A STREAM PROCESSING BASED REGTECH SOLUTION ARCHITECTURE FOR BANKS

Arulnayagam Yasothar

189307T

Degree of Master of Science in Computer Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

April 2021

A STREAM PROCESSING BASED REGTECH SOLUTION ARCHITECTURE FOR BANKS

Arulnayagam Yasothar

189307T

Thesis submitted in partial fulfillment of the requirement for the degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

April 2021

DECLARATION

I declare that this is my own work and this report does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or Institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgement is made in text.

Also, I here by grant to University of Moratuwa the non-exclusive right to reproduce and distribute my theses in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)

……. ……

A.Yasothar Date

The above candidate has carried out research for the Masters thesis under my supervision.

……. ……

Dr. Indika Perera (Research Supervisor) Date

ACKNOWLEDGEMENT

I would like to thank Dr. Indika Perera for accepting this work to be done under his supervision. Grateful and thankful to my batch mates at the University of Moratuwa for the encouragement and support. Great appreciation for my current and past work colleagues specially from N-Able, DFCC, BOC and HNB banks. Finally, to my family for all the patience and love during the never ending, long thesis writing hours.

ABSTRACT

Recent advancements in financial technology have given the opportunity for the rise of regulatory technology. RegTech or smart regulation is the use of cutting-edge technology for compliance and regulatory purposes. With technology invading the finance domain in the form of FinTechs, manual regulations and compliance will become obsolete in the near future. Banks should come up with architectures to build RegTech systems surrounding existing banking systems. Sri Lankan banks which are currently at the verge of implementing open banking to open-up banking services to FinTechs, have the biggest necessity of implementing RegTech solutions. With millions of transactions happening per second, and billions of amounts being moved between continents, the monitoring mechanism also should be smart and efficient. Biggest challenge of architecting a RegTech solution surrounding the current legacy Sri Lankan banking eco system is that they should have a minimal footprint on the operation and should have zero visibility on its existence. Stream processing on the other hand, a technology paradigm that took strides in the recent times, is a suitable candidate to address these challenges. Once we tap into the open banking API stream data and architect a RegTech solution surrounding it, the possibilities will become endless. Another major challenge is once we architect a solution, we need to evaluate it whether it caters the needs of all the stakeholders of the project. Architecture trade off analysis method and cost-based analysis method are two such analysis methods which brings all the stakeholders of a project to a single table and addresses their concerns. These methods are widely accepted and practiced by architects. This project will select a Sri Lankan bank, analyze on their current RegTech capabilities, propose a stream-based solution architecture, evaluate this new architecture using ATAM and CBAM methodologies, and implement few RegTech use-cases using the proposed solution. This proposed architecture can be used as a blueprint for any future RegTech solution implementation.

TABLE OF CONTENTS

Declaration i
Acknowledgement ii
Abstract iii
List of Figures vi
List of Tables vii
List of Abbreviations viii
1 Introduction 1
1.1 Background 1
1.2 Problem Statement 2
1.3 Case Study 2
1.4 Scope 3
1.5 Thesis Outline 3
2 Literature Review 4
2.1 FinTech 4
2.2 Open Banking 5
2.3 RegTech 9
2.4 Stream Processing 11
2.5 Architecture Evaluation 14
2.5.1 Architectural Tradeoff Analysis Method (ATAM) 16
2.5.2 Cost Based Analysis Method (CBAM) 22
3 Methedology 26
3.1 RegTech in Sri Lankan Banks 26
3.1.1 DFCC 26
3.1.2 Bank of Ceylon 28
3.1.3 Hatton National Bank 29
3.2 DFCC for Implementation Choice 30
3.3 Stream Processing Technology Analysis 30
3.3.1 WSO2 Stream Integrator 31

3.3.2 IBM Streams 31
3.3.3 Apache Flink 31
3.3.4 Azure Streams 32
3.4 WSO2 Stream Integrator for Implementation Choice 32
3.5 DFCC – Proposed RegTech Solution Architecture 33
3.5.1 RegTech Use-case 1 – KYC 33
3.5.2 RegTech Use-case 2 – Risk Analysis 34
3.5.3 RegTech Use-case 3 – Audit Reporting 34
4 Architectural Analysis 36
4.1 Architecture Evaluation 36
4.1.1 Evaluation Participants 36
4.2 ATAM for Proposed Solution 37
4.2.1 Utility Tree for the Proposed Solution 39
4.2.2 Report Summary 40
4.3 CBAM for Proposed Solution 42
5 Implementaion 47
5.1 WSO2 Stream Integrator 47
5.2 Solution Architecture 49
5.2.1 RegTech Implementation 1 – KYC 49
5.2.2 RegTech Implementation 2 – Risk Analysis 51
5.2.3 RegTech Implementation 3 – Audit Reporting 52
5.3 Sample API Performance Results 54
6 Conclusion and Future Work 55
6.1 Conclusion 55
6.2 Future Work 55
7 References 57

LIST OF FIGURES

Figure 2.1 : Impact of open banking 6
Figure 2.2 : Core banking FinTech integration 8
Figure 2.3 : Stream processing flow 12
Figure 2.4 : ATAM conceptual flow 22
Figure 2.5 : Context of CBAM 23
Figure 2.6 : CBAM - Process flow diagram 24
Figure 3.1 : System architecture - DFCC 27
Figure 3.2 : Systems architecture - BOC 28
Figure 3.3 : Systems architecture - HNB 29
Figure 3.4 : DFCC - Proposed solution architecture 33
Figure 4.1 : ATAM - DFCC utility tree 40
Figure 5.1 : Siddhi block diagram 49
Figure 5.2 : RegTech implementation - KYC source code 50
Figure 5.3 : RegTech implementation – KYC source code 51
Figure 5.4 : RegTech implementation - Risk analysis 52
Figure 5.5 : CRIB API call 53
Figure 5.6 : RegTech implementation - audit reporting 53
Figure 6.1: DFCC Future architecture proposal 56

LIST OF TABLES

Table 2.1 ATAM participants 18
Table 2.2 ATAM evaluation phases 20
Table 4.1 Tools and technologies used for implementation 48
Table 4.2 : No of APIs without SSL 54
Table 4.3 : No of APIs with SSL 54
Table 5.1 : DFCC quality attribute trade-offs 41
Table 5.2 : CBAM step 1 - collate scenarios 43
Table 5.3 : CBAM Step 2 - refine scenarios 43
Table 5.4 : CBAM step 3 - prioritize scenarios 44
Table 5.5 : CBAM step 4 - assign utility scores 44
Table 5.6 : CBAM step 5 - develop strategies 45
Table 5.7 : CBAM step 6 - determine utility 45
Table 5.8 : CBAM step 7 - calculate benefits 46
Table 5.9 : CBAM step 8 - choose strategy 46

LIST OF ABBREVIATIONS

ATAM	Architectural Tradeoff Analysis Method
CBAM	Cost Based Analysis Method
FTP	File Transfer Protocol
CDM	Cash Deposit Machine
KYC	Know Your Customer
CBSL	Central Bank of Sri Lanka
IB	Internet Banking
ATM	Automated Teller Machine
ESB	Enterprise Service Bus
API	Application Programming Interface
PSD2	Payment Services Directive 2
SOAP	Simple Object Access Protocol
REST	Representational State Architecture
FCM	Financial Crimes Mitigation

1 INTRODUCTION

1.1 Background

Technology has made great impact on all the industries and banking is no exception. From onsite banking to online banking, from file-based FTP to online real time integration with corporates, technology has shaped banking and finance in so many ways. We do not need to visit a bank counter to deposit money, there are cash deposit machines (CDMs) available all over the country. We do not need to send heaps of documents to get a loan, we can digitally scan them, email them, and get a loan approval within days. Obtaining a credit card is as easy as sending an SMS, where an agent will call you and confirm the details and then send the card via a courier. With the advent of open banking initiative, most of the banking services are opened for third parties. With so many advancements, so much openness, and technology also becoming cutting edge, so should the regulatory aspect. Banks which make money from people's money are prudently monitored and regulated by the Central Bank of Sri Lanka. With the advent of FinTechs, this regulatory requirement becomes even more essential and compulsory. FinTechs have gained so much popularity for their innovativeness and tech savvy consumers are willing to move towards mobile applications which provide seamless banking functions. Similar to the rise of FinTechs, RegTech is a new concept that is taking so much prominence during this modern banking era. RegTech can be described as the management of regulatory process within financial industry through technology [1]. Since Sri Lankan banks are in the process of implementing open banking, architecting a RegTech solution becomes mandatory. In this project we will propose a stream processing based RegTech solution architecture for a local Sri Lankan bank, analyze this architecture with architecture evaluation methods such as Architectural Trade-off Analysis Method (ATAM), Cost Benefit Analysis Method (CBAM). ATAM and CBAM are tried and tested methodologies to evaluate architectures and will be helpful to analyze and improve the proposed architecture.

1.2 Problem Statement

In the Sri Lankan context of banking, banks use legacy Core Banking systems. These are tightly coupled modularized systems serving day to day banking needs. To achieve regulatory compliance, banks have purchased fraud guard solutions, compliance software at a very high cost, and importantly make use of data warehouse technologies to maintain regulatory requirements. These compliance software's are costly and have a burden of integration complexity and operational overhead. Also, the data warehouse solutions are reactive rather proactive solutions. They require a skilled employee specifically to maintain the smooth operation of the said compliance, data warehouse software, to look after software failures and make configuration changes and make sure the system is always up and running. These monolith software's can be considered as RegTech in current Sri Lankan banking space since they are playing the role of compliance and regulatory software in the current context. What if there is an open-source cost effective solution architecture of implementing a RegTech solution. What if the implemented solution will require only an average level skill set to troubleshoot and ensure smooth operation. This project will try to identify a suitable RegTech architecture using steaming technology.

1.3 Case Study

As part of this project a bank would be selected for case study. Existing regulatory and compliance software will be analyzed and discussed. Then a stream processing based RegTech solution architecture will be proposed. This solution architecture will be evaluated by using standard architecture evaluation methods such as CBAM and ATAM. Once analyzed, few RegTech use cases will be selected and implemented as a prototype. This implementation will only be limited to the open banking API paradigm.

1.4 Scope

Scope of this project will be limited to analyzing and understanding Sri Lankan regulatory landscape, how regulations are incorporated in Sri Lankan banks and proposing a RegTech solution architecture to open banking initiatives which are in the grass root levels. A sample bank will be chosen for this purpose and the analysis will be conducted. Detailed study will be carried out on how the regulatory requirements are adhered with the existing bank systems architecture. An analysis will be carried out on steaming technology on how it will be suitable candidate to build a RegTech solution surrounding the existing Core Banking System and Open API suit. The scope will only consider RegTech architecture for open banking. An architecture will be proposed to the bank and few RegTech use cases will be simulated and built as prototype. At the end, an architectural analysis will be conducted using standard architectural analysis methods ATAM, CBAM, with the help of few stakeholders and the proposed architecture will be validated.

1.5 Thesis Outline

This thesis is structured in the following manner: Chapter 2 will discuss literature review, where relevant literature pertaining to the topic will be discussed and analyzed. Chapter 3 will discuss the current status of RegTech solution architecture in banks, discuss about existing problems and a suitable methodology to overcome these problems. This methodology ultimately would be proposing an adoptable solution architecture to the bank. Chapter 4 will be about the analysis of the proposed solution architecture. Architecture evaluation methods such as ATAM, CBAM will be taken up and used to analyze the proposed architecture. Chapter 5 will discuss the implementation of this proposed architecture. A sample stream processing solution prototype will be proposed and implemented in a selected bank. Chapter 6 will conclude the thesis with the observations, discussions, and possible future work.

2 LITERATURE REVIEW

This chapter will discuss on existing literature. Literature review will be conducted based on following topics playing a crucial role in the RegTech space.

- FinTech
- Open Banking
- RegTech
- Stream Processing
- Architecture Evaluation

2.1 FinTech

During the past decade FinTechs have taken prominence in the financial industry. With the help of cutting-edge technology FinTech companies have started replacing and fulfilling banking services [2]. FinTech is nothing but new technologies that seek to improve and automate delivery and use of financial services. At a high level, FinTechs can be attributed as any companies making use of internet, cloud services, software technology, any device which is mobile (RFID, payment strips), that used to connect or interact with financial services [41]. Though primarily FinTechs enable connectivity between customers finances with the uses of technology, the term can be applied to the collaboration between business to business (B2B) technologies as well. Initially FinTech referred to only technology used in back-end systems but with time, this has changed and now consist a wide range of other applications that are more consumer focused [4]. Some new age FinTech examples are as follows.

- An airline ticket booking application making use of a bank payment gateway or a fund transfer open API exposed by bank.
- An insurance company allowing customers to make premium payments via mobile application which is integrated with a bank.
- A fitness tracking device allowing users to get special savings account rates for health goals achieved daily.

Furthermore, there can be crowd funding, crypto currency, robot advisors etc. The above three examples given coincide with the use cases related to this project, and they involve open banking APIs [42] which is under the scope of this project. With FinTechs evolving at a faster rate, so should the regulatory component.

Few interesting characteristics of FinTechs are [43],

- They focus on either a single product or service which they provide exceptionally well in a lower cost with uninterrupted user interaction. TransferWise, is a great example which provides international borderless money-transfer in a seamless manner.
- They focus on clear customer direction. Without depending heavily on the legacy of traditional banks, Fintech can focus on solving users' problems. One simple example is the PickMe Sri Lanka, where it brings ATM cash withdrawals to your doorstep. PickMe a ride hailing service mobile application providing banking need is a great feat.
- They make competitive advantage by using some sort of advance technology. One simple example is Sri Lankan bank, Seylan trying to build intelligent data via customers visiting frequent merchant stores.

