Continuous Delivery in Product Development Teams

Staying relevant in the age of big giants and fast moving start ups
About Me

I am Amir Mohtasebi,
Head of Engineering - Trade Me Jobs,
a.k.a former software developer with a pinch of IT leadership,
In the previous life studied: Failed IT projects, production system, social networks
Twitter: @amirmohtasebi
About Trade Me Jobs

• One of the largest Job boards in New Zealand,
• One of the four main commercial business units in Trade Me,
• Focused mainly on blue-collar Job hunters.

Delivery team history:
• Four years ago: 0 (Purely sales-oriented)
• Today: ~ 30 (Dev, Test, QA, Product, Design)
Product Delivery team (vs. Tech/Engineering)

- Relatively concrete product roadmap
- Driven by commercial results (Revenue, Engagement, Audience, NPS)
- Focused on product problems instead of engineering problems
- Very customer focused
- Requires a translator between product team and delivery team
Context: Lean Product Management

Image Credit: Jeff Patton
Problem: Discovery & Delivery & Operation
Inability to release fast/often/reliably

• Inability to fail fast/often
• Experiments take longer than building the actual product
• Inability to measure and learn from the experiments
• Mediocre product
• Falling behind competition
Continuous Delivery Recipes
Leadership: Senior management buy-in

• Make sure your senior management team understand the importance of Continuous Delivery capabilities
• Dedicate a swim-lane in product roadmap for continuous delivery
Leadership: Senior management buy-in

Image Credit: Lean Enterprise

How?

- Test and deployment automation
- Continuous integration
- Trunk-based development
- Shifting left on security
- Loosely-coupled architecture
- Empowered teams

Continuous Delivery

Non-commercial performance

IT performance

R²=0.288

R²=0.141

Dev pain

Organizational performance

This is how it looks like:
Resetting Team Metrics

• Four Metrics:
  • Cycle time (Lead Time)
  • Frequency of releases
  • Time to Recover from Failure
  • # Bugs

• Two Unofficial Metrics:
  • Team happiness
  • Junior Developers stress during the release
Fish, (Code), and visitors stink after three days

-- Benjamin Franklin (allegedly)
Vision

• Set the clear vision for the team, and frame a bright, dreamy future
• Build your roadmap/change culture to make it a reality in few years

Some examples:
• Developers: Trunk Based Development / One release per day
• QAs: Happily/willingly opt-in to “deploy on green”
• Ops: Low TTFHW
Build Pipeline as part of Production
Lack of Trust: Fitness Functions to the Rescue

Source: ThoughtWorks
Change Architecture

• First responsibility of engineering team is to see if it can be a non-engineering problem:
  • Change the tooling that is not CD friendly
  • High engineering cost

• Microservices Architecture Practices:
  • Circuit Breakers, Bulkheads, Feature Toggles (MTTR)
  • Avoiding Code Sharing (Release Frequency, # Bugs)
  • Micro-frontends (Cycle Time, Release Frequency)

• Evolutionary architecture
Process Change

• Evolution of testing/QA in our company:
  • Dedicated QA should test every piece of code that goes to production
  • Sometimes a dev can test another dev’s code
  • Dev can by default test their own code if it meets certain conditions

• Evolution of “who can release to production?”
  • Infra team
  • Leads
  • Developers

Result: Reduced cycle-time, less burn out, focus on automation, lower number of bugs
## SLI/SLO/Error Budget

<table>
<thead>
<tr>
<th></th>
<th>SLI</th>
<th>SLO</th>
<th>SLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicator</td>
<td>Objective</td>
<td>Agreement</td>
<td></td>
</tr>
<tr>
<td>If this happens...</td>
<td>This amount of time...</td>
<td>Then this is the outcome...</td>
<td></td>
</tr>
<tr>
<td>What’s the metric?</td>
<td>What’s the customer need?</td>
<td>What’s the consequence?</td>
<td></td>
</tr>
</tbody>
</table>

No more Product vs. Engineering vs. Operations
SLO Suggestions

• This is a UX problem - have your product owners go talk to people!
• Trial it with just one service to start with
• Limit your SLOs - three is a good number per service
• Keep your rules as simple as possible - avoid exceptions
• Prefer standard SLO percentages - 90%, 99%, 99.9%, 4 nines etc.
• Alert when your SLOs break
• SLO for existing services?
On-Call Engineer

- Have developers and QAs on an on-call roster
- Increased ownership and understanding of pain points
- Improves code quality as developers see when things go wrong
- Don’t expect people work for free
Alerts + Dashboards

- Use a real-time chat/paging service like Slack/PagerDuty
- No email alerts!
- All alerts SHOULD be actionable
- Keep your alerts simple and ideally good predictors of major issues
- Audit alerts and remove those that aren’t actionable or good predictors
PCR Runbooks

• **Problem**: Something has gone wrong and the on-call engineer doesn’t know what to do.

• **Cause**: The application is big and complex and more than one person is working on it.

• **Resolution**: Have a collection of runbooks in a Problem, Cause, Resolution format - the on-call engineer can follow this to know what to do. Even better, have it linked to from alerts.
Post Mortem + Incident Management

What we’ve learned...

• Don’t panic
• Have a process - write a clear flow chart so everyone understands
• The person who leads an incident should not do the fix(es)
• After the issue is fixed, meet in person to discuss
• Write it up with action points so it doesn’t happen again
• Tooling and automation can help here
• Keep it blameless
Welcome to an incident room! Please ensure that all communications relating to the incident take place here so anyone can join the room and catch up!

For more information on the incident response process check out the Incident Response.

If you have any feedback on the bot or incident process, post it to the #Incident-bot-feedback channel.

The process for resolving an incident is as follows:

1) Establish the details of the incident
   - `/incident sev [1-4]`
     - Set the severity of the incident so people can react accordingly
   - `/incident desc [description]`
     - Set the description of the incident so others know what’s going on
   - `/incident lead [@user]`
     - Show more

---

**Dates**

- **Created:** 19/Jun/18 5:16 PM
- **Updated:** 3 days ago
- **First Occurrence:** 19/Jun/18 1:58 PM
- **Incident Start Time:** 19/Jun/18 1:58 PM
- **Incident Resolved Time:** 19/Jun/18 2:20 PM
Summary

• It is more about people, then process, and then technology
• Start small, big bang fails big
• Create a clear vision
• Reassure team members about their future in the team
• Have senior management buy-in
• Adopt, extend, and localise tools and practices from other companies
Thank You!