

CS130: Software Engineering

Lecture 16: Postmortems Team Structure



<https://forms.gle/CFEmn3GqKH3dJg8k7>

- Tweet: Give a tip for writing docs
- Tweet: Give a tip for handling interpersonal conflict

Documentation

Starting on a new project...

1. Read requirements docs
2. Read design docs
3. Read API docs
4. Read code

Product requirements documents (PRD)

- Why?
- What?

Technical design docs (DD)

- Why?
- What?
- How?

DNS Prefetching

- Why? (The problem)

DNS resolution is slow

- What? (The solution)

Resolve addresses before user clicks link.

- How? (The implementation details)

Product requirements documents (PRD)

- Vision
- Motivation
- Goals
- Requirements
- Success criteria

Technical design docs (DD)

- Objective
- Background
- Requirements
- Detailed design
- Alternatives considered

Good requirements are...

- Unambiguous
 - Complete
 - Verifiable
 - Consistent
 - Modifiable
 - Traceable
 - Usable for the lifetime of the product (implementation, maintenance)
- Use strong language like “shall” and “will”
 - Avoid ambiguous language like “could” and “may”
 - Quantify verification criteria, not just “be good”

Technical design docs: extra credit

- Security
 - Privacy
 - Testing plan
 - Work estimate (how long?)
- Performance - latency, throughput, scalability
 - Migration strategy
 - Reliability - SLA
 - Deployment and launch plan

API docs

- How to use the code
- What are the classes?
- What do the methods do?

Example of good API docs:

<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>

JavaDoc

- Generate API docs from special comments
- `/** ... */`

```
/**
 * Returns a new string that is a substring of this string. The
 * substring begins at the specified <code>beginIndex</code>
 * and extends to the character at index <code>endIndex -
 * 1</code>. Thus the length of the substring is
 * <code>endIndex-beginIndex</code>.
 * <p>
 * Examples:
 * <blockquote><pre>
 * "hamburger".substring(4, 8) returns "urge"
 * "smiles".substring(1, 5) returns "mile"
 * </pre></blockquote>
 *
 * @param    beginIndex    the beginning index, inclusive.
 * @param    endIndex      the ending index, exclusive.
 * @return    the specified substring.
 * @exception IndexOutOfBoundsException if the
 *           <code>beginIndex</code> is negative, or
 *           <code>endIndex</code> is larger than the length
 *           of this <code>String</code> object, or
 *           <code>beginIndex</code> is larger than
 *           <code>endIndex</code>.
 */
public String substring(int beginIndex, int endIndex) {
    ...
}
```

Quick survey

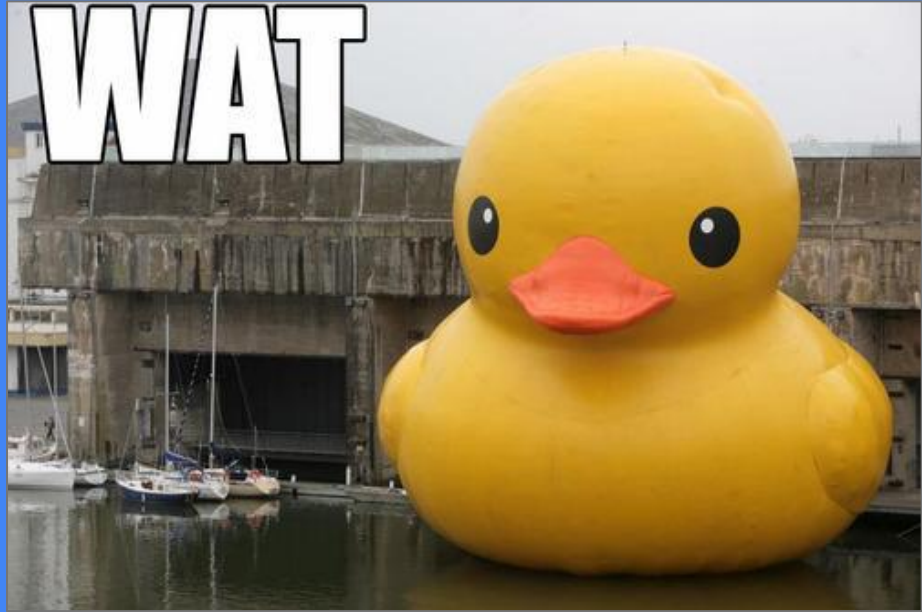
- How many people have written code for this class?
- How many people have written design docs for this class?
- How many people have written API docs for this class?

