FinE

Turning Commonalities Into Components

Crafting Reusable Data Engineering Products

Spotify Finance

Koray Kocak

Sr. Data Engineer

Spotify > Finance > FinE

Agenda

- ★ Data Quality Requirements in Finance
- ★ Delivering transactional & analytical data
- ★ Entangled with Domain-Specific Complexities
- ★ Communicating with Non-Tech Audience
- ★ Isolating Domain-Specifics
- ★ Standardised Processes and Datasets

Set the Stage

Spotify Financial Engineering

Spotify FinE

Data Engineering in Spotify

Generating Analytical and Transactional Data

Data Engineering Golden Path

A Golden Path at Spotify is a structured, opinionated approach to help employees quickly learn Spotify's specific ways of working in a given domain.

Development

Scala

Orchestration

Flyte

CICD, Deployment, Catalog, Lineage, Backfill Experiment, Compliance, Monitoring, GDPR Controls, Documentation, Access Management, Team Management

Backstage

Batch & Near Real Time ETL Processing

Apache Beam

Scio

Data Lake & Analytics

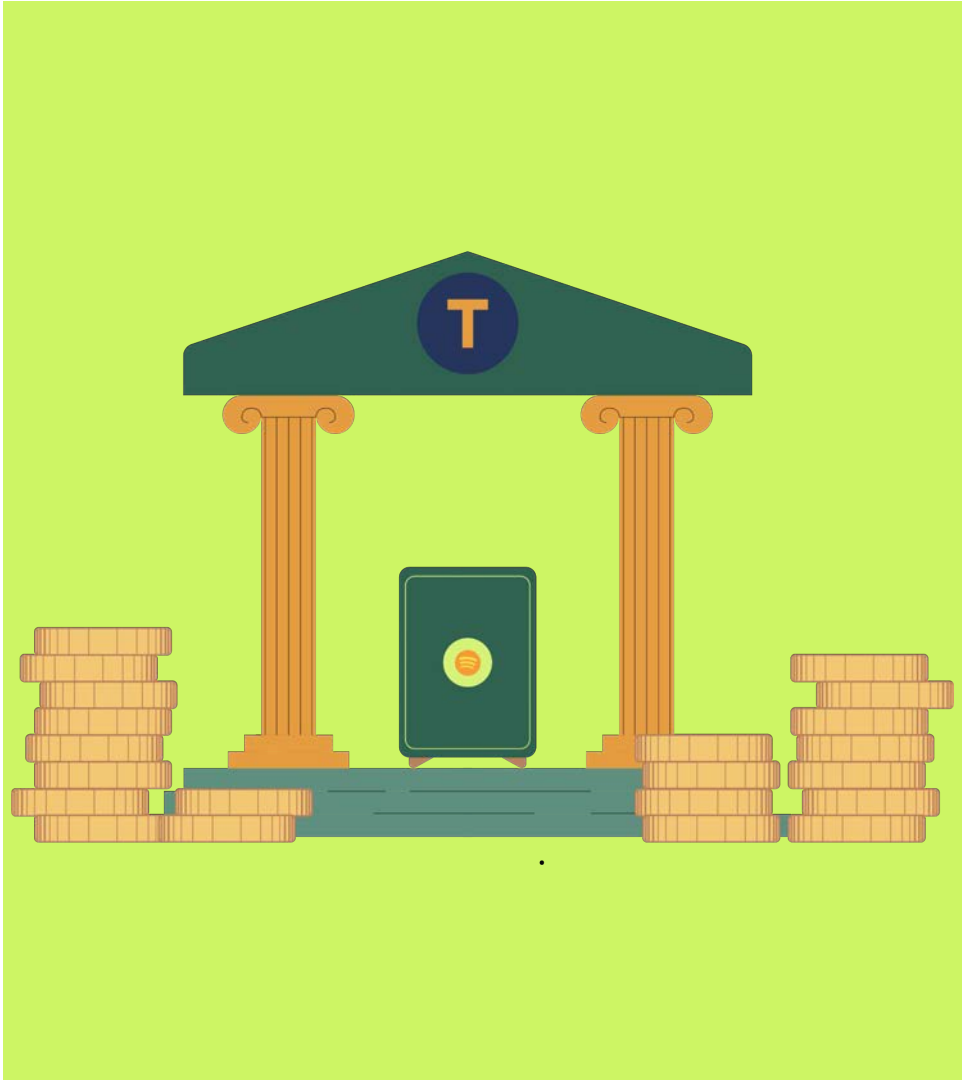
Backstage is a Spotify product that is open-sourced and adopted by many high-tech firms
Scio is an Apache Beam wrapper open-sourced by Spotify

