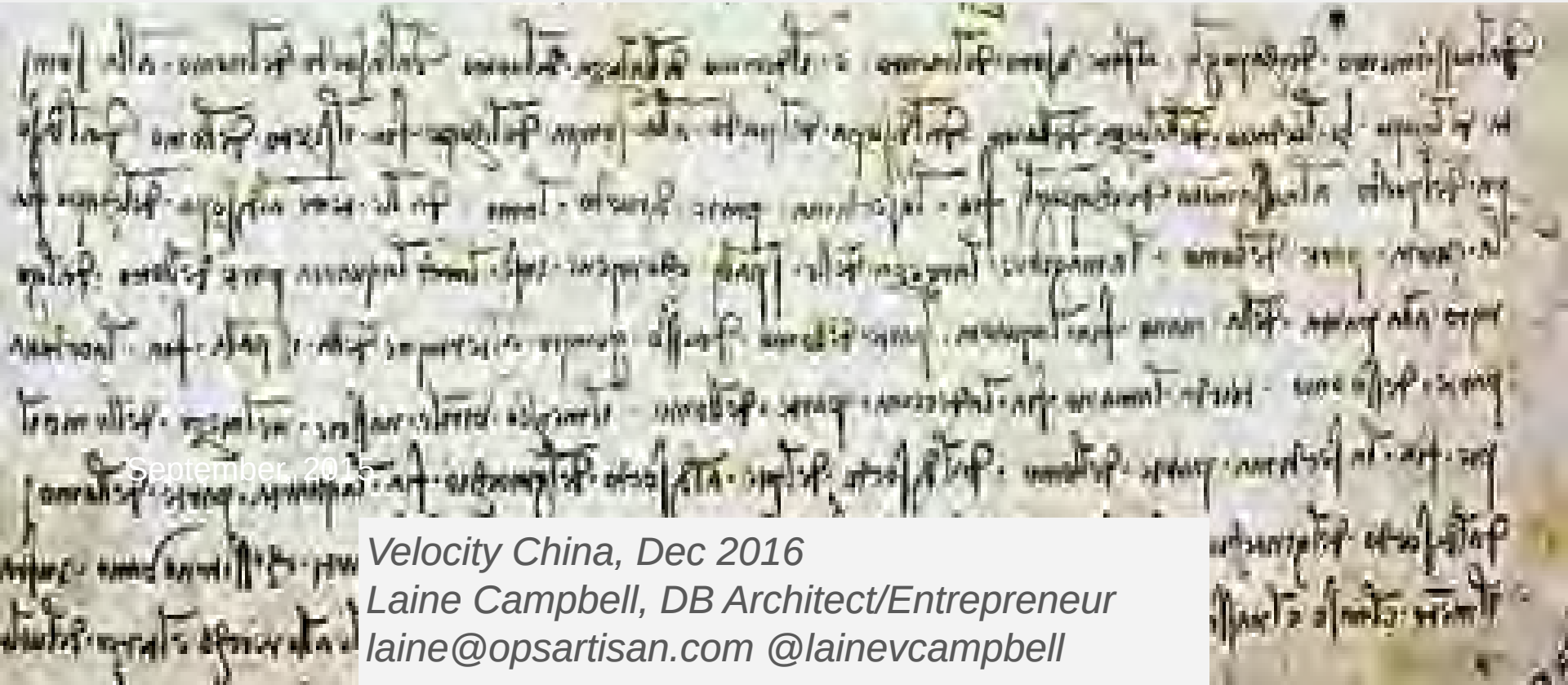
Database Reliability Engineering



September, 2015

Velocity China, Dec 2016

Laine Campbell, DB Architect/Entrepreneur
laine@opsartisan.com @lainevcampbell

engineering, not administration

yesterday's DBA

gatekeeper

master builder

superhero

siloed

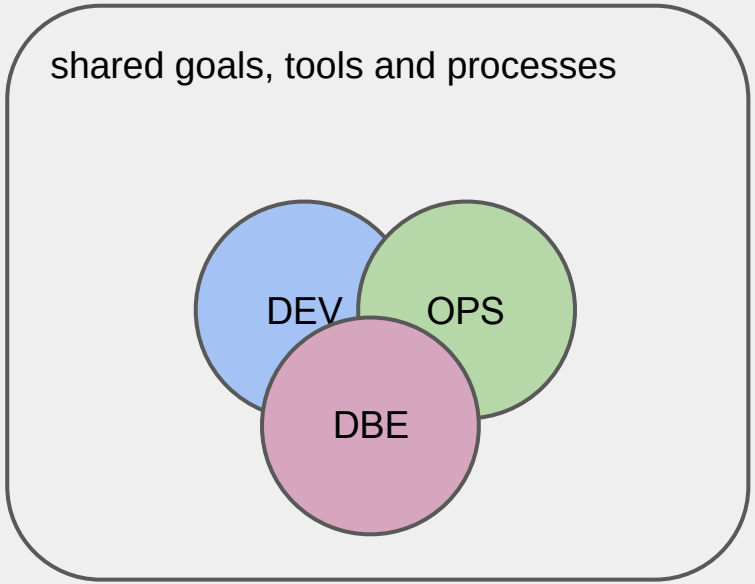
specialized

Reliability Engineering

Ben Treynor, VP of Engineering at Google says the following about reliability engineering:

fundamentally doing work that has historically been done by an operations team, but using engineers with software expertise, and banking on the fact that these engineers are inherently both predisposed to, and have the ability to, substitute automation for human labor.

Database Engineers; O.G. Devops



Database Engineering

Guiding Principles

Protect the Data
Self-service for Scale
Elimination of Toil
Databases are not Special Snowflakes
Eliminate the Barriers between Software and Operations

Protect the Data

- Responsibility of data protection shared by cross-functional teams.
- Standardization and automation with resiliency over expensive/complicated infrastructures.
- Durability and integrity baked into every part of the architecture and software development lifecycle

