

# Unlocking Developer Productivity and Happiness



**Kelly Hirano**  
Director of Engineering  
[hirano@meta.com](mailto:hirano@meta.com)



**Akshay Patel**  
Engineering Manager  
[akshaypatel@meta.com](mailto:akshaypatel@meta.com)

## Who is DevInfra?

*Create a new era of engineering that amplifies developer creativity.*

We are part of Infra and build developer tools and systems. Meta invested early in this area and is a core value to the company and our engineering culture.

# Developer Productivity: Challenges and Assets

## Scale and Diversity

- 10k's developers
  - Over a dozen programming languages
  - Globally distributed, WFH and office
  - 100+ device types
  - Invisibly scaling to billions of users
- 100k changes/day, continuous deployment
- Billions lines of code
- Targets: Android, iOS, www, ASIC, firmware, AR/VR devices, AI/ML, Research

*Engineers want the speed and ease of a startup at Meta's scale*

# DevInfra Components and People

## The Relatable

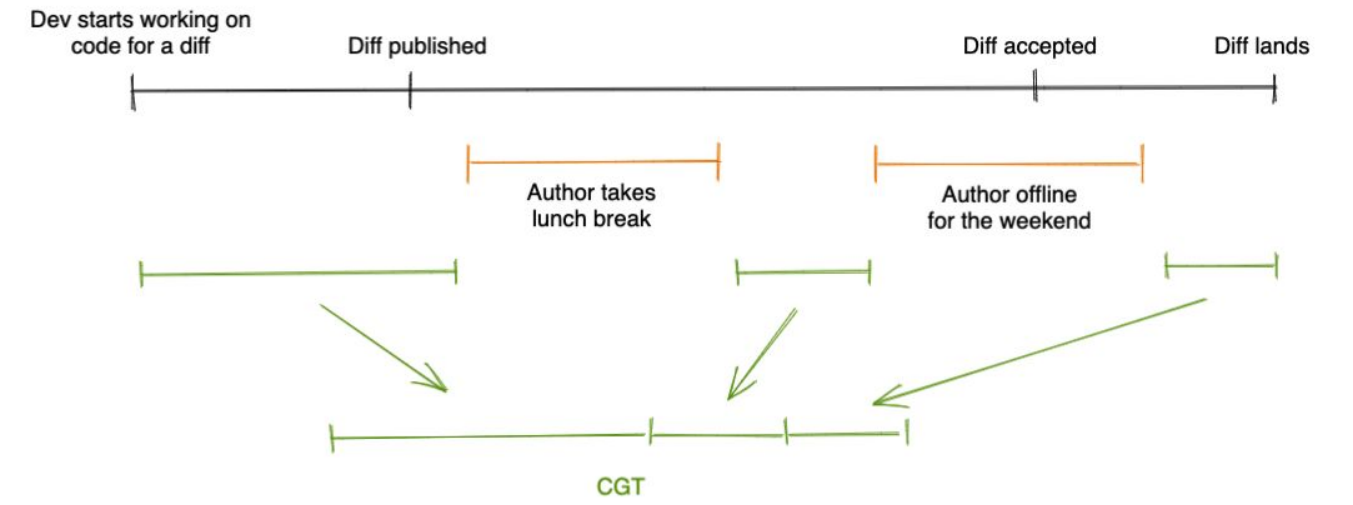
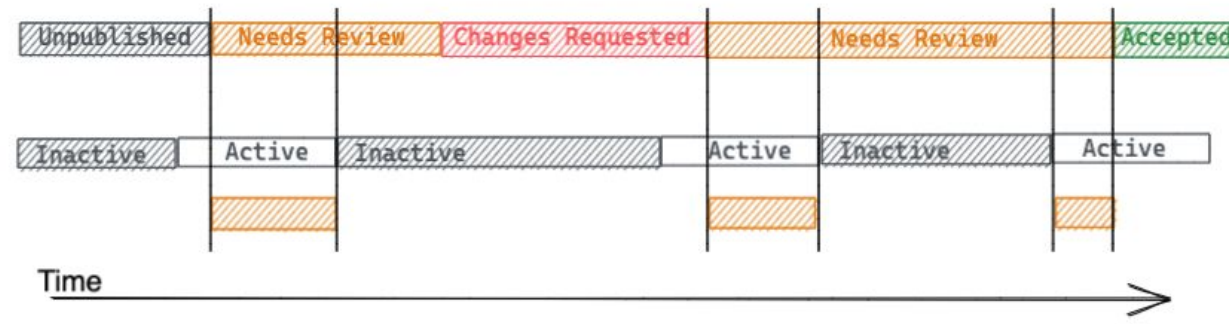
- Task Management
- Source Control
- Code Authoring
- Code Review
- Build
- Test
- CI
- Outage Remediation
- Sentiment Survey

## The Less Common

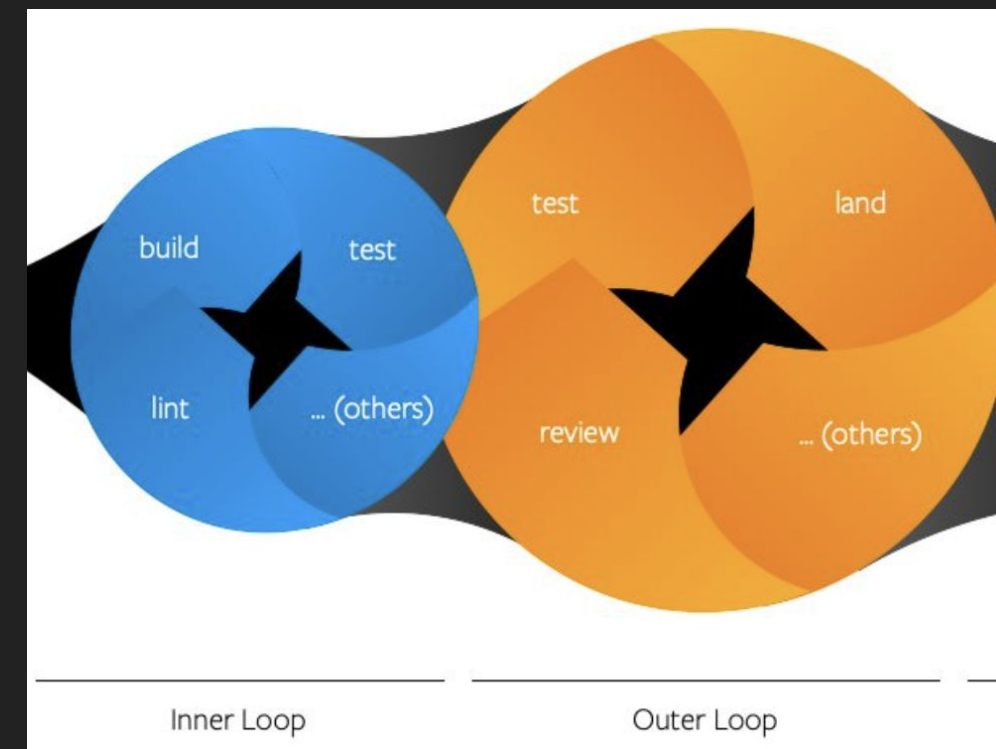
- Data Scientists, Data Engineers
- User Researchers, Designers, PMs
- Research SWEs, MLE/MLR (GenAI)
- Language Developers
- Compiler, Compression Experts
- Productivity Insights
- Impact-driven prioritization

