Written by
Ilia Pavlichenko

Designing a Product Group

An Eight-Step Design Algorithm

December **2025**

CreatingAgileOrganizations.com

Executive Summary

This whitepaper presents a practical, repeatable algorithm for designing a **product group** — a semi-autonomous organisational unit responsible for the end-to-end creation, development, and financial performance of a product family. The method is grounded in the multidimensional organisational design of Russell Ackoff and the principles introduced in *Creating Agile Organizations* .

The paper demonstrates the full design process using a real case of establishing the **Daily Banking Product Group** in a large financial organisation. The algorithm starts from defining the mission and essential parts of the unit, identifies the organisational capabilities required for strategic differentiation, and progressively narrows the scope of the product group by analysing:

- **frequency and specificity** of functions and components (based on Transaction Cost Economics),
- **operational dependencies** (using James Thompson's framework),
- **criticality and uncertainty** of supporting functions,
- **leadership models** suited for both product and line management,
- **functional coupling** between units (based on axiomatic design),
- and the feasibility of forming Value Areas and cross-functional teams.

This structured approach made it possible to design a product group that balances **agility and scale**, minimises transactional and coordination costs, and increases the organisation's ability to deliver value end-to-end.

The final design for Daily Banking includes:

- two types of cross-functional teams (Digital Core and Integration Cluster),
- a Marketing & Sales shared capability,
- IT Security as a matrix-coordinated but independent control function,
- and a Triad Leadership model, which proved the most effective for managing diverse technical and commercial units.

The result is a coherent organisational architecture where strategy, structure, capabilities, and flow are aligned — enabling faster product delivery and greater adaptability.

The Product Group Concept

In this section, we introduce two key concepts on which the subsequent organizational design is built: the product group and the product family.

A **product group** is a formal organizational unit responsible for creating, developing, and achieving the financial outcomes of a product family.

A **product family** is a group of interrelated products that share a common architecture and/or common functional elements and are developed on the basis of a unified product platform to meet similar customer needs or market segments.

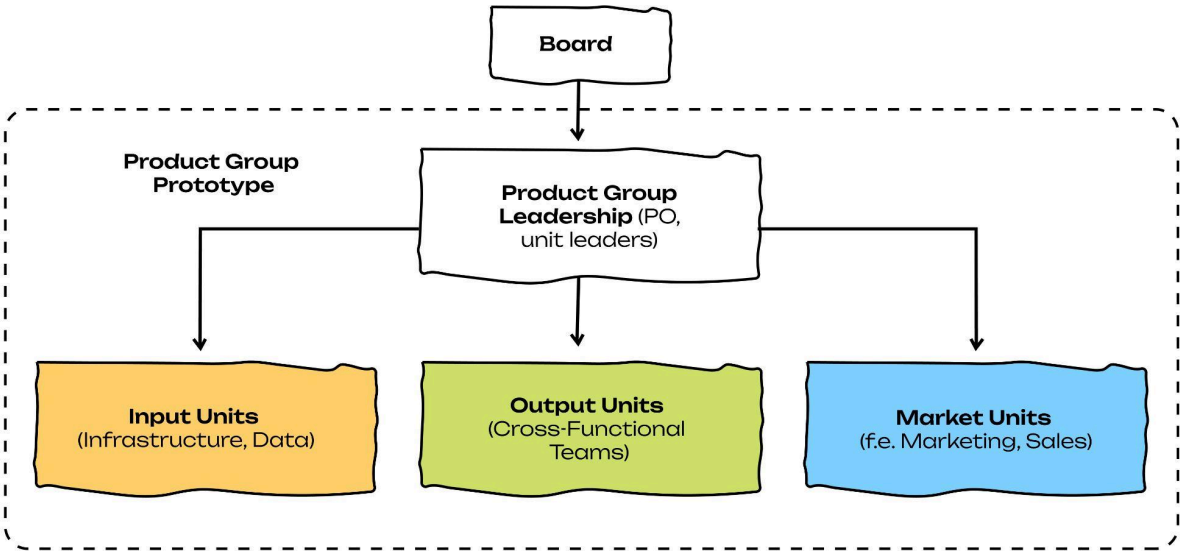
The concept of a product group, described in the book *Creating Agile Organizations*, is based on the multidimensional organizational design of R. Ackoff (*Re-Creating the Corporation*). In a multidimensional organization, each structural level contains three types of units:

- **input units** — provide other departments with resources;
- **output units** — create the product;
- **market units** — work with customers and the market.

At the same time, the importance of each unit is determined by the strategic focus.

The product group structure includes:

- **cross-functional teams** as output units that create the product end-to-end;
- **shared market** and **input** functions within the group, such as marketing, digital sales, or platform services;
- **Product Group Leadership** — the Product Owner and unit leaders who set direction and create the conditions for the group's work.



Product Group Prototype

Thus, a product group is a semi-autonomous multidimensional unit that integrates the functions of product creation, market interaction, and internal services. This allows decision-making to be optimized, transaction costs to be reduced, and the end-to-end flow of value to be accelerated.

Design Principles

When designing the structure of a product group, we rely on the following principles:

- **Capability-based design.** Organizational design is built from the required capabilities: first we determine what the organization must be able to do, and then we create the structure, roles, and processes that make these capabilities real and sustainable.
- **From whole to parts.** Organizational design begins with the entire system, its mission, and its role in the business, not with individual functions. First — the purpose and architecture of the whole, then — filling it with elements.
- **Balance of agility and scale.** Organizational design must combine the advantages of decentralization (speed, adaptability) and centralization (standards, efficiency, scale). The task of leaders is to find the point of balance between the autonomy of product groups and the cross-organizational benefits of scale.
- **Minimization of transaction costs.** The structure should reduce the costs of interaction — coordination, switching, approvals, and adaptation. Good design places functions where the cost of their interaction is minimal.

