CONF42

SEP 5, 2024

CONF42 PLATFORM ENGINEERING

AI AUGMENTED PLATFORM ENGINEERING

brillio

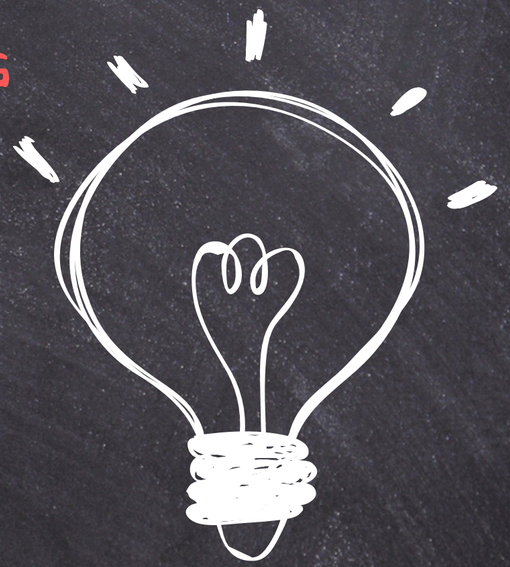
AJAY CHANKRAMATH

CHIEF TECHNOLOGY OFFICER / MANAGING DIRECTOR
PLATFORMS & PRODUCTS
BRILLIO

BRILLIO.COM | CHANKRAMATH.COM

INTRODUCTION

EMPOWERING INDUSTRY-SPECIFIC PLATFORM ENGINEERING THROUGH AI AND LLMS FOR UNPARALLELED EFFICIENCY, RELEVANCE, AND SCALABILITY



- Improving the way platform engineering works
- Predictive & Generative AI
- Use of LLMs and why EPOCHs matter
- Applying RAGs - A Case Study in BFSI domain
- How to get started on the AI journey in PE?

WHY?

METRICS THEMSELVES DOES NOT CHANGE
LLMs enhance each of them!

TIME TO MARKET

Code Generation
Developer Productivity
Streamlining CI/CD
Improve Collaboration
Optimize Infra management
Optimize Path-to-production
Better ephemeral envs
Rapid Prototyping

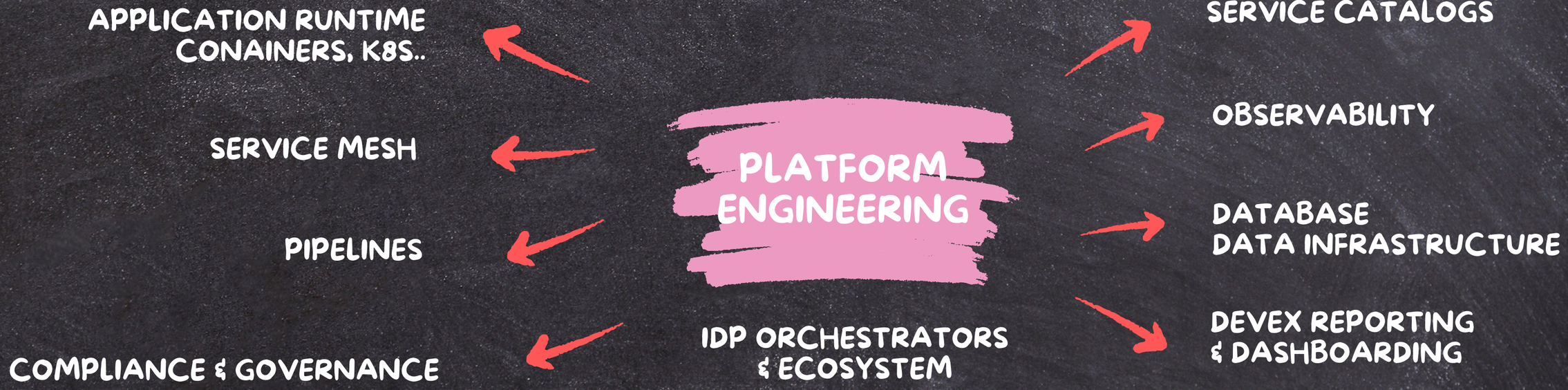
COST REDUCTION

Automate/Reduce repetitive tasks
Improve Utilization of resources
Reducing human error

