# **Groundedness of codes (number of quotations coded by a code)**

|  | **Code** | **Grounded** | **Code Groups** |
| --- | --- | --- | --- |
| ● | collaboration | 54 | CULTURE |
| ● | responsibility/ownership sharing | 49 | SHARING |
| ● | automated application life-cycle management | 42 | AUTOMATION |
| ● | cross-functionality/skills | 36 | SKILLS & ROLES |
| ● | organizational silos/conflicts | 32 | ORGANIZATIONAL STRUCTURE |
| ● | skills/knowledge sharing | 28 | SHARING |
| ● | enabler (platform) team | 28 | ORGANIZATIONAL STRUCTURE |
| ● | devops (bridge) team | 19 | ORGANIZATIONAL STRUCTURE |
| ● | platform servicing | 19 | AUTOMATION |
| ● | team self-organization & autonomy | 15 | MANAGEMENT |
| ● | communication | 15 | CULTURE |
| ● | role definition/attributions | 14 | SKILLS & ROLES |
| ● | alignment of dev & ops goals | 11 | SHARING |
| ● | metrics, visibility & feedback | 11 | MANAGEMENT |
| ● | transfer of work between teams | 10 | MANAGEMENT |
| ● | automated infrastructure management | 8 | AUTOMATION |
| ● | culture, values & best practices | 7 | CULTURE |
| ● | cultural silos/conflicts | 6 | CULTURE |
| ● | training, evangelization and mentoring | 6 | SKILLS & ROLES |
| ● | change management | 5 | MANAGEMENT |
| ● | blame | 4 | CULTURE |
| ● | platform builder | 4 | AUTOMATION |
| ● | stack & tools sharing | 4 | SHARING |
| ● | product management | 3 | MANAGEMENT |
| ● | continuous improvement | 3 | CULTURE |

# **Groundedness of semantic domains (number of quotations coded by a semantic domain)**

|  | **Code** | **Grounded** |
| --- | --- | --- |
| ● | CULTURE | 89 |
| ● | SHARING | 88 |
| ● | ORGANIZATIONAL STRUCTURE | 79 |
| ● | AUTOMATION | 69 |
| ● | SKILLS & ROLES | 52 |
| ● | MANAGEMENT | 44 |

