We executed the T3 tests in three stages. In each stage, we conducted semi-structured interviews with open and closed-ended questions, as the previous chapters demonstrate. During the analysis, we validated or refuted hypotheses and identified proposals for refinement and/or improvement (especially at the start of the interview, where we discussed the theory broadly). This approach allows us to initiate the discussion broadly before refining and focusing on the propositions.

We divided the interview rounds into three groups:

First, I sought individuals with diverse perspectives who played significant roles in their respective DevOps contexts. Next, I felt the need to involve someone from academia. Leonardo Leite currently serves as a reference on the subject, having published X articles in the field that discuss these specific team structures. Finally, we validated the model within a public enterprise characterized by a rigid structure and strict regulations.

Round 1, Specialists 1, 2, and 3

S1 - Manoel Paes

In the first interview round, I spoke with Manoel Paes, Nitai, and Alexandre. I conducted the interview with Manoel Paes in Spanish at a café in Madrid and transcribed it in the same language. I held the other two conversations via video call, recording and transcribing them in their respective languages.

I interviewed Manoel Paes first because we used his book to develop the theory. This step allowed us to validate the theory—specifically, "the extent to which the theory, once it is instantiated, represents the theories that were the input for this study". Subsequently, I interviewed the other two experts to verify the theory in a practical context.

According to our selection criteria:

C1: Deep Domain Expertise and Experience (Seniority)

With a career spanning approximately 25 years in the IT industry, Pais possesses the extensive professional background necessary to mitigate errors caused by inadequate knowledge. His seniority is evidenced by his Master of Science in Software Engineering from Carnegie Mellon University and his long-term role as Lead Editor for DevOps at InfoQ. In the interview, his

capacity for judgment is demonstrated by his ability to immediately identify systemic patterns in complex organizational structures.

C2: Integrative/Theoretical Knowledge:
As the co-author of the seminal work Team Topologies (2019), Pais is a prominent Academic-Industrial Specialist. His career is defined by the synthesis of abstract principles, such as Cognitive Load and Conway's Law, with concrete industrial applications. During the validation process, he exhibited "actively open-minded thinking" by challenging the static nature of the proposed UML model and suggesting a shift toward dynamic interaction modes (collaboration, facilitation, and X-as-a-service), which directly addresses the reduction of Pattern Noise.

C3: Diversity of Organizational Context:
Pais provides an International Consulting and Global Tech perspective. His experience encompasses advising diverse organizations worldwide on how to evolve their team structures. This diversity of context was evident in his feedback, where he contrasted the needs of large-scale enterprises with those of startups and government entities, ensuring the proposed theory's generality across different sectors.

The interview with Manoel Paes highlighted several key points:

1 - Model Dynamics and Representation

Paes suggests that we describe how these interactions occur, although this research primarily focuses on describing the forms and how this theory evolves over time. He recommends that the theory should not merely describe "who is who" (the structure), but rather how the entities interact and how these interactions evolve over time (e.g., a team that starts in Collaboration mode and evolves into X-as-a-Service).

Given its descriptive-analytical nature, this theory describes a state—specifically, the team structure at a particular moment. Ensuring the theory is instantiable allows practitioners to identify the structure that fits their context or determine which variables they need to alter to reach a different state (team structure).

2 - Team Cognitive Load as a Team Variable

We must measure the validity of a DevOps structure (such as the presence of a platform team) by the extent to which it reduces the mental effort of product teams.

If a structure or platform increases cognitive load rather than reducing it, it becomes an anti-pattern, regardless of how well it is designed on paper.

> *"Para nosotros, el propósito de una plataforma es reducir la carga cognitiva de los equipos de producto. Si no reducimos la carga cognitiva, entonces no estamos haciendo bien nuestro trabajo en la plataforma. Puede que estemos estandarizando, puede que estemos ahorrando costes, pero si los equipos de producto todavía tienen que saber demasiado sobre cómo funciona la*

infraestructura o cómo se despliega, entonces la carga cognitiva sigue siendo muy alta."

3 - Autonomy (The Developer Journey)

Autonomy does not simply mean "having permission to do things"; it means having the tools and support to ensure that bureaucracy or a lack of technical knowledge does not interrupt the workflow.