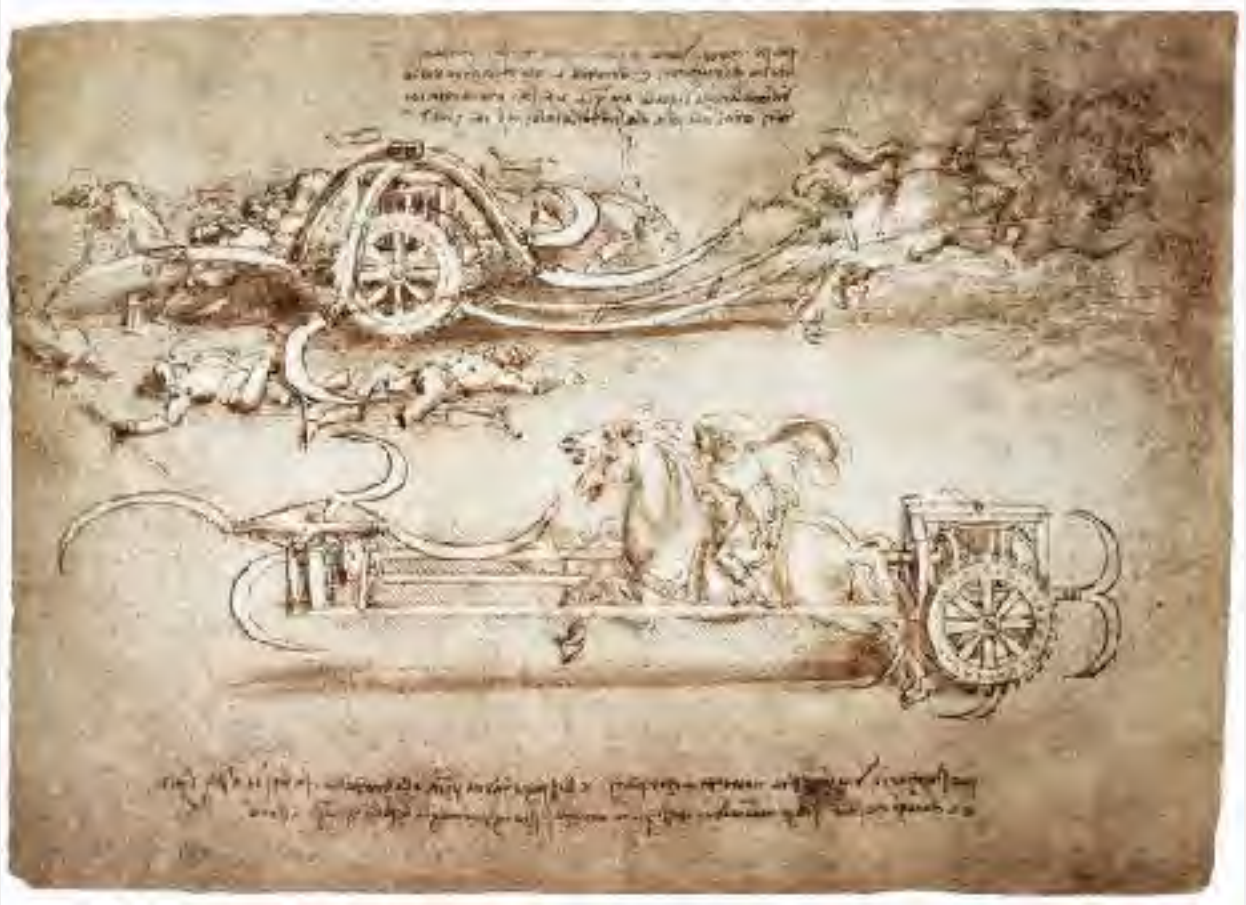
Self Service for Scale

- Metrics knowledge base and automated discovery/collection
- Backup/recovery utilities and APIs auto-deployed for new builds
- Reference architectures and configs for data stores deployment.
- Security standards for data store deployments.
- Safe deployment patterns and tests for database changesets

