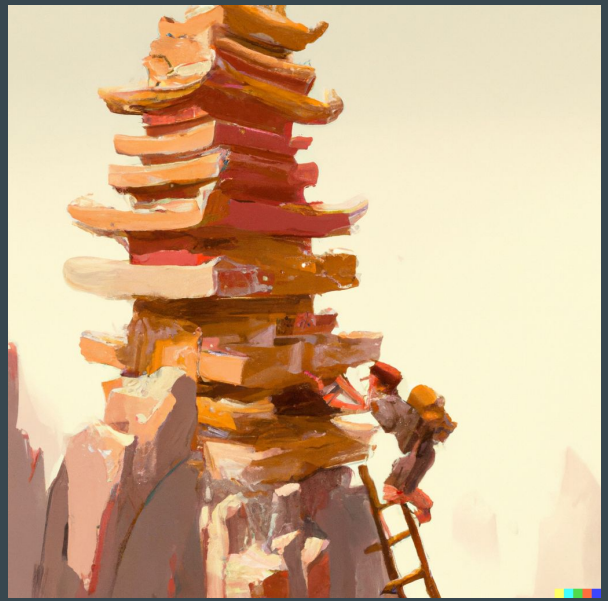
Teaching Site Reliability Engineering as a Computer Science Elective



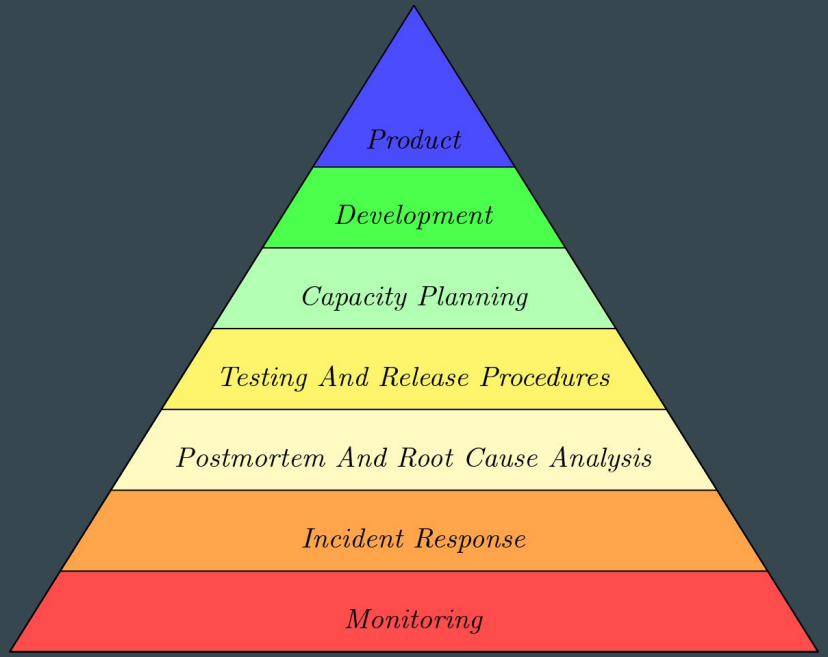
DALL-E: a mountain climber on the side of a towering pagoda made of jenga blocks, digital art

- Mikey Dickerson
  Layer Aleph LLC
  Seattle, Washington
  mikey@layeraleph.com

SRECon 2024: March 20, 2024
San Francisco, CA

Mikey

- B.A.: Math, Pomona College
- Unix systems administrator, Pomona College 2002-06
- "Site Reliability Manager," Google 2006-14
- Administrator of U.S. Digital Service, 2014-17
- Consulting as Layer Aleph LLC since 2017, doing "crisis engineering"?



Dickerson's Hierarchy of Reliability

Motivation

Mikey observes:

New grad hires struggle with the jump to workplace expectations (self-sufficiency)

New grads are best prepared to write new code on a new green-field project with minimal dependencies

They rarely get this job

seen on linkedin, 30 minutes ago

Motivation

State of many CS departments:

Many more students in CS. Now usually the largest major in this self-regarded liberal arts college. (40-50 out of a class of 400.)

College has worked to shift the student body to more lower-income, first-generation, and underrepresented groups.

Identity crisis in higher education.

- Don't want to be vocational training
- Don't want to exclusively serve trust fund babies
- ????

Course design: two parallel sequences

Practicum

Teams of 4 or 5

Run through a series of ~weekly assignments ("milestones") that start with a git repository and end with a reasonable facsimile of a customer-facing production service

Reading

Well-studied systems disasters: Challenger, Three Mile Island, Air France 447

Sensemaking, from Karl Weick &co.

