

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

How Agile Project Management Shapes the Incremental Creation of Digital Platforms

TESI DI LAUREA MAGISTRALE IN

MANAGEMENT ENGINEERING – INGEGNERIA GESTIONALE

Author: **Melisa Saltoglu**

Student ID: 270264
Advisor: Prof. Daniel Trabucchi
Co-advisor: Lorenzo Lione
Academic Year: 2025-26

Abstract

Digital platforms differ from traditional businesses because they must serve two distinct groups of users at the same time, and because the value each group receives depends on the presence of the other. Agile project management, which was developed for settings with a single customer group, is widely used by the startups that build these platforms. How these two logics interact during the creation of a platform has received little attention. This thesis examines that relationship through a multiple-case study of three digital platform startups operating in different sectors and at different stages of development, based on semi-structured interviews with founders and managers.

Three mechanisms were identified. First, the balance between the two sides was a continuous concern rather than a problem solved at launch, and because decisions about this balance affect both sides at once, they were not handled through formal Agile practices. In the earliest-stage case, agility appeared as a mindset rather than as a set of tools. Second, what drove iteration changed with the completeness of the platform's sides rather than with the age of the venture, since behavioural data becomes difficult to interpret while only one side is populated. Third, none of the startups settled disagreements between the two sides by ranking one above the other in the backlog. The conflict was moved instead into the design of the platform or into its governance, depending on whether the two sides could be addressed separately.

Taken together, these findings suggest that platform characteristics shape how Agile is used, and that the questions Agile cannot settle within its own process become permanent features of the platform itself.

Keywords: digital platforms, platform startups, Agile project management, two-sided markets, multiple-case study

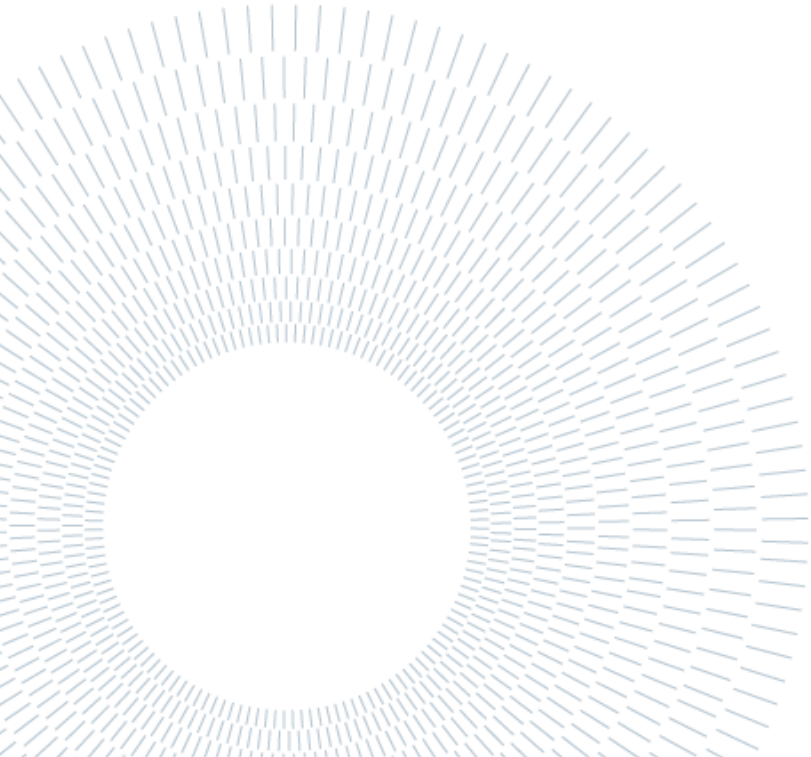
Abstract in italiano

Le piattaforme digitali si distinguono dalle imprese tradizionali perché devono servire contemporaneamente due gruppi distinti di utenti, e perché il valore che ciascun gruppo riceve dipende dalla presenza dell'altro. L'Agile project management, sviluppato per contesti caratterizzati da un unico gruppo di clienti, è ampiamente utilizzato dalle startup che costruiscono queste piattaforme. Il modo in cui queste due logiche interagiscono durante la creazione di una piattaforma è stato finora poco indagato. Questa tesi esamina tale relazione attraverso un multiple case study di tre startup di piattaforme digitali, operanti in settori diversi e in diverse fasi di sviluppo, sulla base di interviste semi-strutturate a fondatori e manager.

Sono stati individuati tre meccanismi. In primo luogo, l'equilibrio tra i due lati si è rivelato una preoccupazione costante, e non un problema risolto al momento del lancio; poiché le decisioni relative a tale equilibrio incidono su entrambi i lati allo stesso tempo, esse non sono state gestite attraverso pratiche Agile formali. Nel caso in fase più iniziale, l'agilità si è manifestata come un mindset più che come un insieme di strumenti. In secondo luogo, ciò che guida l'iterazione cambia in funzione della completezza dei lati della piattaforma, piuttosto che dell'età dell'impresa, poiché i dati comportamentali risultano difficili da interpretare finché è popolato un solo lato. In terzo luogo, nessuna delle startup ha risolto i disaccordi tra i due lati anteponendo l'uno all'altro nel backlog. Il conflitto è stato invece spostato nella progettazione della piattaforma o nella sua governance, a seconda che i due lati potessero essere affrontati separatamente.

Nel complesso, questi risultati suggeriscono che le caratteristiche della piattaforma influenzano il modo in cui l'Agile viene utilizzato, e che le questioni che l'Agile non riesce a risolvere all'interno del proprio processo diventano caratteristiche permanenti della piattaforma stessa.

Parole chiave: piattaforme digitali, startup di piattaforma, Agile project management, mercati a due versanti, multiple case study



Contents

Abstract ..... **i**

Abstract in italiano ..... **iii**

Contents ..... **vii**

1. Introduction **1**

2. Literature Review **5**

2.1. Digital Platforms 5

2.1.1. Typology of Platforms 7

2.1.2. Platform Sides, Network Effects, and Early Growth Challenges 12

2.1.3. Platform Thinking 15

2.1.4. Platform Characteristics and Their Evolution 17

2.2. Agile Project Management 20

2.2.1. Agile as a Tool 21

2.2.2. Agile as a Culture 22

2.3. Research Gap: Agile Project Management and Digital Platform Creation 24

3. Methodology **26**

3.1. Research Setting and Case Selection 26

3.2. Data Collection 28

3.3. Data Analysis 29

4. Findings **32**

4.1. Case A 32

4.1.1. Platform Configuration 32

4.1.2. Platform Development 33

4.1.3. Ways of Working 35

4.2. Case B 36

4.2.1. Platform Configuration 36

4.2.2. Platform Development 37

4.2.3. Ways of Working 39

4.3. Case C 39

4.3.1. Platform Configuration 40

4.3.2. Platform Development 41

4.3.3. Ways of Working 42

4.4. Cross-Case Analysis 43

5. Discussion **45**

5.1. Overview....45
5.1.1. The two-sided balance and the form Agile takes....45
5.1.2. The second side and what drives iteration....49
5.1.3. Conflicting sides and the limits of the backlog....51
5.2. Theoretical Propositions....54
6. Conclusion and future developments....57
Bibliography....61
Appendix A: Semi-Structured Interview Protocol....67
List of Figures....71
List of Tables....73
Acknowledgments....75

1. Introduction

Platform-based businesses operate differently from traditional firms. While a conventional company produces a product and sells it to a single customer base through a linear value chain (Porter, 1985), a platform creates value by connecting distinct groups of users (Rochet & Tirole, 2003; Van Alstyne et al., 2016). Consequently, building a platform is structurally different from developing a standard digital product. Platform creators cannot design for a single target audience. They need to consider at least two user groups whose participation depends on each other. This need to win over multiple sides makes the early stages of platform creation highly complex and uncertain, since the response of each side depends on the presence of the other and cannot be assessed separately (Ojala & Lyytinen, 2022). Founders must make critical decisions about the platform's design and rules before a stable user base exists. For example, a new marketplace has to decide which features to build first, which side to attract first, and how much to charge each side, often before it knows how either side will respond. Because the requirements of a platform are not clear at the start and are likely to change while it is being created, the detailed upfront planning assumed by traditional management approaches is rarely effective (Rigby et al., 2016). Instead, managing this uncertainty requires taking small steps, continuously learning, and making quick adjustments. These adaptive practices are the core of Agile project management (Beck et al., 2001; Bianchi et al., 2020; Ghezzi & Cavallo, 2020). Rather than planning the whole product in advance, Agile teams work in short cycles, test small versions, and use what they learn to decide the next step.

Despite this connection, the intersection of these two domains has received little attention. Agile research has developed largely within traditional software development contexts (Dingsøyr et al., 2012), typically assuming a single, unified customer group. In a platform environment, however, there are at least two sides whose needs frequently conflict. Conversely, platform research has focused primarily on architecture, governance, and ecosystem dynamics (Gawer, 2021; Tiwana et al., 2010), but rarely examines the platform project itself: the design decisions, the process behind the platform's creation, and the changes. That continues as the platform develops. As a result, it remains unclear how step-by-step development works when multiple

interdependent user groups are involved. For example, when the two sides of a platform ask for different things, it is not clear whose feedback should guide the next iteration. It is also not clear how Agile works in the early stage of a platform, when the chicken-and-egg problem means that both sides are still missing. Agile relies on feedback, but at this stage there are few users to provide it, even though the startup still has to decide what to build first.

To address this gap, this thesis investigates the intersection of Agile practices and platform development through a qualitative multiple-case study of three digital platform startups in Türkiye. The sample was designed so that both phenomena of interest would be present. Startups were selected because Agile ways of working are commonly found in this setting (Ghezzi & Cavallo, 2020), which makes it likely that Agile practices and mindset can actually be observed. The selected companies are also born-platforms, that is, ventures founded around a platform business model rather than firms that moved to a platform model later (Teece et al., 2022). This makes it possible to observe the creation of the platform itself, while decisions about design, rules, and pricing are still open. In established firms, many of these decisions have already been made and are harder to change (Jovanovic et al., 2022). Within these criteria, the cases were selected to cover platforms at different stages of development. Data were collected through eight interviews with founders and advisors, conducted in multiple rounds and complemented by written follow-up exchanges. The study is guided by two main research questions. RQ1 asks: “What role do Agile practices and mindset play in the creation of platform startups?” RQ2 asks: “How do platform characteristics, particularly side configuration and development stage, shape the use of Agile practices and the resulting platform changes?” In this study, Agile is considered in two ways: as formal practices, such as sprints and planned meetings, and as a mindset of learning and adjusting step by step. While RQ1 looks at the role of Agile in the creation of the platform, RQ2 looks at the opposite direction: how the characteristics of the platform influence the way Agile is used.

Across the cases, Agile ways of working and platform creation were found to affect each other in several ways. First, the chicken-and-egg problem was a continuous concern rather than something solved at launch. The startups studied kept adjusting how they approached each side, for example by changing which side was charged and which side was given free access, in order to attract and keep both sides on board as the platform grew. Because these decisions affected both sides at the same time, they could not be divided into separate tasks, and the earliest-stage case worked iteratively without formal Agile practices, relying on an Agile mindset rather than

on Agile tools. The more mature case also used formal processes such as sprints, and its changes to the platform became smaller over time. Second, the stage of development appeared to affect what drove these adjustments. In the earliest stage, there was little feedback from either side, so the startups relied mainly on assumptions made by the founders on the basis of their industry knowledge. In the more mature case, these assumptions were increasingly replaced by user data. Third, having more than one user group changed how decisions were made about what to build, since the two sides provided different and sometimes conflicting feedback. Rather than choosing between competing needs in a single task list, the startups addressed these differences elsewhere, for example by building separate interfaces for each side or by introducing rules that protected both sides at once. Overall, based on the cases studied, these findings suggest that Agile and platform creation are closely connected. In these cases, the characteristics of the platform appeared to influence how Agile was used, while Agile ways of working in turn influenced how the platform was built and changed over time.

From a practical standpoint, the findings may help managers in platform startups, as well as in established firms launching platforms, to better understand how Agile ways of working can support platform creation. For startups, this may be useful when decisions have to be made quickly and with limited information. For established firms, it may help explain why a platform project needs a different way of working than the firm's traditional products.

The remainder of this thesis is structured as follows. Chapter 2 reviews the literature on digital platforms and Agile project management and identifies the research gap. Chapter 3 describes the research design, the selection of cases, and how data were collected and analysed. Chapter 4 presents each case and then compares them in a cross-case analysis. Chapter 5 discusses the findings in relation to existing literature. Chapter 6 summarises the contributions, discusses the limitations of the study, and suggests directions for future research.

2. Literature Review

2.1. Digital Platforms

Over the past two decades, platform-based businesses have become some of the most valuable and fastest-growing organisations in the digital economy (Van Alstyne et al., 2016). What makes them different is that they do not follow the traditional linear value chain, in which a firm carries out a sequence of activities that each add value to a product before it reaches the customer (Porter, 1985). In a traditional firm, value is created inside the company and then sold to customers (Kopalle et al., 2020). A platform instead creates value by bringing together two or more groups of users and letting them interact (Rochet & Tirole, 2003). This difference can be explained in more detail through the concept of two-sided markets. Evans (2003) identifies three conditions that must be met for a two-sided platform to exist. First, there must be two or more distinct groups of customers. Second, the value that one group receives must depend on the other group, for example, because it increases when more members of the other group join. This interdependence is what the literature calls network effects. Third, there must be an intermediary, the platform, that connects the groups, because they cannot easily create this value by interacting on their own. By taking on this role, the platform internalises the effects that each group has on the other.

Based on these conditions, not every digital business can be considered or started as a platform. Amazon, for example, started as a digital retailer. It bought books from publishers and sold them directly to customers, which is a linear value chain. It became a two-sided platform only after launching Amazon Marketplace, which connected buyers with third-party sellers (Trabucchi & Buganza, 2019). Only at that point did it meet the three conditions described above (Figure 2.1). The key feature of a platform is therefore not the technology behind it, but the presence of at least two user groups that create value for each other. Amazon's example also shows that these groups can change over time. Later, Amazon added a new group, advertisers, such as brands that pay to make their products more visible to buyers. The relationship between these groups is different: buyers and sellers complete transactions with each other through the platform, while advertisers pay to reach buyers without a direct transaction between them.

Trabucchi and Buganza (2022) describe the first type of relationship as transactional and the second as orthogonal, and refer to platforms that combine both as hybrid multi-sided platforms. Following this definition, Amazon can be seen as a hybrid multi-sided platform (Figure 2.2). These platform types are explained in more detail in the next section.

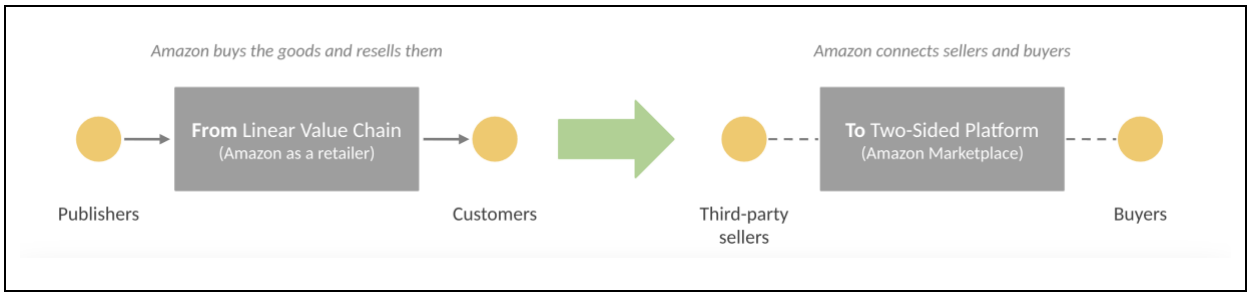


Figure 2.1. Amazon's shift from a linear value chain to a two-sided platform. Source: author's elaboration based on Hagiu & Wright (2015).

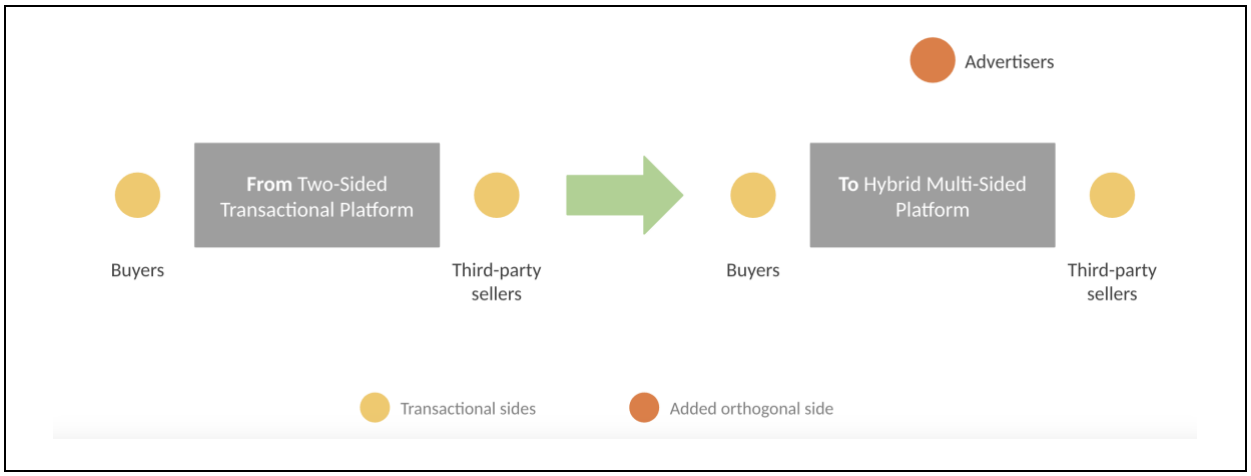


Figure 2.2. Amazon's evolution from a two-sided transactional platform to a hybrid multi-sided platform. Source: author's elaboration based on Trabucchi & Buganza (2022).