Algorithm for designing a product group

The presented algorithm for designing a product group follows the principles described above, and to carry out the algorithm we need to answer a number of questions.

Following the principle “from whole to parts,” everything begins with forming the mission of the unit (its purpose) and then identifying its essential parts. The essential parts are the necessary and sufficient set of system elements that allow it to perform its function. But not all essential parts of the system should be included in the unit by default: some of them may remain corporate centralized functions for the sake of economies of scale.

An Eight-Step Design Algorithm:

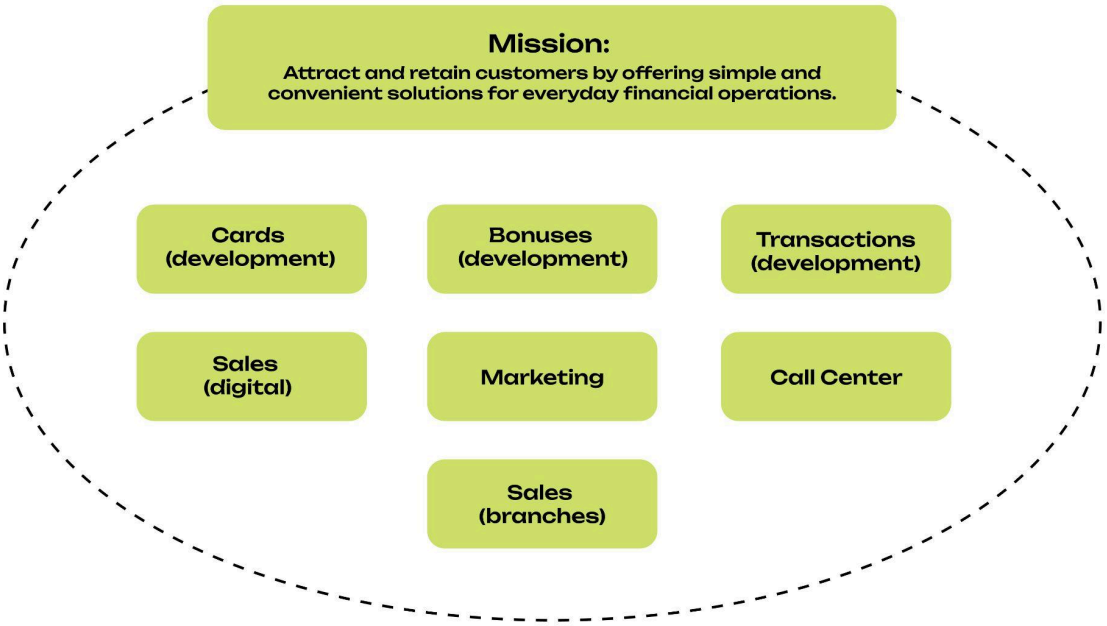
1. What is the mission of the unit and what essential parts are needed to fulfill it?
2. What capabilities must be preserved and developed?
3. What are the most frequent / specific components and functions?
4. What operational dependencies exist between the parts?
5. Which supporting functions should be included, considering the Cost of Delay (CoD) and criticality / uncertainty?
6. What model of line and product leadership?
7. Which organizational design has minimal functional coupling?
8. What are the Value Areas and what is the composition of teams?

At the first step, following the “from whole to parts” principle, we fill the product group with the widest possible set of elements. Then, step by step, we begin to cut off the unnecessary, following the principle of balancing agility and scale.

What is the mission of the unit and what essential parts are needed to fulfill it?

The mission clarifies the reason for the product group’s existence as a separate unit within the larger organization, points to the customer and the outcomes they will receive. Management formulated the mission of the product group “Daily Banking” as follows:

Attract and retain customers by offering simple and convenient solutions for everyday financial operations.



Mission and essential parts of the product group

To fulfill the mission, management identified the following essential parts:

- Cards
- Bonuses
- Transactions
- Sales (digital)
- Marketing (advertising)
- Call center
- Sales (payroll clients)
- Sales (retail in branches)

What capabilities must be preserved and developed?

Based on the strategy of the retail business in which the product group **Daily Banking** operates, the key organizational capabilities were defined as:

- **Adaptability** — quick response to changes in the market and technology;
- **Rapid launch of new products**.

The selection of these two capabilities indicates a **product-centric focus** of the product group.

What are the most frequent / specific functions?

At this step of the algorithm, we determine which functions and components should be located inside the Daily Banking product group. The primary analytical tool is a **Heat Map of interaction frequency**, which shows how often each function participates in feature delivery.

We classified interaction frequency into four categories:

- **Rare** — 0–20%
- **Occasional** — 20–40%
- **Frequent** — 40–80%
- **Constant** — 80–100%

Frequency alone, however, is insufficient for making structural decisions. Therefore, we introduce a second dimension — **specificity**, derived from Transaction Cost Economics (TCE) and the works of Ronald Coase and Oliver Williamson. Specificity reflects how strongly a function depends on unique product context, architecture, and domain knowledge. In TCE logic, highly specific functions should remain within the unit, while low-specificity functions can be centralized without increasing transaction costs.