Why Stop at Analytics?

Multi-Purpose Data Engineering by Transactional Data Generation

In Spotify, Data Engineering Teams are structured in a unique way to generate transactional data and analytics data.

In Financial Engineering operations, we produce granular and segregated data for analytics while generating Invoices and Journals for our ERP System.



Spotify FinE

Financial Data Quality Requirements

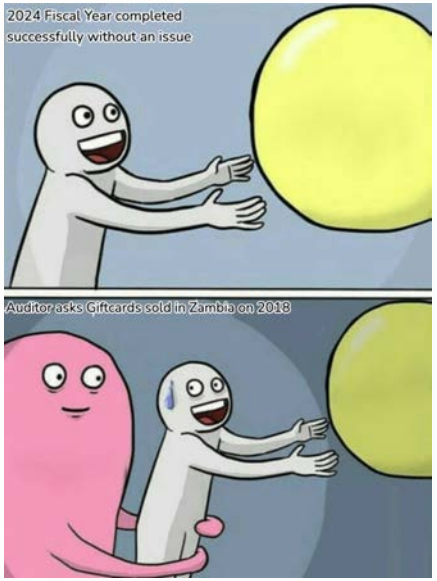
Data Operations
in a challenging domain

There is always a rounding issue in financial data

0.000001%

This is the acceptable rounding error in FX conversions

It is about money, data loss has serious consequences



Finance data is subject to strict **Tax Audits**, **ITGC***, and **SOX Controls****, requiring complete accuracy and traceability.

Inconsistent data can result in compliance violations, financial penalties, and reputational damage.

* Information technology general controls (ITGC)
** Sarbanes-Oxley Act of 2002 to safeguard against corporate fraud and financial misstatements

Challenge

Growth comes with complexity

Handling Custom Processes of the New Businesses and the Acquired Firms

A rapid growth trajectory

2008

Spotify Freemium

2008

Spotify launches Premium

2016

Dynamic Ad Insertion technology is widely adopted

2016-2017

Spotify launches podcasts and establishes Spotify Studios to create originals

Spotify acquires Soundtrap

2018

Spotify launches Premium through 3rd party partners

(e.g. Telco Companies)

2019

Spotify acquires Gimlet, Anchor, and Parcast

2020

Spotify announces Streaming Ad Insertion; acquires The Ringer, Kim Kardashian West, Joe Rogan, Addison Rae, and DC Partnership

2021

Spotify Acquires Megaphone

2022

Spotify acquires Findaway

2023

Spotify launches **Audiobook**

2024

Spotify launches Learning and Video Podcasts

Different Businesses Come with Different Challenges

Spotify started different businesses or acquired many companies along the way.

This brought a very high cognitive load to Finance operations due to varying and Financial Processes as well as Technical Tooling.

