Evolving Evolve

The facts and our analysis

12th April 2016
Version 1.1

HSBC

The facts

	2014	2015
Production releases	24	628
Serious incidents	252	323
External customers	- SECRET -	400x 2014

- 100 people working on Evolve
  - 15 business
  - 60 developers in the UK, Poland and China
- Team in China delivering 10+ releases to production a week
  - And working on core business value: metals trading, straight through processing, etc…

Things we didn’t have

- External agile consultancy
- A manual on some brilliant agile process
- Everyone co-located
- Expensive tools
- A tech debt free environment

What is Evolve?

A single dealer platform designed to allow institutional and corporate customers to use the same FX trading platform as our internal sales teams.

- On/Off funding commitment over 4 years leaving significant tech debt and lack of trust with some internal customers
- “The Incident” in August 2014 took Evolve out for 2 market days

How we think we did it

Things you do really need

Tools

- A work ticket system that you can use safely; we use JIRA
- Somewhere to put documents with good version control and safety; we use Confluence
- Version control for your software; we use GIT and the Stash service
- Open source – it's impossible to do much with cost constraints without it

People

- Engaged, onboard, business
- Someone to chase organizational process issues
- Someone who knows what good looks like

Principles

- Have a clear understanding of what is good and bad
- Hire principled people

Discipline

- Don't hack the release process
- Don't break security because it's hard to do well
- Don't not release because it's hard going to production

Just do releases

Making Objectives for teams works

- In January 2015 Raj tells all the teams to release every 2 weeks
- We explain to all the teams about some core lean/agile fast delivery concepts:
  - Use toggles instead of source code branches
  - Non-breaking changes
- Teams managed 2 weekly by March; Raj keeps moving the objective
  - First weekly
  - Then bi-weekly
  - Then daily

Make it easy to release

- Create a system that's high visiblity
- Not too much copying
- Doesn't have to be perfect

Changing the Bank is possible

- In May 2015 we had broken the release process and Change helped us make a new lean release process inside GSD
  - No embargoes – not even Christmas; we did 2 releases on Christmas Eve 2015
  - Only us approving – Business, Dev, Support

Why does releasing increasingly quickly work?

- increasing the number of releases decreases release size
- Decreasing release size reduces risk and complexity…
- … and increases visibility for everyone
- Increased visibility increases the ability for everyone to understand and notice problems
- Increased visibility increases the businesses technical empathy
- Increased technical empathy in the business increases trust around tech debt issues
- increased speed, decreased size means more value earlier to customers

“When you are doing TDD you don’t necessarily type faster or think faster, it’s just that you do less useless things.” - Kent Beck

Other releasing tricks

“My application needs to be deployed at the weekend – that means I only get to do 50 changes a year”

You could start to introduce metrics via Grafana or something, this means you can roll out metrics stuff during the week and understand things at the same time!

“This ticket is really small, it’s not related but I might as well put this in the other release eh?”

Go home, you’re drunk.

Fast release true stories #1

Ms Customer: Hey Lee, I really don't like the frobnicator, it should flange the other way

Lee Butler: Right. I do see what you mean. Tell you what, we'll get that changed for you.

Ms Customer: Ha! Thanks Lee.

A day passes

Lee Butler: Hey Ms Customer, we changed that frobnicator for you, we also polished the flanges. Take a look.

Ms Customer: Great! That's what I wanted… wait… you're a bank! How did you do that?

It’s the businesses business

- IT doesn’t have a great record of delivery
- Trust is low
- Take small steps to building trust
- Be ruthlessly honest with each other
- Too often
  - *“Let’s get together and agree what we’re going to say before we get in front of the business”*
- Deliberately try to build technical empathy in the business
- The business should commit product owners
- Product owners should commit to doing the job well and understanding technology
- The business should help build domain empathy in the technical team
- If your business sponsor is not the main proponent of a change to agile *why not?*

