Extreme Agile

Things I learned from
managing fully distributed teams

Alan Bennett

My Background

- SW engineer turned manager
  - Consumer, Commercial Aviation and Military
  - Embedded Linux, Native Applications, Web Services
- Been managed by and managed with a variety of development methodologies
  - Waterfall & Iterative (Scrum, Kanban)
- Spent the last 2.5 years At Linaro
  - Dedicated to improving 'Linux on ARM'; directed engineering team
  - Engineering manager for Automation & CI teams
  - Director of Linaro Product teams (LAVA, Stable Kernel, Toolchain)

Managing a remote workforce is becoming very valuable

- 2014: Gartner estimates 54 Million people involved in some sort of remote work
- 2013: US Average Commute time 25 minutes
  - 4.3% work from home (13.5 Million)
- Home-based engineers, digital nomads
- Key Benefit: You aren't restricted to the local talent pool,

https://blog.elucidat.com/mobile-learning-statistics-for-2014-6-trends-you-need-to-know/
http://www.usatoday.com/story/news/nation/2013/03/05/americans-commutes-not-getting-longer/1963409/

Fully Distributed

Fully Distributed

~vs~

Distributed Teams

Questions

- How many people in here work with fully-distributed teams?
- In companies that have remote teams?
- As I go through the rest of slides, I'm curious
  - What are *your* challenges?
  - How do *you* overcome them?

What worked best for us?

(in my opinion)

Tailor Your Process

- Off-the-shelf processes didn't fit
- Waterfall is Waterfall
- Iterative methods are preferred
- Scrum didn't work due to ceremony's & value of co-location
  - Tailoring to meet our needs almost removed the 'scrum' in scrum

Customized Kanban

- Core team creates and manages 'Business Backlog'
  Tech Lead, Product Owner/Manager, Product Manager
- Weekly design meetings 'as-needed'
- Quick stand-ups on days there weren't other meetings
- Team was empowered to add and/or lobby for next tasks
- Extract information from tools to assist with reporting
- Designed with latency in mind

Build Friendships

- Week long Bi-annual company-wide conferences
  - Build strong friendships and trust
  - Learn new technologies
  - Company vision/mission & general joint hacking time
- In the first 3 months at Linaro, I felt I knew my team better than I knew previous teams after years

"timezone close"

- Proactively fight isolation
- though fully distributed works, try to have some team members within a timezone or two
- Create reasons to get engineers that are "timezone close" to work together
  - training new engineers, 'pair' programming

Cross-team Projects

- Initiatives are great to bring people together
- Find ways to bring people together across projects
- It's more exciting when it 'matters'
  - kernelci.org, though it started as an initiative, soon I realized I couldn't stop the team working on it if we wanted to

Tools & Systems Matter

- fully collaborative tools
    - You won't realize the power until you don't have them
    - Google Apps (Hangouts, Calendar, Docs, Sheets, etc)
- Chat platforms - IRC with server and/or Slack

Tools, Cont.

- Development tools
  - git, Gerrit, Jenkins, LAVA
- Project management tools
  - None are perfect, but if you can, find a simple gui that has command line interface-ability

Development Practices

- Establish a Sound development process that the team uses, compare that to information you & company need
    - Often times, just write it down
- Automate as much as you can to protect your developers
    - Find a few engineers to champion testing
- Identify Key metrics

Experience/Hiring

- When hiring remote workers, you should only hire experts?
- You should only hire people with previous remote experience?
- Actually, I strongly believe previous 'remote' experience was not a key indicator of success
- A key in any business is the ability to attract and find the right people

The one thing that worked better than anything else

- Past open-source contributions
  - proven ability to work with critical code reviews
  - ability to work with experts to extend software
  - Pride; Proud of their code and they are open to criticism
  - Naturally autonomous, self-starters
- New engineers seemed more willing to being open than those trained to keep things closed

Location

- Unfortunately there is a trick to open-source engineers, they don't 'currently' grow where 'you' want them to grow
- They are in demand
- May or may not be open to relocation
- If you want to hire them you may need to get good at managing 'fully' distributed teams

Thoughts
Questions
Discussion