# **Density of codes (number of relationships between codes – co-occurrence in the same quotation)**

## Co-occurrence table per code

| **● alignment of dev & ops goals Gr=11** | 0 | 0 | 0 | 0 | 0 | 5 | 3 | 0 | 2 | 1 | 0 | 0 | 1 | 1 | 4 | 0 | 0 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | **21** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **● automated application life-cycle management Gr=42** | 0 | 0 | 0 | 0 | 1 | 12 | 1 | 0 | 4 | 1 | 1 | 2 | 4 | 5 | 9 | 0 | 4 | 0 | 4 | 0 | 5 | 1 | 1 | 0 | 1 | **56** |
| **● automated infrastructure management Gr=8** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 1 | 0 | 0 | **12** |
| **● blame Gr=4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | **2** |
| **● change management Gr=5** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **2** |
| **● collaboration Gr=54** | 5 | 12 | 1 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 1 | 3 | 3 | 4 | 13 | 0 | 3 | 0 | 14 | 6 | 10 | 0 | 2 | 1 | 3 | **89** |
| **● communication Gr=15** | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 1 | 2 | 0 | 3 | 0 | 2 | 0 | 5 | 0 | 3 | 0 | 1 | 1 | 0 | **25** |
| **● continuous improvement Gr=3** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **2** |
| **● cross-functionality/skills Gr=36** | 2 | 4 | 2 | 0 | 0 | 8 | 3 | 0 | 0 | 2 | 1 | 1 | 5 | 0 | 5 | 1 | 3 | 2 | 10 | 4 | 6 | 0 | 3 | 0 | 1 | **63** |
| **● cultural silos/conflicts Gr=6** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | **11** |
| **● culture, values & best practices Gr=7** | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 2 | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | **12** |
| **● devops (bridge) team Gr=19** | 0 | 2 | 2 | 0 | 0 | 3 | 1 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 6 | 1 | 2 | 0 | 0 | 0 | 4 | **28** |
| **● enabler (platform) team Gr=28** | 1 | 4 | 0 | 0 | 0 | 3 | 2 | 0 | 5 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 15 | 0 | 5 | 1 | 4 | 0 | 4 | 5 | 0 | **53** |
| **● metrics, visibility & feedback Gr=11** | 1 | 5 | 0 | 1 | 0 | 4 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | **15** |
| **● organizational silos/conflicts Gr=32** | 4 | 9 | 0 | 0 | 0 | 13 | 3 | 0 | 5 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 7 | 1 | 4 | 0 | 4 | 0 | 1 | **56** |
| **● platform builder Gr=4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **6** |
| **● platform servicing Gr=19** | 0 | 4 | 0 | 0 | 0 | 3 | 2 | 0 | 3 | 0 | 1 | 0 | 15 | 0 | 1 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 3 | 2 | 0 | **38** |
| **● product management Gr=3** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | **4** |
| **● responsibility/ownership sharing Gr=49** | 2 | 4 | 4 | 1 | 0 | 14 | 5 | 0 | 10 | 1 | 0 | 6 | 5 | 1 | 7 | 0 | 4 | 1 | 0 | 4 | 2 | 0 | 5 | 0 | 4 | **80** |
| **● role definition/attributions Gr=14** | 1 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 4 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 4 | 0 | 2 | 0 | 0 | 0 | 3 | **24** |
| **● skills/knowledge sharing Gr=28** | 0 | 5 | 2 | 0 | 0 | 10 | 3 | 0 | 6 | 0 | 1 | 2 | 4 | 1 | 4 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 1 | 0 | **43** |
| **● stack & tools sharing Gr=4** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** |
| **● team self-organization & autonomy Gr=15** | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 3 | 1 | 0 | 0 | 4 | 0 | 4 | 0 | 3 | 0 | 5 | 0 | 0 | 0 | 0 | 1 | 0 | **26** |
| **● training, evangelization and mentoring Gr=6** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 5 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | **12** |
| **● transfer of work between teams Gr=10** | 0 | 1 | 0 | 0 | 0 | 3 | 0 | 0 | 1 | 1 | 1 | 4 | 0 | 0 | 1 | 0 | 0 | 0 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | **19** |

## 2- Density of codes

| **CODE** | **DENSITY** |
| --- | --- |
| **● collaboration Gr=54** | **89** |
| **● responsibility/ownership sharing Gr=49** | **80** |
| **● cross-functionality/skills Gr=36** | **63** |
| **● automated application life-cycle management Gr=42** | **56** |
| **● organizational silos/conflicts Gr=32** | **56** |
| **● enabler (platform) team Gr=28** | **53** |
| **● skills/knowledge sharing Gr=28** | **43** |
| **● platform servicing Gr=19** | **38** |
| **● devops (bridge) team Gr=19** | **28** |
| **● team self-organization & autonomy Gr=15** | **26** |
| **● communication Gr=15** | **25** |
| **● role definition/attributions Gr=14** | **24** |
| **● alignment of dev & ops goals Gr=11** | **21** |
| **● transfer of work between teams Gr=10** | **19** |
| **● metrics, visibility & feedback Gr=11** | **15** |
| **● automated infrastructure management Gr=8** | **12** |
| **● culture, values & best practices Gr=7** | **12** |
| **● training, evangelization and mentoring Gr=6** | **12** |
| **● cultural silos/conflicts Gr=6** | **11** |
| **● platform builder Gr=4** | **6** |
| **● product management Gr=3** | **4** |
| **● blame Gr=4** | **2** |
| **● change management Gr=5** | **2** |
| **● continuous improvement Gr=3** | **2** |
| **● stack & tools sharing Gr=4** | **1** |

## Charts for codes with greater density

Gráfico

Descripción generada automáticamente

Patrón de fondo

Descripción generada automáticamente

Patrón de fondo

Descripción generada automáticamente

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

# **Density of semantic domains (number of relationships between semantic domains – co-occurrence in the same quotation)**

## 1- Co-occurrence table per semantic domain

| **● AUTOMATION Gr=69** | 0 | 21 | 12 | 33 | 19 | 12 | **97** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **● CULTURE Gr=89** | 21 | 0 | 16 | 34 | 41 | 22 | **134** |
| **● MANAGEMENT Gr=44** | 12 | 16 | 0 | 14 | 14 | 9 | **65** |
| **● ORGANIZATIONAL STRUCTURE Gr=79** | 33 | 34 | 14 | 0 | 30 | 17 | **128** |
| **● SHARING Gr=88** | 19 | 41 | 14 | 30 | 0 | 24 | **128** |
| **● SKILLS & ROLES Gr=52** | 12 | 22 | 9 | 17 | 24 | 0 | **84** |

