Post Ops: *Personal*

A Non-Surgical[1] Tale of Software, Fragility and Reliability

Todd Underwood, Google

[1] No Systems or Software Engineers were harmed during the production of this presentation. Void in Guam, Alaska, Hawai'i, Puerto Rico and the USVI or where prohibited. Engineer Subjects Review Board approved all procedures, all subjects were appropriately consented.

Post Ops:
A Quirky, Cranky Call to Action

Systems Administration is finished. DevOps is a helpful band aid. We need to move beyond the entire notion of "Administration" and "Operations" in the field of computing (software, systems and networking).

Let us stop doing the machines' work for them.

Let us stop feeding the machines with human blood.

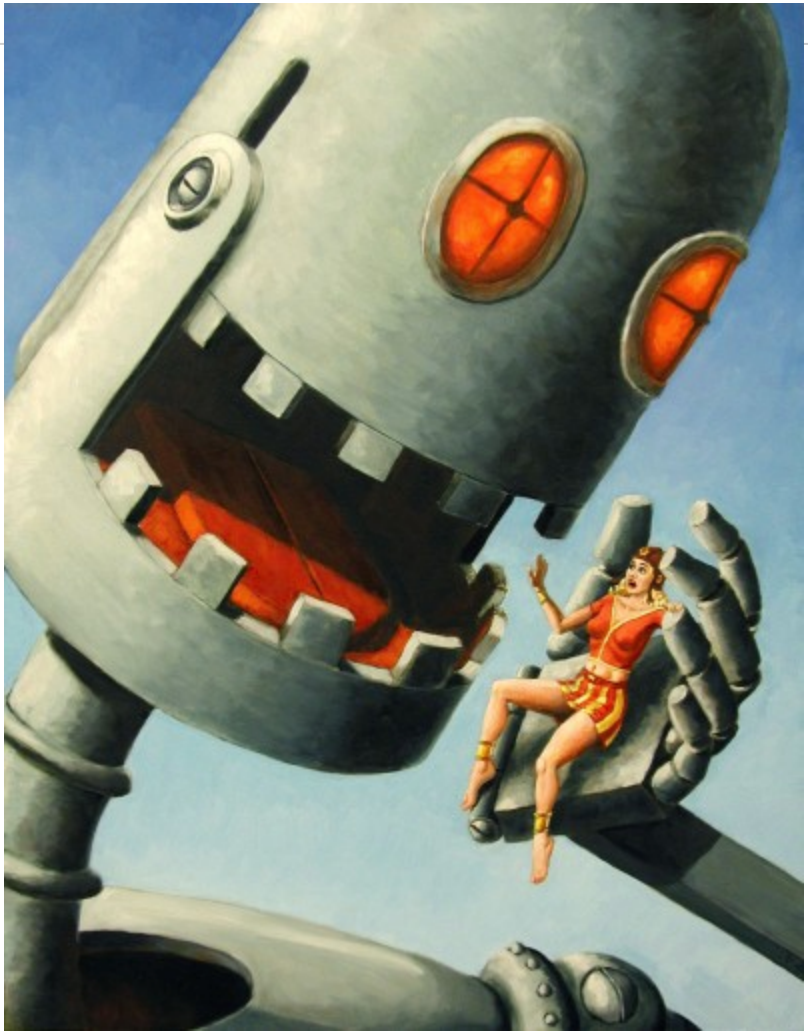
[For Internet-centric, software-built infrastructures. This probably doesn't apply to factories, but then factories never really applied to software, either.]

TOIL

http://www.shorpy.com/node/4324 *Multi-processing computing division, Bonus Bureau, Computing Division. Washington, D.C., USA. 1924. Shall we glorify this set of responsibilities, too?*

http://themanicramblingsofaswede.wordpress.com/2009/07/16/upcoming-military-robot-could-feed-on-dead-bodies/

This blog may be more terrifying than the picture lets on.



http://billzeman.blogspot.com/2006/05/pretty-girls-and-robots.html

http://thegrumpyowl.com/2008/05/11/robots-will-eat-you/

A Bold and Surprising Set of Claims

…to make at a conference of/by/for SysAdmins. Apologies in advance. I was asked to present. Angering or annoying the audience is a happy coincidence. :-)

- I got my first systems gig in the mid-1990s.
- Grew up running Internet Services at an ISP (AS2901 anyone?)
- I've swapped tapes and run cable, I've written crontabs and automated account creation across 5 Unices and WNT.
- I assembled datagrams by hand and calculated checksums on my fingers
- I did some good work and had fun! I think….

