Introductions & law of 2 feet.

Very Large, Highly Visible Project with the US Treasury
Been working with them for about 1.5 years, so this is a very quick overview.
PEOPLE PROBLEMS

- Command & Control Leadership
- Low Team Morale
- Siloed Skill Sets
- Cranky Sysadmins
- Poor Hiring Practices

THE PROBLEM WITH SOME PEOPLE...

IS THAT THEY EXIST
we'll tackle process first, because it's the easier of the two.
PROCESS PROBLEM #1
LONG RELEASE CYCLES
• 6 months to get a normal, low-risk change into production due to gates & manual testing, but...

Critical & high impact change released to production within a couple days bypassing all gates and testing because “we needed it”.
diagram of how the deployment pipeline works.

How does Change management fit in here…

From check in to production in ~ 2 hrs
 SETUP FOR CONTINUOUS DELIVERY

- Team Co-location
- Small Features
- Three Amigos (ATDD)
- Continuous Integration
- Pushbutton Deployments
- Automate Everything

Going to be referring to Acceptance Test Driven Development a few more times, so let's describe it.
Red
Refactor
Green
TDD Cycle
WHERE DO I FIT IN?

RED

REFACCTOR

GREEN

TDD CYCLE
Developers, Testers and Business Owners sit together to write Acceptance Tests.
PROCESS PROBLEM #2

SLOW HARDWARE PROCUREMENT
BEFORE

1. Write business case - 1 day
2. Wait for approval - 1 week
3. Order Server - 1 day
4. Wait for delivery - 1 week
5. Install Server in Rack - 1 week
6. Install Software - 2-3 weeks

Total: 6+ weeks

Also, required to always pay for your maximum expected capacity.
AFTER

1. Decide I need a server - 5 minutes
2. Log in to AWS and create server - 5 minutes
3. Wait for server to start (5 minutes)
4. Run Chef Cookbooks (5 minutes - 2 hours)

Total: < 3 hours

Only pay for what you need. Shut down servers when you don’t need them and you don’t pay for them. Obtain more capacity on demand.
STILL NEEDS WORK

• Production deployments still occurring in internal data center...

• So new production servers still have a long lead time
• Become familiar with the process
• Quickly submit any new desired technologies
• If it meets a business need, use it.
• Never deselect technology because it “isn’t on the list”
PROCESS PROBLEM #4
SECURITY REVIEWS
**BEFORE**

- Code:
  - Team level code reviews
  - Pre-deployment scans
- DB:
  - Manual updates by DBAs
- Server
  - Manual updates by SAs

**Code:** delayed scanning coupled with long release cycles, results in long feedback cycle to address security reviews.

**DB:** Heard many times that DBA’s need to run updates because of security.
  - Leads to inconsistencies between environments
  - No auditability

**Server:** Same issues
AFTER

- Code:
  - Paired Programming
  - Scans integrated into CI
- DB:
  - Versioning & Automation (Flyway)
- Server
  - Versioning & Automation (Chef)
STILL NEEDS WORK

• Paired Programming not occurring 100% of time

• Database replication causing some issues with automation

• Internal Data Center has some manual work to create servers
PROCESS PROBLEM #5

CHANGE MANAGEMENT
• Address root regulations rather than accept standard practices

• Build process around best & most lightweight tools possible

These days, the best tools are most often free.
PEOPLE PROBLEMS

- Command & Control Leadership
- Low Team Morale
- Siloed Skill Sets
- Cranky Sysadmins
- Poor Hiring Practices

THE PROBLEM WITH SOME PEOPLE...

IS THAT THEY EXIST
PEOPLE PROBLEM #1
COMMAND & CONTROL LEADERSHIP
- **Problem:** Repeated disappointments lead to lack of trust in teams to deliver.

