OBSERVABILITY

Code telemetry tool kits

Software Engineering + Observability = Success!

Unit tests
Integration tests
Automated load testing
Early validation models

APEX
Application Performance Index

Satisfied count + Tolerating count / 2
Satisfied + tolerated + frustrated counts

Deployment frequency
Lead time to changes
Time to restore services
Change failure rate

SUSTAIN

DEVS

DEVOPS

Bring coders into the process

Observability Driven Development

APPS

Containers/Clusters make it harder for cybersecurity team

CYBER SECURITY

Anomaly detection

IT Resiliency

Feed uptime data to ITSM

Asset autodiscovery

CMDB

IT SERVICE MANAGEMENT

Export data for O11Y

SRE

Focuses on mission-critical code and how it runs in production

BUSINESS UNITS

MRTG

KPI

SERV

O11Y understands key business drivers

UI problems
Time on site
Etc.

Can generate too many or too few alerts

Alerts on symptoms
Alerts on breakages

APM

It takes a village!

OBSERVABILITY

Enable all teams to be observers

Deep monitoring
Broad monitoring
Back-end
End user

Classic APM
Infrastructure Monitoring
Internal Experience Marketing
Digital Experience Marketing

Metrics and monitoring
Will need to match timestamps
May need to be customized

OpenTracing + OpenCensus

OPENTELEMETRY

May only be needed for critical systems

Opensource is free and a good second set of eyes on data

Single vendor needs checks and balances

Opensource may be implemented via vendor APIS

Has vendor grown through acquisitions?

Single vendor database tech is preferable

Robust database system

How well have those techs been integrated

Is vendor FedRAMP compliant?

Schedules workloads
Manages creation and shutdown
Restarts and takes corrective actions
Monitors status of workloads

Developers use:
Continuous delivery (CD)
Continuous integration (CI)
Speeds of delivery

Automate grouping of containers

NETWORK

HARDWARE

KUBERNETES/OPENSHIFT

Cluster

YAML config

KEPTN

Control plane for monitoring cloud

Microservices

Network performance management

Port scans

SOFTWARE

CODE

Automatic instrumentation

Container

Add "counter" lines into code to record times code is used

Automatically insatiate & install monitors

Dockers are containers of code with supporting libraries

Ideate
Create
Release
Operate

Software Delivery

SERVERLESS

Run code without provisioning or managing servers

Do software vendors supply observability data to the client organization?

Vendors may control some of the mission-critical software needed to run business

Vendor lock?

TENANT

Leverage all of IT to help with observability
Apply best-in-class AI to track non-obvious events
Build confidence so IT can deploy more and faster
Use "counter" lines to determine most used code
Observability vendor choice not most critical
Control states greater than states needing control

O11Y can focus observability on most important assets

O11Y brings all of IT together and uses AI to perform deep analysis

More than dashboards
Which metrics are important
Focus on what matters

Find shortcomings before they become bottlenecks

Proactive narrative about overall health

START VENDOR SERVICES

SaaS better than on-prem

TELEMETRY DATA

Act of collecting data from remote systems

Collection of data services across endpoints
Central store for all observability data

DATA FABRIC

Noise reduction
Relevance scaring
Digital fingerprinting
Custom dashboard

Supervised model

Handles large amounts of data
Predict incidents
Reduce false positives
Identify root causes
Perform self-healing
Quickly add new datasets

AUTOMATED REMEDIATION

Automatic ticket creation
Automatic ticket closure

+ HISTOGRAMS

AIOPS

METRICS

EVENTS

LOGS

TRACES

Health & performance

Bucket storage
High compression
Reduce ingestion lag
Encrypt in transit and rest

Vendor APIs

Automatic correlation

Provides the necessary context

Consider bloom filters

Distributed (request) traces
Find slow calls
Find long running operations
Find failing servers

CRITICAL NEED FOR ROBUST AI

END OF VENDOR SERVICES

Causation

DEPENDENCIES

RELATIONSHIPS

CONFIGURATIONS

CODE PROFILES

Runtime of a single operation

Parent-child hierarchies

Performance comparisons after a new deployment

Production-ready profilers

3-5 metrics, no more than 10
Add wrappers at start of coding
Instrument automatically

Changes to expected latency
Changes to error rates
Changes to # of database calls
Changes to # of downstream calls

High data cardinality is critical
Aggregation enemy of variation

Sometimes engineers need to find a needle in a haystack

Data correlation between all parts of the system is critical

Internal and external systems

EXPORT SINK

O11Y can use BigQuery to search raw data as needed

Data lake?

Distributiveness: many nodes in a single operation
Transaction depth: # of calls in a single transaction
Elasticity: system changes as state changes
Metadata variance: attributes that distinguish workloads

ARTIFICIAL INTELLIGENCE

Single-pain-of-glass view contextualized across journeys

And can quickly coordinate with other coders

A coder will detect a problem before anyone else in IT

More likely predict!

It's critical for developers to have visibility into observability outputs

Send daily "health" email to each coder specific to his/her work product

But only outputs for code each specifically built

Include "counter line" metrics in emails

Integrated visibility

Report "footprint" of O11Y data/effort to each business unit

Very unique to each individual company

Google AI
H2O.ai
Amazon Machine Learning
databricks
data iku
IBM Watson
OpenAI
Azure

Multiple models provide checks and balances

MACHINE MODELS

Use best-in-class, cutting-edge AI

NATURAL LANGUAGE PROCESSING

Bleeding-edge tech

Unsupervised and autonomous

Observability/APM vendor AI/ML is not robust

Would be too expensive to go all-in on AI with so many other priorities

FEATURE ENGINEERING

ANOMALY DETECTION

Automatically find exceptions

O11Y TEAM

DEVELOPMENT TEAMS

Release code to production much faster

Rapid debugging

Sustain team also benefits from daily email reporting

Two sets of eyes (sustain & coder) on all potential problems

Code-specific logs in developer IDEs

Very strong relationship

DEVOPS

Guards production (profiling)

SITE RELIABILITY ENGINEERS

APPLICATION SUPPORT

APPLICATION OWNERS

Test in production using feature flags

Break application on purpose

BUSINESS LEADERS

Need to hear good news too!
Systems are very complicated
But are running well each day

O11Y TEAMS

David Gossett
david.gossett [at] informatics.com

Jason McMunn
mcmunn [at] mcmunn.com

Release 1.1
04/18/2022
24" by 84"

Copyright 2022
Informatics
Incorporated