Due to this clear demarcation, customers flock to consume FinTech services, hence the chance of fraudulence are also high.

2.2 Open Banking

The focal point of this project would be RegTech for open banking. As the name indicates, Open banking is a financial services term as part of open financial technology that refers to:

- Technology that allows non-banking third party developers to allow development of application and services around the financial institution.
- For account holders, provide greater transparency options ranging from private data and public data.
- Use cutting edge open-source technology to achieve aforesaid [44].

For few reasons Open Banking has gained traction in recent years.

- Bank's inability to innovate with their products and services. Banks are so tightly coupled to legacy procedures and practices.
- The advent of cutting-edge technologies like Artificial Intelligence, Machine Learning, Stream Processing and Blockchain.

Figure 2.1 illustrates the how open banking will change the landscape of customer-bank interaction [45].

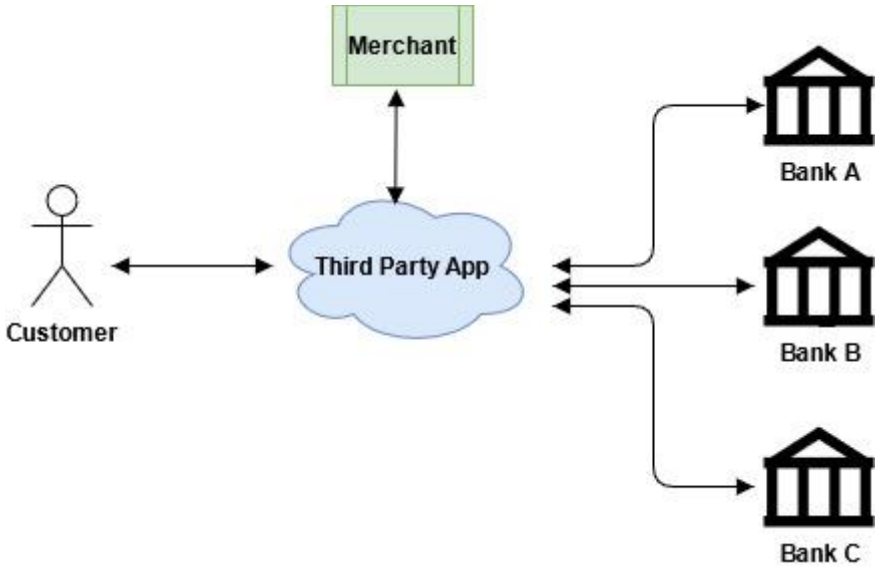

Figure 2.1 : Impact of open banking

Assume a customer deals with 3 banks namely Bank A, Bank B, Bank C. There is a merchant who's the customer needs to interact for some financial need. If there is no open banking available, customer will have the pain staking process with dealing with each of this entity (merchant bank, customer banks) separately. If the Banks had opened up their services and a third-party app integrates the merchant and the banks, the customers experience would be seamless and effortless.

Following are few reasons why banks should embrace open banking [44].

- Compliance

With the European Union's implementation of PSD2, banks are required to share customer data with third party applications and sharing this information will come up with strict and

stern compliance requirement. Compliance will not be about adding additional business but staying in business. This regulation will definitely hit the south Asian region in near future and banks would be pushed comply this regulation.

- Improved agility

The hardest part of open banking will be to share information between various systems in a secure, faster and efficient manner. With the current legacy banking architectures, this might be not possible, and banks might need to rethink their architecture being API centric. An improved architecture will allow the bank to use data effectively which in turn will improve customer experience, thereby increasing customer retention. The more customer stays in the bank, the more the income generating possibilities for the bank.

- Premium API products

Another exiting feature of open APIs are to create premium products with ease. As an example, HNB recently released a fitness financial product integrating a wearable device connected with a customer's bank account, and when achieving certain fitness goals, a special savings interest was given for that day. For this special type of account, fitness enthusiast signed up with great eagerness. Though fitness had zero relation with savings, this innovative idea combined, brought in new customers to bank.

- Increased customer satisfaction

Open banking provides great freedom since the number of possible financial services opportunities are also high. On the other hand, banks will have a clear disadvantage as the customer data has to be shared with third parties and those organizations will make use of this data. With the availability of open banking and financial integrations, customer experience will improve, and customer will be less likely to find alternatives.

- Potential for collaboration

Open banking will set up the pathway to third party companies to access customer data. Banks should take this as a positive step and work together for both parties to excel. This collaboration will be beneficial for both parties. Opening up banking services to third party apps will ultimately end up attracting the third-party app customers also to the bank [45].

Below are some useful use cases of open banking initiative [24].

- Provide a single interface to customers having multiple engagements with multiple banks.
- To help customers manage their finances better, so that they can develop analytical models to identify patterns around their financial profiles.
- Based on customer requirements build tailor made solutions for specific customers which cater customer needs.
- For corporates, help with consolidation of other non-business activities, financial activities such as invoicing, reconciliation, payments.
- Facilitate payment services directly through banks, try to minimize usage of credit cards, cheques, and cash since these will lure additional burden of maintenance of these entities.

With exiting so many avenues open, RegTech has become crucial in handling these opportunities.

Figure 2.2 illustrates how banks and FinTechs work together.

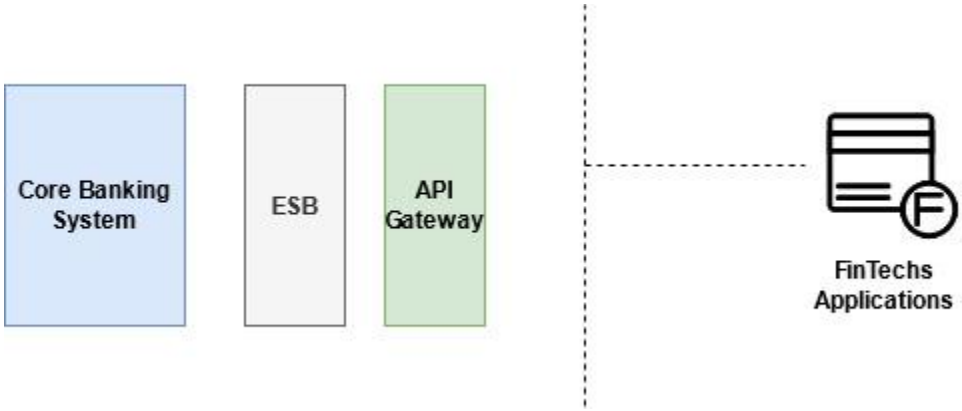

Figure 2.2 : Core banking FinTech integration

A core banking system of a bank, with the help of an ESB and API Gateway exposes banking services to external FinTechs. FinTechs provide these banking services with their lucrative mobile applications.

2.3 RegTech

The term RegTech refers to the use of cutting-edge technology to facilitate the delivery of regulatory requirements [1]. RegTech also can be defined as technological solutions for regulatory process [2]. There is a great deal of compliance requirements for financial institutions. When working with people's money to make money, which is a bank's core business, it is a must they must be regulated. Few mandatory regulatory concepts adhered by Sri Lanka banks currently are,

- Client onboarding – Know Your Customer
- Operational risk management
- Regulatory reporting
- Risk analytics

Know Your Customer

KYC is one of the most important regulatory requirements in banking. Whenever a new customer checks in to fulfill a banking need, almost all the details of that customer is recorded. From his personal details to previous loans, asset information, connections with other banks, all the information that is required accept him as a legitimate customer is recorded. This process is straight forward, when the customer walks into the bank physically and shares all the personal information. A bank staff can then and there immediately check and confirm the information and documents provided, and if there are any discrepancy, he can request the customer to validate. This process becomes complicated if the customer is being onboarded via an open API exposed by bank [5], through an Open Banking solution. To support open banking customer onboarding stern regulations must be in place and a RegTech solution might be an ideal choice.

Operational Risk Management

When a bank endeavoring in any business activity with a business entity, may it be a customer or corporate, it is crucial to assess and analyze the risk involved with that

business. The Basel committee on Banking and Supervision has described operational risk as the risk of loss resulting from inadequate or failed internal process, people, and systems or from external events [24]. With the advent of technology and digitization operational risk management becomes even harder since the artifacts are digitized and can be easily forged.

Regulatory Reporting

Regulatory reporting is an integral part of any financial institution. Bank being no exception must have reporting in standard formats (eg: IFRS) to gain customer and investor confidence. Any financial institutions financial reporting should be released to the public and should be up to the standard and transparent which will lure customer and investor confidence.

Risk Analysis

Risk should be analyzed in all possible ways. From when a large sum of money being transferred between banks to checking a legitimacy of a customer, risk is involved everywhere. Risk should be analyzed and always tried to be mitigated.

RegTech Evolution

Rather seeing as an intertwined phenomenon, RegTech and FinTech should be seen as separate entities. Though FinTech is a Financial services specific entity, RegTech has its vast applications and even can be extended to other industries such as insurance, healthcare as well. Possibilities include monitoring corporations' compliance with environmental regulations and real-time tracking of the location of airliners, to name few but two simple examples of how technology could be used to improve not only regulation but also the regulated industry itself. Also, different driving forces support the advancements of the RegTech itself. Primary reason for the growth of FinTechs can be attributed to startups and the distrust of financial institutions, the commoditization of technology, and the recent rise in unemployed professionals seeking new ways to apply their skill sets. RegTech is not

something new but is at a rapid phase of evolving. Because of revolutions in computer power and the decrease in cost in emerging tech [2] RegTech is also evolving. The RegTech evolution has been categorized into following three phases.

RegTech 1.0

During the 1990s and 2000s, when the technology forayed into industries the requirement for smart monitoring and compliance was realized. Little by little these practices became standard practices with are adhered by modern organizations even today in the form of audit reports, IFRS, AML etc.

RegTech 2.0

Know your customer has been the main theme in the past decade. Details were gathered, processed and analyzed of individuals making transactions. These details were then used to improve customer protection and eradicate bad behavior. More and more innovative technology was used in the compliance process.

RegTech 3.0

RegTech 3.0 is more of a concept than implementation at the current stage. Data has become so sparce analyzing this data is a hard task. The move from Know Your Customer to Know Your Data would have been impossible without the technology evolutions such as artificial intelligence, machine learning and big data technologies.

2.4 Stream Processing

Stream processing is the technology that allows users read in continuous data and detect anomalies as and then when they are received. This detection time may vary based on computing power from nano seconds to minutes [16]. There are various other names for stream processing such as real-time analytics, streaming analytics, complex event processing, real time streaming analytics and event processing. But all these terms umbrella

under a common name stream processing in the current context. Stream processing got popularized because of Apache Storm and big data technologies [20].

Stream input can come multiple different type data sources and will be continuous. All these data streams would be ingested to processing engine where a logical analysis will be performed, and filtration will take place. Based on an event defined for the filter the output can be either a steam or might be passed over to an analytical engine [23]. This concept very much suits when designing a RegTech solution, since insight should be derived within minimal time. The historical alternative to stream processing would be batch processing. Data is read into either databases, files or other forms of mass storage and processed in batches. Once a batch is processed based on a schedule next batch will be processed. Then will arise the problem of correlating entities between these processed batches. Streams on the other hand provide data driven insights on the go [26].

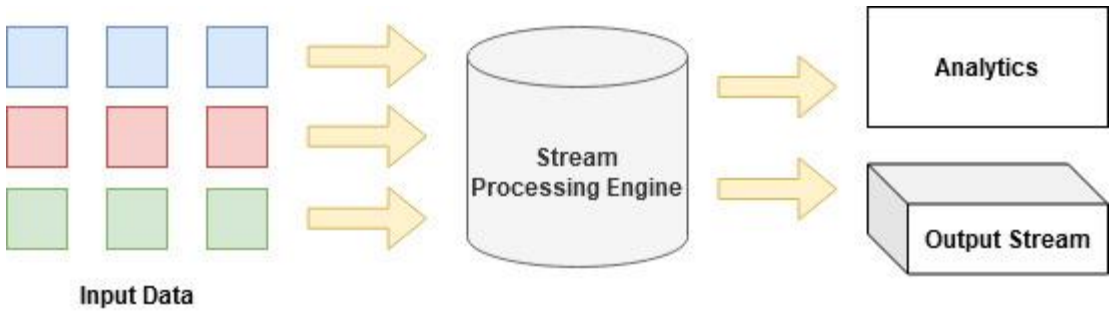

Figure 2.3 : Stream processing flow

Processed bigdata has it's own value and this value deteriorates with time. Also, this value is not equal between data segments. Some values are high at the time of receival, and some have significant value over time. How we make use of this data value helps us in so many ways and stream processing is great tool to achieve this. For example, if a high value customer enters a bank and requires special attention this insight only has value until he is in the bank premise, and dissolves once he leaves. Within that time period necessary action should be taken. Stream processing targets such scenarios. The key strength of stream processing is that it can provide these insights faster, often within milliseconds to seconds [20]. Following are some secondary reasons for stream processing.