DEV PRODUCTIVITY

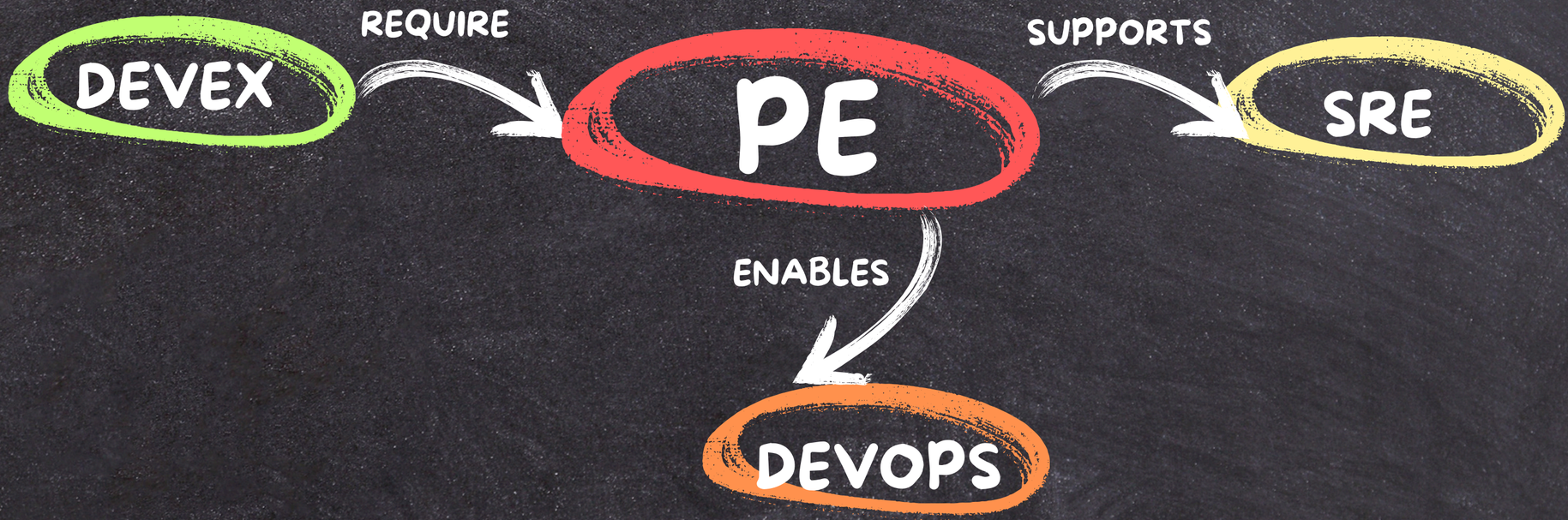
Enhanced error detection
Streamline documentation
Reduced cognitive load

PLATFORM ENGINEERING?

