Rebooting the Team

Lessons Learned in an Introspective Year

Peter Clark
pclark@umn.edu
@pclark

Fran Fabrizio
fran@umn.edu
@franfabrizio
In the next 39 minutes, you’ll hear about:

• Intro
• Characterizing what’s dragging on your dev team
• Getting to root causes
• Doing something about it
• Lessons we’ve learned
• Q&A
The Setup - Who We Are

- Pete - the former IT Director
- Fran - the current IT Director
- Minnesota Population Center mission and environment
- Who are you?
What’s a reboot?

• A conscious decision to engage in deeper, more radical change than just incremental improvements.

• Typically impacts staff structure, work processes and communication patterns for your team and organization.
Why Do We Need Reboots?

- Engineering teams are good at solving engineering problems.
- Wetware (people/process/communications) problems don’t lend themselves to engineering-style solutions.
“Happy families are all alike, every unhappy family is unhappy in its own way.”

-- Leo Tolstoy
Today’s Goals

• Share processes and activities we’ve used during our reboots to try to make our team better, and a little wisdom we picked up along the way.

• Takeaway: Techniques and tools to help you figure out what’s ailing your team, distill symptoms of team dysfunction down to actionable root causes, and collaboratively enact lasting change.
Today’s Anti-goals

• NOT a takeaway: A silver bullet for fixing your team. You will have homework! The best we can offer is some design patterns.

• NOT about assigning blame, it's about problem solving! Blame is an activity focused on the past. Problem solving is an activity focused on the present and future.
while employed? do
  observe symptoms and smells
  find root causes
  design and execute actions to address
  review outcomes
end
Abstract

Concrete

Symptoms  Smells  Root Causes  Actions

Diagnosis

Treatment
What’s that smell?

<img src="diapers.jpg" alt="stinky!" />
People are quitting!

Nothing ever ships

Every decision is made by committee

People don’t have the right tools to do their jobs

Setting up a dev environment takes 3 days

Small changes take longer than they should

Each project is doing things its own way

Can’t say no to anything

You’re on a death march, and everyone knows it.

Staff members reluctant to share knowledge

Expected to do 5 things at once

Build is always broken!

Projects getting later and later

You get a pit in your stomach when you walk in the door

People don’t take pride in their work

Estimates are fictional

No consensus on priorities

Sick time is spiking

Staff with similar experience not collaborating

Estimates are unreliable

You’ve been committed to things without your knowledge

No 1:1 meetings

Every deployment results in a big mess

Status meetings turn into bitchfests

New requirements appear late in projects, when they should have been known from the start

Nobody knows what ‘done’ means

People are socializing

Secondary projects fall through the cracks

People aren’t socializing

Documentation is wildly out of date

People don’t trust IT, with good reason

It’s too quiet!

Customers don’t trust IT, with good reason

Build is always broken!

No one person knows how to do a full deployment

Estimates are unreliable

Documentation is wildly out of date

No 1:1 meetings

Every deployment results in a big mess

New requirements appear late in projects, when they should have been known from the start

You’ve been committed to things without your knowledge

Nobody knows what ‘done’ means

Secondary projects fall through the cracks

Small changes take longer than they should

Setting up a dev environment takes 3 days

Can’t say no to anything

You’re on a death march, and everyone knows it.

Projects getting later and later
Bad Office Vibe!

People are quitting!
Nothing ever ships
No one person knows how to do a full deployment
Customers don’t trust IT, with good reason
Estimates are unreliable
Documentation is wildly out of date
No 1:1 meetings
Every deployment results in a big mess
New requirements appear late in projects, when they should have been known from the start
You’ve been committed to things without your knowledge

Routine Things Are Complicated

Build is always broken!
Projects getting later and later
You get a pit in your stomach when you walk in the door
People don’t take pride in their work
Estimates are fictional
No consensus on priorities
Turn into bitchfests

It’s too quiet!
Sick time is spiking
Staff with similar experience not collaborating
Estimates are fictional

Something’s Always on Fire

Secondary projects fall through the cracks
Changes take longer than they should
We don’t have the right tools to do the job
Setting up a dev environment takes 3 days

You’re on a death march, and everyone knows it.
People don’t have the right tools to do their jobs
Every decision is made by committee

People aren’t socializing

You can’t say no to anything

Customers don’t trust IT, with good reason
Estimates are unreliable

People don’t take pride in their work

Build is always broken!

Projects getting later and later
You get a pit in your stomach when you walk in the door
People don’t take pride in their work
Estimates are fictional
No consensus on priorities
Turn into bitchfests

Something’s Always on Fire

Secondary projects fall through the cracks
Changes take longer than they should
We don’t have the right tools to do the job
Setting up a dev environment takes 3 days

You’re on a death march, and everyone knows it.
People don’t have the right tools to do their jobs
Every decision is made by committee
• Code smells: Patterns in source code which hint at a deeper flaws. Usually not bugs themselves, but characteristics that indicate weaknesses in code design.

• Org smells: Patterns in a team or organization which hint at deeper flaws. Usually not catastrophic themselves, but indicative of weakness in team communication, process, or architecture.
Smells that we found

Unhappy team members
  Single points of failure abound

Bad office vibe

Leadership is elsewhere

Every day is a surprise party

Customers don’t trust the developers

Overpromising/Underdelivering

Something’s always on fire

Bad operational hygiene

The team is too silo’ed

Routine things are complicated
Getting to the Root of Things

$ sudo why
The Cubicle Story
Peeling off the Layers

• “My cube sucks” really meant “I’m bummed out because management doesn’t seem to be listening to or valuing my needs.”