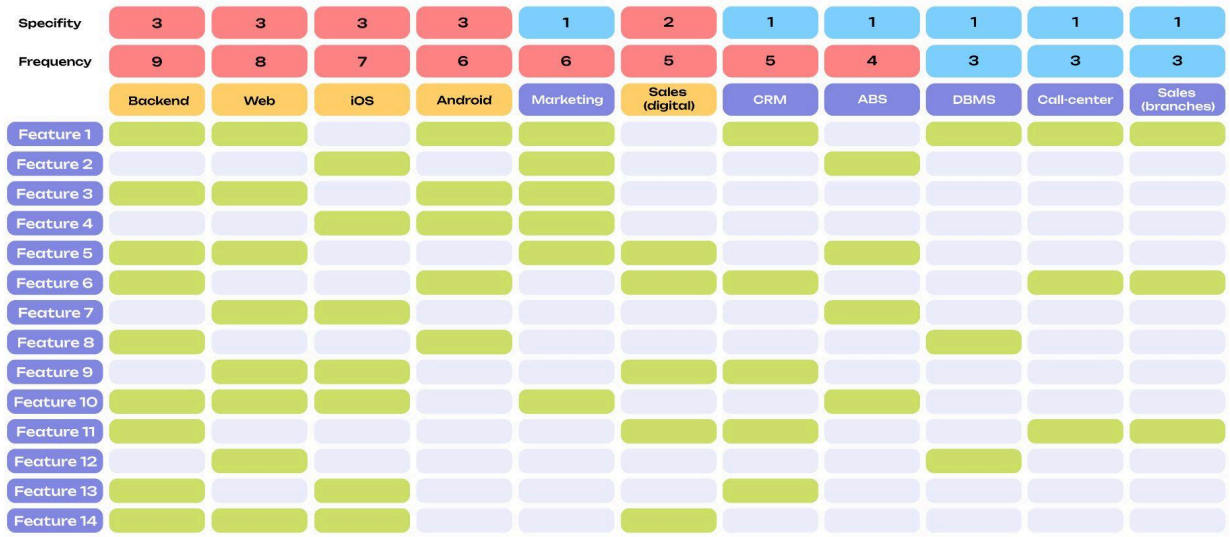
We use a four-level specificity scale:

- **0 — Low**: standardized, repeatable, generic function
- **1 — Moderate**: requires some domain context but is easily reusable
- **2 — High**: product-dependent, requires deep domain expertise
- **3 — Very high**: critical for key scenarios and tightly coupled to the product core

To assess specificity in the context of Daily Banking, we asked the following questions:

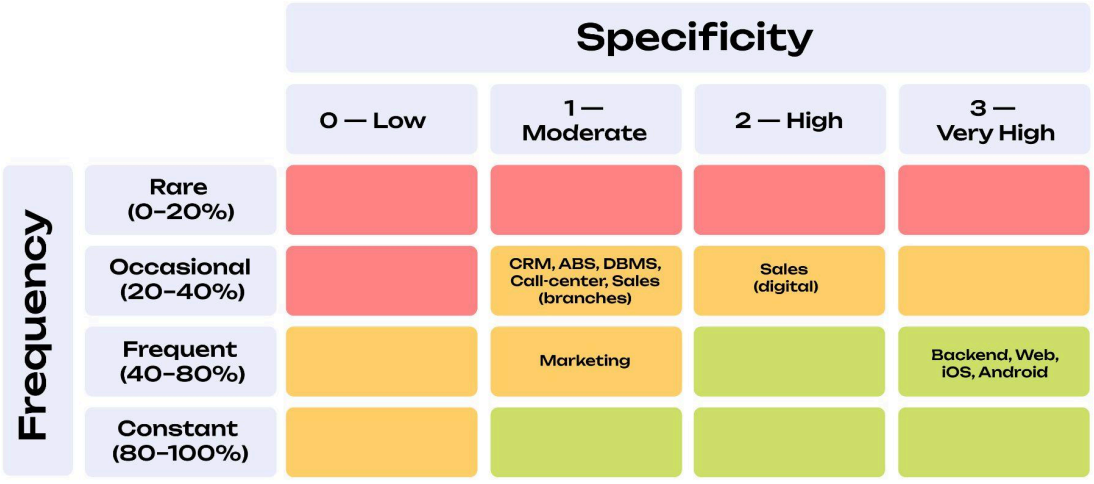
- How strongly does the function depend on unique product or domain context?
- Can it be reused across multiple product groups without modification?
- How deeply is it embedded in the product architecture?

Combining **frequency** and **specificity** allows us to justify which functions should be included in the product group and which should remain part of a centralized platform. To focus on the majority of the workload, we applied the Pareto principle. We analyzed 14 major features; therefore, we considered functions appearing in more than three features as sufficiently frequent.



Heat Map by frequency and specificity

This frequency–specificity logic is directly linked to one of the key organizational capabilities we aim to develop — **rapid launch of new products**. The objective is to create a semi-autonomous unit with minimal external dependencies. Based on this logic, we constructed a decision matrix that distinguishes between functions that should be included, excluded, or require further consideration.



Decision matrix for frequency / specificity

Based solely on the frequency–specificity analysis, **CRM, ABS, DBMS, Call Center, and Branch Sales** do not qualify for automatic inclusion in the product group at this stage. Their interaction

frequency is occasional, and their asset specificity is moderate, placing them in the **“consider” zone** of the decision matrix. From a Transaction Cost Economics (TCE) perspective, this means that centralized provision remains economically reasonable, but the decision is not definitive.

In contrast, several functions **clearly meet the criteria for inclusion**. **Backend, Web, iOS, and Android** combine high or very high asset specificity with frequent interaction, making internalization economically justified and critical for end-to-end delivery.

At the same time, **Marketing and Digital Sales also fall into the “consider” zone**. Their interaction frequency is relatively high, but their asset specificity remains moderate, which does not provide sufficient economic justification for automatic inclusion at this stage.