READ THE SOURCE LUKE



"The source code is
the documentation"

```
initServletSystem(  
    true, false, 1500, true, true);
```



"The source code is the documentation"

Better...

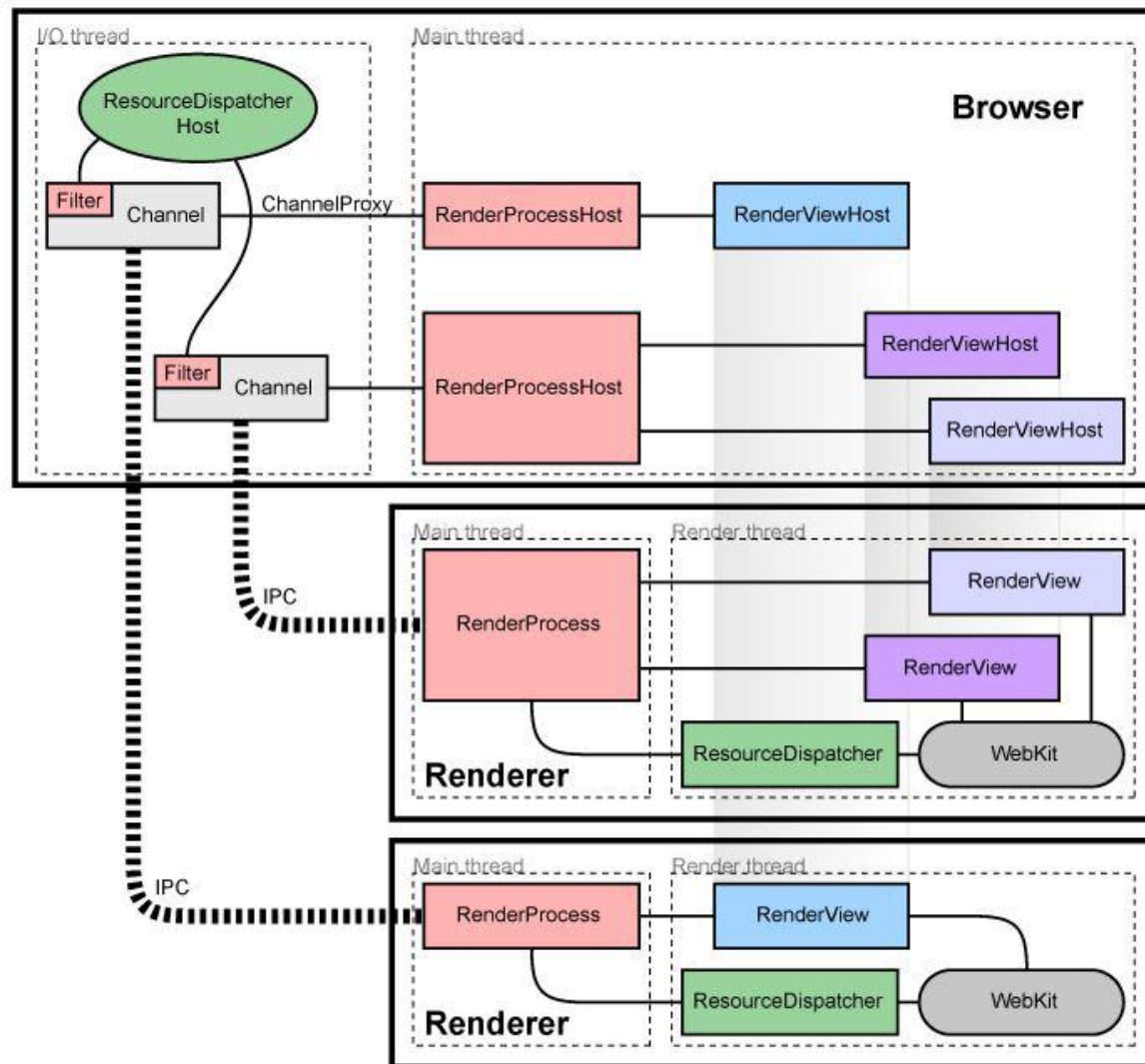
```
initServletSystem(true, // Use authentication
                  false, // Don't enable the cache
                  1500, // 1500 ms initialization timeout
                  true, // Continue startup on servlet init error
                  true); // Enable logging
```

