CS 4530 & CS 5500 Software Engineering

Lesson 12.3: Strategies for successful software teams

Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences

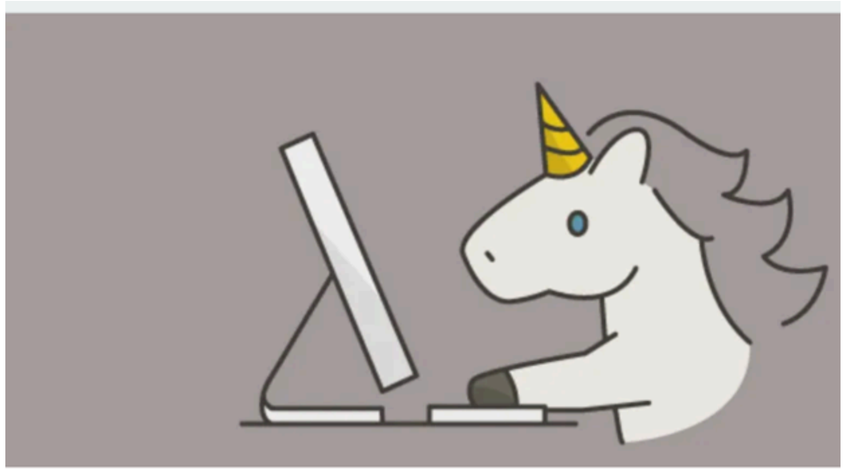
Learning Objectives for this Lesson

By the end of this lesson, you should be able to…

- Explain key advantages of working in a team and sharing information with your team
- Apply root-cause analysis to construct a blameless post-mortem of a team project

“The 10x Engineer”

AKA “The Rock-Star Engineer,” “The Ninja Developer”



What makes a 10x Developer?

#10xdeveloper #productivity #beginners #career

Davide de Paolis Mar 11, 2019 · 6 min read

Playing Nicely with Others

HRT: Three Pillars of Social Skills

- **Pillar 1: Humility**: You are not the center of the universe (nor is your code!). You're neither omniscient nor infallible. You're open to self-improvement.
- **Pillar 2: Respect**: You genuinely care about others you work with. You treat them kindly and appreciate their abilities and accomplishments.
- **Pillar 3: Trust**: You believe others are competent and will do the right thing, and you're OK with letting them drive when appropriate.

From "Debugging Teams" by Ben Collins-Sussman and Brian Fitzpatrick

HRT Applied in Code Review

Consider the following code comment

This is personal

Is this really that black and white?

“Man, you totally got the control flow wrong on that method there. You should be using the standard foobar code pattern like everyone else’

Are we demanding a specific change?

Everyone else does it right, therefore you are stupid

From “Debugging Teams” by Ben Collins-Sussman and Brian Fitzpatrick

HRT Applied in Code Review

Consider the following code comment

> “Man, you totally got the control flow wrong on that method there. You should be using the standard foobar code pattern like everyone else”

> “Hmm, I’m confused by the control flow in this section here. I wonder if the foobar code pattern might make this clearer and easier to maintain?

Humility! This is about *me,* not you

From “Debugging Teams” by Ben Collins-Sussman and Brian Fitzpatrick

The 10x Team, or the 1/10 Team?

Mythical Man-Month: “adding manpower to a late software project makes it later”

- Knowledge sharing needs to scale linearly (or sub linearly) with org growth:
  - Mentorship
  - Q&A
  - Mailing lists
  - Tech talks
  - Documentation

For more on knowledge sharing in teams, see SE@Google Ch 3

Bus Factor & Importance of Knowledge Sharing

How many of your team members are irreplaceable?

Failures are inevitable

In software, humans, and processes

Post-Mortems

Google's Example Case Study

- Shakespeare search service down for 66 minutes during period of high interest in Shakespeare
- Engineer quickly noticed the problem and eventually got a fix in
- How does an organization learn from this?
  - What went well
  - What went wrong
  - Where we got lucky
  - How do we prevent it from happening again?

Name, Blame and Shame

How not to respond to this failure

1. Some engineer contributes to failure or incident
2. Engineer is punished/shamed/blamed/retrained
3. Engineers as a whole become silent on details to management to avoid being scapegoated
4. Management becomes less informed about what actually is happening, do not actually find/fix root causes of incidents
5. Process repeats, amplifying every time

Blameless Post-Mortem

Why blameless?

- What actions did you take at the time?
- What effects did you observe at the time?
- What were the expectations that you had?
- What assumptions did you make?
- What is your understanding of the timeline of events as they occurred?

Blameless Post-Mortem

Google's Example Case Study

Shakespeare Sonnet++ Postmortem (incident #465)
Date: 2015-10-21
Authors: jennifer, martym, agoogler
Status: Complete, action items in progress
Summary: Shakespeare Search down for 66 minutes during period of very high interest in Shakespeare due to discovery of a new sonnet.
Impact: Estimated 1.21B queries lost, no revenue impact.
Root Causes: Cascading failure due to combination of exceptionally high load and a resource leak when searches failed due to terms not being in the Shakespeare corpus. The newly discovered sonnet used a word that had never before appeared in one of Shakespeare's works, which happened to be the term users searched for. Under normal circumstances, the rate of task failures due to resource leaks is low enough to be unnoticed.
Trigger: Latent bug triggered by sudden increase in traffic.
Resolution: Directed traffic to sacrificial cluster and added 10x capacity to mitigate cascading failure. Updated index deployed, resolving interaction with latent bug. Maintaining extra capacity until surge in public interest in new sonnet passes. Resource leak identified and fix deployed.
Detection: Borgmon detected high level of HTTP 500s and paged on-call.

WHAT WENT WELL

- Monitoring quickly alerted us to high rate (reaching ~100%) of HTTP 500s
- Rapidly distributed updated Shakespeare corpus to all clusters

WHAT WENT WRONG

- We're out of practice in responding to cascading failure
- We exceeded our availability error budget (by several orders of magnitude) due to the exceptional surge of traffic that essentially all resulted in failures

WHERE WE GOT LUCKY

- Mailing list of Shakespeare aficionados had a copy of new sonnet available
- Server logs had stack traces pointing to file descriptor exhaustion as cause for crash
- Query-of-death was resolved by pushing new index containing popular search term

Conducting Postmortems

Reflecting on past team performance

- Apply this technique after any event you would like to avoid in the future
- Apply this to technical and non-technical events
- Focus on improvement, resilience, and collaboration: what could any of the actors have done better?
- Google's generic postmortem template

This work is licensed under a Creative Commons Attribution-ShareAlike license