Acknowledged:

- Many of you lived through this time and similar environments
- We should re-look at that history with something other than nostalgia and reverence

A Word About Google and Relevance

Google has challenges presenting at operations-centric gatherings. We work on interesting stuff, but, there are always questions:

- Too big?: Google scale is not your scale
- Talent-rich?: Massive amounts of software engineering expertise per m^2. Solutions may be impossible/irrelevant for you.
- Privilege?: Already solved all the problems that you need to solve

The truth:

We are generally engineers trying hard to solve interesting problems that many others will face soon. Our successes, our failures and our reasoning are generally useful, if applied judiciously.

Site Reliability Engineering

A brief diversion into Googleland Production Engineering. This is one useful way to approach these sets of problems, organizationally.


DURI ET PERITI
MISSION CONTROL
SRE
SITE RELIABILITY ENGINEERING

Site Reliability Engineering: A Mythical History

(This is not exactly what happened, but it's somewhat True anyway)

- Google was (still is) ~~cheap~~frugal: costs that scale linearly (or super-linearly) are bad when things get big
  - Applies to hardware
  - Applies to people: no "NOC", no "sysops" in production where possible.
- Software developers run their own code in production
- They ~~hate~~dislike it, so they automate everything:
  - build, push, monitoring,
  - task restarting, task configuration and location,
  - distributed debugging, etc.
- Some are more production-oriented than others. These become the first "Site Reliability Engineers."

SRE is systems and software engineers who solve production problems with software.

We are constructive cynics. Productive pessimists. Creative cranks. Alliterative as……. Nevermind.

SRE Basics

Keep the site up—whatever it takes

- "Site" == google.com
- Site unavailable? Our problem, whatever the reason

Work at a Large Scale

- Many services, lots of data, many machines
- Not so many people. People must scale sub-linearly to services.

