International Journal of Management, IT & Engineering
Vol. 16 Issue 02, February 2026,
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Integrated End to End Reporting Automation

Author: Sudhansu Sekhar Behera
Enterprise Architect & Program Manager
Tata Consultancy Services, USA

ABSTRACT

Purpose of this research work is to automate the daily process of reporting that connects various end points from start to the end customer/consumer. In most IT practices, there are several IT teams representing their own peripheries that produces their regular reports with its own start and end points that makes business users do their respective maths to figure out any loss of data.

Whilst this paper demonstrates the modern tech stacks such as Azure, Adobe Experience Platform, Splunk, Stargate, kafka etc., the reader will find it useful across the board as this paper do no over complicate in joining the dots.

At the end of this paper, the reader will be able to produce a business-friendly useful report that will reduce the efforts of daily consolidation from business standpoint.

Keywords: Reporting, Business KPI, Data Analysis, Data Loss, Business Reporting Automation, End to End Reporting

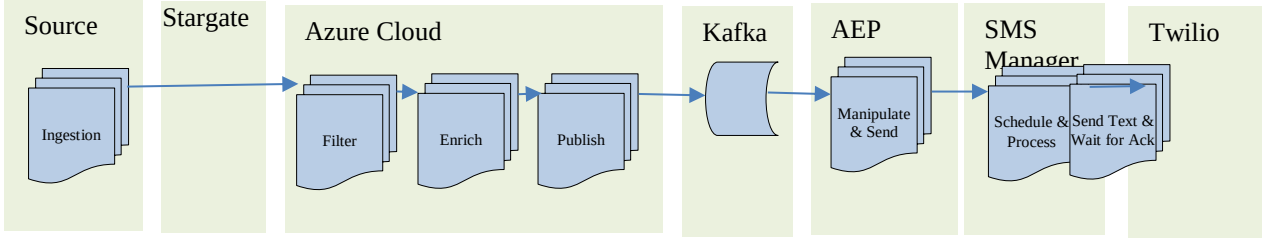
INTRODUCTION

In a typical business enterprise, even in a simpler scenario, the data travels through various channels for specific function and reasons such as

a. Ingestion
b. Data filtration
c. Data enrichment
d. Data Transformation
e. Publishing (to a messaging layer or topic)
f. Data translation
g. Data processing
h. Data consumption by final recipient

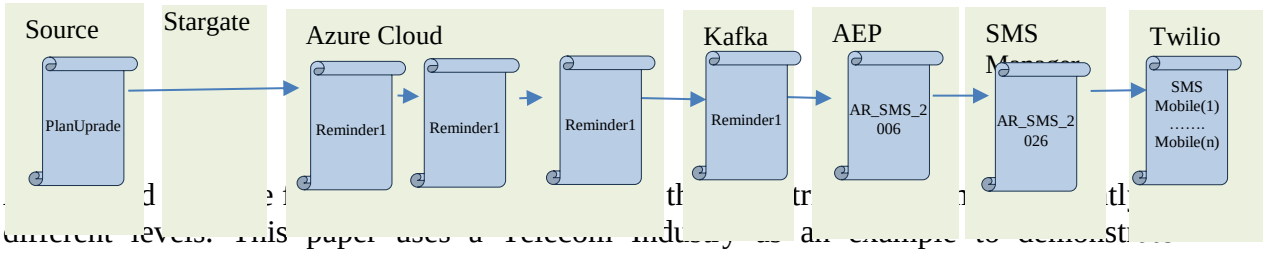
Then again, above functions do not happen in a single technological layer or server. Hence in a typical engagement, there are 4-8 IT operation teams that produce their daily reports within their own boundary oblivious to upstream or downstream systemic data flow. As a result, it makes it hard for business users to determine any possible data loss unless they put together all the IT reports on a single day to compare the ingested vs received data from one extreme end to another extreme.

Figure-1



As we see from Figure-1, business would engage 7 different IT Operation teams to obtain reports from each of their boxes after which they would try to evaluate and then synthesize. Even during the process of amalgamating (or stitching them together), business will have to keep in mind the data head definitions would often-times vary depending on the heads defined by each individual system.

