BIAN-Driven Domain Model Design in Banking Transformation

Results from Customer Engagements

Alfredo Muñoz Ríos

Global Banking and Financial Markets Center of Competency

alfredomunoz@es.ibm.com

Table of Contents

1 Executive Summary

The next few years are going to see a large number of transformation and modernization programs for core banking systems. Core banking systems support the bank's critical banking processes and products, such as personal accounts, cards, loans, etc. and process billions of dollars in financial transactions every single day. These transformations programs will aim to provide financial institutions with the necessary business agility to compete in an increasingly complex market, and a reduction in their high operating costs, through the use of cloud infrastructures and new technologies.

Many of these Financial Institutions have initiated transformation programs following different patterns. Some are trying to replace current systems with market solutions, now that these solutions are beginning to mature enough to be deployed in cloud environments. Others opt for tactical approaches of migrating current applications to cloud environments and new technologies, trying to minimize changes in them (an approach called lift & shift). However, the majority, at least among the largest institutions, have realized that technical transformations provide little value for the business and that lift & shift approaches are constrained to certain types of applications, with low coupling, and they are applicable to most core systems, tightly coupled and with very high operational requirements. These are the financial institutions that plan and have started to execute transformation programs that involve the redesign of the systems. This redesign is not trivial and many of these programs are at risk of repeating current application designs instead of reinventing business processes to adapt to new business models and new competitive and regulatory environments. Transformation-by-application rather than transformation-by-business-process approaches are especially at risk. To avoid this, it is important to be clear about the Target Stage Solution, the final stage of the transformation, the application architecture reference model that should guide the transformation.

This document deals with how to design this Target Stage Solution by analyzing current applications and business processes against a reference model of the banking industry, BIAN (www.bian.org). It details the analysis and design methodology and describes the most common findings and recommendations that originate in this type of project. Defining the transformation strategies of the different systems against BIAN allows the identification of business opportunities based on business capabilities described in the model that are not currently available in the legacy applications; opportunities to improve efficiency by identifying redundant functions; opportunities to evolve from product-oriented models to business domain models that allow the objectives and imperatives of the many and very different stakeholders in business processes to be adequately resolved.

The Target State Solution will also suggest important changes on the IT organization itself, from teams built around technical skills and technical platforms, to teams build around business capabilities.

The Target State Solution is defined by a domain model, the structure of which is described in the first chapter.

This document is a compendium of results and lesson learned in strategic definition projects carried out with our clients in different geographies.

2 What is a Domain Model?

A Domain Model structures the business capabilities in an Organization into Domains. Each domain is made of teams, which are responsible of the applications that support the business capabilities of the domain. All teams in the domain share a common business language and share Domain Experts, both from Business and IT. Domains are very independent from each other. A Domain is completely unaware of the internal details of other domains and relates to them also through standard interfaces such as API or Business Events. A domain contains one of several applications, but an application never comprises more than one domain. The applications can be of several kind, such as commercial solutions, SaaS, custom developments, etc. but never exceed the boundaries of the domain and never interact with other domains' applications with other than APIs or Events.

There are different ways to define Domain Models. We favor BIAN-driven domain model, which are domains models built with BIAN Industry Standard as Reference Model.

A BIAN-driven domain model structures the business capabilities of the Financial Institution in Business Areas, Business Domains and Service Domains.

- **Business Areas**: Large blocks in the value chain, with different business priorities. Very independent from each other.
- **Business Domains**: Responsibility groups, which provide similar related capabilities so that teams can specialize in specific knowledge areas, and which share a common business language. They represent a reference model for possible IT reorganizations. For practical purposes, you should have a single head of the IT organization.
- **Service Domains**: subdivide the domain into different responsibilities and is the government unit for the purposes of design, implementation or integration of applications. A subdomain is a black box for the rest that only see the APIs that it exposes. Internally, a service domain can be implemented by one or more components (for example microservices) but a component must not span more than one subdomain

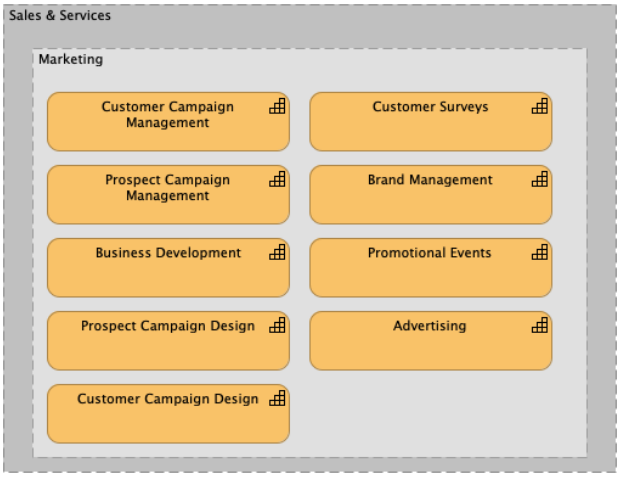


Figure 1. Example of Business Area, Business Domain and Service Domain

2.1 Business Areas.

Banking Processes are usually complex processes, where different actors intervene and where each actor has very different objectives to fulfill: there are commercial processes with client managers who wants to sell products, customers who need financing, deposits or manage investments but wants requests to be resolved quickly and in the simplest way; there are risk managers whose imperative is that the operation complies with the established Risk Policies; there are legal and internal audit departments obsessed that the signed contract is valid and does not give problems in the future; back office manager, who seeks to process operations at the lowest possible cost; a regulatory compliance officer who does not want sanctions on the bank for non-compliance; and so on.

How is it possible to reconcile such disparate and often conflicting interests in a single process? Who is the owner of each process? Can someone who design a process really put himself in the shoes of all the actors or is he always going to favor the objectives of one of the parties?

The Business Areas structure the business capabilities in large blocks in the value chain. Business Areas must be very independent from each other because each area has different business priorities and imperatives and requires very different skills. No teams should own responsibilities in more than one area. Dependencies between applications in different area should be minimal to none.

The following diagram includes the business areas according to BIAN v9 and enumerates the different business imperatives that each business area usually aims.

