



Improving life in smaller, heterogeneous projects

LShift @ Oliver Wyman
Via James Uther

About Me

- Phd in CS from University of Sydney
- Software engineer with ~25 experience
- Worked in education & research (USYD), Security (F-Secure), Mobile (Nokia), and consulting (Mobile Innovation, LShift/Oliver Wyman)
- @hemul on twitter
- <https://www.linkedin.com/in/jamesuther/>

- What do I take DevEx to mean?
- What's 'small and heterogeneous' mean?
- Where have I picked up these ideas?
- So, how does one 'Improve life in smaller, heterogeneous projects'?

Developer Experience

"Minimising the distance between a good idea
and production"

-Adrian Trenaman

Developer Experience

Yes, but.

Developer Experience

Questions to ask:

Why would an engineer want to work on this project?

Why would she do good work on it? (? good)

Why can't she work better? (? better)





:no-silver-bullet:

Seriously.

There are better or worse tools, certainly.

And architecture can affect deployment speed and so on.

But at it's core, the experience a developer has is a human one.

small and heterogeneous?

Everyone wants to
build a skyscraper



By Nina - Own work, CC BY 2.5, <https://commons.wikimedia.org/w/index.php?curid=282496>



But it takes a decade
or more.

It's more fun doing lots
of houses



Image taken from 'The Architecture of Glenn Murcutt' and 'Thinking Drawing / Working Drawing' published by TOTO, Japan, 2008. Photos : Anthony Browell



Image taken from 'The Architecture of Glenn Murcutt' and 'Thinking Drawing / Working Drawing' published by TOTO, Japan, 2008. Photos : Anthony Browell



Image taken from 'The Architecture of Glenn Murcutt' and 'Thinking Drawing / Working Drawing' published by TOTO, Japan, 2008. Photos : Anthony Browell

Smaller, Heterogeneous

Consulting. A client wants X done.

Everything is negotiated with a client.

The client is often not technical.

The client is often nervous.

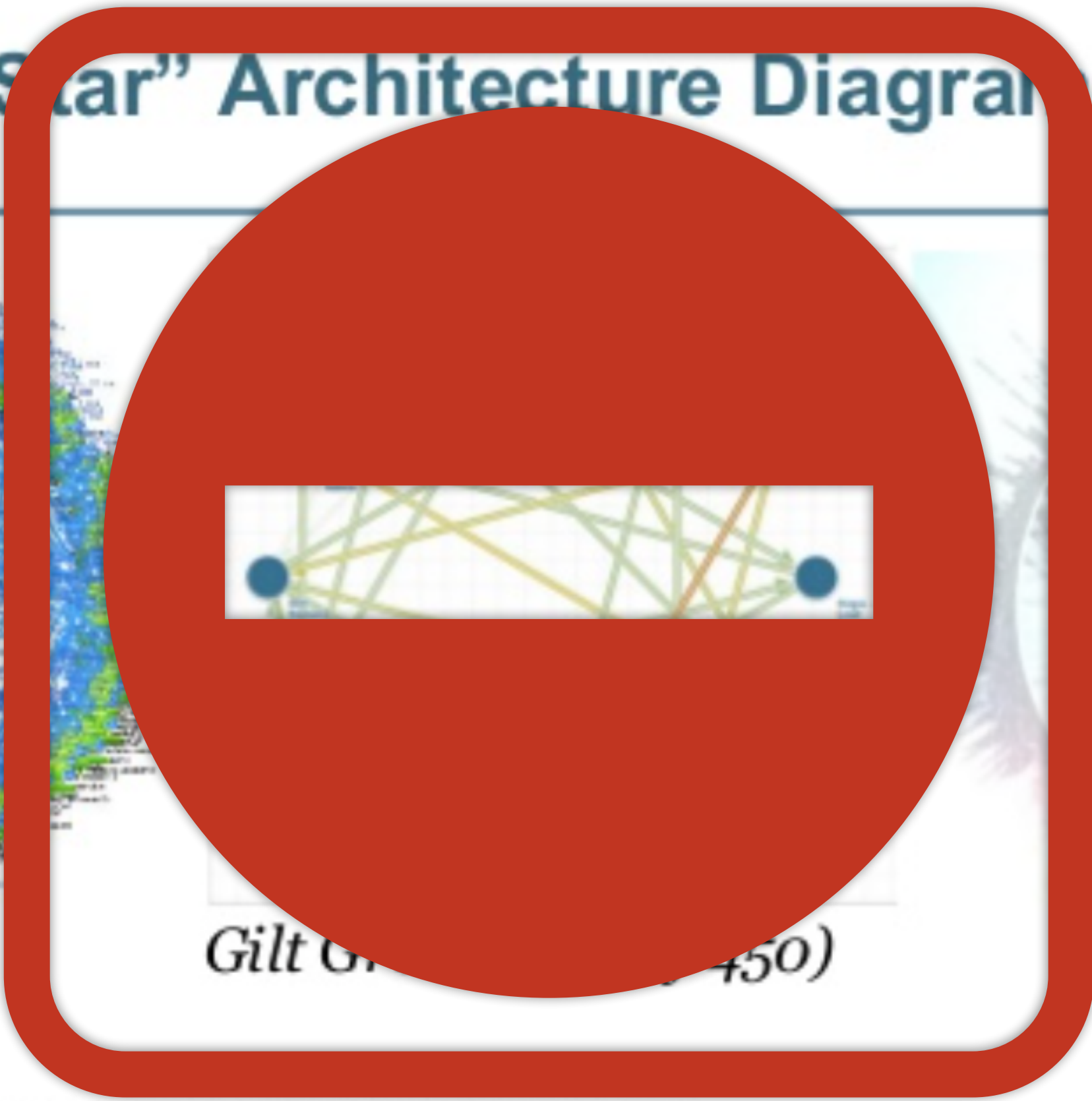
There is usually a back story (legacy)