## 2- Density of semantic domains

| **CODE** | **DENSITY** |
| --- | --- |
| ● CULTURE Gr=89 | 134 |
| ● ORGANIZATIONAL STRUCTURE Gr=79 | 128 |
| ● SHARING Gr=88 | 128 |
| ● AUTOMATION Gr=69 | 97 |
| ● SKILLS & ROLES Gr=52 | 84 |
| ● MANAGEMENT Gr=44 | 65 |

## 3- Charts for semantic domains with greater density

Patrón de fondo

Descripción generada automáticamente con confianza media

Patrón de fondo

Descripción generada automáticamentePatrón de fondo

Descripción generada automáticamente

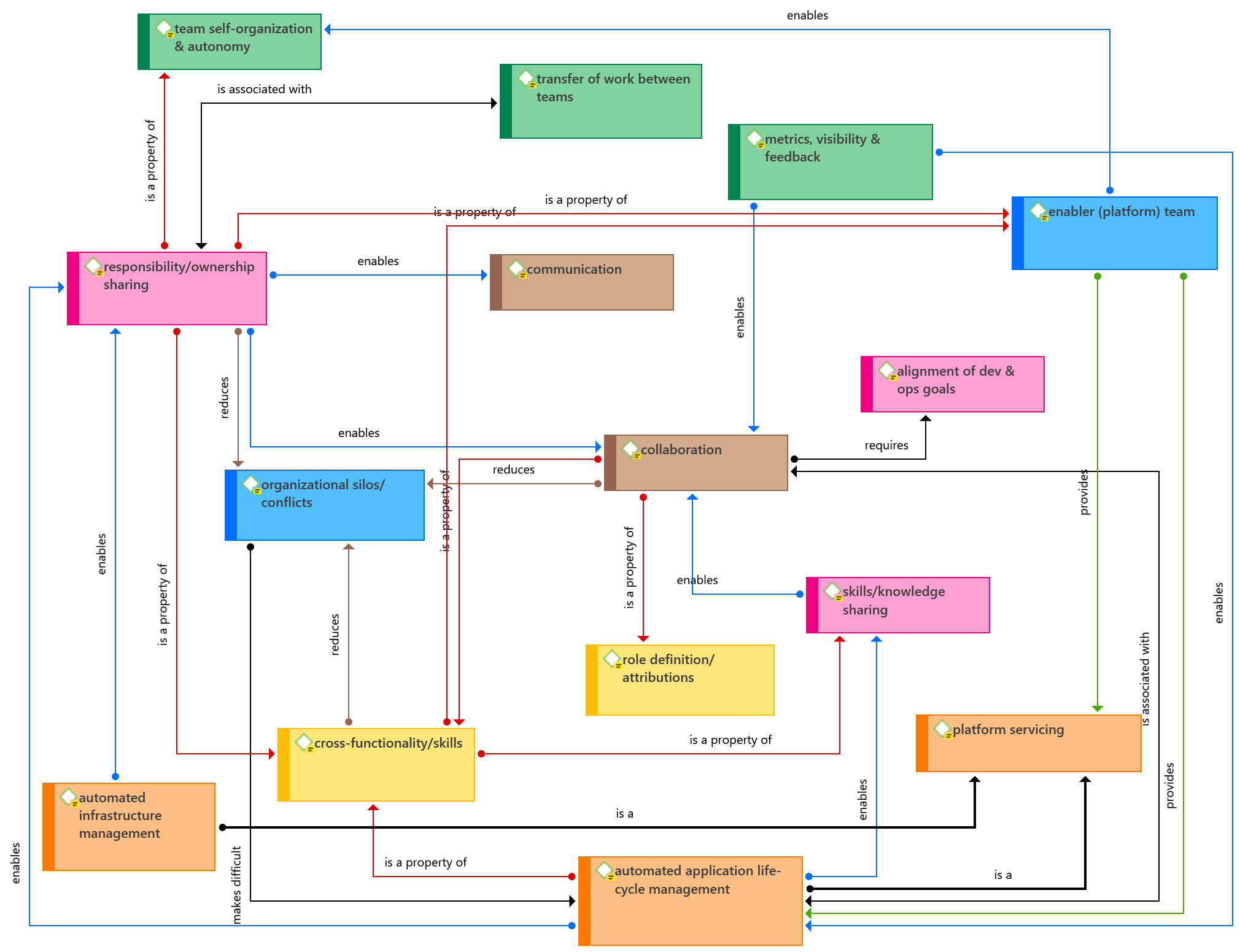
# **Networks**

## Network – Semantic Domains

Diagrama

Descripción generada automáticamente

## Network – Codes



A key word in the definition of DevOps is **collaboration** between teams. This collaboration ranges from the lack of (or eventual) collaboration, which may generate conflicts and disagreements on decisions, to daily collaboration (working together regularly on a daily basis). Starting from the code collaboration, which is the code with the highest groundedness **(grounded=54)** and density **(density=89)** (also at the level of semantic domain, this is, culture), we are going to analyze the enablers of collaboration between teams and the relationships and impact of collaboration on teams structures.

A team culture based on **responsibility/ownership sharing** ***enables* collaboration (density=14)** “5:18 *Mixing the responsibilities brought Devs and Ops closer to each other. Employees mentioned that Devs and Ops now collaborate on different tasks, since they now realize the importance of collaboration*”.

Promoting **Collaboration *reduces* organizational silos/conflicts** **(density=13)** “7:1 *A collaborative culture essentially aims to remove the silos between development and operations teams and activities*.” “2:11 *The classical DevOps structure focuses on collaboration among developers and the infrastructure team. It does not eliminate all conflicts, but promotes a better environment to deal with them*”. In fact, collaboration and automation of the application lifecycle (density=12) become two fundamental levers to reduce silos and conflicts between teams “7:3 *When a collaborative culture is fomented, teams collaborate to perform the tasks from the first day of software development. With the constant exercise of provisioning, management, configuration and deployment practices, software delivery becomes more natural, reducing delays and, consequently, the conflicts between teams*.” In addition, collaboration is strengthened after adopting practices such as CD “6:1 The amount of collaboration between teams and team members, in particular application developers and operations team, increased after adopting CD.”