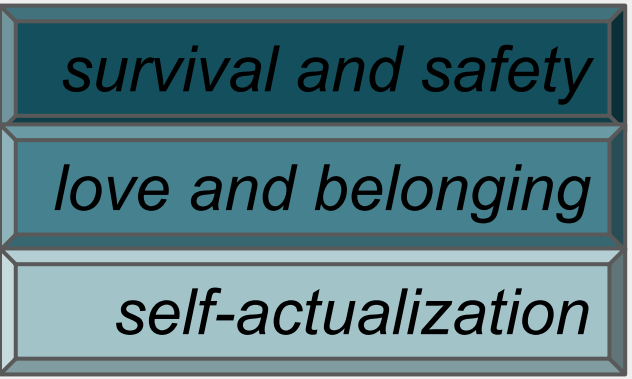
DBs are not Special Snowflakes

- Cattle not Pets, Bill Baker
- DBs are the last holdouts of commoditization

A day in the life



A database's hierarchy of needs

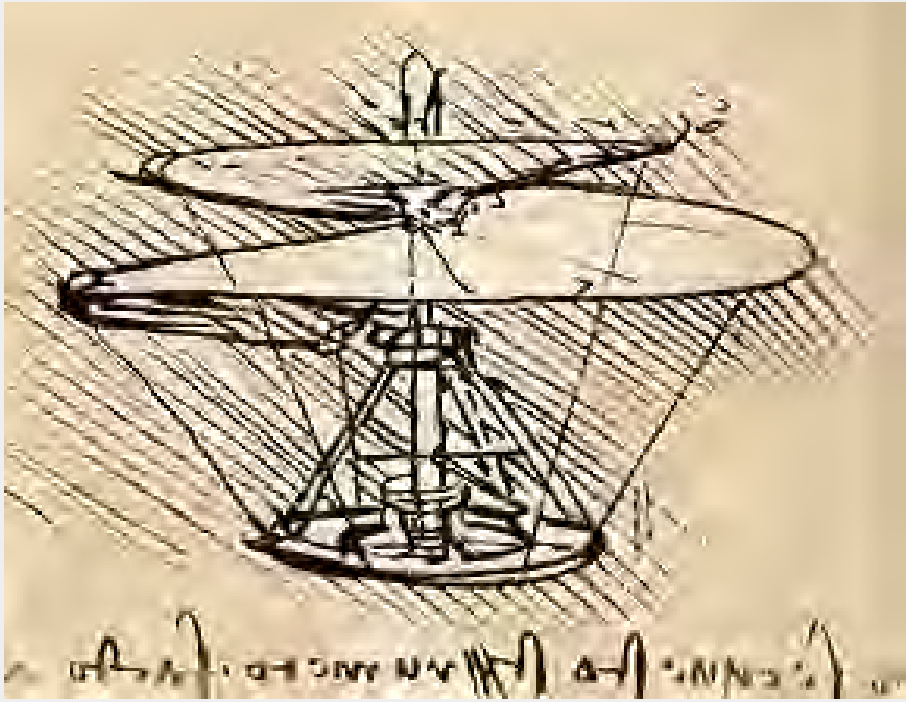


with loving credit and glory to Charity F. Majors

Survival and Safety

- Is your database alive?
- Is your data safe?
- Do you have a scaling plan?