Platform value is also highly concentrated geographically. Most of the value of the world's largest platforms comes from American companies, which account for around 86% of the total, while European platforms represent only about 2% (Schmidt, 2025) (Figure 2.3). Schmidt (2025) does not expect this situation to improve soon, as European investors tend to lose patience with the often long process of building a platform. This suggests that creating a platform is not a single launch decision, but a long process that requires continuous development over time.

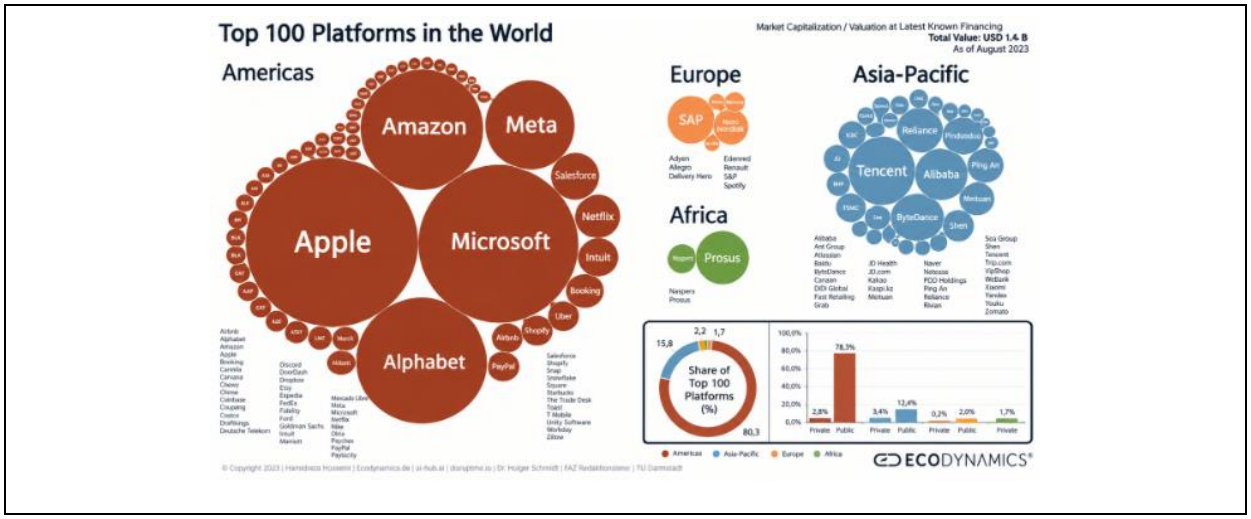


Figure 2.3. Geographical Distribution of the Market Value of the World's 100 Largest Platforms Source: Schmidt (2025)

2.1.1. Typology of Platforms

Platforms are not alike. The literature groups them into types based on how their sides relate to each other. This thesis follows the typology of Trabucchi and Buganza (2022), which distinguishes three main types: transactional, innovation, and orthogonal platforms. These types differ in who the sides are, how they interact, and how the platform creates and captures value.

Transactional platforms are two-sided markets that bring together two distinct groups of customers, such as hosts and travelers or buyers and sellers. Their main role is to enable a direct exchange between the two sides. The two sides interact with each other to complete a transaction, while the platform provides the rules and the environment for these transactions without producing or selling the goods or services itself (Täuscher & Laudien, 2018). To make these exchanges possible, platforms often help participants who do not know each other to trust the transaction. Airbnb, for example, allows hosts and guests to review each other after each stay, so that people who have never met feel safe doing business with each other. Each group becomes more valuable to the other as it grows, a dynamic known as cross-side network effects (Eisenmann et al., 2006). More hosts give travelers more choice, and more travelers give hosts more bookings. Transactional platforms usually capture value by charging a commission on these exchanges: in a study of 100 marketplaces, most platforms generated their revenues in this way (Täuscher & Laudien, 2018). This mutual dependence gives rise to the classic "chicken-and-egg" problem of having to attract both sides at once (Caillaud & Jullien, 2003): travelers do not join a platform without hosts, and hosts do not join a platform without travelers. Figure 2.4. illustrates these elements using Airbnb as an example.

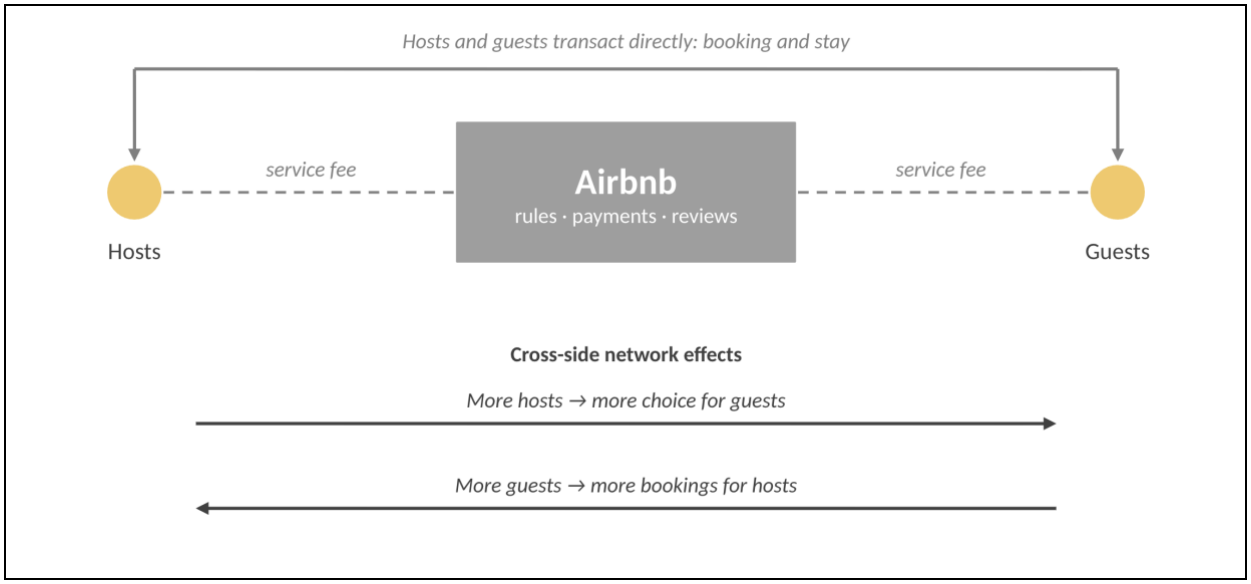


Figure 2.4. Transactional platforms: the example of Airbnb. Source: author's elaboration based on Trabucchi & Buganza (2022) and Eisenmann et al. (2006).

In Figure 2.4, hosts and guests are shown as two equal sides. In practice, however, the two sides are not equally easy to attract. In most cases, one side is harder to bring on board than the other. In many marketplaces, for example, attracting sellers is more difficult than attracting buyers, because sellers need to join before there are buyers to sell to. The harder side therefore requires more attention, more resources, and stronger incentives. The platform owner cannot treat both sides in the same way, and often needs different features, different incentives, and different pricing for each side. Eisenmann et al. (2006) describe this pricing decision as a choice between a "subsidy side", which is attracted with low prices or free access, and a "money side", which pays for access. The subsidised side is usually the one that is more sensitive to price, while the money side is the one that values the presence of the other side more.

Innovation platforms instead provide a technological foundation on which third parties build complementary products and services. Apple's iOS is a well-known example. These third parties are called complementors: independent developers or firms, such as the companies that create apps for the App Store. The platform owner provides the core technology, together with the tools and interfaces that allow complementors to build on top of it (Tiwana et al., 2010). In this way, the platform owner does not need to develop every product itself, since much of the variety and innovation comes from outside the firm (Gawer & Cusumano, 2014). The main difference from transactional platforms therefore lies in what the platform makes possible: transactional platforms mainly enable exchanges between sides, while innovation platforms mainly enable others to create new products (Cusumano et al., 2019).

The value of an innovation platform grows with the number and quality of its complements: more complements attract more users, and more users in turn attract more complementors. Platform owners can capture this value in different ways, for example by selling the core product or by keeping a share of the revenues that complementors earn. Because complementors join an already valuable technical base, the chicken-and-egg dynamic is weaker in this case (Gawer & Cusumano, 2014). At the same time, the platform owner has to decide how open the platform should be and which rules complementors must follow (Tiwana et al., 2010). More control can help the platform owner protect quality, but it can also limit the freedom of complementors.

Apple's iOS illustrates these elements well. The more useful apps are available on the App Store, the more attractive the iPhone becomes to users, and the larger the user base, the more attractive the platform becomes to developers. Apple captures value both by selling its devices and by keeping a share of the revenues that developers earn through the App Store. When the App Store opened in 2008, the iPhone already had millions of users, which gave developers a reason to join from the start. Apple also reviews apps before they are published, which gives it more control over quality and security but limits the freedom of developers. Figure 2.5 summarises these elements.

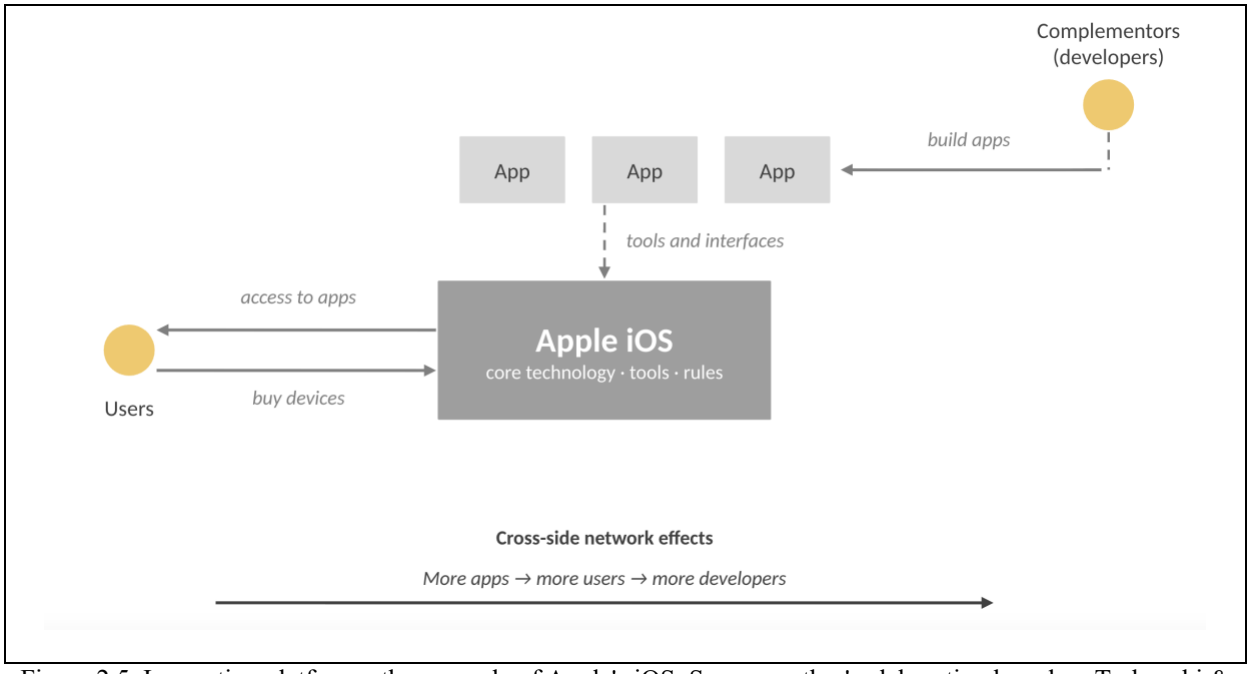


Figure 2.5. Innovation platforms: the example of Apple's iOS. Source: author's elaboration based on Trabucchi & Buganza (2022) and Gawer & Cusumano (2014).

Orthogonal platforms instead connect sides that do not need to interact directly, such as users and advertisers. In economic terms, these are non-transaction markets: there is no observable

transaction between the two sides, so the platform cannot charge a fee per transaction and instead charges one side for access to the other (Filistrucchi et al., 2014). The two sides are not looking for each other: users come to the platform for the service itself, not to meet the other side. In this case, the effect runs in one direction, as a larger and more active user base and the data it produces make the platform more valuable to the other side (Trabucchi & Buganza, 2022). The two sides are also treated very differently, since the service is usually offered to users for free, while the second side pays to reach them or to access the data they generate (Filistrucchi et al., 2014). This also changes how the platform starts: since the first side does not depend on the presence of the second one, the platform can build its user base first and add the paying side later. The chicken-and-egg problem is therefore weaker than in transactional platforms, but the platform has to reach a large enough user base before the second side becomes interested.

Strava illustrates this logic. The platform offers a service to people who run or cycle: they record their activities, follow their progress, and compare their results with others. These users do not join the platform because of a second side. Their activities, however, produce a large amount of data on how people move through a city. Strava makes this data available, in aggregated form, to a second side, such as city planners and transport authorities, who use it to understand cycling and running routes. Runners and planners never interact with each other, but the value that this second side receives depends entirely on how many people use the app and how active they are. Figure 2.6 summarises these elements.

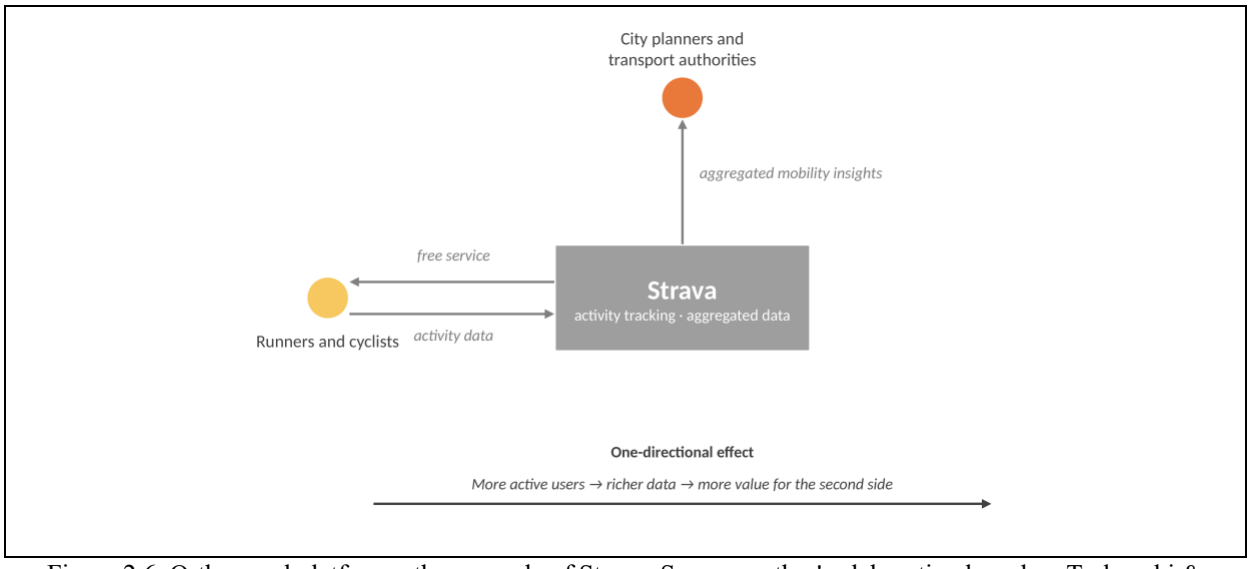


Figure 2.6. Orthogonal platforms: the example of Strava. Source: author's elaboration based on Trabucchi & Buganza (2022).

The number of sides can change over time. Many platforms start with two sides and add new ones later. Amazon is an example of this: it started by connecting buyers and third-party sellers and later added advertisers as a new side. Uber and Airbnb followed a similar path, moving from a transactional two-sided structure towards a hybrid multi-sided one as they added new sides over time (Trabucchi & Buganza, 2022). The number of sides is therefore not a fixed feature of a platform but a boundary decision that the platform firm makes and revises over time (Gawer, 2021).

Adding a side is not a simple extension. The platform has to create new interfaces for the new side, decide which rules apply to it, and make sure that the existing sides are not harmed by the change. Decisions that were taken when the platform had two sides may therefore need to be revised later. Because adding a side changes architecture, governance, and ecosystem relationships at the same time, side configuration is not only a way of classifying platforms but also part of what evolves during platform creation (Tiwana et al., 2010).

Trabucchi and Buganza (2022) represent these configurations with two intersecting lines: a transactional line between two customer sides, and an orthogonal line between a customer side and an orthogonal one. Combining them produces three forms: transactional two-sided, orthogonal two-sided, and hybrid multi-sided platforms (Figure 2.7). This means that side configuration is not only a way of describing a platform at a given moment, but also something that changes while the platform is being built. This thesis therefore uses side configuration as one of the dimensions for comparing the cases.



Figure 2.7. Transactional, Orthogonal, and Hybrid Multi-Sided Platforms Model (Trabucchi & Buganza, 2022)

2.1.2. Platform Sides, Network Effects, and Early Growth Challenges

A digital platform usually serves more than one group of users. In a transactional platform, these groups may be buyers and sellers, passengers and drivers, or customers and service providers. Both groups are customers of the platform, even though they use it for different reasons. The platform therefore needs to offer a clear and different value proposition to each side (Muzellec et al., 2015). At the same time, it has to manage the relationship between them. This makes the development of a platform different from the development of a digital product designed for one main customer group. When studying platforms, the literature broadly agrees on their defining characteristics, such as the presence of interdependent sides and the network effects between them (Eisenmann et al., 2006). Few studies, however, examine how to manage two groups that might go in different directions. When a firm builds a product for a single customer group, it can focus on understanding the needs of that group. A platform firm, in contrast, has to understand two groups with different and sometimes conflicting needs, and every decision it takes affects both of them at the same time.