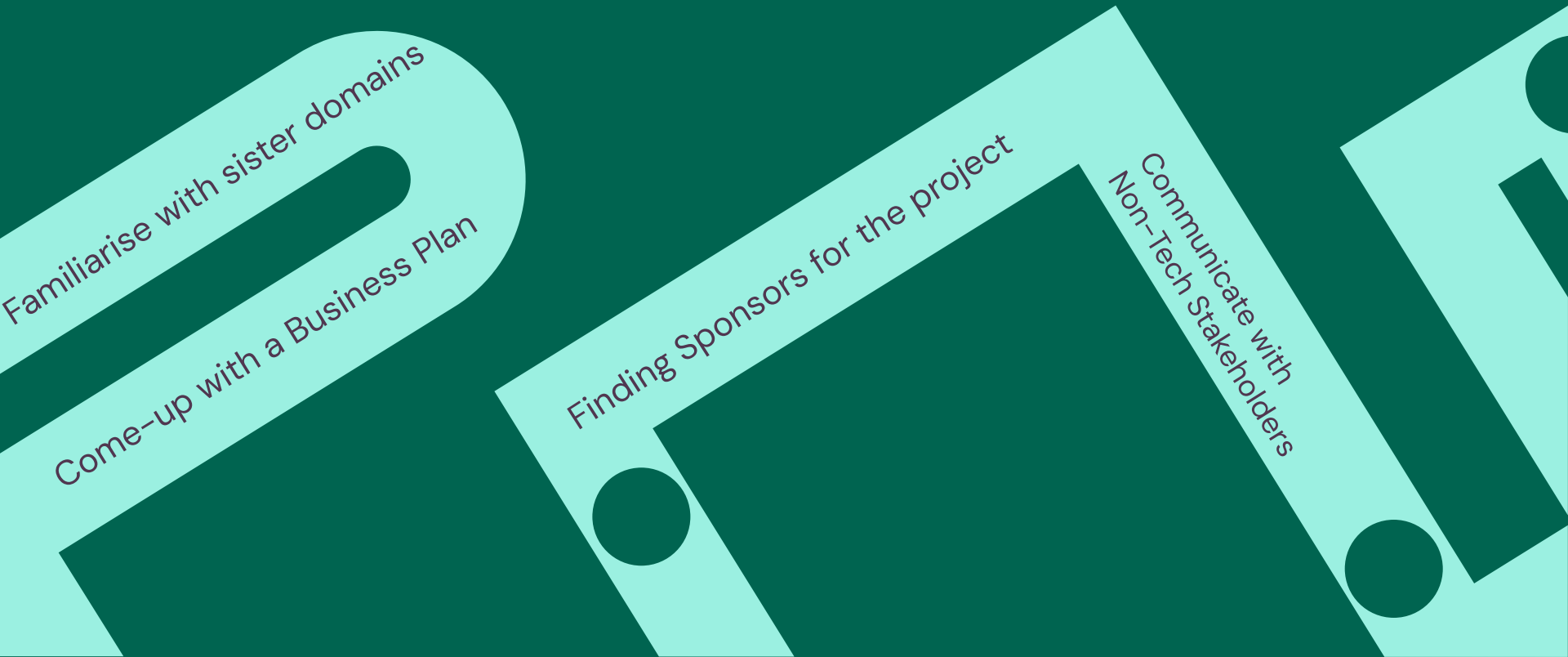
Domain Heavy Process

Typical Revenue Processing Operations became entangled with domain specific details

New Businesses and Newly Acquired Operations require minimum effort to bootstrap a running system, but with a big cost in the mid/long term.

Even though there are different products, there are similar underlying processes, like all products should produce invoice. These similar products are bucketed in different categories

Process Design



“The Big Shift”

Spotify Financial Engineering senior management saw the opportunity to break “Knowledge Silos” in the organisation.

An existing program, so called “Embed” heavily activated and the developers started switching sister teams to analyse, understand and the report existing processes.

Conclusions were similar, there were so many similar processes with different flavours to generate invoices, recognise revenues and confirms payments while creating analytics datasets out of them

Business Plan

To propose a new feature or improve a process those require heavy investment, you're required to prepare Strategy and Scope documents

Several senior engineers joint forces to come-up with a plan to create re-usable components instead of a direct data engineering product.

Similarities identified Proof of Concepts performed.