- **Solution:** Demos every 2-3 weeks showing running, tested, features. No Mockups!
• **Problem:** Desire to build “ilities” framework - security, configurability, scalability

• **Solution:** Define specific examples of these in the context of a feature using the INVEST mnemonic
INVEST

- **I - Independent:** is self contained
- **N - Negotiable:** can be changed until started
- **V - Valuable:** for the end user
- **E - Estimable:** have some idea of how big it is
- **S - Small:** no more than 3 days development effort
- **T - Testable:** must know whether it works. (automate!)
NEED NEW EXAMPLE

Given a user with a valid account

When she navigates to any page in the system she has access to

Then she should see the page within 1 second

Bad Example

I - Independent: is self contained
N - Negotiable: can be changed until started
V - Valuable: for the end user
E - Estimable: have some idea of how big it is
S - Small: no more than 3 days development effort
T - Testable: must know whether it works. (automate!)
Given a user with a valid administrator account
When she navigates to the update balance page
Then she should see the page within 1 second

Good Example
I - Independent: is self contained
N - Negotiable: can be changed until started
V - Valuable: for the end user
E - Estimable: have some idea of how big it is
S - Small: no more than 3 days development effort
T - Testable: must know whether it works. (automate!)
• **Problem:** Desire to develop a process that can be dropped in to other teams

• **Solution:** Continuous education about the core principles of agile
  
  • [http://j.mp/nayan-agile-principles](http://j.mp/nayan-agile-principles)
  
  • [https://speakerdeck.com/nhajratw/agile-from-principles-to-practice](https://speakerdeck.com/nhajratw/agile-from-principles-to-practice)
It's about the journey
takes a 12 months to start appreciating/understanding why it works & how to improve.
• **Problem:** Desire to have a fixed scope, fixed date solution

• **Solution:**
  - Never compromise on quality
  - So what has to give?

Except for the most transient of projects, compromising on quality leads to slower change, increased maintenance cost, and unhappy customers.
FIXED DATE

- Date was arbitrary - based on vendor contract period, not business need.
FIXED SCOPE

- Estimation Failure
- WBS does not work for projects with high variability.
- https://www.youtube.com/watch?v=GiPe1QiKQuk&t=6
- > 2 weeks does not work

WBS does not account for unexpected events
All tasks are not knowable.
Attempting to removing variability from software projects also removes innovation.
Fixed scope is not as fixed as you think it is.

“We need feature X”
FEATURE SPLITTING

“Which of these smaller features do we need?”

High level features often contain a lot of waste.
Once did an analysis of a live production system and found that only 30% of the code was actually ever invoked. The rest was feature bloat.

Never seen a situation where scope could not be cut.

Determine the first few features that provide the highest value and build them, end-to-end, including automated tests, then move onto the next.
PEOPLE PROBLEM #2

LOW TEAM MORALE
Get your team what they need!

$4K for a high end setup that lasts 3 years or so. For a dev making 90K/yr, that's less than 2% of their salary.

Macs, VideoConferencing, AWS,
They don't have to cost money: Git, Text Editors, Chat.
• Involve them in decisions about WHAT needs to be built
• Ensure they know WHY it needs to be built
• Empower them to decide HOW to build it.

Let team pick technologies, not dictated by Architecture Board. Trust them not to be reckless. If you can’t trust them, you’ve hired the wrong people.
PEOPLE PROBLEM #3

SILOED SKILL SETS
• Pairing different roles
• ATDD - Three Amigos again
• Ran into problems with organizations outside sphere of influence

Which leads us to...
PEOPLE PROBLEM #4
CRANKY SYSADMINS
• Attempted to alleviate work by cross-training others so backups have sufficient capability

• Root cause of this is the organization’s focus on maximizing utilization

• Sysadmins are overworked.
• Cross-training is seen as waste. 2 people doing the job of 1.
• Limited influence in this area.
Systems like this behave in an exponential curve like this. When people have time to think, they become more creative, can easily help others when needed, and be nimble.
PEOPLE PROBLEM #5

HIRING PRACTICES
• “We Want Top Talent”
• “We pay Market Rates”
• Limited influence in this area so far

You pay for average, you get average.
AGILE & THE VERY LARGE ORGANIZATION: A PERFECT MATCH