Pricing is one example of this difference. In a traditional market, the price is determined by costs and by the demand elasticity of the customer group. In a two-sided market, the price charged to one side also depends on how the other side responds (Rysman, 2009). As a result, one side may be priced below cost, or even offered the service for free. The platform is therefore not only setting a price, but also deciding how to divide the total price between the two sides (Rochet & Tirole, 2003). Which side is subsidised depends on how sensitive each side is to price and on which side creates more value for the other (Eisenmann et al., 2006). A platform may, for example, offer free access to users in order to attract advertisers or sellers who are willing to pay for reaching them.

Pricing is not the only decision of this kind. Similar trade-offs appear when the platform decides which features to build, whose requests to prioritise, and which rules to apply to each side. Because the two sides depend on each other, a decision that makes the platform more attractive to one side may make it less attractive to the other.

The relationship between the sides is commonly explained through network effects. A same-side network effect occurs when the number of participants on one side changes the value experienced by participants on that same side. For example, a social platform may become more

useful when more users join because there are more people to interact with. Same-side effects can also run in the opposite direction on the other side of the platform: for buyers, a larger number of other buyers may mean more competition for the same limited supply. However, growth on the same side is not always positive. A larger number of sellers in the marketplace may also increase competition between them. Negative effects can also appear when a platform grows too quickly without managing quality, since users may face more irrelevant listings or unreliable counterparts, which makes the platform less useful even though it has more participants. Cross-side network effects refer to the effect that the growth of one side has on the other. More sellers can make a marketplace more attractive to buyers, while more buyers can make it more attractive to sellers (Eisenmann et al., 2006). Figure 2.8 summarises these effects. These effects are also referred to using different terms in the literature: same-side effects are often called direct network effects, and cross-side effects are often called indirect network effects. This distinction matters for platform creation, because the two types of effects do not appear at the same time. Cross-side effects only become visible once both sides are present, which means that in the earliest stage a platform often cannot rely on them yet.

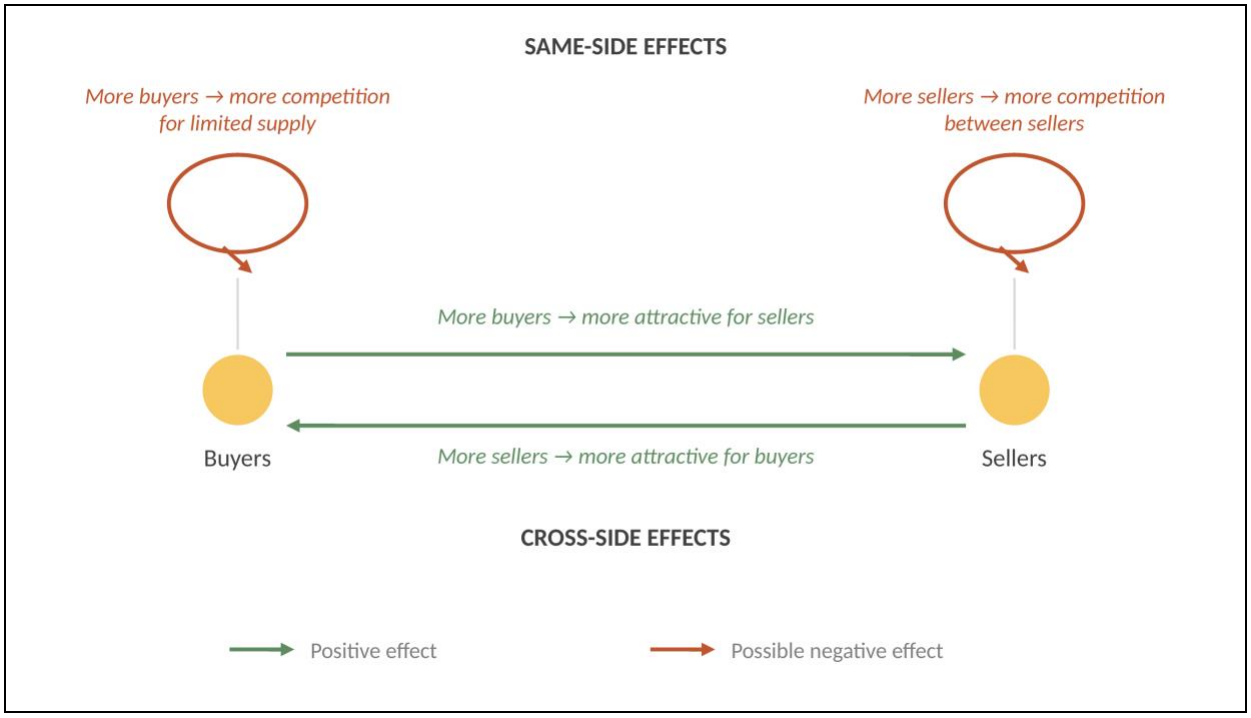


Figure 2.8. Same-side and cross-side network effects in a transactional platform. Source: author's elaboration based on Eisenmann et al. (2006).

Ojala and Lyytinen (2022) show that the number of participants alone does not fully explain the development of network effects. The type and variety of participants also matter. A platform

with many similar participants may generate weaker effects than a platform with fewer but more diverse ones, because variety increases the chance that participants find what they are looking for. Their study indicates that early actions often focus on making the platform technically usable, while later actions aim to support a wider range of participants and interactions. Network effects therefore change not only in size but also in nature: what makes participation valuable for each side can shift over time. They are thus built gradually, through a series of decisions, rather than appearing at a single moment. They are not an automatic outcome of user growth. They also depend on the choices made by the platform owner during the development process. Since these effects depend on decisions taken along the way, the platform cannot be planned entirely in advance. The owner has to observe how participants actually behave and adjust the platform accordingly, since the conditions that made the platform attractive at one stage may not be the ones that keep it attractive later.

The need to develop more than one side creates a particular problem during the launch stage. In a transactional platform, potential buyers may not join when there are too few sellers, while sellers may not join without sufficient demand. This situation is generally described as the chicken-and-egg problem. The problem is not only that the platform has few users, but that neither side has a reason to join first. Each side is waiting for the other, so the platform can remain empty even when the service itself works well (Caillaud & Jullien, 2003). It is especially relevant for startups because they normally begin without an established user base or a proven value proposition. A startup therefore has to attract its first users before it can show what the platform is worth, and it usually has to do so with limited resources. The decisions taken at this stage are made under high uncertainty, since the founders cannot yet observe how either side will behave.

Trabucchi (2020) identifies several practical responses to this problem. A platform may start from an existing network, using a group of users it already has access to; involve one or more important participants, whose presence attracts others; focus its activities on one side, concentrating its resources where the need is stronger; or temporarily perform the role of a missing side itself. For example, a marketplace may initially provide some of the supply itself before attracting independent suppliers. These decisions affect more than the number of users. They can also influence the first version of the service, the features developed by the startup, and the way the platform positions itself in the market. In other words, solving the chicken-and-egg problem is not only a marketing decision but also a product decision: what the platform

builds first depends on which side it decides to attract first. Since the outcome of these choices cannot be predicted in advance, they are often adjusted as the platform grows.

Network effects may also create a long-term attachment known as lock-in. As the platform grows, users build history, contacts, and routines on the platform. Leaving means losing these, which makes switching costly (Shapiro & Varian, 1999). The more each side invests in a single platform, the more difficult switching becomes. Lock-in does not affect both sides in the same way. A seller who has built a reputation and a customer base on a platform is usually more tied to it than a buyer who can easily try another one. The platform owner therefore has more freedom to change some parts of the service than others. Lock-in is an advantage for the platform owner because it makes the user base more stable. At the same time, it limits what the platform owner can change over time. Once participants depend on particular user interfaces, rules, or pricing arrangements, adjusting them may push some users away. Decisions taken early in the creation process consequently become harder to reverse, which narrows the space available for later changes to the platform (Tiwana et al., 2010). This creates a tension that is specific to platforms. The same growth that makes a platform valuable also makes it harder to change, which means that the room for adjustment becomes smaller exactly when the platform starts to work well.

The decisions made at the beginning play an important role in platform creation. However, they cannot all be made at once. When the platform is created, the owner has little information about future users, demand, and how each side will respond. These decisions are therefore revised as the platform develops. The platform owner learns from how participants actually use the service, and adjusts the sides, the features, and the rules accordingly. Some of these adjustments are small, while others change the structure of the platform, for example when a new side is added or when the paying side changes. Creating a platform is a step-by-step process that is shaped by feedback. Platform research, however, has paid more attention to the decisions themselves than to the way they are made and revised over time. How platform owners work while these decisions are still open is therefore less well understood.

2.1.3. Platform Thinking

Beyond classifying platform types, the literature also treats the platform as a strategic mindset rather than only a technical structure. Trabucchi and Buganza (2022) call this perspective Platform Thinking, a way of looking at a company's assets and relationships through a platform

logic. It is the ability to look at assets a company already has. These assets can be tangible, such as unused physical capacity, or intangible, such as data, knowledge, and customer relationships. They can be used to create value between different sides. In this perspective, a company does not need to start from a new idea. It can look at what it already has and ask whether these resources could serve a second side. A company with a large customer base, for example, may find that the data it collects is valuable to another group, or that its unused capacity can be offered to others. Trabucchi and Buganza also propose the Platform Thinking matrix, a tool that helps managers decide which platform opportunities to pursue and which platform type fits a given asset. The matrix combines the type of asset with the type of platform that can be built on it, so that the same asset can lead to different platform configurations depending on how it is used. Figure 2.9 presents this matrix.

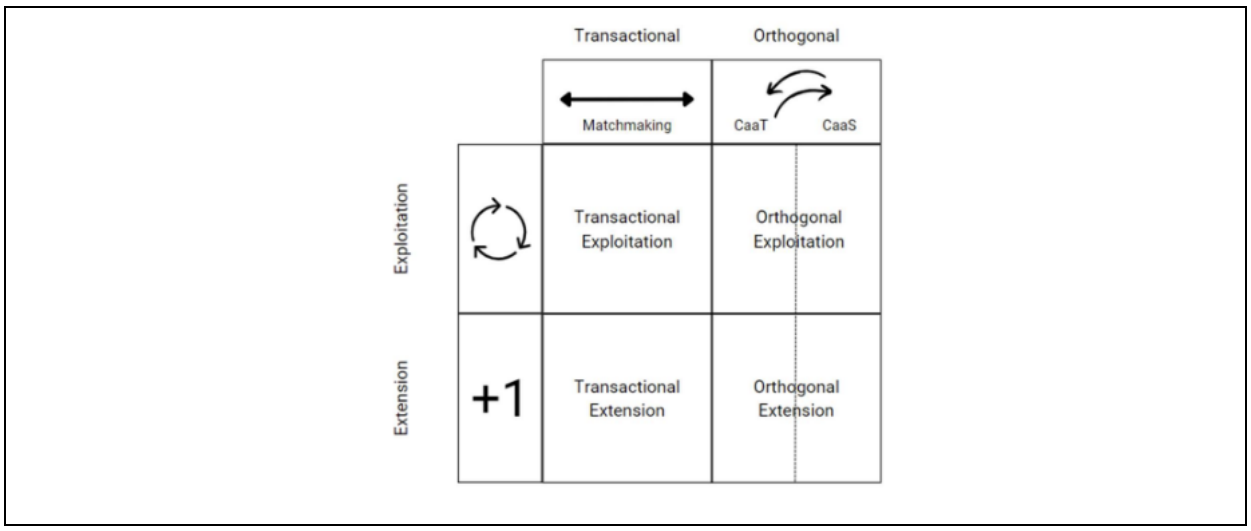


Figure 2.9. The Platform Thinking Matrix (Trabucchi & Buganza, 2022)

Platform Thinking was originally developed with established companies in mind, since it starts from assets that a company already owns. This may suggest that only established firms, which already hold such assets, can apply it. However, startups can also build on underused assets. Airbnb, for example, made use of spare rooms that were already available. The same applies to many other platform startups: Uber did not own cars, and marketplaces do not own the products they sell. What these startups contribute is not the asset itself, but the structure that allows others to offer it. These assets are not owned by the platform owner but by the participants in the ecosystem. The startup therefore does not reinterpret its own assets. Instead, it builds the structure that makes these assets accessible and usable. In a startup context, Platform Thinking is therefore mainly about building architecture and governance. This difference also has implications for how the platform is created. An established firm applying Platform Thinking

already knows the asset it wants to exploit and can plan around it. A startup, in contrast, has to discover which assets the participants are willing to share, under which conditions, and at what price. These are questions that cannot be answered before the platform exists, and the answers often change as the platform develops.

From this perspective, a platform is not a finished product. It keeps changing as new sides, complementors, and data are added. Platforms also change where innovation happens. Instead of keeping innovation inside the firm, they spread it across an ecosystem, which allows continuous experimentation by external participants. The role of the platform owner changes accordingly: rather than producing all the innovation itself, it provides the conditions under which others can innovate. This also means that the platform owner no longer sets the pace of development on its own, since changes can come from the ecosystem at any time. Supporting this kind of development requires organisational approaches that can keep up with the platform's pace.

2.1.4. Platform Characteristics and Their Evolution

The previous sections have shown that a platform is not created once and then left unchanged. Its sides, its network effects, and the decisions behind them keep developing over time. Building on this idea of platform creation as an ongoing process of change, growth, and adjustment, scholars have examined the complexity this brings for the firms that manage these platforms. For instance, Springer et al. (2025) view digital platforms not as static technological artefacts, but as evolving organisational systems whose architecture, governance, and ecosystem relationships adapt continuously over time. Despite this continuous evolution, platforms share a set of characteristics that can be identified and compared. This section focuses on the ones that are most relevant to platform creation.

Platform owners gradually expand the platform by introducing new functionalities, refining governance mechanisms, and attracting new participants to the ecosystem. Uber started as an exchange between drivers and passengers. It later added Uber Eats, which brought restaurants and couriers into the platform. These steps are not random. Adding a new side requires functions and rules that are already in place, such as payment, location tracking, and ratings. Uber Eats was built on this existing architecture. Established platforms add these steps to a structure that already exists. In startups, the first structure is still being built while these expansion decisions are made.

Three characteristics are particularly important during platform creation: architecture, governance, and ecosystem relationships. These dimensions do not evolve independently. They continuously influence one another during platform development. Changes in one dimension frequently trigger adjustments in the others. For example, introducing new functionalities or interfaces may require revisions to governance mechanisms that define participation rules, while the entry of new users or complementors can reshape architectural requirements and ecosystem relationships (Jovanovic et al., 2022).

The first of these dimensions, platform architecture, defines which parts form the core, and which parts are modules that can be changed. It also determines which interfaces are opened to different sides. These decisions affect how easily the platform can be changed later. In a modular structure, parts can be replaced individually. In a tightly coupled structure, every change affects the whole (Tiwana et al., 2010). Modularity is therefore not only a technical choice but also a way of managing uncertainty: when the platform is modular, decisions taken early can be revised later at a lower cost, and the firm does not need to know from the start what the final platform will look like.

Governance is not only about participation rules. It also determines how openness, access rights, and decision-making authority are shared among platform participants. These governance choices influence not only platform growth but also the pace and direction of platform evolution (Schreieck et al., 2025). Recent research emphasises the co-evolution of platform architecture, platform governance, and platform services as a defining characteristic of digital platform development (Jovanovic et al., 2022). Co-evolution means that these dimensions develop together rather than one after the other: a change in one creates pressure to adjust the others, and each adjustment opens or closes possibilities for the next. A similar argument is made for platform architecture, governance, and environmental dynamics (Tiwana et al., 2010). For the platform owner, this means that architecture, governance, and services cannot be designed separately or fixed once and for all.

Digital platforms also depend on an external ecosystem. In traditional firms, value is created mainly through internal activities. Platforms instead rely on users, complementors, developers, and business partners who contribute to the platform's development. As this ecosystem expands, participants introduce complementary products, services, knowledge, and data that change the platform's value proposition. The platform owner therefore does not create value directly. The role is to organise interactions and set the conditions under which participants can create value

together. Platform creation is therefore not only the development of a technological solution. It also involves building and coordinating an ecosystem that keeps changing (Gawer, 2021; Springer et al., 2025). However, this description assumes that a sufficient number of participants is already in place. In startups, such an ecosystem does not yet exist. The work of a startup is therefore not coordination. The main goal is to attract the first users. Startups need to build a user base before they can orchestrate an ecosystem (Trabucchi, 2020).

Architecture, governance, and ecosystem relationships apply to all digital platforms. However, their configuration differs across platform types and stages of development. Transactional platforms, for instance, tend to prioritise balancing interactions between multiple user groups, whereas innovation platforms place greater emphasis on attracting complementors and expanding technological capabilities. Digital platform startups are therefore unlikely to follow identical development paths. They differ in platform type, ecosystem maturity, and governance arrangements, and in startups these elements are still under active development. Comparing startups with different platform characteristics therefore helps to understand how these characteristics relate to differences in the way platforms are built and managed.