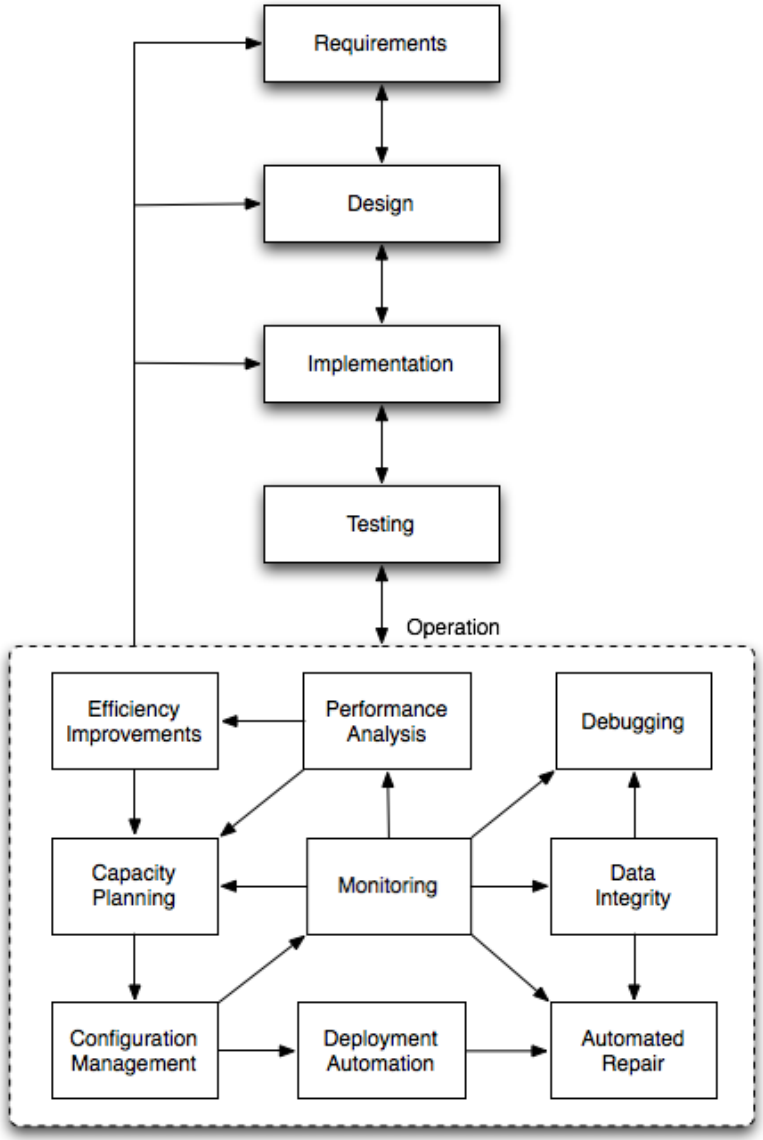
Balance competing demands

- Improve availability and reliability.
- Improve efficiency
- Take on new services (post-launch)

Solve production problems with software. It's all just software.

Post-Deployment Evolution

- Most of a system's life is spent *after* release
- Systems evolve
  - emergent behaviour and interactions with other systems
  - environment changes
  - desired functionality changes



SRE Organizational Structure

The SRE Organization is independent of business units/divisions.

SRE teams are organized around a single service or a collection of related services or technologies:

- Search Ads Serving
- Bigtable[1] Storage Service
- Payments
- Geo Serving
- Search

Services are jointly owned by development engineering and SRE. Both can choose to extend or end SRE work on the service.

SREs are in short supply and can easily work on other services.

SRE—A New Role

- Unlike some other approaches, this **is** a role.
- To build an organization of scale you have to hire and train people at scale.
- There were no SREs in the market—not a job yet.
- There were few SREs hidden among thousands of software developers and senior sysops already at Google.
- None of them knew they might be SREs (or that they might want to be SREs).
- We weren't positive we knew what we were doing:
  - organizationally
  - culturally
  - structurally within google
- This has all taken some time. Things are somewhat sorted now.

SRE Catches On

Linkedin has over 800 postings for a “Site Reliability Engineer” at

- Apple
- Akamai
- Best Buy
- Salesforce
- VMWare
- Twitter
- Tumblr
- Microsoft
- Facebook

This is obviously a popular role now.

Hold That Thought

More on SRE shortly.

Let’s look at the world outside of Google that may seem more familiar.

Whither DevOps?

Isn’t SRE just DevOps. Also, doesn’t the name “DevOps” just sound way better and more clear than SRE? Is this just more evidence that Google hasn’t always marketed our technology as well as we do now? :-)

What About DevOps?

- Devops: a cultural and professional movement.[1]
- A buzzword
- Automation, solving production problems with software
- Communication, collaboration between application developers and infrastructure/operations/IT

Real problem: Siloed software development and production.

Solution: Break down the silo:

- Embed operations in software development
- Establish cultural value of communications
- Build software skills on the operations side
- Solve production problems with software, not labor

[1] Adam Jacob, Velocity 2010, http://www.youtube.com/watch?v=Fx8OBeNmaWw

DevOps and the Validation of Operations

DevOps dramatically improves practices and organizational structure around operations. But it does not forcefully eliminate operations.

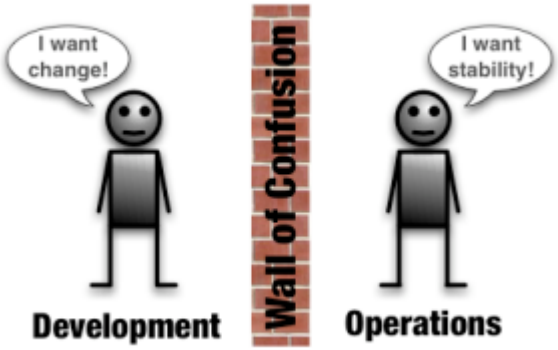
Look at two issues:

- Organizational Structure
- Role of Operations

Each of these contributes to the success, and limitations of DevOps

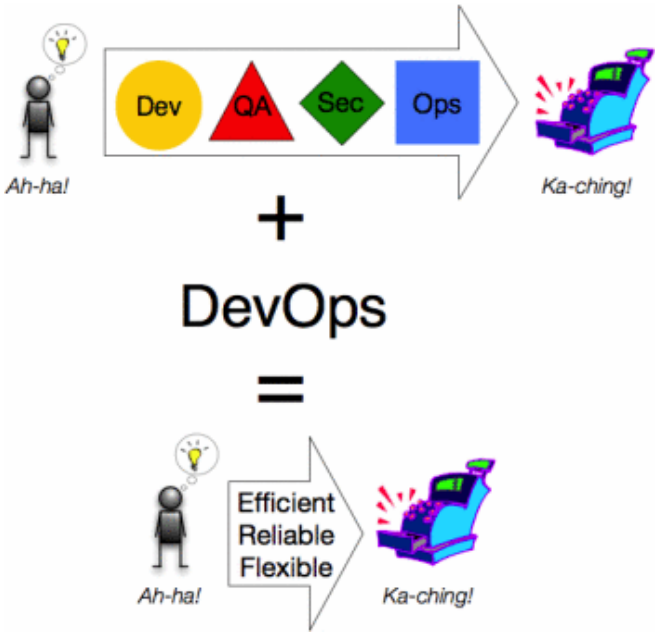
Organizational Structure

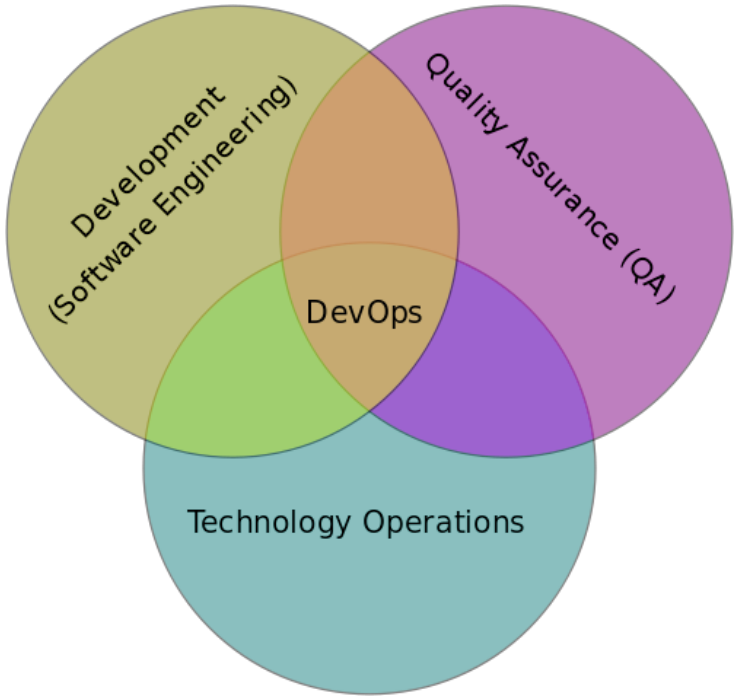
Classical:



(http://dev2ops.org/2010/02/what-is-devops/)

DevOps:





(http://dev2ops.org/2010/11/devops-is-not-a-technology-problem-devops-is-a-business-problem/)

Organizational Structure

DevOps is a way of being and doing. Not a role.

Organizations that practice DevOps have a variety of structures, including, traditional:

- Application Development
- Operations
- Test
- Security

...And a variety of different integrations

Operations: Valued or Vilified

Valued.

DevOps strives to **improve** operations as a discipline, function and role.

DevOps strives to **integrate** operational concerns into business practices and software skills/capabilities into operations.

Operations is where the software meets the users. Operations staff working closely with (and integrated into) development teams.

Operations is central.

Shout Out: Adrian Cockcroft's "NoOps"

Adrian Cockcroft's "NoOps"[1]

Principle: Software developers work directly with production. No operations organization. Automation removes operational tasks entirely.

Platform as a Service: PaaS ; No operations required.

Implicit assumption (me, not Cockcroft):

- Services/jobs not VMs/machines
- scalable/programmable cluster OS that works
- [machine] configuration is the wrong level of abstraction
- configuration management **is** one right problem

[1] http://perfcap.blogspot.com/2012/03/ops-devops-and-noops-at-netflix.html

Operations Research, {Sys|Net}Ops

The historical, academic part of the talk you can ignore, perhaps with some useful tidbits.

