Ömür Şentürk | Experienced Domain Architect & Business Analyst & Product Owner | Expert in Strategic Integrations & Digital Transformation

Summary

Experienced Domain Architect, Business Analyst, Product Owner and with around 15 years of experience driving strategic digital transformations and complex integrations in Telecom, E-Commerce and Banking industries. I architected an enterprise-scale data model for a cloud-based iPaaS solution. I successfully led a CRM migration project, including a roll out phase that dealt with 25 million customers with zero downtime. I created entire backlogs from scratch, managed priorities and dependencies for e-commerce re-platforming projects. I am highly skilled in leveraging TMFroum and BIAN frameworks, UML syntax for technical documentation, stakeholder management, domain design and data modeling, API documentation, Python scripting for automation, and agile methodologies.

Experience

Sr. Domain Architect | Odido, Den Haag, Netherlands (2025 September - ...)

Worked on Simpel (subbrand of Odido) IT Delivery Team within Mass Market domain and now working within large initiatives / projects within Odido that requires overarching architecting across multiple domains and applications. Some of my duties and accomplishments are as follows:

- Provided architectural guidance to implementation teams through high-level solution designs, while aligning on low-level designs to support seamless customer and user journeys.
- Delivered strategic guidance and informed decision-making processes for major projects, and KTLO-related requirements, by actively investigating data flow patterns on multiple systems to protect the product vision and delivery priorities.
- Initiated and designed the target-state documentation approach for both Simpel and Odido technology stack, covering high-level solution diagrams, UML sequence diagrams for key processes, and project-specific integration documentation.
- Made Simpel’s and Odido's complex solution landscape easier for delivery teams and stakeholders to understand by creating clear and practical documentation that had previously been missing.
- Investigate and apply TMForum API Specifications for new implementations.

Associate Domain Architect & Enterprise Business Analyst | Backbase, Amsterdam, Netherlands (2024 May - 2025 August)

Working on internally built productized cloud based (Azure) iPaaS Solution, Grand Central with duties of an Enterprise Business Analyst / Jr. Domain Architect in the banking industry. Some accomplishments and details of my duties:

- Spearheaded creation of organization-wide data model structures and event specs for productized OOTB integrations, therefore creating better developer experience for new API implementations and increasing commercial value of Grand Central.
- Managing organization-wide BIAN framework, adoption roadmap in alignment with Product Managers, Domain Architects, Lead Developers and Business Analysts, therefore assisting implementation teams with feature scoping and API design.
- Aligned the Unified Rest API Specs with BIAN Framework for Deposit, Loan, Cards, Payments, Party, Party Lifecycle, Fraud and Open Banking / PSD2 domains, therefore paving the path to become BIAN compliant.

- Worked with clients / fintechs to create OOTB connectors on iPaaS that aligns with product vision of Backbase and BIAN framework, identified gaps, supported solution engagements with the development teams, ultimately creating seamless connectivity solutions with each fintech / client.
- As a member of an architecture group as the only Business Analyst, prepared and proposed solutions to solve organization-wide integration problems between Engagement Banking Platform and iPaaS Platform.
- Created supporting UML Sequence diagrams and integration architecture and high level (logical) designs for new integrations that aligns with TOGAF framework. Gathered requirements, created user journey documentations, analyzed and consolidated with multiple stakeholders as the business analyst of Open Banking / PSD2 Berlin Group team, therefore unlocking the capabilities of Consent Management, AIS / PIS journeys.
- Working on designing and implementing workflow based AI agents to analyze market by crawling, summarizing and reasoning with LLM's to elevate market readiness of Grand Central.
- Created a PoC of n8n based workflows that provide competitor analysis with local LLM's. The stack consisted of Python, n8n, Supabase (Postgresql DB) and Github Actions for deployment pipeline.

