Why Building the Right Thing Means
Building the Thing Right

Liz Keogh
@lunivore
http://lizkeoghs.com
Bain Analysis, 2007

Alignment Trap:
- Highly Aligned: +13
- Less Aligned: -14
- Percentage: 11%

IT-Enabled Growth:
- Highly Effective: +35
- Less Effective: -6
- Percentage: 7%

Maintenance Zone:
- Highly Aligned: 0
- Less Aligned: -2
- Percentage: 74%

Well-Oiled IT:
- Highly Effective: +11
- Less Effective: -15
- Percentage: 8%
The Innovation Cycle

Spoilers

Differentiators

Build on

Commodities
Cynefin

Obvious

Complicated

Complex

Chaotic

With thanks to David Snowden and Cognitive Edge
Cynefin

Disorder

Complacency
Estimating Complexity

5. Nobody has ever done it before
4. Someone outside the org has done it before (probably a competitor)
3. Someone in the company has done it before
2. Someone in the team has done it before
1. We all know how to do it.
Estimating Complexity
If a project has no risks, don’t do it.
“Nobody has ever done this before”
“Bad code isn't technical debt, it's an unhedged call option.”

- Steve Freeman
1000 Chocolate Santas
Deliberate Discovery

Assume ignorance

Assume second order ignorance

Optimize for discovery
Real Options

Options have value.

Options expire.

Never commit early unless you know why.
COMMITMENT

NOVEL ABOUT MANAGING PROJECT RISK

OLAV MAASSEN
CHRIS MATT'S
CHRIS GEARY
Cynefin

Complex

Try something out!
“If in doubt, pick the one that’s easy to change.”

- Chris Matts
<table>
<thead>
<tr>
<th>Discovery</th>
<th>Commitment</th>
<th>Ignorance</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
“I pay a premium now so that I have more choices about where I might take the code later.”

- Steve Freeman
Show some progress

Change direction

Ship some progress

Disrupt the market
Refactoring

Horizon 3

Horizon 2

Horizon 1
Dan North’s “Three Ages”

Explore
Optimise for Discovery

Stabilize
Optimise for Repeatability

Commoditize
Optimise for Efficiency
A Safe-To-Fail Probe has...

A way of knowing it’s succeeding
A way of knowing it’s failing
A way of dampening it
A way of amplifying it
Coherence
What if a meteor hits our data centre?

What if AWS fails?

What if our people don't have the skills?

What if the 3rd party system doesn't work that way?

What if someone gives away the admin password?

What if the release has a bug in it?
A Safe-To-Fail Probe has...

A way of knowing it’s succeeding
A way of knowing it’s failing
A way of dampening it
A way of amplifying it
Coherence

A way of avoiding failure completely
The Fourth Age

Explore
Optimise for
Discovery

Delete
Optimise for
Simplicity
Another refactoring rule

Keep

(Remembering that they move.)

These

Separate!
Liz’s Refactoring Rule

Stop when the next step to make it pretty is obvious.
Liz Keogh
http://lizkeogh.com
@lunivore