Figure-2



comprehensive reporting. Its communication can include various such key attributes such as below to name a few

a. Plan upgrade Reminder via SMS
b. Plan upgrade Reminder via Email
c. Network Outage reminder via SMS
d. Network Outage reminder via Email
e. Add On Feature Notification via SMS
f. Add On Feature Notification via Email
g. Data Usage Notification via SMS
h. Data Usage Notification via Email
i. Roaming Service Education via SMS
j. Roaming Service Education via Email

To work with many such attributes on a daily basis, it makes the consolidation and data comparison difficult for business users, unless the report is formulated in its entirety with the most minimal number of columns possible with the outliers standing out prominently as given in Figure – 3.

Figure-3

			Source	Azure			Adobe Experience Platform				Channel M	3rd Party Vendor
#	Communication Type	Channel	Ingestion	Filter	Enrich	Publish	AEP Entry	AEP Exit	AEP Fail	Failure %	Channel Out	Channel received
1	PlanUpgrade	SMS	2500	2500	2496	2496	2496	2496	23	0.92%	2473	2412
2	PlanUpgrade	Email	3200	3198	3156	3156	3156	3150	120	3.81%	3030	3025
3	NetworkOutage	SMS	21000	21000	21000	21000	19859	19859	254	1.28%	19605	19600
4	NetworkOutage	Email	19000	19000	18965	18965	18965	18960	457	2.41%	18503	18495
5	AddOnFeature	SMS	900	900	896	896	896	896	21	2.34%	875	871
6	AddOnFeature	Email	800	800	800	800	785	784	45	5.74%	739	731
7	DataUsage	SMS	1500	1498	1485	1485	1480	1476	96	6.50%	1380	1345
8	DataUsage	Email	2000	2000	1989	1989	1989	1989	0	0.00%	1989	1956
9	RoamingEducation	SMS	21000	20992	20985	20985	20985	20985	568	2.71%	20417	20400
10	RoamingEducation	Email	17000	17000	16875	16875	16875	16875	712	4.22%	16163	16125

A report like above gives insightful elucidation to both business and IT departments revealing potential contributors of data loss; in above example, one can notice that

a. A portion of data got lost in stargate layer for Plan Upgrade-Email, Data Usage SMS and Roaming Education SMS
b. A higher percentage of failures occurred in data manipulation in Adobe layer for Add on Feature Email and Data Usage SMS.

To adhere to the timeliness of communications, business can then take action whether or not to give it another try and/or correct the data before channeling out the comms.

METHODOLOGY

Referring back to Figure-3, the tabular output consolidates information from various sources and queries before stitching them together and formatting them. The list of activities that are performed are as follows:

a. Get ingestion data from Source
b. Query Azure insights to list down the communication types that are filtered, enriched and published
c. Query Adobe Experience Platform to find entry along with successful and failed exits
d. Query Channel management system to obtain the comms that are sent & scheduled
e. Query response data from 3rd party vendors such as Twilio or Sinch etc.
f. Stitch the data together with mapping fields (e.g. PlanUpgrade vis-à-vis Reminder1 and Reminder 2026)
g. Format the data in excel sheet and enhance with colored outliers

Although this paper does not focus on basics of queries from different dataset tables and sources, it gives a few pointers for the sake of clarity and benefit for non-technical readers. The queries will later be referenced in the main file for sequential execution and stitching.

Get Ingestion Data from Source

Let's assume the initial ingestion emerges from an Oracle database that schedules the batches for different communications listed in previous sections.

Oracle Query Sample for Ingestion Data (source.sql)

```
select communication_type, communication_channel, count(*)
where date_communication =  TO_DATE(:report_date, 'YYYY-MM-DD')
group by communication_type, communication_channel
order by communication_type, communication_channel;
```

Above script takes "report_date" as a parameter counting the communications ingested on a given date.