> *"La autonomía es clave para que los equipos puedan ir rápido. Pero no es una autonomía donde cada uno hace lo que quiere. Usamos a menudo el libro de Daniel Pink, Drive, que habla de Autonomía, Maestría y Propósito. Si los ingenieros sienten que tienen autonomía para tomar decisiones técnicas sobre su servicio y tienen la maestría (las habilidades) para hacerlo, entonces están mucho más motivados. La plataforma y los equipos enablers están ahí para asegurar que el desarrollador no se quede bloqueado, que su jornada sea fluida."*

Paes emphasizes that "mastery" must exist for autonomy to function. This insight validates the need for Enabling Teams, which helps close the knowledge gap.

S2 - Nitai

I selected Nitai Bezerra for his extensive practical experience in bridge-building across data engineering, software development, and infrastructure, particularly within the Brazilian public administration. He plays a crucial role in this research because he represents the profile of a specialist who operationalizes large-scale data and facilitates the transition of organizational cultures in complex environments.

According to our selection criteria:

C1: Deep Domain Expertise and Experience (Seniority): Nitai has a professional trajectory that includes roles as a Software Engineer, Data Engineer, and Tech Lead. His experience spans over a decade, with significant contributions to the Ministry of Culture (2015-2018) and the Ministry of Economy, where he led the creation of data engineering teams. He currently operates in the private sector as a senior leader in data infrastructure, managing complex environments involving Kubernetes, Airflow, and automated CI/CD pipelines.

C2: Integrative/Theoretical Knowledge: Nitai demonstrates a strong capacity for "integrative knowledge" by connecting DevOps principles with the specific challenges of Data Engineering (DataOps). He is a specialist in Open Data and Free Software, contributing to government integration platforms such as the "Prisma de dados" and the "Programa de Gestão" (PGD) API. In the interview, he provided a detailed narrative of how he transitions from theoretical

automation to the actual transfer of responsibility between teams, showcasing an "actively open-minded" approach to organizational change.

C3: Diversity of Organizational Context: He provides a unique perspective from the Brazilian Public Sector and Private Enterprise. Having worked in strategic government entities, he understands the "Blame Culture" and the rigid separation of duties often found in bureaucracy, contrasting this with the agility required in the private sector. This dual background is essential to challenge the theory's propositions regarding silos and shared responsibility across different institutional cultures.

> "A gente trabalhou muito forte na API do Programa de Gestão... o objetivo era que qualquer órgão pudesse bater lá e pegar os dados, sem precisar falar com a gente no dia a dia. Isso gerou uma autonomia que antes não existia."

Collaborations:

1 - Achieving Autonomy Through Automation Automation acts not merely as a technical improvement but as a driver of cultural change. He highlights how using tools like Airflow and CI/CD pipelines enabled teams to gain the autonomy to deploy code to production without relying on constant manual intervention. He also asserts that automation "forces" knowledge diffusion; “quando um membro da equipe cria um hook de automação (como o exemplo do lint no Git), todos passam a utilizá-lo e aprendem sobre o processo por necessidade.”

Automation serves not only to "gain speed" but to educate the team and ensure technical knowledge spreads effectively, eliminating the need for a human specialist to validate every step.

In Nitai's context, the data team dealt with resistance or a lack of standardization. He mentioned that by automating "lint" (code standardization) and "deploy" (code installation) processes, the team stopped debating "who was right" and started following what the tool dictated. This generated an educational autonomy: the developer learns to do the right thing because automation prevents them from following the wrong path.

> "A gente começou a colocar hooks no Git... Então, o cara ia fazer o commit e o hook barrava porque não tinha o padrão. No começo, o pessoal reclamava, mas depois eles entendiam o porquê daquilo estar ali. A automação acabou difundindo o conhecimento na equipe por uma força da necessidade. Não precisava mais de alguém ficar olhando o código de todo mundo o tempo todo; a ferramenta já dava o feedback imediato para o desenvolvedor."

Automation reduces interpersonal dependency: Knowledge leaves the "specialist's" head and moves into the platform code.

Real-Time Feedback: Real-time feedback strengthens autonomy because the developer receives immediate correction (via the tool), without the "noise" or delay of a manual human review.