Function / Component	Frequency (count)	Frequency (%)	Specificity	Status
Backend	9	64% (Frequent)	3	Included
Web	8	57% (Frequent)	3	Included
iOS	7	50% (Frequent)	3	Included
Android	6	43% (Frequent)	3	Included
Marketing	6	43% (Frequent)	1	Consider
Sales (digital)	5	36% (Occasional)	2	Consider
CRM	5	36% (Occasional)	1	Consider
ABS	4	29% (Occasional)	1	Consider
DBMS	3	21% (Occasional)	1	Consider
Call-center	3	21% (Occasional)	1	Consider
Sales (branches)	3	21% (Occasional)	1	Consider

However, these conclusions reflect only the **economic boundary** of the unit. To account for operational realities and coordination costs, we introduce an additional analytical lens in the next step — **the type of operational dependencies**.

What operational dependencies exist between the parts?

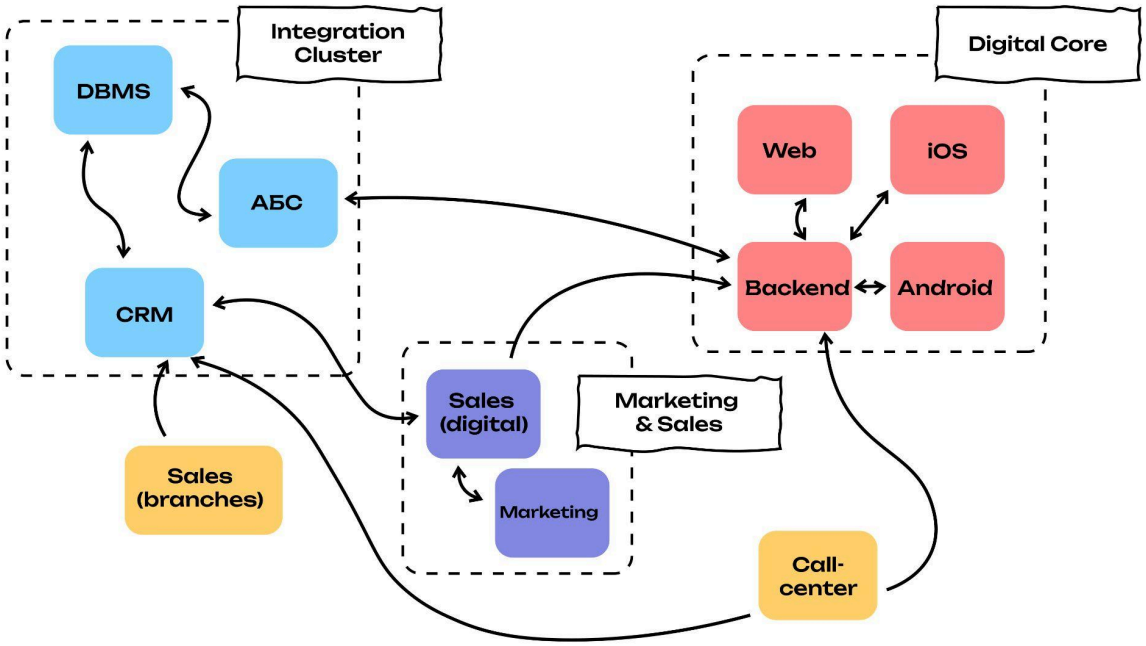
The frequency–specificity analysis defines the **economic boundary** of the product group and identifies a set of functions that fall into the *consider* zone. To resolve their placement, we introduce an additional analytical lens — **the type of operational dependency**, following the framework proposed by James D. Thompson (*Organizations in Action*, 1967).

Thompson distinguishes three types of operational dependencies:

- **Pooled** — units share common resources but can operate largely independently;
- **Sequential** — the output of one unit becomes the input for another;
- **Reciprocal** — units mutually depend on each other's outputs and must continuously coordinate their work.

From an organizational design perspective, **reciprocal dependencies create the highest coordination costs** and are the strongest argument for organizational proximity. Sequential dependencies can often be managed through interfaces and processes, while pooled dependencies are usually well suited for centralized provision.

After conducting a workshop with representatives of the relevant functions, we mapped the operational dependencies between the components identified in the previous step.



Analysis of operational dependencies and formation of clusters

Digital Core.

Web, Backend, iOS, and Android must be part of the cross-functional teams because these are the most high-frequency and mutually dependent functions of the digital product. Changes in Backend require synchronization with Web and mobile platforms, and interface updates affect server logic.

Integration Cluster.

ABS, DBMS, and CRM exhibit strong reciprocal dependencies not only among themselves, but also with the Backend. At the data and integration level, changes in business logic and data models require synchronized updates across ABS, databases, and backend services, while backend changes frequently trigger adjustments in integration components.

As a result, these components form a **chain of reciprocal dependencies** that spans the Digital Core and the integration layer. Although ABS, DBMS, and CRM fall into the *consider* zone in the frequency–specificity analysis, excluding them mechanically would externalize a set of tightly coupled interactions and significantly increase coordination and integration costs.

To address this, these components were grouped into a dedicated Integration Cluster, forming a second type of cross-functional teams.

Alternative grouping option.

During the analysis, we considered the possibility of combining the entire Digital Core (Backend, Web, iOS, Android) and the integration components (ABS, CRM, DBMS) into a single cluster. In theory, an ideal cross-functional team in such a case could implement changes fully end-to-end without dependencies. However, this option was rejected: the combined cluster would not be high-frequency; resources are limited, and we would not be able to staff all teams with a full set of competencies; and the team size would become too large and inefficient due to the narrow specialization of experts.

Therefore, we deliberately divided the structure into two clusters — Digital Core and Integration Cluster — and obtained two different types of cross-functional teams that together provide the full value-creation flow.

Marketing & Sales Cluster.