**Automated application life-cycle management is associated with collaboration (density=12).** Collaboration impacts automated application life-cycle management and vice versa. Automation and collaboration mutually facilitate the adoption of the other, so they are complementary. *"4:1 Generally, participants described DevOps as better collaboration between developers and Ops teams. Some interviewees also described DevOps as end-to-end automation of the software development pipeline, providing better software quality, and creating seamless workflow of products at the shortest possible time." "6:1 [...] The amount of collaboration between teams and team members, in particular application developers and operations team, increased after adopting CD." "6:11 Organizations are increasingly improving collaboration among teams and team members to effectively initiate and adopt CD practices. We asked the survey respondents to rate how they strongly agree or disagree that the collaboration between teams (e.g., developers, quality assurance team, testers, and operations personnel) has increased in their respective organizations since the adoption of CD practices.". “7:10 Although transparency and sharing can be used to ensure collaboration even in manual tasks, with automation the points where silos may arise are minimized.” H*owever, as mentioned in quotation 2:21, they alone are not enough to achieve a high delivery performance *"2:21 Collaboration and delivery automation, critical values of the DevOps movement, are not enough to achieve high delivery performance."*

A team culture based on **knowledge sharing** ***enables* collaboration (density=10)** “7:10 [...] transparency and sharing can be used to ensure collaboration” “7:15 Sharing concepts contribute with the collaborative culture. For example, all team members gain best insight about the entire software production process”.

**Collaboration *is a property of* cross-functional teams (density=8)** “6:16-18 Collaboration and communications among team members can considerably increase by establishing cross-functional teams”.

**Collaboration *is a property of*** teams in which skills take precedence over roles, this is, the code **role definition/attributions (density=6)**; so if there are already separate roles, responsibilities are very clear and collaboration is not fostered or promoted *“2:23 Infra as development collaborator. The infrastructure staff contributes to the application code to optimize the system’s performance, reliability, stability, and availability. Although this aptitude requires advanced coding skills from infrastructure professionals, it is a suitable strategy for maintaining large-scale systems, like the ones owned by #I31.”.*

A **collaboration**-based culture ***requires* alignment of dev & ops goals (density=5)**, “7:5 A collaborative culture requires product thinking, in substitution to operations or development thinking.

