Customer-centric Metrics
• Twitter - platform/API team during the great de-monolithing

• Nest - led service engineering, launched their developer program

• Turbine Labs - turning practices developed at Twitter/Nest/etc. into products you can buy

• @mccv on the Twitter
Engineering Realities

Ideal Day

Most Days

Build Software

Worry About Software I Built

Build Software
Care vs. Anxiety

- Caring about whether your system is working is the heavy burden of being a grown up software professional
- Caring about things you can’t quantify leads to anxiety
- Caring about too many things leads to anxiety
- Caring about a tractable number of quantifiable things frees you from anxiety
A Twitter Metrics History

- 2009 - Ganglia
  - Open source
  - Limited scalability
  - URL based UI
  - Limited exploration
  - Chart until the system falls over
A Twitter Metrics History

- Early 2010 - Reunion
- Custom built for diagnosis
- Exploration oriented
- Limited scalability
- Only look at what you need to resolve the incident at hand

“it def wasn’t great” @iano, 2009
A Twitter Metrics History

- 2010 - Cuckoo/Viz
  - Custom built for scalability
  - Dashboard oriented
  - Highly scalable
  - Charts for everything
A Twitter Metrics History

- The cardinality of services exploded
- Blindness was replaced with cognitive overload
Switch Focus to Customers, Not System Internals
Simplified View of Service Health

• What do customers care about?
  • Success Rate - I want it to work
  • Latency - I want it to be fast
  • RPS - I want to be able to use this much of the service

• What don’t customers care about?
  • CPU burn, disk utilization, leader elections, etc. etc. etc.
Success Rate

- Stuff obviously needs to work
- Necessary but not sufficient
- “Nines don’t matter if customers aren’t happy” - @mipsytipsy
Request Rate

• A contract about how much service you’re willing to provide

• This is often implicit

• It’s still important
Latency

• Don’t (just) measure medians

• Measure the experience of your worst N% of customers, then make them happy
Putting it All Together

- A simple view of customer happiness
- This is *not* a diagnostic view
- This lets you know whether you *need* to start diagnosing
- More importantly, it quickly lets you know that you *don’t* need to worry
Not All Actions are Equal
Segmentation

- Consider an online shop
  - Customers want to look at stuff
  - Customers want to buy stuff
- Aggregates Hide Information
  - Looks >>> buys
Segmentation

- Break down health by action
- This gives you a clearer view of all the ways customers could be unhappy
- Scan quickly to ensure your service is ok, get back to building features
But I Don’t Serve Customer Traffic
Teammates Become Customers

• Teammates
  • Have a relatively deep understanding of your system
  • Are willing to be flexible with system changes
• Customers
  • Don’t understand the internal details of your system
  • Unwilling to adapt to incidents with your system
Teammates Become Customers

- As teams grow, specialization necessarily becomes the norm
- Treating customers like teammates is going to hurt
A Twitter History

- 2011 - Viz Standards
- Pick good metrics
- Show them for your service
- Measure them for all your dependent services
Embrace Customerhood

• Manage your internal systems like they serve customers
• Strong abstraction boundaries (APIs)
• Measure customer experience, not system internals
Embrace Customerhood

- This means you get to manage your internal dependencies like you’re a customer!
  - You don’t have to worry about what’s behind the API
  - But you can easily observe the *behavior* of the API you’re calling
- Common metrics create a common language for discussing system behavior
- Common language reduces anxiety team wide
Observability the Easy Way
Network Abstraction Boundaries

• Now that everything is behind APIs, everybody is talking over the network

• Proxies make this a programmable abstraction boundary

• Proxies give you a common point of observation and control

• Capabilities span languages and runtimes

• Capabilities can be applied globally
The Network

• Stop thinking of the network as (only) a tax
• Abstraction boundaries are good things
• Programmable abstraction boundaries are better
• Behind that boundary there is freedom to move
• Across those boundaries you can observe and control service interaction
Care Without Anxiety
Todos

• Watch the metrics that make happy customers
• Watch metrics for all customer actions
• Develop a common language for discussing system health
• Treat teammates like customers
• Use your network as a common point of observability and control
@mccv
mark@turbinelabs.io
https://www.turbinelabs.io