- Some data naturally comes as a continuous never-ending stream of events. To process these as batches, you need to store it, stop data collection for some time and process the data. Then you have to deal with the next batch and then worry about aggregating across multiple batches. In contrast, streaming handles never ending data streams attractively and naturally. You can identify patterns, inspect results, look at multiple levels of focus, and easily look at data from multiple streams at the same time.
- Stream processing works well with time series data on pattern detection. For example, if we want to detect and analyses a pattern from user internet banking logging data in this example trying to detect a sequence pattern, stream processing can be an ideal choice, since whenever a user logs we have the steam data and with time we can detect patterns. Other examples pertaining time series data may be IOT data, sensor data, and live stream data and stream processing can be perfect fit in as a programing model to analyze these data.
- If we compare batch processing and stream processing batches accumulate continuous data and works on it, whereas stream processing works on data as and then it arrives. Biggest challenge of batch processing is once the batch is received there will be a time required to operate on the batch and within the period data may not be captured, or packet dropped. Further more stream processing allows load distribution of stream data and approximate calculations are made. This is very helpful when the requirement is to make decisions on approximate values rather exact values.
- Another issue arises when some data is very large and hard to operate on or might require heavy computational power. Stream data on the other hand works on small chunk of continuous data to overcome this challenge.
- Finally the increase of sources of streaming data. May it be vehicle sensor data, customer transaction processing data, or even mobile application data. Stream data will become a standard concept in the near future.

In our current use-case with DFCC bank all these regulatory operations are handled in data warehouse and in batches. The reasons mentioned above clearly indicates how stream processing is well suited for a RegTech solution.

2.5 Architecture Evaluation

Once we propose the stream processing-based architecture it is important to evaluate it with some standard evaluation methodology. There are so many methodologies to evaluate a proposed architecture. In this project we will evaluate the proposed architecture using following methodologies [21].

- Architectural Tradeoff Analysis Method (ATAM)
- Cost Based Analysis Method (CBAM)

The reason to take up these two methods is, while ATAM analyzes the architecture in a technical point of view, CBAM analyzes it in a cost centric view, two crucial factors for a success of a project, technical specificity, and cost.

Types of Architecture Evaluation

Briefly architecture evaluation can be categorized based on two types.

Technical: Evaluates architecture against quality attributes such as performance, security, modifiability, design sustainability. ATAM falls under this category.

Economic: Evaluates architectures based on a cost perspective, how cost will affect architecture decisions and how it will affect the overall project budget.

Benefits of Architecture Evaluation

Following are some benefits of conducting an architecture evaluation [22].

- Brings stakeholders together for the first time.

An architecture evaluation might be the first-time architect meets the responsible stakeholders and share the ideas which he had for himself all the time. All these times everything that went through the architect’s head will be shared and discussed by various

people. Their goals may have been conflicting with each other, this platform might provide an opportunity to talk and clear out these differences.

- Provides a clear understanding of quality goals.

An architecture evaluation clearly communicates what are the quality goals and how they will affect the responsible stakeholder during the life span of the software. These goals are usually not captured in the usual requirements, design documents. An early discussion at the point of submitting the architecture will help the stakeholders to be ready with what to be expected.

- Conflicting quality attributes can be prioritized.

During a software architecture evaluation there would be conflicting quality goals. For example, for a maintenance engineer adequate logging should span through the flow of transaction whereas a performance engineer logging would be a unwanted performance setback. An architecture evaluation will bring these two stakeholders together and come up with an agreeable level or at least prioritization. Ultimately project management might have the last word but hearing and understanding the concerns would end up in a better improved architecture.

- Improves the Quality of Architectural Documentation

Often at the stage of architectural evaluation there will be basically no finalized documents such as requirements documents, design documents, integration analysis document etc. During the discussions the need for these kinds of documents may arise. For example a performance engineer might require a no of pods related documentation which he might have not been captured during requirements discussions. During the evaluation this quality attribute can be identified and performance engineer can raise the documentation requirement hence the development teams will start prepared.

- Uncovers Opportunities for Cross-Project Reuse

A common mistake which architects usually makes is they prepare the architecture focusing on the current project only. An architecture evaluation may result in cross project reusable artifact since there will be a pool of experts who dealt with different situations in

multiple projects. Teams may spot components that can be either modularized or bundled which can be used in multiple projects.

- Results in Improved Architecture Practices

Architecture is a continuous practice and there is no concise architecture for any specific system. With experience an architecture evaluation can bring refined and best practices.

2.5.1 Architectural Tradeoff Analysis Method (ATAM)

In software development, there cannot be a perfect architecture. There should be tradeoffs between quality attributes [32]. Software architecture depends on the quality attributes we maintain. Quality attributes defined in ISO 25000 are as follows [13].

- Functional sustainability
- Performance efficiency
- Compatibility
- Usability
- Reliability
- Security
- Maintainability
- Portability

It is almost near impossible to build a system having all the above quality attributes satisfied. When architecting solutions these quality attributes should be tradeoff with one another. For example, if we expect extreme performance of the system, there might be some tradeoff between usability, since at one-point system might crash. If we expect the system to be available all the time, there might be tradeoffs between security. So once developing architectures there is a necessity of trading off quality attributes. ATAM analyses the architecture based what are the tradeoffs we can make based on quality attributes and come up with an optimal architecture. All these quality attributes are ultimately meant to achieve system goals. The purpose the system is to build and solve a

business problem. In a large and complex system following may be some issues during architecture evaluation [15].

- Large systems are hard and complex to understand hence hard to evaluate within limited time.
- Architectures can go through multiple cycles hence during each cycle the business goals must be supported, and architects might need to map between these goals and technical decisions.
- Typically, large systems involve multiple stake holders, and it will be near impossible to satisfy each and every one of them.

Participants of ATAM

ATAM requires three group of people working together [10].

- Evaluation team

Typically external members to the project. Might be from external organizations. They can be architecture experts but should have minimal knowledge of the current project. Consists less of than 5 experienced people.

- Project decision makers

These are members who speak for the whole project and drive the initiative. They have complete authority to manage the project and have the capability take necessary decisions and crucial time. Examples may be project managers, identifiable customers, business stakeholders

- Architecture stake holders

The actual stakeholders who will be involved in implementing the architecture to an artifact. Anyone ensuring the project achieves the quality attributes scalability, modifiability, security. For example, developers, testers, designers, maintenance engineers, etc.

Table 2.1 provides the participants and their desirable characteristics of an ATAM evaluation [12]

Table 2.1 ATAM participants

Role	Responsibility	Expected Characteristics
Team Leader	Initiates the evaluation. Main SPOC with stakeholders, ensures client's requirements are met; main person to establish an evaluation contract	Organized, good in dealing with clients, adheres to strict deadlines
Evaluation Leader	Conducts evaluation; identifies/elicits scenarios; Administers scenarios selection/ priorities	Very engaging with audience, good facilitation skills, knowledgeable in architecture issues
Scenario Scribe	Writes down scenarios on either white board or cards, organizes them, have the authority to extract the correct word depicting the scenario	Good handwriting, should stick to a discussion and not allow to move forward until a proper scenario is identified.
Proceedings Scribe	Scribe takes note of all the proceeding in electronic form either uses a laptop or workstation.	Faster typist, a stenographer might be better. Organized. Good knowledge on architectural matters
Timekeeper	Keeps time, makes sure evaluation leader is keeping focus on the topic, Helps manage time devoted to each scenario	Should be an authoritative person to call on time and interrupt.
Process Observer	Keeps track of the proceedings and can make suggestions on how the proceeding can be kept on track and	Thorough knowledge of the steps and process.

	focused. Usually doesn't comment on anything but if he feels the topic is going out of track should be able to make a stop	Guides the evaluation leader on matters
Process Enforcer	Supports evaluation leader of matters. Makes sure the steps are correctly adhered	Knowledgeable in architecture. Not afraid to raise concerns.
Questioner	Thinks about issues that any other stake holder hasn't thought of. Knowledgeable in similar domain.	Good with architecture knowledge. Experienced architect. Domain expert. Straight forward and thoughtful when raising questions

Outputs of an ATAM

Following are some typical outputs of an ATAM evaluation [9].

- Detailed architecture presentation which can last an hour.
- Articulation of clear business goals. For the some the architecture evaluation discussions are the place where they realize the business goals and might have contributed their views. There goals are documented.
- Prioritized quality attribute requirements and translated quality attribute goals
- Set of Risks and non-risks
    - Any undesirable outcome due to some quality attribute requirement can be considered as risk.
    - An architectural decision based on analysis, if it has no consequence then it can be considered a non-risk.
    - The risk identified will lead to a risk mitigation plan.

- Set of risk themes.

  Risk themes are risks that might popup during discussions. Evaluation teams will examine these risk themes and assess the impact for the project.
- Architecture decisions and quality attribute mappings.

  Any scenario that will impact the architecture decision will be documented.
- List of sensitivity and tradeoff points.

  What are the sensitivity points and want are the tradeoff that should be made will be documented.

Steps of Evaluation Phases

Table 2.2 describes the phases of an ATAM [11].

Table 2.2 ATAM evaluation phases

Phase	**Activity**	**Participants**	**Duration**
Phase 0	Partnership and preparation	Evaluation team + Project decision makers	Few weeks
Phase 1	Evaluation	Evaluation team + Project decision makers + Architect	1-2 days
Phase 2	Evaluation	Evaluation team + Project decision makers + Architect + Stake holders	2 days
Phase 3	Follow up (prepare report)	Evaluation team	1 week

Steps of an ATAM

Step 1

The ATAM is presented so all the participants are familiar with the process.

Step 2

Business drivers are presented. Expected understanding of business goals and priorities from all the participants. This step ensures all the core basics are clear to all the participants.

Step 3

Architecture is presented and participants are expected to be familiar with the system and process. A brief review of at least one module or component is thoroughly analyzed. Simple connector view is presented, and a couple of scenarios are traced and analyzed.

Step 4

Possible architectural approaches are identified. For certain quality attributes all the views from participants are gathered and brought to the attention of architect. This can be done as part of step 3 also.

Step 5

Populate quality attribute tree. During requirements elicitation, previous evaluations scenarios might have been identified. These are gathered and the quality attribute tree is populated.

Step 6

Architectural approaches are analyzed. Scenarios are ordered and ranked according to importance. Might consume so much time hence the role of timekeeper is heavily required during this step.

Step 7

Scenarios are brainstormed and gathered stakeholders are expected to involve and contribute to the discussions.

Step 8

In case of a no clear conclusion Step 6 can be reworked.

Step 9

Everything discovered, risks, quality attributes, risk themes, non-risks trade-offs, are reviewed.

Conceptual Flow of ATAM

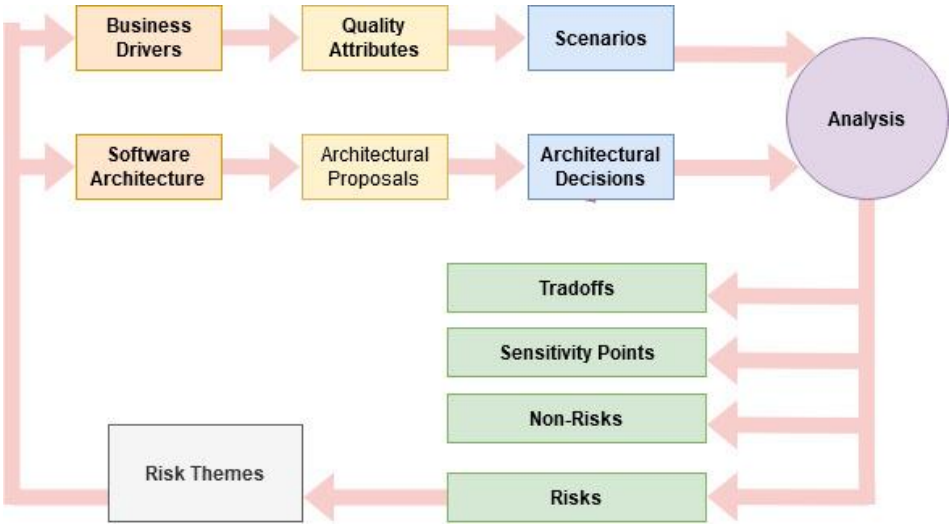

Figure 2.4 : ATAM conceptual flow

2.5.2 Cost Based Analysis Method (CBAM)

ATAM misses an important aspect of any project. The biggest trade off in a large complex software project, cost. CBAM provides a technical and economical overview encompassing the project. CBAM is an architecture centric method for analyzing the costs, benefits, and schedule implications of architectural decisions. CBAM is different from ATAM where it adds cost as its' quality attribute [15].

Context of CBAM

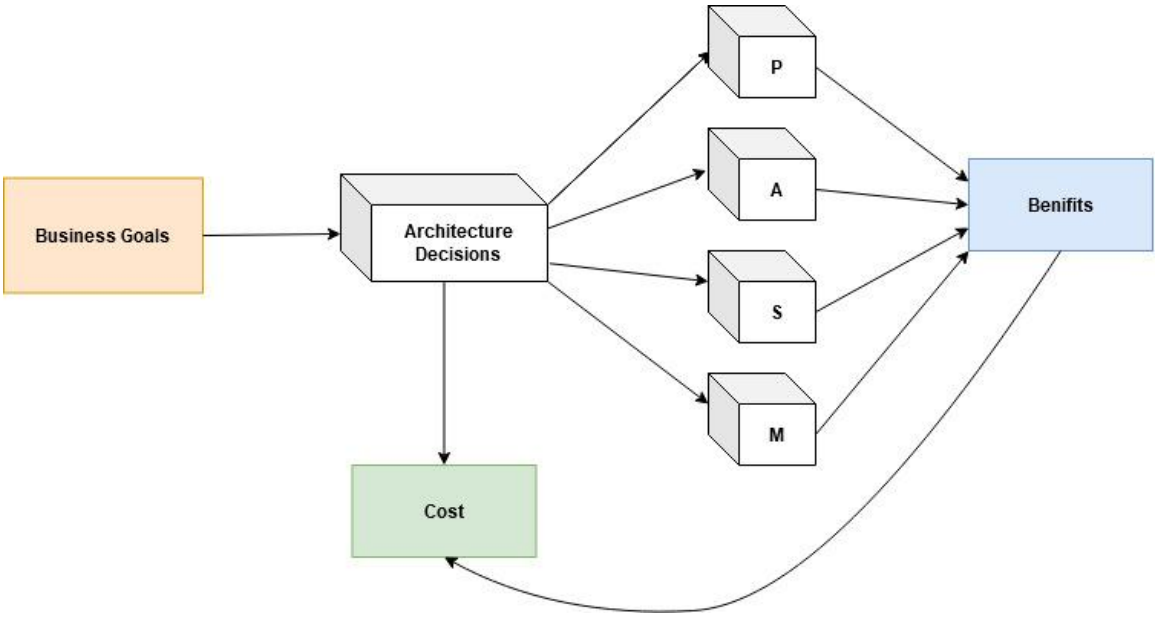

Figure 2.5 : Context of CBAM

Here P represents Performance, A represents Accessibility, S represents Security and M represents Maintainability.