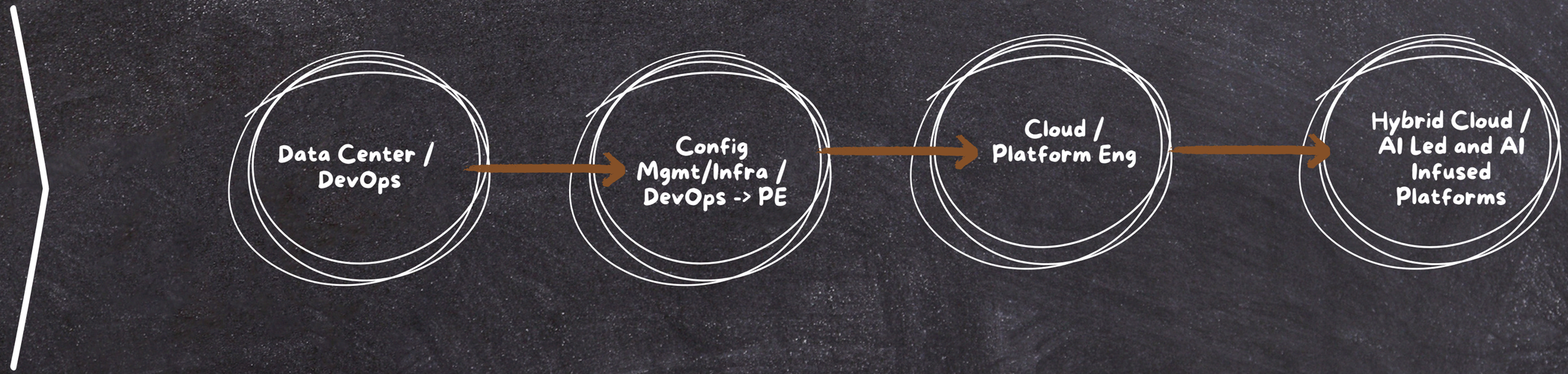


CONF42

IN CONTEXT

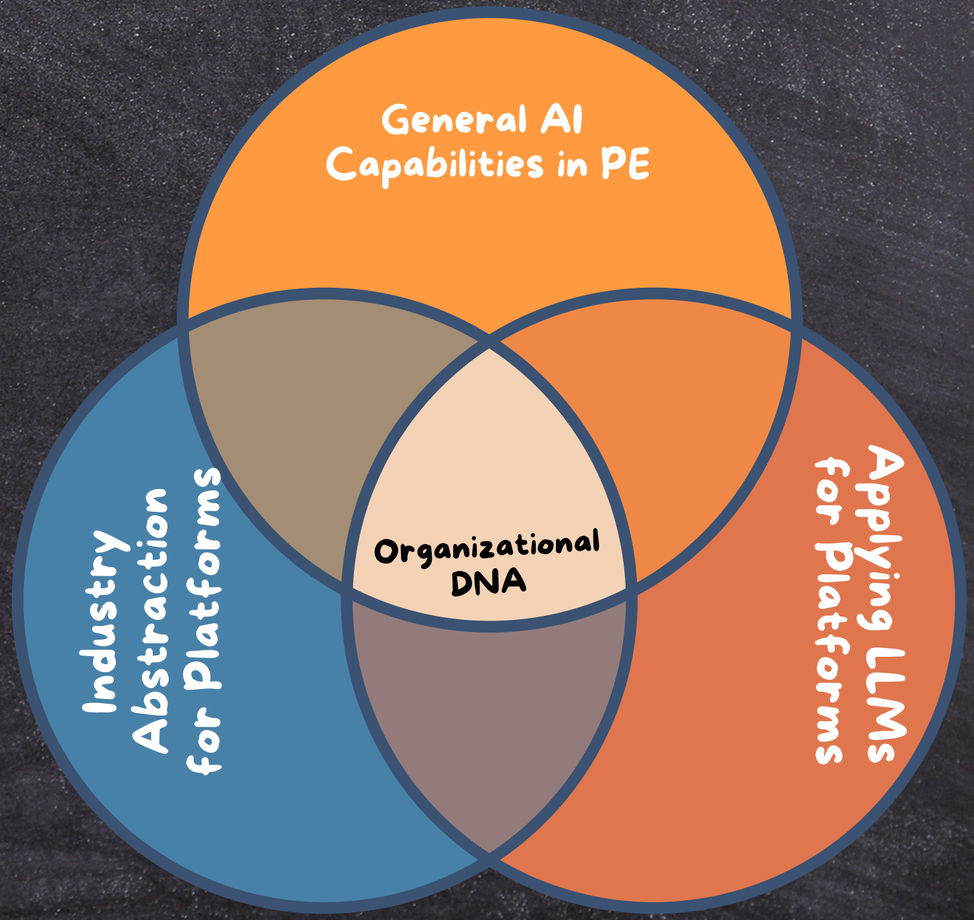


PE EVOLUTION



CONF42

DOMAIN SPECIFIC



CONF42

AI HYPE?