Behavioral Change: The infrastructure "encodes" the DevOps culture, making adherence to best practices natural and mandatory for the workflow.

For him, the ideal involves not just daily frequency, but also planned, intentional, recurrent interaction (such as Dojo sessions or knowledge sharing) to prevent the team from depending on sporadic interactions to "put out fires."

S3 - Alexandre Ferreira

Alexandre Freire brings the perspective of "Extreme Scale" and mature engineering management. As a Director of Engineering at Google, he validates the framework against the rigorous demands of global systems serving billions of users.

C1: Deep Domain Expertise and Experience (Seniority): Freire possesses over 25 years of experience in software engineering and leadership, currently serving as the Director of Engineering at Google and Site Lead for the São Paulo Engineering Center. He oversees the "Counter Abuse Technology" platform, managing a highly specialized organization of approximately 150 engineers (direct and indirect reports) responsible for ensuring safety across products like Gmail and Maps. His seniority is further evidenced by previous executive roles at Nubank and Locaweb, where he led hyper-growth engineering teams.

C2: Integrative/Theoretical Knowledge: Holding a Master's degree in Computer Science from the University of São Paulo (USP), Freire combines academic rigor with pragmatic application. He is a frequent speaker at major industry conferences (e.g., The Developer's Conference, QCon) on topics such as Modern Agile and Engineering Culture. In the validation interview, he demonstrated deep integrative knowledge by actively applying concepts from *Team Topologies*(such as "Platform Teams" and "Cognitive Load") to discuss real-world trade-offs in Google's infrastructure, moving beyond theory to discuss the implementation of SLOs (Service Level Objectives) and Error Budgets as management tools.

C3: Diversity of Organizational Context: Freire represents the "Big Tech" and Global Scale context. His input challenges the theory to account for environments where "daily deploys" and "gradual rollouts" are standard practice. This contrasts sharply with the public sector and consultancy perspectives of other experts, ensuring the framework is robust enough to handle scenarios of extreme complexity and distributed systems where manual intervention is impossible due to the sheer volume of data.

Extreme Scale: Alexandre validates that, at a certain scale, total automation acts not as an option but as the sole means of survival.

The "Ivory Tower": He warns against the risk of the platform team isolating itself (the "Platform Silo"), suggesting that the theory must include internal Customer Success mechanisms to prevent the platform from becoming useless.

Hybrid Roles: He notes that finding the "unicorn" profile (someone who knows everything about product and engineering) remains rare, which validates the need for the ontology to separate responsibilities so that "normal" people can deliver extraordinary value.

Contributions: He validates that, in extreme scale environments, team structure concerns not just organization, but survival and efficiency through abstraction.

1 - The Platform as a Product Alexandre strongly validates the role of Platform Teams in the ontology but adds a critical success condition: the team must manage the platform as a product with internal customers.

A platform team creates a useless silo (an "Ivory Tower") if it builds tools that no one uses. For this theory, this means that the team's existence alone does not suffice; the team requires Customer Success mechanisms and adoption metrics. Adopting a platform creates a dependency, but he views this as an acceptable compromise in exchange for gains in scale and speed. (This connects to the proposition regarding the platform team sharing responsibility, P11, which rounds 1 and 3 refuted.)

> “eles cuidam da plataforma e se ninguém usar plataforma eles entregaram zero valor, né podem estar ali numa Torre de Marfim pirando em utilizar um negócio que tá desconectado com as necessidades dos clientes e os clientes não usam, se otimizou algo que ninguém tá usando e né você não entrega o valor. Acho que tem esse risco sim. E por isso que é importante ter aí uma proximidade e esses stakeholders management e essa visão de produto mesmo para garantir que você está investindo em algo que vai ser valioso para a maioria dos seus clientes.” P2

2 - Cross-functionality He confirms that the team, and especially its leadership, must be multidisciplinary. He describes a "leadership quartet" (Engineering Manager, Product Manager, Technical Lead, and Program Manager) that, together, covers all necessary skills. This suggests that multidisciplinarity exists as a property of the team, not necessarily of the individual. This insight corroborates our cross-functionality variable for describing the team.