Systems safety from Nancy Leveson

Host institution

Pomona College

Claremont, CA

4-year liberal arts college, student body around 1600

No relation

Practicum sequence

00 Get an AWS account, set up a VM, ssh

01 Install postgres, import the starter data, answer some questions with SQL.

02 Install tomcat, python, and the webapp and service process

03 Make the above actually work together (difficulty spike)

04 Set up APM-style instrumentation

05 Set up an oncall rotation with pagerduty etc

06 Show improvement over week 05 availability under synthetic load of about 100 rpm (confidence improves)

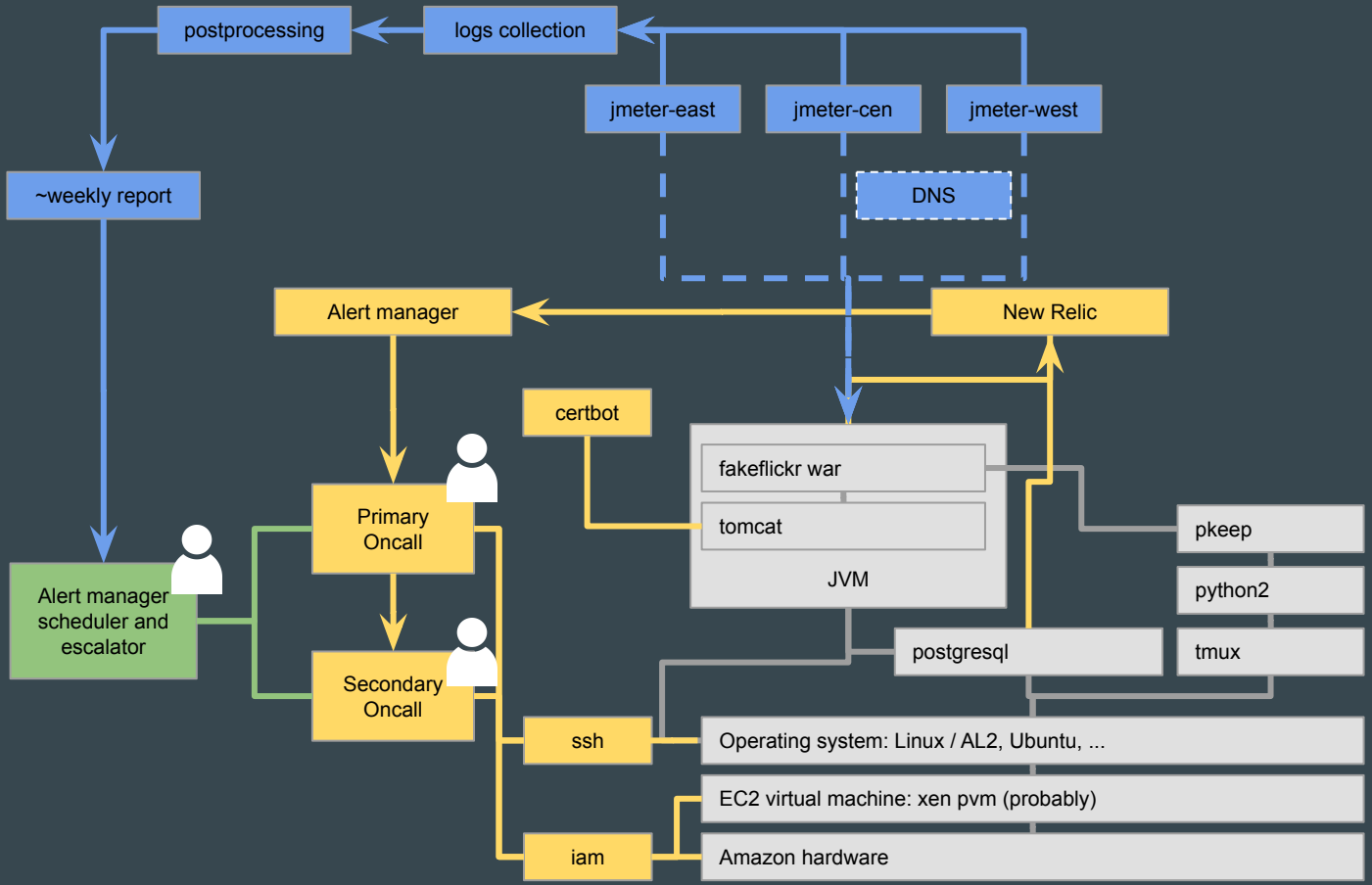
07 Show 90% availability (+side activity)

08 Solve a synthetic outage and write a postmortem

09 Rearchitect and scale up to withstand a 15 minute load test of 10,000 rpm

10 Deploy changes with CI/CD, sometimes containers

as of week 05


postprocessing
logs collection
jmeter-east
jmeter-cen
jmeter-west
~weekly report
DNS
Alert manager
New Relic
certbot
fakeflickr war
tomcat
JVM
Primary Oncall
Alert manager scheduler and escalator
Secondary Oncall
pkeep
python2
tmux
postgresql
ssh
Operating system: Linux / AL2, Ubuntu, ...
EC2 virtual machine: xen pvm (probably)
iam
Amazon hardware

Course design: Non-prescriptive problem solving

Intention is to address the mindset

They start at "if the step by step instructions do not work exactly as written, I am stuck"

Need to get to "I can solve this problem with the assortment of tools and capabilities that I have, even though all of them are imperfect."

