Platform Concepts

5

5.1 Overview

The rise of platform concepts can be observed in numerous industries. The automotive industry was already hailing product platforms as a recipe for success back in the 1990s. Many other industries followed suit (cf. e.g., Köhler 2004 on product platforms in the media industry). In the following, we will explore the role of platforms in the software industry. Drawing on the work of Gawer (2009), we can distinguish between two generic types of platforms: product platforms enable products and services to be produced efficiently through the re-use of existing modules (Wheelwright and Clark 1992); while the main purpose of industry platforms is to attract complementary products and/or services from third parties in an industry (Cusumano and Gawer 2002). Figure 5.1 shows other characteristics of these two types.

Both varieties of platforms are now also found in the software industry. In the field of consumer software, there is already a multitude of examples: the best known is Apple's AppStore for the iPhone. Industry platforms for business software are still in their infancy, for example salesforce.com's AppExchange, Google Apps Marketplace, Microsoft Pinpoint, or SugarCRM's SugarExchange (Burkard et al. 2010).

5.2 Product Platforms in the Software Industry

5.2.1 Cost Structure of Platform-Based Software Development

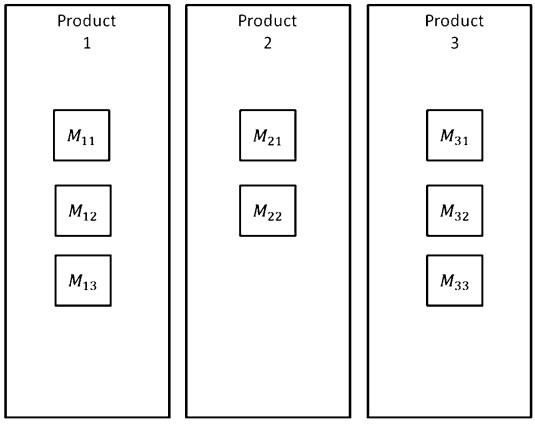
Like their high-profile counterparts in the automotive industry, product platforms in the software industry are intended to reduce development costs while maintaining or enhancing quality. In particular, software product platforms are used to make and deliver products in a product line that address different target groups with different price categories and system environments—flexibly and above all efficiently (Reussner and Hasselbring 2006).

P. Buxmann et al., *The Software Industry*, DOI: 10.1007/978-3-642-31510-7_5,

Fig. 5.1 Platform typology (based on Gawer 2009)

Plattform Typ	Produktplattform	Branchenplattform
Participants	One company (and in some cases its suppliers)	Multiple companies with compatible products
Objectives	• Faster and more efficient development and production • Greater product variety at lower cost • Increased flexibility when designing new products	• For platform operators: Boost a platform's usefulness through complementary products and services • For suppliers of complementary services: Increased sales
Design principles	• Reuse of components • Stable underlying architecture	• Stable interfaces for extensions

Fig. 5.2 Example of the development of three products without a platform



These product platforms are designed to exploit reusable software modules, just as the automotive industry example cited above involves the reuse of components (Boysen and Scholl 2009). A software module is a unit of software that performs a defined task or function and can communicate with other modules via interfaces, for example on the basis of the SOA standards outlined in the previous chapter, but can also be run independently. Modules are managed via configuration software, in other words the product platform. Most crucially, a module should be used in as many products as possible (Baldwin and Clark 1997; Miller and Elgård 1998; Meyer and Lehnerd 1997).

A simple cost model demonstrates how the introduction of a product platform can alter a software provider's cost structure. We shall assume that each of the provider's products is developed separately, but in a modular way. Accordingly, all N products, each consisting of M_n $(n = 1 \ldots N)$ modules, are developed independently of each other. Figure 5.2 illustrates an example with three products, each comprising two to three product-specific modules.