Many of the things you're used to work for some reason

- **A Change Approval Board – we used a weekly one of these to criticize the content of releases and to notice when there were no releases**
  - Crucially we had business presence at this board
  - We were relatively ruthless with each other
  - *"Why aren't you bringing change to the CAB this week?"*
  - *"Why is this change so big?"*
- **GSD – we did not do standard change but one GSD change per release**
  - Because it's more controlled to do that
  - We get to approve all changes
- **Approvers – our change process has *only* 3 approvers – but they're *REAL* ones**
  - No check box approving – if you don't sign off it doesn't happen
  - Biz sign off that this is what they really want, otherwise they don't sign off
  - Dev sign off that this is technically sound and small enough
  - Support sign off that this is supportable
- **When a release is not approved fix the problem and raise another one**

Building trust is hard and you work on it iteratively

- The agile manifesto says: *"We prefer relationships over contracts"*
- Build trustful relationship with the business
- Expect scepticism – IT have promised much and rarely delivered well
- Realize it goes both ways: IT need to trust the biz when they need to change direction; don't just complain about it
- Get facetime, ask to explain what you are trying to do and how you are trying to do it
- Show evidence for how you are working
- Be honest about the evidence
  - We have very little hard evidence about Software Engineering methodologies, we have lots of empirical and anecdotal evidence, see The Leprechauns of Software Engineering by Laurent Bossavit
- Expect to iterate trust, no one is going to trust you straight away
- Agile retros are a great way to improve trust
  - Read the Prime Directive
  - Do a safety check
  - Do a quality check
- Production metrics help to build trust
  - Show people what IS not tests of what it MIGHT be
  - Performance testing is INCREDIBLY expensive or INCREDIBLY useless

The prime directive

“Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand”

less formally -

“If we say things are rubbish we don’t mean people are rubbish”

Retro safety checks

Doing an anonymous safety check in a retro makes people at least consider that it might be a safe place to talk.

Stop beating people up for failures and they will begin to respect the aim to improve.

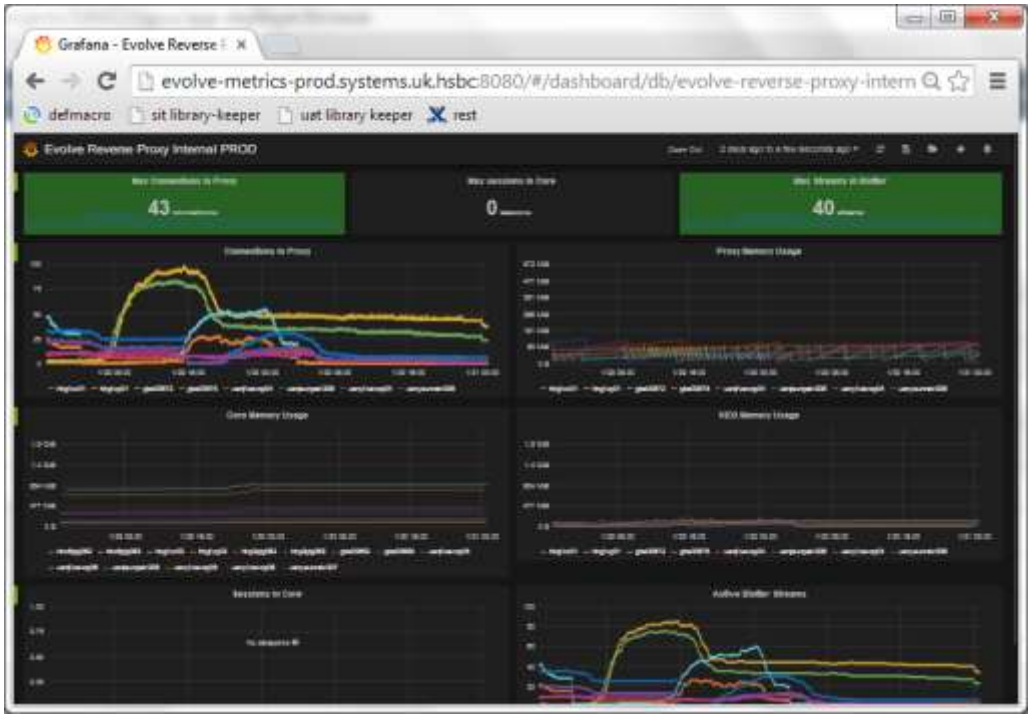
Gets us more towards no-blame retrospectives.

