ISPMA

INTERNATIONAL
SOFTWARE PRODUCT MANAGEMENT
ASSOCIATION

Software Product Management

Excellence in Product Strategy

V.2.1 Student Edition

This syllabus was written by the following members of the International Software Product Management Association (ISPMA®): Hans-Bernd Kittlaus (editor), Georg Herzwurm, Marc Hilber, Barbara Hoisl, Mahvish Khurum, Katharina Peine, Karl Michael Popp, Krzysztof Wnuk.

We thank all honorary authors and contributors.

Preface

The goal of the International Software Product Management Association (ISPMA®) syllabus for "ISPMA Software Product Manager Excellence in Product Strategy" is to promote in-depth understanding of the *discipline of product management for software products* including the management of software parts of software-intensive products, i.e. systems or services, in the area of product strategy.

The syllabus "SPM – Excellence in Product Strategy" covers the full spectrum of elements of software product management related to product strategy that are well supported by literature and industrial practice. The syllabus corresponds to a 3-day industry course.

The syllabus addresses the needs of people involved in software product management and helps them to address the needs of people they interface with, e.g. general management, marketing and sales, research and development, service and support, and controlling.

The syllabus is the basis for examination to certify that the examinee has achieved the degree of knowledge described in this syllabus. The terms used in this syllabus are consistent with the glossary of the ISPMA®.

Purpose and structure of the Syllabus:

The syllabus is the basis for consistent training, learning, and examination of software product management. It provides:

- Explicitly phrased educational objectives for each chapter, and
- Informal explanations to detail the educational objectives.
- Informal references to literature (without limiting the interpretation of the syllabus to this literature only).

This syllabus consists of ten chapters. Each chapter covers one major educational unit (EU). Each chapter also includes the duration suggested to teach it. Each educational unit has educational objectives (EO) that are enumerated following the chapter headers (EO1.1.1, EO1.2.1 …). An educational objective has a defined cognitive level of knowledge that the course participant is expected to achieve. The numbering scheme for these objectives is aligned with the chapter numbering.

The educational objectives are expressed in terms of three cognitive levels of knowledge phrased using the verbs "knowing" for level 1, "understanding" for level 2, and "applying" for level 3. These three verbs are placeholders for the following:

- L1 (know): enumerate, characterize, recognize, and name.
- L2 (understand): reflect, analyze, justify, describe, judge, display, complete, explain, elucidate, elicit, formulate, identify, interpret, reason, translate, distinguish, compare, understand, suggest, and summarize.
- L3 (apply): perform, execute, develop, and adapt.

Each EO in the syllabus has one of the three cognitive levels assigned to it.

In order to address L3 objectives, ISPMA®'s Excellence syllabi are designed to put special focus on exercises. It is the trainer's responsibility to select exercises and to define concrete realistic scenarios in which all the selected exercises can be performed by the participants. ISPMA® recommends to spend about 50% of the available time on exercises. In trainers' material, exercises are described in abstract terms.

Included and excluded key areas:

The syllabus covers knowledge applicable for any kind of software systems and organizational contexts. A training course may cover more domain-specific details if needed by the course participants. This syllabus, however, does not provide guidance for such specialization, rather describes the base knowledge necessary, which can be complemented with domain specific items.

The syllabus is independent of any specific process model, and thus defines knowledge of a software product manager without prescribing exact interfaces to other roles in a product organization.

Training Courses:

The syllabus corresponds to a three-day industry course. The syllabus does not prescribe the specific form and approach of learning, however. It can also be administered with other forms of teaching or learning, such as self-learning supplemented by coaching or courses at universities or universities of applied sciences.

Training providers are encouraged to tailor training courses to the participants, and to add examples and an appropriate realistic scenario for the exercises described in this syllabus so that participants get an opportunity to apply the training contents. A participant should carefully choose the training provider. A list of training providers can be found on the ISPMA® web site www.ispma.org.