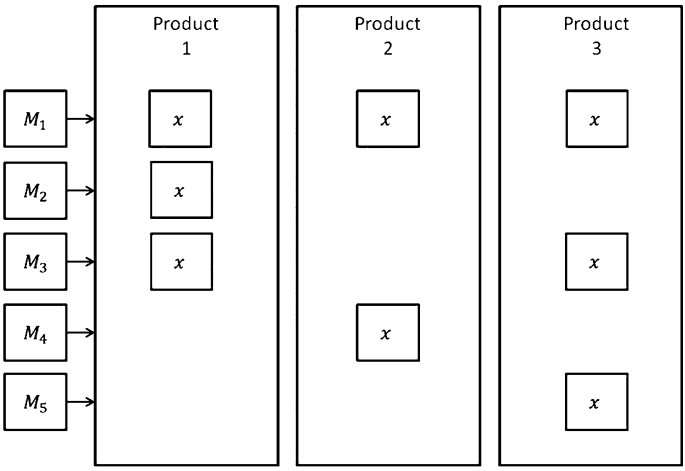


Fig. 5.3 Configuration of three products via a product platform

The system n has development costs K_n, which are made up of the total of the K_{nm} development costs of M_n individual modules $(M = 1 \ldots .M_n)$. The costs of the development of the provider's N systems, therefore, amounts to:

$$K = \sum_{n=1}^{N} K_n = \sum_{n=1}^{N} \sum_{m=1}^{M_n} K_{\mathrm{nm}}$$

This approach will now be contrasted with the cost structure after the introduction of a product platform, or in other words, the reuse of components. For the vendor, there will be upfront expenditure to establish the product platform. We will call this K_p. Development of the modules incurs costs for each of the D modules, which we will transcribe as K_d $(d = 1 \ldots .D)$. Each module will be implemented in $1 \leq \mathrm{n} \leq \mathrm{N}$ systems. Integrating all these modules into a single system creates additional costs. To keep this simple, we shall assume an average cost rate, which we shall call K_d. Accordingly, the development of a vendor's N systems using a platform concept entails the following costs:

$$K = K_p + \sum\nolimits_{d=1}^{D} K_d + N \times K_1$$

Figure 5.3 depicts a usage matrix showing which modules are used in which products. In our example, module 3 is used in products 1 and 3; module 1 is included in all three products. Modules 2, 4, and 5, on the other hand, are used in only one product.

By comparing cost functions (1) and (2), we can draw conclusions about the changes in cost structure that can be expected on the introduction of a product platform. First, we can clearly see that introducing a product platform requires a