PREDICTIVE

Forecasting

Trend Analysis

Incident Management

Predictive Observability

GENERATIVE

Code Generation

Test Generation

Documentation

Automated Decisions

LLMS

OpenAI GPT

Codex

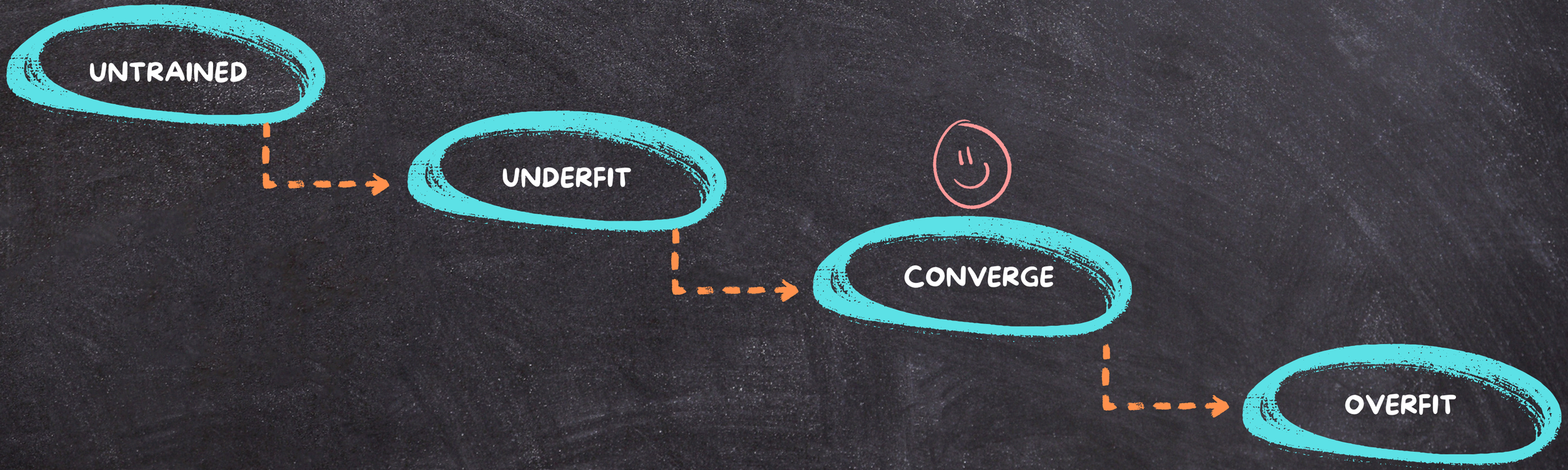
CONF42

LARGE LANGUAGE MODELS

GPT4	Code Generation, Documentation, DevOps Assistant
Codex	Autocompletion, IaC, API Integration
BERT	Log Analysis / Error Detection, Search & Doc retrieval, Knowledge management
T5	Data Transformation, Config Management, Continuous Integration
ROBERTa	Semantic search of repos, knowledge extraction, Chatbots
LLAMA	Chatbots, code reviews, AI powered devex tools