Note: Scaling Patterns



Love and Belonging

- Are DB practices the same as SWE and SRE teams?
- Are SWEs empowered to make/ push DB migrations?
- Is there cross-functional sharing and teamwork?

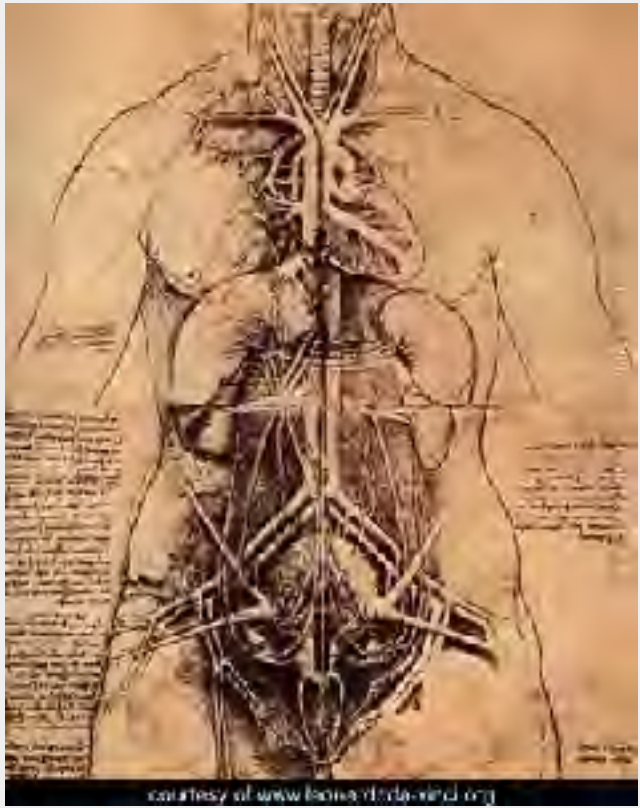
Self-Actualization

- Database workflows accelerate velocity rather than hindering it.
- Developer work is safe and not impacting availability.
- Safe and quiet, databases do not take away excessive resources from other work.

the craft

Ops Core Competencies

- Service Level Mgmt.
- Operational Visibility
- Infrastructure Engineering and Mgmt.
- Release and Change Mgmt.
- Backup and Recovery



Service Level Management

Service Level Objectives

This Stuff is hard!

Social science rather than computation

Drives behaviors for the entire organization

Must reflect customer experience

Service Level Management

Why does this matter to the DBRE?

- This drives instrumentation plans
- Helps understanding of the DB’s role in the greater picture
- This drives your architectural decisions

Service Level Management

Service Level Indicators

- Latency
- Availability
- Throughput
- Durability
- Cost/Efficiency

Note: Latency vs. Response Time

Service Level Management

Defining Service Level Objectives

use distributions over monolithic averages

multimodal workloads require tiered objectives

target resiliency over robustness

consider impacts of percentages rather than whole populations

use your downtime budgets strategically

Service Level Management

Sample mature SLOs - Availability

- 99.9% availability averaged over 1 week.
- No single incident greater than 10.08 minutes.
- Downtime is called if > 5% of users are impacted.
- One annual 4 hour downtime allowed, if:
    - Communicated to users >= 2 weeks ahead of time.
    - Impacts <= 10% of users at a time.

Operational Visibility

What is it for?

Break/fix and Alerting

Performance and Behavior Analysis

Capacity Planning

Debugging and Post-Mortems

Business Analysis

Situational Analysis

Operational Visibility

The New Rules

Metrics and Events are a BI system

Distributed/Ephemeral environments are trending towards the norm

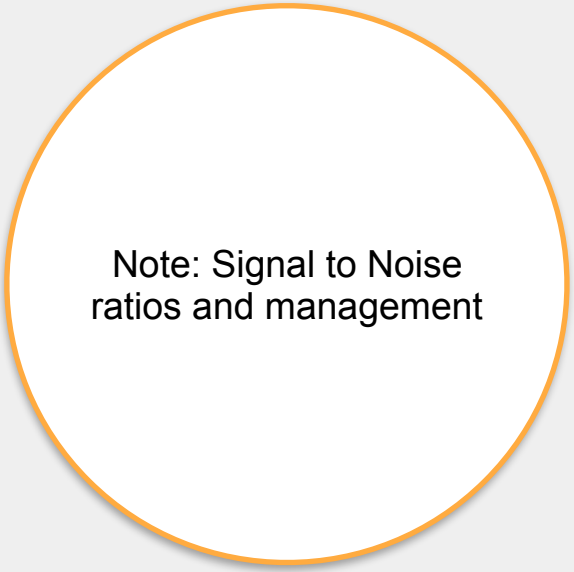
High resolution is becoming standard

Greater opportunity for more noise than signal

Operational Visibility

How does this impact the DBRE?