Business Analyst & Product Owner | rb2, Purmerend, Netherlands (2022 August - 2024 April)

Served dual roles as client facing Business Analyst and Product Owner internally in the e-commerce industry. Some accomplishments and details of my duties:

- Spearheaded two comprehensive web shop re-platforming initiatives. Defined MVP scopes, helped clients identify product strategy, held story mapping sessions, created entire backlogs full of with user stories that are enriched with technical details (that included user journey workflows, use case details), helped / guided external Product Owner with the prioritization of backlog items after aligning with external and internal stakeholders, as well as managing dependencies as the primary contact from a project management perspective.
- Proactively initiated data and log analysis activities with Python scripting. Focused on potential lost sale cases with the help of Google Analytics events and found a number of inconsistencies. Got some help from AI tools with the scripting in the meantime.
- Designed end to end customer journey diagrams for customer go-live documentation as part of the project handover, ensuring seamless transition.
- Followed MACH architecture (microservices, api-first, cloud-native and headless) principles at all times.

Tech. Business Analyst | G-Star, Amsterdam, Netherlands (2022 March - 2022 July)

Partnered with internal Product Owners to gather requirements and develop high-level solution documents tailored for the e-commerce industry.

Domain Architect & Product Owner (Dual Roles) | Turkcell, Istanbul, Turkey (2012 August – 2022 February)

Worked in almost every role in modern SDLC processes: Business Analyst, Product Owner, Software Developer, and Domain Architect in telecommunications industry. Some accomplishments and details of my duties:

- Architected and managed the implementation of a comprehensive CRM database within the CRM renovation project. Became the data model owner of CRM afterwards.
- Designed and carried out a successful migration of 25 million customers. Owned and managed the entire roll out plan that caused zero (0 minutes to be precise) downtime. The migration project team later got rewarded by the CEO for the quality of the work and smooth transition from legacy CRM to new internally built CRM system.

- Worked as the Domain Architect / a Sr. Business Analyst on an internally built CRM application that was used by 500+ employees and integrates with pretty much every application in a Telco company -ERP, Billing, Charging, Order Management, Product Catalogue, Campaign Management to name a few. Made sure every integration and project follows the product vision of our CRM system by signing off on technical designs.
- Worked as the Product Owner of a team of 14 that builds solutions on a number core capabilities of CRM domain: Order Fulfillment, Billing integrations, New Acquisitions, Suspend & Resume, Blacklisting, Cancellation, Charging integrations, Product Move, Product Take Over and Scheduled / Bulk Operations, therefore playing a pivotal role in the capability management of the entire unit which made resource planning and roadmapping of the utmost importance.
- Worked as a Product Owner inside a team of Product Owners to manage and prioritize all of the backlogs of teams inside the unit (+80 FTEs, 9 teams).
- Worked also as a Software Developer on Oracle BPEL business process management and order management platform. Analyzed and implemented (developed), deployed and did the production support of BPMN workflows with synchronous, asynchronous and scheduled integration models.
- First business analyst to write Java and BPEL (XSLT based low code order fulfillment engine) code and deploy it to production inside the unit. Inspired a bunch of others to follow in my steps as well.
- Automated a few simple daily tasks with Python (a few emailing, some error monitoring etc.).

Jr. Test Engineer | Ericsson, Istanbul, Turkey (2009 January – 2011 December)

Developed and executed manual test scenarios for product releases, maintaining close collaboration with developers, business analysts, sales, marketing and other key stakeholders.

Education

Galatasaray University – Engineering Management, Master of Science, 2017, Istanbul Turkey

University of Southern Denmark – Manufacturing and Engineering, Erasmus, 2010, Odense, Denmark

Yildiz Technical University – Industrial Engineering, Bachelor of Engineering 2010, Istanbul Turkey

Skills & Experience

Domain Architecture | Business Analysis | Digital Transformation | Integration Solutions | Product Ownership | Agile Methodologies | UML Sequence Diagrams | Architecture Documentation | BIAN Framework | Data Flow Diagrams | System Integration | CRM & ERP Systems | Core Banking & Payments | Open Banking, PSD2, Berlin Group | Python Scripting | SQL (PL-SQL, T-SQL) & Database Management | Java Development | BPMN, XSLT & Process Automation | REST, SOAP, GraphQL, gRPC | Event Based Systems | Data Analysis & Visualization | Stakeholder Engagement | Roadmap & Backlog Management | Project Management

Languages

English: Fluent

Turkish: Native

Dutch: Beginner (A2)

Appendices

Turkcell Superonline CRM Migration & Rewrite Project

Reasoning of the project

Turkcell Superonline had a tailor made CRM from a vendor, which had an expensive license and maintenance fee. The management decided to go for an inhouse-built solution which would enable faster time to market and lower the costs.