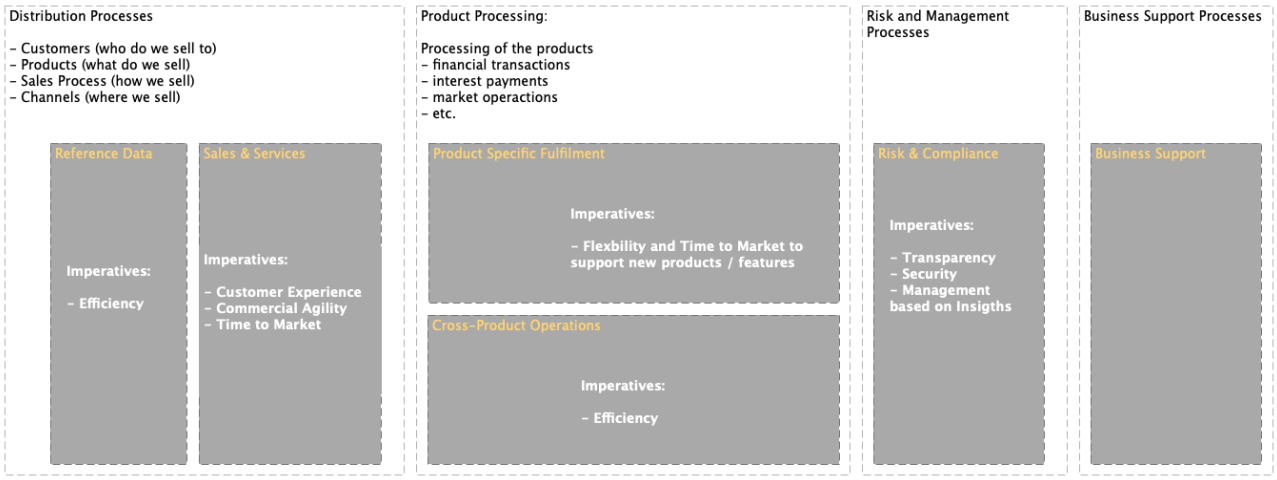


Figure 2. Key Strategic Imperatives in BIAN Business Areas

2.2 Business Domain

Business Domains subdivide the areas into groups of domains that provided related business capabilities and share a Common Business Language, SMEs and IT Development Teams.

For practical purposes, they should be owned by a single manager within the IT organization (and desirably, within a business units)

While the business areas will usually follow the model proposed by BIAN, the Domains will differ greatly, to adapt to the reality of each Financial Institution.

The IT organization should be designed to meet the business domains models, with different team assigned to different domains. The teams can them specialize in specific knowledge the domains require.

Figure 3. Example of Business Domains in the Sales & Services Business Area

2.3 Service Domains

Business Domains are structured in Service Domains. Each Service Domain has a unique business purpose, are elemental (not composed of other Service Domains), and collectively comprehensive (any business activity can be model using Service Domains).

Services Domains that implement business capabilities under the principle of **single business responsibility**. They expose the business capabilities using APIS or public business events. Interactions and collaborations are implemented between service domains, while business domains remains an organization and governance element.

Service Domains are implemented, integrated, deployed and/or consumed as Self-Contain Systems. They provide business capabilities exclusively through standard API and Business Events. A Service Domain can be implemented using any kind or combination of technical solution, from application servers, BPMaaS, IA solutions, etc. Service Domains do not share data nor transactional contexts with other domains.

Figure 4. Example of Service Domains in Customer Management Business Domains

3 Delivery Method.

The Method used for the analysis and customization of the Domain model is structured in 3 stage:

- **Application Portfolio Analysis**. Identification of business functions provided by each application and mapping to the reference model (BIAN Service Landscape)
- **Domain Model Customization**. Changes to the reference model to align to the Financial Institution´s Organization, Business Models, Preferences, etc.
- **Decomposition Pattern Description**. Identification of decomposition patterns that repeats for different applications. Used as a quality check and, specially, for use in external application design projects.

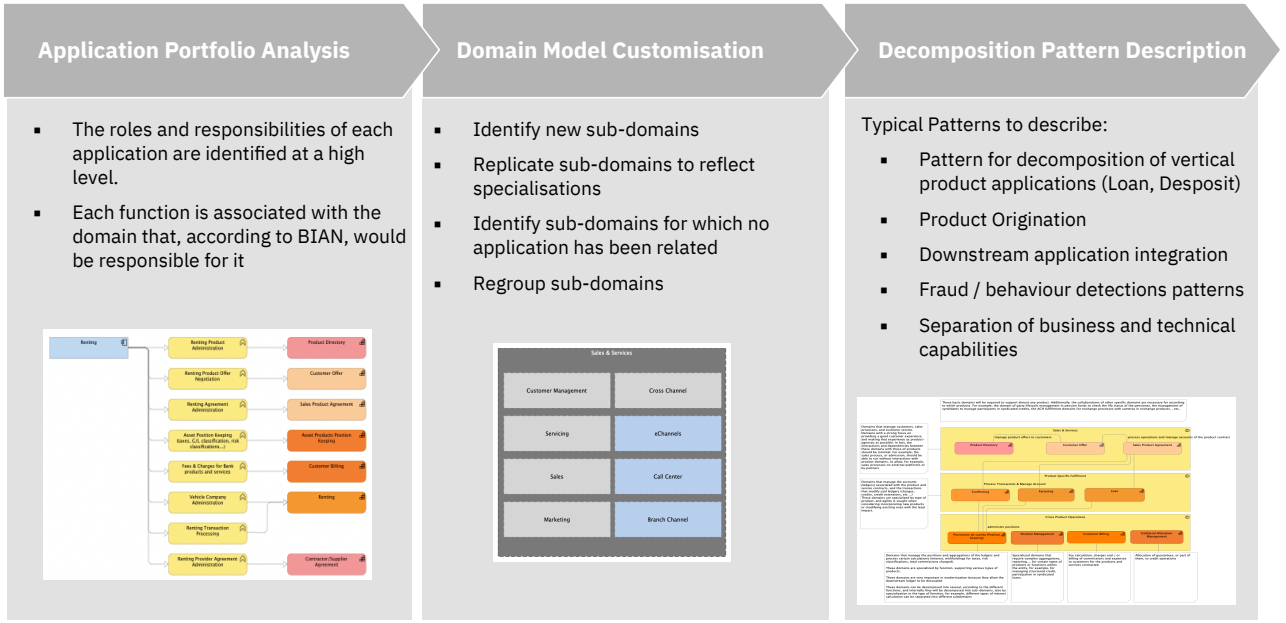


Figure 5. Delivery method for BIAN-driven transformation strategy design

3.1 Application Portfolio Analysis

In the first activity, every application in the application portfolio is analyzed. This means that the functions and responsibilities of each application are identified at a high level and each function is associated with the service domain that, according to BIAN, would be responsible for it.

Identifying functions that are repeated in several applications and associating them to the same domain; this enables the identification of opportunities for reuse and standardization.