3. Automation as the Sole Path to Scale Alexandre positions automation as a prerequisite for the scale at which he operates. He states that complete lifecycle automation (deploy, testing, gradual rollouts) allows a small team to manage global services. He emphasizes that automation serves not only repetitive tasks but also complex processes such as data structure migrations.

> "É muito importante você ter uma equipe multidisciplinar porque senão você tem […] só um bando de pessoas que sabem fazer a mesma coisa então na hora que eu estou falando com cliente, eu gosto de ter ali um quarteto de liderança que representam algumas das habilidades. Então tem um enginner manager que alguém entende a visão a necessidade do negócio [...] tem um product manager que tá pensando do ponto de vista do produto [...] tem um técnico Lider, um Staff, um Senior Staff Engineer que tá pensando nas questões técnicas [...] e um

> programmer manager que alguém tá pensando de como eu amarro todas as pontas."

The team structure must compensate for individual limitations, ensuring that the group, as a whole, possesses all necessary competencies through well-defined roles (such as the quartet he mentions).

4. "Useful" Silos and Abstraction Alexandre offers a more nuanced view of silos. He does not view the separation of Dev and Ops/Platform as a problem, provided the interface acts as a clear abstraction. The developer must understand "what" they are configuring (e.g., database redundancy), but they do not need to know "how" the infrastructure resolves it. Therefore, the platform must hide complexity to reduce cognitive load, keeping the developer focused on the business. (Check the propositions related to this; it corroborates the creation of proposition P29 due to cognitive load. However, I can also add a section here.)

Alexandre Freire contributes by validating that specialization (and certain silos) remains inevitable and beneficial at scale, provided that clear contracts (SLOs) manage it and that the platform serves the product team to reduce complexity rather than creating bureaucracy.

> “Você tá perdendo uma oportunidade de entregar mais valor, então acho que na sua essência essa colaboração é que permite com que a gente tem os ganhos de escala de criar esse silos de maneira deliberada, quando a gente cria esses times de plataforma, a gente tá criando uma dependência, o essencial seria você poder ter um time ali, mas isso não funciona em escala. [...] O ruim de você criar uma dependência é se essas pessoas não se falarem. E aí você mitiga esse risco criando laços de colaboração aí entre elas, garantindo que existe a confiança e muito alinhamento aí para você não pegar só o ônus de ter introduzido essa dependência.” P2

However, Alexandre does not view the silo solely as a villain; in large corporations, silos represent necessary abstractions that allow the team to maintain focus.

> “...então acho que você mantém aí o silo nesse sentido, mas para eu fazer um bom trabalho como desenvolvedor, se eu não entendo o que é aquela configuração significa é meio que tentativa e erro ali, né? [...] Acho que a barra mínima você conseguiu entender e conseguir escrever pelo menos em alto nível […] é algo que não tem como você cobrar de uma pessoa especialista em todas essas partes de uma estaca tão complexa, né?”

THIS REFUTES PROPOSITION P11

Round 1 output

In this initial validation cycle, we addressed 32 out of 84 hypotheses (38%). Among these, 25 were validated (representing 29% of the total) while 7 were refuted. This round drove the evolution of the model from a static taxonomy into a dynamic framework managed by Cognitive Load, generating critical insights to guide subsequent validation rounds. Based on the empirical evidence gathered from the first round of expert interviews, the initial ontology underwent significant structural and conceptual refinements. These changes aim to move the framework from a static taxonomy of roles to a dynamic model centered on cognitive load management and high-quality interactions. In total, 3 attributes were added, 2 new propositions were formulated, and 3 definitions were adjusted.

The most critical insight from Manuel Pais was the identification of Cognitive Load as the primary variable that justifies the existence of platform structures. Then, we added A new attribute, Cognitive_Load, to the Team class to measure the mental effort required for delivery. We also added a New Proposition (P29): *"Platform Servicing reduces team cognitive load."* aiming to explicitly link the consumption of platform services to the reduction of cognitive strain on product teams, shifting the platform's value proposition from mere "tooling" to "enabling focus." Experts argued that "Stream-Aligned Teams" cannot survive with full end-to-end ownership unless complexity is abstracted. As stated by Paes, *"The platform must reduce the cognitive load of the product teams consuming it."*

We also refined the concept of collaboration to prevent the anti-pattern of "endless meetings" or dependency disguised as teamwork. To accomplish this, we added three new attributes to the Collaboration class: Motivation (the explicit purpose of the interaction “Why”), Duration (The expected timeframe “When”), and Definition of Done (the success criteria “What defines the end”). We also added a new Proposition (P30): *"A high-quality collaboration reduces dependency, increasing team autonomy."* Collaboration is now modeled as a temporary, high-cost investment to solve specific problems (e.g., automating a test pipeline), not a permanent state. Paes emphasized that interactions should have a clear "arrival point" (e.g., define an API) to avoid creating new bottlenecks.