Communication Type	Channel	Ingestion
PlanUpgrade	SMS	2500
PlanUpgrade	Email	3200
NetworkOutage	SMS	21000
NetworkOutage	Email	19000
AddOnFeature	SMS	900
AddOnFeature	Email	800
DataUsage	SMS	1500
DataUsage	Email	2000
RoamingEducation	SMS	21000
RoamingEducation	Email	17000

Filter vs Enrich vs Publish from Azure

This section outlines how different logs can be combined into a single tabular format that makes it easier to compare.

Kusto Query Sample – filter.kql

```
declare query_parameters (
      report_date: datetime
)
traces
| where cloud_roleName has “<your micro-service name>”
| where timestamp >= report_date
| where message has “<your generic log text that filters>”
| parse message with * “<matching filter pattern>” communication_type “channel” communication_channel
| summarize Filter=count() by communication_type, communication_channel
```

Above query can be run passing an argument for report_date such as 2026-01-02T00:00:00Z as below.

Kusto Query Sample – filter.kql

```
Kusto-cli --cluster “your cluster” --database “DB” --query-file “filter.kql” \
      --parameters “report_date=datetime(2026-01-02T00:00:00Z)
```

But since it obtains only the filtered counts, the output will look something like below.

Communication Type	Channel	Filter
Reminder1	SMS	2500
Reminder1	Email	3198
Outage Warning	SMS	21000
Outage Warning	Email	19000
FeatureNotification	SMS	900
FeatureNotification	Email	800
UsageData	SMS	1498
UsageData	Email	2000
Roaming	SMS	20992
Roaming	Email	17000

Hence we need to enhance the script that would include filter, enrich and publish in a single tabular view.

Kusto Query Sample – filter_enrich_publish.kql

```
declare query_parameters (
      report_date: datetime
)
//filter table
Let Filter= traces
| where cloud_roleName has “<your micro-service name>”
| where timestamp >= report_date
| where message has “<your generic log text that filters>”
| parse message with * “<matching filter pattern>” communication_type “channel” communication_channel
| summarize Filter=count() by communication_type, communication_channel;
//enrich table
Let Enrich= traces
| where cloud_roleName has “<your micro-service name>”
| where timestamp >= report_date
| where message has “<your generic log text that enriches>”
| parse message with * “<matching filter pattern>” communication_type “channel” communication_channel
| summarize Enrich=count() by communication_type, communication_channel;
//publish table
Let Publish= traces
| where cloud_roleName has “<your micro-service name>”
| where timestamp >= report_date
| where message has “<your generic log text that publishes>”
```

```
| parse message with * "<matching filter pattern>" communication_type "channel" communication_channel
| summarize Publish= count() by communication_type, communication_channel;
// Join three tables
Filter | lookup Enrich on by communication_type, communication_channel | lookup Publish by by
communication_type, communication_channel | order by communication_type, communication_channel
```

Upon running above kusto query, it would produce a table something like below:

Communication Type	Channel	Filter	Enrich	Publish
Reminder1	SMS	2500	2496	2496
Reminder1	Email	3198	3156	3156
Outage Warning	SMS	21000	21000	21000
Outage Warning	Email	19000	18965	18965
FeatureNotification	SMS	900	896	896
FeatureNotification	Email	800	800	800
UsageData	SMS	1498	1485	1485
UsageData	Email	2000	1989	1989
Roaming	SMS	20992	20985	20985
Roaming	Email	17000	16875	16875

It's important to note that the communication type in "ingestion" table looks different from above table in Azure due to data translation. The later sections in this paper will cover the procedure to map them together.

Adobe Experience Platform Query

Now that the ingestion and Azure data are available, next step is to get their respective results in Adobe Experience Journey datasets.