When distributing responsibility by domains, the specialization of the teams responsible for the domains is encouraged.

In most of legacy applications, there will be a one-to-many decomposition of the application based on they function it implements. Having one-to-one mapping of application to service domains is usually a sign of poor analysis or understanding.

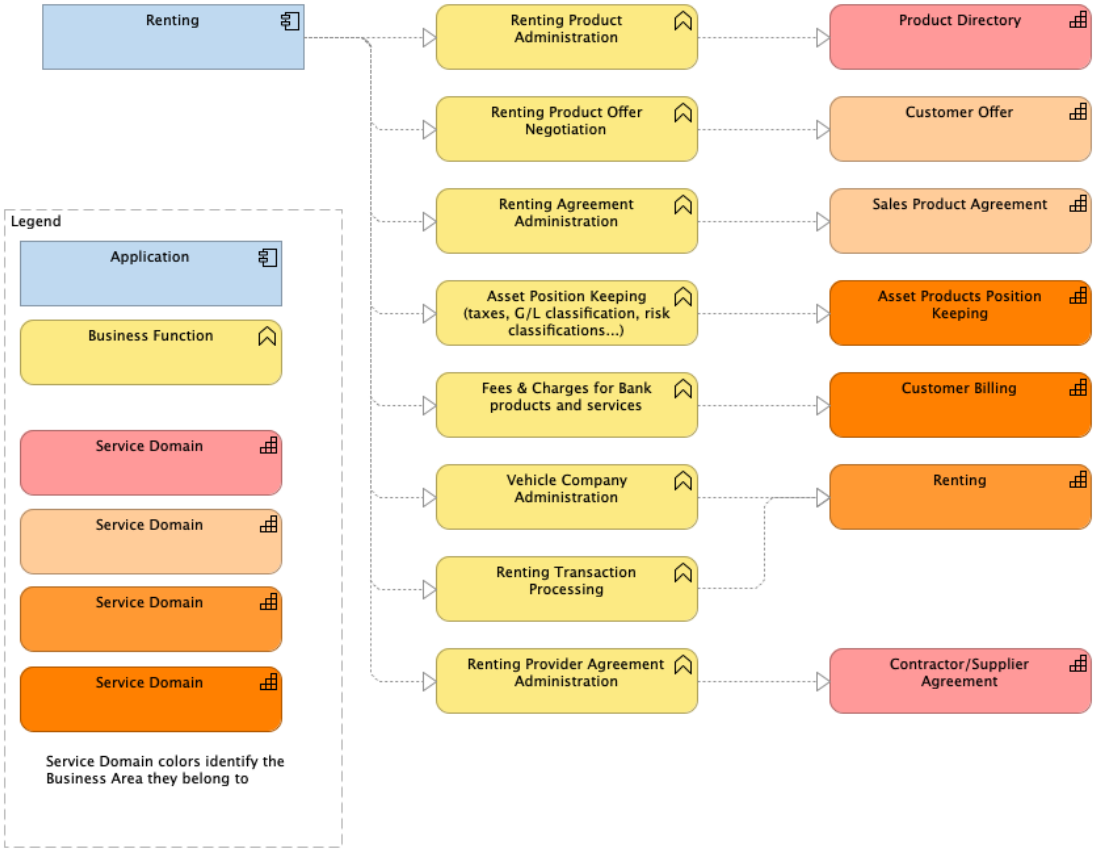


Figure 6. Example of application decomposition into Service Domains

3.2 Domain Model Customization

Customization of the reference model to the Financial Entity to reflect the different Business Models, Organizations, Products & Services, etc.

The customization involves identifies new service domains not existing in the reference model, specialization of service domains (split the domain in several according to different capabilities supported), identification of domains with no relation with any existing application, regrouping of service domains into business domains which will describe the recommended changes in the IT organization.

3.2.1 New Service Domains Identification

Identification of new sub-domains that do not exist in BIAN, rename sub-domains to the business language used in the Financial Entities. It´s important to use always the business language of the entity and not to force new terms which will not be understood by the Financial Institution stakeholders.

3.2.2 Split sub-domains to reflect specializations.

This is done when there are important functional differences or when there are different responsibilities within a Service Domain. The principle is not to have domains with more than one "owner", neither in IT nor in Business. Subdomains are split to avoid such situations.

For example, "Position Management" specializations for "Refinancing Management" or "Management of Structured Loan Positions".

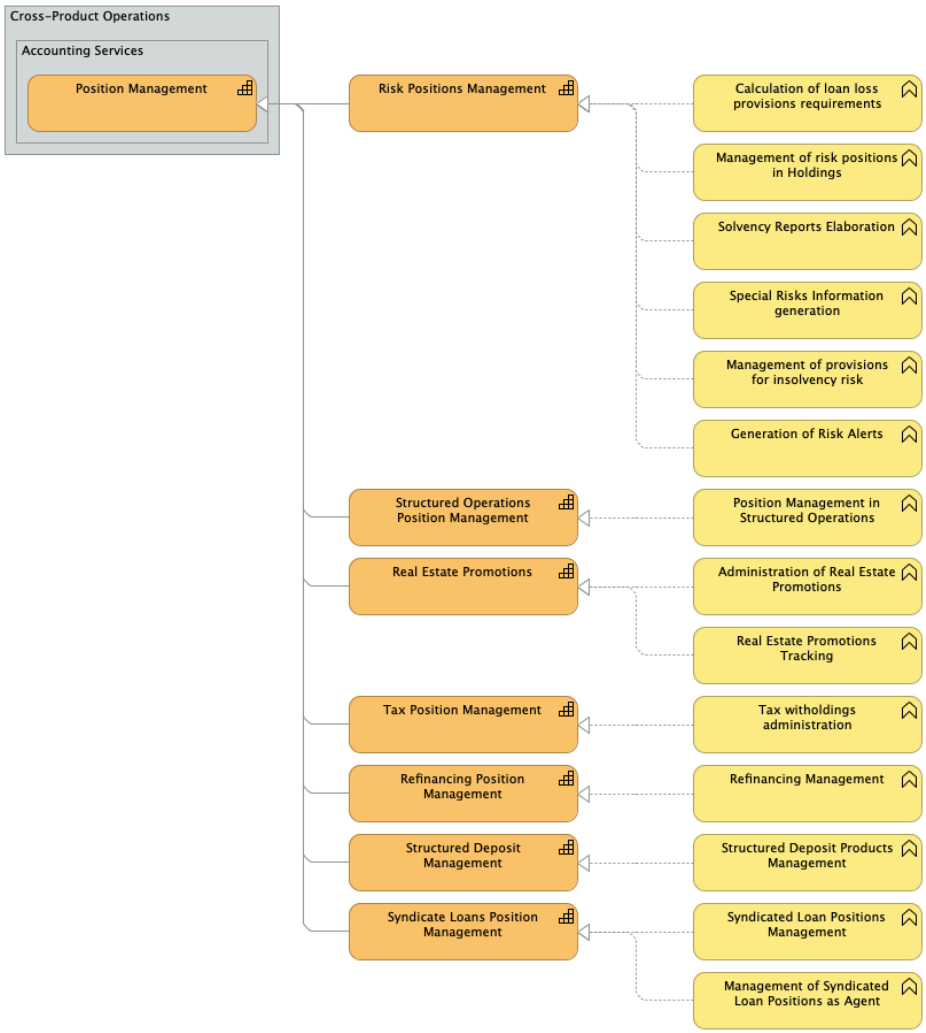


Figure 7. Example of Service Domain split to reflect specializations of responsibilities