The continuous evolution of these characteristics also changes how platforms are managed. As new users, partners, and services are added, platform owners need to make ongoing decisions about architecture, governance, and ecosystem coordination. These decisions cannot be made once at the beginning of the project but must be revised as the platform evolves. In startups, decisions are made quickly by a small team without much formal process (Ghezzi & Cavallo, 2020). Established platforms, in contrast, have a much more layered decision-making process. As a result, startups make decisions at a faster pace, though often based on limited information.

A key challenge during platform creation is that many critical decisions must be made before the ecosystem is fully established. Platform owners often have limited information when these decisions are made. At this stage, it is not yet clear how many users will join, which side will grow faster, which features will be used, or whether the pricing model will work for both sides. This makes it difficult to define the most suitable architecture, governance mechanisms, or value proposition from the outset.

Detailed upfront planning is difficult in this environment. Any plan has to rest on assumptions that have not yet been tested. These assumptions can only be checked once the platform is in use. A plan therefore starts to become outdated as soon as it is put into practice. This is why the next section examines Agile Project Management.

2.2. Agile Project Management

Change and uncertainty have always been considered a challenge when managing projects. Agile was born as a response to this challenge, originating in software development with the Agile Manifesto (Beck et al., 2001) as a way of managing projects with a higher degree of flexibility. It prioritised individuals and interactions, working software, customer collaboration, and responsiveness to change over rigid, plan-driven processes. These values are accompanied by twelve principles that describe how they are put into practice, summarised in Table 2.1. Over time, these principles spread well beyond software to become a broader approach for organising work under uncertainty. A useful way to understand Agile adoption is the distinction between doing agile and being agile (Neumann et al., 2024). Doing agile means implementing practices, ceremonies, and tools such as sprints, stand-ups, and iterative releases. Being agile means internalising the mindset, values, and culture that give those practices meaning. Research suggests that the cultural shift is the more difficult one (Neumann et al., 2024). Agile practices may deliver little value when they are added to an organisation whose culture has not changed. The same division appears in the wider agile literature. Agile is studied either as a tool, focusing on practices and processes, or as a culture, focusing on organisational values (Magistretti & Trabucchi, 2025). This distinction is central to the present study. What matters is not only whether startups use agile methods, but how far they have internalised an agile mindset.

#	**Principle**
1	Delivering valuable output to the customer early and continuously is the highest priority
2	Changing requirements are welcomed, even late in the process, and used to the customer's advantage
3	Working results are delivered frequently, in cycles of weeks rather than months
4	Business and technical people work together on a daily basis
5	Projects are built around motivated people, who are given the environment, support and trust they need
6	Direct conversation is the most effective way of sharing information within a team
7	Progress is measured by what actually works, rather than by plans or documentation
8	Development proceeds at a pace that can be maintained indefinitely

9	Continuous attention to technical quality and good design increases the ability to change
10	Simplicity, understood as not doing unnecessary work, is essential
11	Architectures, requirements and designs emerge from teams that organise themselves
12	At regular intervals the team reflects on how it works and adjusts accordingly

Table 2.1. The twelve principles of the Agile Manifesto. Source: Beck et al. (2001).

2.2.1. Agile as a Tool

Doing agile is most visible in the practices and frameworks that teams use to organise their work. The best known is Scrum, which structures development into short iterations called sprints. Each sprint includes recurring events such as sprint planning, daily stand-ups, reviews, and retrospectives (Schwaber & Sutherland, 2020). Rather than specifying a product fully in advance, agile teams release small pieces of working output and use the feedback before moving on. Another key feature is the use of cross-functional teams. Members with different skills, such as design, engineering, and product management, work towards a shared goal instead of working in separate functional silos. Spotify is a well-known example. It reorganised around small, autonomous, cross-functional units known as squads to accelerate delivery and learning (Salameh & Bass, 2019). In digital startups in particular, these practices often appear alongside lean startup approaches, in which rapid build-measure-learn cycles are used to test and refine the business model under uncertainty (Ghezzi & Cavallo, 2020). Figure 2.10 illustrates how these events follow one another within a single sprint.

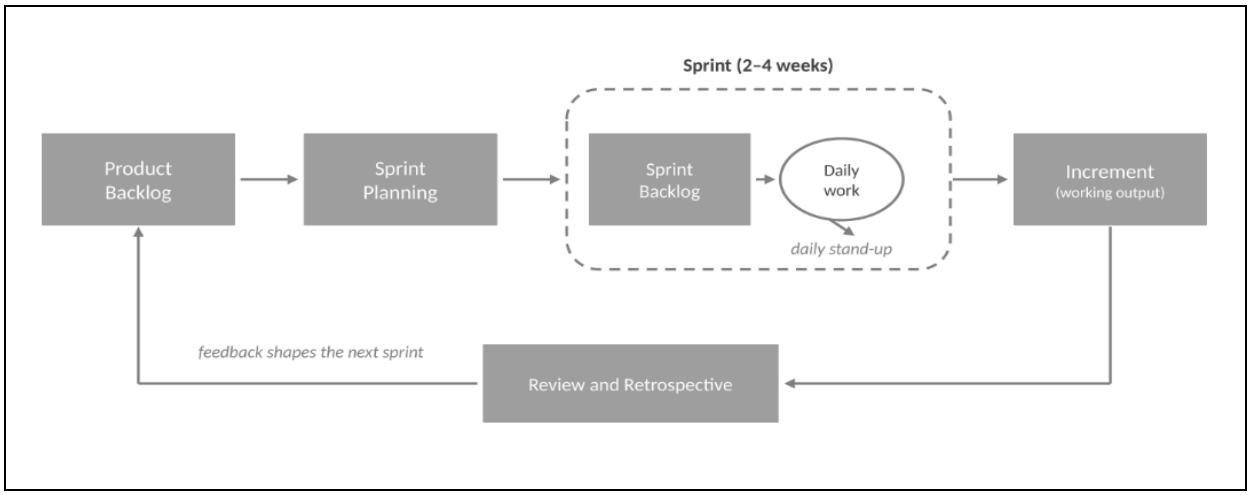


Figure 2.10. An illustration of the Scrum cycle. Source: author's elaboration based on Schwaber & Sutherland (2020).

These practices were developed around a single target user group. When feedback comes from more than one user group, this complexity increases, and the tools themselves are challenged: it becomes less clear whose feedback should guide the next iteration and how conflicting requests should be handled.

2.2.2. Agile as a Culture

While doing agile concerns visible practices, being agile concerns mindset, values, and culture. Being agile means that openness to change, collaboration, transparency, and continuous learning are internalised as shared assumptions rather than imposed as procedures. Neumann et al. (2024) suggest that the most persistent obstacles to agile transformation are cultural rather than technical. When an organisation adopts agile practices without the underlying values, the expected benefits fail to materialize. This explains a common pattern: firms that have formally gone agile continue to behave in hierarchical, control-oriented ways. They achieve the form of agility without its substance.

Understanding how agile practices translate into actual ways of working requires looking at organisational culture. Schein (2010) defines culture as a pattern of shared basic assumptions and distinguishes three layers through which it operates: visible artifacts, espoused values, and the underlying assumptions that ultimately guide behaviour. Hogan and Coote (2014) drew on this framework in an empirical study of innovation culture. They found that stating values alone does not improve firm performance. Instead, concrete artifacts and behavioural norms are what turn stated values into actual outcomes. Culture therefore acts as a bridge: practices alone do not produce results unless the organisation internalises the mindset behind them.

A culture clash occurs when new agile practices do not fit the existing corporate mindset. As Neumann et al. (2024) observe, layering agile ceremonies onto a traditional, control-oriented structure without reshaping its core assumptions creates friction that reduces flexibility. Overcoming this friction requires specific organisational attributes, including high psychological safety, a strong learning orientation, an explicit tolerance for safe-to-fail experimentation, transparency, and deep collaboration. When these values are genuinely embedded, teams are less afraid of failure and more willing to experiment and adjust on their own. Whether these values take hold depends on a number of conditions, discussed below.

A first condition is psychological safety, one of the most studied cultural conditions for agility. It is defined as a shared belief that the work environment is safe for interpersonal risk-taking,

allowing team members to express unconventional ideas, admit mistakes, and challenge the status quo without fear of retribution or marginalization. In settings where uncertainty is constant, this matters for the iterative process. Consistent with this, Buvik and Tkalich (2021) find that within agile development environments, higher levels of psychological safety are associated with greater team reflexivity and stronger problem-solving capabilities. When organisations actively cultivate this cultural environment, agile meetings become opportunities for learning rather than routine reporting.

A second condition is a learning orientation, combined with an explicit tolerance for experimentation. In agile settings, teams are expected to test small versions of a solution and learn from the outcome, which means that some attempts will not work. This only functions when unsuccessful attempts are treated as a source of learning rather than as failures to be avoided. In startups in particular, this logic is close to lean startup approaches, where rapid build-measure-learn cycles are used to test assumptions under uncertainty (Ghezzi & Cavallo, 2020). Where this orientation is missing, teams tend to avoid uncertainty and stay with what is already known, which limits the value of iterative practices.

A third condition concerns leadership. Leaders play an important role in creating these conditions, since agile values take hold only when they are visibly modelled and reinforced. Leadership in linear organisations has traditionally relied on command and control, which is designed to standardise outputs and minimize risk. As organisations become more adaptive, leaders have to unlearn many of these habits (Werner & Le-Brun, 2025). Rather than micromanaging tasks, agile leaders trust their teams and focus on removing friction, clarifying purpose, and creating the conditions in which others can work well (Eva et al., 2019; Uhl-Bien & Arena, 2018). Leaders can also undermine these conditions, for example by setting vague goals or rewarding predictability over exploration (Werner & Le-Brun, 2025). Bianchi et al. (2020) find that command-oriented control is associated with weaker project outcomes, while agile practices such as iterative sprints are associated with higher speed, better cost management, and better product quality. While Agile relies on these conditions, they differ across startups. As a result, implementing identical Agile practices often leads to different results in different startup contexts.

2.3. Research Gap: Agile Project Management and Digital Platform Creation

As discussed in the previous sections, digital platform creation is characterised by the continuous co-evolution of platform architecture, governance, and ecosystem relationships (Tiwana et al., 2010; Jovanovic et al., 2022). Platform studies have described these dimensions in detail, together with side configuration, network effects, platform boundaries, and scaling (de Reuver et al., 2018; Gawer, 2021; Springer et al., 2025). This literature explains what a platform is configured to do. It explains less how that configuration comes about: which decision was taken first, on what information, and how often it was revised before it settled. The literature accounts, for example, for why a marketplace may charge one side and subsidise the other (Eisenmann et al., 2006), but not for how a particular firm arrived at that arrangement, or what led it to change it later. This process is what the present study refers to as the platform project, and it is the aspect of platform creation that has received the least attention.

On the other hand, Agile literature is characterised by a concern with managing projects whose requirements keep changing (Beck et al., 2001; Rigby et al., 2016). It offers tools for working under uncertainty, such as short iterations, continuous feedback, and backlog prioritisation, together with a cultural perspective on the conditions under which these tools take hold (Magistretti & Trabucchi, 2025). Its purpose is to manage ever-changing projects more effectively. This literature, however, has developed largely within software development contexts, where a single customer group is assumed (Dingsøyr et al., 2012; Schwaber & Sutherland, 2020).

The conditions of platform creation described above closely match the principles of Agile Project Management, and Agile offers a way of managing the uncertainty that platform creation involves. Agile should therefore not be seen simply as a project management methodology adopted by platform firms. Its principles respond directly to the organisational requirements of platform creation. The incremental nature of platform development is reflected in iterative sprints and continuous releases. The collaborative nature of platforms aligns with customer collaboration, cross-functional teams, and continuous feedback. Agile also emphasises adaptability rather than detailed upfront planning, which fits the need to respond to changing ecosystem conditions (Schwaber & Sutherland, 2020; Neumann et al., 2024).

Despite this apparent fit, the two research streams have developed in parallel, and relatively limited attention has been paid to how Agile supports the incremental evolution of core platform

characteristics during platform creation. This is especially true for digital platform startups, where platforms are still emerging and organisational structures are still forming. Two questions in particular remain open. First, core Agile concepts such as customer collaboration, feedback loops, and backlog prioritisation implicitly assume a single customer group, whereas platforms involve at least two distinct sides whose interests may conflict. The Agile literature therefore offers little guidance on whose feedback should guide the next iteration when more than one side is involved. Second, Agile relies on feedback, but in the earliest stage of a platform the chicken-and-egg problem means that both sides are still missing, and there are few users to provide it. The platform literature, in turn, studies these sides in detail but pays little attention to the iteration process itself.

This thesis therefore investigates the intersection of these two streams. It examines the role that Agile practices and mindset play in the incremental creation of digital platforms, and how the characteristics of the platform in turn shape the way Agile is used. The study compares digital platform startups with different platform characteristics in order to identify similarities and differences in the ways of working that emerge during platform creation.

This study focuses on two platform characteristics: side configuration and stage of development. The first characteristic, side configuration, shapes whose feedback the platform responds to. As shown in Section 2.1.1, the sides of a platform are not equally easy to attract, and one side is often subsidised. As a result, side configuration determines which side receives more resources and attention. The second characteristic, stage of development, changes how agile practices work over time. Before the chicken-and-egg problem is solved, there is little user feedback to collect. Later, as architecture and governance lock in, key decisions become harder to reverse and the space for iteration narrows. Focusing on these two characteristics also reflects other platform dimensions. Network externalities relate closely to side configuration, while ecosystem maturity and the chicken-and-egg problem are tied to the stage of development.

These considerations lead to the following research questions:

RQ1: What role do Agile practices and mindset play in the creation of digital platform startups?

RQ2: How do platform characteristics, particularly side configuration and development stage, shape the use of Agile practices and the resulting platform changes?

3. Methodology

This study adopts a qualitative, exploratory research design. This is appropriate because the relationship between agile management and incremental platform creation in startups is context-dependent and not yet fully theorized (Eisenhardt, 1989). Rather than testing predefined hypotheses, the research seeks to understand how agile practices and the agile mindset shape the way digital platform startups build their platforms step by step. It also examines how platform characteristics relate to similarities and differences in the Agile ways of working that emerge in these startups.

The empirical work follows a multiple-case study design (Yin, 2018). Multiple cases allow patterns to be compared across settings, which makes the findings more robust than those drawn from a single case (Eisenhardt, 1989). The unit of analysis is the platform project: the design, the process behind the platform's creation, and the changes that continue up to and beyond launch. Because all three ventures are born-platforms, the platform project and the organisation largely coincide, since the startup was founded in order to build this platform and the decisions examined here are taken by the same small group that runs the company. Observations are therefore collected at the level of the startup, while the object of analysis remains the platform project rather than the firm's wider organisational development. Agile is examined from a managerial perspective, that is, in terms of how work is organised, how priorities are set, and how decisions are made, rather than from the perspective of software engineering.

3.1. Research Setting and Case Selection

The cases are digital platform startups based in Türkiye. Studying startups has two advantages for this research. First, agile and lean approaches are widely discussed in the context of digital entrepreneurship (Ghezzi & Cavallo, 2020), which makes it likely that Agile practices and mindset can be observed in this setting. Second, startups generally operate a single business model, whereas larger firms often manage a portfolio of several, which makes the phenomenon easier to isolate (Täuscher & Laudien, 2018).

The cases were selected according to three criteria. First, each company had to be a born-platform, that is, a venture founded around a platform business model rather than a firm that moved to a platform model later (Teece et al., 2022). This makes it possible to observe the

creation of the platform itself, while decisions about design, rules, and pricing are still open. In established firms, many of these decisions have already been made and are harder to change (Jovanovic et al., 2022). Second, each platform had to meet the conditions of a two-sided platform described in Section 2.1. Third, the cases were selected to cover different stages of platform development, so that ways of working could be compared across stages. Additional startups were contacted during the sampling phase. They were excluded because their business model was not platform-based.

Case A is an early-stage digital health startup whose platform generates personalised supplement recommendations for individual users, who access it through pharmacies and clinics. The pharmacies pay an annual licence fee, while the service is free for users.

Case B is a mature platform in the same sector, connecting individual clients with licensed mental health professionals. Clients pay per session, and the platform retains a share of this amount.

Case C is a peer-to-peer marketplace for secondhand fashion at an intermediate stage of development. Its two sides overlap, since the same person can act both as a buyer and as a seller.

Table 3.1 summarises the characteristics of the cases. The platform configuration and the way of working of each case are described in more detail in Chapter 4.

Case	**Sector**	**Sides**	**Stage of Development**	**Company Size (Internal Team)***	**Founded in**
Case A	Digital health / supplement guidance	Individual users / pharmacies and clinics	Early creation	4	2025 (idea validation 2024)
Case B	Digital health / online therapy sessions	Individual clients / licensed specialists	Mature	45–50	2019
Case C	Secondhand fashion	Individual buyers / individual sellers	Intermediate	61	2015

Table 3.1. Overview of the cases.

*Company size refers to the internal team of each startup and does not include the independent professionals or users active on the platform.

3.2. Data Collection

The research is based on primary and secondary sources that have been triangulated to increase the reliability and robustness of the findings (Miles & Huberman, 1984).

Secondary sources represented the first step of the data-gathering phase. The aim was to build knowledge of each startup before the interviews, so that the interview protocol could be adapted to each startup. Three types of secondary sources were considered: the institutional websites of the startups, press articles covering their development, and posts published on LinkedIn by the companies and their founders. These sources were used to reconstruct the timeline of each platform and to identify changes that could then be discussed during the interviews.

Primary sources were then used to understand from the inside how each startup manages the creation of its platform. Informants were selected on the following criteria: they are (i) founders or advisors directly involved in the development of the platform, in order to have a comprehensive overview of the decisions under analysis, and (ii) able to describe both the platform configuration and the way of working of the team.