Examination:

The syllabus is the basis for the examination for the ISPMA® certificate "ISPMA® Software Product Manager Excellence in Product Strategy". All chapters are relevant for the exam. The exam takes the following form:

- Demonstration of knowledge with a multiple-choice test

Multiple-choice tests can be held immediately after a training course, but also independently from courses (e.g. publicly announced exams of the examination authorities). A list of accredited examination authorities can be found on the ISPMA® web site www.ispma.org.

Course participant prerequisites:

The training and learning of the syllabus assumes general knowledge of, and some experience in, the management or development of software products or software in software-intensive systems.
The formal background of the course participant is not crucial (whether it be engineering or management), rather the level of experience is predominantly the factor associated with the prerequisites. A course participant should have the ISPMA® certificate "ISPMA® Certified Software Product Manager – The Foundation" or at least three years of experience in software product management. However, this is a generic recommendation and might not be applicable for all situations or course participants.

Terminology

The term SPM is used as an abbreviation for Software Product Management. It represents the function of SPM and not the individual person.

This curriculum usually uses a gender-neutral form. In cases where the masculine form is used, this is done for readability reasons and represents any other gender as well.

The terms used in this syllabus are consistent with the glossary of the ISPMA® available at ispma.org.

Table of Contents

EU1 Introduction

Duration: 1:30 h

Educational Objectives:

EO1.1 Understand the role of product strategy, its elements and their interdependences.
EO1.2 Understand the role of product strategy within the SPM Framework.
EO1.3 Understand the business model concept and the business model canvas.
EO1.4 Understand the mapping between product strategy and business model canvas.

The primary objective of software product management is to achieve sustainable success over the life cycle of the product (or family or line). This generally refers to economic success, which is ultimately reflected by the profits generated. Since profits lag behind investments, i.e., an investment phase involving losses will be followed by an extended profitable phase, a longer-term perspective is appropriate. Therefore the role of the software product manager is sometimes referred to as mini CEO of their product.

