Strangler Abstraction Transformation

Boddam Linga Reddy

Abstract:

The challenge to substitute the Device Financing Eco-System is demanding. As systems age, the development tools, hosting technology, and even system architecture that the product was built on, have become increasingly inextensible. Significant time and money were vested without success to upgrade the current COTS lending engine. As necessity is the mother of invention, we have devised a new modus operandi which successfully addressed the long remonstrance list. The "Strangler Abstraction Transformation (SAT)" methodology employed Strangler, Saga and CQRS (Command Query Responsibility Segregation) patterns to enhance the routing of transactional data into two different financial systems and updating corresponding realm of ecosystems, making the paradigm scalable and seamless.

Keywords: *Saga, CQRS, Strangler, Cloud*

INTRODUCTION

T-Mobile US [1] provides wireless voice, messaging, and data services in the United States, Puerto Rico, and the U.S. Virgin Islands under the T-Mobile and Metro by T-Mobile brands. The company operates as the third largest wireless network in the U.S. market with over 83 million customers and annual revenues of $34 billion. Its nationwide network reaches 98 percent of Americansthrough its EDGE 2G/HSPA 3G/HSPA+ 4G/4G LTE networks, as well as, through roaming agreements.

T-Mobile's very own idea, implementation and launch of Device Financing Product in 2009 was a revolution. Known in the telecom landscape as "Equipment Installment Plan" (EIP) [2], it was the predecessor of many other finance products and programs such as Lease, JUMP (Just Upgrade My Phone) and 'Un-Carrier'[3].The impacts of these products were so huge, that the competition was forced to adapt these products and programs to their establishments.

The Equipment Installment Plan (EIP) is a home-grown system which currently serves around 35 Million and counting 'Active' Loans and Leases. The system feeds largeamount of data to over 150+systems in the T-Mobile landscape, which serves Customers, Accounting, Billing, Auditing, Ordering and the Reporting verticals.

The Challenge

The current EIP system had evolved into a very tightly coupled architectural system with numerous interactions and validations. Many functions, though not required by the system itself, are forced to fit-in. As business and market needs evolved, the system ended up having orchestrations and dependencies on credit decisioning, billing, business rule engine, government compliance rules, 3+ million lines of code, 250+ database tables with more than 500 million transactional and historical records, making it an extremely tightly coupled system.

The major drawback of the current system is that it is not 'Payment – aware'. Currently, the system marks the payment as 'Done' as soon as the CIG (charge Injection) process is done and sent to the billing system. However, the reconciliation between the payment, billing and finance system is not in sync; resulting in revenue leaks and finance auditing discrepancies.

The monolithic nature of the system is also a key driver that triggered need of a new finance system. The functional and technical disadvantages include,less scalability, as the system is not designed to support new business products and ideas.This includes, interest rate calculations, and flexible finance options, High test and operation costs which puts the ROI (Return onInvestment) factor at a minimum. Fault tolerance and speed to market of the code/product is also low.

When a New Finance system (COTS product from Oracle; a.k.a OFSLL (Oracle Finance System for Loans and Leases) [4] was determined to be onboarded with the intention to replace the in-house system, the task was daunting. De-coupling systems and their corresponding functionality without impacting real time transactions and front-line applicationswas unachievable.

Lessons Learned:The finer level details during the engagement of a monolithic system should be given proper care, before its introduction into the landscape, given the fast-changing business and technical needs.

DISCUSSIONS

The method of "SAT (Strangler Abstraction Transformation)" primarily forms a nucleus around finance serving application and back office

ecosystems.This nucleus is made up of micro components built using Spring Boot framework with cloud-native Twelve-Factor App principles.This offers a clean contract with the underlying operating system and maximum portability between execution environments. Employed the cloud computing [5] PaaS cloud platform as infrastructure to build and deploy and operationalize the transformation.

The components use a combination of Saga [6], Strangler [7] microservice engagement along with CQRS patterns [8]. Since the idea is to route transactions into two mutually exclusive, yet functionally similar, finance systems (EIP and OFSLL); Saga microservice patterns represent a high-level business process that consists of several low-level requests that update data within a single service. Each Request has a compensating request that is executed when the request fails, or the Saga is aborted.

Strangler microservice patterns ensure that the incoming transactions are validated against the legacy system setup, in addition to creating the same record in the new finance system,If the legacy finance system is incapable of consuming the transaction.This ensure the Backoffice ecosystems does not receive unwanted updates. For instance, New Customers trying to "Activate" their device through T-Mobile are routed to the New Finance System and existing customers trying to "Add a line" to their accounts are routed to the Legacy system. This is achieved without technical contract changes between the calling applications and the legacy system but simultaneously routed the wanted transactions to the new finance system.

The engagement of CQRS pattern on RabbitMQ [9] framework supports multiple denormalized views that are scalable. This provides the benefit of improved separation of concerns, and simpler command and query models. This paradigm allows implementing a query that retrieves data from multiple services in a microservice architecture.