Projects END.

“Death Star” Architecture Diagrams



Netflix



Gilt Group (2007-2010)



Twitter

As visualized by Appdynamics, Boundary.com and Twitter internal tools

LS Shift

 OLIVER WYMAN



More or less thrived for 18 years

Built RabbitMQ

And some other stuff for <clients>

Engineers sometimes leave for a startup, but often come back

- Organize the world's information and make it universally accessible and useful
- to give people the power to build community and bring the world closer together
- Apple designs [stuff]

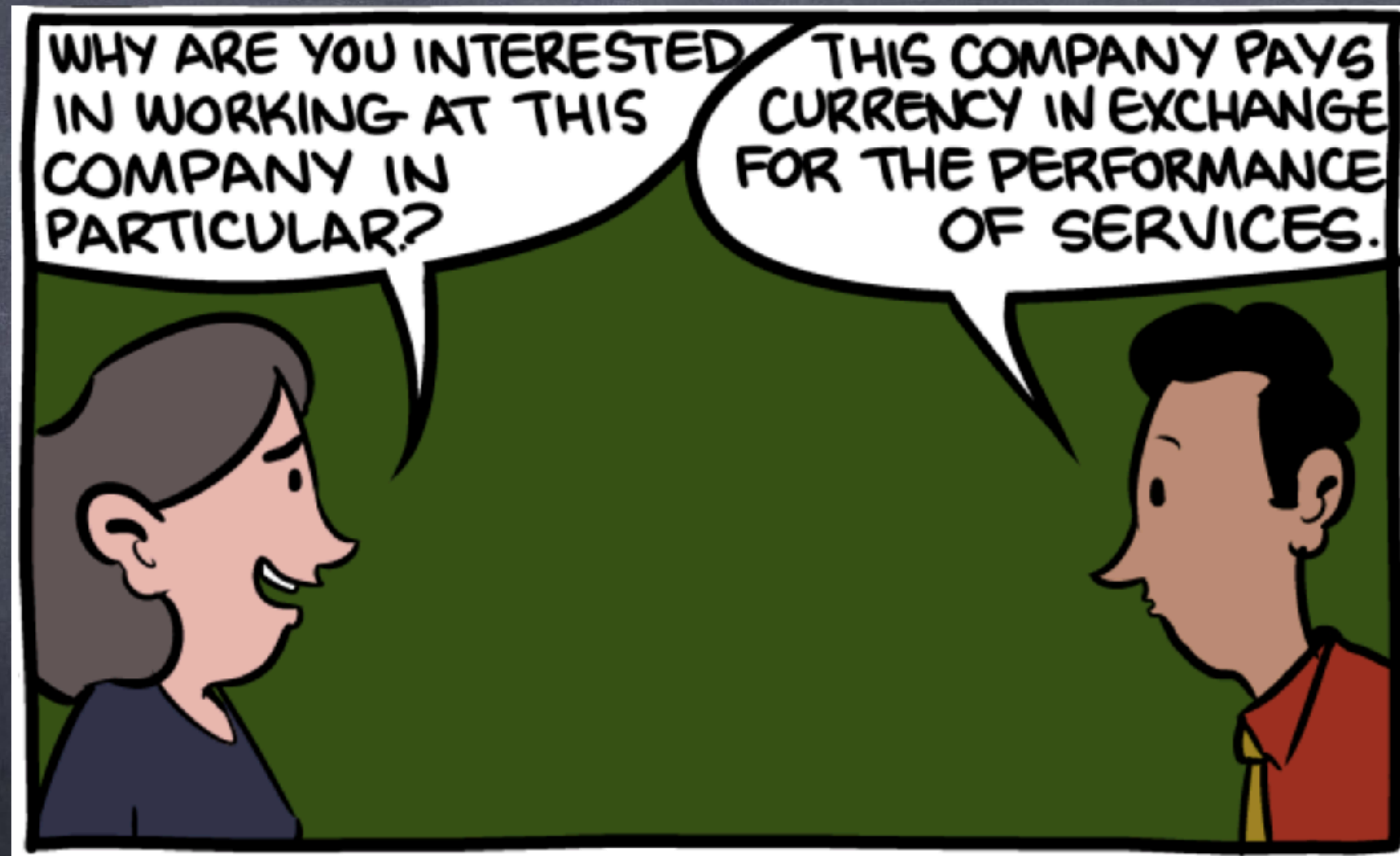
LShift origin story

Founded to employ known good engineers.

"What does the company you would join look like?"

(circa 2000)

Honest interview questions



Learn/do new things = consulting

No budget project managers = do that yourself

This new agile thingy

aka common sense

Empowered Engineers

Hierarchy considered harmful

Make tech decisions

No CTO (currently)

And most project ones

Supporting programme mgrs



This is our foundation of
Developer Experience

Developers empowered to do their
jobs well, and get better

We also have a jukebox
And a kitten video feed
And a coffee machine
And free fruit while it lasts
And are friendly
But these are not the point
(yes, we're always hiring)

Necessaries

Give the client what they need (value)

While being engaged with the work

And actively improving skills

In no particular order, because they
each support the others

Client Value (useful)



Engaged

Improving

How does an
engineer driven
services organisation
do these things?

Client value

It really arises from good engaged engineers
And from a few other things I'll mention here
But they are intermingled

Choose your projects

or at least your attitude to them.

There must be room for a good developer experience somewhere

→ deliver value, while being engaged, while learning.

Even if you're not as enthused as the entrepreneur/partner, what can you learn/contribute/do better?

Make it harder

Look for new things to try even in old domains

-> Exploration: What you learned yesterday is probably obsolete.

-> Deliberate Practice, Growth Mindset: get better by doing hard things.

It is clear that skilled individuals can sometimes experience highly enjoyable states ("flow"[...]) during their performance. These states are, however, incompatible with deliberate practice, in which individuals engage in a (typically planned) training activity aimed at reaching a level just beyond the currently attainable level of performance by engaging in full concentration, analysis after feedback, and repetitions with refinement.