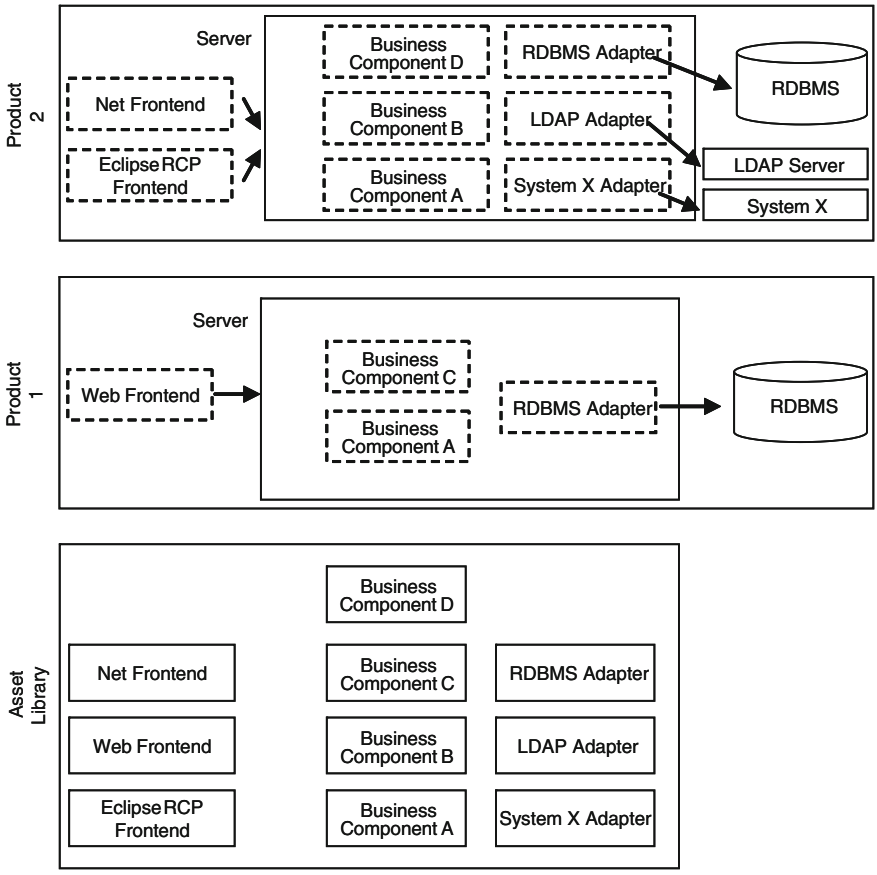


Fig. 5.4 Relationship between product and components (based on Zacharias 2007, p. 74)

large and potentially risky investment to develop it. For each of the N products there are integration costs of K_I. In real life, the size of these costs is determined by the quality and the future-proof design of the platform interfaces. Initially, at least, this upfront investment is a certain challenge.

However, the deployment of product platforms leads to sizeable cost savings, particularly if the module reuse rate is high. In other words, if the modules, once developed, can be reused in as many products as possible, or if the number of modules required to create for a given number of products can be reduced. Our example illustrates the latter scenario: the number of products remains the same, but the number of modules has been trimmed from eight to five.

In the automotive industry, as cited above, the leading manufacturers produce only a small proportion of the modules themselves. Instead, they concentrate mainly on integrating modules to create their products. The development and production of modules is the task of highly specialized suppliers.

5.2.2 Organizational Support for Implementing Product Platforms

A major challenge for software providers is to align their organizational structures with the rather abstract notion of a product platform. Here, both theory and practice are still embryonic. In addition to the simpler concepts found in the Web application development space, there are also more comprehensive approaches, such as component-based software development (Messerschmitt and Szyperski 2003, p. 244–263).

Zacharias provides some interesting initial insights (Zacharias 2007), dividing tasks into domain engineering, application engineering, and product line management:

- *Domain engineering* constitutes the technical, specialist, and organizational basis for component-based development. In particular, this includes the maintenance of an asset library, which contains all available components. Domain engineering also involves the provision of development environments and procedural models.
- *Application engineering* develops software solutions, drawing on the components provided by domain engineering, and with new, complementary components. Simultaneously, application engineering passes these newly developed components back to domain engineering.
- *Product line management* oversees domain engineering and application engineering and ensures these are in alignment.

In the final analysis, products are nothing more than special configurations of components, which are linked at the time of development, installation, or at runtime. Figure 5.4 illustrates the concept of component-based engineering, after Zacharias.

Van der Linden et al. (2004) have developed a guideline for the implementation and evaluation of product platforms. They look at a product platform from four perspectives: the business dimension, the architectural dimension, the procedural dimension, and the organizational dimension. For each of these four dimensions, there are different evaluation levels that can be used to assess an organization's current status in relation to the implementation of a product platform.

5.2.3 Add-on: Industrialization as a Management Concept for the Software Industry

In the previous section, we looked at component reusability and the related principle of standardization. In Chap. 4, we discussed various forms of outsourcing. In Sect. 3.4.2, we analyzed current approaches to automating software development. These three concepts are all part of the broader concept of industrialization, which is currently the subject of much discourse and is becoming increasingly important for the software industry.