CONF42

TRAINING MODELS & EPOCHS



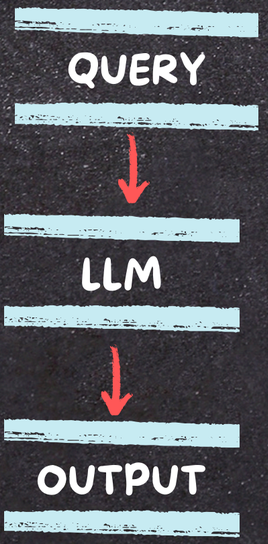
TRAINING LLMS USING EPOCHS

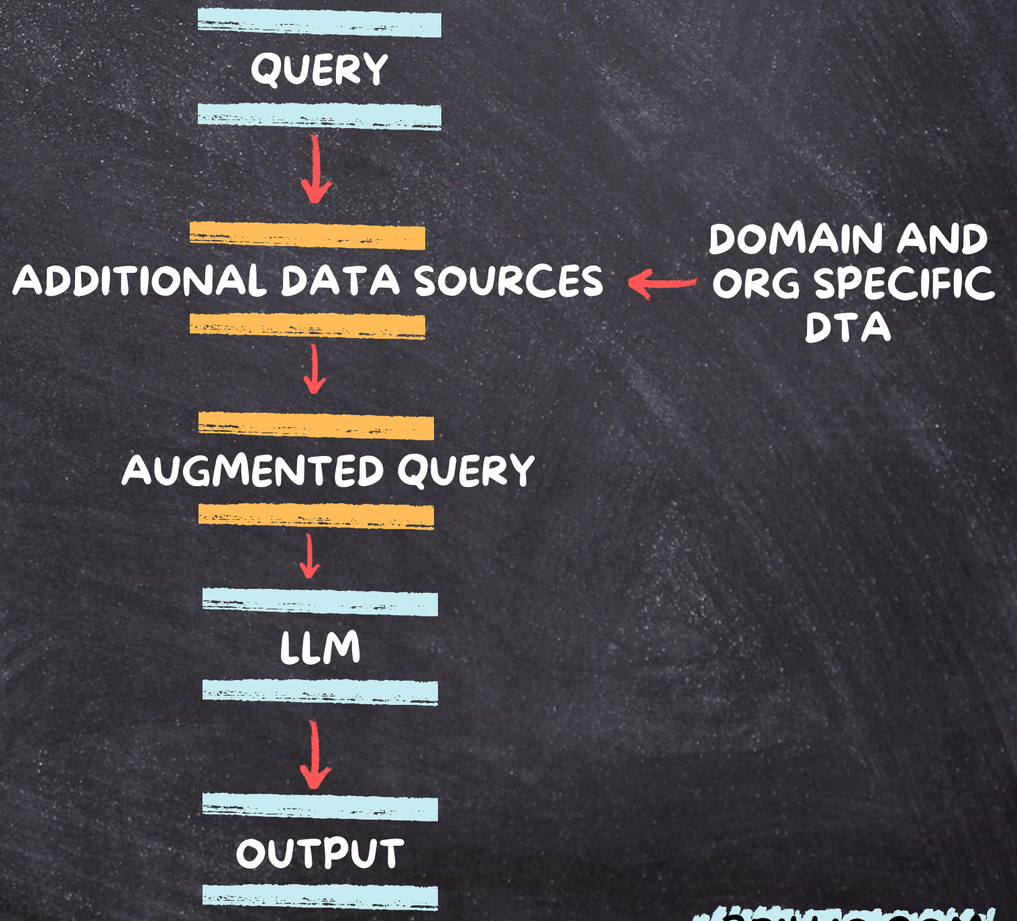
```
initialize_model_and_tokenizer()

set_hyperparameters(epochs, batch_size, learning_rate, max_length)

create_dataset_and_dataloader(texts, tokenizer, max_length, batch_size)

initialize_optimizer(model_parameters, learning_rate)

for epoch in range(epochs):
    for batch in dataloader:
        clear_gradients()

        inputs, attention_masks = prepare_batch(batch)

        outputs = model_forward_pass(inputs, attention_masks, labels=inputs)

        loss = calculate_loss(outputs)

        backpropagate_loss(loss)

        optimizer_step()

        log_training_progress(epoch, batch, loss)

save_trained_model_and_tokenizer()
```

CONF42

RETRIEVAL AUGMENTED GENERATION

RAG BRIDGES THE TRAINING DATA GAPS BY INTEGRATING EXTERNAL KNOWLEDGE BASES, ALLOWING LLMS TO DELIVER MORE RELEVANT, ACCURATE, AND UP-TO-DATE CAPABILITIES IN YOUR PLATFORM





RAG CASE STUDY FOR BFSI DOMAIN

SIMILAR DOMAIN SPECIFIC IMPROVEMENTS CAN BE APPLIED TO ANY BUSINESS DOMAIN

PROBLEM STATEMENT

Enhance the developer experience for BFSI domain developers by integrating traditional platform engineering activities with the RAGs

CONF42

RAG CASE STUDY FOR BFSI DOMAIN

APPROACH

1. Identify critical BFSI APIs
2. Develop domain-specific RAGs >> APIs
3. Create a Platform Abstraction Layer
4. Create a self-service portal for developers
5. Standardize development environments
6. Setup e2e Observability
7. Automate CI/CD Pipelines
8. Automate Security & Compliance

RAG CASE STUDY FOR BFSI DOMAIN

1.Identify domain specific APIs

AccountCreation / Balance / Transactions
PaymentProcessing
Loan Management
Investment Management
Reporting and Analytics
Trading
Customer Management

CONF42

RAG CASE STUDY FOR BFSI DOMAIN

2. Develop Domain Specific RAGs

- Internal Docs
- Regulatory Guidelines
- Ontologies/Taxonomies
- Market Data
- Research Publications
- Vendor APIs
- Financial Data Lakes
- Community Forums/SDK
- Openbanking APIs
- Models & Knowledge Graphs

CONF42

RAG CASE STUDY FOR BFSI DOMAIN

3.Platform Abstraction Layer

```
function createAbstractionLayer():
    // Design APIs that abstract the complexity of BFSI services and RAG integration
    bfsAPIs = defineAPIsForCommonTasks()
    // e.g., customer data retrieval, transaction processing
    ragIntegrationAPIs = defineAPIsForRAGUsage()
    // e.g., document retrieval, augmented query response
    exposeAPIs(bfsAPIs, ragIntegrationAPIs)
```