- Storage for metrics becomes a significant part of your responsibility.
- Abstraction layers for data is required as specific instances/servers go in and out of service.
- Proficiency with analysis of distributions becomes critical.
- Focus on key metrics for alerting is mandatory to avoid pager fatigue.
- Recognizing opportunities for automated remediation becomes a high priority.



Operational Visibility

DBRE Responsibility

Defining metrics and events that must be stored for each persistence store

Educating other teams on analysis of metrics and events data

Ensuring SLOs for monitoring stores are met

Identification of patterns and issues that require depth of knowledge

Operational Visibility

What data do you need?

USE - Utilization/Saturation/Errors - Brendan Gregg

POV - From app (user context) and from inside the DB (details)

- DB Connection Layer Metrics
- DB Internal Metrics
- Database objects Metrics
- Database calls/queries
- Events - logs, external input

Operational Visibility

A day in the life

- Defining and encoding domain knowledge and best practices into the operational visibility stack.
- Advanced forensics and DB Context when problems surpass generalist knowledge.
- Supporting SWE in proper instrumentation and analysis
- Work with SRE on issue identification and resolution playbook and automation.

Infrastructure

The Landscape

Virtual hosts

Cloud computing

Local and remote Storage

Containers

DBaaS

Infrastructure

How does the DBRE fit in?

Abstracted storage requires even more paranoia about data loss

Databases must become cattle

Advances in performance require continued testing and benchmarking of the DBMS

DBaaS reduces toil, allowing for focus on high-value operations

Virtualized Infrastructure

- Data integrity testing becomes more critical.
- Performance characteristics often require horizontal design and scale out.
- Dataset portability becomes a common and significant problem.
- Automation must be utilized as more moving parts are inevitable.
- Unpredictability of latency requires new design profiles for storage access.

Containerized Infrastructure

Data attachment and bootstrapping often makes this model unfeasible

Network and IO needs often impact ability to thrive in shared host models

Excellent tool for prototyping, integration and testing.

Database as a Service

- Reduces toil / work from DBRE resources
- Impacts visibility at OS, network, hardware
- Black box infrastructures create surprises, particularly around durability.
- When used well, DBRE can focus on high value tasks.
- When used without DBRE input, vendor lock-in, lack of forward vision and minimal knowledge of DB internals can impact latency, availability, performance and durability.

Infrastructure Mgmt.

Version control

Componentizing

Building from Configuration Mgmt.

Maintaining configuration

Infrastructure Definition and Orchestration

Service Discovery

Infrastructure

Paradigm Shift: Building

- Decisions on baked images vs. frying at build time
- Maintaining Configuration
    - Idempotent changes: (flexible, drift able)
    - Immutable infrastructures (simple, predictable, recoverable)
    - Enforcement

Infrastructure

Paradigm Shift: Orchestration

- Entire clusters and services in version control
- Great power, great responsibility
- High potential for data loss and integrity issues if mistakes occur.

Infrastructure

Paradigm Shift: Service Catalogs

- Extensive and complex infrastructures outgrow manual, crafted approaches.
- Failover, sharing and state management become responsibilities of the service catalog.

Infrastructure

Day in the Life

- Changes to configuration definitions as required.
    - Testing, integration and deployment
- Launching of new clusters via terraform
- Rolling upgrades of significant changes via service catalog updates.
- Durability and recovery tests of new storage solution from AWS
- Setting up tests in build system with docker for integration

Backup and Recovery

Really recovery…

one of the most crucial processes in the organization

a culture of durability and recovery integrates these processes into everything

backup is merely a means to the end

Backup and Recovery

The New Rules

Dataset portability is king

Potential data loss scenarios are legion, defense in depth is required

Detection becomes critical

Where possible recovery must be automated and used extensively

Backup and Recovery

Considerations

Service level objectives must dictate all choices

Workflows and event driven architectures create dependency webs

Rapid development organizations can change data structures before a recovery need is even discovered