**Where to start?**

Phabricator Status



	E2ECT	UCT
Calculation	Inclusive, wall-clock time	Exclusive, summation of time spent by subprocesses
Useful for	Tracking team work rates over time and across changes	Tracking performance of tool chains or other pipelines
Proxy for	Team productivity	Tooling costs on engineering time



**Without a common way to describe the problem,  
we can't effectively talk about it, let alone solve it.**

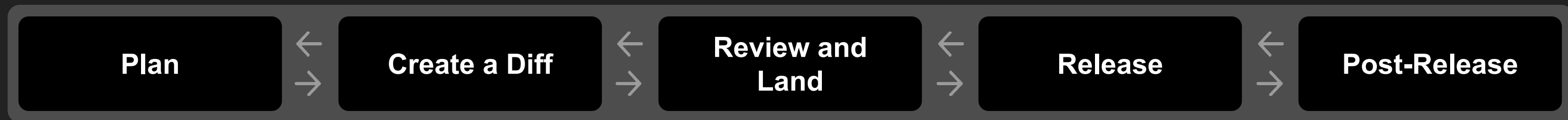


# Getting to Canonical

# Developer Journey: High Altitude



# Developer Journey: Medium Altitude



Coordinating & Managing Work
Tasks
Team Backlogs
Calendar
Technical Design
Code Browsing

Code Authoring
Dev Env
IDE
Source Control
Code Verification
Test
Debuggers
Incremental Builds

Code Review
Phabricator
Continuous Integration/ Land
Integration Test Build
CI/Test
Land

Release
LLVM, Redex, etc
Health Validation
Phased rollout
Release Builds
App Stores distro

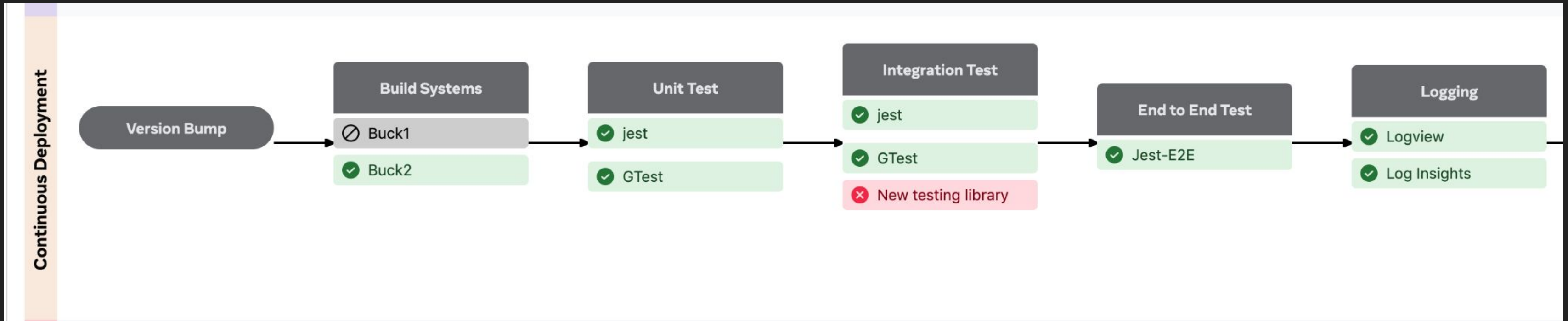
Monitor, Investigate, Mitigate
Health: Crashes, Scroll Perf, etc
Detection
Investigation
Mitigation
Dashboarding

# Supported Workflows

Dev Env  
Code  
Test and Debug  
Local Preview

⋮

✔ High Quality	Feature complete and provides a good experience to the majority of users. Has an internal team to provide timely support, and has metrics associated with it.
⚠ Moderate Quality	Works but may have feature gaps, usability issues, and/or lack an internal support team.
✖ Needs Attention	Does not perform basic expected operations, and/or is planned to for further engineering investment.
⊘ Deprecated	Discouraged from use. It has been superseded or is no longer considered efficient or safe.
❓ Not Sure	Not yet evaluated for a quality score