- Anders Ericsson 2007

In a growth mindset, people believe that their most basic abilities can be developed through dedication and hard work—brains and talent are just the starting point. This view creates a love of learning and a resilience that is essential for great accomplishment

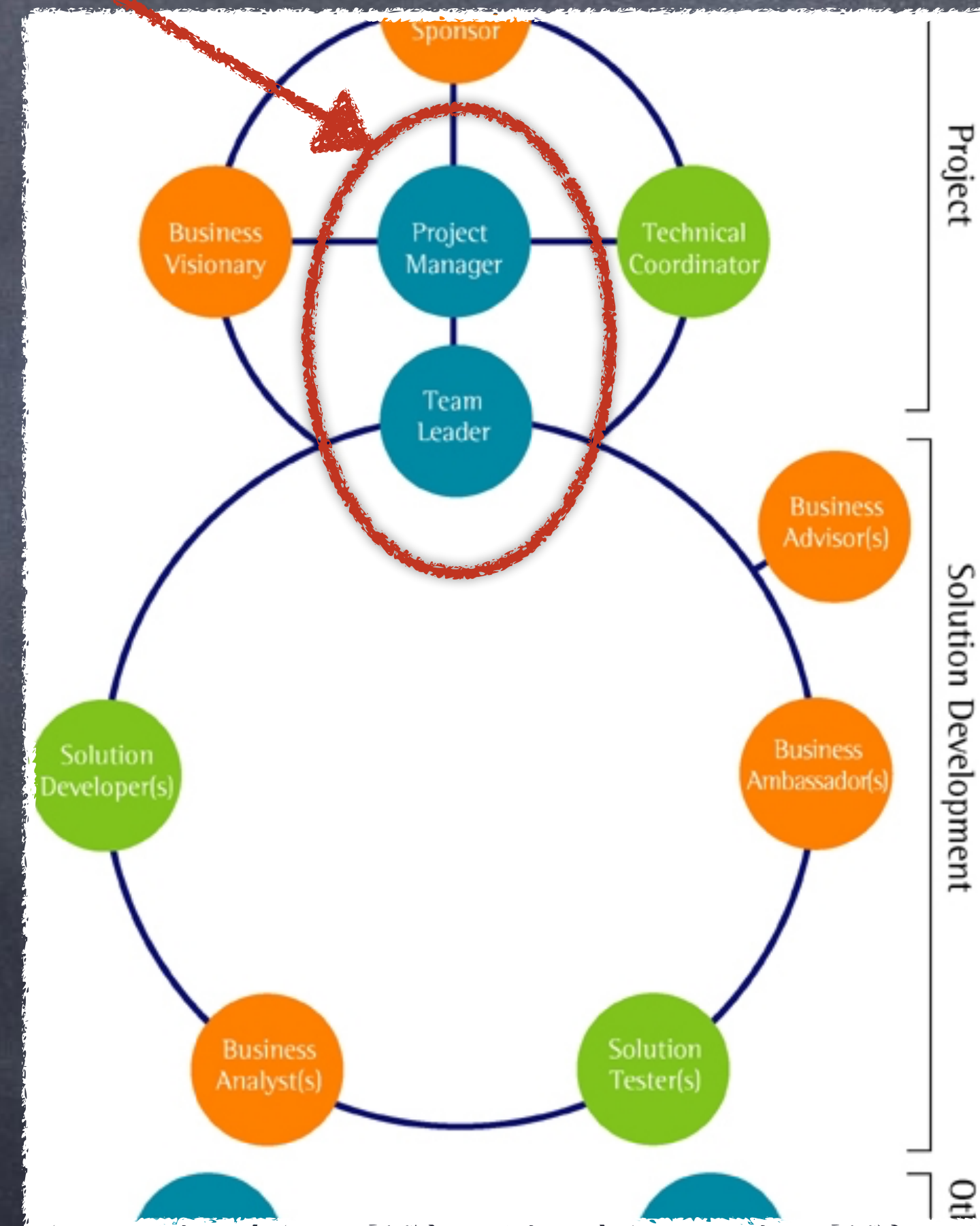
— Carol Dweck

- Take retrospectives seriously. Analyse and improve
- Use that new tool everyone is talking about
- Use that language you've been learning on the side
- Could this project use that fancy algorithm/data structure you've been looking at?
- You know AWS too well. Use another cloud.
- Resolve to blog about something you've learned

Lead Developers

Lead Developers

- Responsible for delivering value to the customer
- Owns the technical direction/delivery
- First among equals in team?