The above rationale can be illustrated in the following use case; When a Loan is created and required to be recorded in the designated finance system, Saga pattern coordinates with multiplesystems (Credit, TOOP, Eligibility, Device validation services etc.,) to Read/Write data. Strangler patterns validate thefunctional variance of the loan, meaning if it's a 'Brand New Activation', 'Add a Line' or an 'Upgrade' scenario, and routes the record to the anticipated finance system.

THE STACK

The high-level architecture including the technology stack used to achieve "Strangler Abstraction Transformation (SAT)" is given below Figure 1 and Figure 2

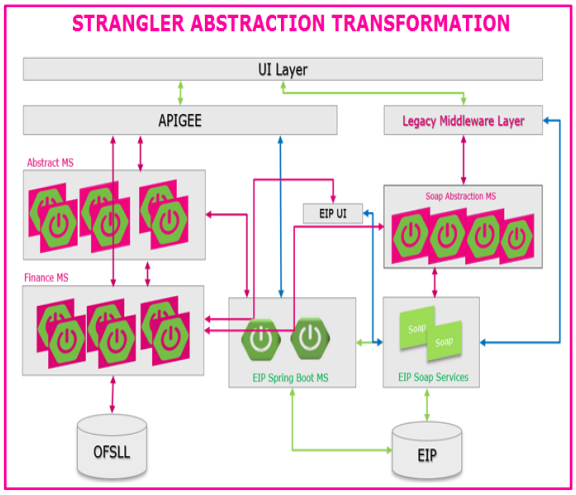

Figure 1

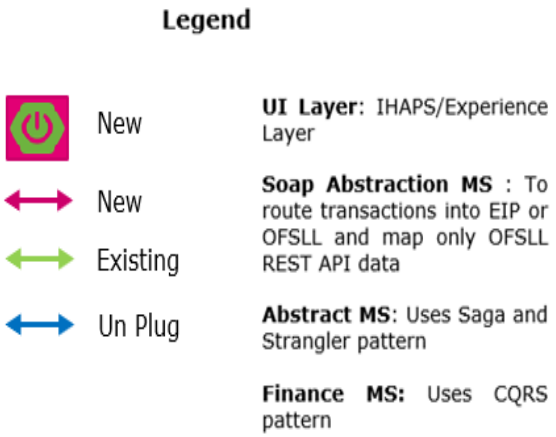

Figure 2

The high-level architecture Figure 3 including the technology stack used to achieve "Abstraction for Back Office Batch Processing" is given below;

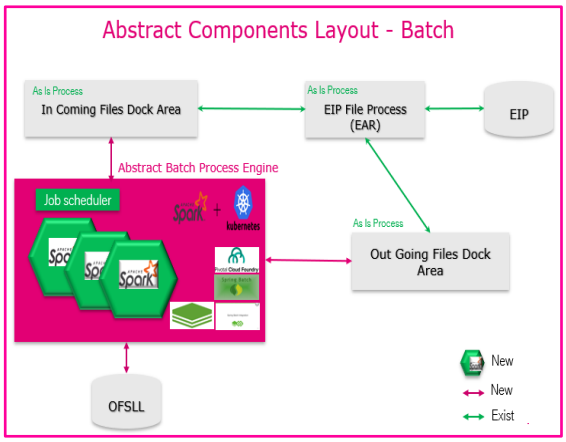

Figure 3

We have employed Spring[10] Batch framework to build abstract layer between existing and new finance systems for small sized data depending on business use cases. This gives the advantage to decouple layers, in addition to, implementing a generic business logic through out the application. The framework also enables separation between infrastructure and application at the executable JAR level, resulting in easy and scalable deployment.

Apache Spark framework [11] used to process large files and data sets generated by both finance systems. We utilized power of Spark to data load and migration of existing data in EIP to new system OFSLL. The advantage we got migration of over 90 million records in 15 hours, so that no customer experience impact and customer payments impact during course of migration records old system EIP to new system OFSLL.

CONCLUSIONS

The idea to not impact existing customer facing applications was attractive. The legacy applications can continue to call the same services and still achieve the business and enterprise goal of moving to a New Finance system. Historically, this type oflarge move typically creates unrest among multiple ecosystems. However, Due to the digital nature of 'Abstraction', the labor process was natural and painless. The testing effort is de-centralized, and the transition was smooth.

From a funding and operational perspective, significant savings were achieved. A critical system 'replacement' usually warrants tedious planning sessions along with budget and resource allocations. In this case, the budget and workforce savings were at a record high in comparison to traditional replacement efforts, this could be directly attributed to the technicaldecoupling and rerouting paradigms followed in 'Abstraction'.

The design pattern of 'SAT'is leveraged by many applications and systems across the enterprise who are introducing new systems in their corresponding landscape. This includes verticals such as Billing, Customers, Credit, Audit and Orders. Many accolades were received from multiple domains. The launch of new business products has become very achievable without high costand resources. Time-to-Market of business products is reduced with improved customer experience and servicing.