A Brief Diversion in History:
Operations Research / Operations Management

- Basic principles:
  - statistical and process expertise to improve some process (often manufacturing).
  - maximum yield of some metric given a set of constraints
- Widespread application outside of manufacturing:
  - War
  - Critical path analysis
  - Network fault analysis
  - Scheduling
  - Project Management

Software-based production? What are the constraints?

OR: Probably Irrelevant for Software

- “Production” in software systems simply lacks the constraints of factory production
- OR critically relevant for the physical infrastructure stack
  - Warehouse-scale computing
  - External adjacencies (power, fibre, other companies) unamenable to quick change
  - That’s a different talk
- Relevant for maximizing SLAs in the short run
  - System has constraints (for now)
  - Small changes can make big differences
  - Constraints shouldn’t be enshrined in costly analysis

It’s all a simple matter of software engineering. Many software developers make mistakes and make them slowly. We should type the right stuff faster.

Software Organizations Should Nix Operations

- Operations comes from the notion of extracting value out of a fixed set of assets.
- System/network operations:
  - Fixed asset (minicomputer, mainframe, RS6k, BFR, T640 name your poison)
  - Depreciating rapidly
  - Extract value by keeping it running

What is the fixed asset being depreciated in EC2? As a cluster application service engineer at Google, what am I "administering"?

Abstractions/software changes the cost of change.[1]

[1] 3D printing will ultimately make factories more like software. It's more important to try figure this out for our world than to roll out OR to our world.

Production vs. Operations

...of babies and bathwater

Operations culture and practice has many admirable characteristics we must not lose:

- Fast, careful troubleshooting
- Ethic of caring about production, availability, users
- Ethic of privacy, security
- Constructive pessimism brought to capacity planning, outage prediction, scaling, future failures in general
- <Insert your best “why I’m proud of being a sysadmin” here>

Production engineering practices and systems should recreate these values effortlessly.

Organizations that move post-Ops should embody these values.

Ops->DevOps->{NoOps|PostOps}

- Ops separate from everything else is terrible (wall of confusion).
- Embedded ops into other teams is a good start but not far enough.
- **Clueful Management** is mandatory
  - Some companies clearly have very technical management
  - If your management cannot understand your job, then either your job or your management don't matter.
- Need just enough operations to identify **new** problems and prioritize development. Cap ops work (50% or less?).
- Every other aspect of operations should be eliminated. With prejudice.

Burden of proof: Operations. Why do we want to preserve this specialty and expertise?

Clusters, Jobs, Tasks

Oh, my! What does a systems engineer or software engineer do in the PostOps/NoOps age?

A Possible Future for Production

Platforms

- hardware selection/custom building
- datacenter selection/building/operations
- power work, physical infrastructure, hardware operations