Adobe Experience Python Sample – adobe.py

```
import aepp
import sys
import pandas as pd
aepp.importConfigFile(config.json) #contains credentials and connection url
report_date = system.argv[1]
query = """ select Journey, AEPEntry, AEPExit, AEPFail,count(profile) from aep_dataset where timestamp::DATE
= {report_date} group by Journey, AEPEntry, AEPExit, AEPFail;"""
result = aepp.QueryService.query(query)
with open("AEPResults.csv", "w", newline="") as f:
  writer=csv.DictWriter(f, fieldNames=result[0].keys())
  writer.writerows(result)
df = pd.read_csv("AEPResults.csv")
#calculate percentage
df["Failure%"] = (df["AEPFail"] / df ["AEPEntry"]) * 100
#save to new csv
df.to_csv("aep.csv",index=False)
```

Journey	AEP Entry	AEP Exit	AEP Fail	Failure %
AR_SMS_2026	2496	2496	23	0.92%
AR_EM_2026	3156	3150	120	3.81%
Outage_SMS_2025	19859	19859	254	1.28%
Outage_SMS_2025	18965	18960	457	2.41%
AddOnFeature_SMS_2026	896	896	21	2.34%

AddOnFeature_SMS_2026	785	784	45	5.74%
DU_SMS_2026	1480	1476	96	6.50%
DU_SMS_2026	1989	1989	0	0.00%
Roaming_SMS_2026	20985	20985	568	2.71%
Roaming_SMS_2026	16875	16875	712	4.22%

Here again it's important to note that the communication types are different from Ingestion and Azure output, yet distinguishing the Emails and SMS channels due to the fact that the Adobe team have designed their journeys as per their standards.

Channel Management & 3rd Party Vendor Query

As this paper has touched upon the querying details from Oracle, Azure Insights and Adobe Experience Platform, the similar technique can be used to query other servers depending on their tech stack. Let's assume, once that's built, the output from Channel Management system and 3rd Party vendors looks something as below table.

Journey	Channel Out	Channel received
AR_SMS_2026	2473	2412
AR_EM_2026	3030	3025
Outage_SMS_2025	19605	19600
Outage_SMS_2025	18503	18495
AddOnFeature_SMS_2026	875	871
AddOnFeature_SMS_2026	739	731
DU_SMS_2026	1380	1345
DU_SMS_2026	1989	1956
Roaming_SMS_2026	20417	20400
Roaming_SMS_2026	16163	16125

MODELING AND ANALYSIS

As discussed in the introduction section of this paper, in a typical IT engagement, teams from each individual tech stack produce their reports within their boundaries using their defined nomenclature. In order to satisfy the need of business, this section will focus on stitching these reports together and formatting in a business-friendly legible way.

Put them together

For simplicity in understanding, this paper minimizes the complexity of differences in naming conventions that are spread across platforms as opposed to the scenarios in a typical enterprise ecosystem. Hence, the naming convention is left same for Adobe Experience Platform, Channel Management and 3rd Party vendors. But this section focuses on the basics of stitching different pieces together to create a single useful report.

As seen in Figure-4, the communication types appear differently in three separate tables and the AEP table even combines its communication type and Channel such as "AR_SMS" which corresponds to Plan Upgrade communication via SMS channel.

Figure-4

Ingestion/Source		Azure		AEP/ChannelM/3rd Party
Communication Type	Channel	Communication Type	Channel	Journey
PlanUpgrade	SMS	Reminder1	SMS	AR_SMS_2026
PlanUpgrade	Email	Reminder1	Email	AR_EM_2026
NetworkOutage	SMS	Outage Warning	SMS	Outage_SMS_2025
NetworkOutage	Email	Outage Warning	Email	Outage_SMS_2025
AddOnFeature	SMS	FeatureNotification	SMS	AddOnFeature_SMS_2026
AddOnFeature	Email	FeatureNotification	Email	AddOnFeature_SMS_2026
DataUsage	SMS	UsageData	SMS	DU_SMS_2026
DataUsage	Email	UsageData	Email	DU_SMS_2026
RoamingEducation	SMS	Roaming	SMS	Roaming_SMS_2026
RoamingEducation	Email	Roaming	Email	Roaming_SMS_2026

Adopted from the AEP naming convention, the mapping can be made easier by combining the two different columns in a similar manner, which can then look like below:

Figure-5

Ingestion/Source	Azure	AEP/ChannelM/3rd Party
Communication	Communication	Journey
PlanUpgradeSMS	Reminder1SMS	AR_SMS_2026
PlanUpgradeEmail	Reminder1Email	AR_EM_2026
NetworkOutageSMS	Outage WarningSMS	Outage_SMS_2025
NetworkOutageEmail	Outage WarningEmail	Outage_SMS_2025
AddOnFeatureSMS	FeatureNotificationSMS	AddOnFeature_SMS_2026
AddOnFeatureEmail	FeatureNotificationEmail	AddOnFeature_SMS_2026
DataUsageSMS	UsageDataSMS	DU_SMS_2026
DataUsageEmail	UsageDataEmail	DU_SMS_2026
RoamingEducationSMS	RoamingSMS	Roaming_SMS_2026
RoamingEducationEmail	RoamingEmail	Roaming_SMS_2026

The ideation can take inspiration from the most suitable approach, hence the methodology varies in different scenarios.

Above concatenation approach can be achieved in several ways using either awk or python scripting.

In this example, the source.csv file contains three columns, which can be simplified using below awk command:

$ awk -F”,” ‘{print $1$2”,”$3}’

Concatenation guidance

```
$ cat source.csv
CommunicationType,Channel,Ingestion
PlanUpgrade,SMS,2500
PlanUpgrade,Email,3200
NetworkOutage,SMS,21000
NetworkOutage,Email,19000
AddOnFeature,SMS,900
AddOnFeature,Email,800
DataUsage,SMS,1500
DataUsage,Email,2000
RoamingEducation,SMS,21000
RoamingEducation,Email,17000

$ awk -F"," '{print $1$2","$3}' source.csv
```

```
CommunicationTypeChannel,Ingestion
PlanUpgradeSMS,2500
PlanUpgradeEmail,3200
NetworkOutageSMS,21000
NetworkOutageEmail,19000
AddOnFeatureSMS,900
AddOnFeatureEmail,800
DataUsageSMS,1500
DataUsageEmail,2000
RoamingEducationSMS,21000
RoamingEducationEmail,17000
```

Now the next step will be to have a mapping file created in order to stitch the three CSV files together. Guided by the Figure-5 and achievability of concatenation, the mapping file can look something like below:

Mapfile.csv

```
$ cat mapFile.csv
sourcehead,azurehead,aephead
PlanUpgradeSMS,Reminder1SMS,AR_SMS_2026
PlanUpgradeEmail,Reminder1Email,AR_EM_2026
NetworkOutageSMS,Outage WarningSMS,Outage_SMS_2025
NetworkOutageEmail,Outage WarningEmail,Outage_SMS_2025
AddOnFeatureSMS,FeatureNotificationSMS,AddOnFeature_SMS_202
6
AddOnFeatureEmail,FeatureNotificationEmail,AddOnFeature_SMS_20
26
DataUsageSMS,UsageDataSMS,DU_SMS_2026
DataUsageEmail,UsageDataEmail,DU_SMS_2026
RoamingEducationSMS,RoamingSMS,Roaming_SMS_2026
RoamingEducationEmail,RoamingEmail,Roaming_SMS_2026
```

The three CSV data files look like below:

sourceConcat.csv

```
$ awk -F"," '{print $1$2","$3}' source.csv > sourceConcat.csv
$ cat sourceConcat.csv
CommunicationTypeChannel,Ingestion
PlanUpgradeSMS,2500
PlanUpgradeEmail,3200
NetworkOutageSMS,21000
NetworkOutageEmail,19000
AddOnFeatureSMS,900
AddOnFeatureEmail,800
DataUsageSMS,1500
DataUsageEmail,2000
RoamingEducationSMS,21000
RoamingEducationEmail,17000
```