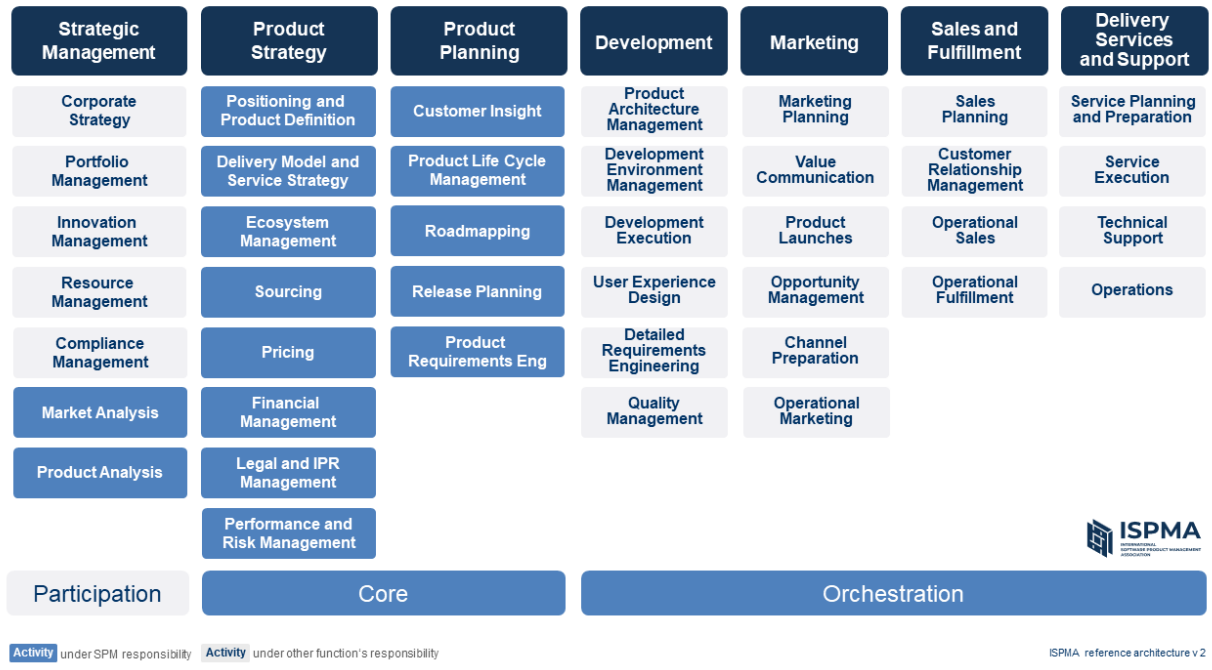

Fig. 1 SPM Framework V.2.0

Software product managers are responsible for defining the strategy for their product (or platform or family) and for supporting and updating it over time. Normally, a strategy covers a time span of about one to five years, however this varies dependent on the product's context, i.e. maturity, domain,

technologies, and market segments. The product strategy describes how the product is supposed to evolve over this strategic timeframe. The contents of the Product Strategy document is described in the ISPMA® "SPM – The Foundation" syllabus which follows the ISPMA® SPM Framework (see Fig. 1).

In the startup world, the development of a product strategy is a highly iterative process that aims at finding and optimizing the product-market fit. The steps of an iteration are as follows: hypothesis – MVP – test – conclusion, where MVP stands for minimum viable product which Eric Ries defined as "That version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort." Since the aforementioned definition of an MVP lacks a customer value perspective, we prefer the following definition: „The minimum feature set of a new product that is derived through a learning phase and that some customers are willing to pay for in the first release." Through this iterative process, development and implementation of a product strategy are intertwined.

When a product (and company) is successful and more mature, the product strategy tends to be more stable. An update of the product strategy is a more evolutionary process, and its implementation is more separated. Since the elements of the product strategy cover all product-related functional areas of the software organization, all of them need to be involved in and contribute to its implementation. That is shown in ISPMA®'s SPM Framework (Fig. 1).

<table>
<tr><td rowspan="2">Key Partners</td><td>Key Activities</td><td rowspan="2">Value Propositions</td><td>Customer Relationships</td><td rowspan="2">Customer Segments</td></tr>
<tr><td>Key Resources</td><td>Channels</td></tr>
<tr><td colspan="3">Cost Structure</td><td colspan="2">Revenue Streams</td></tr>
</table>

Fig. 2 Business Model Canvas (A. Osterwalder, Y. Pigneur (2010))

Syllabus Chapter	Business Model Canvas	ISPMA SPM Framework
EU3 Customer Segments and Value Proposition	Customer Segments Value Propositions	PS: Positioning PS: Product Vision and Definition PS: Delivery Model PS: Service Strategy
EU4 Partnerships and the Software Ecosystem	Key Partners	PS: Ecosystem Management PS: Sourcing
EU5 Channels <see also syllabus "SPM - Excellence in Orchestration"	Channels	PS: Positioning and Product Definition Orchestration of Marketing
EU6 Competition and Other Alternatives	./.	PS: Positioning and Product Definition
EU7 Revenue Streams	Revenue Streams	PS: Business Case PS: Pricing
EU8 Cost Structure	Cost Structure	PS: Business Case and Costing
EU9 Business Measures, KPIs, Risk Management	./.	PS: Performance and Risk Management
EU10 Legal Aspects	./.	PS: Legal and IPR Management
<not included, see syllabus "SPM – Excellence in Orchestration">	Customer Relationships	Orchestration of Marketing and Sales
<not included, see syllabus "SPM – Excellence in Orchestration" and syllabus "SPM – Excellence in Strategic Management">	Key Activities Key Resources	Orchestration of the Functional Areas Strategic Management: Resource Management

Fig. 3 Mapping of syllabus chapters, Business Model Canvas and ISPMA® SPM Framework (PS stands for Product Strategy)

A business model describes the rationale of how an organization creates, delivers and captures value by interacting with suppliers, customers and partners. It is often considered at the corporate or business unit level, but its consideration can also make sense on a solution level that spans multiple

products and services or on an individual product level. This syllabus primarily focuses on the individual product level, but we will touch on some solution-related topics where products cannot be considered stand-alone. The Business Model Canvas is one of the tools that are used to review and challenge existing business models and systematically invent new ones that change the way a product competes (see Fig. 2). The canvas consists of 9 segments, on the right-hand side those representing a more market-/customer-facing view (e.g. customer segments and revenue streams), on the left-hand-side those representing a company-internal view (e.g. key resources and cost structures), with the value proposition linking the two sides or views.