Infrastructure Software Engineering

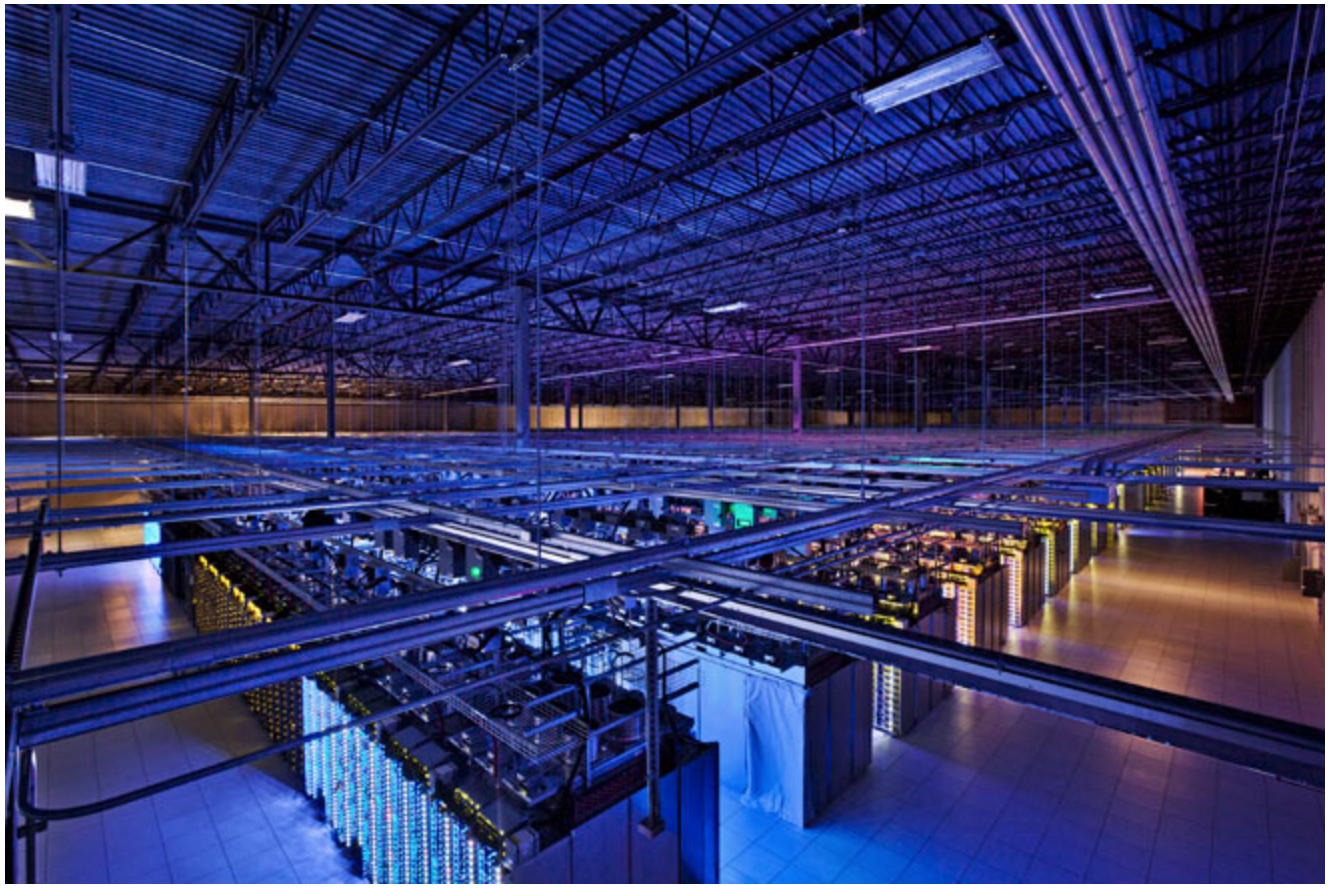
- server/custer OS development/testing
- network/storage/naming/shared services development