3.2.3 Identify sub-domains for which no application has been related

These situations are carefully analyzed because it can be caused by an incorrect analysis (most banks will have capabilities in most of the BIAN domains) but also can lead to business opportunities. For example, “product matching” or “product training” are usually capabilities not provided by legacy systems. Developing such capabilities will be of value to the Financial Institution. The Gap Analysis help to identify potential business opportunities. The next diagram is an example of a gap analysis. Service Domains in green could not be mapped to any existing legacy application. Many of them were later identified as business opportunities.

Figure 8. Example of Gap Analysis

3.2.4 Re-grouping of Subdomains into Business Domains

BIAN structures the service domains into Business Area and Business Domains. The first one represents the different steps in a value chain, and the second a grouping by responsibility.

The grouping into functional domains must be changed to adapt the domain model to the responsibilities and organizational model within the Bank. This is very important. Otherwise, the business domain will not provide value further than from a communication or presentation purpose.

Our PoV is to use Business Domain to group together domains that will share business language, Domain Experts and IT Teams responsible for the development, maintenance and evolution of the applications in the domain. This is a key driver for an institution to adopt Domain Driven Design practices.

This outcome is one of the most valued by the customer we have worked with.

The final result will provide recommendations on changes in the organizational model, and should be carried out maximizing the knowledge and experience of the different teams of the entity (grouping sub-domains around teams with the most appropriate skills and experiences for them)

3.3 Decomposition patterns description

This activity identifies and describes patterns of decomposition that are repeated for many applications.

The objective is that these patterns can be applied in future transformation projects, even with applications that have not entered in the scope of the domain model design.

For example, product applications almost always break down into:

- Product management.
- Sales Process (Account opening, Loan origination, etc.)
- Agreement management.

- Account processing and management (operations that affect the product ledgers)
- Classifications, aggregation and accounting treatments (taxes, withholdings, ...)
- Calculation and settlement of fees and service expenses.

To these sub-domains others are added depending on the characteristics of the product (guarantees, management of participants in products involving groups of parties, management of third parties in services in which the bank acts as a "broker", etc.).

Some typical patterns that usually emerge in the analysis are:

- Pattern for decomposition of vertical product applications (Loan, Deposit…)
- Product Origination.
- Downstream application integration.
- Fraud / behavior detections pattern.
- Separation of business and technical capabilities.

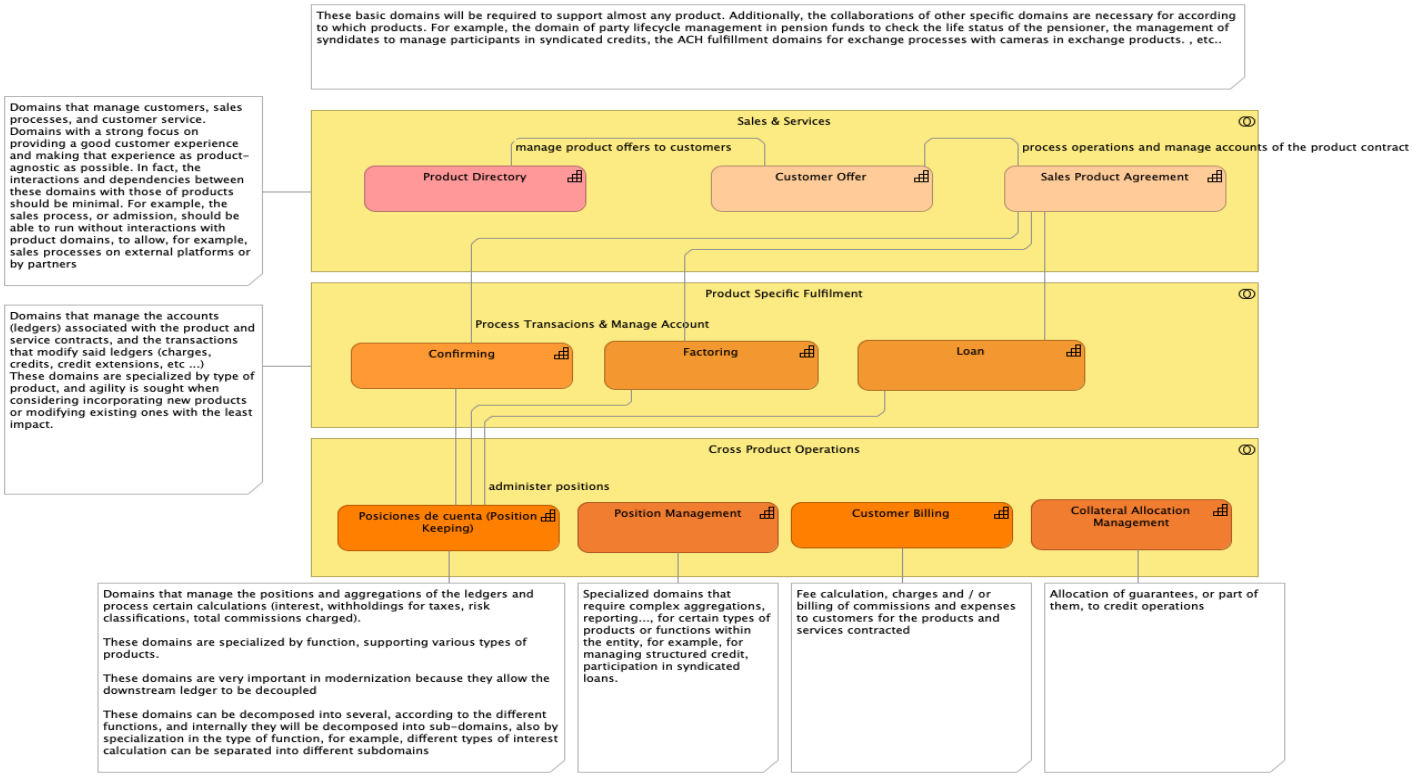


Figure 9. Example of decomposition pattern: Product Silo Decomposition

Next picture provides an example on how this pattern applies to a siloed product application