The Scope

Rewrite the entire capabilities of the existing CRM application with inhouse resources on a codebase and migrate all the data from legacy CRM system into the new one. No new features unless they are absolutely mandatory or very easy to implement that would not derail the project timelines.

Timelines

8 months to go live and start from scratch.

Roll out phases & Data Migration Steps

As the details were ironed out by the migration team that I was leading, we came up with the following steps.

1. Start with very limited scope (with only 2 cities involved): Migrate the customers that receive a service from those cities into the new CRM, and block access to those customers in the legacy system. While Call Center would be able to use both the CRM systems (legacy and new), the physical branches in these 2 cities would only be able to use the new system.

2. Expansion phase (Add 2 more cities): After the initial success of the 1st phase of the rollout, we expanded the extent of new CRM system into 4 cities. Call Center is expected to use both CRMs, but now more physical branches would use the new CRM system.

3. Full rollout: After careful consideration and eliminating first batch of bugs from the new system, we rolled out to the entire country. Call Center and all the physical branches were expected to use the new CRM system.

Data Migration & Order Migration

This is where things get interesting. For a successful data migration, we thought solutions for a variety of problems. Data model differences between systems, and the migration process speed per each customer, also the data quality issues (the source of the data is an old system that you can find multiple variations of the same product's configuration or missing pieces of data all over the place because specific features were introduced later but backward compatibility was never implemented so the data is scattered and even broken on some instances.)

We have categorized these focus points into 3 groups

1. Physical data migration
2. Live data migration
3. Rollout phase migration (a.k.a. Silent Migration) & Edge cases & Broken data issues fixing

Physical Data Migration Model

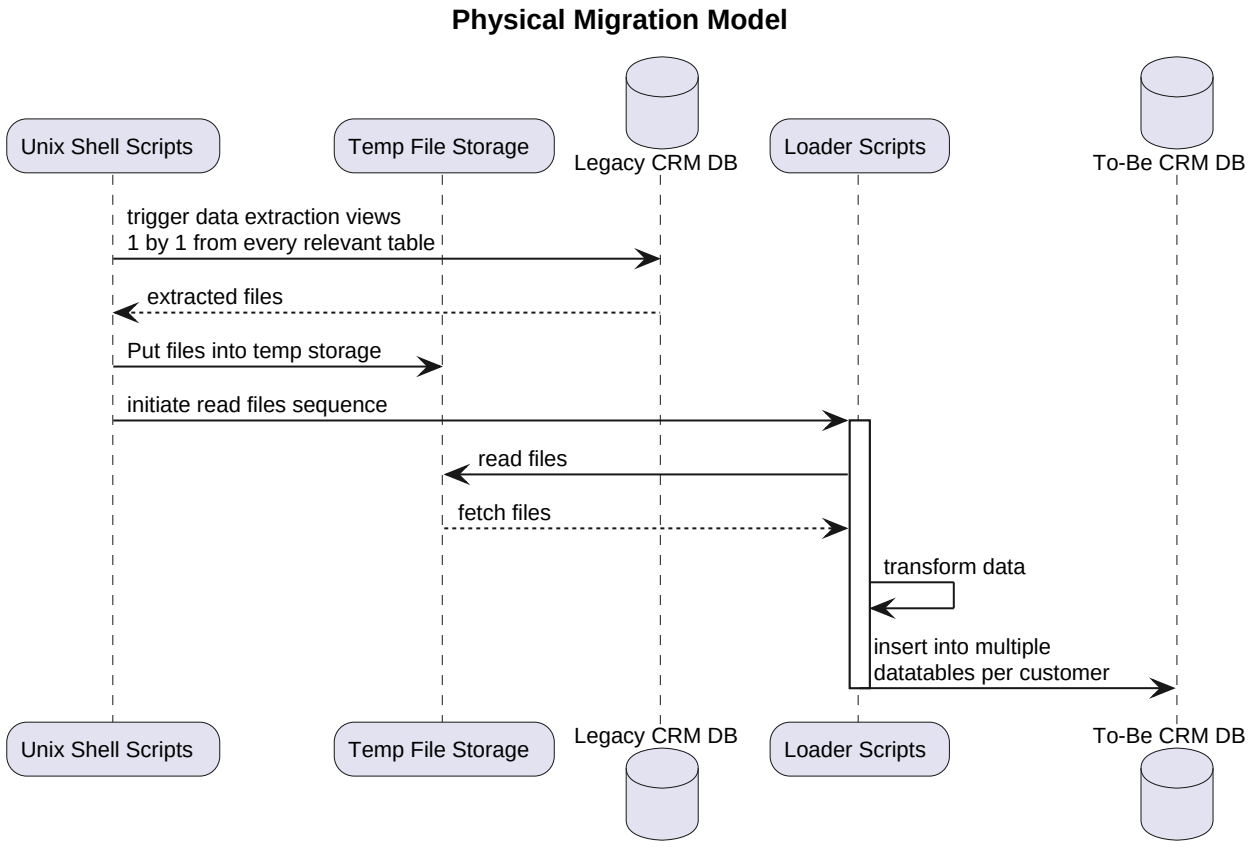
Problem Statement: The data model between legacy and to-be systems are different. There is not a 1-1 match of the database tables. **Solution**: Create a specific migration process for each table that would work per a customer id.