Industrialization is a historically evolved management concept that offers a framework for cost-effective mass manufacturing. According to this concept, the key factors are: increased standardization of products and processes; greater specialization (i.e., an increasing division of labor); and automation (Heinen 1991, p. 10; Schweitzer 1994, p. 19). These three elements are interrelated, standardization being the most important prerequisite for implementing both specialization and automation. To put it another way: specialization and automation are not possible without standardization. But even standardization alone can lead to lower unit costs. This is because only standardized processes can be carried out by a machine or divided among multiple parties. Similarly, standardized processes are only possible with standardized products.

New technologies play a central role in industrialization. As they create new potential for automation, specialization, and standardization, they are often described as drivers of industrialization.

Based on these technologies, we can demarcate three stages of industrialization, two of which have already come to a close, and one that continues to this day (Condrau 2005). The first period begins with the Industrial Revolution, from 1780 onwards. The invention of the steam engine, railroad, and power loom saw artisan-type one-at-a-time manufacturing replaced by an early form of industrial mass production. Whereas goods had previously been made individually for the craftsman's own use, machines enabled the more standardized manufacture of large quantities of products for sale. This first stage of industrialization was a time of enormous productivity gains and rapid economic growth.

The concept evolved further during the second stage of industrialization, which commenced in 1840. In addition to studies conducted by academics, Taylor's work on scientific management is a case in point, key drivers in this phase were the discovery of electricity and the invention of electric motors. While in the first stage of industrialization, the production process was shaped by a quantitative division of labor (i.e., dividing similar activities between multiple machines or human resources), the installation of conveyor belts and the consequent increase in standardization meant that work could be divided according to the type of task. As a result of this greater specialization, highly standardized material goods, such as the Model T Ford, could be created, stimulating further productivity gains.

The IT revolution, which began in the late twentieth century, is now seen as ushering in a third stage of industrialization. While the first two stages involved the manufacture of material goods, the focus is now on creating information-intensive services and products. This development has been driven by the well-known waves of innovation in information and communication technology. Figure 5.5 summarizes the factors behind the industrialization of software development which were discussed previously in various parts of this book.

Figure 5.6 outlines the three stages of industrialization and their key characteristics. In the literature, the service sector is sometimes presented as the primary target of the third stage of industrialization. We do not share this point of view: Industrialization, through modern ICT, is also impacting information products such as books or software, which are not part of the service sector.

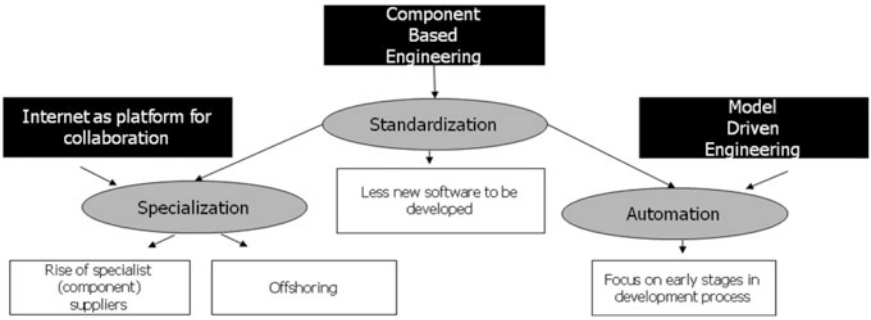


Fig. 5.5 Drivers and expected effects of industrialization in the software industry

	1st stage of industriali-zation	2nd stage of industriali-zation	3rd stage of industriali-zation
Drivers	Steam engine, railroad, power loom	Electricity, electric motors	Information and communication technology
Sector	Material goods	Material goods	Information-intensive products and services
Time period	Approx. 1780-1840	Approx. 1840-1960	Since the late 20th century
Keywords	Industrial Revolution	Fordism and Taylorism	IT Revolution

Fig. 5.6 Three stages of industrialization

However, it should be noted that there are limitations to the division of labor and automation, and therefore to the concept of industrialization. These limitations concern the motivation of employees, the costs of distributed manufacturing, and the flexibility of the production processes. When there is a very deep division of labor, tasks become increasingly similar. Taken to excess, this becomes mind numbing and monotonous for workers. As a result, their motivation sinks, diminishing the unit cost advantage. In addition, a high level of automation requires a high level of standardization: This makes changes to processes, and therefore also to products, increasingly difficult. In light of both of these aspects, the manufacturing industry no longer sees maximum industrialization as its ultimate ambition.