The agile retro quality check

- Ask everyone in a retro to rate your current quality from 1-5
- What's the range of opinion?
- Is there a correlation with the safety check?

Evolve's production metrics

- Influx/Grafana – a very simple open source solution picked by one of our engineers
- Long time storage
  - months is relatively easy
- Simple architecture, easy to hack together
- Get metrics from log files
  - Write processes to tail the log files and send to grafana
  - Or use syslog
- Get metrics from processes
  - Poll the processes JMX
  - Or send from the process
  - Use UDP as much as possible
- We seek feedback from users
  - But we want to find it in logs, not in conversations
- Showing what is actually happening deepens trust between business and IT



Make simple tools

- Tools are over rated as change agents
- Small simple tools can move immature teams a lot
- Substitute super tooling with humans doing some tasks
- Automate in small steps
- Evolve's deploy script
  - Some really bad bash
  - Uses ssh to remote to different boxes
  - Automated targetting multiple machines so reduced engineers logging into multiple boxes
  - It's very messy but not so hard - we have 60 people using it including BA's with no specific UNIX experience
- Evolve's metrics
  - Simple iteration around open source stuff
- Evolve's reporting tools
  - Visibility tools like an email telling everyone about production status
    - We've had 25 releases in the last 3 days
    - We've had 0 incidents
    - What's in queue is…
  - And a version drift report
    - keeping the environments clean and consistent

Making “Off shore” work

- It’s not off shore – it’s a first class part of the whole
- Make sure important business value is added there
  - It’s not just “some testing” or “some ancilliary tools”
- Make sure you invest in coaching to build capability
- Try to visit a lot and give clear direction
- Make everyone talk
  - If you add real business value and you are off-shore you’re going to need to talk to the business every day
  - You also need to talk to fellow delivery people in your other locations
  - Learn to share the best of ideas, processes, tools

Are we doing DevOps?

- DevOps is NOT Lean Six Sigma
- Dev and Support don't work together enough
- Our Support tooling Geneos, is awful
  - It takes about a day to get a new service into Geneos
  - That's twice as long as it takes to deploy the new service into production
  - We struggle to work out a way to make it better

- More granular support teams seems to be the only answer
  - Evolve now has it's own small support team dedicated to it
  - Still huge cultural resistance to change
  - Fear of authority prevents improvements
- Who owns production?
  - In Evolve, too often, it is our Managing Director phoning us to tell us he's seen the platform down
  - Need to get developers and support people as one team capable of owning and improving production

Things we want so we can make it better

- Control of our own boxes
  - Puppet root access to create control and vizibility – this is underway
- To contribute to external Open Source
  - We want to hire developers who are motivated not just by money
  - We want to share our code with the world – there is no security through obscurity
  - We worry about the reputational damage of not contributing back when we use so much
- Better tools provided by group
  - Laptops, screens, web based development tools are awful
  - We need Mac laptops for mobile
  - We need big screens for web development
  - We need Chrome 48, not a crippled internal Chrome
  - We need access to industry standard knowledge bases like http://security.stackexchange.com
- More real authority/responsibility less checkboxing
  - Trust verify works really well but requires real consequences when people don't behave

Are we done?

- At the end of the year Evolve people marked the project out of 10
  - no one gave more than 5
- The people on the Evolve programme want to get better
- Next year Evolve will
  - Do 4000 application releases
  - Have 0 red days
  - Do 500 automated box rebuilds
  - Increase the external customer base by 600%
  - Give our customers a revolutionary mobile experience

We are building a personalised, efficient, beautifully joined up mobile ready experience, making the complex simple to delight customers - Martyn Ranns

Summary

*"It wasn't until we were so ****ing bored of saying we were going to change that we had almost stopped saying it that people started to listen" – Peter Mandelson on changing the Labour Party*

- If you want people to understand something, make it simple and say it over and over again
- Release more often with fewer incidents is a simple, achievable objective with the positive side effect we want