- AccountCreation / Balance / Transactions
- PaymentProcessing
- Loan Management
- Investment Management
- Reporting and Analytics
- Trading
- Customer Management

RAG CASE STUDY FOR BFSI DOMAIN

4. Setup a Self-serve portal

```
function buildSelfServicePortal():
    // Set up a portal with integrated development tools and resources

    portal = initializePortalFramework()
    addDocumentation(portal, bfsAPIs, ragIntegrationAPIs)
    addSandboxEnvironment(portal, bfsAPIs, ragIntegrationAPIs)
    deployPortal(portal)
```

5. Ephermeral Environments

```
function setupEphemeralDevEnvironments():
  // Create container images with pre-installed BFSI tools and RAG integrations
    devContainer = createContainerImage(bfsAPIs, ragIntegrationAPIs, ciCDTools)
    publishDevContainer(devContainer)
    provideInstructionsForLocalSetup(devContainer)
```

6. Enable E2E Observability

```
function implementObservability():
  // Integrate logging, metrics, and tracing for BFSI and RAG-related activities
    loggingService = setupCentralizedLogging(bfsAPIs, ragIntegrationAPIs)
    monitoringService = setupMonitoringDashboards(bfsAPIs, ragIntegrationAPIs)
    tracingService = implementDistributedTracing(bfsAPIs, ragIntegrationAPIs)

    exposeObservabilityToolsToDevelopers(loggingService, monitoringService, tracingService)
```

RAG CASE STUDY FOR BFSI DOMAIN

7. Setup Pipelines

```
function automateCICDPipelines():
    // Set up CI/CD pipelines with BFSI-specific stages (e.g., security checks, compliance testing)
    pipeline = createCICDPipeline()
    addStagesForSecurityCompliance(pipeline)
    addAutomatedTestingStages(pipeline, bfsAPIs, ragIntegrationAPIs)
    integratePipelineWithVersionControl(pipeline)
    //Developers build the pipelines from the published template
    publishPipelineTemplatesToDevelopers(pipeline)
```

8. Security & Compliance

```
function automateSecurityCompliance():
    // Automate security and compliance checks within the platform

    securityScanner = integrateVulnerabilityScanner(bfsAPIs, ragIntegrationAPIs)
    complianceChecker = automateComplianceChecks(securityStandards, bfsAPIs)
    integrateSecurityComplianceIntoPipeline(securityScanner, complianceChecker)
    alertDevelopersOnSecurityComplianceIssues(securityScanner, complianceChecker)
```