Marketing and Digital Sales demonstrate a mixed dependency pattern. Digital Sales has sequential dependencies with the Digital Core (especially Backend) and reciprocal dependencies with Marketing. This configuration explains why, despite moderate specificity, both functions are critical for completing most features end-to-end. Including them within the product group as a shared capability reduces coordination overhead and prevents recurring cross-unit synchronization issues.

Not included in the product group.

The Call center and Sales (branches) are not included in the product group because they have low frequency and low specificity, and their links to the product are sequential rather than reciprocal.

Which supporting functions should be included, considering the Cost of Delay (CoD) and criticality / uncertainty?

Returning to the concept of the product group and the multidimensional organization, input-functions — such as HR — may be present inside the product group, although they are not part of the essential value chain. Traditionally, such functions are centralized as corporate services. However, when over-centralized, they may become bottlenecks, creating a high **Cost of Delay**. In such cases, it may be economically beneficial for the Product Owner to "pay for" this function and include it inside the product group.

Any function outside the product group should be evaluated across two factors:

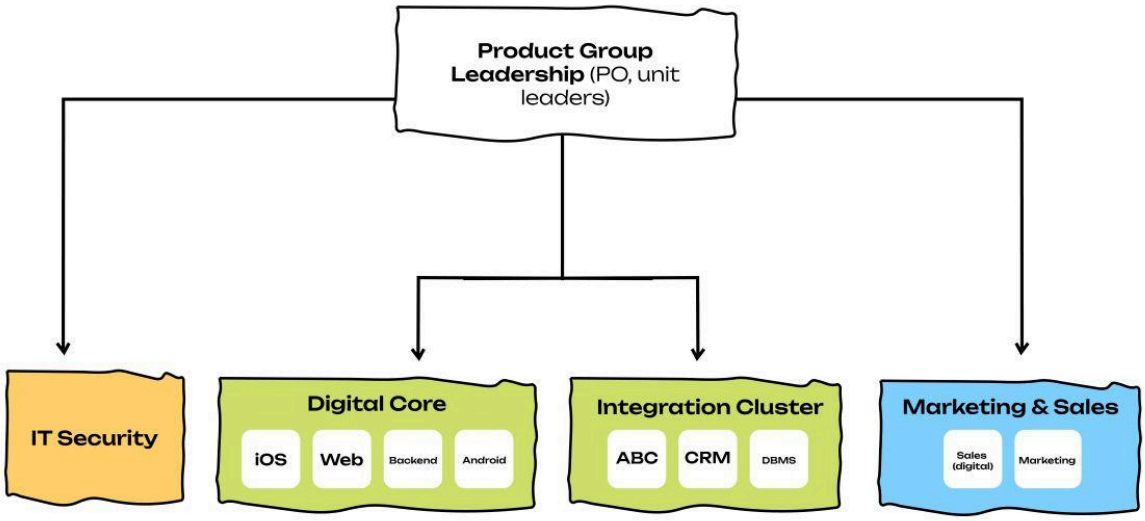
- **Criticality** — how strongly the function affects the business outcomes of the product group (for example, NPS or P&L). Scored from 1 to 10, where 10 is the most critical.
- **Uncertainty** — the degree to which centralized performance of this function interferes with predictable and reliable delivery. Scored from 1 to 10, where 10 is maximum unpredictability.

In *Creating Agile Organizations* we write that functions that receive high scores (above 5) across both factors should be considered for inclusion within the perimeter of the product group.

Thus, after the workshop, we evaluated the corporate functions that remain outside the perimeter of the product group and are not part of its essential parts:

- **IT Security** — (7, 7)
- **HR** — (2, 2)
- **Finance** — (4, 2)
- **Legal** — (2, 7)

The analysis across the two criteria — criticality and uncertainty — shows that the only potential candidate for inclusion in the product group is **IT Security**.



Product Group concept

At this stage of design, the concept of our product group looked as follows:

- **Output units:** the Digital Core and Integration Cluster;
- **Input unit:** Information Security (IT Security);
- **Market unit:** Marketing & Sales.

What model of line and product leadership?

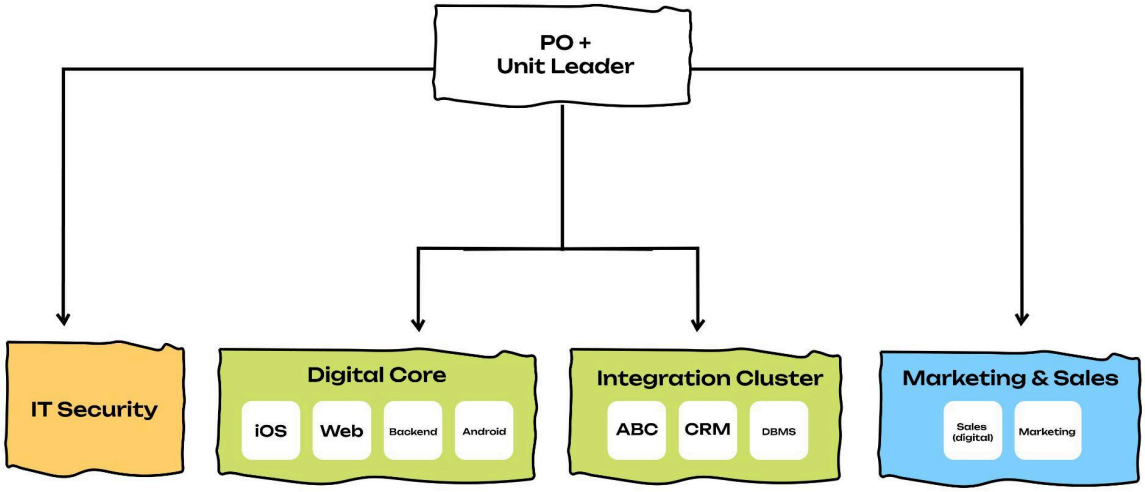
In *Creating Agile Organizations* we emphasized the need to separate product and line management. In large organizations, the Product Owner carries a significant product workload: strategy, vision, market interaction, defining value, and managing the Product Backlog. Simultaneously developing the organizational system — people, competencies, processes, performance evaluations, hiring, rewards — is practically impossible without losing quality. Therefore, in agile organizations, the product and line managerial circuits are separated. Moreover, conflict of interest may arise between line and product management.