A semi-structured interview protocol was designed, coherent with the exploratory nature of this research (Yin, 2018). The semi-structured format increased data completeness and allowed the conversation to follow the respondents' answers, giving proper relevance to what emerged during the interviews. The protocol is organised into core themes to support cross-case comparability. Two of these, side configuration and stage of development, derive from the platform framework established in Section 2.3. The remaining themes, iteration triggers, feedback prioritisation, and formal agile practices, draw from the agile literature reviewed in Section 2.2. The full protocol is provided in Appendix A.

Eight interviews took place between the end of May and the beginning of September 2026, lasting on average around 30 minutes each. With some informants, interviews were conducted in several rounds: initial conversations aimed to understand the business model and the platform structure, while subsequent interviews followed the protocol. All interviews were conducted in Turkish, the primary language of the participants, and relevant excerpts were translated into English. Where further clarification was needed, follow-up questions were addressed by email

and instant messaging. Not all conversations are directly quoted in the findings; however, they informed the overall understanding of each case. Table 3.2 summarises the data sources used in the study.

Case	Interviewed Person	# Interviews	Total Duration	Follow-up exchanges	Archival data
Case A	• Founder	3	≈ 1 h	Email, instant messaging	Website, LinkedIn posts
Case B	• External Consultant • Senior Product Designer	3	≈ 50 min	Email	Website, press articles, LinkedIn posts
Case C	• Founder	2	≈ 45 min	None	Website, LinkedIn posts

Table 3.2. Data collected for each case

3.3. Data Analysis

The coding was performed following the guidelines of Saldaña (2021). In a first cycle, relevant passages of the transcripts were labelled with descriptive codes that stayed close to the words of the informants, such as "deciding which side pays", "relying on the founders' experience", "separate interfaces for each side", or "working without formal routines". In a second cycle, these codes were grouped into sub-themes and then into the broader themes that structure the findings: maintaining the two-sided balance, triggers of iteration, handling conflicting needs across sides, and the form agility takes. Table 3.3 presents the resulting coding structure, with illustrative evidence from each case.

Theme	Sub-theme	Illustrative evidence
Maintaining the two-sided balance	Breaking the initial cycle	Known individual sellers brought in first to attract buyers (Case C); access points placed inside pharmacies so that one side brings the other (Case A); free introductory session offered to lower the barrier on the paying side (Case B)

	Revising the terms for each side	Commission moved from sellers to buyers to increase supply, several years after launch (Case C); pharmacies pay an annual licence while users access the service free of charge (Case A)
Triggers of iteration	Founders' knowledge as the benchmark	System logs and user scenarios reviewed against what a qualified professional would recommend (Case A); repeated adjustment of the recommendation algorithm without external requests (Case A)
	Behavioural data as the trigger	Drop-off rates and A/B tests used to locate problems without asking users directly (Case B); usage data rather than direct requests guiding what is built next (Case C)
Handling conflicting needs across sides	Separating the sides in the interface	Pharmacists see detailed scientific information while individual users receive a simple summary (Case A)
	Rules that satisfy both sides at once	Payment held by the platform and released to the seller only after the buyer confirms receipt (Case C)
	Prioritising without reference to side	Priority set by the number of users requesting a feature and the effort required, with brand identity as a final filter (Case B)
The form agility takes	Iterating without formal practices	No sprints, stand-ups, retrospectives or structured backlog; logs reviewed daily and coordination through continuous conversation (Case A)
	Adopting formal practices as the team grows	Two-week sprints and regular review meetings (Cases B and C); additional decision-making layers introduced as the team expanded (Case B)
	Narrowing scope of change	Whole features and interfaces added and removed in the early stage (Case A); changes down to the colour of a button (Case B)

Table 3.3. Data structure of the coding process

Since the coding was performed by a single researcher, the codes were reviewed a second time after an interval, and coded passages were compared with secondary sources to check consistency.

The information was then organised in a matrix (Miles & Huberman, 1984), allowing the results to be compared in two different ways: first within each case, in order to understand the way of working in its own context, and then across cases, in order to identify which aspects were common to all startups and which differed according to their platform characteristics (Eisenhardt, 1989). The result of this cross-case comparison is presented in Section 4.4.

4. Findings

4.1. Case A

Case A uses an AI-based clinical decision engine that analyses users' lifestyle data and drug interactions to generate personalised supplement routines. The system is positioned directly in pharmacies and clinics, where users access it through on-site QR codes. The venture is based in Türkiye, with initial idea validation in late 2024 and formal incorporation in 2025, and operates with a core team of four. At the time of the interview, the platform had approximately 2,000 users.

4.1.1. Platform Configuration

The platform connects two sides: individual users and pharmacies or clinics. The founder described the revenue model as follows:

"The revenue source of our business model is actually pharmacies and clinics. We do not take any payment from individual consumers."

While the B2C side offers a premium subscription option, it is intentionally set at a high price point to deter direct purchases. Users are deliberately steered towards pharmacies, where the same access is provided free of charge. The founder summarised her role as follows:

"I sell an annual licence to the pharmacy. That is what I do."

What pharmacies buy is access to the other side of the platform. Each licence includes a monthly quota of 100 premium user memberships. Pharmacies are connected to the platform through QR codes, and the pharmacy dashboard shows when a customer's supplement supply is running low:

"The pharmacy panel shows that the user's supplement is running low. Through the app notifications, you can make sure she buys again. It increases direct sales, it makes them scientifically grounded, and it also increases repeat sales."

The two sides are connected to one another through the platform, rather than each using it as a separate tool. Figure 4.1 summarises this configuration.

However, the cross-side network effects are not yet symmetrical. While user base growth directly increases value for pharmacies, an increase in the number of participating pharmacies offers limited additional utility to existing individual users.

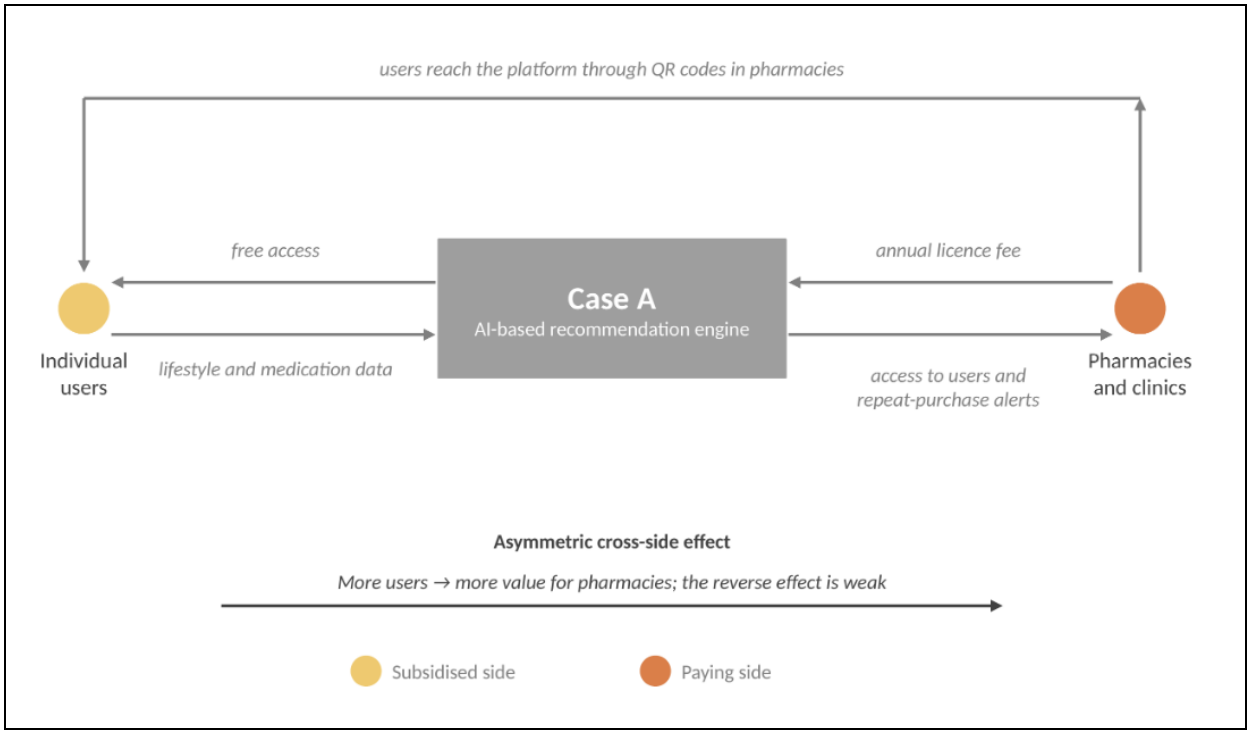


Figure 4.1. Platform configuration of Case A. Source: author's elaboration based on interview data.

The startup is also working to add new sides to the platform. At the time of the interview, it was in discussion with an insurance company, and its website lists gyms, dietitian clinics, and wellness centres alongside pharmacies.

4.1.2. Platform Development

Development began by mapping out clinical decision-making rather than by writing code. To establish the foundational logic, the founder first manually structured a list of questions based on active ingredients:

"First we wrote down the questions: when I give advice at the pharmacy, which questions do I ask? Then we said, if someone answers these questions, they can build a routine that is safe and specific to them."

Once this decision tree was defined, the initial Minimum Viable Product (MVP) was assembled by integrating ready-made tools. The technical architecture connected a custom database of 100

drug interactions with an API searching scientific articles, passing this data through a low-code AI integration platform connected to a lightweight large language model configured to act as a pharmacy assistant. This architecture was embedded into a website form, allowing the venture to validate its core recommendation mechanism after its initial concept validation in late 2024.

Moving from a basic functional model to a live environment revealed unexpected complexity. The founder initially anticipated a straightforward recommendation logic, based on identifying the user's primary goal, listing suitable supplements, checking for any conflicting medications or underlying conditions, and recommending the products if no contraindications existed. Dietary supplements, however, belong to preventive health rather than acute care. Consequently, the algorithm had to include lifestyle variables such as alcohol consumption, which significantly alter safe dosage and product efficacy:

"The weight-loss supplement you can give to someone who drinks alcohol regularly is very different from the one you can give to someone who never drinks. The dosage changes a lot as well."

The number of variables that had to be considered at the same time increased further as the system was used in practice:

"You write into the clinical decision system: this person had a heart attack last year, is now using a blood pressure medication, has high cholesterol, and wants a supplement for stress. What should I give?"

This grew into a complex scoring and multiplier system. As the founder put it:

"The whole thing turned into an incredibly complex algorithm. It was much harder to work through than we expected."

To manage this complexity, the venture entered a phase of continuous iteration. Recommendation coefficients and scoring prompts are evaluated daily by reviewing system logs against clinical reasoning. There have been around 8,000 updates to the core algorithm, and R&D is expected to continue through the end of 2027. The founder described the process as follows:

"I look at it with a health professional's eye and think we could have seen a better output. Then we look at the logs to see how it scored, and we discuss whether a different prompt command would give better scoring."

Feedback reaches the startup from both sides of the platform. At this early stage of the platform's creation, however, this feedback does not take the form of specific requests for features or algorithmic changes. The more detailed work of deciding what to adjust is carried out by the founder, who reviews user scenarios and system logs against her own pharmaceutical knowledge.

4.1.3. Ways of Working

Communication within the main team is frequent and flexible. The founder described talking to the developer on a daily basis, maintaining weekly meetings with the co-founder, and holding broader team syncs every two weeks. Outside these set meetings, most daily coordination happens through an active group chat:

"I talk to the developer every day. I have weekly meetings with the co-founder. We have meetings every two weeks. But our group chat is incredibly active in any case."

Despite this constant communication, formal Agile routines are largely absent from the team's practice. The founder did not mention sprints, daily stand-ups, retrospectives, or structured backlogs at any point in the interview. Tasks and priorities are handled informally, as issues or log results come up, rather than through scheduled project management routines. At the same time, the work itself is highly iterative, with around 8,000 updates to the algorithm and daily reviews of system logs.

In a follow-up exchange, the founder was asked what happens when requests from the two sides conflict. She explained that this situation has not come up so far, and pointed to the two different interfaces as an example. The two sides use the recommendation for different purposes. A pharmacist advises a customer in person and has to be able to stand behind that advice, so the interface shows the scientific reasoning behind each suggestion, including the ingredients involved and the basis on which they were matched. An individual user is deciding what to take, not justifying a recommendation to someone else, and receives a short summary instead. Because the two views are built as separate parts of the product rather than as one screen

showing more or less information, a request from one side does not remove anything from the other. A pharmacist asking for additional detail does not make the user's view more complex, and a user asking for a simpler summary does not reduce what the pharmacist can see. Most differences between the two sides are therefore resolved in the design of the platform before they reach the point of a decision. If such a conflict were to arise, both requests would be reviewed, but the founder expects prioritisation to favour the paying side, the pharmacies.

4.2. Case B

Case B is a Türkiye-based online mental health platform that connects individual clients seeking therapy with licensed mental health professionals. According to company disclosures, the platform serves over 1 million registered users and includes a network of more than 2,000 licensed specialists.

4.2.1. Platform Configuration

The platform connects two sides: individual clients seeking psychological support and licensed specialists, including psychologists, psychiatrists, and dietitians. In contrast to Case A, the paying side here is the individual client. Clients pay per therapy session, whereas specialists join the platform to access a continuous stream of clients and to use the platform's operational tools.

One distinguishing feature is the platform's active matching mechanism. According to company disclosures, the system connects the two sides rather than simply listing available specialists. Upon registration, clients complete a short questionnaire about their needs. An algorithmic matching process then recommends the most suitable specialist. To lower the barrier for new clients, the platform offers a free 15-minute introductory video session. If the match does not work, the client can be matched with another specialist.

Specialists receive more than access to clients. The platform also provides tools for appointment scheduling, automated billing, secure video conferencing, and client messaging within a single panel. Rather than letting specialists set their own fees, the platform applies the same session price across the network. Figure 4.2 summarises this configuration.

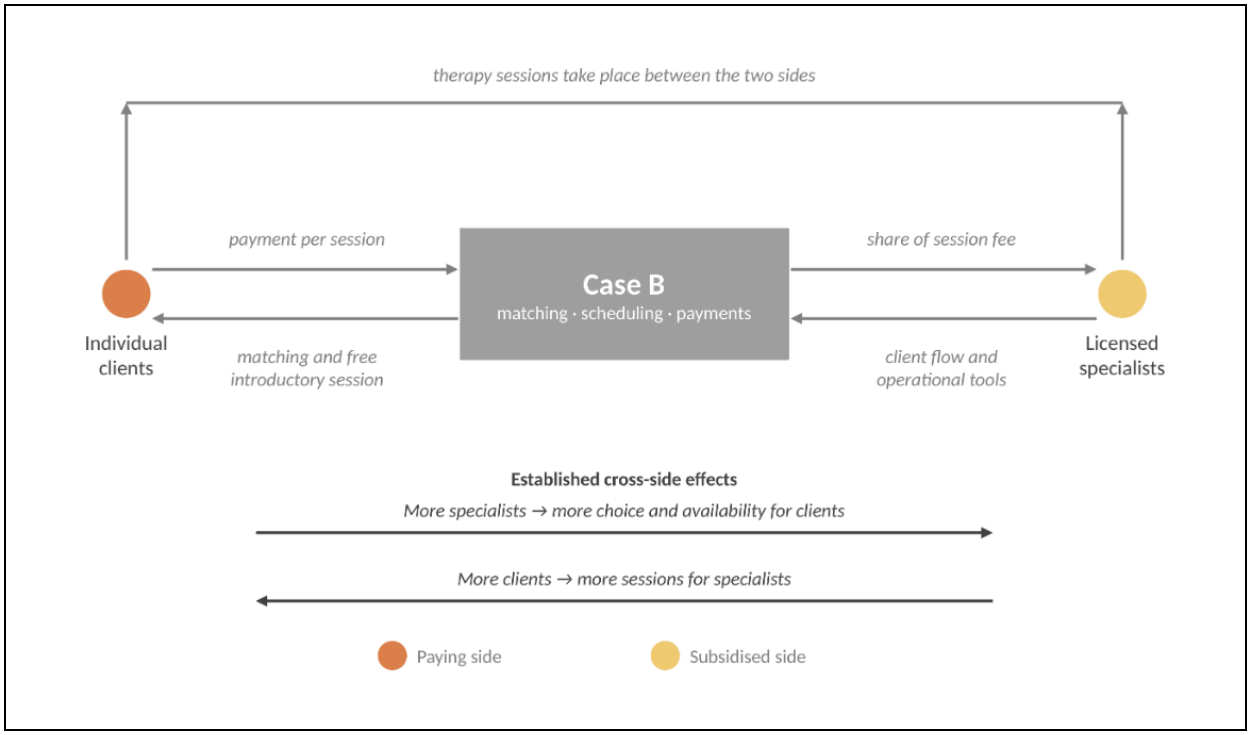


Figure 4.2. Platform configuration of Case B. Source: author's elaboration based on interview and secondary data.

The configuration of sides has changed over time. While the service initially connected clients only with clinical psychologists, the ecosystem was subsequently expanded to include psychiatrists and dietitians. The interviewee described the reason for this expansion:

"Someone receiving therapy who also needed medication had to go to a psychiatrist outside the platform. Significant demand built up in this direction, and now clients can also see psychiatrists."

Both sides are now established. A large number of specialists is available to clients seeking immediate availability and diverse expertise, while a steady flow of clients makes the platform attractive to specialists.