We expanded the role of the Horizontal/Platform Team to include product management capabilities, countering the "Ivory Tower" risk identified by Expert E2. We added the method customer_focus() to Horizontal Team. Crucially, for these teams, the customer is defined as Internal (other development teams), whereas for Product Teams, it remains External. In the theory model representation, we stabilished a new connection from Product Management to Horizontal Team. Platform teams often lack product ownership, leading to tools that no one uses. E2 stated, *"If no one uses the platform, they deliver zero value,"* necessitating a product manager role within the platform team to drive adoption and alignment with internal customer needs.

Finally, we expanded the definition of Automation beyond technical CI/CD tasks to explicitly include governance, security, privacy, and regulatory compliance. Automation is re-contextualized not merely as a mechanism for streamlining technical tasks, but as a strategic

instrument for reducing cognitive load. Expert E1 (Manuel Pais) argued that traditional DevOps definitions often overlook the 'business of the internal platform,' which includes complex domains such as legal compliance, financial controls, and security.

In the previous model, these processes were treated as external dependencies managed by specialized silos (e.g., a Legal Department or Security Team), which frequently acted as bottlenecks that interrupted the value stream. By transforming these manual checkpoints into automated platform services (e.g., automated compliance checks in the pipeline), the platform abstracts this complexity. This allows stream-aligned teams to adhere to governance standards without needing deep domain expertise in regulation, thereby removing blocking dependencies and redirecting focus toward the delivery of customer value.

“Automation is an essential DevOps activity. Beyond its role in streamlining technical tasks, automation is instrumental in reducing the cognitive load on teams. This principle enables extending automation to complex domains such as governance, security, privacy, and regulatory compliance. These processes, traditionally manual and reliant on a specialized team, often introduce significant dependencies and bottlenecks. By transforming these activities into automated platform services, teams can redirect their focus toward their core objective: the delivery of customer value…”

Round 2, Specialist 4

Following the initial focus on human dynamics, Round 2 aims to bring theoretical rigor and precise definitions to the theory. We selected Leonardo Leite to introduce a layer of theoretical rigor and academic-industrial validation. Leite brings the perspective of a researcher-practitioner who has systematically mapped the DevOps domain. Beyond the validated hypotheses, Leite contributed by merging academic rigor (precise definitions) with his practitioner view. In this interview, Leonardo focused on ontological precision and the dynamic nature of roles.

1 - The "Enabler" as a Transitory Role, Not Just a Fixed Team

Leite challenged the idea that the Enabler Team must exist as a fixed "box" in the organizational chart. He suggested that "Enabling" often refers to a state or role that specialists assume temporarily, rather than to a team. However, when the need for this team arises, we can instantiate or map it within the harmonized model.

> "...a dúvida que eu tenho é se o Enabler precisa ser um time. Porque, às vezes, o Enabler é um papel que alguém assume. Por exemplo, eu sou um SRE sênior, ou um Arquiteto, e eu vou atuar como Enabler naquele time por um tempo. Eu vou lá, passo o conhecimento, ajudo a configurar a ferramenta, destravo o time e saio. Se a gente define como um 'Time Enabler', parece que aquelas pessoas só fazem isso

> o tempo todo, e talvez não precise. Pode ser um papel rotativo ou uma missão temporária."

Leonardo's findings are pertinent and may impact adjustments in the explanations, since they do not alter or impact other artifacts. The team does not possess a variable to represent its size. E1 Team and specializations.