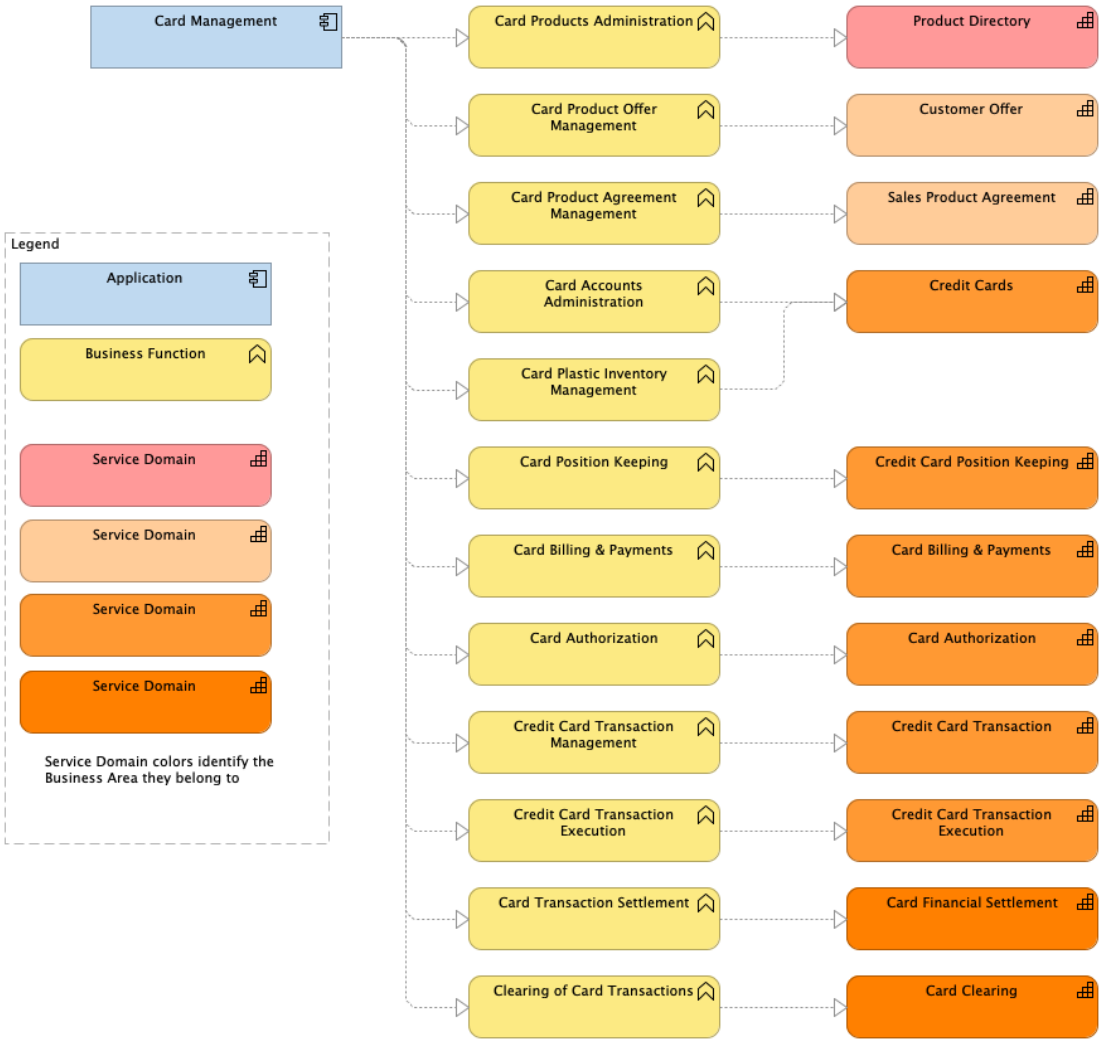


Figure 10. Example of application where the siloed product decomposition pattern is used

4 Findings & Recommendations

This section of the document resumes the typical findings that are usually found in most of customer engagement. They are included here with the purpose of:

- Help the practitioner on what situations to look for.
- Provide recommendations of potential business opportunities to be identified when these findings are confirmed.

4.1 IT is organized by Technology Skills, and not around Business Capabilities.

The application teams are usually strongly organized by technology and not by business domain.

There is usually a strong separation between operational, informational and channel application. Each of them has different managers and teams but all of them sharing responsibilities in domains such as clients, assets, services.

The first consequence is the distribution of the business logic of a domain in multiple systems with different responsible teams, making maintenance difficult (resolution of incidents is complex) and evolution, and creating inefficiencies.

In order to apply Agile Methodologies, DevOps practices, or new styles of architectures such as microservice architectures, it is necessary to change the organization to teams organized by domains, responsible for end2end business capabilities, and capable of managing the different technologies that are required.

It may be challenging for teams having to cope with a wide range of different technologies for their applications. This can be helped if the technical architectures or the application platforms facilitate the use of the technologies and abstract the application developer from technological complexity moving low-code environments. Also, modern agile organization bases on squads, tribes, chapters and guilds (as suggested by, for example, Spotify), will help with this challenge.