5.3 Industry Platforms in the Software Industry

5.3.1 Openness of Industry Platforms

As we noted at the start of this chapter, industry platforms provide a basis for products and/or services whose functionality is extended by complementors. For example, most game consoles include only the basic functionality required for playing games, such as a graphics accelerator. The games themselves are developed by third parties. Industry platform operators' main objective is to maximize the attractiveness of their platform, by offering the largest possible number of complementary products to the largest possible group of end customers (who can be consumers or companies).

In more general terms, platform operators are primarily concerned with achieving the right degree of openness for their platform. A platform can be described as completely open, if it has no limitations as to participants, development, use, and commercialization of the platform (Eisenmann et al. 2009). This extreme form is rare, however, especially when it comes to commercially operated platforms. Rather than simply taking an either-or decision, platform operators need to find the ideal degree of openness for their platform.

The difficulty herein lies in the trade off between openness and control: while more openness encourages third parties to participate, increasing the platform's attractiveness, it will generally lead to a loss of control over the platform design (West 2003). In the following section, we distinguish between vertical and horizontal openness.

5.3.1.1 Vertical Openness

Vertical openness refers to what extent complementary products or services from external providers can or should be provided. In this regard, we can differentiate between three platform-specific parameters which we will address below.

Exclusivity

A key parameter in deciding the degree of openness is the agreement of exclusive rights. Their goal is to ensure that a specific complementary product is offered exclusively via the platform (Eisenmann and Wong 2004). For example, new versions of the popular computer game Grand Theft Auto GTA are first available only on Sony's Playstation platform. A second form of exclusivity agreement is category exclusivity. In this case, the platform operator and complementor agree that a certain type of application may only be offered by that particular complementor. Strategies such as these, which essentially reduce openness, make particular sense when one side needs to make high and specific investments, and is only prepared to do so when the other side ensures exclusive access to the distribution channel in return. This form of agreement can also be observed on the market for console games: Platform

operators commonly limit console games' access to the market—ensuring that, on one hand, the chosen games are of sufficiently high quality; and on the other, that they can command high licensing fees.

Reverse Compatibility

When a platform is further developed, new and/or additional functionality usually becomes available to the developers of complementary applications. An important decision is the required about reverse compatibility; in other words, whether to ensure that complementary extensions developed for an older version of the platform will also be able to run on newer versions. Platform providers need to weigh up whether to accept potentially higher costs and limitations in the further development of the platform, in order to make all existing complementary offerings available on new platform versions (Choi 1994). When partner applications are a central feature of the platform, ensuring reverse compatibility is essential. As proclaimed by its slogan, "There's an app for everything", Apple has made the availability of complementary applications a key competitive advantage of the iPhone. In consequence, the company needed to ensure reverse compatibility when developing the fourth generation iPhone. This is why the company deliberately opted for a new display resolution with exactly double the number of pixels, both vertically and horizontally. This step ensured that all applications developed for previous generations of iPhone will run on the new version without a hitch. If Apple had overlooked this, all developers of complementary applications would have been forced to provide a new version tailored to the new platform. In turn, this would have diminished the amount of vertical openness.