"The source code is the documentation"

Even better...

```
initServletSystem(ServletAuth.ENABLED,  
                  ServletCache.DISABLED,  
                  ServletSystem.INIT_TIMEOUT_1500_MS,  
                  ServletSystem.CONTINUE_ON_ERROR,  
                  Logging.ENABLED);
```

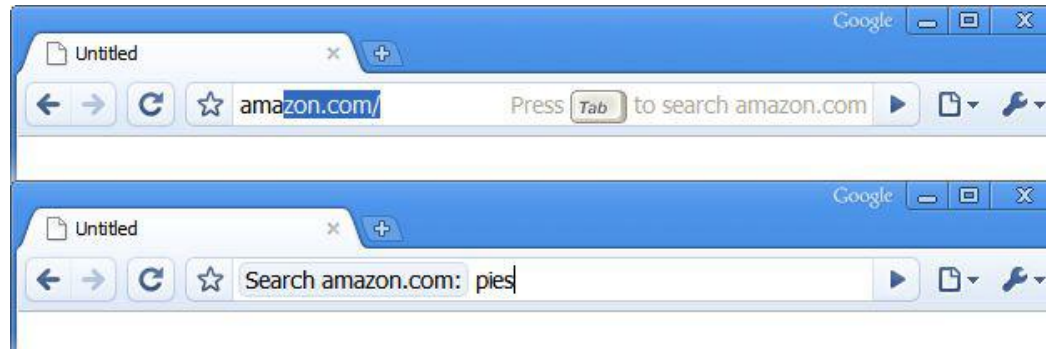
This is just the beginning...



Use cases....

Tab to Search

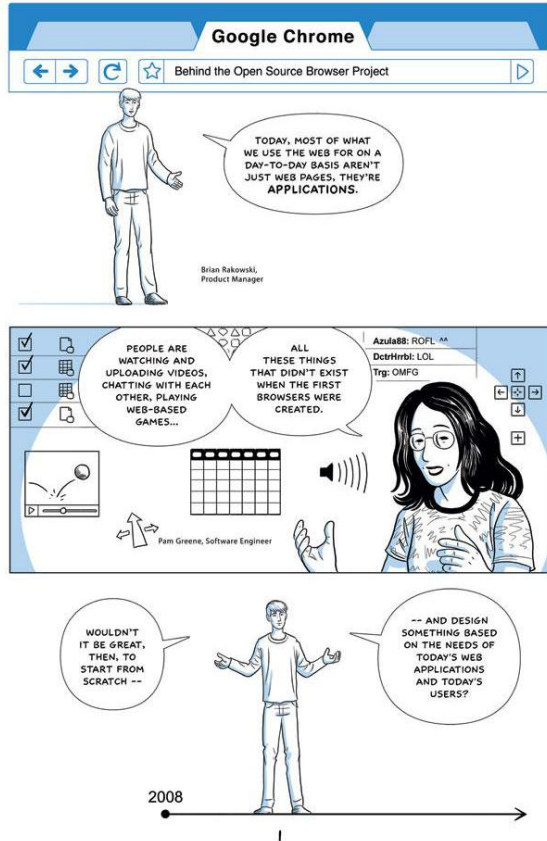
If a user is partway through typing something that auto-completes to a keyword, we allow the user to press **Tab** to jump to the end of the auto-completion and begin their keyword search term input. This is powerful when combined with our automatic keyword creation - a user who does a search on amazon.com in the course of their normal browsing will be presented with the option of pressing **Tab** to do an Amazon search the next time they type something that auto-completes to amazon.com, leading to the ability to type 'ama [tab to complete] pies'. We call this functionality 'tab to search'.



And more...

- User stories
- UI mocks
- Tutorials
- End-user documentation
- Contributor docs (how to be a team member)
- FAQs

Even cartoons!



Postmortems

“Cherish your system-failures”

—*The Systems Bible* by John Gall

A long ago outage



How a typo took down
S3, the backbone of the
internet

The Verge · 2 days ago



Amazon S3 outage
spotlights disaster
recovery tradeoffs

SearchAWS.com - TechTarget · ...