4.2.2. Platform Development

The findings for this section draw on an interview with an external consultant who worked directly with the platform. Development decisions in Case B rely primarily on user behaviour data. Rather than asking users what they want, the team identifies problems from how users interact with the application:

"You look at the data in the background. If the drop-off rate on a certain page is high, there is a problem there; you do not need to ask the user. For example, even the colour of the buy button is tested."

The interviewee described data-driven decision-making, including A/B testing, as the standard approach for feature optimisation:

"In an established application, data speaks in approximately 90% of decisions."

This figure is an estimate rather than a measurement. The interviewee also noted that behavioural data can reveal needs that users themselves would not articulate:

"The data tells you about behaviours the user might not be able to express, or might not even be aware of."

Direct feedback channels also play a role. User emails and in-app complaint or suggestion buttons are reviewed regularly, but they are secondary to behavioural data.

Features are prioritised along two axes: demand volume and development effort. Proposed updates are evaluated according to "the intensity of user demand and the effort required to implement it", and their position in this matrix determines what gets built. Beyond these two criteria, proposed updates must also fit the platform's brand identity:

"There is one more layer: if the feature does not conflict with the brand identity, it goes onto the to-do list."

The addition of psychiatric services is a concrete example. Clients undergoing therapy previously had to consult external medical professionals for prescription management. Once regulatory changes permitted online psychiatric consultations, the platform responded to accumulated user demand by onboarding psychiatrists as an additional professional user group.

Updates are released continuously. Security updates are deployed as needed, application fixes are released rapidly, and new features are rolled out as development is completed.

4.2.3. Ways of Working

Case B uses formal Agile development practices. Product development is organised around structured two-week sprint cycles guided by the Scrum methodology, supported by regular sprint meetings:

"Running two-week sprints. Scrum methodology is used."

Despite this formal framework, the team can still work flexibly. When urgent technical issues or unforeseen requirements emerge, teams meet outside the sprint schedule to reorganise the work:

"For urgent developments, teams can also come together outside the regular schedule and plan."

The type of changes has shifted as the platform matured. During the platform's early phase, iterations involved larger functional changes, such as launching or removing entire feature sets and user interfaces. In the mature phase, changes are smaller:

"While in the early stage it was possible to add or remove a feature, in the mature stage small changes occur, such as changing the colour of a button."

The interviewee added that the speed of adaptation itself has not decreased:

"We were fast in the early period, and we can still adapt quickly now."

The decision-making structure has also changed as the team grew. While the startup can still respond quickly to market demands, team expansion has introduced formal executive layers:

"As the team grew, C-level executives began to form, leading to an increase in decision-making authorities."

4.3. Case C

Case C is a Türkiye-based secondhand fashion platform that enables individual users to buy and sell pre-owned clothing and accessories. The company was founded in 2015 and works

with an internal team of 61 people. According to public disclosures, the platform features over 1 million listed items.

4.3.1. Platform Configuration

The platform operates on a peer-to-peer model where both sides consist of individual consumers. Unlike Case A and Case B, where the platform sides represent distinct categories of user groups, such as pharmacies versus individual users or specialists versus clients, the two sides here overlap, as the same user can act as both a buyer and a seller. The founder described this as the defining feature of the platform:

"The most interesting situation for us is that the same user can be both a buyer and a seller on the platform. So when we develop a feature, we do not focus on a single audience but on this intertwined two-sided experience."

The revenue model has changed since the launch of the platform. The platform initially charged a commission fee to sellers. Over time, the seller commission was removed and a service fee was applied to buyers instead:

"When we first set up the system, we charged a commission from sellers like a classic marketplace. But we saw that what would really grow the platform was increasing the supply of products, so we decided to give sellers more incentive. We removed the seller commission completely and instead applied a small service fee on the buyer side."

In terms of governance, the platform controls the payment process. Payments are held by the platform in what the founder describes as a pooled payment system, and released to the seller only after the buyer receives and confirms the item. Because both participants are individual users without institutional credentials, this mechanism serves as a key governance tool to protect both parties. Figure 4.3 summarises this configuration.

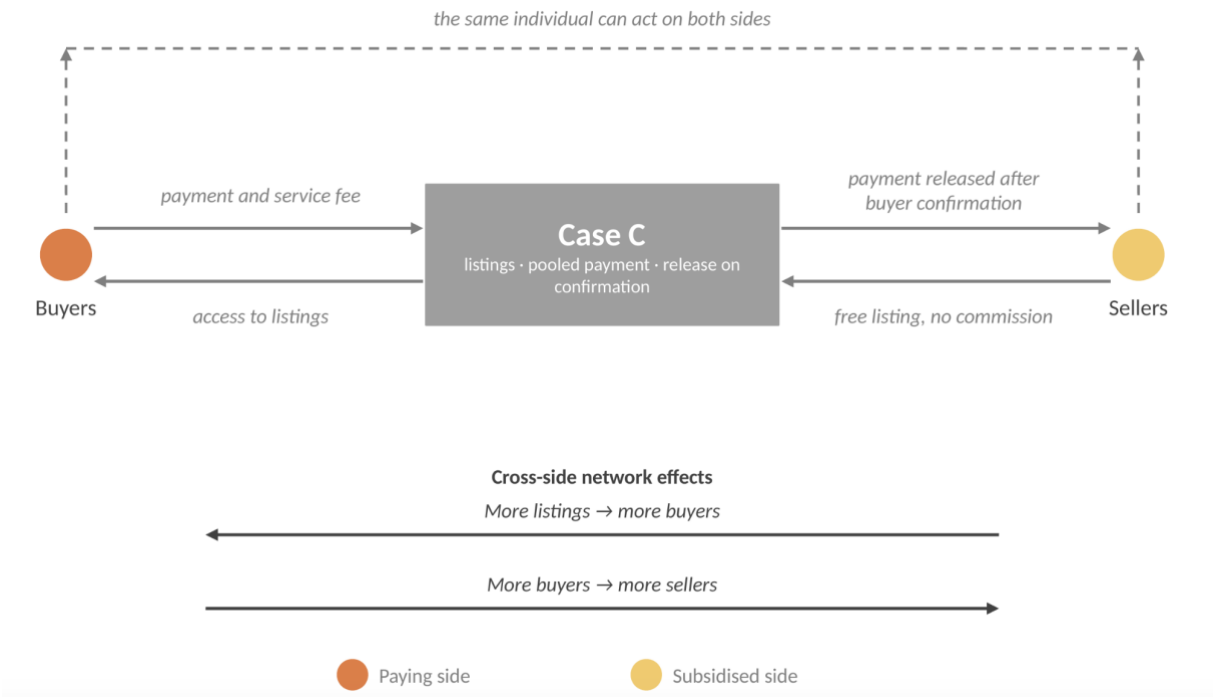


Figure 4.3. Platform configuration of Case C. Source: author's elaboration based on interview and secondary data.

The platform currently lists over 1 million items, and network effects operate in both directions:

"Every new product uploaded to the platform attracts more buyers, and the growing crowd of buyers attracts new sellers."

4.3.2. Platform Development

Development decisions in Case C are based on reading data and user behaviour. The founder described this as a continuous practice since the company was founded in 2015:

"Since we were founded in 2015, we have continuously tried to read the data and user movements."

In the early stage, the platform faced the difficulty of attracting both sides at the same time:

"In the beginning we experienced the classic nightmare of all marketplaces, the chicken-and-egg problem. Buyers did not come because there were not enough products on the platform, and nobody opened their wardrobe and uploaded products because there were no buyers."

To break this cycle, the startup worked with influencers who opened their accounts on the platform. This addressed two problems at once, attracting users and establishing trust in a setting where both sides are individuals:

"By opening the wardrobes of well-known and trusted influencers for sale on the platform, we created that first big spark. When people saw names they liked and trusted on the platform, the barrier came down: trust was created and we started to attract significant user traffic."

The decision to remove the seller commission followed a similar logic: it was taken to increase the supply of products rather than in response to a specific request from either side.

The nature of development work has changed since then. The founder described a shift from the user acquisition tactics of the early days towards finer adjustments aimed at making the existing system run smoothly.

4.3.3. Ways of Working

The team works in sprints, with regular meetings in which the work is reviewed:

"Our teams run sprints and we regularly put on the table what we could do better."

Conflicts between the two sides are a recurring issue rather than an exception. Because both sides are individual consumers without institutional credentials, the central difficulty is trust:

"The buyer is afraid of sending the money and not receiving the product, while the seller is rightly worried about shipping the product and not getting paid."

These two competing concerns were addressed through the payment mechanism described in Section 4.3.1:

"To solve these two conflicting and legitimate demands at the same time, we placed a governance mechanism in between: the pooled payment system."

The overlap between the two roles also shapes how new features are evaluated. The founder described the question the teams ask for every proposal:

"Does this new feature make the user happy when wearing the seller hat, while creating discomfort when wearing the buyer hat?"

Development work is therefore organised around keeping this balance, rather than prioritising one side over the other.

4.4. Cross-Case Analysis

Table 4.1 brings together the three cases along the dimensions used to structure the findings, and the patterns it shows are discussed below.

	Case A	**Case B**	**Case C**
Platform Characteristics			
Side configuration	Two sides, users and pharmacies. Pharmacy side pays.	Two sides, clients and specialists. Client side pays.	Two sides, buyers and sellers, overlapping. Buyer side pays, changed from seller side.
Cross-side effects	Asymmetric, mainly one direction	Established in both directions	Established in both directions
Stage of development	Early, configuration still expanding	Mature, chicken-and-egg resolved	Intermediate, initial chicken-and-egg problem overcome
Agile Ways of Working			
Trigger of iteration	Founder's clinical review of user scenarios and system logs	Behavioural data and A/B testing, supported by emails and in-app suggestions	Data and user behaviour
Feedback entering backlog	No conflicts reported; separate interfaces for each side; founder expects to prioritise the paying side	Demand volume × effort matrix, filtered by brand identity	Every feature assessed for both roles; pooled payment system resolves the underlying conflict
Formal Agile practices	No formal ceremonies, informal daily contact	Two-week sprints, Scrum	Sprints and regular review meetings

Table 4.1. Cross-case comparison

The cross-case comparison points to differences across the platform characteristics. Regarding side configuration, the three cases represent different structural setups: Case A connects an

institutional paying side with non-paying individual end users; Case B connects paying individual clients with professional specialists who use the platform as an operational tool; and Case C features individual consumers on both sides of a peer-to-peer model, where the same person can act in both roles. The paying side therefore differs across the cases, and only in Case C did it change over time, moving from sellers to buyers.

Cross-side network effects and development stages follow a similar pattern. In Case A, cross-side effects remain asymmetric and one-directional, matching its early creation phase. In Cases B and C, cross-side effects are established in both directions, with Case C at an intermediate stage and Case B at a mature stage. Together, the three cases form a progression across different stages of platform development.

The Agile ways of working differ along the same progression. What triggers iteration changes across the three cases: in Case A, where neither side yet provides structured feedback, iteration is driven by the founder's own professional judgement; in Cases B and C, where both sides are established, iteration is driven by user behaviour data. The use of formal practices follows the same order. Case A works iteratively without sprints, stand-ups, or structured backlogs, while Cases B and C both organise development around sprints and regular review meetings.

One pattern is common to all three cases. None of the startups resolves conflicting needs by ranking the two sides against each other. In Case A, the conflict is addressed through the design of the platform, since separate interfaces keep the two sides apart. In Case C, where the two roles overlap and cannot be separated in this way, it is addressed through a governance mechanism instead, since the pooled payment system answers the concerns of both sides at once. In Case B, priority is determined by demand volume and development effort rather than by which side made the request. In Case C, this constraint also extends to ordinary feature decisions, since every proposal has to be assessed for both roles at the same time.

5. Discussion

5.1. Overview

This chapter brings the findings together and relates them to the literature reviewed in Chapter 2. Rather than discussing Agile and platform development separately, the discussion is organised around three mechanisms that emerged across the cases. Each mechanism links a characteristic of the platform to a consequence for the way Agile is used, and each traces what that use of Agile leaves behind in the platform itself.

The first mechanism concerns the continuing balance between the two sides and the form that Agile takes while that balance is still being settled. The second concerns the point at which both sides become active and behavioural data becomes interpretable, and what guides iteration before that point. The third concerns the separability of the two sides and the limits of the backlog as a device for resolving conflicting demands.

Each section starts with the platform characteristic involved, since this is something the startup does not choose. It then looks at what that characteristic does to the way Agile is used. Finally, it looks at what this use of Agile leaves behind in the platform. Each section also specifies what drives the mechanism, at what point it changes, and why it changes at that point. Taken in the first direction, from the platform to Agile, the three sections address RQ2. Taken in the second, from Agile back to the platform, they address RQ1. The two research questions are therefore not treated in separate sections but at the two ends of the same three mechanisms. Section 5.2 restates these mechanisms as propositions that can be tested on a larger sample.

5.1.1. The two-sided balance and the form Agile takes

The literature treats the chicken-and-egg problem as a launch problem. Neither side joins before the other, so the platform has to find a way to attract both at once (Caillaud & Jullien, 2003; Trabucchi, 2020). The choice of which side to subsidise and which side to charge is presented as a design decision taken at the outset (Eisenmann et al., 2006; Rochet & Tirole, 2003), and once critical mass is reached, the problem is considered solved (Evans, 2003). The literature on Agile approaches the same period from the opposite direction. It distinguishes between Agile as a tool, meaning practices and ceremonies, and Agile as a culture, meaning the values and

mindset behind them, and observes that adoption is usually measured through the practices (Magistretti & Trabucchi, 2025). Studies of Agile adoption tend to explain the gap between the two in organisational terms, as a matter of maturity, leadership, or cultural resistance (Dingsøyr et al., 2012; Neumann et al., 2024; Schein, 2010). Neither literature asks whether the state of the platform itself has anything to do with which of the two forms of Agile is available to the team.

The cases show different ways of breaking the initial cycle. One platform brought in trusted individual sellers whose presence attracted buyers, which corresponds to one of the tactics Trabucchi (2020) describes. What the case adds is that the same move solved a second problem: where both sides are private individuals, the barrier was not only supply but trust, so the identity of the participants mattered as much as their number. Another platform placed its access points inside the institutional side's premises, so that one side physically brings in the other. A third lowered the entry barrier on the paying side by offering a free introductory session, converting a pricing decision into an acquisition mechanism.

What these moves have in common is that they did not stop once the initial cycle was broken. In all three platforms, decisions about how to treat each side were revised as the platform grew: which side is charged, which side is given free access, and which side receives additional tools. One platform removed the commission it charged sellers and moved the fee to buyers in order to increase the supply of products, several years after launch and well past the point at which critical mass could be said to have been reached. Keeping both sides on board therefore looks less like a threshold that is crossed and more like a balance that has to be maintained. This is closer to the lifecycle view of platform business models advanced by Muzellec et al. (2015), and to Gawer's (2021) treatment of platform boundaries as decisions that managers revise over time, than to the idea of a launch problem that is solved once and then left behind.

This has a direct consequence for the form Agile can take. A decision about which side pays cannot be divided into increments. It cannot be released to part of the user base, since the sides are joined by cross-side effects and a change on one side is transmitted to the other; and it cannot be evaluated on one side alone, for the same reason. It is, in other words, not the kind of work a backlog item is made of. Agile practices presuppose that work can be decomposed into units that are independently estimable, independently releasable, and independently assessable (Schwaber & Sutherland, 2020). Decisions about the balance between the sides satisfy none of

these conditions. While such decisions remain open, the venture's adaptiveness lies outside formal Agile frameworks. Instead, it stems from the founders' working style: daily conversations, continuous revision, and a willingness to reverse initial commitments.

The earliest-stage platform illustrates this directly. It used none of the formal practices. There were no sprints, no stand-ups, no retrospectives, and no structured backlog. Yet the work was highly iterative: the core algorithm had been updated thousands of times and logs were reviewed daily, with coordination happening through continuous conversation and an active group chat rather than through scheduled ceremonies. By any behavioural measure, this is an Agile organisation; by the measure usually applied in adoption studies, it would not register as one at all. The two more established platforms use two-week sprints and regular review meetings alongside the same iterative logic, which indicates that the practices were added to an existing mindset rather than introduced in place of one.

The practices appear later than the mindset, and they appear as the team grows. Four people in daily contact had no need for ceremonies, whereas teams of around fifty and sixty do, and in one of them growth also introduced executive layers and additional decision-makers. This is consistent with accounts of scaling Agile, in which coordination mechanisms are introduced to substitute for the shared context that small co-located teams hold informally (Rigby et al., 2016; Salameh & Bass, 2019). The cases suggest that this substitution is not only a response to headcount. It is also a response to the work having become divisible: once the terms on which each side participates are stable, what remains is feature work, and feature work is what a backlog is designed to hold.

The scope of change narrows in the same direction. In the early stage, whole features and interfaces were added and removed. In the mature stage, changes are smaller, down to the colour of a button. This is consistent with the lock-in argument: as users build history and routines on the platform, altering it becomes more costly (Shapiro & Varian, 1999; Tiwana et al., 2010). What the cases add is that the room for iteration narrows precisely when the platform starts working well. Formal practices are therefore adopted at the point when the team most needs coordination and least needs the freedom to make large changes.

The effect that returns to the platform is the counterpart of this. Because agility in the early stage is carried by the mindset of a small team rather than by a formal process, the founding terms of the platform remain revisable for much longer than the launch-oriented literature

suggests. Pricing, subsidy, and access are treated as open questions rather than as settled parameters, and they are reopened when the balance between the sides shifts. The commission that was moved from sellers to buyers is an instance of this: a boundary decision of exactly the kind Gawer (2021) describes, taken years after launch, and taken outside any sprint. The cases therefore specify a condition under which such revision remains available: it remains available while the organisation is still small enough for agility to operate as culture rather than as process.