Finding Sponsor For your Project

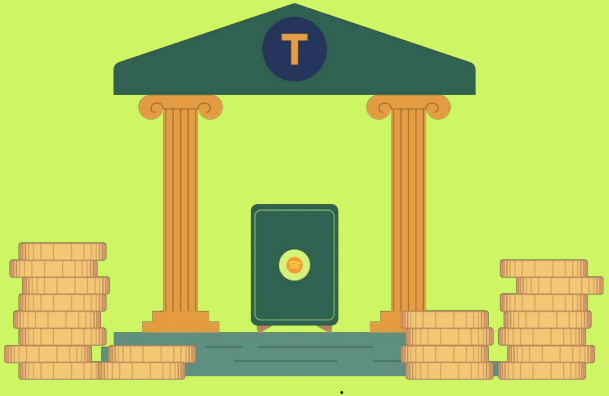
Upon Business Plan submission, C-Suite management assess projects and decide the project for the investment in a prioritised categories, so called “Company Bets”.

Your project has a workforce reserved and a timeline to deliver the project!

Communication with finance stakeholders

Convincing your ultimate clients with limited technical background to a heavy technical refactoring is not easy.

You need to demonstrate your long term vision with real-world mocked applications to make sure they will accept your significant change that impacts their daily operations without an immediate positive outcome.



Implementation

Disentanglement domain-specifics

Domain independent schema design

Creating unified interface for multiple components

Version and Dependency Matrix Management

Disentangling domain-specifics

Analyze Existing Processes – Examine workflows to detect patterns and redundancies.

Distinguish Shared vs. Domain-Specific Elements – Identify which components can be standardized and which require customization.

Document Process Variations – Create a structured map of variations across different business units or applications. Design runtime variations to minimise possible unintended user behaviours.

Define Standardization Opportunities – Highlight areas where modularization can reduce complexity and improve efficiency.

Assess Dependencies & Constraints – Understand how domain-specific rules interact with broader system architectures and shift them to the very beginning of the process.

Analyse the cost and benefit of the standardisation – If you do not have enough use cases to standardise, don't standardise them. If standardisation would be costly, you may end up sacrificing the project.

Schema design

Separation of Business Logic & Data – Decouple schema structure from domain-specific rules for reusability.

Flexible Data Modeling – Use adaptable structures like entity-attribute-value (EAV) or polymorphic relations.

Normalization & Standardization – Ensure data consistency and integrity while allowing domain-specific extensions.

Metadata-Driven Approaches – Leverage metadata layers to dynamically configure schema without rigid dependencies.

Scalability & Interoperability – Design schemas that support cross-domain data components.

Enforced Schema Integrity – Design schema libraries and present entire schema system in a versioned way to enforce schema consistency across the components

Unified Interface for multiple components

Packaging Components – Billing, Revenue Recognition, Cash Application, Financial Reports, and PSP Transaction Cost accrual components are encapsulated as executable workflows in individual repos.