2 - The Genesis of Automation: "Pain" vs. "Hype" He validated the hypothesis that automation stems from experience (from the "pain" of doing things manually), and corroborated P20, but he added a market variable: Hype. Ideally, automation emerges organically from problem repetition (e.g., creating Ansible scripts out of necessity). However, he warned that, in the industry, Hype often introduces automation (e.g., "let's use Kubernetes because it's trendy"), which can generate accidental complexity without resolving a real problem.

> “[...] ele nasce em algum momento disso [da experiência/dor]. Embora, talvez, uma vez que alguém criou... sei lá, o Ansible ou alguma coisa, aí, em outro contexto, às vezes, alguém pode querer trazer essa automação por hype. Mas, mesmo que, ah, eu tô querendo trazer por hype, porque eu vi falar que o Kubernetes é hype e eu quero trazer, mas, o Kubernetes nasceu, é, porque, alguém, passou por isso antes.”

Impact on Theory: This refines the proposition regarding automation, suggesting that it only reduces cognitive load if it stems from a real need (experience), and not from the imposition of tools.

Nitai already mentioned this; automation emerges from experience. People perform tasks, and many do so repeatedly without realizing the repetition. Thus, the opportunity for automation arises from experience—seeing something elsewhere and applying it here, or recalling a similar action in another scenario. Or even the sensitivity to realize that a task remains too manual because it consumes too much time. I realize I am repeating code, repeating commands—how do I automate this? I believe all of this constitutes experience. Both life experience and diversity of disciplines.

Round 2 output Finally, the second round expanded the empirical coverage of the theory. Following the refinements from Round 1 expanding to 88 hypotheses (incorporating the 4 new hypotheses derived from the new propositions P29 and P30). In this round, 17 hypotheses were validated, with 16 of them being unique validations (not covered in Round 1). This progress elevated the total empirical coverage of the theory to 46%, providing a solid foundation for the core constructs of the ontology.

Leonardo introduced nuances regarding how structure impacts human behavior. He validated that silos create a blame culture and reinforced Proposition P1 (Shared Responsibility), arguing that the only way to eliminate blame involves breaking the wall between Dev and Ops through cross-functional product teams, where the team abstracts the infrastructure but shares the responsibility.

> "Se a gente tem aquele modelo clássico, onde o Dev é medido por feature entregue e o Ops é medido por uptime(estabilidade), você cria naturalmente o conflito. O Dev joga o pacote por cima do muro. Se quebrar, a culpa é do Ops que não soube rodar. [...] Isso gera a cultura do blame. O cara fala: 'na minha máquina funciona'. E ele está certo, na máquina dele funciona. O problema é que o incentivo dele acabou ali. Ele não é responsável pelo ciclo de vida." S4

The refutation and subsequent refinement of hypotheses regarding collaboration frequency (H2.2) and quality (H2.4). The experts refuted the original premise that "eventual" or "low-quality" collaboration could contribute to reducing silos, even if minimally.

The expert consensus revealed that sporadic interaction is not a "partial solution" to silos, but rather a symptom of their existence. A team that only collaborates "eventually" effectively operates within a silo. Consequently, the relationship between these variables was inverted to prepare for validation in the next round. Sporadic interaction maintains the "throw it over the wall" culture. To break a silo, collaboration must be frequent or high-quality.

We rewritten the hypotheses to assert that low-frequency interaction is positively associated with the presence of silos, rather than the reduction of them.

H2.2 Teams with eventual collaboration are associated with **fewer** organizational silos -> H2.2 Teams with **eventual collaboration** are associated with **organizational silos**

H2.4 Teams with low-quality collaboration are associated with **fewer** organizational silos -> H2.4 Teams with **low-quality collaboration** are associated with **organizational silos**

Round 3, Specialist 5

In Round 3, I interviewed Igor Simões, who represents the Large-Scale Financial/Traditional Sector context (Banco do Brasil), where governance and separation of duties undergo testing to the limit.

We selected Igor Regis da Silva Simões to represent the Large-Scale Financial Sector, a context defined by the duality of maintaining critical legacy infrastructure while accelerating digital transformation under strict regulatory compliance. As a senior technology leader at Banco do Brasil (one of the largest financial institutions in Latin America), his participation validates the theory in an environment where "Team Autonomy" and "Cognitive Load" must be balanced against systemic risk and financial governance.

