressively simplifies it
scientific

methodically attacks problems, reducing paths of inquiry
creative

dreams up new ideas continuously & asynchronously
professional

invests in long-term value & maintainability of their work
entrepreneurial kills failing projects before over-investing in them
hungry

relentlessly improves through learning, practicing, and sharing
☐ empathetic
☐ analytical
☐ visionary
☐ scientific
☐ creative
☐ professional
☐ entrepreneurial
☐ hungry
non-technical folk can identify these
I’d only entrust my big ideas with developers that embody most of these:

- empathetic
- analytical
- creative
- professional
- hungry

[ ] visionary
[ ] scientific
[ ] entrepreneurial
we talked about

1. projects
2. teams
3. developers
let projects Fail
Communication is Critical
Communication is Critical but it isn’t free
great developers are Willing to wear any hat
test double

twitter: @searls
email: searls@gmail.com
github: https://github.com/searls
blog: http://searls.test-double.com