Strategy is to remove a lot of the scaffolding

Goal statements try to be concrete and clear (eg "site responds with HTTP 200 over 99% of the time")

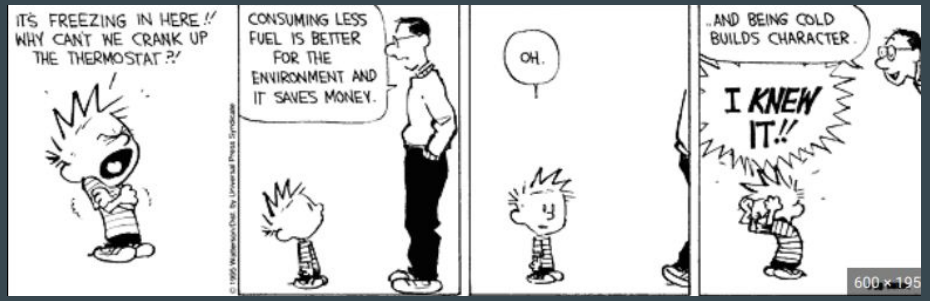
Instructions are generally only in the form of hints and pointers to published documentation

Course design: Planned surprises

Intention is to build resilience and adaptability

see for instance the work of John Allspaw, Richard Cook, DD Woods re: "adaptive capacity"

A successful complex system has computers doing all the rote tasks and humans handling the unpredictable challenges.



Strategy is to change requirements, violate assumptions about the course.

See Ray Dawson 2000, "Twenty Dirty Tricks"

Nuclear reactor sensemaking game

Course design: Unplanned surprises

Tools in everyday use don't have the safety margin to which students are accustomed

In Year 1, one group did an "rm -rf /" on their server VM in week 4, obliterating their first month of work.

Afterwards they created backups and version control

Hole Hawg:
see "In the beginning was the command line" by Neal Stephenson

In Year 2, one group destroyed their serving VM image and backups in week 8, with confused application of package upgrades

This would have been an excellent time to introduce containers

Reading and discussion sequence

Goals:

- encourage systems thinking
- build resilience
- (light) awareness of the effects of technology on the world

Disasters of a specific complex-systems nature:

- Mann Gulch wildfire
- Air France 447
- Chernobyl / Three Mile Island
- Therac-25

Theory drawn from two main sources:

"Sensemaking" by Karl Weick et al.

"Systems Safety" by Nancy Leveson et al.

Reactions and results

17 students enrolled in 2021

33 in 2022

31 in 2024

24 women, 19 from underrepresented groups

Three drops, two incompletes (so far)