A team culture based on **metrics/visibility/feedback** ***enables* collaboration (density=4)** “7:21 *regularly performing the measurement and sharing activities contributes to avoiding existing silos and reinforces the collaborative culture […]*” “6:19 *we found that the lack of suitable awareness on status of project (e.g., build status, release status) among team members can be a bottleneck for collaborative work and significantly hinders the CD success*.”

The second code to be analyzed is **responsibility/ownership sharing (grounded=49 & density=80)**. This code ranges from shared responsibility of the products to separate responsibilities and tasks (each team member has different responsibilities and tasks). The analysis to be done now starts from this code and verifies the relationship with the others.

**Responsibility/ownership sharing *is a property*** of **cross-functionality/skills (density=10) teams***"1:25 We observed that there exists a relationship between how product teams share the product ownership and how these teams are structured. For example, ID29 shows a high level of sharing of the product ownership within product teams, which are cohesive, small (less than 12 people), and multidisciplinary."*

**Responsibility/ownership sharing *reduces* organizational silos/conflicts (density=7)** *"9:16 Collaboration is front and center for DevOps work. An effective shared ownership model and partner team relationships are necessary for SRE to function. Like DevOps, SRE also has strong values shared across the organization, which can make climbing out of team-based silos slightly easier.". This impacts* productivity *"9:11 Rigid boundaries between “application development” and “production” (sometimes called programmers and operators) are counterproductive.".* This confirms a transitive relationship as **responsibility/ownership sharing** ***enables* collaboration and collaboration** promoting ***reduces* organizational silos/conflicts**.

**Responsibility/ownership sharing *is a property* of** organizational structures that rely on an **enabler (platform) team (density=6).** The existence of platform teams does not lead to a separation of responsibilities but rather they become facilitators and make ownership sharing possible, unlike **devops (bridge) teams** that become new silos with their own responsibilities (e.g., deployment, monitoring, etc.).*"2:33 Although the product team becomes fully responsible for NFRs of its services, it is not a significant burden that developers try to refuse (#I33). The platform itself handles many NFR concerns, such as load balancing, auto-scaling, throttling, and high-speed communications between data-centers (#I4, #I8, #I16, #I33). [...] Moreover, we observed infrastructure people willingly supporting developers for the sake of services availability, performance, and security (#I9, #I14)."*

**Responsibility/ownership sharing *is a property of* teams self-organization & autonomy (density=5)**. *"8:8 Giving developers operational responsibilities has greatly enhanced the quality of the services, both from a customer and a technology point of view. The traditional model is that you take your software to the wall that separates development and operations, and throw it over and then forget about it. Not at Amazon. You build it, you run it. This brings developers into contact with the day-to-day operation of their software [...]"*.

A team culture based on **responsibility/ownership sharing** ***enables* communication (density=5) "**7:16 *with a solid understanding of shared responsibilities. A shared vocabulary also emerged from sharing and this facilitates communication.”*

**Responsibility/ownership sharing *is associated with the* transfer of work between teams** (density=4). If there is no shared responsibility, there is necessarily a transfer of work between development to production and operation teams (and vice versa). *"4:10 Here, developers are not responsible for application deployment and management. Completed applications or features are handed over to the DevOps teams for deployment and management [...]".* *"4:18 DevOps teams are responsible for all deployments. Developers handover applications to these teams, who oversee the journey through the CI/CD pipeline. The teams also monitor the applications and function as the first line of support."*

**Automated infrastructure management *enables* responsibility/ownership sharing (density=4)** *"5:6 [...] “If infrastructure is code, then almost by definition, infrastructure becomes to some degree a function of development, or at least so hard to separate from development that the distinction becomes almost irrelevant.” Assuming that infrastructure is code, this statement suggests that Approach 1 (mix responsibilities) is a natural approach, because Ops will be involved in Dev tasks by developing the infrastructure together with the Devs."*

**Automated application life-cycle management *enables* responsibility/ownership sharing (density=4)** *"6:15 [...] Our results reveal that the speed and frequency demanded by DevOps and CD practices drive the need for a more holistic view, in which team members from each side of the fence are needed to jointly work together and adopt shared responsibility as much as possible."*

The next code to be analyzed is **cross-functionality/skills** **(grounded=63 & density=63).** This code goes from multidisciplinary/poly-skilled teams (i.e., teams with all the necessary skills such as development, infrastructure, etc.) to teams with a lack of skills/knowledge/background to deal with dev and ops tasks.