Pre-Requisites and Inputs for CBAM

- Presentation on business goals.
- Architecture decisions and any trade-offs identified in previous ATAM sessions.
- Expectation level of quality attributes and all the economical constraints [9].

Process Flow Diagram for CBAM

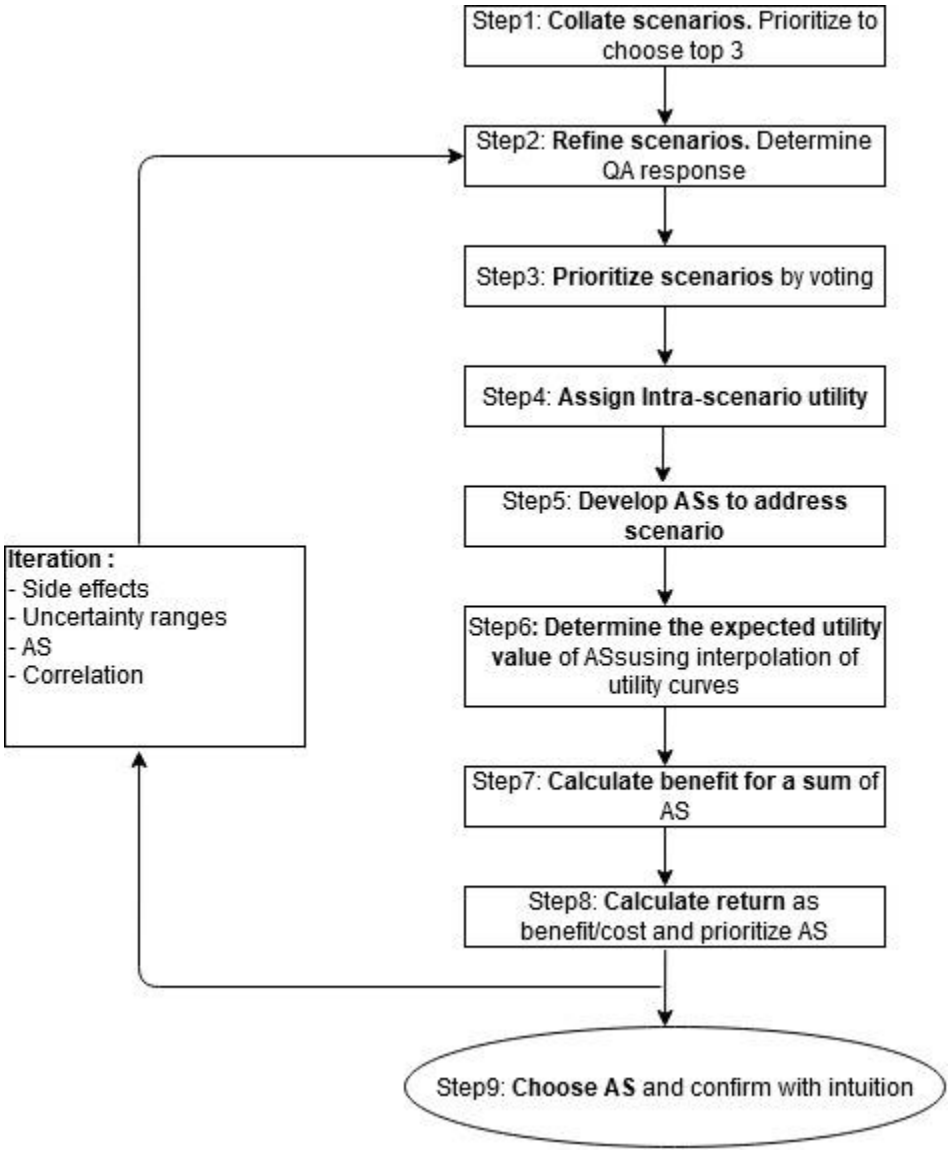

Figure 2.6 : CBAM - Process flow diagram

Implementation of the CBAM

Step 1 – Collect and combine scenarios.

Scenarios identified during ATAM are collected and combined. Stake holders are given a chance to provide either new ones or to refine existing ones.

Step 2 -Refine scenarios

Scenarios identified from step 1 are discussed and refined.

Step 3 – Prioritize scenarios

Important scenarios are prioritized.

Step 4 – Assign utility

Quality attributes are assessed whether it is a worst case or best case.

Step 5 – Develop architectural strategy.

An appropriate architectural strategy is taken up and discussed.

Step 6 – Interpolate expected levels of quality attribute response level.

Step 7 – Total benefit calculation

Step 8 – Based on ROI, schedule constraints adopt the correct architectural strategy.

Step 9 – Confirm results with intuition.

CBAM is an iterative process and based on intuition [14]. The result might vary in case of the experience and expertise of the evaluation team. Also, an important point to take note is whether all the decisions taken during these sessions are aligning with organizational and business goals [11].

CBAM is an economic based approach to decide architecture decisions. Assume benefit of an architectural strategy Bi (weighted and summed individual utility of strategy) and cost of architectural strategy is Ci then the return of investment Ri of the strategy would be

$$Ri = \frac{Bi}{Ci}$$

3 METHEDOLOGY

This chapter will try to understand the current landscape of RegTech in Sri Lankan banks. It will also discuss about stream processing technologies. Once a suitable technology is identified, we will try to implement few RegTech use-cases which are commonly used in a bank. The proposed architecture will be also validated using ATAM and CBAM architecture evaluation methodologies.

3.1 RegTech in Sri Lankan Banks

Without mentioning the term, RegTech has been part of all the Sri Lankan banks form the early days. Regulatory compliance is strictly imposed to Sri Lankan Banks via Central Bank of Sri Lanka (CBSL). Banks deploy millions worth monolith systems in the back end and monitor transactions, generate regulatory reports and adhere compliance. We will analyze three banks, on how the regulatory requirement is achieved.

3.1.1 DFCC

Established in 1955 as Sri Lanka's pioneer financial institution with the backing of world bank, DFCC is one of the oldest banks in Sri Lanka [38]. In the recent years DFCC has massively invested in their core banking digital transformation project and revamping their whole systems architecture. Also, they are looking forward to foray in the open banking space with the help of IBM App Connect Enterprise ESB and IBM API Connect API Manager. Figure 3.1 illustrates the DFCC bank's current systems architecture diagram.

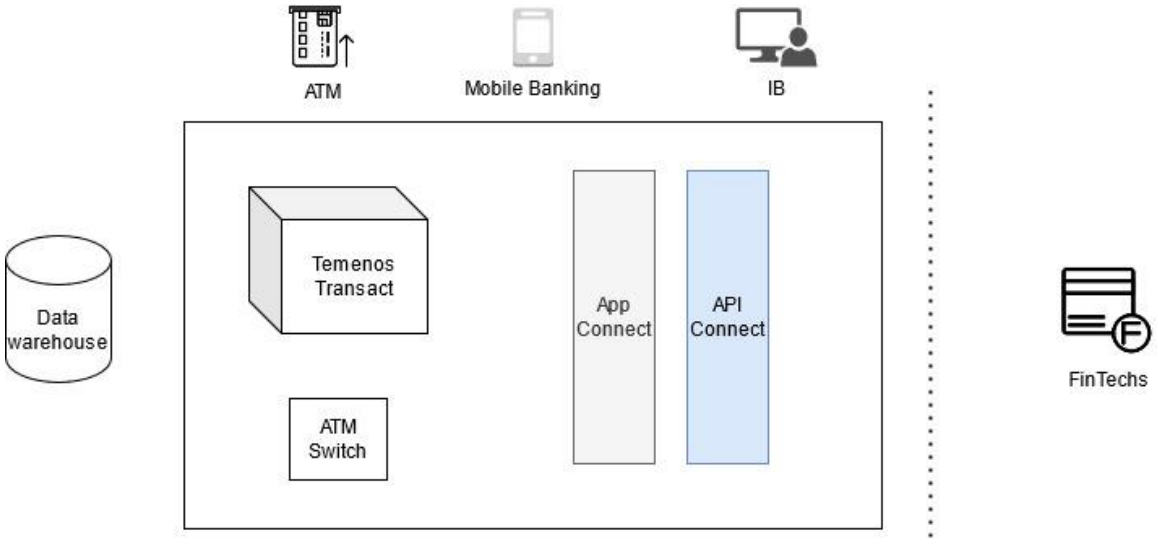

Figure 3.1 : System architecture - DFCC

DFCC has a Temenos Transact core banking system. There in the middle of bank transformation process with adding an IBM App Connect Enterprise ESB layer to interact with external system and implementing an IBM API Connect for open banking initiative. Most of their regulatory implementation happens through either by data warehouse or Financial Crime Mitigation module which is an integral part of the Temenos Core Banking system suit.

DFCC Back-end RegTech Solution – FCM

Temenos' Financial Crime Mitigation (FCM) product family enables banks and financial institutions to avoid regulatory fines, detect fraud and mitigate reputational risks whilst improving throughput and optimizing total cost of ownership all in line with the banks' risk mitigation Approach [30]. This requires a separate license which involves a high cost. FCM can be deployed in its entirety or partially to cover only specific business needs. Customers can then add capability as these needs change or evolve with time. DFCC as part of their core banking system use FCM as their RegTech solution.

3.1.2 Bank of Ceylon

One of the largest and oldest state-owned banks in Sri Lanka, Bank of Ceylon is a market leader in terms of profit, assets, and customer reach [40]. Being one of the respectable entities in banking, BOC suffers the lack of modern banking architecture. Though in recent times they have heavily invested in application modernization and digital transformation with implementing IBM App Connect Enterprise and IBM API Connect. This revolution has come bit later than its competitors. Figure 3.2 illustrates the high-level architecture diagram of BOC systems architecture.

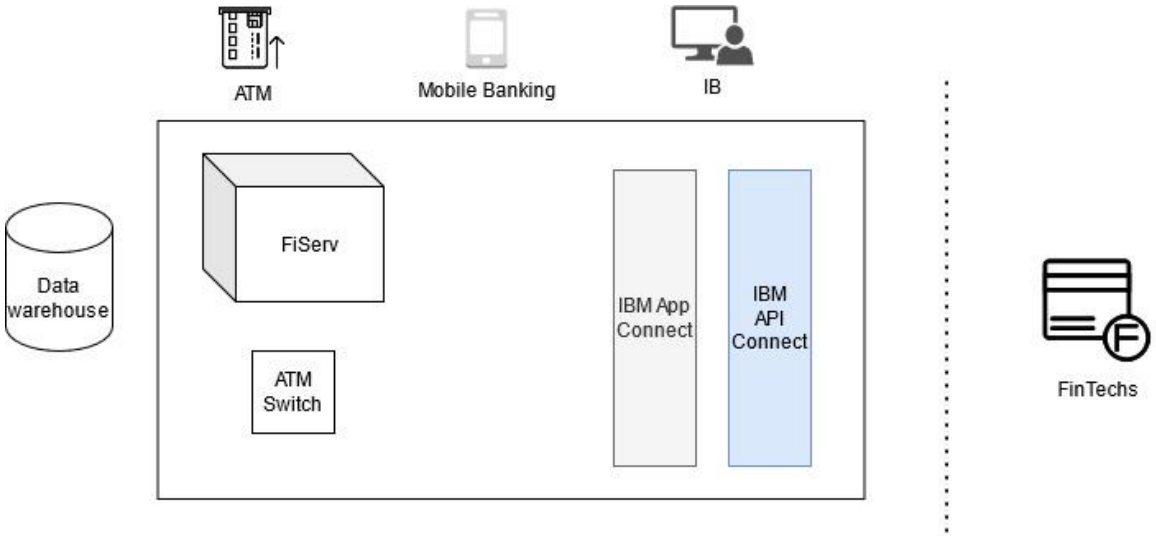

Figure 3.2 : Systems architecture - BOC

Back-end RegTech Solution – Cari5

Clari5 is the BOC's back end RegTech system [31]. This is a monolith system which sits in the back end and acts as a monitoring engine. All the core banking transactions flow into the system and based on rules configured transactions are monitored. Clari5 is an intuitive solution that provides real time monitoring of transactions. Can be spanned to multiple channels and able to work with AML, fraud management, user experience management. It employs AI, ML based models to provide these insights.

3.1.3 Hatton National Bank

HNB is the second largest commercial bank in Sri Lanka, having the largest retail customer base than any other private bank [39]. Processing millions of transactions per day, HNB is a technology savvy bank in comparison to other banks. The only bank implementing an API manager and in the state of readiness for open banking, HNB can be regarded to it's openness and innovativeness.