We had three main variants of distributing leadership inside a product group.

1. Dual Leadership

The Product Owner is responsible for strategy and value, while the Unit Leader is responsible for people, processes, and the organizational environment. This is a simple, clear, and universal model, although as the product group grows, the workload on the Unit Leader begins to

increase. In this logic, the Unit Leader becomes a shared cross-functional manager for all units within the product group.

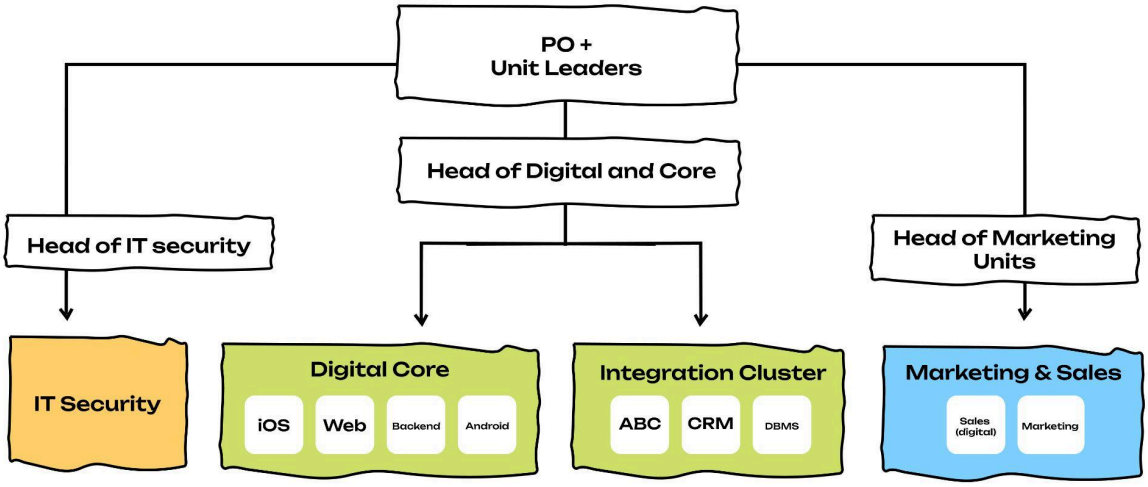


Dual Leadership model

2. Triad Leadership.

This model is suitable for heterogeneous product groups. Here, the Product Owner remains the strategic leader, while line management is distributed between:

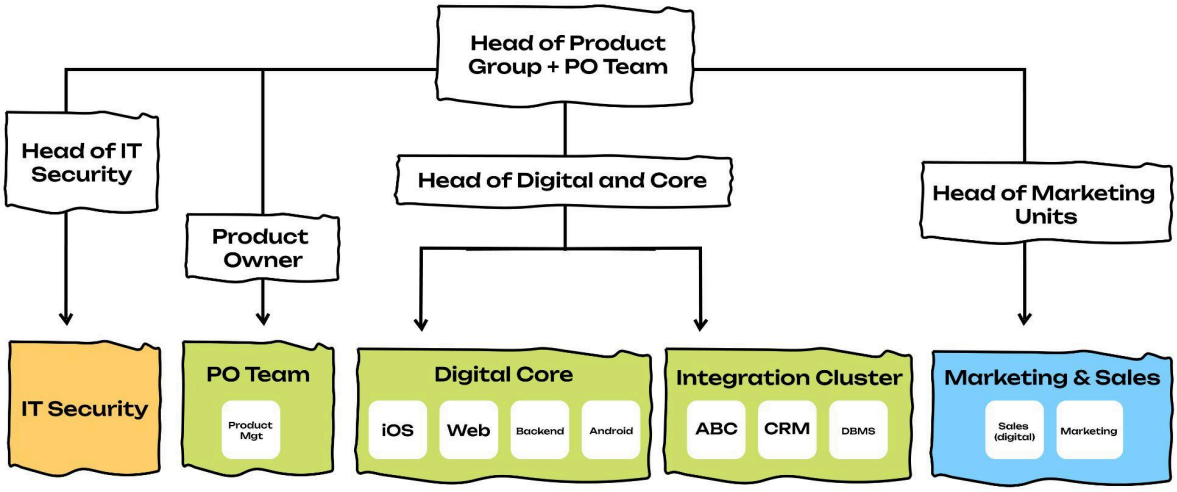
- **Output Lead** (responsible for cross-functional teams of both types);
- **Market Lead** (responsible for Marketing and Digital Sales);
- **Input Lead**



Triad Leadership model

3. Head of Product Group (GM) + PO Team

This model appears when the product group effectively becomes a mini-business. In this case, a **Head of Product Group (GM)** emerges, responsible for the organizational system, resources, budgets, and the business domain, while the Product Owner leads the product strategy supported by the Product Owner Team. The model is suitable for large product families — those that, by scale, autonomy, and operational complexity, function as a mini–business unit (potentially separatable organization). It requires a mature managerial architecture, multiple units of different types, significant budget, and strategic independence. For small and medium product groups, such a structure creates excessive organizational overhead.



Head of Product Group + PO Team model

Hybrid constructions that combine elements of all three approaches are also possible.