No Silver Bullet —Essence and Accident in Software Engineering

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill

There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.

Therefore it appears that the time has come to address the essential parts of the software task, those concerned with fashioning abstract conceptual structures of great complexity. I suggest:

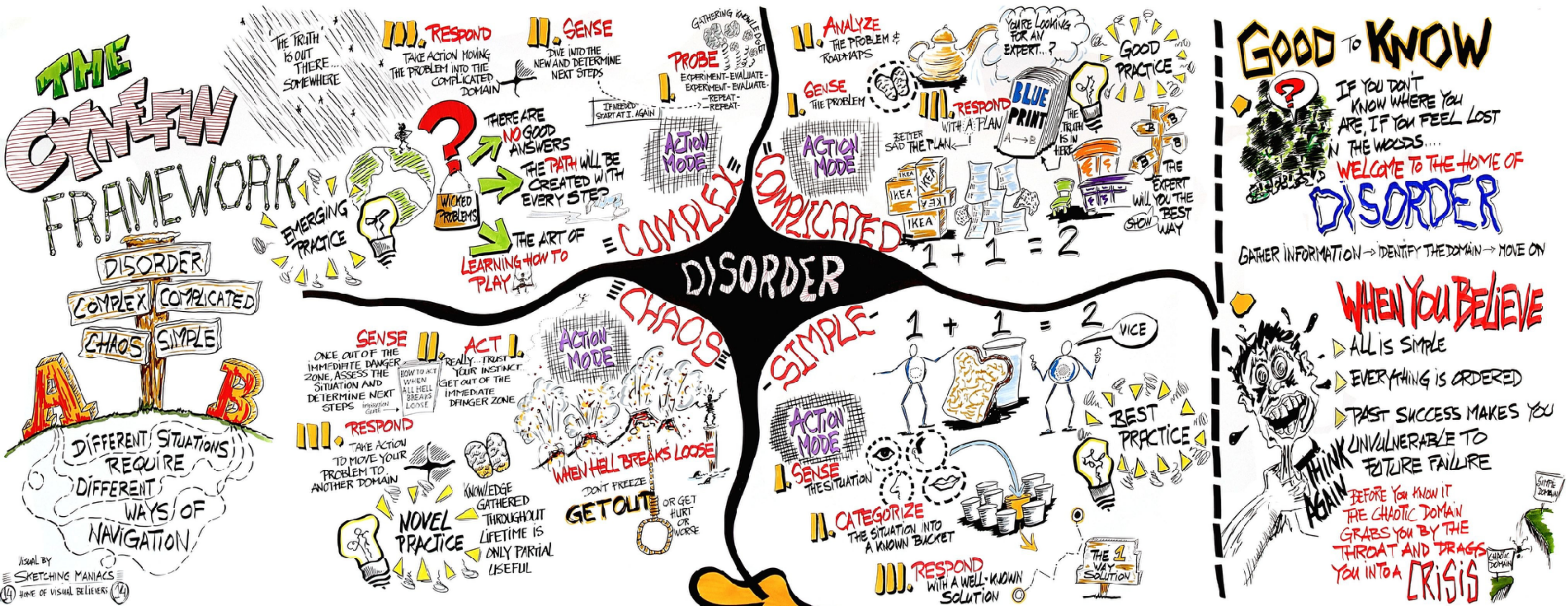
- Exploiting the mass market to avoid constructing what can be bought.
- Using rapid prototyping as part of a planned iteration in establishing software requirements.
- Growing software organically, adding more and more function to systems as they are run, used, and tested.
- Identifying and developing the great conceptual designers of the rising generation.