<Insert SRE/Production Software Engineer here>

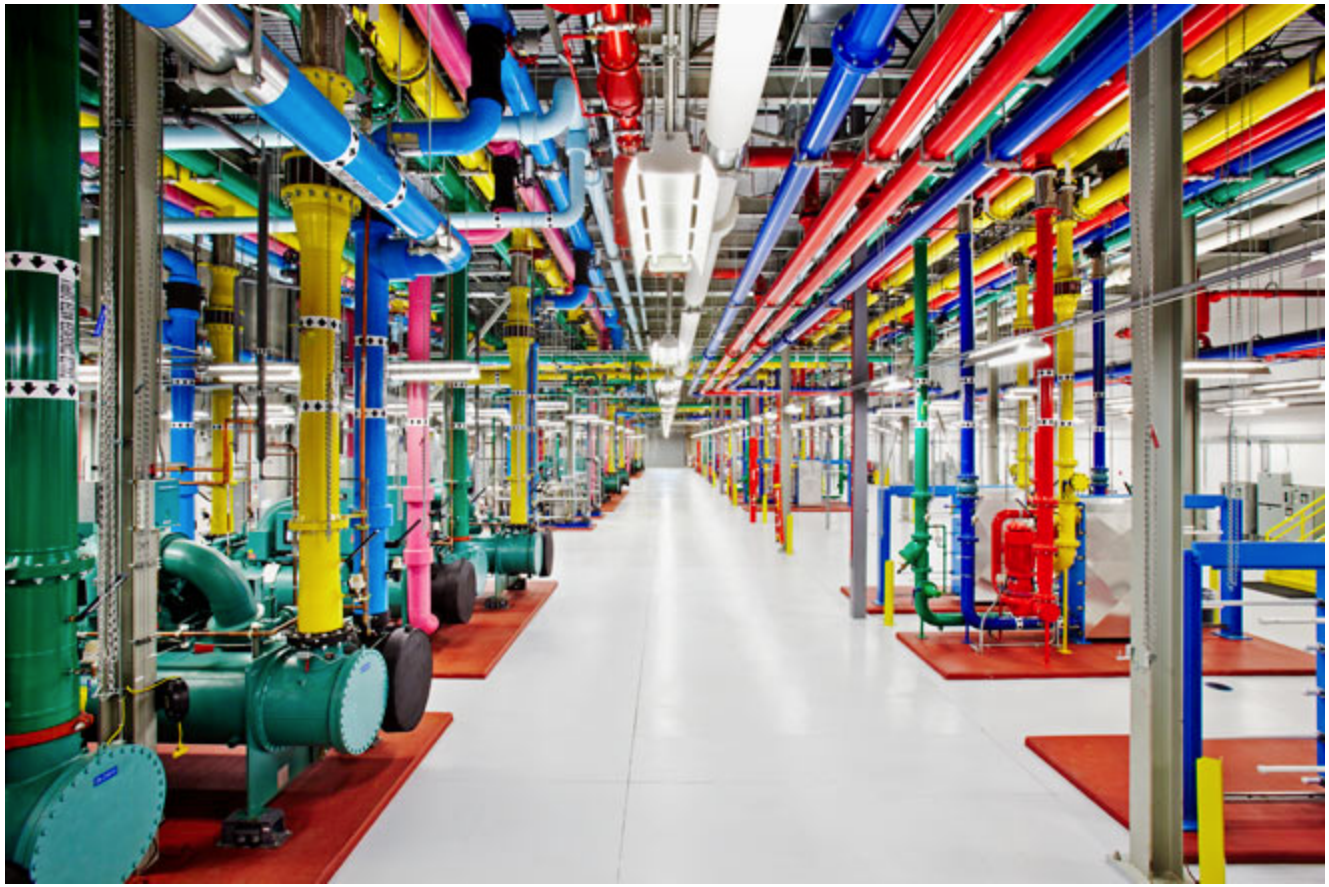
Application Development / Application Administration

Job/Role Counts

- Platforms
  - Bimodal distribution: large numbers of unskilled operators, tiny numbers of superskilled platforms engineers
- Infrastructure Software Engineering
  - It depends… are we going to innovate?
  - Likely: Huge opportunities here for people who have excellent systems software skills
- Systems Administration: The middle that gets squeezed
- Production Software Engineering/SRE
  - Small, real niche. Important but unlikely to grow
- (Distributed) Application Development
  - Huge. You might think it's a long way down the road to the chemists, but that's just peanuts compared to this.



Infrastructure: Google datacenter computer in Iowa



Infrastructure: Cooling systems in Google Datacenter in Georgia

Systems/Production/Operations

Babies

- Production ethic
- Troubleshooting / problem solving
- Building Automation Systems
- Job/system/intent-based configuration management
- Monitoring systems and implementations
- Release engineering / canarying / testing / roll-out
- Capacity planning
- SLA definition/monitoring
- Constructive cynicism

Bathwater

- Rote/repeatable work
- Automatable work
- One-offs
- (OS) configuration management
- Logging in to machines
- Processing individual files
- Heroism

SRE as a Separate Organization/Role

Reliability engineering should pervade your entire software/infrastructure/IT/operations organizations. No one disagrees.

Hegelian Dialectics

Thesis: New Version Shipped!

Antithesis: Current Version Fixed!

Synthesis: New Version Fixed and Shipped!

We found that independence of SRE matters and works. Other organizational solutions may work.

TL;DR

- ~~Life~~The industry moves pretty fast. If you don't stop and look around once in a while, you could miss it.[1]
- Operations tasks for sys admins are, and should be, going away.
- Sys admins should massively improve software skill/attention and either:
  - Build software for infrastructure, or
  - Do production software engineering (SRE)
  - Do something else they love
- Drop the nostalgia of the earlier days. We didn't know what we were doing, no matter how fun it was. We should not want to go back.
- Let's invent a better future.

[1] Ferris Bueller's Day Off. But you knew that, already, obviously.

Questions? Kvetches? Rotten Tomatoes?

A cute kitten to distract the audience.