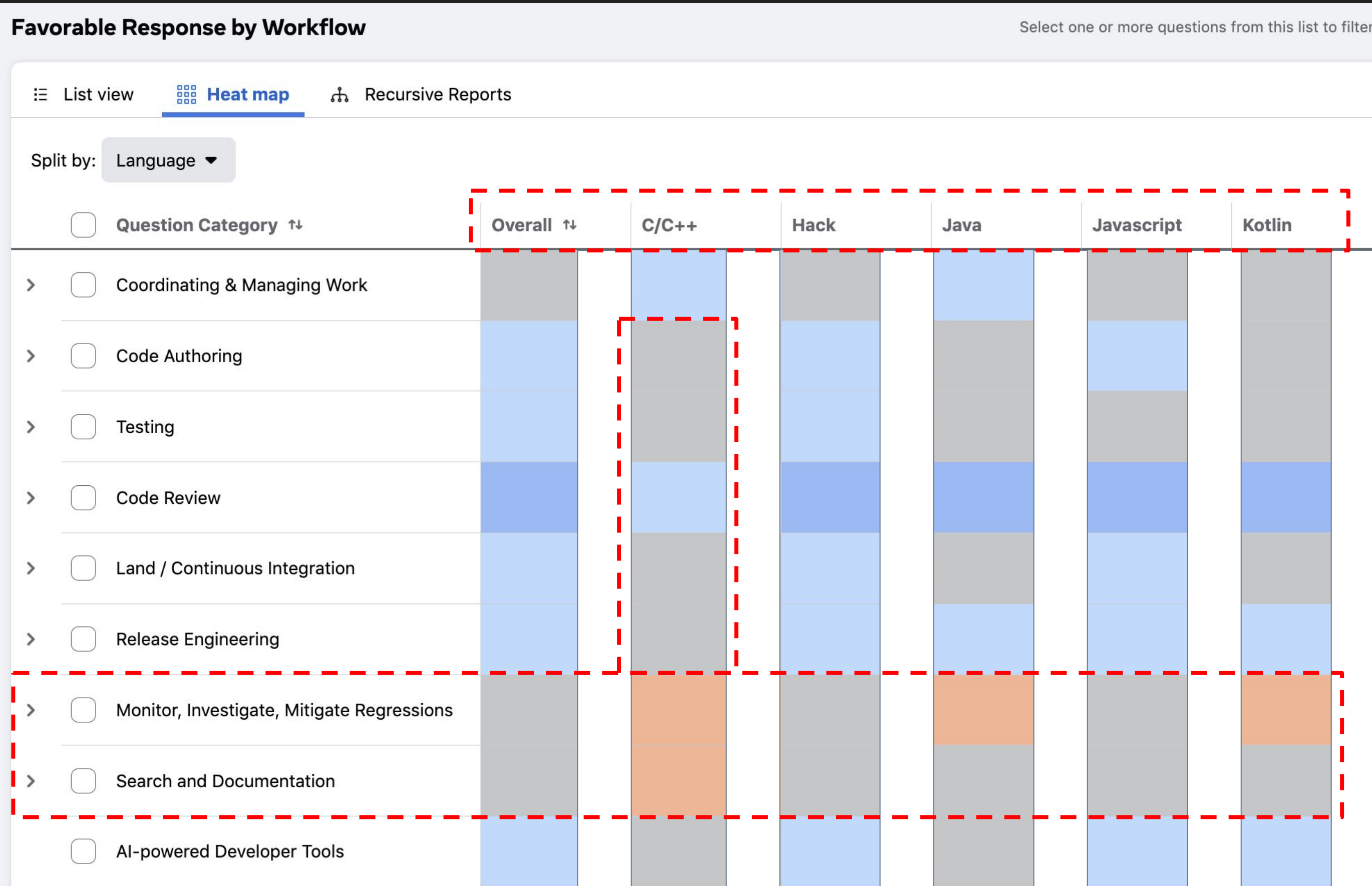
⋮

Release

# Cohorts and Data

**We can't solve everyone's problems**

# Survey Data



**Self-service splits by:**

- Repo
- Language
- IDE
- Ecosystem
- Tenure
- IC Level

# Survey: Low C++ Code Authoring Sentiment

*“In the last 3 months, how satisfied or dissatisfied have you been with each of the following aspects of your experience authoring code in C++ using VSCode?”*

- Tools are fast
- Tools are reliable
- Tools have the right features to help you accomplish what you are trying to do
- Available libraries and frameworks for this language are comprehensive and ergonomic

**C++ was worst across all languages**



% of C++ files successfully opened with fully functional language services

P90 time it take for C++ files to open and have all language services ready

% of C++ code navigation actions give accurate results against 100 top used files opened by devs every day