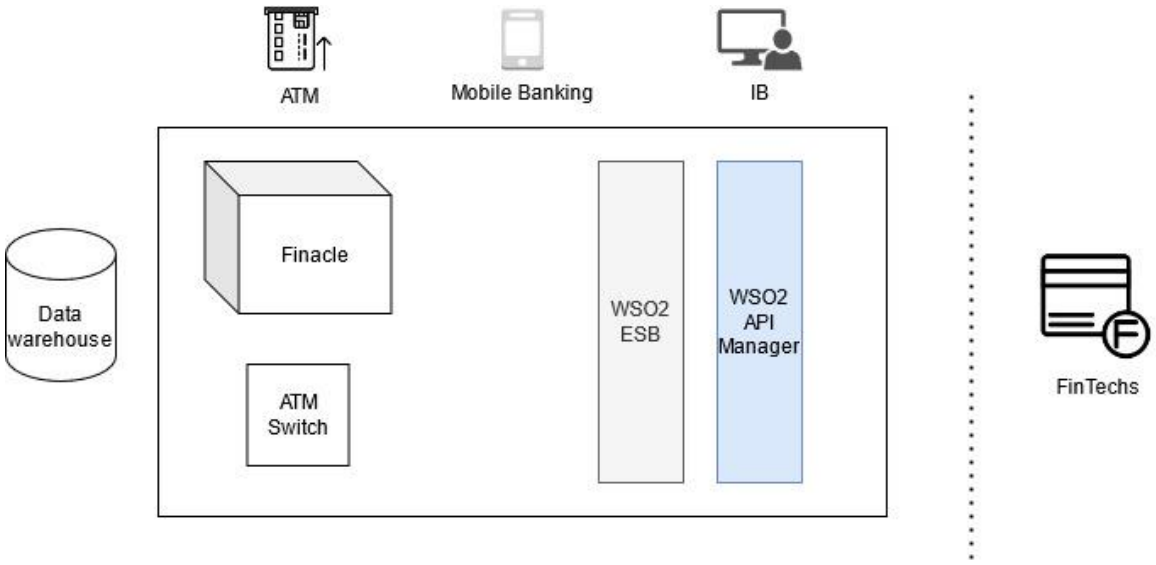

Figure 3.3 : Systems architecture - HNB

As the diagram illustrates HNB mostly relies on Data warehouse and Security Information and Incident Manager (SIEM) to implement regulation and compliance.

Back-end RegTech Solution – SIEM

Security Information and Event Management (SIEM) is a set of tools and services and provide an overall view of organization security.

SIEM provides,

- Real time visibility on all the organization security system.
- Works well with event logs and management.

- Takes in necessary details from logs, correlates and builds insights based on rules.
- Provides an elegant dashboard for notifications, and advanced notification features via TCP, Email.

3.2 DFCC for Implementation Choice

According to the details gathered and analyzed on the architectures and systems, DFCC was selected to implement the use-cases and perform architecture analysis due to the following reasons.

- DFCC is in the verge of digital transformation. While DFCC core banking system is being completely revamped, third party systems are also upgraded, and an open API initiative is taking place.
- BOC had an older version of Fiserv core banking system and the digital transformation journey would be hard to implement hence incorporating stream based RegTech solution would be hard to implement.
- HNB recently updated their Finacle core banking system to the latest version, and they have already invested and completed on most of the digital transformation investments.

Hence for the above reasons DFCC was considered as a suitable candidate to experiment the solution.

3.3 Stream Processing Technology Analysis

Since the proposed solution architecture involves a stream processing solution the analysis was done to identify a suitable stream processing solution.

- WSO2 Stream Integrator
- IBM Streams
- Apache Flink
- Azure Stream Analytics

3.3.1 WSO2 Stream Integrator

WSO2 Streaming Integrator includes a stream flow designer and a stream processing engine with strong monitoring and analytics functions [26]. The streaming runtime capabilities enable users to treat all data sources as streams of data, apply stream processing operations, and publish to one or more destinations. It also comes with advanced error handling and correction [34]. WSO2 Streaming Integrator allows you to implement streaming ETL, change data capture (CDC), and process large files and real-time APIs which makes it a good choice for implementation at DFCC.

3.3.2 IBM Streams

IBM Streams is a software platform that enables the development and execution of applications that process information in data streams [27]. IBM Streams enables continuous and fast analysis of massive volumes of moving data to help improve the speed of business insight and decision making. IBM Streams consists of a programming language, an API, and an integrated development environment (IDE) for applications, and a runtime system that can run the applications on a single or distributed set of resources. The Streams Studio IDE includes tools for authoring and creating visual representations of streams processing applications. It has an embedded Zookeeper, which takes out the external hassle of configuration and maintaining during scaling.

3.3.3 Apache Flink

Apache Flink is an open-source, unified stream-processing and batch-processing framework developed by the Apache Software Foundation [28]. The core of Apache Flink is a distributed streaming data-flow engine written in Java and Scala. Flink executes arbitrary dataflow programs in a data-parallel and pipelined (hence task parallel) manner. Flink's pipelined runtime system enables the execution of bulk/batch and stream processing programs. Furthermore, Flink's runtime supports the execution of iterative algorithms

natively. Flink provides a high-throughput, low-latency streaming engine as well as support for event-time processing and state management. Flink applications are fault-tolerant in the event of machine failure and support exactly once semantics. Programs can be written in Java, Scala, Python, and SQL and are automatically compiled and optimized into dataflow programs that are executed in a cluster or cloud environment.

3.3.4 Azure Streams

Microsoft Azure Stream Analytics is a serverless scalable complex event processing engine by Microsoft that enables users to develop and run real-time analytics on multiple streams of data from sources such as devices, sensors, web sites, social media, and other applications [29]. Users can set up alerts to detect anomalies, predict trends, trigger necessary workflows when certain conditions are observed, and make data available to other downstream applications and services for presentation, archiving, or further analysis.

3.4 WSO2 Stream Integrator for Implementation Choice

Discussion with DFCC team resulted that they are more into an open-source solution since they have massively invested on other software and systems. Also, a fact that DFCC had a WSO2 Identity and Access Manager solution in place and the development teams were familiar with the WSO2 platform. Another reason was WSO2 Stream Integrator was massively scalable with the help of Apache Kafka. Due to this reasons WSO2 SI was selected to develop the prototype solution. Since DFCC had already purchased an IBM product, IBM App Connect Enterprise and was in the process of procuring IBM API Connect, they were aware of the capex which needed to be done if the selection was IBM Streams. There are several regulatory constraints when implementing a cloud-based solution hence Azure was not considered. Apache Flinx was a strong contender for the final decision but due to the DFCC's familiarity with WSO2 and project implementation time constraint WSO2 Stream Integrator was ultimately selected.

3.5 DFCC – Proposed RegTech Solution Architecture

Below will be a proposed solution architecture involving WSO2 stream integrator. Since the scope of this work bounds only with APIs of an Open Banking solution, all the provisioned APIs exposing to the external parties would be flowing through stream integrator. Stream integrator would online real time analyze these API details based on rules devised by DFCC operations, regulatory and compliance teams. We will discuss few use cases in this chapter on the analysis methodology.

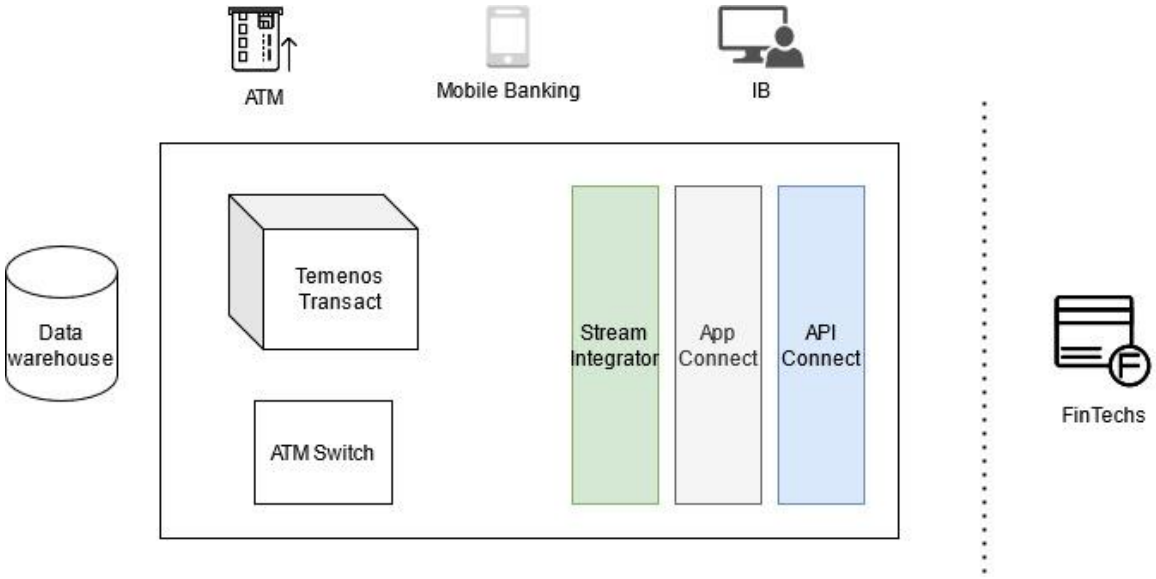

Figure 3.4 : DFCC - Proposed solution architecture

3.5.1 RegTech Use-case 1 – KYC

When a customer checks in to a bank for either to open an account or apply for a loan, basically for any banking need a detailed information gathering is done of that customer. This is known as Know Your Customer (KYC). This process will ensure the customer who is requesting the service is legitimate. This process is straight forward when the customer visits the bank physically and submits, declares details. The same process becomes cumbersome when the onboarding takes place via an API integrated to a FinTech. Bank opens an API to a FinTech, the FinTech consumes the API and allows an external user to

on board a banking service before allowing the FinTech specific service. At this point an imposter can try to misuse the service in the following possible ways.

- Make multiple requests with the same details.
- Same customer using a different address to get onboard multiple times.

These can be considered anomalies and should be thwarted or at-least altered.

3.5.2 RegTech Use-case 2 – Risk Analysis

There will be lots of risk involved when opening an API to external parties especially FinTechs. Transactions will involve great amount of risk. If an imposter gets access to a fund transfer API then it may lead to unprecedented consequence. Transactions can be basically categorized as follows.

- Fund transfer

A transaction which moves fund from one account to another. The second account can be the same user's bank account, account from the same bank or account from another bank.

- Bill payment

Banks usually partner with billers and allow customers to make payments for these billers. The from account would be the customers account and the to account would be the billers collection account.

When banks open up a fund transfer APIs a risk might be involved on whether the transaction is legitimate. What if, within a small time period multiple transactions take place for a single from and to accounts. DFCC should place a proactive RegTech solution to identify these kind of anomalies.

3.5.3 RegTech Use-case 3 – Audit Reporting

During analysis and discussions, it was understood that when processing loans analyzing the CRIB report is a tedious process. Credit Information Bureau (CRIB) is a separate independent entity in Sri Lanka which maintains and manages credit information of institutions and individuals [35]. If a person takes a loan from a bank or financial institution

that information is kept and tracked at the Credit Information Bureau. This information on request by other banks are shared between banks to so that the next time the same customer applies for another loan in a different bank his credibility if checked via CRIB report [36]. Currently CRIB has provided different types of API which provides the loan details of a customer if called. This API has following types.

- Single hit API

If we know the exact identification detail say national ID no, passport no this API can be used.

- Multi-hit API

If there is an ambiguity in the customer identification say only a partial name is known, if the NIC no is not clear, this API can be used.

Once called, these APIs will return list of loan details. The important point here is for every single API call there will incur a charge which CRIB imposes to banks.

This use-case we will call the API and make use of the stream processing engine to store the data in a database. This will help us to avoid unnecessary cost by calling the API multiple times. The data fetched from the CRIB will be stored in a database with a flag. This flag would indicate the time period the data may be valid and is configurable. This is based on the assumption that it's unlikely for a customer to take multiple loans within a specified time interval.

4 ARCHITECTURAL ANALYSIS

This chapter will discuss architecture evaluation methodologies ATAM and CBAM and how these methodologies were applied at DFCC use-case.

4.1 Architecture Evaluation

Since we are proposing a new architectural solution, it can be evaluated compared with the existing solution. An evaluation can either be qualitative or quantitative. Qualitative analysis tends to measure something by its quality rather than quantity which is a conscience-based approach. During a qualitative analysis we usually explore how we describe something. Like a quantitative analysis this does not deal with numbers. When we do qualitative work, we work with descriptions. We work with feelings, thoughts, perceptions. We attempt to understand motivations and behaviors. Quantitative analysis is the opposite; to measure by quantity rather than quality. When we do quantitative analysis, we are exploring facts, measures, numbers, and percentages. When we do quantitative work, we work with numbers, statistics, formulae, and data. Hence architectural analysis is purely conscience based methodical approach. These methods are outlined and refined by Carnegie Melon University and is a widely accepted technique.

4.1.1 Evaluation Participants

Following participants were involved in the architecture evaluation.

- An external team of architects from N-able Private Limited
    - Architects/Leads from N-able Private Limited
- Project team members
    - Project Manager from DFCC
    - Business users from DFCC
- Architecture stake holders
    - Developers from DFCC

- Testers from DFCC
- IT Operations staff from DFCC
- Integration team from DFCC

4.2 ATAM for Proposed Solution

Phase 0 - Activities

The proposed architecture was presented within teams and a feedback was received. The system was described in detailed so that everyone had a clear understanding on the change to be done. There was a decision made, whether to go or no-go on the proposed solution. The statement of work for the whole ATAM was negotiated. Then the core evaluation team was formulated comprising architects and team leads. Then the architecture was reviewed.