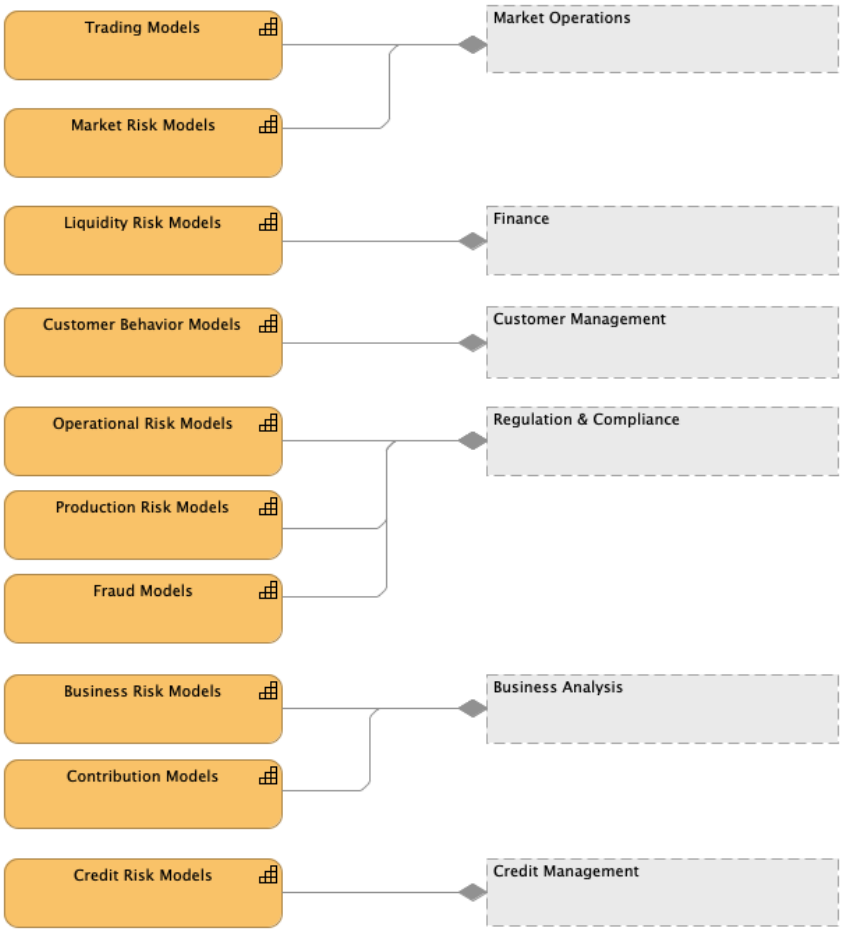
If we look at BIAN Service landscape, it´s evident that the business and service domains are around business capabilities and not technology platforms. There are no domains of the type "data-lake", "DWH", "integration platforms, models ....". Those would be cross-domains, technology oriented and business logic and data models belonging to multiple service domains. If an IT organization is to follow BIAN standard, it would avoid this type of “technical” organization.

The Domains are channel agnostic, providing the capabilities regardless of the access channel. The consequence is that channels should not implement business logic from any domain and should focus on providing interaction experiences in the most efficient way for the characteristics of each channel.

The clearest examples of this findings can be found in almost every Financial Institution in the data architectures or analytical systems. Traditionally, the applications have been separated in two kind “operational systems” and “informational systems”. The operational systems include mostly the transactional services and run usually in mainframes or application servers. The informational systems include the DWH, data-lakes, data-marts, reporting systems and are usually build on dedicated data platforms. There are always different teams in charge of the operational and the informational systems. The data and business logic from the business domains, such as Deposits, Loans, Customers, Payments… are now dispersed in different platform under control of different teams and the result is always the same: lack of data consistency, business logic duplicated, inefficiencies in the maintenance, etc.

BIAN provide a disruptive approach to move from this situation to an organization where teams are structured by domains, and the team can make use of the different technical platforms or services that better suit for each situation.

The next diagram includes a number of BIAN service domains whose applications are usually found in the “informational systems” under cross-domain, technical team specialized according to the technical data platform they are deployed on. Under this new approach based on domains, they would be managed by domain teams, experts in a specific business domain and who are responsible for both transactional and informational systems for the business domain.



Other important examples of this findings that could be found in many financial institutions and usually related to the use of middleware. For example, there are specialized teams to create and maintain business processes because the use a BPM solution, or APIs, because the use the API Manager. They are different teams from the ones who really owns the business applications (for example, the customer onboarding process is maintained by the BPM team and not the Customer Management Team who managed the customer database). This results again in scattered business logic across technical specialized team, leading to inefficiencies and redundancies.

There are a significant number of these technical team created around both commercial and in-house developed middleware, that distribute business logic from one domain to different technical elements managed by separated teams.

What is worse, in some of these applications there is a strong coupling between function and technical application making very difficult to extract the logic and move it to a different technical solution. This is more frequent in in-house developed middleware but is also found on commercial solutions with proprietary standards.

Although the strategy of having highly reusable services such as these applications is very efficient, it is important to separate the responsibilities and in no case that those responsible for the cross application are the ones who implement the business logic of the domains.

These applications should offer “technical capabilities”, so that domain teams implement their own processes and functions, will implement them using the technical platforms and should completely own the software lifecycle of their processes and functions,

For example, when using DB2, each application creates, owns and is responsible for its tables. This separation of responsibilities has been lost in recent times with teams dedicated to developing business logic that belongs to other teams, in middleware solutions such as APIs, BPMs.

4.2 Moving from Product-Oriented application to Value Chains

Most of financial institutions have organization heavily structured in product areas. It will make it difficult for processes to be designed with the customer's experience in mind, or to maximize reusability and specialization by functions and capabilities.

This is reflected in product teams being responsible for all the business processes in front, middle and back-office. The processes are specific for a line of products and are not reusable with other lines of products.

This can be identified by:

- Different sale processes (account opening, loan origination, credit card applications…) for type of products.
- Lack of existence of a commercial product catalogue.
- In extreme cases, lack of single customer database, with several customer databases for each type of products.
- Risk Assessment and Underwriting deeply couple in the loan originations
- Product specific back-office operations, specialized and implemented for a specific product and not by business capability.

The next diagram shows the decomposition of a typical product applications. In current implementations all the functionalities are exclusive of the product and not reusable across products. But after the assignment of the functionalities to the BIAN domains it´s evident that all but one of the service domains actual could provides cross-product, reusable capability.

Moving to such a capability model is a source of countless efficiency gain for the Financial Institution, by creating product agnostic capabilities that will replace tens of equivalent functions in tens of product specific applications.