In our case, the **Triad Leadership** model was considered the most appropriate. We did not consider it realistic for a single line leader to effectively develop both technical teams (Web, Backend, Mobile, Integration) and marketing-commercial functions (Marketing, Digital Sales). These units require fundamentally different expertise, different approaches to competency development, different performance metrics, and different operating models. Dividing leadership into an Output Lead and a Market Lead enabled effective management of each direction and avoided overloading a single leader, which is critical for the sustainable development of the product group.

Which organizational design had minimal functional coupling?

In *Creating Agile Organizations* we formulate the guideline *Decouple Unit Functions*: when designing organizational units, it is important to ensure that they can achieve their goals as independently as possible, without negatively affecting other units.

To formally assess the degree of such functional independence, we used the approach of **axiomatic design**. It proposed describing the organization through two sets:

- **Functional Requirements (FR)** — what the unit had to ensure;
- **Design Parameters (DP)** — the structural elements that implemented these requirements.

Good design aims for **minimal functional coupling**, meaning each FR depends primarily on one DP. When a single DP begins to affect several functional requirements, undesirable interdependence and the risk of conflicts arise.

During the workshop, management formed an FR–DP table by assigning functional requirements to each unit within the product group. Initially, IT Security was considered a candidate for inclusion in the product group — based on the two criteria of criticality and uncertainty, both of which indeed turned out to be high.

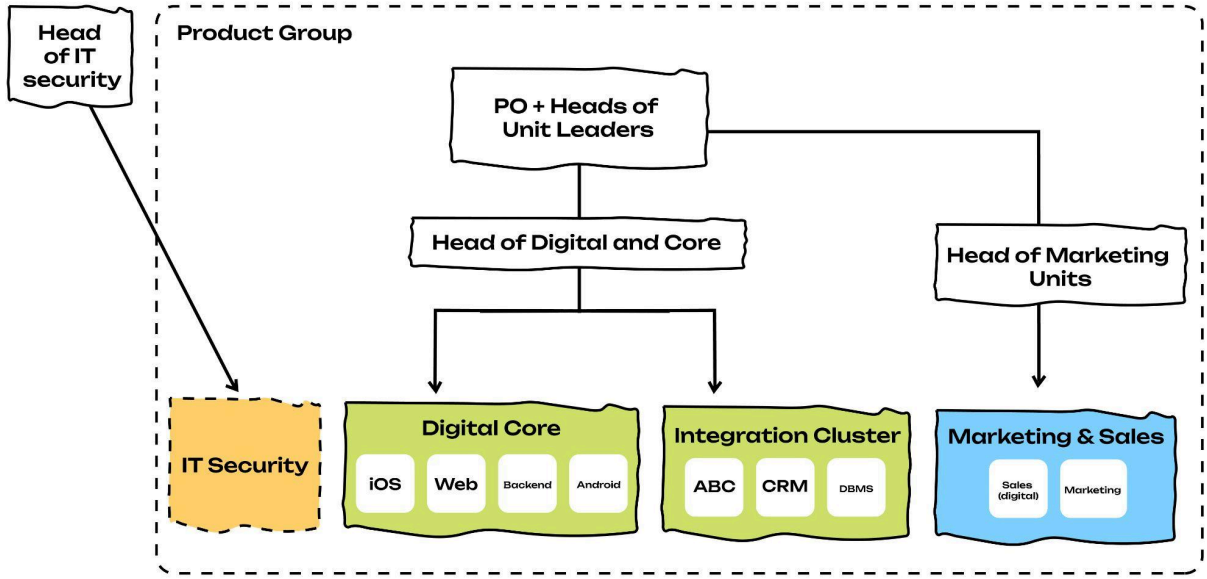
Functional analysis revealed an important issue: a structural conflict of interest arose between the Product Owner and IT Security. Product goals (value, speed, priorities) could exert pressure on the security control functions, which led to functional coupling and reduced the independence of IT Security.

	Output Lead (Digital Core)	Market Lead (Mkt & Sales)	PO / PO Team	IT Security
Ensure high-quality and reliable end-to-end delivery of features	R			
Customer acquisition and retention through marketing and digital sales activities		R		
Strategy and product family's business outcomes (P&L)			R	
Security, compliance, and risk management			r	R

Potential conflict of interest through functional coupling

Therefore, despite high scores for criticality and uncertainty, we made a decision: **IT Security remained outside the product group**, preserving independence and reducing functional coupling.

By the way, nothing prevented the product group from including a **security representative functionally**. Line-wise, this person would report to their manager in the centralized function, while receiving work from the Product Owner. In other words, the issue was resolved at the **process level**: we used **matrix coordination**, preserving the independence of the control function while ensuring its operational participation in the work of the product group.