Phase 1 – Activities

Evaluation leader described the evaluation method to members, tried to set their expectations and answered questions.

Following questions were raised during the session.

- What is the impact of performance when placing the streaming solution between the Core Banking System and the API manager?
- What is the cost deviation model for the proposed solution?
- What are the hardware requirements for including the new software?

Then the DFCC Project Manager described about the primary business goals motivating this initiative and what are the architectural drivers. It was a common agreement within the group this should be a cost-effective architectural solution.

During this phase following matters were discussed.

- A description of business environment, history, market differentiations, driving requirements, stakeholders, current need, and how the proposed system will meet those needs.
- A description in business constraints.

    - Parallelly running the implementation with the Core Banking transformation
    - Regulatory push from CBSL
- A description of technical constraints.

Based on these understanding and incorporating them an architecture was presented addressing how it represents the business drivers. This presentation covered the following information.

- The driving architectural requirements, the measurable quantities associated with these requirements
- Major functions, key system abstractions
- Subsystems modules and layers
- Hardware involved including CPU
- Architectural issues and risks.

Based on the discussion outcomes an architectural approach was identified. The ideal approach agreed by all members was to place the solution in between the CBS and API manager. Ultimately a quality attribute tree was generated.

Phase 2 – Activities

Phase 2 starts with the preparations for phase 2. During the session team leader address the session on the following points.

- Augment the evaluation team if necessary, by a questionnaire.
- Re-iterate the scope of the system being evaluated.
- Share the utility tree generated.
- Participants are allowed ask questions and brainstorming took place.
- Brainstorm the prioritized scenarios.
- At the end the results were presented.

Phase 3 – Activities

Phase 3 would be a follow up phase. The steps of the follow up phase are

- Produce the final report.
- Held the postmortem meeting.
- Build the portfolio and update the artifact repositories.

4.2.1 Utility Tree for the Proposed Solution

One of the outputs of an ATAM analysis would be utility tree. This is a structured tree that describes all the quality attributes and its constraints. During discussions, these attributes would be identified, and a voting would be held on which attribute can be trade off to which attribute. For example, during the discussions at DFCC following quality attribute tradeoffs were identified.

- The implementation and introduction of a streaming service in between the current solution core banking system and API manager should not affect the API processing time. If there is a delay, the limit should be less than 5ms, otherwise solution should be not approved.
- Regardless of integrating solution, the streaming service should work with full compatibility with core banking system.
- The streaming solution should be developer friendly and internal teams should quickly learn and able to make changes for future requirement changes.
- The payload passing through the streaming service should not be tampered, encoding, byte sequences should not change.
- Security should be at higher standards. Data should be consistent and data integrity should be maintained.
- Introduction of the streaming solution should not put an overhead on operations.
- Availability of the streaming solution is expected at 99%. In case of failures there should be fail over mechanisms, eg: clusters.

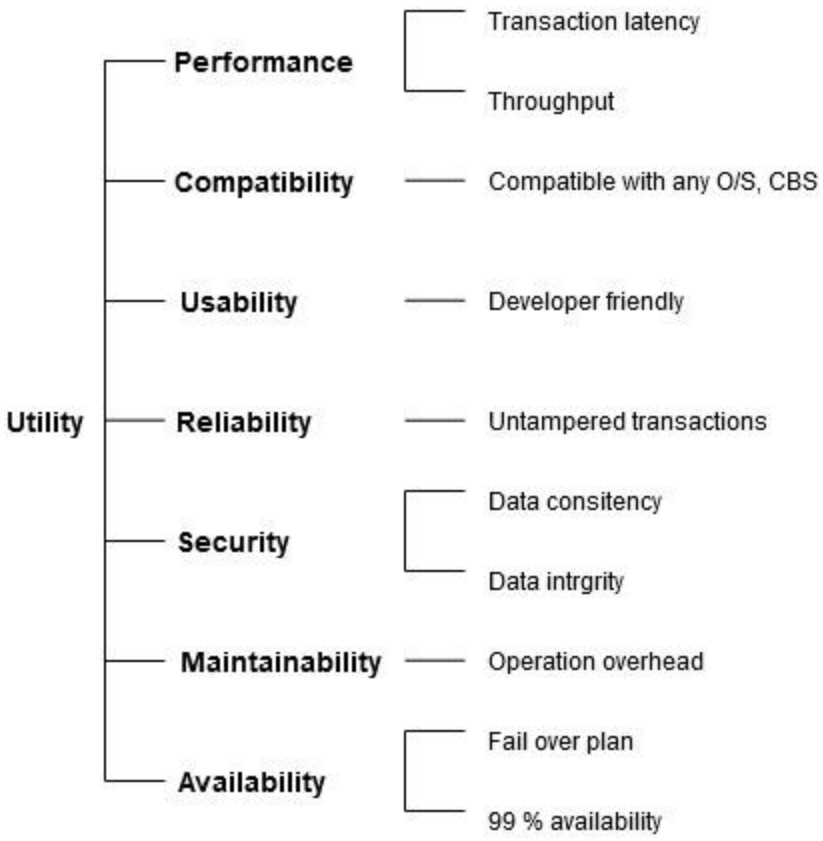

Figure 4.1 : ATAM - DFCC utility tree

4.2.2 Report Summary

Below is the summery of the report complied after the ATAM.

DFCC is in the process of introducing the Stream Based RegTech architecture to cater their regulatory and compliance needs. For this purpose, this ATAM session is conducted. After multiple discussions WSO2 Stream Integrator was identified as a suitable candidate to implement the solution. Since the Core Banking transformation project was also parallelly going on it was decided to take up the implementation straight away. Due the open source nature of WSO2 Stream Integrator it was understood this solution will be cost effective. But on the other hand we had make a trade-off between security. Solution was agreed to get the WSO2 annual support in case of an issue or a threat. Another risk identified was

the performance impact to the Core Banking API due to the SI introduction. To overcome this a detailed test with and without the solution, with and without TLS were tested and complied. Business driver for this project was to introduce a cost effective RegTech solution. Following table provides the architectural tradeoffs identified and how they will be handled.

Quality Attribute Trade-Offs

Following are few quality attribute trade-offs negotiated during sessions.

Table 4.1 : DFCC quality attribute trade-offs

Quality	**How it is achieved**	**Tactics proposed**
Availability/ Reliability	Redundant processors Networks Databases Load balancing	Active redundancy Transactions Introduce concurrency
Security	Firewalls Dedicated user Identity access management solution (Single sign on)	Limit access Being undivided Limiting the exposure Ensuring secure access once the API accessed and sessions are established.
Modifiability	Achieved by separation of API functions Database architecture Disperse business logic into multiple tiers	Encapsulate common services logical coherence Intermediary Stable interfaces
High performance	Load balancing Network address translation	Introduce parallelism

	Proxy servers	Increase available resources Multiple copies

Risks

Following risks were identified during the sessions.

- Implementation of a streaming solution in the middle of API manager and Core banking system should not affect the API response time
- Even if the streaming solution fails the API's should be available and accessible
- Implementation of security mechanisms such as TLS should not degrade API response times.
- The solution should be compatible and independent from operating system and core banking system

4.3 CBAM for Proposed Solution

The CBAM analysis was held in the following manner. Scenarios were collated from ATAM. The ATAM final report was used for this purpose. According to the business goal scenarios were prioritized. The top three were chosen. The scenarios were refined and focusing on their stimulus-response measures the quality attribute elicitation was done. Worst-case, current, desired, best-case. The grouped scenarios were then again prioritized, stake holders gave votes based on desired response values. The top 1 and 2 were selected and given a weight 1. The pending scenarios were assigned values based on their votes received. Finally a list of associated quality attributes were generated. Then the utility for each quality attribute was determined. Then architectural strategies were laid out for quality attribute levels. We developed architectural strategies for the chosen scenarios and determine the quality attribute level. Based on this the expected utility value was determined using interpolation. The total benefit obtained from an architectural strategy

was calculated and a suitable strategy was chosen based on return of investment. The final results were confirmed with intuition.

CBAM implementation at DFCC

Step 1

First step of CBAM is collating scenarios. As a sample we take following scenarios which had high priority.

Table 4.2 : CBAM step 1 - Collate scenarios

Scenario	**Scenario Description**
1	Introduction of stream processing solution should not hinder API performance
2	In case of stream processing solution failure, the core banking system API should not fail
3	Introduction of the streaming solution should not bypass security
4	The stream processing solution should have minimal footprint

Though there were many system related scenarios were discussed, above four were prioritized and filtered due to the relevance of streaming solution.

Step 2

Refine scenarios. Focus on stimulus response measures. We elicit worst-case, current, desired and best-case response levels for each scenario.

Table 4.3 : CBAM step 2 - Refine scenarios

Response Goals				
Scenario	Worst	Current	Desired	Best
1	0.1%	0.03	0.05%	0%
2	15 mins	5 mins	1 min	10 Sec

3	10	5	3	1
4	1%	0.5%	0.01	0.0001%

Step 3

Prioritize scenarios. Each stake holder is given 100 votes for each scenario. Top scenario is assigned weight 1.0 and the rest a weight value relative to that.

Table 4.4 : CBAM step 3 - prioritize scenarios

Response Goals					
Scenario	Votes	Worst	Current	Desired	Best
1	75	0.1%	0.03	0.05%	0%
2	50	15 mins	5 mins	1 min	10 Sec
3	100	10	5	3	1
4	75	1%	0.5%	0.01	0.0001%

Step 4

Assign utility scores

Table 4.5 : CBAM step 4 - assign utility scores

Utility Scores					
Scenario	**Votes**	**Worst**	**Current**	**Desired**	**Best**
1	75	10	10	60	100
2	50	0	0	85	100
3	100	10	40	80	90
4	75	0	20	90	100

Step 5

Develop strategies.

Table 4.6 : CBAM step 5 - develop strategies

Strategy	**Name**	**Scenarios Affected**	**Current Response**	**Expected Response**
1	Failover	1	10 min	1 min
		2	5 min	1 min
		4	0.1%	0.001%
2	Redundancy	1	5 min	1 min
		3	1%	0.004%
3	Optimal performance	4	2%	0.1%

Step 6

Determine utility.

Table 4.7 : CBAM step 6 - determine utility

Strategy	**Name**	**Scenarios Affected**	**Current Utility**	**Expected Utility**
1	Failover	1	10	60
		2	0	100
		4	20	90
2	Redundancy	1	40	95
		3	20	10
3	Optimal performance	4	20	85

Step 7

Calculate benefits

Table 4.8 : CBAM step 7 - calculate benefits

Strategy	Scenarios Affected	Scenario Weight	Benefit	Normalized Benefit	Total Benefit
1	1	75	50	3750	
	2	50	100	5000	
	4	75	70	5250	14000
2	1	100	55	5500	
	3	75	-10	-750	4750
3	4	75	65	4875	4875

Step 8

Choose strategy

Table 4.9 : CBAM step 8 - choose strategy

Strategy	Strategy Cost	Total Benefit	Strategy ROI	Strategy Rank
1	200	14000	70	2
2	1200	4750	396	3
3	40	4875	12186	1

This will be the ranking of design decision with respect to ROI. This will provide support for structured decision making. Some side effects of this method may be

- Discussion of costs and benefits
- Clarifications of scenarios and requirement

5 IMPLEMENTAION

This chapter will discuss on implementation strategy for the solution. IBM App Connect will be used for the purpose of mocking the bank APIs and few RegTech use-cases will be implemented using WSO2 Stream Integrator.

5.1 WSO2 Stream Integrator

For this prototype implementation work WSO2 Stream Integrator (WSO2 SI) was selected. WSO2 SI an open-source stream processing engine and widely used by many organizations. WSO2 SI can be used in following context for RegTech Analysis.

- Real Time ETL – Extract, transform, load real time data. This can be used if data from certain system need to be cleansed for compliance analysis purpose by another system.
- Work with streaming messaging systems – Streaming messaging systems can provide real time alerts on detection of anomalies.
- Streaming data integration – SI can help to integrate with various streaming systems if required to scale or add more functionality.
- Execute complex integrations based on streaming data – The SI CEP engine provides great proven support for this functionality.

Since the scope of the projects limits to open API banking the exposed core banking API endpoints will be routed to the WSO2 SI for processing.

WSO2 SI comprises of following components.

- A server where the stream processing takes place.
- Tooling component where we can write and deploy processing logic to streaming server. Stream processing logic can be written using Siddhi streaming SQL.

The main processing engine of WSO2 SI is Siddhi which has following main components [33].

Stream - Is a sequentially ordered list of events which take place over time, can be identifiable via a unique name, contains a set of defined attributes and schemas.

Event - Belongs to a single stream, there can be many identical events which belong to a single stream, an even is uniquely identified by a timestamp and ordered in chronological order.

Source – Is where the data in generated, can be either TCP, Kafka, HTTP source. These generated data or basically an event is then converted to a Siddhi event and sent for stream processing.

Sink – Is where the data is sent to an external endpoint in a predefined format such as XML, JSON, binary from a Siddhi event. This can be either a TCP, Kafka, HTTP sink.

In our use case, stream would be continuous JSON messages over HTTP from the API and an event would be a single API JSON message. Source would be an external system that is consuming the Open Banking API and sink would be a log entry. For this prototype use case we will be only logging as a sink. In a real-world implementation this sink can be a database entry or another streaming solution.