Fundamental Problem: The initial estimations of the physical data migration completion reached to staggering numbers. A single customer migration would last an average of 5 seconds and that's just the physical data migration itself (not taking into account the data extraction and loading of the data onto temporary storage). With 25 million customers, this would mean around

> *~125.000.000 seconds*
> *~2083 minutes*
> *~35 hours*
> *~1.4 days*

We didn't have 1.4 days JUST for data migration. No business in their sane mind would allow IT teams to do migration, which everyone was sure that it wouldn't go smooth because of the data quality issues. All our tests proved that it would take massive amounts of efforts to migrate all these customers.

Enter silent migration (see below live data migration section).



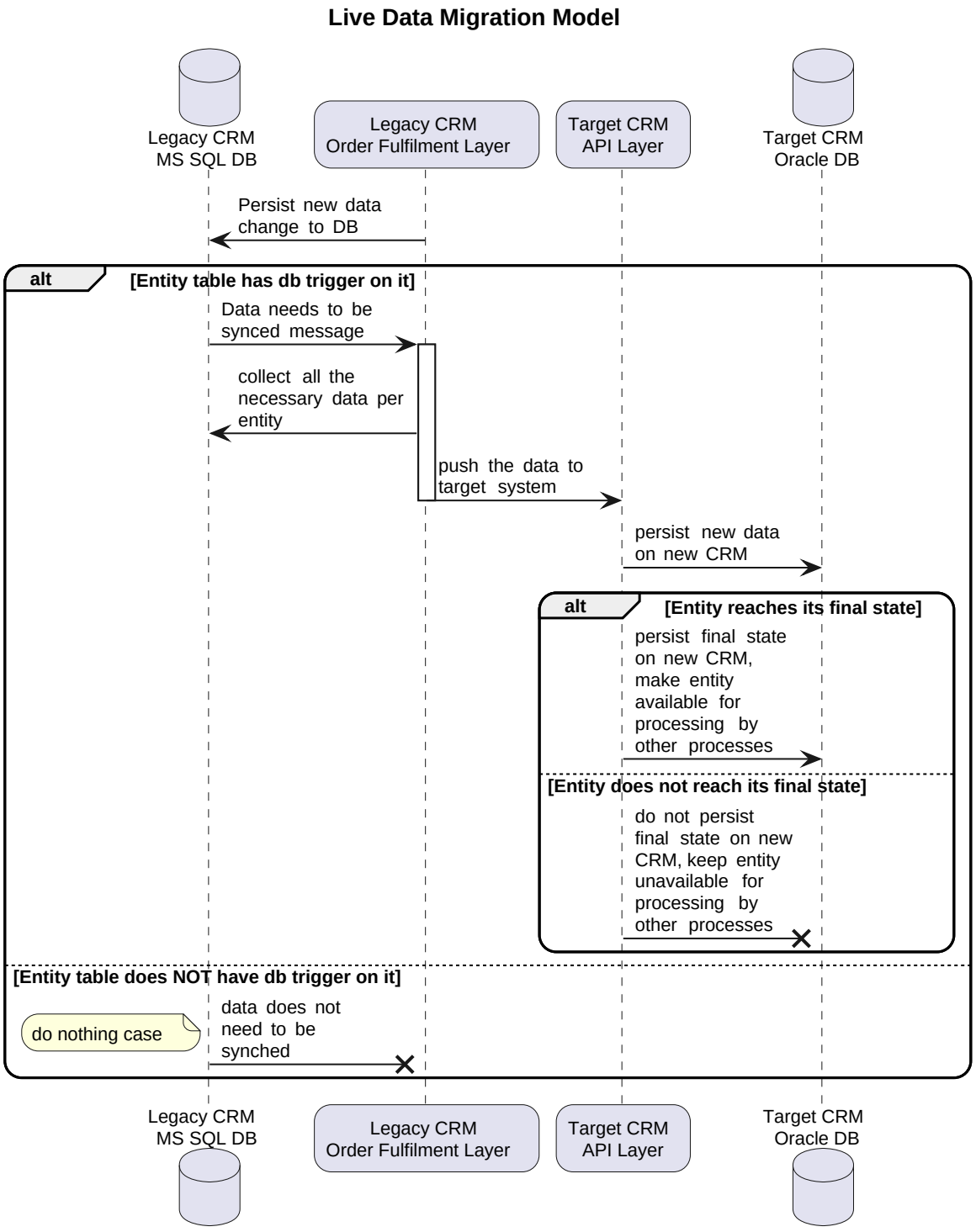
Live Data Migration Model

Problem Statement: The data still is alive in the legacy system due to long-running order fulfilment flows (sometimes days, weeks and even months long, some of them are also in a fault state which guarantees that they will not be completed until someone deals with the problem and manually fixes whatever wrong with the fulfilment flow).

Solution: As the new system will need to be aware of what is happening to the data in the old system, we introduced database level triggers (DB Trigger on MS SQL Server) on the legacy system, on various entities. So, whenever "something" changed on the legacy CRM, it would trigger an event that would eventually push the data to the new CRM system.