Matrix coordination of IT Security

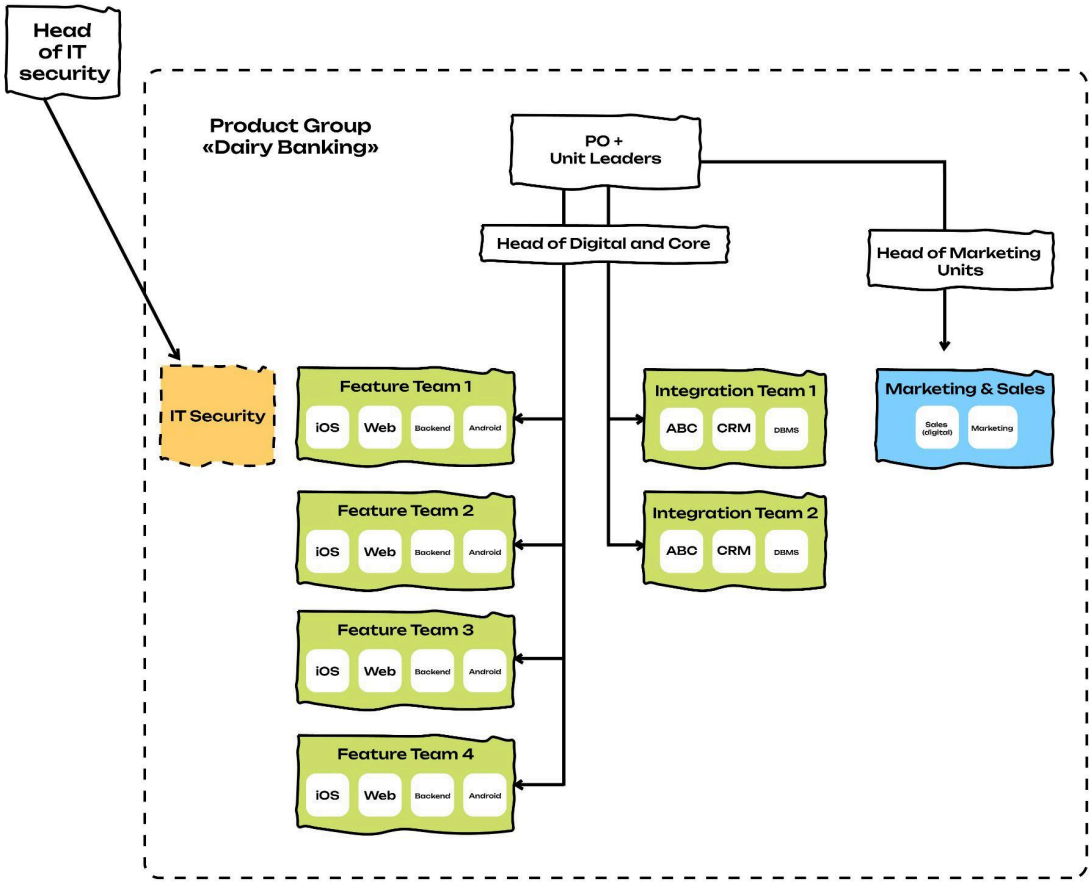
What are the Value Areas and what is the composition of teams?

In the practice of designing product groups, there is an approach in which teams are organized around **Value Areas** — large value streams or customer jobs for which a team is responsible end-to-end. However, in the product family **Daily Banking**, the entire development workload was distributed across only six teams.

When analyzing organizational capabilities, one of the key capabilities was **Adaptability**. In the terminology of Transaction Cost Economics (TCE), Adaptability is associated with **costs of adaptation** — the costs of switching and reconfiguration. These costs increase sharply with

specialization: the narrower the team's profile, the more expensive and slower it becomes to adapt to changes, new features, and workload redistribution.

To avoid this, we applied a different principle: **minimal specialization of teams while preserving end-to-end responsibility**.



Final structure of the Product Grou[

- **4 interchangeable Digital Core teams**, which were able to take any feature of the product family into development (transactions, cards, bonuses, daily customer scenarios);
- **2 interchangeable Integration Cluster teams**, which ensured work with ABS, CRM, DBMS, and other integration components.

This design reduced the cost of adaptation, increased agility, removed queues between domains, and increased the throughput of the entire product group — without the need to create separate Value Areas.

Summary

To briefly summarize the algorithm and the result using the example of the product group **Daily Banking**:

- We began with the mission and essential parts, defining the function the product group played in the large organization and which elements were necessary to fulfill that function.
- Then, relying on organizational capabilities, frequency and specificity of functions, and the types of operational dependencies, we determined which functions were to be included inside the product group and which were to remain corporate.
- Through the assessment of criticality and uncertainty, we refined the composition of supporting functions and the boundaries of the unit.
- Using the principle of minimal functional coupling, we assessed the independence of key functions and adjusted the placement of control roles (such as IT Security).
- Finally, we made decisions about the leadership model (**Triad Leadership**) and the team composition (**4 Digital Core + 2 Integration Cluster teams**), which supported the selected capabilities — **Adaptability** and **Rapid launch of new products** — while keeping transaction costs at a reasonable level.

This is how a product group structure was formed in which strategic focus, architecture, teams, and corporate functions were connected through a coherent and unified design.

References

Pavlichenko, I., & Cesario, S. (2023). *Creating Agile Organizations*.

Ackoff, R. L. (1971). *Towards a system of systems concepts*. *Management Science, 17*(11), 661–671.

Treacy, M., & Wiersema, F. (1995). *The Discipline of Market Leaders: Choose Your Customers, Narrow Your Focus, Dominate Your Market*. Addison-Wesley.
Ackoff, R. L. (1999). *Re-Creating the Corporation: A Design of Organizations for the 21st Century*. Oxford University Press.
Coase, R. (1937). *The nature of the firm*. *Economica, 4*(16), 386–405.
Coplien, J. O., & Bjørnvig, A. (2018). *A Scrum Book: The Spirit of the Game*. The Pragmatic Bookshelf.
Galbraith, J. R. (2014). *Designing Organizations: Strategy, Structure, and Process at the Business Unit and Enterprise Levels*. Jossey-Bass.
Kates, A., Kesler, G., & DiMartino, M. (2023). *Networked, Scaled, and Agile: A Design Strategy for Complex Organizations*. Kogan Page.
Liker, J. K. (2004). *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer*. McGraw-Hill.
Suh, N. P. (2001). *Axiomatic Design: Advances and Applications*. Oxford University Press.
Williamson, O. E. (1975). *Markets and Hierarchies*. Free Press.
Williamson, O. E. (1985). *The Economic Institutions of Capitalism*. Free Press.
Worren, N. (2012). *Organizational Design: A Step-by-Step Approach*. Pearson.