AzureConcat.csv

```
$ cat AzureConcat.csv
CommunicationTypeChannel,Filter,Enrich,Publish
Reminder1SMS,2500,2496,2496
Reminder1Email,3198,3156,3156
Outage WarningSMS,21000,21000,21000
Outage WarningEmail,19000,18965,18965
FeatureNotificationSMS,900,896,896
FeatureNotificationEmail,800,800,800
UsageDataSMS,1498,1485,1485
UsageDataEmail,2000,1989,1989
RoamingSMS,20992,20985,20985
RoamingEmail,17000,16875,16875
```

aep.csv

```
$ cat aep.csv
Journey,AEPEntry,AEPExit,AEPFail,Failure%
AR_SMS_2026,2496,2496,23,0.92%
AR_EM_2026,3156,3150,120,3.81%
Outage_SMS_2025,19859,19859,254,1.28%
Outage_SMS_2025,18965,18960,457,2.41%
AddOnFeature_SMS_2026,896,896,21,2.34%
AddOnFeature_SMS_2026,785,784,45,5.74%
DU_SMS_2026,1480,1476,96,6.50%
DU_SMS_2026,1989,1989,0,0.00%
Roaming_SMS_2026,20985,20985,568,2.71%
Roaming_SMS_2026,16875,16875,712,4.22%
```

Stitching File

```
$ cat map.sh
#!/bin/bash
awk -F, '
FNR>1 && FILENAME=="sourceConcat.csv" { src[$1]=$2; next }
FNR>1 && FILENAME=="AzureConcat.csv" { az[$1]=$2 FS $3 FS $4; next }
FNR>1 && FILENAME=="aep.csv" { aep[$1]=$2 FS $3 FS $4 FS $5; next }
FILENAME=="mapFile.csv" { s=$1; a=$2; j=$3
        print s FS src[s] FS az[a] FS aep[j] }
' sourceConcat.csv AzureConcat.csv aep.csv mapFile.csv > Output.csv
sed -i '1d' Output.csv
echo Communication,Ingest,Filter,Enrich,Publish,AEPEntry,AEPExit,AEPFail,Fail %> report.csv
cat Output.csv >> report.csv
```

For each line in mapFile.csv, the above script finds the relevant values from three CSV files and places them into a output.csv file. After removing the first header (*not relevant*), the script adds a header to produce a report.csv file which looks like this:

Report File

```
$ cat report.csv
Communication,Ingest,Filter,Enrich,Publish,AEPEntry,AEPExit,AEPFail,Fail %
PlanUpgradeSMS,2500,2500,2496,2496,2496,2496,23,0.92%
PlanUpgradeEmail,3200,3198,3156,3156,3156,3150,120,3.81%
NetworkOutageSMS,21000,21000,21000,21000,18965,18960,457,2.41%
NetworkOutageEmail,19000,19000,18965,18965,18965,18960,457,2.41%
AddOnFeatureSMS,900,900,896,896,785,784,45,5.74%
AddOnFeatureEmail,800,800,800,800,785,784,45,5.74%
DataUsageSMS,1500,1498,1485,1485,1989,1989,0,0.00%
DataUsageEmail,2000,2000,1989,1989,1989,1989,0,0.00%
```

```
RoamingEducationSMS,21000,20992,20985,20985,16875,16875,712,4.22%
RoamingEducationEmail,17000,17000,16875,16875,16875,16875,712,4.22%
```

Final Reporting & Archival

For creating a better view for the report audience, it is necessary to present the data values in a readable format in comprehensive fashion.
They can be presented in a colorful excel file that can be produced using a python script as follows:

```
Python file to convert report.csv to an report.xlsx file in excel format
import pandas as pd
from openpyxl import load_workbook
from opempyxl.styles import PatternFill

df=pd.read_csv("report.csv") # read the csv file just generated
excel_file="report.xlsx" # name the output file
df.to_excel(excel_file,index=False)
wb = load_workbook(excel_file)
ws = wb.active

# Fill alternate rows except header row
fill1 = PatternFill(start_color="FFFFFF", end_color="FFFFFF", fill_type="solid")
fill2 = PatternFill(start_color="FFEDED", end_color="FFEDED3", fill_type="solid")

for row in range(2, ws.max_row +1):
  fill = fil1 if row %2 ==0 else fill2
  for col in range(1,ws.max_column +1)
    ws.cell(row=row, column=col).fill = fill
wb.save(excel_file)
```