• Exposed a deeper problem: communication disconnect between development and management. We need to treat this problem, not the “my cube sucks” symptom.
Your Mission

Something smells wrong. You don’t know why yet, but you’re the one who is expected to fix it.

How do you get to the why? Can’t do it alone. You need...
Organizational Debugging

Winning this mission requires *organizational debugging*.

1. Get every stakeholder's point of view
2. Keep digging until you converge on root causes
3. Create and execute action plans for each root cause

When organizational debugging is done collaboratively and with Humility, Respect and Trust (HRT), it *produces momentum for change*. 
The Big Secret!

When you dig into a wetware problem, you’re almost always going to find at least one of these things:

COMMUNICATION  TRUST  PROCESS

But “We suck at communicating” is not actionable.
Get to more specific root causes.
Engaging the Stakeholders

To get the insight you need, you must harvest it from your stakeholders.

Each group must be engaged in a way that’s meaningful to them. Tailor your approach, content and detail to each group.

**Meet them on their turf!**

A sampling of ideas about how to do that...
Engaging the Development Team

• Create dedicated time/space to foster group dialogue about the org smells
  – Focus Friday (or forums/chat rooms)
  – “Squishy” book discussion. Example: Team Geek

• Meet privately with individuals regularly, as well
Engage Management

• Provide management with visibility to the smells and symptoms
• Gain buy-in to the reboot by incorporating their top-down perspective
• Example Activities:
  – Roundtables with leadership group - ask “macro” questions
  – Vision & perception interview with head of organization
  – Use concise, powerful tools. Example: Projects roadmap
Engaging the Customer

- Roundtables with our customer teams
- Postmortems after each major release
- Example: Mapping the communication flow
Example of a Communications Map
Converging on Root Causes

Ex: Why do we think that “Something’s Always on Fire”?

Dev Team Insight: “We get a lot of requests out of the blue that pull us away from our core projects to work on ad hoc things or fringe projects.”

Management Insight: “We want to help campus folks who are doing demographic research, but we don’t have a process for vetting requests.”

Customer Insight: “The MPC agreed to do my web site, but when I call over there it seems that there’s always something more pressing.”

Suggests root cause might be Lack of Focus on Mission
Our root causes in 2012

- Underinvestment in Individuals
- Lack of Focus on Mission
- Lack of Team Cohesiveness
- Operational Deficits
- Lack of Customer-Developer Transparency
Getting to Solutions
(Or, putting the “fun” back in dysfunctional)

• For each root cause, design an action plan that mixes easy wins with longer-term fixes

• This part needs to be collaborative too! Get stakeholder buy-in and help for designing the action plans.

Quick sampling of our 2012 Reboot action plans...
Solutions: Underinvestment in Individuals

• Early Wins:
  – Do a job study across the dev team to clarify roles
  – Leverage org strengths (e.g. robust prof dev support)

• Long-term Work:
  – Address harder-to-solve recruitment and retention issues
Solutions: Lack of Focus on Mission

• Early Wins:
  – Make all hidden work visible to management.
  – Distraction Management: Prioritize work and outsource or kill fringe projects and other distractions. Align effort to the core mission!

• Long-term Work
  – Multi-year strategic plan (for the entire org)
Solutions: Lack of Team Cohesiveness

• Early Wins:
  – HipChat: persistent group chat tool
  – Monthly Thinking Thursdays: group lunch then gathering

• Long-term Work:
  – Cross-project standardization (e.g. tiger teams)
Solutions: Operational Deficits

• Early Wins:
  – Evolve tools: SVN → Git, TeamCity → Jenkins, Chef, Vagrant
  – Management support for our version of 20% time

• Long-term Work:
  – Ongoing technical debt reduction
Solutions: Lack of Customer-Developer Transparency

• Early Wins:
  – Formalize a quarterly planning process
  – Open work tracking tool to customers

• Long-term Work:
  – Refactoring the communications model
New Communications Model

Messier, but more effective!
Measuring Outcomes

• Probably the most important one is qualitative:
  “Hey, things smell better now!” - the Stakeholders

• Quantitative indications can be helpful too, but will come later

Each organization measures success differently. Focus on outcomes that have meaning to your org’s culture.
Not all solutions work out as planned

• When solutions just don't work as hoped
  – The symptoms don't go away, review your original symptom list periodically

• When they fix a problem but create another one
  – New symptoms appear

Be prepared to iterate!
while employed? do
  observe symptoms and smells
  find root causes
  design and execute actions to address
  review outcomes
end
Where are we now?

• We feel good about...
  – Management support for the development team
  – Staffing situation
  – Work output and team scaling
  – Communication patterns

• We still need to work on...
  – Acting holistically within a project-specific funding model
  – More effective use of ops tools to automate routine work

• We are shifting from reboot mode to fine-tuning mode
Lessons Learned

• Know who you are as an organization
• Org-first AND People-first approach
• People are messy
  – HRT - humility, respect, trust - underlie successful org change
  – Expect to do a lot of listening & a lot of talking
  – Assume that everyone has good intentions
  – Don’t hide behind technology - wetware issues require face time
• Change is scary. Expect resistance from the org's immune system
• It Never Ends! The target is moving, be vigilant
Thank you!

• Twitter: @franfabrizio and @pclark
• Email: fran@umn.edu and pclark@umn.edu

while employed? do
observe symptoms and smells
find root causes
design and execute actions to address
review outcomes
propose OSCON session about experience
end