% Find All References weighted consistency for C++

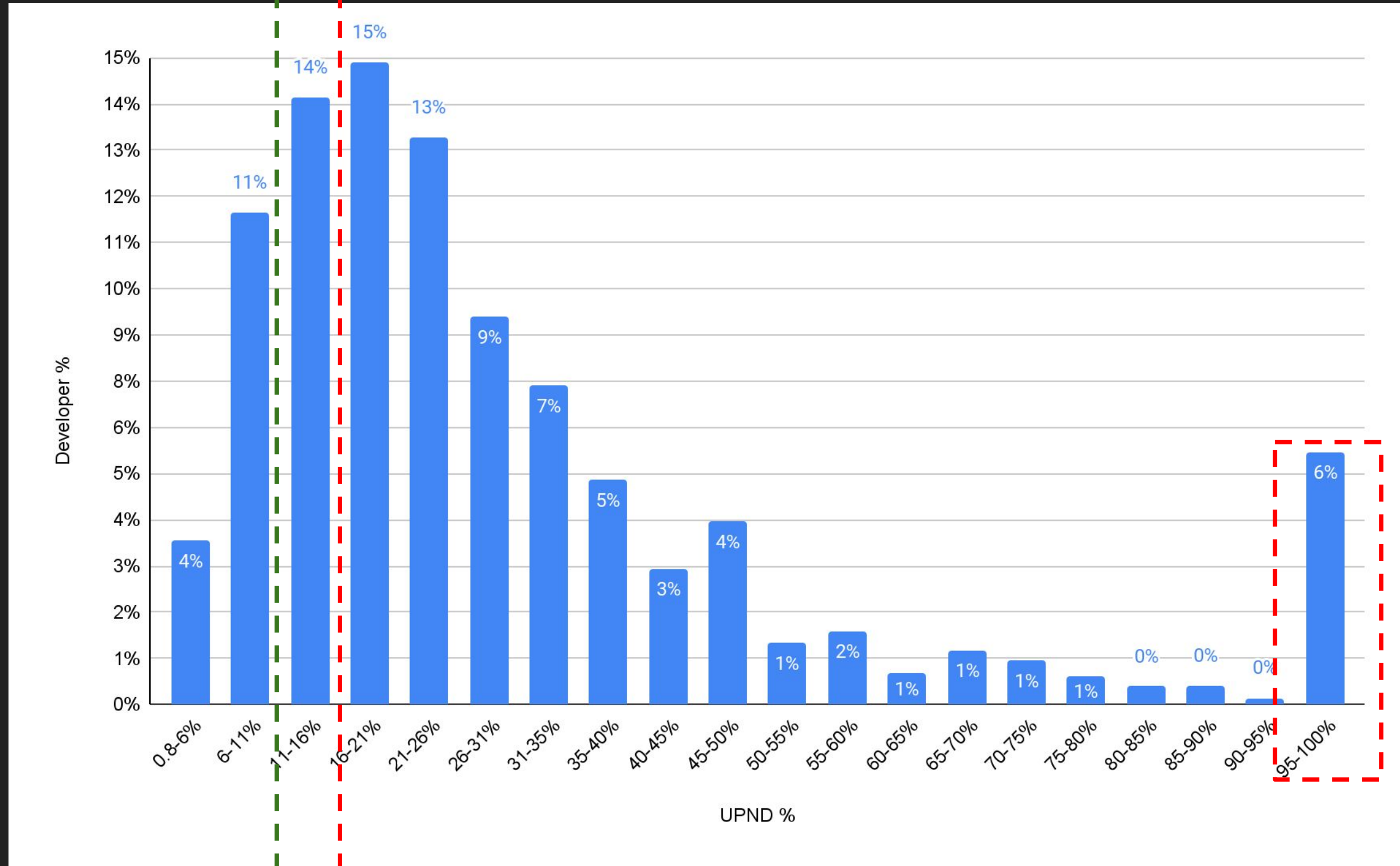


**Come up with a metric that actually represents the user pain and drive it down.**

# Empathy Metric: User Perceived Navigation Disruption

Good

Bad, above this negative feedback increases

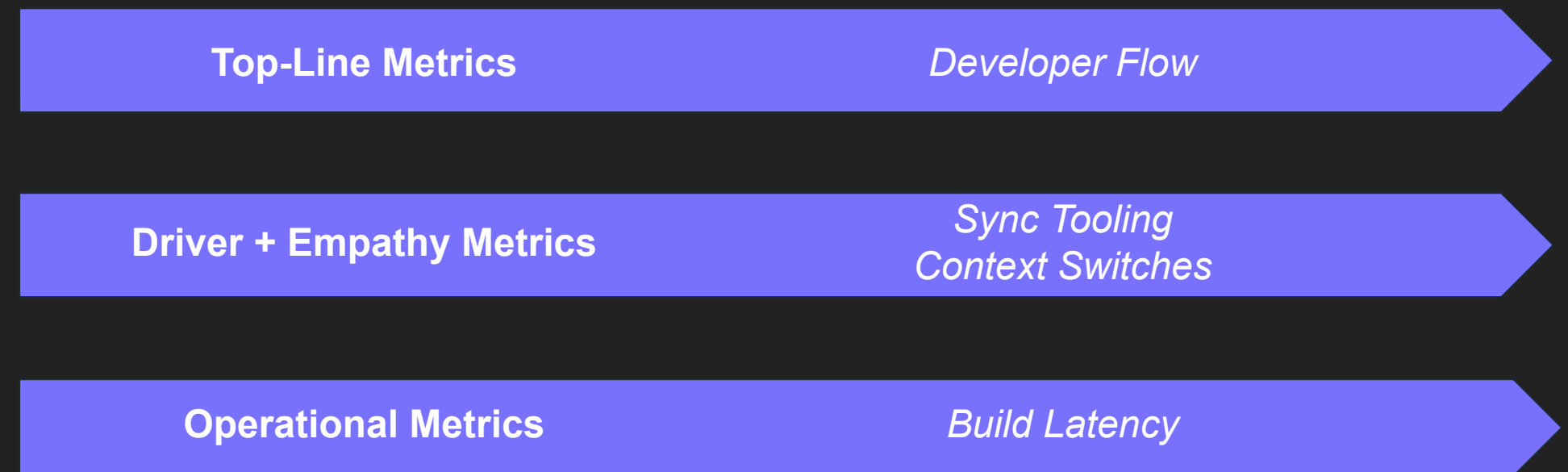
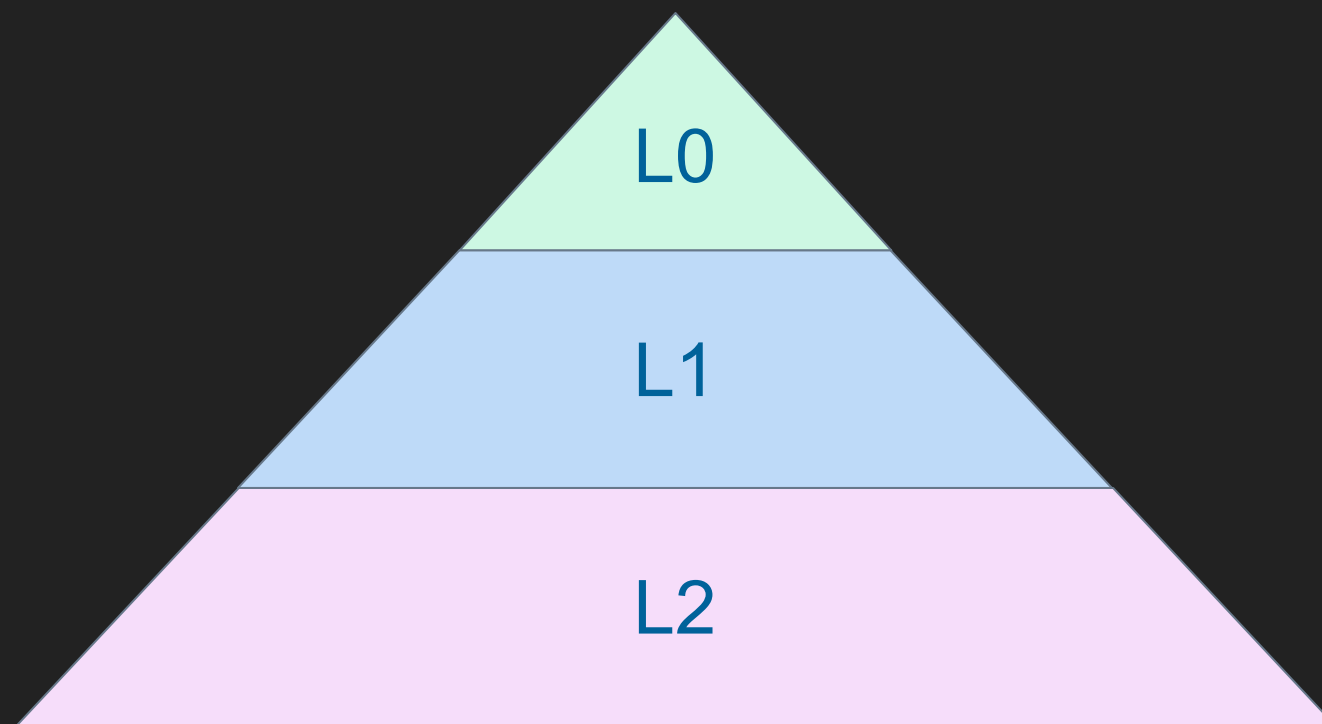