Once executed the excel output should looks something like below:

Communication	Ingest	Filter	Enrich	Publish	AEPEntry	AEPExit	AEPFail	Fail %
PlanUpgradeSMS	2500	2500	2496	2496	2496	2496	23	0.92%
PlanUpgradeEmail	3200	3198	3156	3156	3156	3150	120	3.81%
NetworkOutageSMS	21000	21000	21000	21000	18965	18960	457	2.41%
NetworkOutageEmail	19000	19000	18965	18965	18965	18960	457	2.41%
AddOnFeatureSMS	900	900	896	896	785	784	45	5.74%
AddOnFeatureEmail	800	800	800	800	785	784	45	5.74%
DataUsageSMS	1500	1498	1485	1485	1989	1989	0	0.00%
DataUsageEmail	2000	2000	1989	1989	1989	1989	0	0.00%
RoamingEducationSMS	21000	20992	20985	20985	16875	16875	712	4.22%
RoamingEducationEmail	17000	17000	16875	16875	16875	16875	712	4.22%

In addition to emailing the daily reports, it can also be useful to publish the reports to a share portal such as confluence. Reader want to refer to a paper titled “Business & IT KPI Control Orchestration and Collaboration” in Vol 13 Issue 10, Oct 2023 in IJMRA.us – International Journal of Management, IT & Engineering – link is given in references section.

RESULTS AND DISCUSSION

This research is implemented in a major Telecommunication sector where the business users earlier had to deploy data engineers dedicated to perform the tasks of comparing 50+ reports on a daily basis only to discover any loss or delay in communication.

Although this initiative was earlier considered a bit far-fetched, the business realized surprising benefits along with discovery of hidden loopholes and missed-out bugs in their software releases.

This implementation not only reduced the reporting cost, but also helped business in early detection of data loss who later focused on correction of major fail-over situations as well as troubleshooting missed out communications.

After having realized the benefits, the approach was expanded to other Business Units.

CONCLUSION

After implementing the solution and also having researched around the present trend of IT practices, this approach can culminate in following tangible outcomes:

- Customer satisfaction
- Reduced workload for Business users and data analysts
- Reduced reporting cost
- Proactive Discovery of data loss
- Improved Customer communication in a timely manner
- Reinstate business focus on revenue growth rather than data analysis

RECOMMENDATION

Oftentimes the loss of data goes un-noticed unless a dedicated set of business users meticulously perform mundane tasks on a daily basis closely comparing reported data in every single upstream and downstream systems.

An integrated end to end report in a human readable format helps businesses pinpoint potential leakage of data across several tech stacks. It further encourages users and build teams to make retrospective corrections and/or infrastructure scaling for optimal usage of resources.

REFERENCES

1. Improve Business Reports by solving these 11 challenges – By Derek Nachimow https://www.citrincooperman.com/In-Focus-Resource-Center/Improve-Business-Reports-by-Solving-These-11-Challenges
2. Building Personalized Connections at Scale: T-Mobile's Adobe Journey - S733 https://business.adobe.com/summit/2025/sessions/building-personalized-connections-at-scale-s733.html
3. 8 Common Challenges in Business Reporting And How to Solve Them - https://www.clearpointstrategy.com/blog/8-common-challenges-in-business-reporting-and-how-to-solve-them
4. Fix Reporting Gaps in Your Business - https://blog.origin63.com/overcoming-reporting-challenges-smarter-business-decisions
5. Business & IT KPI Control Orchestration and Collaboration - https://www.ijmra.us/project%20doc/2023/IJME_OCTOBER2023/IJMIE3Oct23_22925.pdf

6. Publishing reports in Confluence - https://www.researchgate.net/publication/378436054_Business_IT_KPI_Control_Orchestration_and_Collaboration