A little retrospection shows that although many fine, useful software systems have been designed by committees and built by multipart projects, those software systems that have excited passionate fans are those that are the products of one or a few designing minds, great designers. Consider Unix, APL, Pascal, Modula, the Smalltalk interface, even Fortran; and contrast with Cobol, PL/I, Algol, MVS/370, and MS-DOS (fig. 1)

Yes	No
Unix	Cobol
APL	PL/1
Pascal	Algol
Modula	MVS/370
Smalltalk	MS-DOS
Fortran	

Fig. 1 Exciting products

Cymefin - Navigating Complexity



Simple / Obvious

- KNOWN KNOWNS
- There are rules (or best practice)
- The relationship between cause and effect is clear
- "sense-categorise-respond"

Complicated

- Known unknowns
- The relationship between cause and effect requires analysis or expertise
- "sense-analyse-respond"

Complex

- Unknown unknowns
- Cause and effect can only be deduced in retrospect
- There are no right answers.
- "probe-sense-respond"

Chaotic

- ??? WTF?
- Action—any action—is the first and only way to respond appropriately
- Act decisively to move to the Complex domain
- "act-sense-respond"

Disorder

- situations where there is no clarity about which of the other domains apply
- break down the situation into constituent parts and assign each to one of the other four realms

- Projects usually complex.
- Lead developer responsible for taking complexity and packaging it into merely complicated sprints.
- And escaping chaos

Build incrementally,
from firm foundations

Build incrementally,
from firm foundations

In cynefin terms, a project start might be chaos.
Create something to move it to complexity.

Build incrementally, from firm foundations

- 'Walking skeleton'

Start with de-risking and proving
the architecture and delivery path

Local dev experience through to
production



Establish control

Establish control

Agile doesn't guarantee this!

Pick a governance structure.

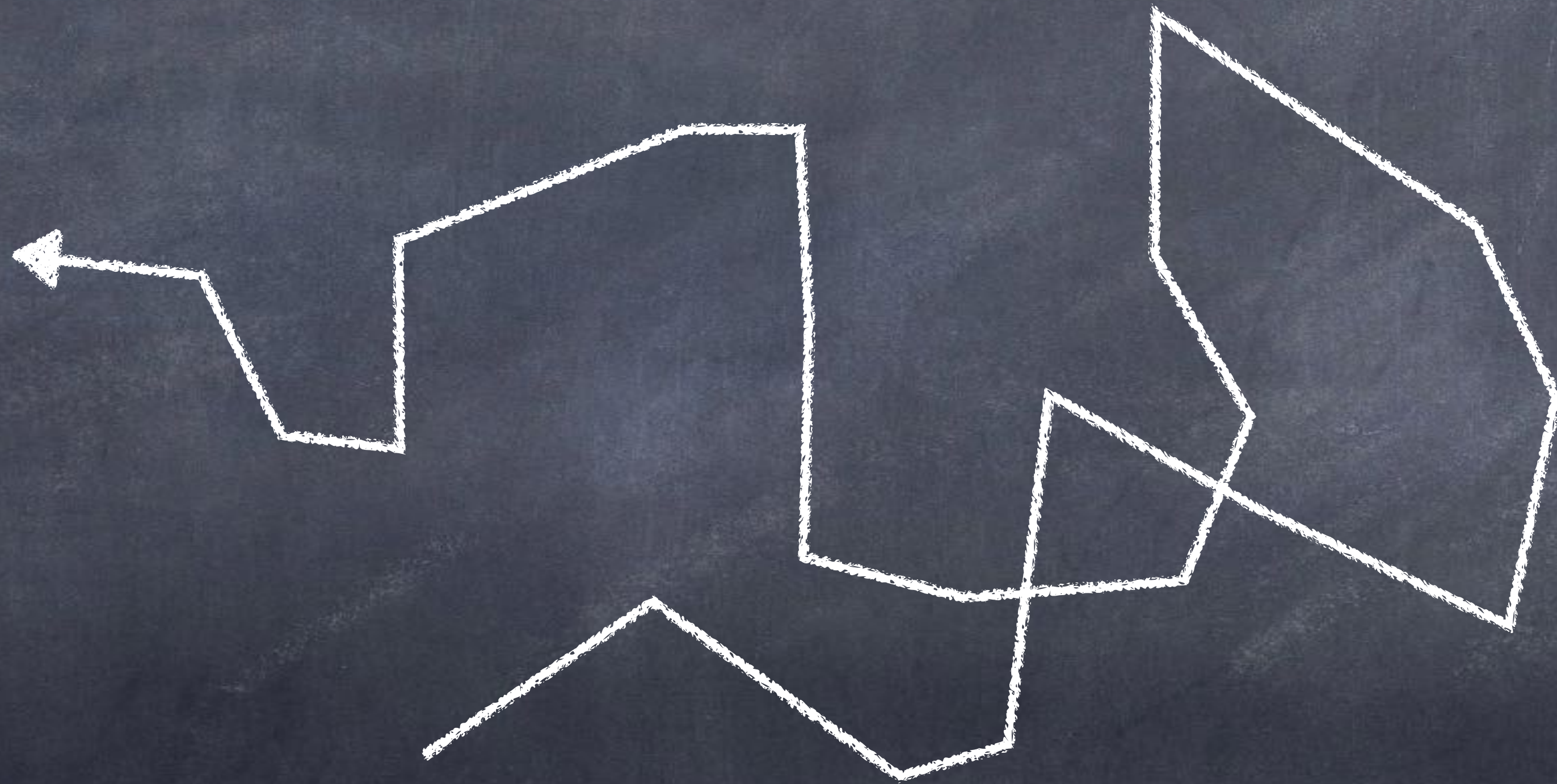
Stick to it!