# Measurement and Metrics

**Engineers**

**Data Science**

# Measurement Principles



# Our Framework

## Goaling

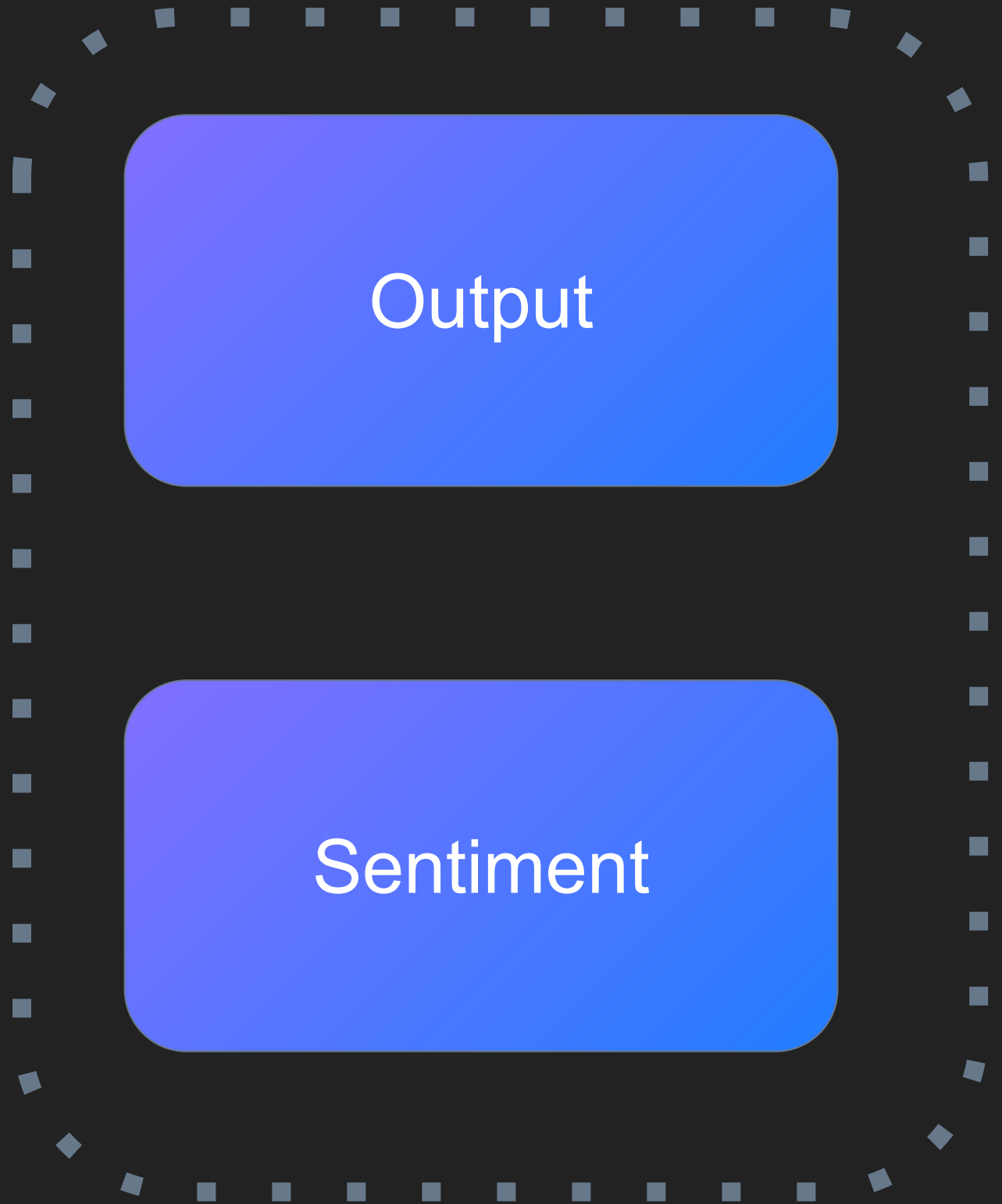
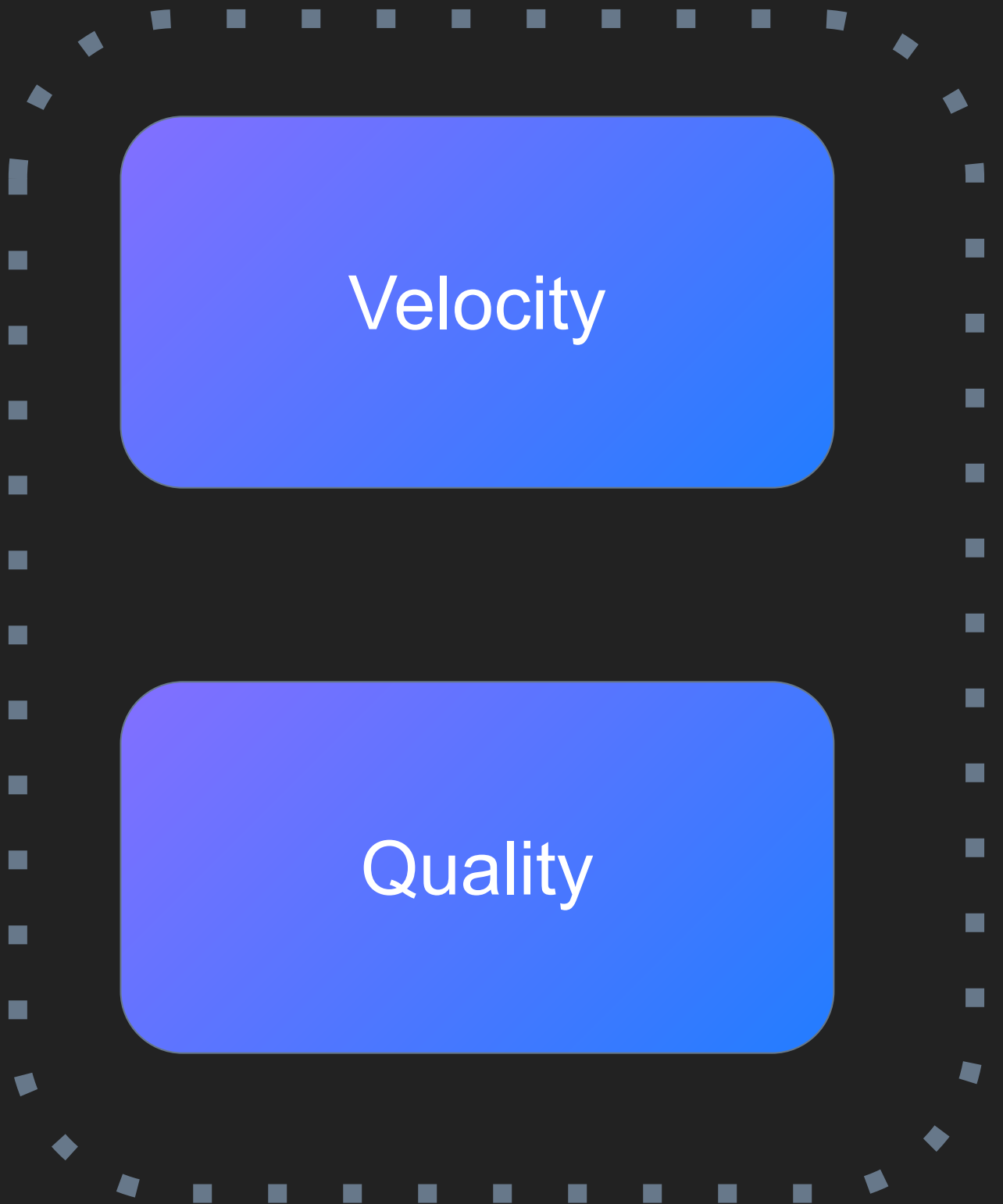
Velocity