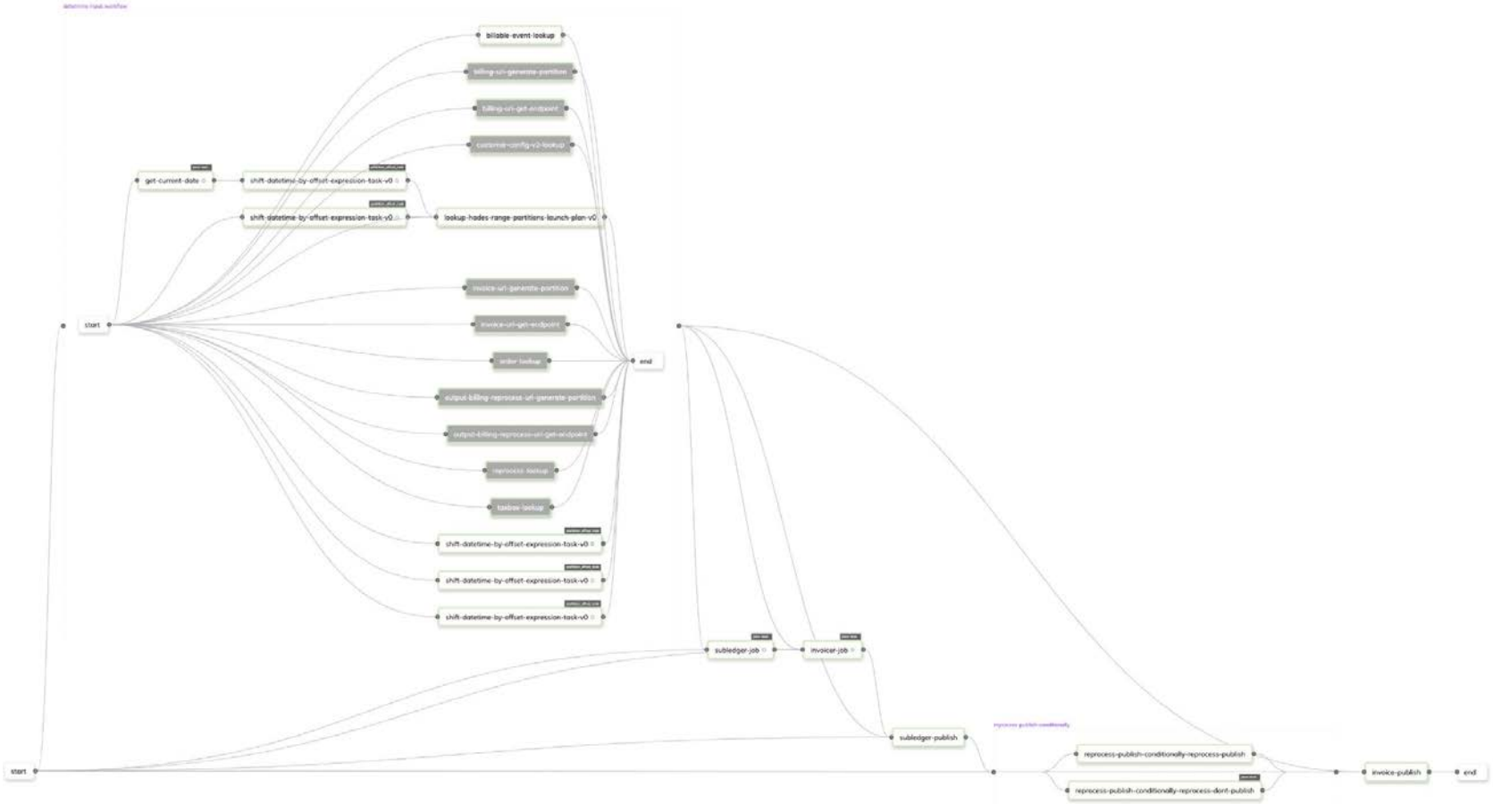
Centralized Library – A dedicated repository provides a unified way to access compiled Flyte JARs and execute workflows.

Integrated Schema Library – All schema definitions are centrally maintained to ensure consistency and seamless integration across components.