**Skills/knowledge sharing *is a property of*** teams characterized by **cross-functionality/skills (density=6)** “5:7 As for Approach 2 (mix personnel), it is stated in [12] that creating cross-functional teams is a good approach when adopting DevOps. These teams should consist of Devs, testers, Ops personnel and others, and then each of them would contribute code to a shared repository.**”**

Enabler (platform) team is a team characterized by cross-functionality/skills, then **cross-functionality/skills *is a property of* enabler (platform) team (density=5).** An enabler (platform) team is responsible for providing platform servicing and tools (mainly for infrastructure and deployment pipelines), consulting, training, evangelization, mentoring, human resources, etc. to product teams. Therefore, at times providing a **platform service** will require development skills. Some companies apply this structure due to a lack of resources.

*"2:39 If the organization develops a new platform to deal with its specificities, it will require development skills from the infrastructure team. Nevertheless, even without developing a new platform, the infrastructure team must have a ‘‘dev mindset’’ to produce scripts and use infrastructure-as-code [43] to automate the delivery path (#I14). One strategy we observed to meet this need was to hire previous developers for the infrastructure team (#I14)."*

*"1:21 Figure 1 shows a cross team composed of highly qualified engineers in DevOps culture, specifically 5 senior developers, 10 testers and quality assurance engineers, and 10 IT operators (second-level operations), who get involved in product teams when necessary. These engineers are involved in product teams with exclusive dedication but limited in time, until product teams are capable of doing all their responsibilities, from planning, analysis, development, testing, deployment, to operation. This means that these horizontal teams are composed of engineers that move through the product teams according to their needs. The reason that these engineers are not part of the product teams is that these organizations (like ID2) do not have human resources enough to involve the necessary engineers in all the product teams."*

**Cross-functionality/skills reduces organizational silos/conflicts (density=5).** Often, as in the following quotation, when dealing with organizational silos/conflicts, organizations reach a team characterized by cross-functionality/skills. "1:11-12 Consolidated product teams, which have dealt with both organizational and cultural silos by aligning dev & ops goals with business goals and show cross-functional teams with shared product ownership, end-to-end product vision and high-levels of self-organization and autonomy." "7:1 A collaborative culture essentially aims to remove the silos between development and operations teams and activities. As a result, operations tasks—like deployment, infrastructure provisioning management, and monitoring— should be considered as regular, day-to-day, development activities. This leads to the first concept related to this core category: operations tasks should be performed by the development teams in a seamless way."

**Automated application life-cycle management *is a property of*** teams characterized by **cross-functionality/skills (density=4).** A cross-functional team is usually responsible for the entire application lifecycle. Therefore, the lifecycle must be as automated as possible. With that, there is a need to learn and apply new practices and tools and understand more about the entire project stack.*"6:21 Interestingly most of the respondents indicated that they have to constantly learn best practices and new tools for reliable release [...]. Working in CD context necessitates developing monitoring skills and spending more time on monitoring to triage and quickly respond to production incidents. As stated by R20 “Ensuring the product stays deployment ready all the time. Each check-in and change gets monitored” [...]. In addition, for some of the respondents adopting CD means to understand the whole stack of the application: database, backend, front-end, OS, and build. This helped them to further and better be involved in bug fixing (e.g., “More in depth knowledge of the entire stack - to debug when something fails” R38)".*

The code **automated application life-cycle management (grounded=42 & density=56)** refers to the automation of some of the processes of the application life cycle (from development and deployment pipelines to monitoring tasks) and the tools adoption or tool providing for supporting these processes. The level of automation may range from high-level of build automation (continuous integration, CI); testing automation (continuous testing or quality assurance automation); delivery automation (continuous delivery, CD); deployment automation (continuous deployment, CD); operational tasks automation (continuous measurement/feedback/monitoring from operations to development); and recovery automation TO non-automation.