Backup and Recovery

Planned Usage to Exercise Process

New production nodes and clusters

Building different environments

ETL and downstream data stores

Operational tests

Backup and Recovery

Unplanned Scenarios

User Errors

Application Errors

Infrastructure Services

Operating Systems and Hardware Errors

Hardware Failures

Datacenter Failures

Backup and Recovery

Building Blocks of Strategy

Detection

Tiered Storage

A Varied Toolbox

Continuous Testing

Backup and Recovery

A day in the life

- Reviewing recovery tests, comparing to SLO needs
- Working with engineers to build data validation tests
- Working with engineers to review schema evolutions with an eye towards integrity requirements
- Reviewing downstream processes and testing for impact of failure scenarios

Release Management

Software Development

developer support

integration and testing

deployment

Release Management

The New Rules

DBREs must become enablers, not gatekeepers

Engineer training and collaboration is critical

DBREs should be intervening during challenging migrations and large impact decisions

Integration, build and deploy become part of the regular SWE lifecycle and toolset

Release Management

SWE Education

Become a funnel

Foster conversations

Domain specific knowledge

Collaboration

Release Management

Integration Prerequisites

Version Control System

Database Build Automation

Test Data

Database Migrations and Packaging

CI Server and Test Framework

Release Management

Deployment

Migrations and Versioning

Impact Analysis

Migration Patterns

Manual and Automated Deployments

Release Management

A day in the life

Brown bag sessions and knowledge sharing of new features, CVEs and benchmarks/use cases

Reviewing last day’s tests and commits

Updating deployment patterns for SWEs

Migration planning for a rolling data change

building bridges

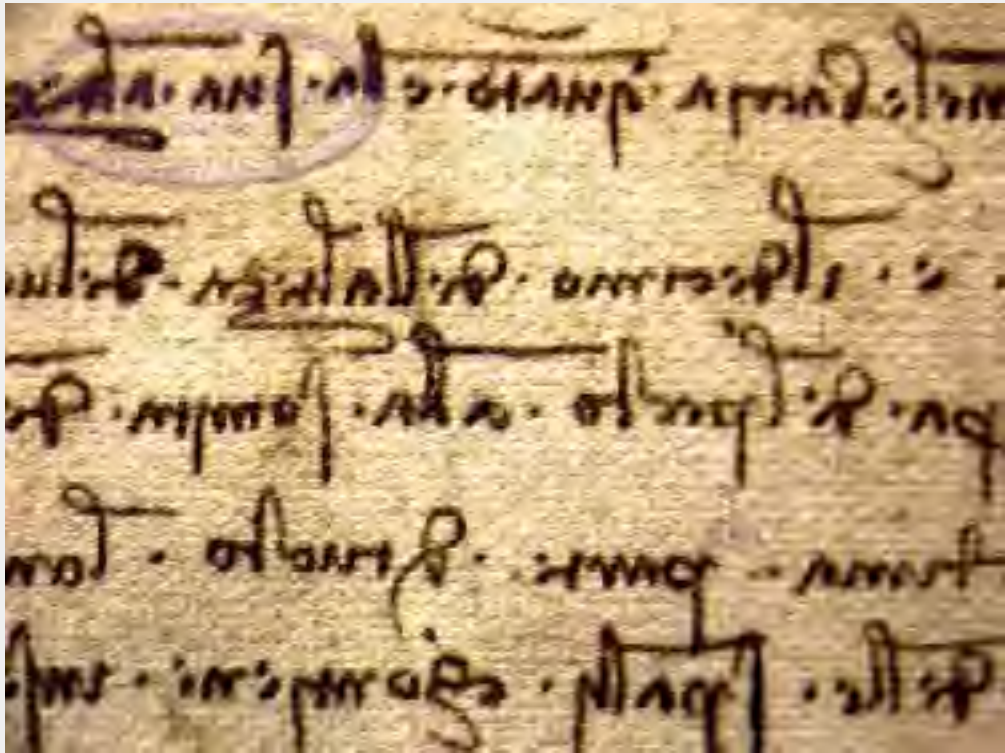
software engineering

bring DBREs into your development processes

integrate DBREs with software versioning system

teach DBREs the testing frameworks

DBREs study the language, the framework, the drivers and the ORMs



systems engineering

Collaborate on infrastructure and operating systems standards

Integrate DBREs with config mgmt. and orchestration

Automate recovery and teach to everyone

Teach SREs about DB forensics and repair

further deep dives

understand the statistics and math around distributions, anomaly detection and correlation

write and push code!

answer the customer service phones

dive into your network layers

teach everyone about the data