Tools and Technologies used for Implementation

Table 5.1 Tools and technologies used for implementation

IBM App Connect Enterprise	ESB to build mock bank web services
MySQL	To store CRIB data
Postman	Testing tool for REST APIs
SOAP UI	Testing tool for SOAP web services
WSO2 Stream Integrator	Streaming solution
WSO2 Tooling	Streaming application building tool
Siddhi	Event processing engine

5.2 Solution Architecture

Whenever an API is called the API JSON payload, will flow through the WSO2 SI where the processing engine will do the necessary processing logic. This processing logic will be written in the form of Siddhi queries.

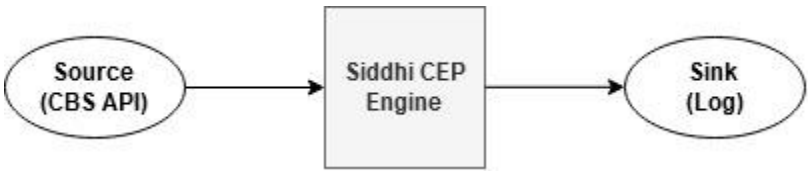

Figure 5.1 : Siddhi block diagram

5.2.1 RegTech Implementation 1 – KYC

At DFCC, KYC is handled by a Temenos Transact Core Banking system, get customer API. This API will have all the details pertaining to prospective customer. This API will have a JSON payload with customer details while being called. We can use WSO2 stream Integrator to write Siddhi queries to handle processing logic. Once called the API event will be routed to WSO2 SI. Just for simplicity assume the API has following key fields, Customer Name, NIC, Address. We can write a simple logic like, for given a time window, if the API is called with the same Customer name for **n** no of times then send the event to an alert stream. The subsequent Siddhi query will look like below.

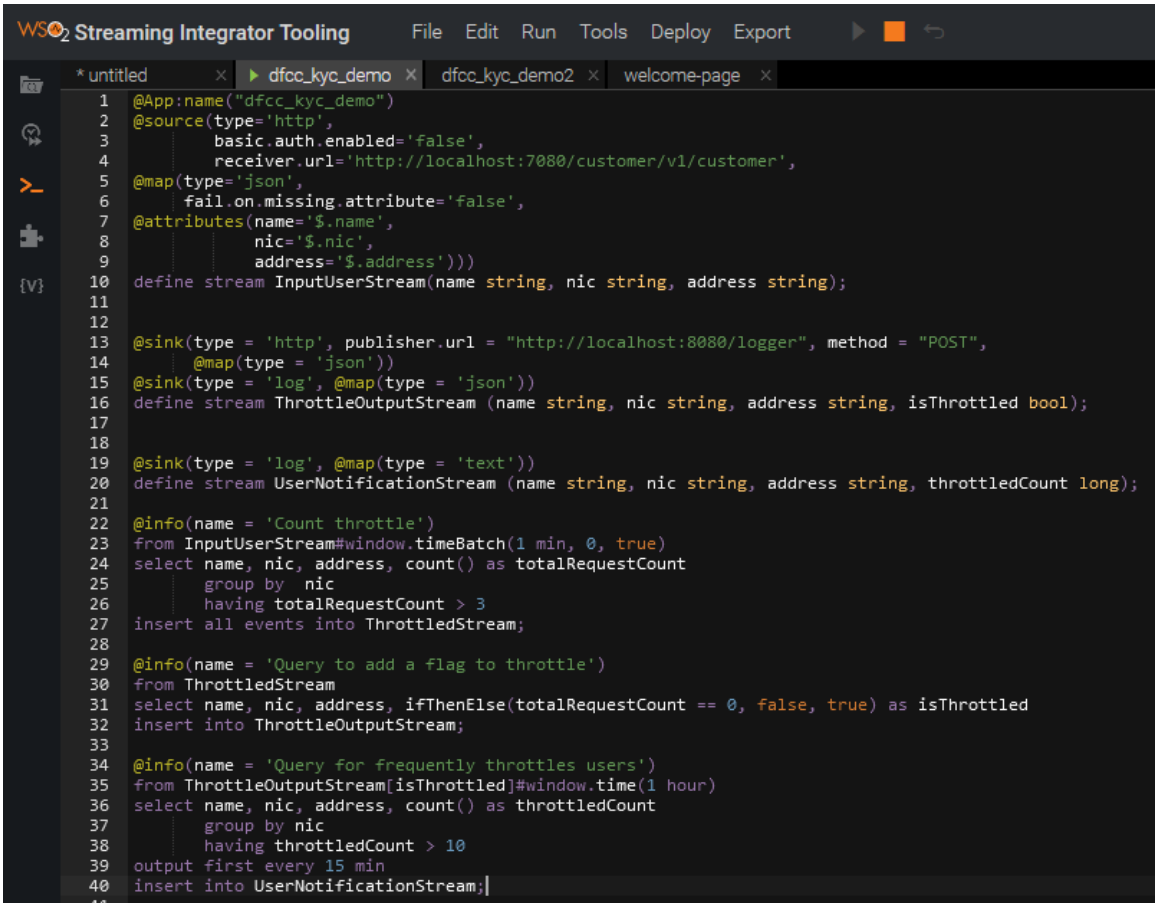

Figure 5.2 : RegTech implementation - KYC source code

If we see this Siddhi application, it receives JSON payload from http://localhost:7800/customer/v1/customer, reads the data into an input stream, do the necessary processing logic and passes it to the output stream. As a prototype version the output stream is logged but as per architecture decision this can be either sent to an alerting system or even another streaming solution to be further analyzed.

Another scenario we applied and tested for KYC RegTech is if the customer tries to call the API from different addresses with the same name within a time window, it can be considered as a suspicious activity. Following would be the subsequent Siddhi query.

```
@info(name = 'Count names within time frame')
from InputUserStream#window.timeBatch(1 min, 0, true)
select name, nic, address, count() as totalRequestCount
        group by  name
        having totalRequestCount > 3
insert all events into FilterStream;

@info(name = 'Count address within time frame')
from InputUserStream#window.timeBatch(1 min, 0, true)
select name, nic, address, count() as totalRequestCount
        group by  address
        having totalRequestCount > 3
insert all events into FilterStream;
```

Figure 5.3 : RegTech implementation – KYC source code

Since other Siddhi code will be the same it has been omitted and main logic is been given here. Within a time, frame customers calling the API with same name would be sent to a stream. Similarly, an API being called with the same address would be sent to another stream and these two streams can be correlated and analyzed for any suspicious activity.

5.2.2 RegTech Implementation 2 – Risk Analysis

The sample API we took for risk analysis was transaction API. A transaction can be either a fund transfer or bill payment. In both the cases funds will be transferred from account A to Account B. We implemented a sample logic if the transaction amount is more than 5000K send it to a notification stream. Though this may be a legit transaction all the transaction being transferred above that amount would be notified and, in the backend, for a later analysis to be conducted. The subsequent Siddhi query will be as follows.

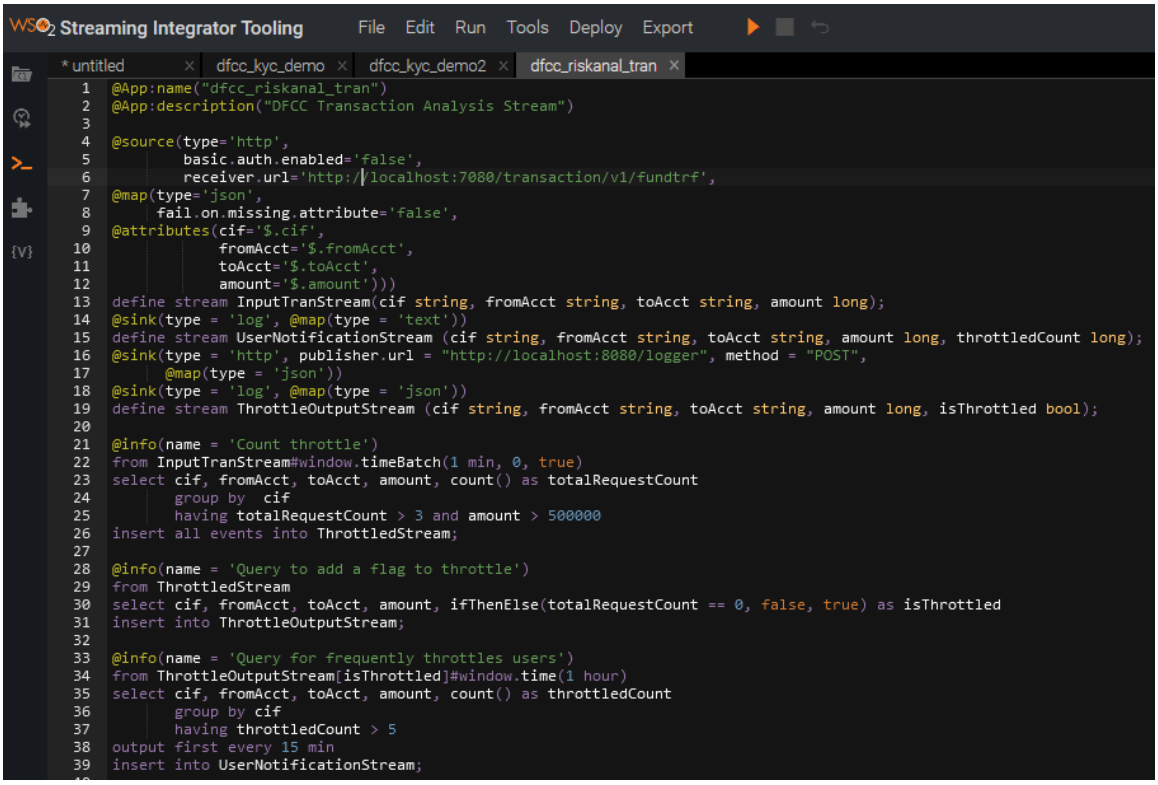

Figure 5.4 : RegTech implementation - Risk analysis

The above Siddhi query accumulates transactions with a time window and notifies an alert stream if the same user tries to process transaction amount more than 500000 and greater than three times. Based on regulatory aspects these parameters can be changes but for a prototype version these values were used.

5.2.3 RegTech Implementation 3 – Audit Reporting

As discussed in the methodology, our object will be to store a CRIB API response in the database for future use. This feature can be achieved by any other tool, but purpose of this use-case is to indicate a stream processing architecture solution can archive the same objective. The CRIB API is a SOAP based and the response will be XML. Using ACE we make the call to the CRIB and get the response. This response will be sent to the stream engine before routing back to the initiator. Following diagram illustrates the flow.

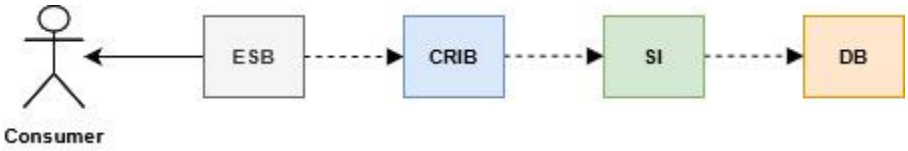

Figure 5.5 : CRIB API call

Consumer will call the ESB exposed service. With the inputs ESB will make the call to the CRIB. The response will be routed to the Stream Integrator where this details will be stored in database. Below is the subsequent Siddhi query to store the response in the database.

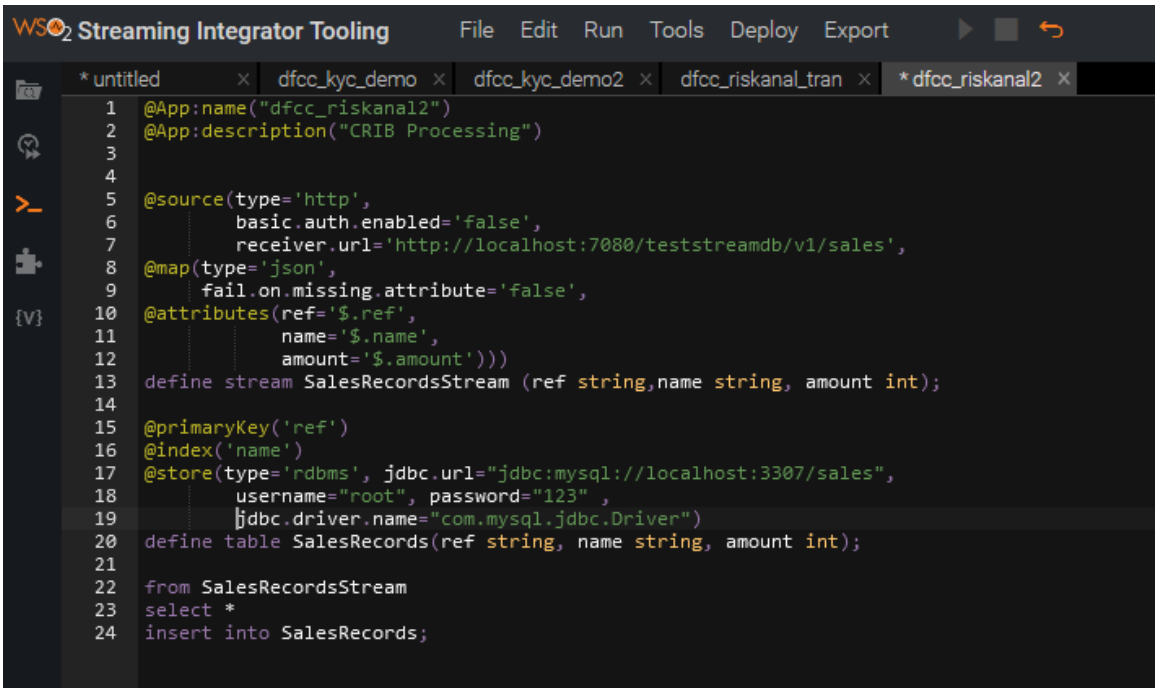

Figure 5.6 : RegTech implementation - audit reporting

5.3 Sample API Performance Results

With the given proposed architecture there was a requirement to provide API response time test results based on the current solution and proposed solution. This was to assure the introduction of a stream processing engine in between API manager and the Core Banking System will have minimal footprint in the current solution. For this purpose, tests were carried out based on the following categories.