**Organizational silos/conflicts *makes*** the adoption of an **automated application life-cycle management *difficult* (density=9).** The lack of automation is visible in organizations with silos/conflicts *"2:10 We observed a lack of proper test automation in many organizations [...] Although siloed organizations are not the only ones that lack test automation (#I3, #I32, #I35), in this structure developers can even ignore its value (#I5, #I23, #I37). "* Futhermore, if automation is not correctly applied *"2:8 p 6 [...] a ‘‘DevOps team’’ maintaining the deployment pipeline behaves as another silo, sometimes bottlenecking the delivery." "6:7 [...] if organizations want to efficiently adopt and implement DevOps practices, in particular CD practices, they cannot really have operations silo (i.e., separate Ops team), even small one. Having operations silo may lead to a lot of frictions in deployment process and fail organizations to achieve the real anticipated benefits of CD practices."*

**Metrics, visibility & feedback enables automated application life-cycle management (density=5)** We noticed that when metrics, visibility & feedback are applied, it is possible to monitor risks and make decisions throughout the continuous delivery pipeline. *"6:14 [...] A few number of the participants emphasized that having shorter feedback loop at each stage in CDP enables teams and team members to partner in producing high quality software." "7:21 [...]In the same way as continuous measurement, quality assurance is a category that can work both as enabler and as outcome. As enabler because increasing quality leads to more confidence between the teams, which in the end generates a virtuous cycle of collaboration. As outcome, the principle is that it is not feasible to create a scenario of continuous delivery of software with no control regarding the quality of the products and its production processes [...]."*

**Automated application life-cycle management *enables* skills/knowledge sharing (density=5).** *"7:10 In addition to contributing to transparency, automation is also considered important to ensure reproducibility of tasks, reducing rework and risk of human failure. Consequently, automation increases the confidence between teams, which is an important aspect of the collaborative culture."*

**Enabler (platform) teams *provides* automated application life-cycle management (density=4)** *"2:32 Platform teams are infrastructure teams that provide highly automated infrastructure services that can be self-serviced by developers for application deployment. The infrastructure team is no longer a ‘‘support team’’; it behaves like a product team, with the ‘‘platform’’ as its product and developers as internal customers." "9:10 an SRE team winds up automating all that it can for a service, leaving behind things that can’t be automated.".* This confirms a transitive relationship between enabler (platform) teams ***provides*** platform servicing and automated application life-cycle management **is a** platform servicing.

Finally, it is important to highlight the following relationships.

**Enabler (platform) team** ***provides* platform servicing (density=15)** *“1:18 […] we also realized that there were different kinds of horizontal teams such as DevOps Center of Excellence (DevOps CoE), DevOps chapter and Platform team. Despite there are some differences among these teams, all of them refer to the same construct, i.e., teams that provide platform, infrastructure, IT operation”.*

**Enabler (platform) team** ***enables*****team self-organization & autonomy (density=4)** “*2:40**All four organizations that have fully embraced the platform team structure are high performers, while no other structure provided such a level of success (Table 3). An explanation for such a relation is that this structure decouples the infrastructure and product teams, which prevents the infrastructure team from bottlenecking the delivery path. As stated by #I20: ‘‘Now developers have autonomy for going from zero to production without having to wait for anyone’’. This structure also contributes to service reliability by letting product teams handle non-functional requirements and incidents.”*

**Automated application life-cycle management *is a* Platform servicing (density=4)** *“2:32 Platform teams are infrastructure teams that provide highly automated infrastructure services that can be self-serviced by developers for application deployment. The infrastructure team is no longer a ‘‘support team’’; it behaves like a product team, with the ‘‘platform’’ as its product and developers as internal customers.”.*

**Automated infrastructure management *is a* Platform servicing** “*7:17 The use of infrastructure as code was recurrently cited as a means for guaranteeing that everyone knows how the execution environment of an application is provided and managed”*