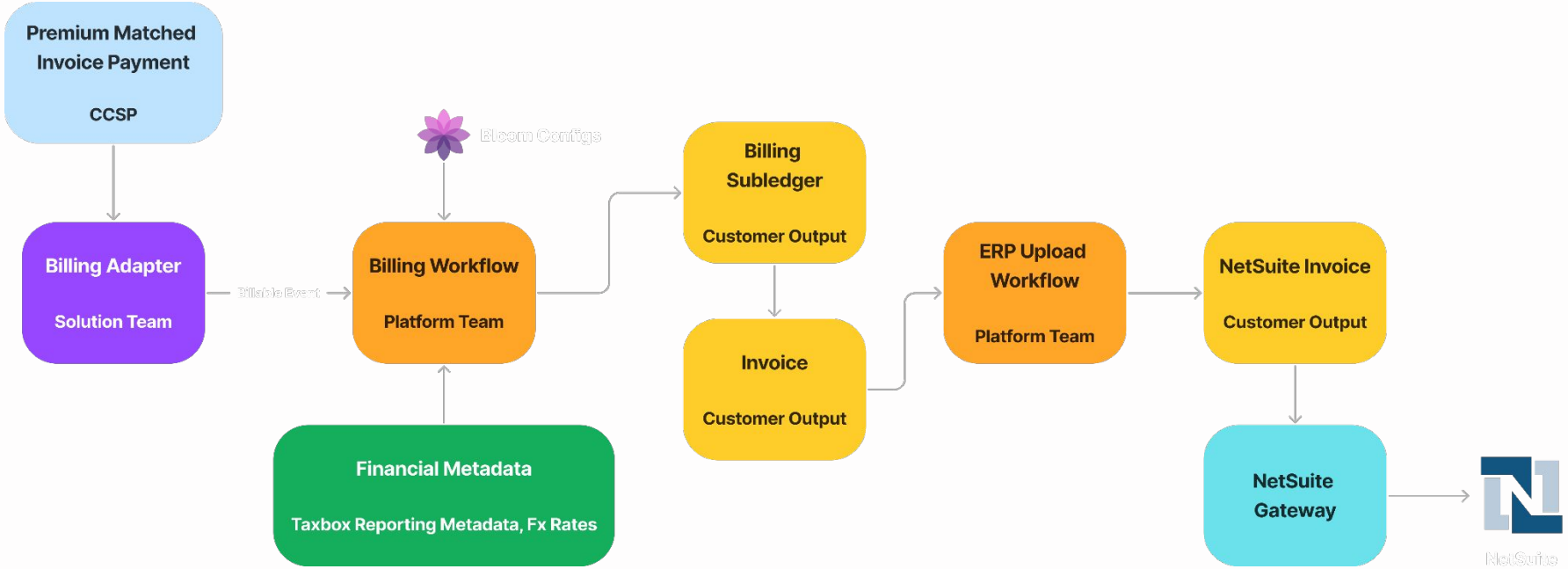
Standardized Execution Mechanism – Ensures consistency in how different components are called and executed, reducing complexity for users.

Version Management & Change Log Handling – Tracks changes using semantic versioning to maintain compatibility and traceability.

Remote Launch Plan Management – Users receive executable launch plans that abstract workflow complexities and enable seamless remote execution.


billable-event-lookup
billing-uri-generate-partition
billing-uri-get-endpoint
customer-config-v2-lookup
get-current-date
shift-datetime-by-offset-expression-task-v0
shift-datetime-by-offset-expression-task-v0
lookup-hades-range-partitions-launch-plan-v0
invoice-uri-generate-partition
invoice-uri-get-endpoint
order-lookup
output-billing-reprocess-uri-generate-partition
output-billing-reprocess-uri-get-endpoint
reprocess-lookup
taxbrw-lookup
shift-datetime-by-offset-expression-task-v0
shift-datetime-by-offset-expression-task-v0
shift-datetime-by-offset-expression-task-v0
start
end
subledger-job
invoicer-job
subledger-publish
reprocess-publish-conditionally-reprocess-publish
reprocess-publish-conditionally-reprocess-dont-publish
invoice-publish
start
end

Billing


Premium Matched Invoice Payment
CCSP
Billing Adapter
Solution Team
Billable Event
Bloom Configs
Billing Workflow
Platform Team
Financial Metadata
Taxbox Reporting Metadata, Fx Rates
Billing Subledger
Customer Output
Invoice
Customer Output
ERP Upload Workflow
Platform Team
NetSuite Invoice
Customer Output
NetSuite Gateway
NetSuite