Employee Error to
Blame For Huge
Amazon S3 Outage This
Week

DSLReports · 1 day ago

Dissecting a recent postmortem

Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region

We'd like to give you some additional information about the service disruption that occurred in the Northern Virginia (US-EAST-1) Region on the morning of February 28th. The Amazon Simple Storage Service (S3) team was debugging an issue causing the S3 billing system to progress more slowly than expected. At 9:37AM PST, an authorized S3 team member using an established playbook executed a command which was intended to remove a small number of servers for one of the S3 subsystems that is used by the S3 billing process. Unfortunately, one of the inputs to the command was entered incorrectly and a larger set of servers was removed than intended. The servers that were inadvertently removed supported two other S3 subsystems. One of these subsystems, the index subsystem, manages the metadata and location information of all S3 objects in the region. This subsystem is necessary to serve all GET, LIST, PUT, and DELETE requests. The second subsystem, the placement subsystem, manages allocation of new storage and requires the index subsystem to be functioning properly to correctly operate. The placement subsystem is used during PUT requests to allocate storage for new objects. Removing a significant portion of the capacity caused each of these systems to require a full restart. While these subsystems were being restarted, S3 was unable to service requests. Other AWS services in the US-EAST-1 Region that rely on S3 for storage, including the S3 console, Amazon Elastic Compute Cloud (EC2) new instance launches, Amazon Elastic Block Store (EBS) volumes (when data was needed from a S3 snapshot), and AWS Lambda were also impacted while the S3 APIs were unavailable.

S3 subsystems are designed to support the removal or failure of significant capacity with little or no customer impact. We build our systems with the assumption that things will occasionally fail, and we rely on the ability to remove and replace capacity as one of our core operational processes. While this is an operation that we have relied on to maintain our systems since the launch of S3, we have not completely restarted the index subsystem or the placement subsystem in our larger regions for many years. S3 has experienced massive growth over the last several years and the process of restarting these services and running the necessary safety checks to validate the integrity of the metadata took longer than expected. The index subsystem was the first of the two affected subsystems that needed to be restarted. By 12:26PM PST, the index subsystem had activated enough capacity to begin servicing S3 GET, LIST, and DELETE requests. By 1:18PM PST, the index subsystem was fully recovered and GET, LIST, and DELETE APIs were functioning normally. The S3 PUT API also required the placement subsystem. The placement subsystem began recovery when the index subsystem was functional and finished recovery at 1:54PM PST. At this point, S3 was operating normally. Other AWS services that were impacted by this event began recovering. Some of these services had accumulated a backlog of work during the S3 disruption and required additional time to fully recover.

We are making several changes as a result of this operational event. While removal of capacity is a key operational practice, in this instance, the tool used allowed too much capacity to be removed too quickly. We have modified this tool to remove capacity more slowly and added safeguards to prevent capacity from being removed when it will take any subsystem below its minimum required capacity level. This will prevent an incorrect input from triggering a similar event in the future. We are also auditing our other operational tools to ensure we have similar safety checks. We will also make changes to improve the recovery time of key S3 subsystems. We employ multiple techniques to allow our services to recover from any failure quickly. One of the most important involves breaking services into small partitions which we call cells. By factoring services into cells, engineering teams can assess and thoroughly test recovery processes of even the largest service or subsystem. As S3 has scaled, the team has done considerable work to refactor parts of the service into smaller cells to reduce blast radius and improve recovery. During this event, the recovery time of the index subsystem still took longer than we expected. The S3 team had planned further partitioning of the index subsystem later this year. We are reprioritizing that work to begin immediately.

From the beginning of this event until 11:37AM PST, we were unable to update the individual services' status on the AWS Service Health Dashboard (SHD) because of a dependency the SHD administration console has on Amazon S3. Instead, we used the AWS Twitter feed (@AWScloud) and SHD banner text to communicate status until we were able to update the individual services' status on the SHD. We understand that the SHD provides important visibility to our customers during operational events and we have changed the SHD administration console to run across multiple AWS regions.

Finally, we want to apologize for the impact this event caused for our customers. While we are proud of our long track record of availability with Amazon S3, we know how critical this service is to our customers, their applications and end users, and their businesses. We will do everything we can to learn from this event and use it to improve our availability even further.