Quality

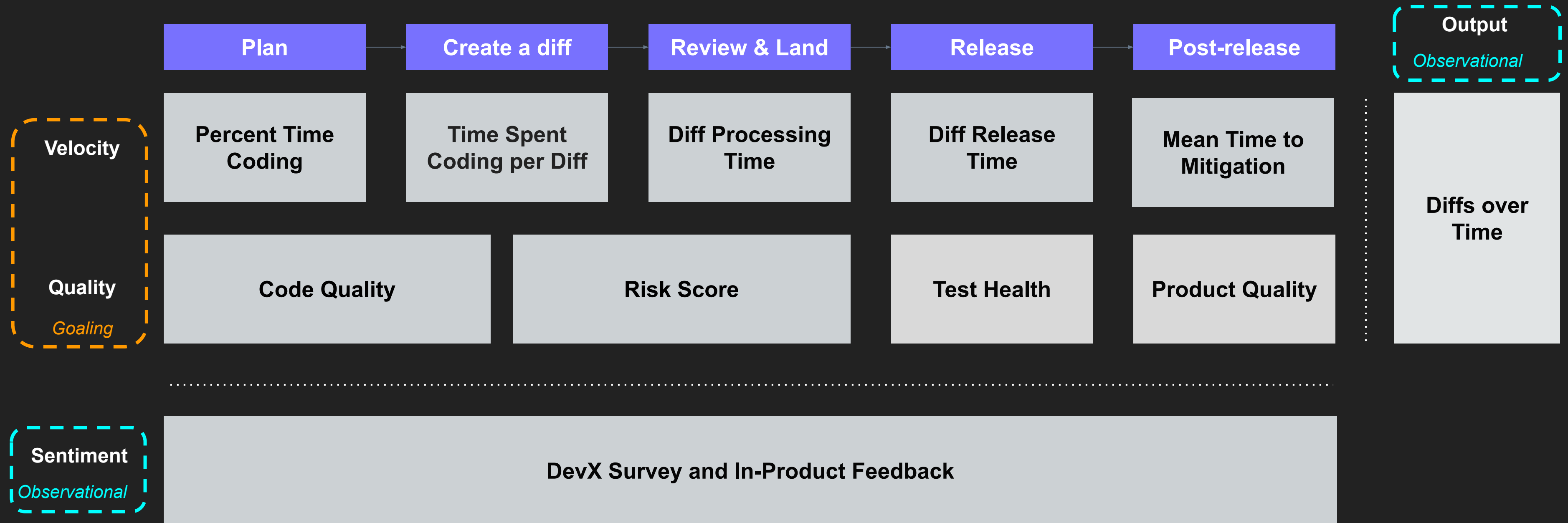
## Observational

Output

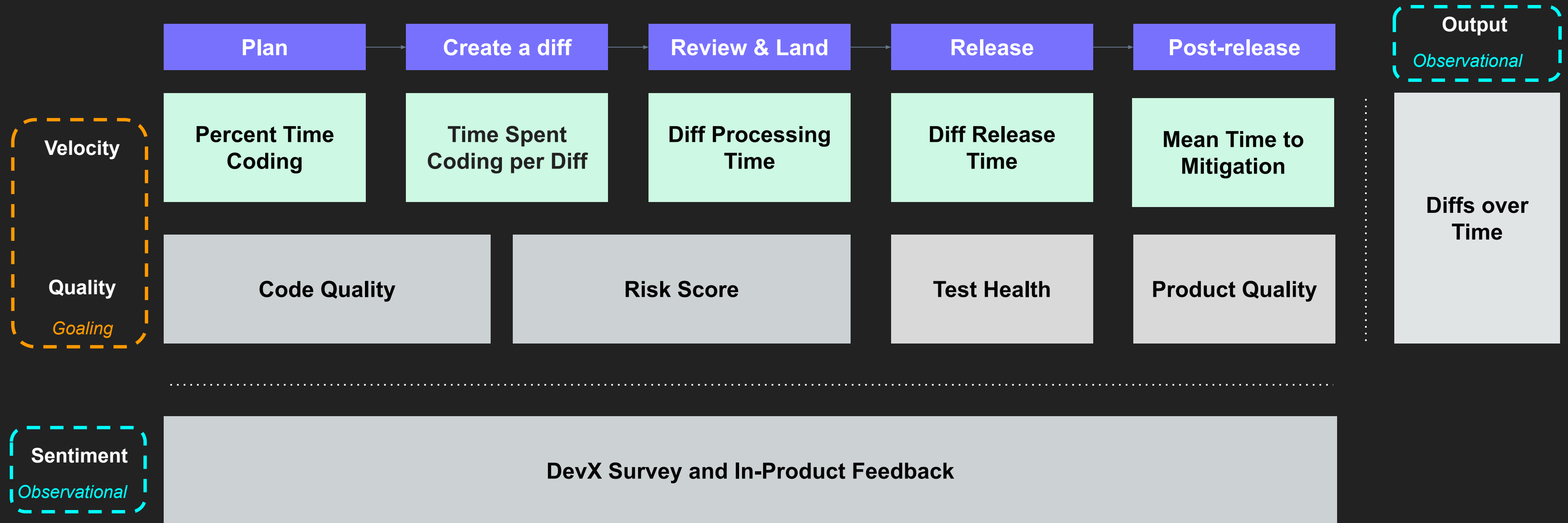
Sentiment



# Productivity Framework

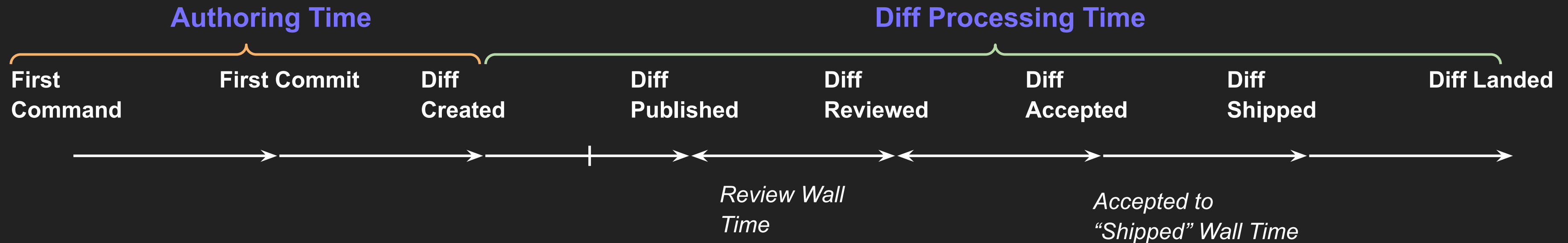


# Velocity



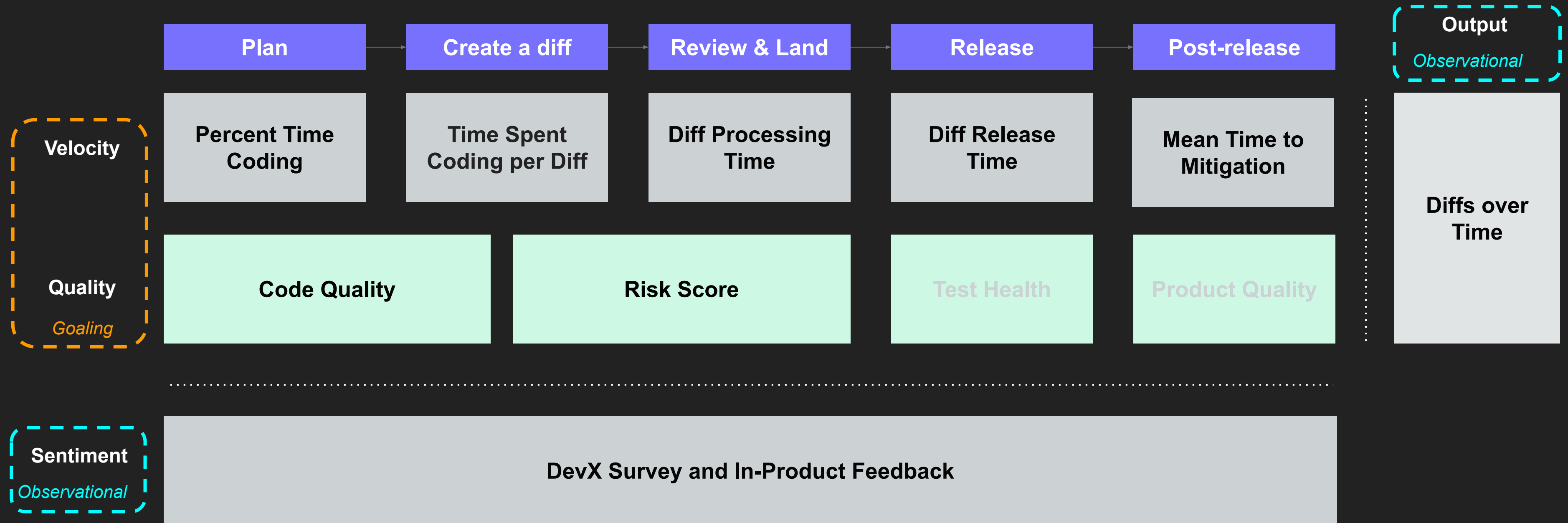


# Velocity



- **Diff Processing Time** is a wall time metric based on the timestamp of a diff, from created to published to reviewed to accepted to landed
- **Authoring Time** captures the engineering time spent per diff that can be further broken down into components such as time spent in IDE vs knowledge acquisition

# Quality



# Quality

## Code Quality Score


- Holistic score to capture health of codebase at rest
- Every signal is validated against velocity and outage prevention


## Sample Signals

- Modern languages *Swift or ObjC*
- Modern frameworks *legacy APIs*
- Dead code *stale experiments*
- Code complexity *code branching*
- Modularity *large dependency graphs*
- Test health *flaky tests*
- Documentation *undocumented APIs*

# Risk Awareness Score

- How likely is it that this change will cause an outage?
- Uses cutting edge LLM's to build predictive models for risk
- Provides engineers a prioritization mechanism to refactor + test code
- Dual benefit of landing low-risk code during code freezes as well as prevent high-risk code from getting in