While the previous rounds focused on taxonomy (Leite) and survival (Freire), Igor refined the theory by delimiting the boundaries of responsibility in large corporations.

1. Delimitation of "Ownership" Igor challenged the romanticized view of "shared ownership." The Platform Team does NOT own the Business Product. He emphatically stated that the platform "does not feel like the owner" of the products (e.g., Direct Consumer Credit/CDC, Credit) that run on it. The Platform owns Service Reliability (SLA), while the Product Team owns Business Risk.

He introduced a nuance regarding Blame Culture in the context of Platforms. In a mature model, "blame" shifts from personal ("so-and-so made a mistake") to contractual ("you didn't contract redundancy"), as seen in AWS contracts.

If the datacenter goes down and the product goes offline, the infrastructure is not necessarily to blame if the Product Team chose not to "buy" the high-availability service (investment) offered by the platform.

Igor also strongly validated that "Breaking Silos" (refined H2.2) happens not just through a "mindset shift," but through Organizational Design. He stated that collaboration only truly increased when they placed Business, Dev, and Infra on the same team (Product Team).

"Saying the goals are the same" does not work if the teams remain separated. The physical/logical structure of the team enables a collaboration culture.

The interviewee explicitly connects the lack of collaboration with the formation of silos and describes interaction in silos as "perfunctory" (*protocolar*) and "poor."

> "Entrevistador: ...as estruturas de silos organizacionais, elas podem ser caracterizadas com uma frequência eventual... / Entrevistado: Sim. Ou a colaboração nesse caso, no caso de silos, ela ocorre de maneira protocolar e com processos muito bem definidos. Se existe uma colaboração protocolar e com processos muito bem definidos, ela é ritualística e pobre."

In this final round, it became clear that the Platform does not share responsibility for the final product (business); it bears responsibility for the service that supports the product. Attempting to force a universal "Shared Ownership" between Platform and Product creates role confusion.

Therefore, we opted to remove Proposition P11 ("Responsibility/Ownership Sharing is a property... of Platform Teams"). The analysis of specialists E2 (Pais) and E5 (Igor) revealed that this proposition contained a conceptual error: it suggested that the existence of a platform led to sharing product ownership, when, in fact, the platform's function is to enable the product team's autonomy through abstraction (service), while maintaining distinct scopes of responsibility.

Specialist E5 refuted the idea that the platform team "shares" a sense of ownership over the business product (e.g., Loans/CDC). He validated that a clear boundary exists:

> "A plataforma tem um produto que é a plataforma. E realmente a plataforma não vai ter o senso de propriedade para os produtos do banco... O senso compartilhado

> por um produto igual o CDC ele ocorre para os atores que estão ali dentro da linha de produto do CDC. A plataforma realmente não entra nisso." (Igor, E5)

Manoel Paes clarified that the relationship does not involve "splitting responsibility" (which could recreate task-transfer silos), but rather removing cognitive load so that the product team can exercise its ownership fully.

> "En teoría, estos equipos [Stream-aligned] pueden hacer todo por sí mismos, pero esto no es factible desde el punto de vista de carga cognitiva... necesitan que los equipos de plataforma y enabling les ayuden a reducir la carga cognitiva, para que puedan abstraerse." (Manoel Pais, E2)

Therefore, we considered Proposition P11 imprecise and conceptually replaced it with Propositions P29 and P30, which describe the correct relationship: the Platform offers services to reduce Cognitive Load (P29), allowing the Product Team to possess Autonomy (P30), without a confusing merger of responsibilities over the final product.

General Conclusion of the Testing Phase

Over three rounds with specialists from different contexts (Global Industry, Academia, and the Financial Sector), we refined the theory into 82 hypotheses. We explicitly validated 54 of these, totaling 72% of the discussed theoretical hypotheses. This process resulted in changes to the theory, including the addition of two new propositions (P29 and P30) focused on reducing cognitive load and autonomy, and the removal of proposition P11.

Qualitatively, the theory's evolution adds Cognitive Load, a focus on the internal customer for enabler (platform) teams. Furthermore, we confirmed that Automation transcends technical efficiency, acting as an educational mechanism that imposes cultural standards and enables the "breaking of silos" by abstracting complexity rather than solely through a change in mindset.