Achievements

Adoption of the components by different teams

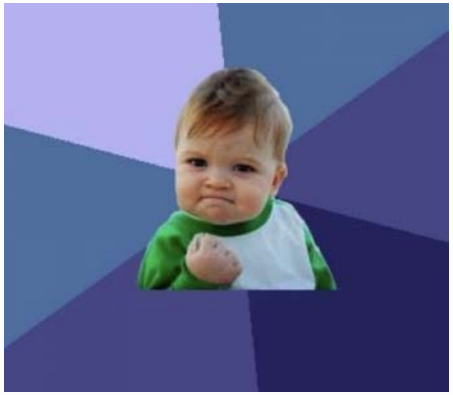
Reduced Overall Maintenance & Workload

Standardised datasets for better analytics

Easier User Acceptance sign-off processes

Adoption of the components by different teams

Product acceptance amongst revenue processing teams and their counterparty financial operations is huge.



Each team is excited with the actual outcomes of significant workload reduction and increased standardisation

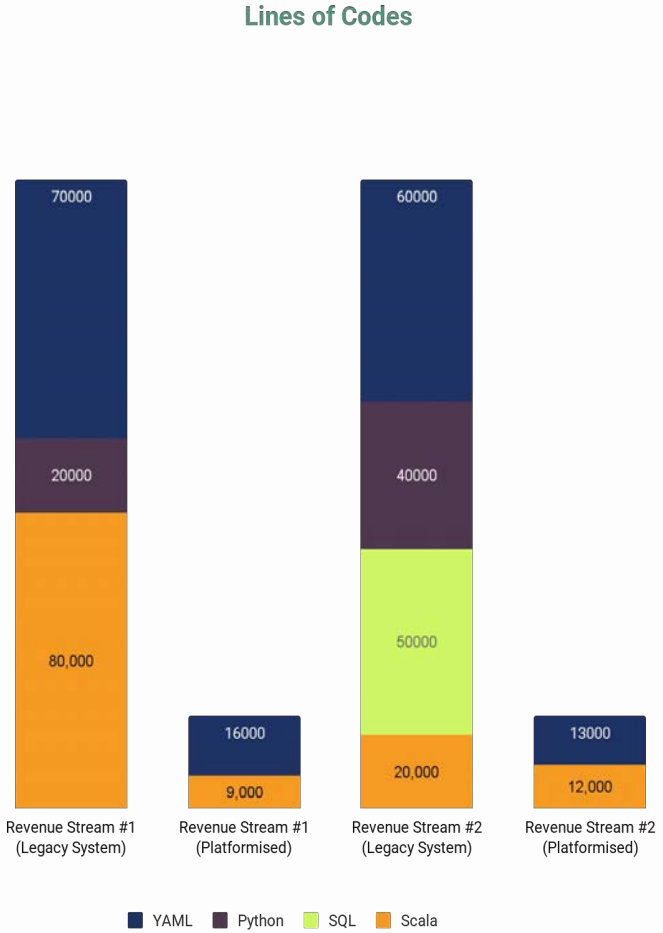
All teams are using a single tech-stack and this increased engineer workforce flexibility and easy onboarding between teams

Quantifying Complexity Reduction: Before & After Impact

Code is an asset, but each line is a liability.

CaaS (Complexity as a Service)

Code is a long-term liability—requiring constant maintenance, updates, and support, while inevitably accumulating technical debt.



Brief Summary for 2 complex revenue streams

Standardised Datasets

Subledgers

All revenue streams start producing identical and high granular data. This products are started to be heavily utilised by analytics and machine learning applications to make better decisions.

Shorter User Acceptance Process

Majority of the revenue processing code is now living in the platform. Since it is heavily used and a combat proven product, new revenue streams can be onboarded within days. Previous TTM durations for new revenue products took months!

90%

90% of the SOX Controls tests are written only once and lives in the platform codebase

3hrs

Average platform ticket resolution time

Spotify FinE

Q & A Session

korayk@spotify.com

linkedin.com/in/koray-ckk/

Spotify Finance

Thank you