Demonstrate Control (Principle 8)

An unmanaged project

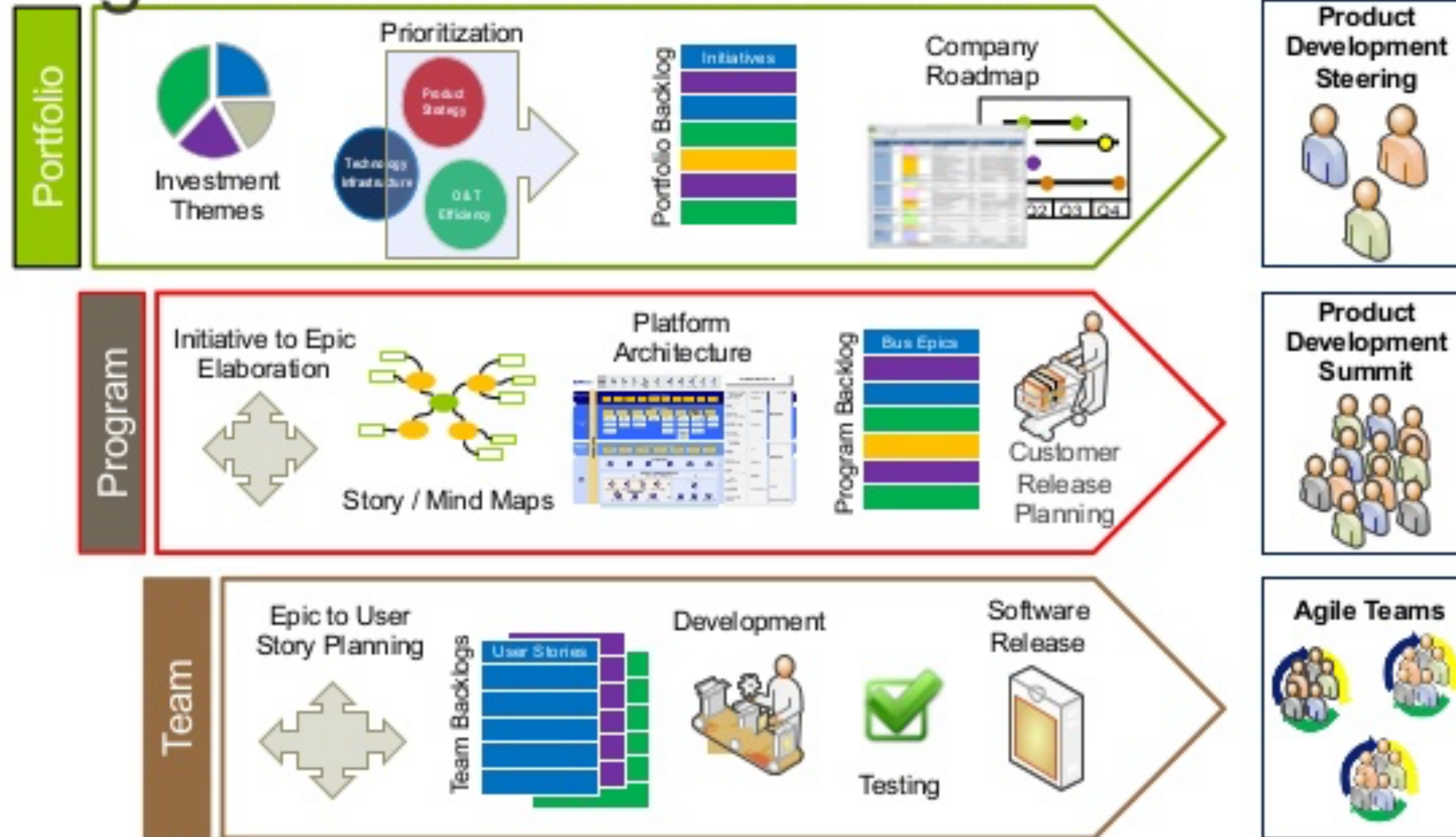


An agile project
(with sprints!)



- Acephalic: having no head or one that is reduced and indistinct, as certain insect larvae.
- Acephalic Agile

Agile Governance Framework





disneyscreencaps.com

Governance is your force field

Foundations

Agree where you are going and how to get there:

- Governance
- Roles
- Reporting - visibility. Over communicate!
- Commercial - breakpoints, deliverables, expectations.
- Bug tracking, CI (tech)

Politics

The activities associated with the governance of a [project], especially the debate between parties having power.

We have a few people who are good at this sort of thing.
We've learned to:

- Take the business context seriously
- Identify who will be in a meeting, and find out/guess their motivations and needs
- Caucus - have a pre-meeting to discuss what we want to do in the meeting



IRA



trac

Integrated SCM & Project Management

Trello



:no-silver-bullet:



Pivotal
Tracker

Wall off legacy
via proxy and
create an
experience bubble.



Quality!

Yes, tests (table stakes)

but also Technical Debt

Clean as you go - it's part of development

If you don't, you will suffer

This is important!

Velocity

This is a reasonable metric to watch. Keep trying to speed up (all else being equal)

To do this, you will have to build a good developer experience

Have you considered 'make'?



This repository

Search


Pull requests


Issues


Mark

 [kelseyhightower](#) / [kubernetes-the-hard-way](#)

 Code

 Issues **12**

 Pull requests **9**

 Projects **0**

 Wiki

Bootstrap Kubernetes the hard way on Google Cloud Platform. No scripts.



This repository

Search

Pull requests

Issues


Marketplace


E

 [kelseyhightower](#) / [nocode](#)

 W

 Code

 Issues **1,644**

 Pull requests **271**

 Projects **0**

 Wiki



The best way to write secure and reliable applications. Write nothing; deploy nowhere.



:no-silver-bullet:

So in conclusion

Small can be more fun

DevEx is about more than the tools,
it's about our experience at work.

There is no silver bullet

Lead bullets include:

So in conclusion

Choose your projects, or how you envision them

Make them hard - Exploration and Deliberate Practice

Single Lead responsible for delivering

Build Incrementally from firm foundations

So in conclusion

Establish and Demonstrate Control - Governance

Build your new experience behind a proxy wall

Maintain Quality, consider your velocity

Do you really need that shiny thing?