 Lightning Talk




## Moving Faster and Reducing Risk

Rui Abreu

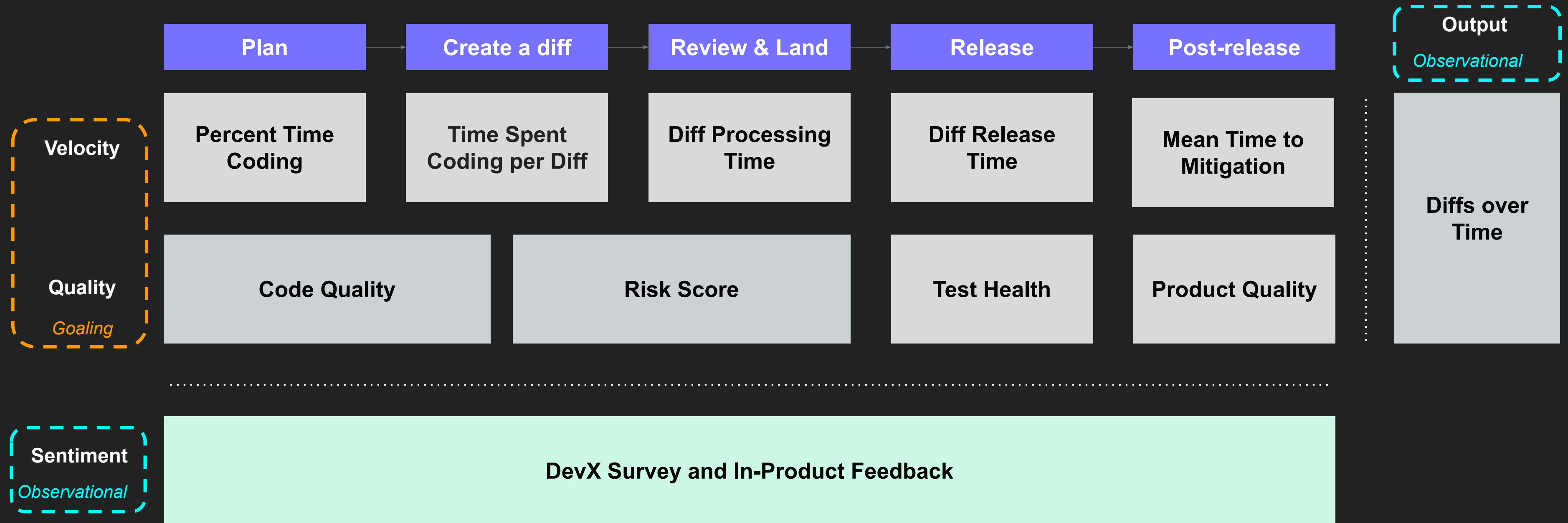
---

Tue, Sept 24 at 4:40pm 

 20mins

This talk discusses the challenge of determining what should be released in large-scale software development, such as at Meta's scale. To address this, we developed models to determine the risk of a pull request (diff) causing an outage (aka SEV).

# Sentiment



# Sentiment

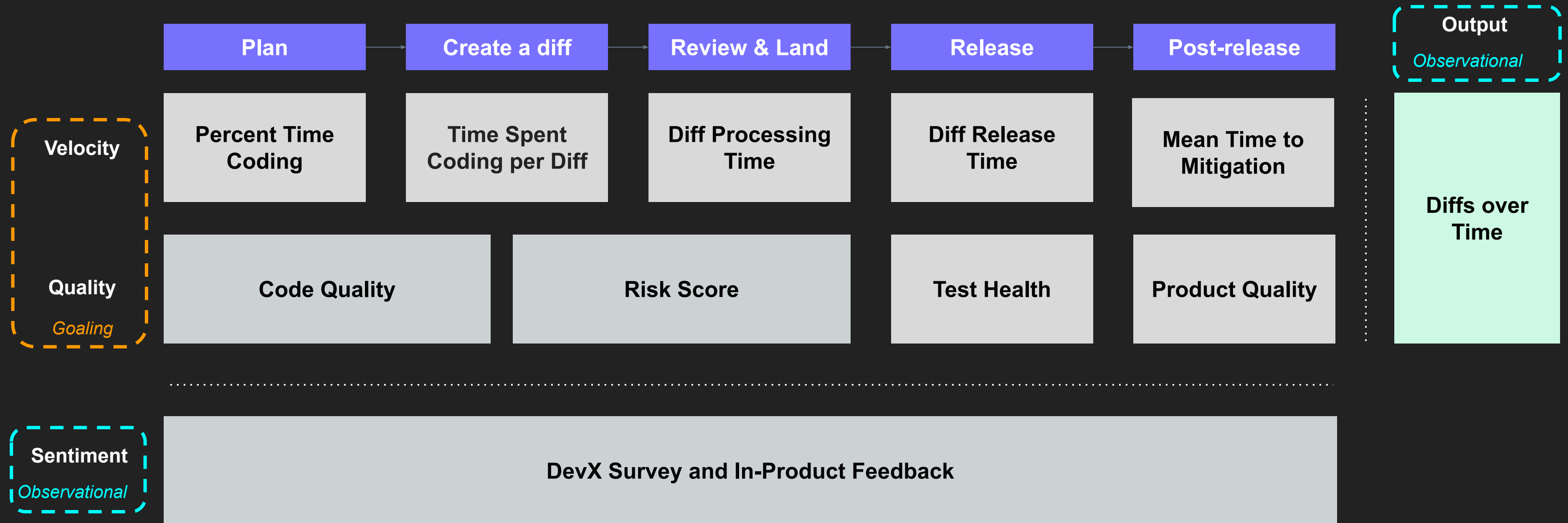
## The Relatable Stuff

- Sentiment Survey across phases of dev journey
- Runs every 6 months
- Self-service cuts for every team
- De-dupe surveys to prevent fatigue

## The Unique Stuff

- Empathy Metrics to predict sentiment where possible to reduce survey length
- e.g., CI Reliability and Focus Time have a strong correlation with sentiment for each area
- Support load for engineers

# Output



# Output

- Number of changes (diffs) each developer lands into the codebase
- Not generally useful for teams to goal on but helps identify systemic trends
- Intuition typically points to levers like Build and CI which are not large enough movers at scale
- Data Science analysis lets us identify behavioral drivers based on output

---

**Executing Roadmaps**

- 1a. Technical Managers**
- 1b. Review Time**
- 1c. Time to Ship**

---

**Setting Team Norms**

- 2a. Project Management**
- 2b. Focus Time**

---

**Building Efficient Teams**

- 3a. Prolific Coder and Reviewers**
- 3b. Fast Ramp Ups for New Hires**



# Takeaways

# Takeaways

- Get to a canonical version of the world
- Intentionally prioritize user cohorts
- Ladder metrics to help teams focus on what they can drive
- Have goaling and observational metrics that fit your business





**Kelly Hirano**

Director of Engineering  
hirano@meta.com

**Thanks**



**Akshay Patel**

Engineering Manager  
akshaypatel@meta.com