# **Analysis per document**

|  | **D1 2021 UPM Paper Gr=29** | **D2 2021 USP Paper Gr=42** | **D4 2020 Macarthy Paper Gr=21** | **D5 2016 Nybon Paper Gr=19** | **D6 2017 Shahin Paper Gr=21** | **D7 2019 Luz Gr=21** | **D8 2011 Humble Gr=21** | **D9 2022 How SRE Relates to DevOps Gr=20** | **D11 2020 State of DevOps Report Gr=3** | **D12 2022 A Cross-Company Ethnographic Study on Software Teams for Gr=4** | **Totals** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **● AUTOMATION Gr=69** | 6 | 16 | 12 | 1 | 9 | 12 | 8 | 2 | 2 | 1 | 69 |
| **● CULTURE Gr=89** | 17 | 18 | 6 | 8 | 10 | 13 | 6 | 9 | 0 | 2 | 89 |
| **● MANAGEMENT Gr=44** | 11 | 5 | 5 | 0 | 3 | 2 | 9 | 8 | 1 | 0 | 44 |
| **● ORGANIZATIONAL STRUCTURE Gr=79** | 19 | 22 | 15 | 5 | 9 | 3 | 1 | 5 | 0 | 0 | 79 |
| **● SHARING Gr=88** | 7 | 23 | 10 | 12 | 6 | 12 | 8 | 8 | 0 | 2 | 88 |
| **● SKILLS & ROLES Gr=52** | 13 | 19 | 4 | 5 | 6 | 2 | 3 | 0 | 0 | 0 | 52 |
| **Totals** | 73 | 103 | 52 | 31 | 43 | 44 | 35 | 32 | 3 | 5 | 421 |

Patrón de fondo

Descripción generada automáticamente

Patrón de fondo

Descripción generada automáticamente

Gráfico, Patrón de fondo

Descripción generada automáticamente

|  | **D1  Gr=29** | **D2  Gr=42** | **D4  Gr=21** | **D5  Gr=19** | **D6  Gr=21** | **D7  Gr=21** | **D8  Gr=21** | **D9  Gr=20** | **D11  Gr=3** | **D12  Gr=4** | **Totals** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **● alignment of dev & ops goals Gr=11** | 1 | 6 | 0 | 0 | 0 | 1 | 2 | 1 | 0 | 0 | 11 |
| **● automated application life-cycle management Gr=42** | 1 | 11 | 4 | 0 | 8 | 9 | 6 | 2 | 0 | 1 | 42 |
| **● automated infrastructure management Gr=8** | 0 | 0 | 4 | 1 | 0 | 2 | 1 | 0 | 0 | 0 | 8 |
| **● blame Gr=4** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 4 |
| **● change management Gr=5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 5 |
| **● collaboration Gr=54** | 8 | 13 | 4 | 5 | 7 | 8 | 5 | 3 | 0 | 1 | 54 |
| **● communication Gr=15** | 1 | 5 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 15 |
| **● continuous improvement Gr=3** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 3 |
| **● cross-functionality/skills Gr=36** | 7 | 13 | 3 | 3 | 5 | 2 | 3 | 0 | 0 | 0 | 36 |
| **● cultural silos/conflicts Gr=6** | 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 6 |
| **● culture, values & best practices Gr=7** | 3 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 7 |
| **● devops (bridge) team Gr=19** | 0 | 0 | 15 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 19 |
| **● enabler (platform) team Gr=28** | 12 | 11 | 0 | 1 | 2 | 0 | 0 | 2 | 0 | 0 | 28 |
| **● metrics, visibility & feedback Gr=11** | 0 | 0 | 0 | 0 | 2 | 1 | 4 | 4 | 0 | 0 | 11 |
| **● organizational silos/conflicts Gr=32** | 7 | 11 | 0 | 2 | 5 | 3 | 1 | 3 | 0 | 0 | 32 |
| **● platform builder Gr=4** | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| **● platform servicing Gr=19** | 5 | 8 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 0 | 19 |
| **● product management Gr=3** | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 3 |
| **● responsibility/ownership sharing Gr=49** | 5 | 12 | 9 | 10 | 3 | 4 | 4 | 2 | 0 | 0 | 49 |
| **● role definition/attributions Gr=14** | 2 | 10 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 14 |
| **● skills/knowledge sharing Gr=28** | 1 | 7 | 2 | 3 | 3 | 7 | 2 | 2 | 0 | 1 | 28 |
| **● stack & tools sharing Gr=4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 1 | 4 |
| **● team self-organization & autonomy Gr=15** | 8 | 3 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | 0 | 15 |
| **● training, evangelization and mentoring Gr=6** | 4 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 6 |
| **● transfer of work between teams Gr=10** | 3 | 2 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 |
| **Totals** | 73 | 112 | 53 | 32 | 43 | 44 | 35 | 33 | 3 | 5 | 433 |