- No of APIs
- With and without security implementation

Following test results were observed.

Test Cycle 01 – No of APIs without SSL

Table 5.2 : No of APIs without SSL

No of API Calls	Without Streaming Solution	With Streaming Solution
10	20	22.3
100	200.3	203.6
1000	2006.9	2010.3

Test Cycle 02 – No of APIs with SSL

Table 5.3 : No of APIs with SSL

No of API Calls	Without Streaming Solution	With Streaming Solution
10	82	89
100	800.96	809.36
1000	8009.6	8021.36

These results are averages from multiple runs. As indicated, there is very little impact on API processing time against the current solution. But these tests may vary if conducted in and end to end integrated system with actual API transaction values.

6 CONCLUSION AND FUTURE WORK

6.1 Conclusion

This project started off with analyzing the current Regulatory Technology landscape in Sri Lankan banks. It was identified that large, expensive monolith systems are used in the bank back end to adhere regulatory compliance. These systems are reactive rather proactive. Since, in Sri Lanka the sole regulatory body for banks is Central Bank of Sri Lank, regulatory technology has been in a backdrop and the RegTech innovation has been in a slow pace. As the Open Banking initiative is catching up in Sri Lanka and few banks have already invested heavily in the process of acquiring open banking suits, we believe this is the ideal time to rethink about RegTech, the use of cutting-edge technology that will be used for regulatory and compliance purposes. After analyzing few selected banks where adequate resources were accessible, DFCC bank was identified to propose a stream based RegTech solution. Few streaming technologies were analyzed and after understanding the economical context, WSO2 Stream Integrator, an opensource stream processing engine was selected to implement a prototype for the proposed solution architecture. Three simple use-case were identified namely, Know Your Customer, Risk Analysis and Audit Reporting. These use-case were then implemented as prototype and results were obtained and analyzed. Also, an Architectural analysis was conducted using the traditional architecture evaluation methods ATAM and CBAM. Few tradeoffs between architecture quality attributes were identified and discussed. The analysis between stakeholders resulted the prototype solution satisfies banks expectation and turned-out to be a cost effective one. Few quality attribute tradeoffs had to be made, but they we within acceptable and agreeable levels within the relevant bank stakeholders.

6.2 Future Work

Though this project covers from proposing RegTech solution architecture, evaluating the proposed architecture and implementing it as a prototype, this can be only considered as a beginning. There are so many avenues that can be used to build a complete RegTech

solution architecture using budding open source technologies such as AI and ML. This AI and ML technologies can work very well with streaming technology and can help in identifying transaction patterns, fraud prevention and detection, credibility of customers and many more. The proposed architecture can be extended to include those technologies and implement a full-fledged RegTech solution. Also, this project only focused on RegTech for Open Banking. This may be extended to other regulatory mechanisms and channels such as Anti Money Laundering or even extended to other banking channels such as ATM, Mobile Baking etc. Following can be he a high-level architecture diagram for a future complete RegTech solution.

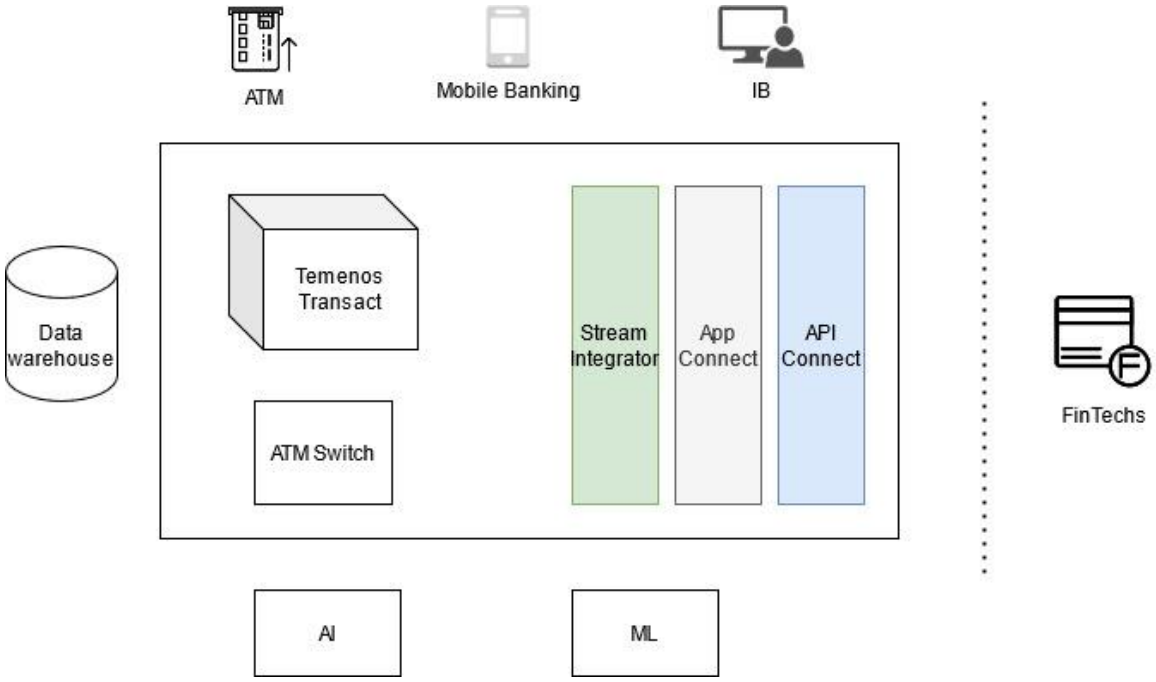

Figure 6.1: DFCC Future architecture proposal

As Figure 6.1 indicates an additional AI and ML components can be integrated where the ingested streaming data will process and analyze for patterns.

7 REFERENCES

[1] Anagnostopoulos, I. (2018). Fintech and regtech: Impact on regulators and banks. *Journal of Economics and Business*, *100*, 7-25.

[2] Arner, D. W., Barberis, J., & Buckey, R. P. (2016). FinTech, RegTech, and the reconceptualization of financial regulation. Nw. J. Int'l L. & Bus., 37, 371.

[3] Arner, D. W., Barberis, J. N., & Buckley, R. P. (2016). The emergence of RegTech 2.0: From know your customer to know your data.

[4] Zetzsche, D. A., Arner, D. W., Buckley, R. P., & Weber, R. H. (2019). The future of data-driven finance and RegTech: Lessons from EU big bang II.

[5] Gurung, N., & Perlman, L. (2018). Use of Regtech by Central Banks and Its Impact on Financial Inclusion. Available at SSRN 3285985.

[6] Goul, M. (2019, July). Services computing and RegTech. In 2019 IEEE World Congress on Services (SERVICES) (Vol. 2642, pp. 219-223). IEEE.

[7] Kavassalis, P., Stieber, H., Breymann, W., Saxton, K., & Gross, F. J. (2018). An innovative RegTech approach to financial risk monitoring and supervisory reporting. The Journal of Risk Finance.

[8] Piri, M. M. (2018). The changing landscapes of fintech and regtech: Why the United States should create a federal regulatory sandbox. Bus. & Fin. L. Rev., 2, 233.

[9] Babar, M. A., Zhu, L., & Jeffery, R. (2004, April). A framework for classifying and comparing software architecture evaluation methods. In 2004 Australian Software Engineering Conference. Proceedings. (pp. 309-318). IEEE.

[10] Barcelos, R. F., & Travassos, G. H. (2006). Evaluation Approaches for Software Architectural Documents: a Systematic Review. In CIbSE (pp. 433-446).

[11] Meedeniya, I. (2010, May). An incremental methodology for quantitative software architecture evaluation with probabilistic models. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2 (pp. 339-340).

[12] Koziolek, H., Domis, D., Goldschmidt, T., Vorst, P., & Weiss, R. J. (2012, August). MORPHOSIS: A lightweight method facilitating sustainable software architectures. In

2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (pp. 253-257). IEEE.

[13] Arcelli, D., Cortellessa, V., & Di Pompeo, D. (2018, April). Performance-Driven Software Architecture Refactoring. In 2018 IEEE International Conference on Software Architecture Companion (ICSA-C) (pp. 2-3). IEEE.

[14] Dragomir, A., Lichter, H., Dohmen, J., & Chen, H. (2014, December). Run-time monitoring-based evaluation and communication integrity validation of software architectures. In 2014 21st Asia-Pacific Software Engineering Conference (Vol. 1, pp. 191-198). IEEE.

[15] Knopel, J., & Lindvall, M. (2009). Software architecture visualization and evaluation. SAVE web site: http://www. fc-md. umd. edu/save/about. aspx.

[16] Stephens, R. (1997). A survey of stream processing. Acta Informatica, 34(7), 491-541.

[17] Hernández-Nieves, E., Hernández, G., Gil-González, A. B., Rodríguez-González, S., & Corchado, J. M. (2020). Fog computing architecture for personalized recommendation of banking products. Expert Systems with Applications, 140, 112900.

[18] Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y., & Zdonik, S. B. (2003, January). Scalable Distributed Stream Processing. In CIDR (Vol. 3, pp. 257-268).

[19] Loganathan, G., Samarabandu, J., & Wang, X. (2018, December). Real-time intrusion detection in network traffic using adaptive and auto-scaling stream processor. In 2018 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

[20] Suhothayan, S., Gajasinghe, K., Loku Narangoda, I., Chaturanga, S., Perera, S., & Nanayakkara, V. (2011, November). Siddhi: A second look at complex event processing architectures. In Proceedings of the 2011 ACM workshop on Gateway computing environments (pp. 43-50).

[21] Nord, R. L., Barbacci, M. R., Clements, P., Kazman, R., & Klein, M. (2003). Integrating the Architecture Tradeoff Analysis Method (ATAM) with the cost benefit analysis method (CBAM). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

[22] Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., & Carriere, J. (1998, August). The architecture tradeoff analysis method. In Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No. 98EX193) (pp. 68-78). IEEE.

[23] Kazman, R., Gagliardi, M., & Wood, W. (2012). Scaling up software architecture analysis. Journal of Systems and Software, 85(7), 1511-1519.

[24] Goodhart, C. (2011). The Basel Committee on Banking Supervision: A history of the early years 1974–1997. Cambridge University Press.

[25] Arner, D. W., Barberis, J., & Buckley, R. P. (2017). FinTech and RegTech in a Nutshell, and the Future in a Sandbox. CFA Institute Research Foundation.

[26] Streaming integrator. (n.d.). Retrieved February 06, 2021, from https://wso2.com/integration/streaming-integrator/

[27] IBM knowledge center. (n.d.). Retrieved February 06, 2021, from https://www.ibm.com/support/knowledgecenter/SSCRJU_4.2.1/com.ibm.streams.welcome.doc/doc/ibminfospherestreams-introduction.html

[28] Stateful computations over data streams. (n.d.). Retrieved February 06, 2021, from https://flink.apache.org/

[29] Azure stream ANALYTICS. (n.d.). Retrieved February 06, 2021, from https://azure.microsoft.com/en-us/services/stream-analytics/

[30] Temenos Financial Crime Mitigation - AML & Fraud Software Solutions. (2020, December 16). Temenos. https://www.temenos.com/products/financial-crime-mitigation/

[31] CustomerXPs Software. (2021, February 3). Real-time Platform to Combat Fraud + Grow Revenue. Clari5. https://www.clari5.com/

[32] ISO 25010. (n.d.). ISO 25000. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[33] Siddhi. (n.d.). Siddhi - Cloud Native Stream Processor. Retrieved February 14, 2021, from https://siddhi.io/

[34] Introduction - WSO2 Enterprise Integrator Documentation. (n.d.). Enterprise Integrator Documentation. Retrieved February 14, 2021, from https://ei.docs.wso2.com/en/7.2.0/streaming-integrator/overview/overview/

[35] The Credit Information Bureau of Sri Lanka. CRIB Sri Lanka, www.crib.lk/en. Accessed 22 Mar. 2021.

[36] “Central Bank of Sri Lanka.” CBSL, www.cbsl.gov.lk. Accessed 22 Mar. 2021.

[37] Hazelcast. “What Is Stream Processing? A Layman’s Overview.” Hazelcast, 24 July 2020, hazelcast.com/glossary/stream-processing.

[38] DFCC Bank PLC, www.dfcc.lk. Accessed 22 Mar. 2021.

[39] Hatton National Bank. “Personal, Corporate & SME Banking Services in Sri Lanka by HNB.” Hatton National Bank, www.hnb.net. Accessed 22 Mar. 2020.

[40] “Bank of Ceylon.” Bank of Ceylon, boc.lk. Accessed 22 Aug. 2020.

[41] Puschmann, Thomas. "Fintech." Business & Information Systems Engineering 59.1 (2017): 69-76.

[42] Goldstein, Itay, Wei Jiang, and G. Andrew Karolyi. "To FinTech and beyond." The Review of Financial Studies 32.5 (2019): 1647-1661.

[43] Gai, Keke, Meikang Qiu, and Xiaotong Sun. "A survey on FinTech." Journal of Network and Computer Applications 103 (2018): 262-273.

[44] Brodsky, Laura, and Liz Oakes. "Data sharing and open banking." McKinsey & Company (2017).

[45] Zachariadis, Markos, and Pinar Ozcan. "The API economy and digital transformation in financial services: The case of open banking." (2017).