Integration of Complementary Applications into the Platform Core

Another method of determining the degree of openness is to integrate complementary applications previously developed by external partners into the core of the platform, in the course of further development. An example of this type of strategy is Microsoft's Windows operating system: A multitude of standard applications, such as browsers, media players, and system utilities, were previously offered only by third parties as optional extras. Now, they are an integral part of current versions. As will be explained in the following section on the management of complementors, this step can be viewed as closing the platform, as these partners run the risk of competing with the operator (Yoffie and Kwak 2006). At the same time, there are reasons for this kind of strategy, such as reducing strategic dependence on particular providers, or the opportunity to realize economies of scale.

5.3.1.2 Horizontal Openness

A platform's openness to other platforms or third parties is described as horizontal. This can be achieved via the following parameters (Eisenmann et al. 2009):

Interoperability with Other Platforms

The main method by which a platform can be opened horizontally is the provision of open interfaces and tools (Katz and Shapiro 1985). The Facebook Connect interface is an example of this way of creating interoperability: Once a user has registered, other platforms, such as Yahoo, can access their profile data. At the same time, Facebook even displays the user's activities on other platforms. As the Facebook platform had already reached a certain maturity by the time this converter was introduced (late 2008), this strategic opening provided an opportunity to grow user numbers beyond the core target group. Conversely, using Facebook Connect is also attractive for competing platforms, as it enables them to access the data of users already registered on Facebook.

Licensing Other Platform Operators

In the development stage of a platform, one-sided subsidies are often employed to prevent the "penguin effect" discussed in the context of network effect theory (see Sect. 2.2.2.1). In this stage, it often makes sense to license only one proprietary platform operator, to prevent freeloaders from taking advantage of one-sided subsidies (Eisenmann 2008). If the platform has reached a certain level of maturity; however, a strategic opening through the licensing of further operators can dramatically accelerate growth. This option is especially attractive when the additional operators apply their specific knowledge to provide innovative forms of the platform, which broadens the potential user base (Gawer and Cusumano 2002).

A successful example of this type of opening can be observed in the market for smartphone operating systems: The Android operating system is developed and provided by the Open Handset Alliance, led by Google. Android is licensed to numerous smartphone manufacturers, such as HTC, Motorola, Samsung, Dell, and Sony Ericsson, as an operating system for their devices. This strategy of licensing further providers (without relinquishing control over the development of the platform), seems to be a success: Android, at least for the time being, is recording strong growth rates, both in absolute terms and relative to its competitors.

Acquisition of Platform Sponsors

In addition, a platform can also be opened horizontally by taking on sponsors. In contrast to licensees, who base their specific extensions on the platform core, platform sponsors, are also involved in the ongoing technical development of the platform. We have already discussed the advantages of these types of development partnerships in Sect. 3.1.1.2. On the other hand, opening the platform to additional sponsors also increases the complexity of coordination between the sponsors and increases the effort required to specify common standards (West 2006).

Compared to licensing additional operators, adding to the number of sponsors also involves a considerable risk: In the worst case scenario, political disagreements between the sponsors could delay or completely impede development. According to West (2003), this type of opening should be pursued if the original

platform operator's business model focused not on licensing the platform, but on selling complementary products or services. In fact, pushing this kind of business was a key factor in IBM's decision to transfer its rights from the Eclipse development platform to the open source community. We will delve deeper into this topic in Chap. 7.

Another possible motivation for opening to new sponsors is when the platform is under considerable pressure from competing platforms. For example, in 1998, after losing the browser wars, Netscape disclosed the complete source code of its Netscape Communicator as part of the Mozilla Project, and opened itself to further sponsors. The result: the Mozilla Foundation produced the Firefox browser.

5.3.2 Management of Complementors

Senior management at software companies tend to focus on analyzing their own strengths and those of competitors, and frequently neglect to evaluate their supposed allies: providers of complementary applications. While platform operators are fully aware of their reliance on complementary functionality; especially those who run industry platforms, they often overestimate the extent to which their interests coincide. Even if, thanks to indirect network effects, both parties are equally eager for the platform to grow, their interests cease to accord when it comes to dividing up the profits.