Fundamental problem: This whole long-running migration situation creates a fundamental problem: How do you deal with data that still has the probability to be updated by the legacy system because of a long-running order fulfilment process that was never able to complete? We have introduced a specific migration flag per each entity that signals that this entity is still in the process of being migrated. So much

so that the other processes and applications and services instantly know that they should be very careful with this record, because it's highly probable that this record is expected to be updated by the legacy system (as a temporary solution of course).



Between legacy CRM order fulfilment layer and target CRM api layer, there is a data contract per each entity. This data contract brings clear definitions and rules as to what to synchronize into the new system and when. Those rules are described in API specification so that any new developer that is building a new sync process would instantly know the nuances.

Silent Migration Model

At this moment, we have solved 2 major issues

1. Data model changes have been reflected onto the physical data model layer
2. Keeping both legacy and to-be systems alive and synchronized

However, there's still a major problem: migration scripts running time, is still estimated at a staggering 35 hours. There is NO WAY that any business owner will let any IT department to run a script that's going to block the systems for more than entire day. Besides, the chances of running any issue is unavoidable, which will mean re-running of the same script again, and again, and again, probably taking weeks to complete.

This still doesn't even take the data quality issues into account. We are definitely going to have to fix a lot of customer data issue before the 1st step (physical migration) of that customer could be completed.