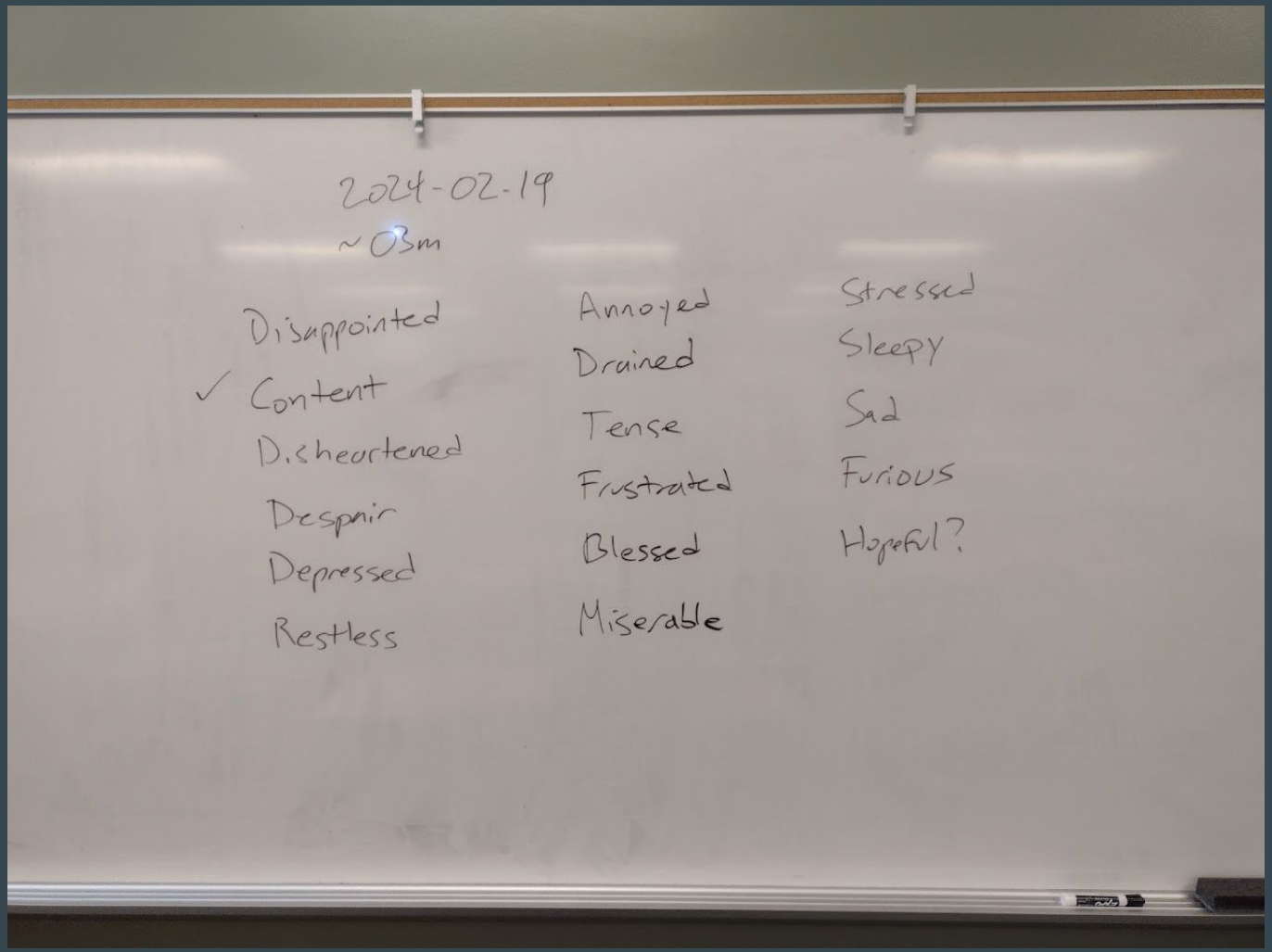
Grades tightly clustered around B+/A-

Qualitative open-form feedback (2021/2022):

11 said "best" or "favorite" class ever, or tied for best/favorite class ever

6 said "increased problem-solving confidence," 5 of these were women

Negative comments were about workload (all would prefer less), and balance of time spent on various topics (no clear trend)


2024-02-19
~03m
Disappointed
✓ Content
Dicheartened
Despair
Depressed
Restless
Annoyed
Drained
Tense
Frustrated
Blessed
Miserable
Stressed
Sleepy
Sad
Furious
Hopeful?

06m 2024.03.06

Complacent
Grateful
+ Secure
+ Mellow
Peaceful
At ease

Sleepy
Pleasant
Chill
Uneasy
Tired
Apprehensive

Ecstatic
Worried

Week 7

	2024-03-07	2024-03-08	2024-03-09	2024-03-10	2024-03-11	2024-03-12	2024-03-13
hotpink	1.000	0.960	0.904	0.987	0.949	0.989	0.999
lightsalmon	0.990	1.000	0.999	0.999	0.999	1.000	1.000
red	0.936	0.999	0.999	0.894	0.951	0.787	0.992
ivory	1.000	0.847	0.906	1.000	0.990	0.817	1.000
honeydew	0.063	0.030	0.649	0.934	1.000	0.713	1.000
darkturquoise	1.000	1.000	1.000	1.000	0.939	0.987	1.000
papayawhip	0.999	0.999	0.999	0.999	0.999	0.999	0.999

Behind-the-scenes work for instructor

Challenging to keep up with fast-mutating AWS and SaaS services

Challenging to troubleshoot oddball corners of system behavior that you would have never gotten into on your own

Must build and operate the synthetic load testing, availability monitoring, etc. (this must be at least an order more reliable than the students!)

Watch closely for problematic team dynamics

Stereotypical gender roles present immediately

Aside: DevOps or SRE?

SRE: Google 2003, The SRE Book 2016

DevOps: 2009, more of a deliberate coordinated push ("DevOpsDays")

Practical differences are few, I don't think they matter for our purposes

DevOps tends to start with "how can we make our programmers more productive"

SRE tends to start with "how can we make this system more reliable"

One perspective: SRE is "intolerance of poorly performing systems" plus "intolerance of rote repetition in human work"

Future is uncertain

Materials available on request, but:

Instructors that can teach this are rarely found in academia

Host institutions usually require academic credentials

Probably running the last class at Pomona right now

Hope to gather data from first two cohorts at 3-5 years post-graduation

Could be adapted for corporate or government settings?

Questions