For this reason, managing complementors is a key task for platform providers. In addition to an intensive analysis of complementors' business models, strategies, goals, skills, and motives, this also encompasses the selection of suitable paradigms for shaping the relationship between platform operators and complementors. Ney (2004) has devised two opposing paradigms, hard and soft power, which will be briefly described in the following section.

The most immediately obvious means for influencing complementors fall into the "hard power" category: By adopting a credibly threatening posture or dangling financial incentives, for example a share of revenue, platform operators hope to ensure that complementors toe the line. A real-life example of the use of hard power can be seen in Bill Gates' threat to cease the development of Microsoft Office for Apple's Mac OS, should Apple continue to refuse to integrate Microsoft's Internet Explorer in the Mac OS. These methods are generally underpinned by traditional sources of power, such as a large market share or exclusive control of a distribution channel. A platform operator can also exercise hard power and reduce complementors' independence by producing and providing strategically important complementary offerings itself. Smartphones are a case in point: Despite the trend towards coordinating a wide array of apps through marketplaces, vital core functionality, such as telephony or text messaging, remains integral to the platform. This strategy lets the platform operator realize economies of scale and generate additional earnings by selling complementary products. In addition, it can also be deployed to convey a clear message. However, especially when the

platform operator relies on the existence of a broad range of complementary offerings, this step can be counter productive: the message that the platform operator can and will encroach on the market and jeopardize complementors' sources of income may cause the latter to think twice about collaborating with this operator.

This reveals the disadvantages of using hard power: entering the market for complementary offerings incurs considerable ongoing costs. In addition, it prevents the development of a long-term relationship of trust with complementors. Furthermore, complementors will presumably try to avoid becoming too dependent on a particularly powerful platform operator, and may well bestow their long-term support on a competitor.

The alternative, namely exerting soft power, is generally a cheaper, and in the long term, more successful method of encouraging complementors to collaborate by pointing out shared objectives and opportunities. Concrete steps to achieve this include the pro-active communication of market data and plans for the future direction of the platform. Soft power can be wielded by proclaiming a common vision, which clearly highlights the advantages for complementors. One example of this, albeit from another industry, is Steve Jobs' efforts to integrate offerings from all the major music labels in his iTunes platform: In 2003, he articulated a compelling shared vision for the industry, and was able to persuade all labels to collaborate with him on terms that ensured the Apple iTunes Stores could offer attractive prices.

The disadvantages of this approach are that soft power is only successful as a long-term strategy, and an operator relying solely on soft power can be outflanked by a more aggressive competitor.

As shown above, both hard and soft power can be successful means to deal with complementors. Yoffie and Kwak (2006) have identified three factors that help to select the most suitable paradigm:

- Strength and dominance: The use of hard power in particular requires the deployment of considerable (generally financial) resources and a correspondingly strong position on the market. If a platform operator cannot meet these requirements, they are better advised to consider soft power. Small and seemingly weak companies can be especially attractive to external partners, as the latter do not have to fear that the operator will encroach on their territory.
- Diversity of complementary offering: If a platform operator is dependent on a large and diverse offering of complementary products, soft power is the better option, as this is the only way to ensure that the platform remains attractive to complementors over the long term.
- Specific investments by complementors: To integrate their offerings into the platform in the best possible way, complementors often need to make specific and irreversible investments. Where such investments are necessary, potential partners will try to guard against a breakdown in their relationship with the platform operator. As a result, it can be assumed that this type of trust is more likely to be built using soft power.

For platform operators, therefore, the question of hard or soft power is not an either-or decision, but rather a search for the ideal combination of the two. A middle path chosen by many operators is to restrict their exercise of hard power by producing only the most strategically significant complementary products themselves. Beyond that, they avoid intervening in the market for complementary offerings, or do so only in a very rudimentary way, for example in the form of quality controls. They will then use soft power to establish as stable and open a relationship with their partners as possible.