Summary

“Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region”

Impact

- “At 9:37AM PST...a larger set of servers was removed than intended”
- “Supported...the index subsystem...necessary to serve all GET, LIST, PUT, and DELETE requests ... [and] the placement subsystem... used during PUT requests”
- “S3 console, Amazon Elastic Compute Cloud (EC2) new instance launches, Amazon Elastic Block Store (EBS) volumes (when data was needed from a S3 snapshot), and AWS Lambda were also impacted while the S3 APIs were unavailable”
- The final S3 system “finished recovery at 1:54PM PST”
- Some other services “had accumulated a backlog of work during the S3 disruption and required additional time to fully recover.”

Even more impact

“From the beginning of this event until 11:37AM PST, we were unable to update the individual services’ status on the AWS Service Health Dashboard (SHD) because of a dependency the SHD administration console has on Amazon S3. Instead, we used the **AWS Twitter feed** (@AWSCloud) and SHD banner text to communicate status until we were able to update the individual services’ status on the SHD.”

Root cause

“An authorized S3 team member using an established playbook executed a command which was intended to remove a small number of servers for one of the S3 subsystems that is used by the S3 billing process. Unfortunately, one of the inputs to the command was entered incorrectly and a larger set of servers was removed than intended.”

Timeline

- 9:37AM: Bad command entered, beginning of outage
- x:xxAM: Discovery from monitoring
- x:xxAM: Internal activities
- 11:37AM: Service Health Dashboard restored
- 12:26PM: Index subsystem has enough capacity to begin servicing S3 GET, LIST, and DELETE requests
- 1:18PM: Index subsystem restored and GET, LIST, and DELETE APIs were functioning normally
- 1:54PM: Placement subsystem finished recovery. End of PUT API outage
- x:xxPM: End of dependent service outage

Analysis

What went well:

- Twitter as backup to health dashboard
- (Other items probably omitted in this public disclosure.)

Poorly:

- Able to take down the service with one command.
- Took a long time to restart systems.
- Didn't know it would take so long to restart.
- Health dashboard depends on S3, single-homed.
- ...