Problem Statement: So, we need a solution that should

1. Give us more time to run migration scripts to migrate all the customers
2. Give us more time to fix the data quality issues and re-run the script for that/those customer/s

Solution: We thought of this plan below:

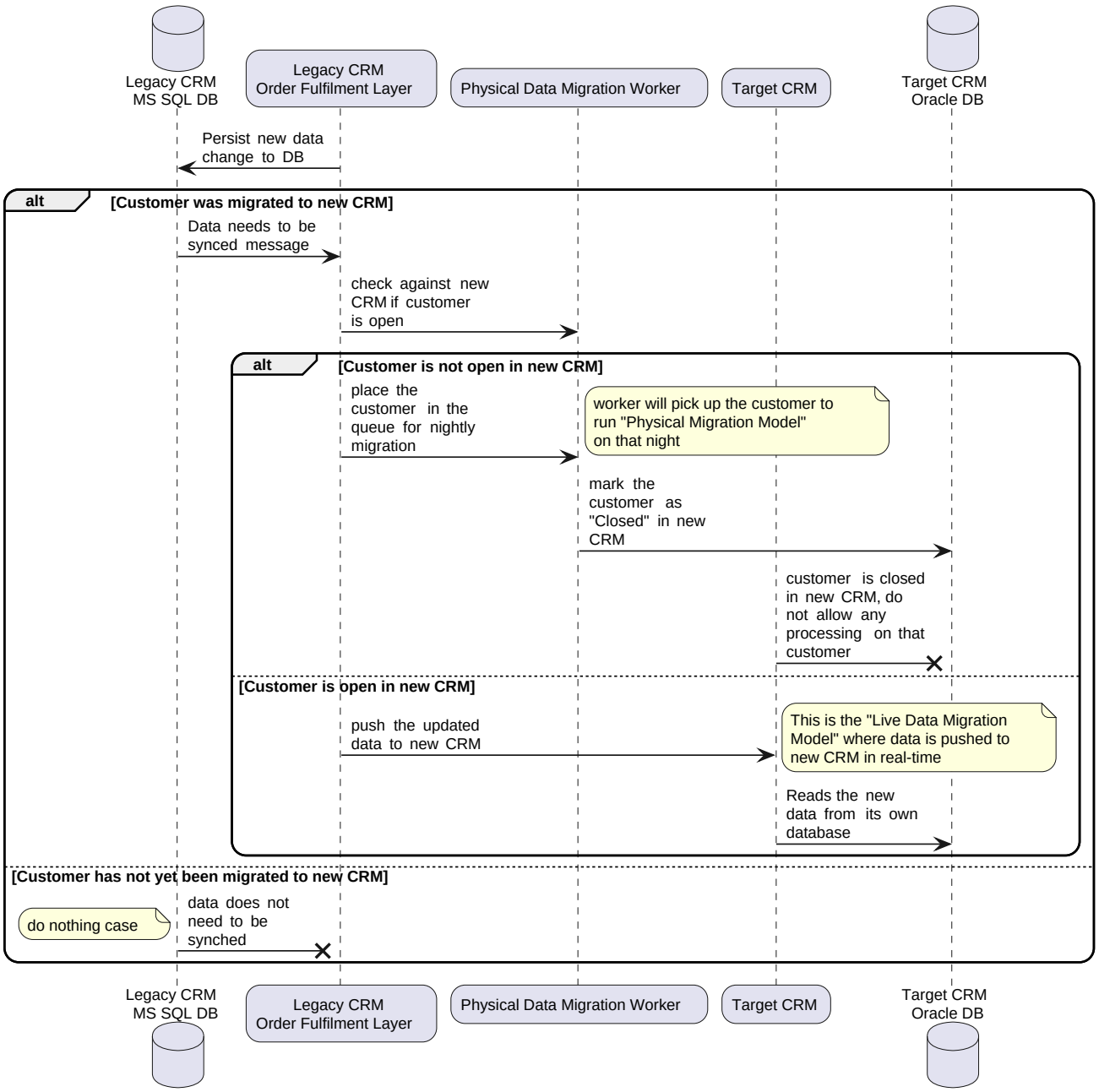
1. Let's start as early as possible and try to migrate the customers BEFORE the new system goes live. With a very simple flag, the physical data that is migrated on the new system can be closed to external processes and API's.
2. When the new system go-live date comes, we can simply flip open a switch on those customers and they will be visible on the new system.
3. In the meantime, all the data quality errors that prevent us to do physical migration can be solved, because right now, we will have more time... Much more time than ever... We can run this migration process for weeks without anyone noticing, and we can golive in an instant with all the customers on the new system.