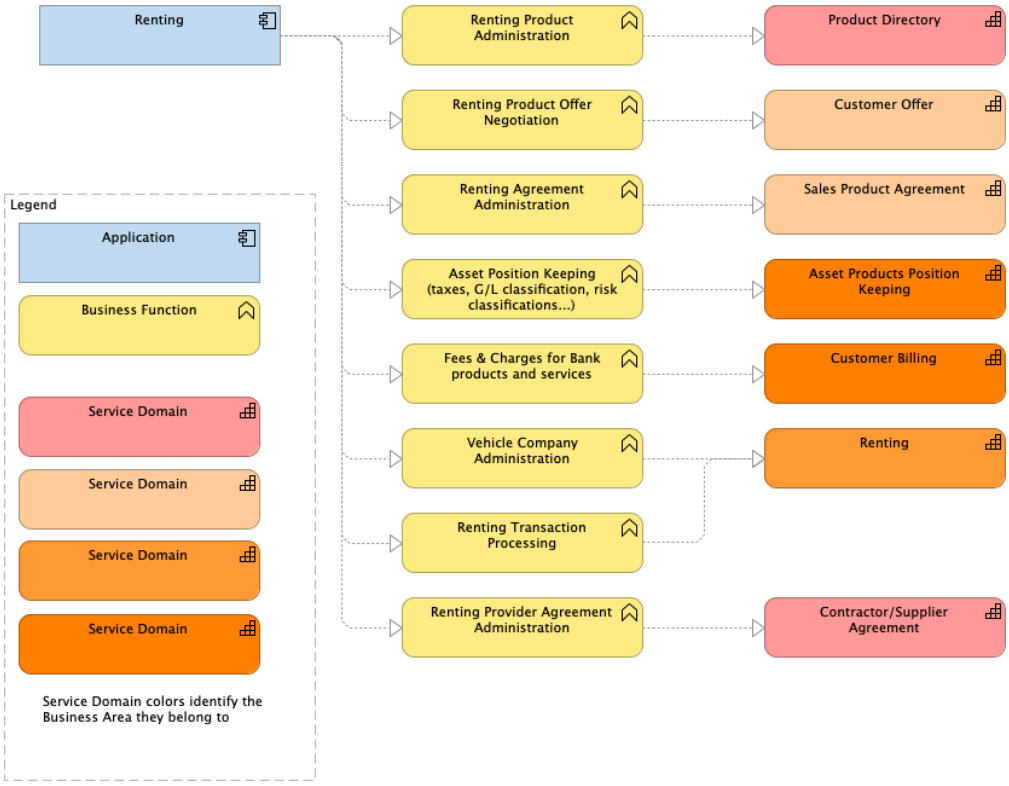


Figure 11. Decomposition of a product silo into domains. All but one of the functions identified could be provided by common business capabilities.

4.2.1 Strict separation between domains of Sales & Services, Products Processing and Cross-Product Operations.

The BIAN Service Landscape clearly separates the areas of Reference Data, Sales & Services, Product Fulfilment, Cross-Product Operations, Risk & Compliance, and Business Support. A BIAN-driven transformation will restructure product-centric business processes breaking them down to capabilities in each of the areas

The business domains within each area have to be very independent so that the responsible teams can focus on the imperatives of that area.

For example, **Sales & Service** processes will have a clearly commercial and user experience improvement imperative. Those responsible for **Product Fulfilment** domains will have as an imperative agility when incorporating new products and the differentiation that they can provide. Those responsible for **Cross-Product** Operations will have as an imperative the efficiency and reusability of the capacities they provide.

This model of separation, reflected in BIAN and other reference models, can be observed in other industries. For example:

- Retailers such as Amazon clearly separate their commercial processes (the portal, with its product catalog, unique processes and personalized experience; the products that they sell directly or from other sellers; the collection and payment operations, commissions, logistics ...)

- Car dealerships, even multi-brand, who are the owners of the clients, manage the commercial catalogs and sell the cars, while the factories (equivalent to the processors of products in the bank) only receive the signed contracts, with the options and colors of the car to be manufactured

This change in paradigm is also a market trend in Banking Industry: corebanking solution begin to specialize their products in this direction. “Figure 2. Key Strategic Imperatives in BIAN Business Areas” highlight the different imperative of each different area.

4.3 Missing Business Domains help to identifying business opportunities for agility and efficiency

Those BIAN Business and Service Domains that cannot be easily map to existing applications but, in the contrary, maps to lots of functionality distributed in different product, channel or analytical applications are great candidates to identify business opportunities. In most of the cases, they will reduce the number of duplicated capability and will decouple the business functions from the product and channel processes, making them much easier to evolve and improve.

The following sections describes those BIAN business domains usually found under this situation and, in consequence, a great source for such opportunities.

4.3.1 Reference Data

The reference data area centralizes information spread over a multitude of applications and used by many other ones. This grouping facilitates its management, quality and availability for the rest of the business domains. This area includes the following domains

- **Product Management**: Centralizes all the management and design of commercial products and their marketing conditions based on templates of the operational products supported by the product applications, or third-party products marketed by the financial institution.
- **Party Directory**: Centralizes the management of the information of all individual and organizations who participates in the business processes. This domain enables the separation of the information of the party (demographics, contacts, identifiers, etc.) from the roles that party (individual / organization) plays in the processes, such as customers, business, supplier, guarantors, etc. Each of the roles is managed in specific business domains while the centralized management of the party enables the centralization of common processes (i.e., Data privacy, Consent registrations, etc.) and provide a unified view of all the relationships of the individual or organization with the institution.
- **Public Reference Data**: centralizes the capture and management of external data, generally standardized, of interest to the domains. For example, repository of financial entities, repository of countries, regions, cities, etc.
- **Interbank Relationships**: Centralizes the business capabilities required to manage the relationship and interaction with other financial entities, such as correspondents, participation in Credit Unions, etc.
- **Market Data**: Centralizes the access and distribution of financial market data and the management of information providers
- **External Agency**: Centralizes the management of relationships with external suppliers, agencies and service agencies, as well as market agreements with third parties.

4.3.2 Cross-Channel Capabilities

The rapid growth of digital channels in traditional financial institutions led to duplication of channel functionalities in channel applications that, by their nature, would clearly be reusable and managed more efficiently in a centralized way. Some institutions created “multi-channel architectures” that may provide some of these cross-channel opportunities, but they are the exception, and, in any case, they will provide a subset of the capabilities and with limited scope (for example, only used by digital channels, but not by assisted channels whose development were older than the multi-channel architecture itself)

Although these functionalities are dispersed in different applications, it is usually the case that the origin of the requirement is the same for all of them (security, regulatory, commercial…) and have to be distributed to different teams. This situation also benefits from centralizing these functions.

The capabilities in "Cross Channel" domain include:

- Client Authentication
- Access Entitlement Management
- Contact handling and routing
- Transaction Authorization
- Analysis of channel activity
- Management of service points, which configure the characteristics of access to the bank services of each user based on their role and attribution

The following is an example of information and responsibility for customer authentication which is spread in different applications and that can be optimized by consolidation in Cross-Channel domains.