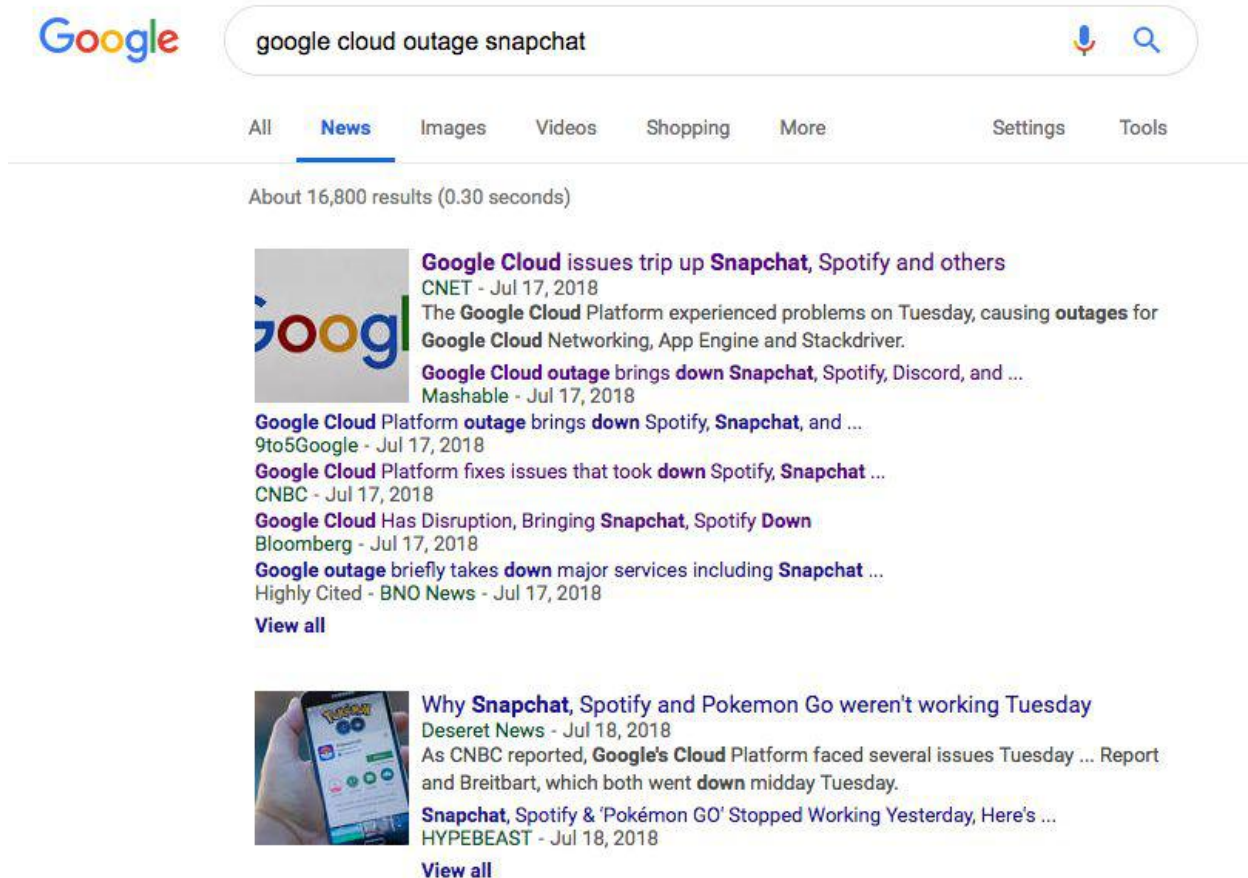
Action items

- “The tool used allowed too much capacity to be removed too quickly. We have modified this tool to remove capacity more slowly and added safeguards to prevent capacity from being removed when it will take any subsystem below its minimum required capacity level.”
- “auditing our other operational tools to ensure we have similar safety checks.”
- “make changes to improve the recovery time of key S3 subsystems”
- “changed the SHD administration console to run across multiple AWS regions”

What is *not* in this postmortem?

“Unfortunately, one of the inputs to the command was entered incorrectly and a larger set of servers was removed than intended.”

Bonus round



The screenshot shows a Google search interface with the query "google cloud outage snapchat". The search results are filtered by "News". The first result is from CNET, dated July 17, 2018, titled "Google Cloud issues trip up Snapchat, Spotify and others". The snippet states: "The Google Cloud Platform experienced problems on Tuesday, causing outages for Google Cloud Networking, App Engine and Stackdriver." Below this is a result from Mashable, dated July 17, 2018, titled "Google Cloud outage brings down Snapchat, Spotify, Discord, and ...". The snippet states: "Google Cloud Platform outage brings down Spotify, Snapchat, and ...". Other visible results include 9to5Google, CNBC, Bloomberg, and BNO News, all dated July 17, 2018. A second result is from Deseret News, dated July 18, 2018, titled "Why Snapchat, Spotify and Pokemon Go weren't working Tuesday". The snippet states: "As CNBC reported, Google's Cloud Platform faced several issues Tuesday ... Report and Breitbart, which both went down midday Tuesday." Below this is a result from HYPEBEAST, dated July 18, 2018, titled "Snapchat, Spotify & 'Pokémon GO' Stopped Working Yesterday, Here's ...".

Google

google cloud outage snapchat

All News Images Videos Shopping More Settings Tools

About 16,800 results (0.30 seconds)

Google Cloud issues trip up **Snapchat**, Spotify and others
CNET - Jul 17, 2018
The **Google Cloud** Platform experienced problems on Tuesday, causing **outages** for **Google Cloud** Networking, App Engine and Stackdriver.
Google Cloud outage brings **down Snapchat**, Spotify, Discord, and ...
Mashable - Jul 17, 2018

Google Cloud Platform **outage** brings **down** Spotify, **Snapchat**, and ...
9to5Google - Jul 17, 2018
Google Cloud Platform fixes issues that took **down** Spotify, **Snapchat** ...
CNBC - Jul 17, 2018
Google Cloud Has Disruption, Bringing **Snapchat**, Spotify **Down**
Bloomberg - Jul 17, 2018
Google outage briefly takes **down** major services including **Snapchat** ...
Highly Cited - BNO News - Jul 17, 2018
[View all](#)

Why Snapchat, Spotify and Pokemon Go weren't working Tuesday
Deseret News - Jul 18, 2018
As CNBC reported, **Google's Cloud** Platform faced several issues Tuesday ... Report and Breitbart, which both went **down** midday Tuesday.
Snapchat, Spotify & 'Pokémon GO' Stopped Working Yesterday, Here's ...
HYPEBEAST - Jul 18, 2018
[View all](#)

<https://status.cloud.google.com/incident/cloud-networking/18012>

Bringing it all together

- Title
- Summary
- Impact
- Root Cause
- Timeline (including beginning and ending of outage)
- What went well/poorly
- Action items

Why are so many things broken?