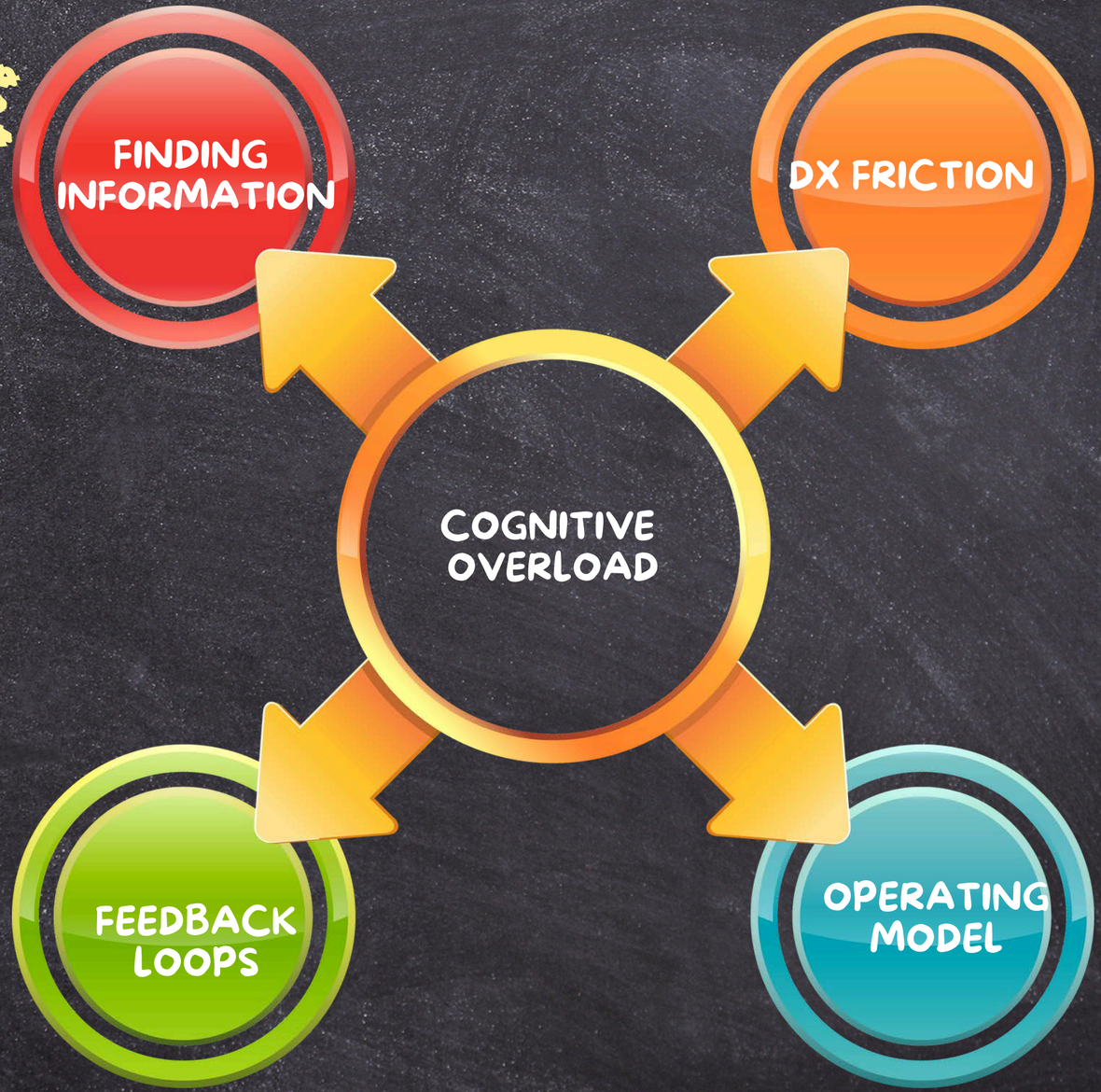
AI POWERED DEVEX

OVERHEAD/WASTE : VALUE-ADD DELIVERY

70:30

40:60

10:90 ???

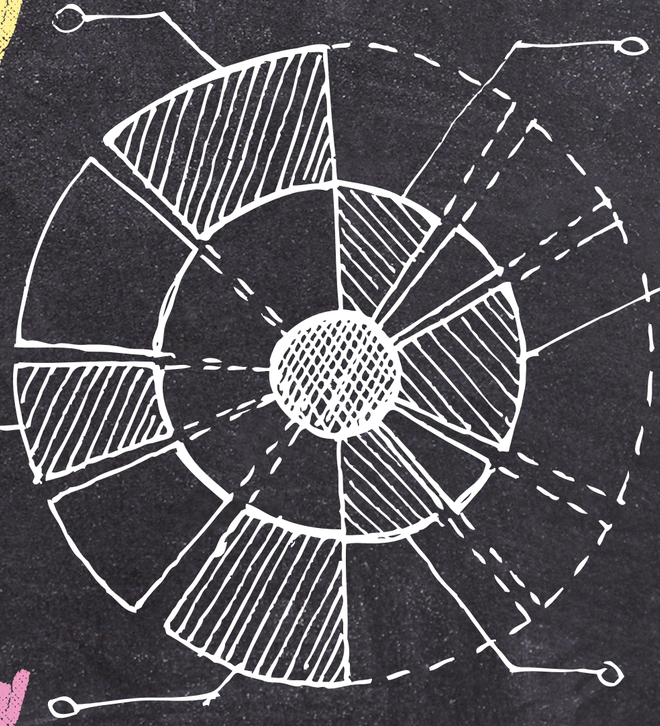


CONF42

LEVERAGE?

USER RESEARCH

MONITORING/OBSERVABILITY



TESTING

REQUIREMENTS ANALYSIS

CODING & DEBUG

SOLUTION ARCHITECTURE

CONCLUSION

AI is changing platform engineering and pushing the boundaries

Training LLMs are still challenging and costly. Focus on epochs

RAGs are starting to make a difference

Metrics that drive the success SHOULD NOT change

CONF42

brillio

CONNECT WITH ME?

BRILLIO / AJAY C

BRILLIO.COM