The mechanism can be stated compactly. The driving force is the divisibility of the work, which is itself a function of the platform's development stage. The change occurs when the terms on which each side participates become stable enough that the remaining work can be decomposed into independent units. It occurs at that point, and not earlier, because Agile practices are instruments for ordering decomposable work, and cannot hold decisions whose effects cross both sides of the platform at once. Figure 5.1 summarises how these elements differ across the three stages represented in the sample.

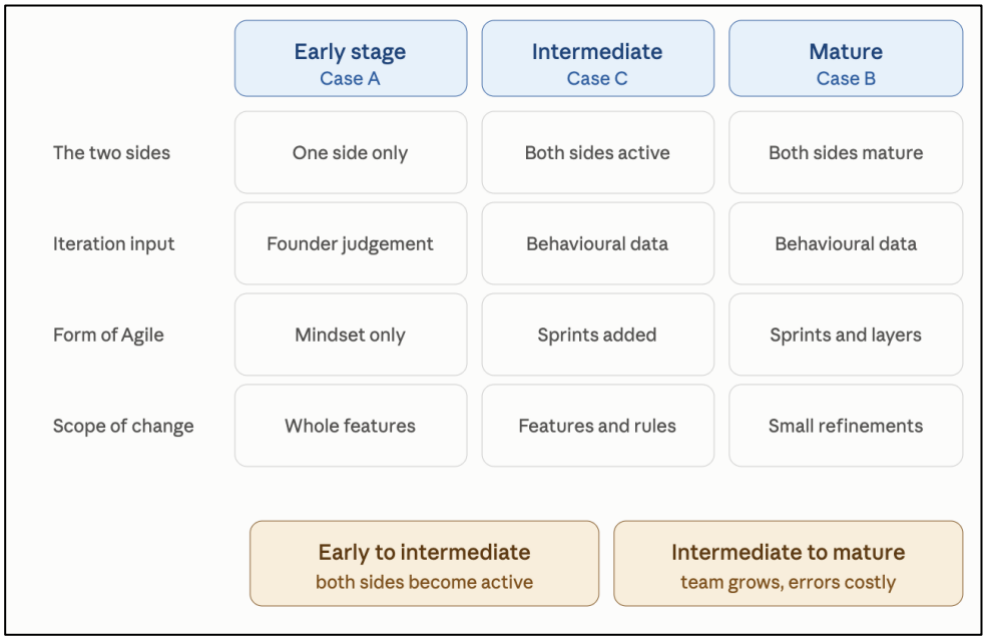


Figure 5.1. Stage characteristics across the three cases, and the two transitions observed between them. Source: author's elaboration based on interview data.

This mechanism contributes to both literatures. For the platform literature, it suggests that the subsidy decision described by Eisenmann et al. (2006) is not a single design choice taken at launch but one that is reopened as the balance between the sides shifts, which means that reaching critical mass ends the acute form of the chicken-and-egg problem without ending the

problem itself. For the Agile literature, it suggests that the distinction between Agile-as-a-tool and Agile-as-a-culture (Magistretti & Trabucchi, 2025) is not only a matter of organisational maturity or preference, but is conditioned by the nature of the work to be done. Where the central decisions are indivisible, only the cultural form is available, and the absence of formal practices should not be read as the absence of agility.

5.1.2. The second side and what drives iteration

Agile assumes that iteration is guided by feedback from the customer (Beck et al., 2001; Schwaber & Sutherland, 2020). The lean startup approaches that have been adapted to digital entrepreneurship extend the same assumption, treating the business model as a set of hypotheses to be tested against market response and revised accordingly (Ghezzi & Cavallo, 2020). On a platform this assumption is harder to satisfy, because feedback has to come from two sides, and because each side's response is conditioned by the presence of the other (Rochet & Tirole, 2003; Rysman, 2009). What the customer does is, on a platform, partly a function of what the other side is doing. The unit that Agile treats as a signal is, in this setting, a composite.

The cases show that what drives iteration changes with the stage of the platform. In the earliest-stage platform, neither side provided structured feedback. The user base was small, and requests for features did not come from either side. Iteration was driven instead by the founder's own professional knowledge: user scenarios and system logs were reviewed against what a qualified professional would have recommended, and the algorithm was adjusted accordingly. The standard against which the output was judged was therefore external to the platform and resided in the founder's training. In the two more established platforms, iteration is driven by what users actually do. Drop-off rates, A/B tests, and usage data indicate where the problem is without asking users directly, and the standard against which the output is judged has moved inside the platform.

The mechanism behind this shift is not the age of the venture but the completeness of the platform's sides. In Gawer's (2021) terms, the stage of a platform is a statement about where its boundaries currently stand: how many of its intended sides are actually inside them, and which digital interfaces exist to mediate the sides that are. This matters for iteration because behavioural data is produced at the interface, so what a team is able to observe is set by the platform's configuration at that moment. Behavioural data is only interpretable when both sides are active, because a weak signal on one side may originate in the state of the other. A drop-off

among buyers may mean that the interface is confusing, or it may mean that there is not yet enough to buy. The same number carries both readings, and nothing internal to the data distinguishes them. Until the second side is populated, the team cannot tell whether it is observing a product problem or an emptiness problem, and an iteration cycle run on such data would optimise the wrong thing with full statistical confidence. The founder's domain knowledge supplies the missing benchmark: it provides an external standard of what the output should have been, in place of a comparison the data cannot yet make. Once the second side is brought inside the boundary and an interface begins to record it, the ambiguity resolves, behaviour becomes self-interpreting, and the external standard can be retired.

Two further pressures push in the same direction, and they help explain why the shift is not reversed once it has occurred. As users accumulate history on the platform, the cost of acting on a misread signal rises, which makes evidence more valuable than judgement (Shapiro & Varian, 1999; Tiwana et al., 2010). At the same time, the team grows. The founder's professional knowledge is difficult to put into words and cannot be shared across dozens of people, whereas a drop-off rate can be read by anyone. In one of the mature cases, the interviewee described decisions as being settled overwhelmingly by data. The constraint therefore changes as the venture grows: at first the problem is what the team is able to know, and later it is what the team is able to pass on to others.

What returns to the platform is significant, and it is the part of this mechanism that the existing literature does not address. Because the assumptions cannot be tested until both sides are present, the platform is built on them in the meantime, and the decisions taken under them are not neutral. They are written into the interface, into the matching logic, and into the rules governing each side, and by the time evidence becomes available they are no longer cheap to revise, for the same lock-in reasons discussed in Section 5.1.1. The founder's professional judgement is therefore not merely a temporary substitute for data; for a period, it is the design. This gives the composition of founding teams in platform ventures a weight that is not usually attributed to it: in the interval before the second side arrives, the platform encodes whatever the founders happen to know.

This also shows a limit to the trial-and-error account of platform entrepreneurship. Ojala and Lyytinen (2022) argue that entrepreneurs cannot know in advance which services will work while markets are still immature, and that they therefore proceed by trial and error. The cases

support the first part of this but qualify the second. On a two-sided platform the result of a trial cannot be read reliably until the second side exists, so the team learns from its errors later than a single-sided venture would. Platform ventures therefore rely on assumptions for longer, not because their founders are less disciplined or less interested in data, but because their feedback is harder to interpret. Ghezzi and Cavallo (2020) describe lean approaches as a way of testing business model hypotheses quickly; the cases suggest that on a platform the speed of testing depends on a condition the venture does not control.

The mechanism can be stated briefly. What drives it is the completeness of the platform's sides, not the age of the venture or the size of the team. The change happens when the second side becomes active in large enough numbers for behaviour on one side to be explained by that side alone. It happens at that point because before it, the two possible readings of the same figure cannot be separated, however carefully the data is collected. Because a side is either present or not, the change is a threshold rather than a gradual improvement. The three cases cannot separate side completeness from team size and venture age, since here they move together, so the distinction between this mechanism and the one in Section 5.1.1 is analytical and would need a more varied sample to test. Figure 5.1 shows how the input to iteration differs across the three cases.

5.1.3. Conflicting sides and the limits of the backlog

Agile tools were built for a setting with one customer group. In Scrum, a single Product Owner orders one Product Backlog on behalf of all stakeholders, and this arrangement assumes that one ordering can represent all the interests involved (Schwaber & Sutherland, 2020). The platform literature, meanwhile, describes the two sides as distinct groups with different interests, held together by cross-side network effects and often facing different prices and different rules (Eisenmann et al., 2006; Gawer, 2021; Van Alstyne et al., 2016). Read together, these two statements produce a problem that neither literature raises on its own: the usual instrument for resolving competing demands assumes one group of users, while the product it is applied to has two, and the two cannot stand in for each other.

None of the three platforms resolves this by ranking the two sides against each other inside the backlog. In one, the two sides are given separate interfaces: the professional side sees detailed clinical information and the individual side sees a simple summary, so requests from the two

sides do not meet in the same place. In another, the conflict was not about features but about trust, and it was resolved through a rule rather than a feature. In this pooled payment system, the platform holds the buyer's payment and releases it to the seller only after the buyer confirms that the item has arrived, which answers the concerns of both sides at once. In the third, priority is set by how many users ask for something and how much effort it takes, rather than by which side asked, with brand identity as a final filter. In each case the decision rule avoids the comparison rather than making it.

The backlog cannot take this decision because it ranks work; it does not settle disputes. Ranking works when items compete for limited development time, since a lower position means later rather than never. It does not work when items compete for the value of the product itself, which is what happens on a platform. Putting one side's requests consistently below the other's is not a scheduling decision. It reduces what the platform offers that side, and because the two sides are held together by cross-side effects, a weakened side eventually reduces the value delivered to the side that was favoured (Rochet & Tirole, 2003; Van Alstyne et al., 2016). A tool designed to postpone work is therefore being asked to do something that damages the platform whichever way it is used. The conflict has to be moved to a level where both sides can be satisfied at the same time. Figure 5.2 sets out this move and the levels at which it occurs.

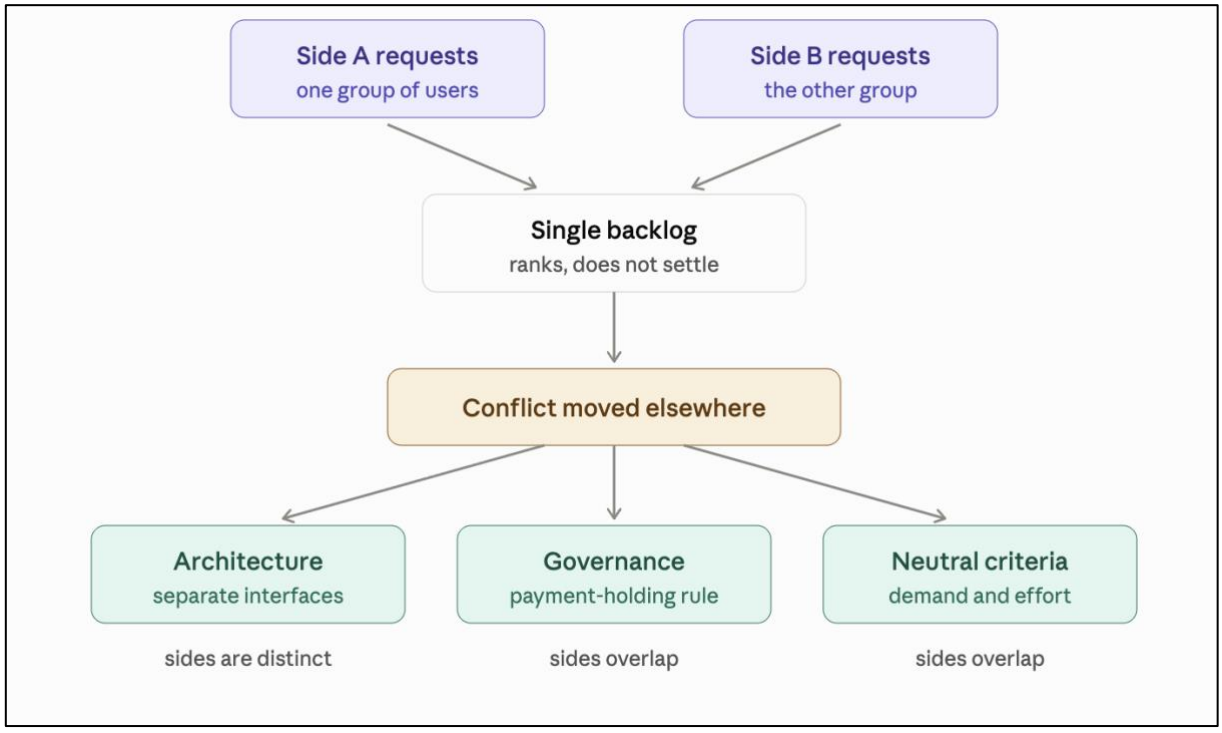


Figure 5.2. How conflicts between the two sides are settled outside the backlog. Source: author's elaboration based on interview data.

Which level is available depends on side configuration. Where the sides are distinct, the conflict can be moved into the architecture, as in the separate interfaces, because a change can be directed at one side without taking anything away from the other. The detail the professional side needs does not have to be withheld in order to keep the individual side's view simple; the two views can exist at the same time, and the difference between what each side wants is absorbed by the structure of the product instead of being decided in favour of one of them. This is an example of what Gawer (2021) describes as the digital interface acting as the boundary between platform sides, and it shows that such boundaries can come out of ordinary prioritisation pressure rather than from deliberate platform strategy. Where the sides overlap, this route closes. In the peer-to-peer platform, the same person is both buyer and seller, so a change cannot be aimed at one role alone, and every feature is assessed for both roles at once. There, the conflict is moved instead into the platform's governance, through the pooled payment system. In the mature platform, it is moved into the prioritisation criteria themselves, which rank work by how many users ask for a feature and how much effort it takes, without reference to which side asked.

Development stage acts as a second condition on the same mechanism. In the earliest-stage platform, where the paying side is institutional, and the free side is individual, the founder expects to prioritise the paying side if the two ever disagree. This fits the platform as it currently stands, because its cross-side effects run mainly in one direction: the institutional side brings users, while users do not yet bring institutions at a comparable rate. Favouring the paying side therefore costs the platform little at present. This expectation is reasonable now, but unsustainable later. Once the effect runs in both directions, favouring the paying side reduces the population that makes the paying side willing to pay, and the same move that protects revenue begins to erode it. What looks like a straightforward commercial rule in an early-stage platform is a rule that will not hold indefinitely, and the move described above becomes necessary at the point when the one-directional effect disappears.

This section contributes to the literature on how platforms change over time. Tiwana et al. (2010) show that platform architecture, governance, and environmental dynamics change together rather than separately, and Jovanovic et al. (2022) find a similar pattern for architecture, services, and governance in industrial platforms. Both describe this at the level of the platform and over long periods. The cases show how the same connection appears in the daily work of a single team. When the two sides ask for different things, the team cannot settle

the question by ordering a backlog, so the answer is built into the interface or written into the rules instead. Separate interfaces and the pooled payment system were not chosen as architectural or governance strategies. They were the outcome of prioritisation decisions that a backlog could not make, and once in place they remain part of the platform. Agile therefore does more than build the platform: the questions it cannot answer within its own process become part of the platform's structure.The mechanism can be stated briefly. What drives it is whether the two sides can be addressed separately, together with how evenly the cross-side effects run at the platform's current stage. The move becomes necessary when the two sides ask for things that cannot both be met in the same design. Where it goes is decided by whether the sides can be addressed independently: into the architecture if they can, into the rules or into side-neutral criteria if they cannot. It happens because ranking, which is what a backlog does, cannot settle a disagreement between two groups whose participation the platform equally needs.

5.2. Theoretical Propositions

The three mechanisms discussed above are stated here as propositions. Each one connects a characteristic of the platform to the way Agile is used, and then to an effect that returns to the platform. A proposition is not the same as a theme: the themes used to organise the findings describe what was observed, while a proposition claims a relationship between two of them, which requires them to remain separate in the coding. They follow from three cases and would need to be tested on a larger sample.

Proposition 1: The chicken-and-egg problem does not end at launch. It becomes a balance that has to be maintained. The decisions that maintain it, such as which side is charged, which side is given free access, and which side receives additional tools, cannot be broken into separate pieces of work and cannot be handled as backlog items. While these decisions are still open, agility works as a mindset rather than as a set of formal practices. Formal practices are adopted once the terms for each side are settled and the remaining work can be divided. This is also the point at which large changes have become harder to make.

Proposition 2: What drives iteration changes with the completeness of the platform's sides, not with the age of the venture. While only one side is populated, behavioural data cannot be interpreted, because a signal on one side may come from the state of the other. Iteration is

therefore guided by the founders' professional knowledge, which provides the comparison the data cannot yet provide. Once the second side is active, behaviour can be read on its own terms, and judgement is replaced by data. The decisions taken before this point are already built into the platform.

Proposition 3: When the two sides want different things, the conflict is not settled in the backlog. Ranking one side below the other does not postpone work. It reduces what the platform offers that side, and through cross-side effects this eventually reaches the other side as well. The conflict is therefore moved to a level where both sides can be satisfied at once. Side configuration decides which level is available. Where the sides are distinct, the conflict is moved into the architecture. Where they overlap, it is moved into the platform's governance or into prioritisation criteria that do not mention the sides.