Things were always broken

- Complex systems are operating in multiple failure modes all the time.
- If you've ever stepped through your code to find a bug, you've probably noticed other bugs.
- The good news: Fixing all the things can help make the system more robust.

Consider the cost of prevention

- Cost > benefit?
- Sometimes preventing errors is too expensive



Small, informal postmortems

- Get into the postmortem mindset
- After anything that goes wrong, spend a moment to think, what went well, what would have gone better?
- In particular, “how could this have been prevented or detected automatically?”

Asking others for postmortems

- When there is a customer-facing outage
- If another team is asking you to do something above and beyond what you normally do

Retrospectives

- Just like a postmortem, but no outage
- What went well, what could go better?
- Usually just fix 1 or 2 things
- If you find they aren't helping, stop doing them
- A good way to deal with prior overzealous postmortem action items
 - “Is this step/system/activity still useful?”

Team Structure

Q&A with Ben Collins-Sussman

What do you know about software engineering that a new grad software engineer probably doesn't?

“I know that software is mostly made of people. That is, to be a successful engineer, you need not just technical skills, but strong social and collaborative skills too.”

—Ben Collins-Sussman, Feb. 2017

Almost certain to be on your team

- Engineers
- Management
- Tech Lead

Likely to be on your team

- PMs
- Testers

If your project has user-facing UI, also:

- UI Designers
- UX Researchers

Sometimes on your team

- Project manager
- Architect
- Operations
- Tech Writer
- Release Engineer

Occasionally on or near your team

- Researchers

The business

- Executives/Upper Management
- Marketing
- Sales
- Product Experts

Human resources

When your issue is *really* about people

Levels (from low to high)

- Fixing bugs, doing small commits on the project
- Doing larger-scale changes
- Designing features or modules
- Designing systems, writing design docs
- What services are we going to need?
- Starting new projects

Over time, you will need less direction and will take more initiative.

Prevailing development philosophies

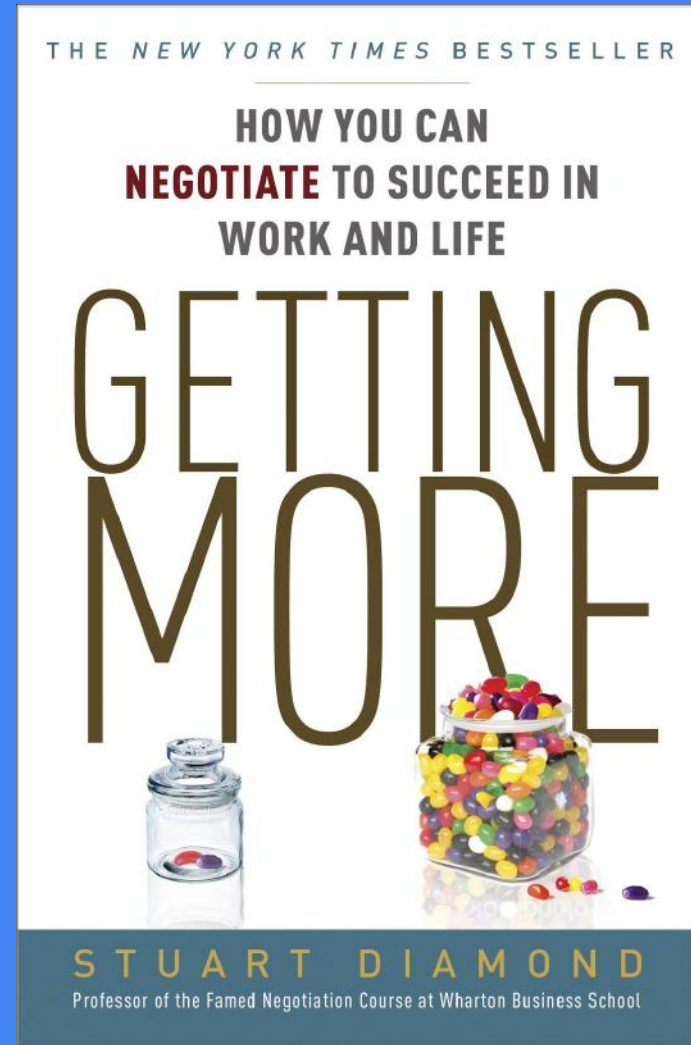
“Waterfall Model”

- Gather requirements
- High-level design
- Low-level design
- Coding
- Testing
- Release

Agile

- Short “sprints” on the order of weeks, keep quality high at all points.