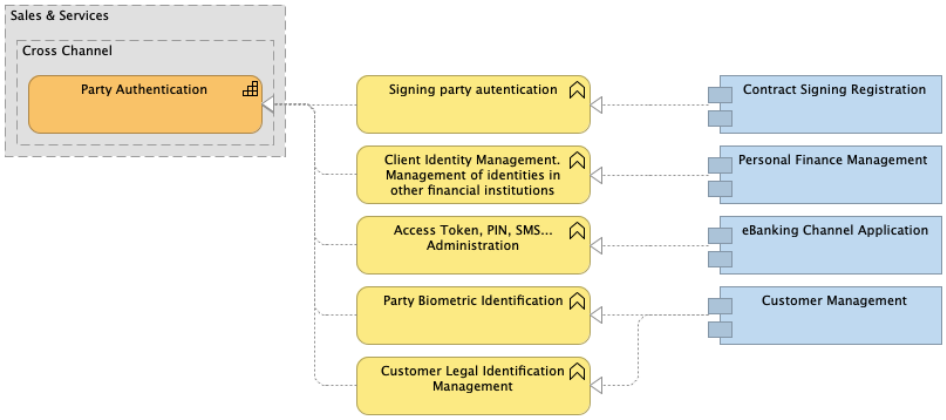


Figure 12. Customer Authentication capabilities spared in different application can be centralized in a Service Domain.

4.3.3 “Sales” Business Domain: Sales Processes versus Product-Specific Opening Processes

There are currently no sales processes in the financial institutions, but rather product-opening processes, which indicates a strong orientation to the product and to regulatory and risk control aspects.

The new approach treats the account or loans opening processes of any product as commercial, digital (digital-first) processes, supported by risk management and regulatory processes according to the characteristics of the product in question, processes supported by the corresponding domains of the areas of operations and risks

The sales process is supported by capabilities that must be differential for the entity, such as advice on the choice of products (Product Matching service domain), hyper-personalization of offers (Sales Product), and the integrated management of offers multi-products (product matching)

This approach allows the financial entities to move from strategies to sell a specific product, into commercial strategies that begin with the customer business needs and wants, and identity the product or bundle of products that better fits this need. It is moving from product-oriented to real customer-oriented.

The following picture describe how a traditional loan origination process is split and the required capabilities redistributed to cross-product, reusable business capabilities that can be developed using different business imperatives (efficiency, customer experience…)

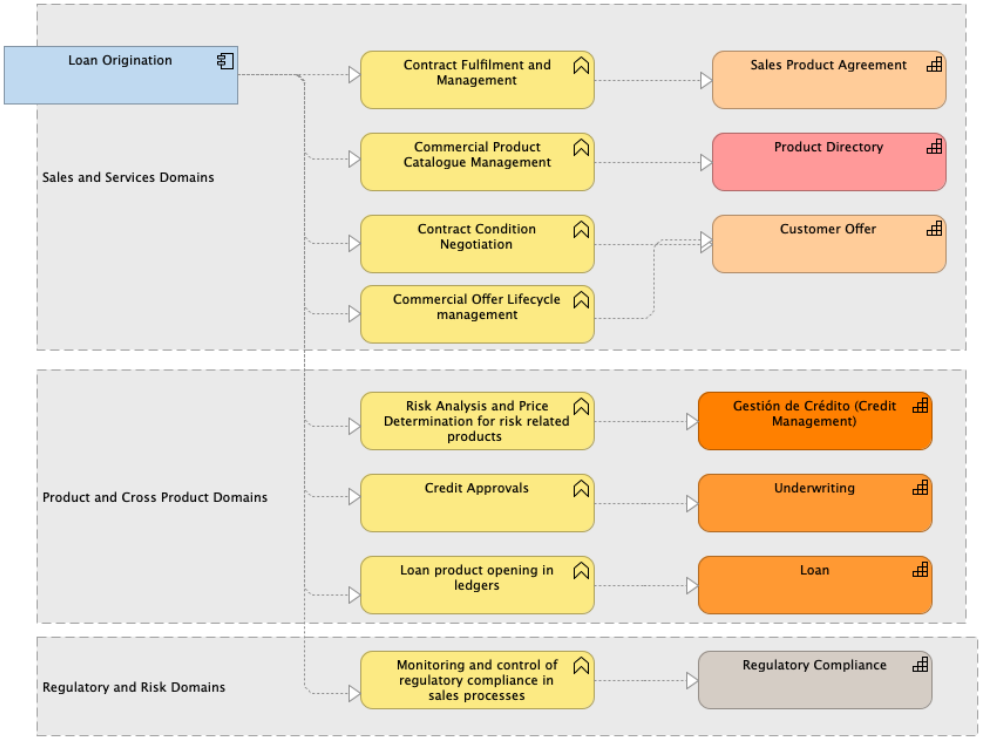
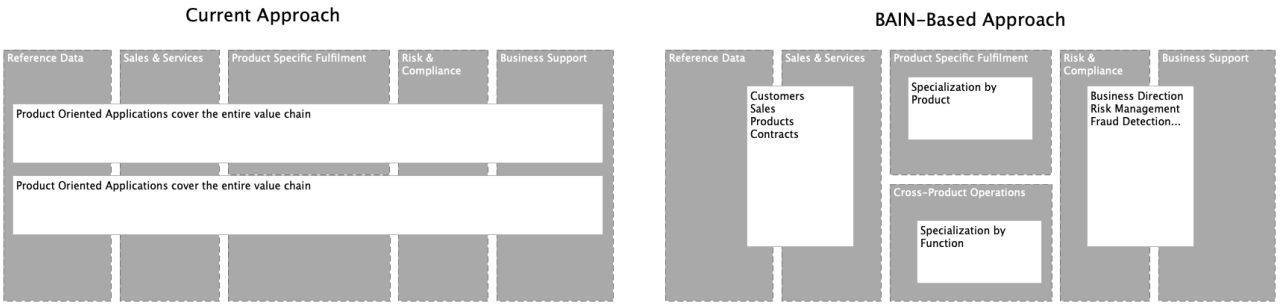


Figure 13. Decomposition of a Loan Origination Process

4.3.4 Restructuring of Products Applications

The product and cross-product domains specialize in product processing and are supported by cross-product support processes: for example, there may be different applications for personal loans, or mortgages (specialization by product), but all are supported by common risk, accounting or reporting processes (specialization by function)



In the product domains, a specialization by product is desired (to support the specific operations of each type of product, for example Personal Guaranteed Loans, Mortgage Loans ...). In cross-product domains, specializations by function are preferred, which will support all products or product groups (for example: risk classifications, calculation and management of tax obligations, classifications for accounting, reconciliations between accounts, etc.).

The different approached provide efficiencies due to reusability and specialization of the teams in complex functions and enables handling different product processors without becoming complex applications that are difficult to maintain. This also help to decouple the product responsible for the product ledgers, and the downstream processes. Coupling of ledgers and downstream processes are the most acute constrain to time to market in corebanking systems.