The benefits after introduction of SAT can be understood in the examples below. (Statistics from Pilot Production Launch)

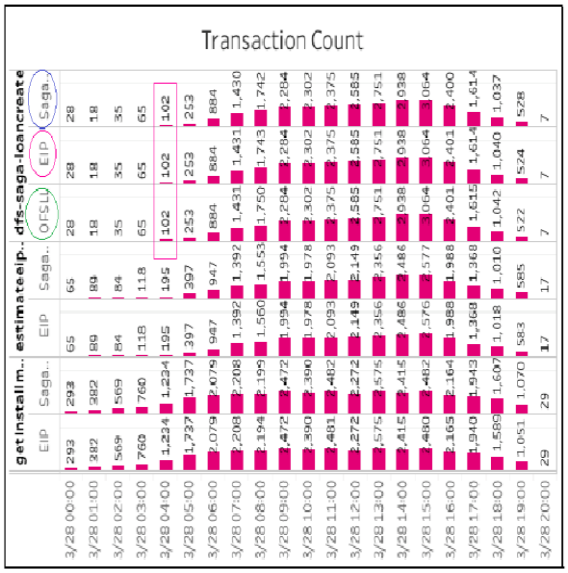

Figure 4

Transaction Count: Figure 4 shows parityin count across EIP and OFSLL (Two Finance Systems). The Transaction Count graph denotes the # of transactions flowing into each finance system in parallel for the same functionality.

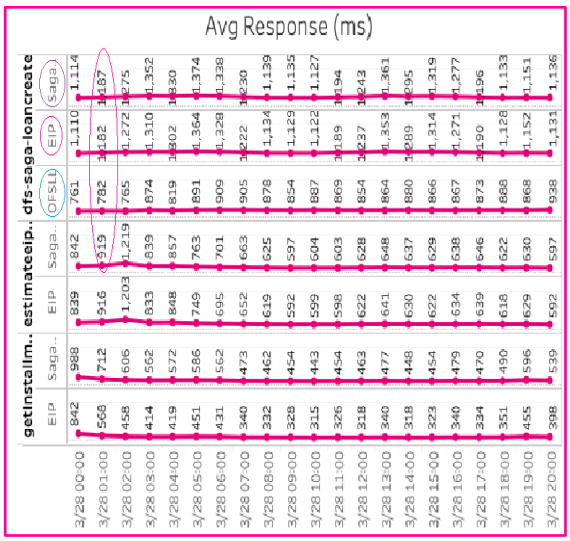

Figure 5

Average Response Times: Figure 5 As shown in the graph above, the introduction of SAT proved that the average response time into OFSLL is actually less that the response time of the legacy system. The benefits of digital transformation are transparent.

ACKNOWLEDGEMENTS

It takes a team to build a village, and the team was nothing but exceptional. Special Thanks to Anu Mahanty and Ram Sadasivam for allowingthe

team to experiment an innovative idea and making it work!

Thanks to invaluable feedback and support from our Enterprise Leadership; Cody Sanford, Robert Gary, Kevin Nelson, Darron Webb, Chuck Knostman and Kris Wilson.

Thanks to our Business Leaders; Peter Oswaldik, Kellie Berndt and Christopher Hooppaw.

ABBREVIATIONS

SAT	Strangler Abstraction Transformation
EIP	Equipment Installment Plan
CIG	Charge Injection
COTS	Commercial Of-the shelf
TOOP	Tailored Out of Pocket
CQRS	Command Query Responsibility Segregation
OFSLL	Oracle Finance System for Loan and Lease
JAR	Java Archive

REFERENCES

[1] T-Mobile USA INC. (2019). About T-Mobile. Retrieved fromhttps://www.t-mobile.com/about-us
[2] T-Mobile USA INC. (2019).Buy T-Mobile Device. Retrieved fromhttps://support.t-mobile.com/docs/DOC-1674
[3] T-Mobile USA INC. (2019). Un-Carrier History. Retrieved fromhttps://www.t-mobile.com/our-story/un-carrier-history
[4] Oracle Inc. (2017). Oracle Financial Services Lending and Leasing. Retrieved from https://docs.oracle.com/cd/E89525_01/index.htm
[5] Gorelik, E. PhD thesis, Massachusetts Institute of Technology. Cloud computing models (2013)
[6] Mell, P., Grance, T. The NIST definition of cloud computing, et al. (2011).
[7] Anderson Chris. (2019). Pattern Saga. Retrieved from https://microservices.io/patterns/data/saga.html
[8] Martin Fowler. (2004). StranglerFigApplication. Retrieved from https://martinfowler.com/bliki/StranglerFigApplication.html
[9] Marin Fowler. (2011). CQRS. Retrieved from https://martinfowler.com/bliki/CQRS.html
[10] Pivotal Software Inc. (2017). RabbitMQ. Retrieved from https://www.rabbitmq.com/
[11] Pivotal Software Inc. (2019). Spring Batch. Retrieved from https://spring.io/projects/spring-batch
[12] The Apache Software Foundation. (2018). Apache Spark. Retrieved fromhttps://spark.apache.org/