Bonus: Negotiation Crash Course



Every interaction is a negotiation

Focus on your goals

The most important person in a negotiation is them

Make emotional payments

Be incremental

Trade things you value unequally

Find their standards/policies

Be transparent and constructive

Prepare

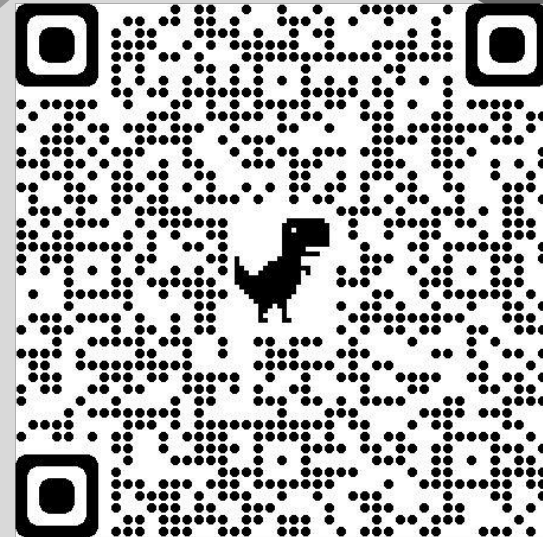
Bonus 2: Watch out for these!

- Everyone is busy, but no one knows what is going on
- No particular reason the org has for what it's doing
- Many levels of indirection to customers
- No customers
- No sponsorship or direct relevance to a business strategy

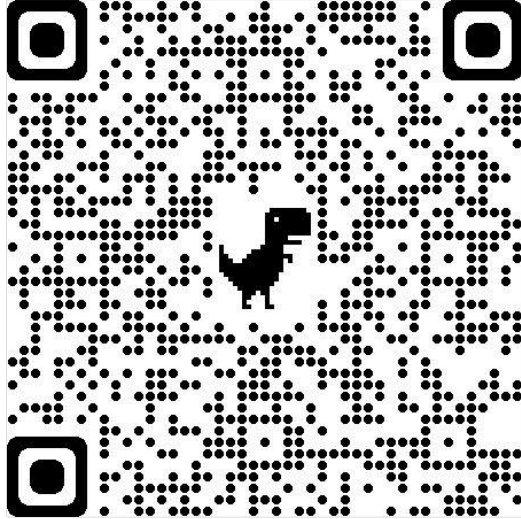
<https://bit.ly/3LHFQpd>

Tweet: What would you put in docs?
(wrong answers only)

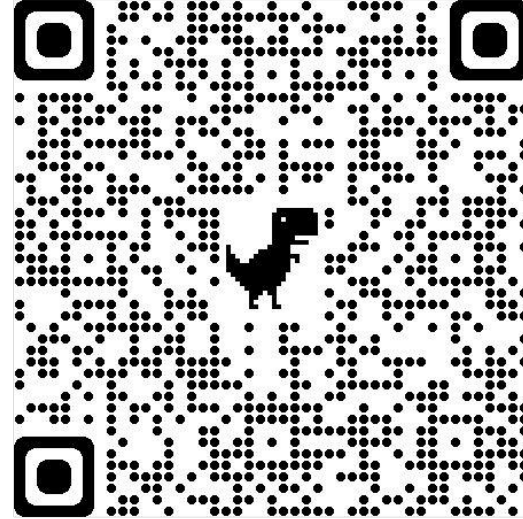
Tweet: Your service had an outage,
What's the worst thing you do after?



it's your favourite time of the quarter...
~ ethics survey time ~



[survey part two](#)
(help fellow students)



[interview interest form](#)
(\$20 Amazon Gift Card!)

any questions? ask me now, shoot me an email at
matt@matthewwang.me, or see [this link](#)