6. Conclusion and future developments

This study set out to understand how Agile ways of working and the creation of digital platform startups relate to each other. Two research questions guided the work. The first asked what role Agile practices and mindset play in the creation of digital platform startups. The second asked how platform characteristics, in particular side configuration and stage of development, shape the use of Agile practices and the resulting changes to the platform. Three born-platform startups operating in different sectors and at different stages were studied through interviews with founders and managers, and their accounts were compared within and across cases.

Three mechanisms were identified. Each one connects a characteristic of the platform to the way Agile was used in these cases.

First, the balance between the two sides was a continuous concern rather than something solved at launch. The startups studied kept revising which side was charged, which side was given free access, and which side received additional tools. These decisions affected both sides at the same time, so they could not be divided into separate pieces of work and were not handled as items in a backlog. While such decisions were still open, agility appeared as a mindset rather than as a set of formal practices. The earliest-stage case worked iteratively without sprints, stand-ups, or a structured backlog. The more established cases had added formal practices as their teams grew and the remaining work became easier to divide.

Second, what drove iteration appeared to change with the completeness of the platform rather than with the age of the venture. In the earliest-stage case, neither side provided structured feedback, and iteration was guided by the founder's professional knowledge. In the more established cases, it was guided by user behaviour. A signal observed on one side may come from the state of the other, so behavioural data could not be interpreted with confidence until both sides were active. The decisions taken in the meantime were already built into the platform by the time evidence became available.

Third, none of the startups studied settled disagreements between the two sides by ranking one above the other in the backlog. The conflict was moved instead to a level at which both sides could be satisfied at once. Where the sides were distinct, it was moved into the design of the platform, as in the separate interfaces built for pharmacies and individual users. Where the sides overlapped, this was not possible, and the conflict was moved into the platform's governance, as in the pooled payment system, or into prioritisation criteria that do not refer to the sides at all.

Taken together, these mechanisms address both research questions. In relation to the second question, the stage of the platform and the configuration of its sides appeared to shape how Agile was used. The stage shaped which form of agility was available and what iteration could be based on. The configuration of the sides shaped where conflicts could be settled. In relation to the first, Agile also shaped the platform itself. Because agility operated as a mindset in the early stage, the founding terms of the platform remained open to revision for longer than the literature on platform launch suggests. Because iteration relied on founder judgement before the second side arrived, the founders' professional knowledge was written into the platform itself. Finally, the backlog could not settle conflicts between the two sides, so those conflicts were settled elsewhere. Agile therefore does more than build the platform. The questions it cannot answer within its own process become part of the platform's structure.

Based on the cases studied, these findings contribute to two streams of literature. For the literature on two-sided platforms, they suggest that the subsidy decision described by Eisenmann et al. (2006) is not a single choice taken at launch, but one that is reopened as the balance between the sides shifts. For the literature on Agile, they suggest that the distinction between Agile as a tool and Agile as a culture (Magistretti & Trabucchi, 2025) is not only a matter of organisational maturity, but depends on whether the work in front of the team can be divided at all. In practice, the findings indicate that founders of platform startups should not read the absence of formal practices as a lack of discipline in the early stage, and that behavioural data is likely to become useful only once both sides are active.

The study has several limitations. It is based on three cases in a single country, so the findings are not intended to be generalised beyond them. The evidence for each case comes from a small number of key people who were directly involved in the decisions under study, complemented by written follow-up exchanges and publicly available material. This provided direct access to the reasoning behind those decisions, but it also means that the accounts are self-reported, and

in the more established cases partly retrospective, which may affect how the early stage is remembered. The coding was carried out by a single researcher, and although the material was reviewed a second time and compared with secondary sources, this remains a constraint. Finally, in these three cases the stage of development, the size of the team and the completeness of the two sides move together, so their effects could not be separated. The distinction drawn between the first and second mechanisms is therefore analytical rather than empirical.

These limitations suggest directions for future research. A larger and more varied sample could separate the conditions that are combined here, for example by studying a platform that starts from an existing community, where both sides are present while the team is still small, or a well-funded venture whose second side has not yet arrived. Longitudinal research following a platform startup as its sides fill in would make it possible to observe these transitions as they happen, rather than reconstructing them afterwards. Such work could also examine what happens to prioritisation when cross-side effects that once ran mainly in one direction begin to run in both, since the present study observed this asymmetry only at a single point in time.

Bibliography

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for agile software development*. https://agilemanifesto.org/

Bianchi, M., Marzi, G., & Guerini, M. (2020). Agile, stage-gate and their combination: Exploring how they relate to performance in software development. *Journal of Business Research, 110,* 538–553. https://doi.org/10.1016/j.jbusres.2018.05.003

Buvik, M. P., & Tkalich, A. (2021). *Psychological safety in agile software development teams: Work design antecedents and performance consequences* (arXiv:2109.15034). arXiv. https://arxiv.org/abs/2109.15034

Caillaud, B., & Jullien, B. (2003). Chicken & egg: Competition among intermediation service providers. *RAND Journal of Economics, 34*(2), 309–328.

Cusumano, M. A., Gawer, A., & Yoffie, D. B. (2019). *The business of platforms: Strategy in the age of digital competition, innovation, and power*. Harper Business.

De Reuver, M., Sørensen, C., & Basole, R. C. (2018). *The digital platform: A research agenda. Journal of Information Technology, 33(2), 124–135.*

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software, 85*(6), 1213–1221.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review, 14*(4), 532–550.

Eisenmann, T., Parker, G., & Van Alstyne, M. W. (2006). Strategies for two-sided markets. *Harvard Business Review, 84*(10), 92–101.

Eva, N., Robin, M., Sendjaya, S., van Dierendonck, D., & Liden, R. C. (2019). Servant leadership: A systematic review and call for future research. *The Leadership Quarterly, 30*(1), 111–132. https://doi.org/10.1016/j.leaqua.2018.07.004

Evans, D. S. (2003). The antitrust economics of multi-sided platform markets. *Yale Journal on Regulation, 20*(2), 325–381.

Filistrucchi, L., Geradin, D., van Damme, E., & Affeldt, P. (2014). Market definition in two-sided markets: Theory and practice. *Journal of Competition Law & Economics, 10*(2), 293–339.

Gawer, A. (2021). *Digital platforms' boundaries: The interplay of firm scope, platform sides, and digital interfaces.Long Range Planning, 54(5), 102045.* https://doi.org/10.1016/j.lrp.2020.102045

Gawer, A., & Cusumano, M. A. (2014). Industry platforms and ecosystem innovation. *Journal of Product Innovation Management, 31*(3), 417–433. https://doi.org/10.1111/jpim.12105

Ghezzi, A., & Cavallo, A. (2020). Agile business model innovation in digital entrepreneurship: Lean startup approaches. *Journal of Business Research, 110,* 519–537. https://doi.org/10.1016/j.jbusres.2018.06.013

Hagiu, A., & Wright, J. (2015). Multi-sided platforms. *International Journal of Industrial Organization, 43*, 162–174.

Hogan, S. J., & Coote, L. V. (2014). Organisational culture, innovation, and performance: A test of Schein's model. *Journal of Business Research, 67*(8), 1609–1621. https://doi.org/10.1016/j.jbusres.2013.09.007

Jovanovic, M., Sjödin, D., & Parida, V. (2022). *Co-evolution of platform architecture, platform services, and platform governance: Expanding the platform value of industrial digital platforms*. *Technovation, 118*, 102218. https://doi.org/10.1016/j.technovation.2020.102218

Kopalle, P. K., Kumar, V., & Subramaniam, M. (2020). How legacy firms can embrace the digital ecosystem via digital customer orientation. *Journal of the Academy of Marketing Science, 48*(1), 114–131.

Magistretti, S., & Trabucchi, D. (2025). Agile-as-a-tool and agile-as-a-culture: A comprehensive review of agile approaches adopting contingency and configuration theories. *Review of Managerial Science, 19*(1), 223–253. https://doi.org/10.1007/s11846-024-00745-1

Mikalef, P., & Pateli, A. (2017). Information technology-enabled dynamic capabilities and their indirect effect on competitive performance: Findings from PLS-SEM and fsQCA. *Journal of Business Research, 70*, 1–16. https://doi.org/10.1016/j.jbusres.2016.09.004

Miles, M. B., & Huberman, A. M. (1984). *Qualitative data analysis: A sourcebook of new methods*. SAGE.

Muzellec, L., Ronteau, S., & Lambkin, M. (2015). Two-sided Internet platforms: A business model lifecycle perspective. Industrial Marketing Management, 45, 139–150. https://doi.org/10.1016/j.indmarman.2015.02.012

Neumann, M., Kuchel, T., Diebold, P., & Schön, E.-M. (2024). *Agile culture clash: Unveiling challenges in cultivating an agile mindset in organisations* (arXiv:2405.15066). arXiv. https://arxiv.org/abs/2405.15066

Ojala, A., & Lyytinen, K. (2022). How do entrepreneurs create indirect network effects on digital platforms? A study on a multi-sided gaming platform. *Technology Analysis & Strategic Management*.

Porkodi, S. (2024). The effectiveness of agile leadership in practice: A comprehensive meta-analysis of empirical studies on organisational outcomes. *Journal of Entrepreneurship, Management and Innovation, 20*(2), 117–138. https://doi.org/10.7341/20242026

Porter, M. E. (1985). *Competitive advantage: Creating and sustaining superior performance*. Free Press.

Rigby, D. K., Sutherland, J., & Takeuchi, H. (2016). Embracing agile. *Harvard Business Review, 94*(5), 40–50.

Rochet, J.-C., & Tirole, J. (2003). Platform competition in two-sided markets. *Journal of the European Economic Association, 1*(4), 990–1029. https://doi.org/10.1162/154247603322493212

Rysman, M. (2009). The economics of two-sided markets. *Journal of Economic Perspectives, 23*(3), 125–143.

Salameh, A., & Bass, J. M. (2019). Spotify tailoring for promoting effectiveness in cross-functional autonomous squads. In R. Hoda (Ed.), *Agile processes in software engineering and extreme programming – Workshops* (Lecture Notes in Business Information Processing, Vol. 364, pp. 20–28). Springer. https://doi.org/10.1007/978-3-030-30126-2_3

Saldaña, J. (2021). *The coding manual for qualitative researchers* (4th ed.). SAGE.

Schein, E. H. (2010). *Organisational culture and leadership* (4th ed.). Jossey-Bass.

Schreieck, M., Ondrus, J., Wiesche, M., & Krcmar, H. (2024). A typology of multi-platform integration strategies. *Information Systems Journal, 34*(3), 828–853. https://doi.org/10.1111/isj.12450

Schreieck, M., Ondrus, J., Wiesche, M., & Krcmar, H. (2025). Platform governance for established companies. *California Management Review, 67*(4), 48–74. https://doi.org/10.1177/00081256251338251

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide.* https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

Shapiro, C., & Varian, H. R. (1999). *Information rules: A strategic guide to the network economy*. Harvard Business School Press.

Springer, V., Randhawa, K., Jovanović, M., Ritala, P., & Piller, F. T. (2025). Platform design and governance in industrial markets: Charting the meta-organisational logic. *Research Policy, 54*(6), 105236. https://doi.org/10.1016/j.respol.2025.105236

Täuscher, K., & Laudien, S. M. (2018). Understanding platform business models: A mixed methods study of marketplaces. *European Management Journal, 36*(3), 319–329.

Teece, D. J., Pundziene, A., Heaton, S., & Vadi, M. (2022). Managing multi-sided platforms: Platform origins and go-to-market strategy. *California Management Review, 64*(4). https://doi.org/10.1177/00081256221109961

Tiwana, A., Konsynski, B., & Bush, A. A. (2010). Research commentary—Platform evolution: Coevolution of platform architecture, governance, and environmental dynamics. *Information Systems Research, 21*(4), 675–687. https://doi.org/10.1287/isre.1100.0323

Trabucchi, D. (2020). Let's get a two-sided platform started: Tactics to solve the chicken and egg paradox. *Journal of Business Ecosystems, 1*(1), 63–77

Trabucchi, D., & Buganza, T. (2019). *Fostering digital platform innovation: From two- to multi-sided platforms*. Creativity and Innovation Management, 28(3), 345–358

Trabucchi, D., & Buganza, T. (2022). Landlords with no lands: A systematic literature review on hybrid multi-sided platforms and platform thinking. *European Journal of Innovation Management, 25*(6), 64–96. https://doi.org/10.1108/EJIM-11-2020-0467

Uhl-Bien, M., & Arena, M. (2018). Leadership for organisational adaptability: A theoretical synthesis and integrative framework. *The Leadership Quarterly, 29*(1), 89–104. https://doi.org/10.1016/j.leaqua.2017.12.009

Van Alstyne, M. W., Parker, G. G., & Choudary, S. P. (2016). Pipelines, platforms, and the new rules of strategy. *Harvard Business Review, 94*(4), 54–62.

Werner, L., & Le-Brun, G. (2025). Become an octopus organisation. *Harvard Business Review, 103*(6).

Yin, R. K. (2018). *Case study research and applications: Design and methods* (6th ed.). SAGE.

Appendix A: Semi-Structured Interview Protocol

The following question set served as the main guide for the interviews. In line with the semi-structured research design, the sequence of questions was adapted to the dynamics of each interview. Additional probing questions were introduced based on the participants' input to examine key themes more thoroughly.

Block 1: Side Configuration

1. Who are the distinct user or customer groups currently interacting on your platform?

2. Which side uses the platform free of charge, and which side pays? What rationale shaped this pricing structure?

3. Has the configuration of sides expanded since launch, or has the initial structure remained fixed?

Block 2: Cross-Side Network Effects

4. How does user growth on one side affect the value delivered to the other side?
5. Is this cross-side effect bidirectional, or is one side noticeably more dependent on the growth of the other?
6. What happens to user engagement on one side if participation on the other side decreases?

Block 3: Stage of Development and Early Growth Challenges

7. How did you acquire your initial users on each side during the early platform launch?
8. Which side was more difficult to attract, and what specific strategies or incentives were required to onboard them?

Block 4: Triggers of Iteration

9. What was the most significant feature change or system update made to the platform in the last two to three months?
10. What specifically triggered this change (e.g., direct user requests, log analysis, internal evaluations, or founder intuition)?

11. Through which channels or indicators do you typically identify that a feature is failing or requires refinement?

Block 5: Backlog and Customer Feedback Prioritisation

12. How do you manage situations where the needs or expectations of different platform sides diverge or conflict?

13. In these instances, which side's feedback or requirements take priority in your product backlog?

14. When general feedback comes in from multiple sides, which side's input carries more weight when deciding what to build next?

Block 6: Ways of Working and Formal Agile Practices

15. How do you plan development tasks on a daily or weekly basis, and do you follow fixed iteration cycles such as sprints?

16. How frequently does the team meet, and do you observe formal agile ceremonies (e.g., daily stand-ups, sprint planning, retrospectives?

List of Figures

Figure 2.1. Amazon's shift from a linear value chain to a two-sided platform. Source: author's elaboration based on Hagiu & Wright (2015). 6
Figure 2.2. Amazon's evolution from a two-sided transactional platform to a hybrid multi-sided platform. Source: author's elaboration based on Trabucchi & Buganza (2022). 6
Figure 2.3. Geographical Distribution of the Market Value of the World's 100 Largest Platforms Source: Schmidt (2025) 7
Figure 2.4. Transactional platforms: the example of Airbnb. Source: author's elaboration based on Trabucchi & Buganza (2022) and Eisenmann et al. (2006). 8
Figure 2.5. Innovation platforms: the example of Apple's iOS. Source: author's elaboration based on Trabucchi & Buganza (2022) and Gawer & Cusumano (2014). 9
Figure 2.6. Orthogonal platforms: the example of Strava. Source: author's elaboration based on Trabucchi & Buganza (2022). 10
Figure 2.7. Transactional, Orthogonal, and Hybrid Multi-Sided Platforms Model (Trabucchi & Buganza, 2022) 11
Figure 2.8. Same-side and cross-side network effects in a transactional platform. Source: author's elaboration based on Eisenmann et al. (2006). 13
Figure 2.9. The Platform Thinking Matrix (Trabucchi & Buganza, 2022)..... 16
Figure 2.10. An illustration of the Scrum cycle. Source: author's elaboration based on Schwaber & Sutherland (2020). 21
Figure 4.1. Platform configuration of Case A. Source: author's elaboration based on interview data. 33
Figure 4.2. Platform configuration of Case B. Source: author's elaboration based on interview and secondary data. 37
Figure 4.3. Platform configuration of Case C. Source: author's elaboration based on interview and secondary data. 41
Figure 5.1. Stage characteristics across the three cases, and the two transitions observed between them. Source: author's elaboration based on interview data. 48

Figure 5.2. How conflicts between the two sides are settled outside the backlog. Source: author's elaboration based on interview data..... 52

List of Tables

Table 2.1. The twelve principles of the Agile Manifesto. Source: Beck et al. (2001). 21
Table 3.1. Overview of the cases. 27
Table 3.2. Data collected for each case 29
Table 3.3. Data structure of the coding process 30
Table 4.1. Cross-case comparison 43

Acknowledgements

I would like to express my gratitude to my advisor, Prof. Daniel Trabucchi, for his guidance and support throughout the development of this thesis. I am also grateful to my co-advisor, Lorenzo Lione, for his suggestions, his detailed feedback, and his availability whenever I needed direction. Their contributions shaped both the structure and the focus of this work.

I would also like to thank the founders and managers of the three startups who agreed to be interviewed for this study. Their willingness to share their experience openly made this research possible.

On a personal note, I would like to thank my family and my friends for their love, patience, and encouragement throughout this journey. They have been my greatest source of strength during this period. I am also grateful to the companies I worked with and to my colleagues for their support along the way.

