How to be Wrong

or ... “How to be Great at Being Wrong”

Russ Miles - @russmiles
Who are you?
Also...
Russ Miles
CEO, co-founder

Sylvain Hellegouarch
CTO, co-founder

Your Host Today
“To support our users in establishing their own Resilience Engineering Capability”
What is “Wrong”?
And why are we scared of it?
“not correct or true; incorrect.”
“an injurious, unfair, or unjust act”...
...“action or conduct **inflicting** harm without due provocation or just cause”
“a violation or invasion of the legal rights of another”
When are we ever wrong?
“the state of being mistaken or incorrect”
I’ve got some bad news for you...
We’re wrong all the time.
October 21 post-incident analysis

Oct 30, 2018  jasoncwarner  Engineering

Last week, GitHub experienced an incident that resulted in degraded service for 24 hours and 11 minutes. While portions of our platform were not affected by this incident, multiple internal systems were affected which resulted in our displaying of information that was out of date and inconsistent. Ultimately, no user data was lost; however manual reconciliation for a few seconds of database writes is still in progress. For the majority of the incident, GitHub was also unable to serve webhook events or build and publish GitHub Pages sites.

All of us at GitHub would like to sincerely apologize for the impact this caused to each and every one of you. We’re aware of the trust you place in GitHub and take pride in building resilient systems that enable our platform to remain highly available. With this incident, we failed you, and we are deeply sorry. While we cannot undo the problems that were created by GitHub’s platform being unusable for an extended period of time, we can explain the events that led to this incident, the lessons we’ve learned, and the steps we’re taking as a company to better ensure this doesn’t happen again.
Why is wrong scary?
Risk?
Consequences.
Why us?!
Two factors
Feature Velocity
Striving for Reliability
Feature Velocity vs. Reliability
Good news!
No conflict!
Feature Velocity vs. Reliability
Feature Velocity + Reliability
Figure 2.2: Year over Year Trends: Tempo
Figure 2.3: Year over Year Trends: Stability
But ... *Microservices*!?
What about tests? gates? pipelines? isolation?
We’re covered...
A Story...
And it gets worse...
Dark Debt...
Reminder that “dark debt” *cannot* be anticipated or prevented. It is a natural byproduct of complexity.

I understand this is uncomfortable, especially for engineers. But that is the concept. If it can be prevented, it’s not “dark debt” - it’s something else.

Dark debt is found in complex systems and the anomalies it generates are complex system failures. Dark debt is not recognizable at the time of creation. Its impact is not to foil development but to generate anomalies. It arises from the unforeseen interactions of hardware or software with other parts of the framework. There is no specific countermeasure that can be used against dark debt because invisible until an anomaly reveals its presence.
And over to the business...
“One Hour of Downtime Costs > $100K For 95% of Enterprises”

http://itic-corp.com/blog/2013/07/one-hour-of-downtime-costs-100k-for-95-of-enterprises/
“lost revenue and lost end user productivity”

http://itic-corp.com/blog/2013/07/one-hour-of-downtime-costs-100k-for-95-of-enterprises/
“not take into account the cost of additional penalties for regulatory non-compliance or “good will” gestures made to the organization’s customers and business partners that were negatively impacted by a system or network failure. In fact, these two conditions can cause downtime costs to skyrocket even further.”

http://itic-corp.com/blog/2013/07/one-hour-of-downtime-costs-100k-for-95-of-enterprises/
Bad news...
You’re not covered
Microservices-based systems tend to look like...
“To be fully described, there are many details, not few”

John Allspaw: https://queue.acm.org/detail.cfm?id=2353017
“The rate of change is high; the systems change before a full description (and therefore understanding) can be completed.”

John Allspaw: https://queue.acm.org/detail.cfm?id=2353017
“How components function is partly unknown, as they resonate with each other across varying conditions.”

John Allspaw: https://queue.acm.org/detail.cfm?id=2353017
“Processes are heterogeneous and possibly irregular.”
Reactions?
Ugly Risk Avoidance!
Blame.
A better reaction?
Being wrong is a key software skill
Get Better At Being Wrong™
Make it Safe(r) to be wrong.
Technical Robustness
Zero Blame
Go Beyond Blame
Remember “Dark Debt”
Deliberately Practice
Being Wrong
“prepare for undesirable circumstances” - John Allspaw
Deliberately Practice Being Wrong = Chaos Engineering
Invest in Resilience
Resilience is a Learning Loop
“Normal”
“Normal” → Outage
Outage Detection Diagnosis Fix

“Normal” Detection

Outage
Learning

Outage

Fix
Diagnosis
Detection

“Normal”

Improvement (Robustness)
“Never Let an Outage Go To Waste” - Casey Rosenthal
Post-mortem Learning is Good
Pre-mortem Learning is Better!
“Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system’s capability to withstand turbulent conditions in production.”
- principlesofchaos.org
“Normal”
“Normal”

Game Day /
Automated Chaos Experiment
“Normal”

Game Day / Automated Chaos Experiment

Fix
Diagnosis
Detection
Learning

Improvement (Robustness)

“Normal”

Game Day / Automated Chaos Experiment

Fix

Diagnosis

Detection
We can *learn after* outages...

but it’s even better to *learn* from weaknesses *before* an outage.
Being Wrong can be a super power, if it leads to learning
Establish a Platform for Pre-mortem, Deliberate Practice
“Being Wrong”
Establish a Platform for Pre-mortem, Deliberate Practice

“Chaos Engineering”
Reading Recommendations

- *The Science of DevOps: Accelerate* (Building and Scaling High Performing Technology Organizations) by Nicole Forsgren, Ph.D., Jez Humble, and Gene Kim
Later on today...

Sylvain Hellegouarch
CTO, co-founder

Chaos Toolkit