But even now, we were going to be short on time. Our estimations still showed a narrow window that we can deliver the entire data migration. After further examination of the data, we categorized the customers into multiple groups. On a very high level there were:

1. Customers that do not have any movement for years (business customers are mostly in this category)
2. Customers that have a very specific set of products that are big in numbers
3. Customers that are highly capable of making a new movement and/or have an active ongoing order

"Customers that do not have any movement for years" were the most likely candidates to start as soon as possible, so as the lowest hanging fruit group of customers, we prioritized them. With these customers, we also tested the validity of migration scripts. "Customers that have a very specific set of products" were the second most likely candidates to reduce the number of running migration scripts (therefore, decrease delivery time per each customer). These customer were relative low on priority, so we left them out of the scope. "Customers that are highly capable of making a new movement and/or have an active ongoing order" were going to be our biggest challenge as these customers had the biggest number of data quality issues. So, we took them as the secondary phase of our migration.

Silent Data Migration Model



Backbase Grand Central iPaas Data Modeling

What Backbase does

Backbase, is a banking solutions provider, that focuses on streamlining the user journeys for everything related to a bank: customer onboarding, case management, accounts, transactions, payments, fraud checks, KYC, AML for retail, business, wealth banking and credit unions as well as many more capabilities are supported through its Model Bank solution. Model Bank is a full suit of applications: mobile app to be used by customers, back office applications to be used by bank employees, an integration layer for all these applications as well as their internal databases, with both managed services and on-premise options. Any bank is able to start using Model Bank given that the rest of the application landscape is able to communicate with Backbase's Model Bank solution.

What Grand Central aims to solve

Model Bank is a very opinionated solution that might not be suitable especially for every bank out there from various perspectives:

1. Data ingestion & integration pattern problems: Some applications in banking industry, (e.g. Core Banking applications) are notoriously bad at exposing their data to external services, not to mention the supported ways of integration (SOAP, RESTful, or even message broker). A solution that offers multiple ways of ingesting data to and fro Model Bank is often needed per each bank.

2. Data Modeling Challenges: Every application has their own terminologies, and expect data in a specific format which requires custom implementations every time. With a unified front end solution, this becomes another bottleneck because the data needs to be transformed specifically for each integration.

3. Repetitive work: So many times, customer success teams build the same integration that they have previously done with a different customer over and over for newer customers. That creates repetitive work and takes a lot of time at the project beginning phase.

How ?

Enter BIAN. A unified and ubiquitous language model for common data model implementation, as well as API specification. BIAN, while treated as a guide, provides clear structures for data models for every entity within a banking space. Every banking term like loan account, deposit account or even non-banking specific basic common models such as location are declared publicly declared with ISO20022 BM references. BIAN even offers sample API specifications with most expected actions through their API portal. While those endpoints do not fully support RESTful practices (e.g. endpoint names are verbs, which doesn't align with RESTful practice. See an example here).

My contribution

Through all the OOTB connectors of Grand Central and all the API's and event specs, I have streamlined the entire data model structures for every model that Grand Central supported at that time. Aligned with the Integration Authority Board of Backbase to review the entire data models, and collected feedback. Compared the new datamodels to the existing Grand Central API specs and prepared the impact, aligned with the teams, got into their roadmap and guided them into implementing the differences from the Common Data Model store.

Goal of this initiative

Common Data Model store is the golden source of truth for

- REST API specification models (both as-is ones and the ones that will be implemented in the future)
- Event specs (again, both as-is ones and the ones that will be implemented in the future)

It is also a knowledge base and a crucial part of developer experience value proposition of Grand Central, therefore it adds commercial value to Grand Central suite as well.