The structure of this syllabus "SPM - Excellence in Product Strategy" combines the elements of the business model canvas with the product strategy aspects from the ISPMA® reference framework. All these items are highly interdependent. As Fig. 3 shows the mapping works to a large degree, but not perfectly. The Business Model Canvas does not address some important strategy aspects like Competition, Business Measures, Risk Management and Legal Aspects. It does contain Customer Relationships, Key Activities and Key Resources which are neither part of product strategy nor under the direct responsibility of software product managers, and will therefore not be considered in this syllabus.

As pointed out in ISPMA®'s syllabus "SPM - The Foundation", software product management covers not only software products, but also software in software-intensive systems and services. To support ease of reading this syllabus will generally use the term "software products" instead of explicitly referring to software-intensive products and services as well. We use the term „service" only for services performed by humans.

Some product strategy elements are only applicable on a product level, e.g. pricing. The relationship between product strategy and corporate or company strategy is described in ISPMA®'s syllabus "SPM – Excellence in Strategic Management".

Software differs from products of other industries in a lot of ways (see ISPMA®'s syllabus "SPM – The Foundation", section 1.1). These differences have a significant impact on the contents of a software product's strategy.

A special case is the management of platforms (see ISPMA®'s syllabus "SPM – The Foundation", section 1.1), e.g. the business model is highly complex and requires ongoing balancing, in particular with multi-sided transaction platforms. Ecosystem management requires significantly more attention compared to non-platform products since achieving win-win-situations with a high number of stakeholders is a key success factor for a platform.

Literature: *Kittlaus, H.-B. (2022, p. 51 ff); McGrath, M. (2000); Osterwalder, A., Pigneur, Y. (2010); Ries, E. (2011); Ries, E. (2009); Kittlaus, H.-B. (2020); Cusumano, M., et al. (2019); Parker, G.G., et al. (2016)*

EU2 Business Models in the Software Industry

Duration 1:30 h.

Educational Objectives:

EO2.1 Understand business models and their structure.

EO2.2 Be able to formulate business models for software companies and products.

EO2.3 Know the business models of some prototypical software companies, their products and their differences.

EO2.4 Understand relationships between business model, revenue streams, activities and resources.

A business model of a company or business unit describes which products and services are offered, which value they offer to which market segments, and how cost and revenue streams relate to the different products and services. The business model concept can also be applied to an individual product.

Business models can be described in three dimensions: types of products/services provided, business model archetypes used and revenue streams. Each of these dimensions leads to particular entries in the business model canvas.

1. **Types of products/services** can be
   - financial products (cash and other assets),
   - physical products (real, physical products, durable and non-durable goods),
   - intangible products (software but also other intellectual property, knowledge and brand image) and
   - human services (people´s time and effort).

2. **Business Model Archetypes** are basic patterns of doing business (see *K.M. Popp, R. Meyer (2010)*). Available archetypes are creator, distributor, lessor and broker.
   - A **creator** uses supplied goods and internal assets and transforms them to create a product sold to customers or used by other creators. The main work done by the creator is designing the product, e.g. Apple as a creator designed the iPod in California.
   - A **distributor** buys a product and resells the same product to customers. Obvious examples are companies in the wholesale and retail industries, like Sears or Saks, or stores that sell shrink-wrapped software.
   - A **lessor** provides the temporary right to use, but not own, a product or service to customers. Examples are landlords, lenders of money, consultants and software companies that license their software to customers. For human services, HR lessors lend their employees´ time to customers.
   - A **broker** facilitates the matching of potential buyers and sellers. A broker never takes ownership of the products and services. An example is a traditional stock broker. A software-based example is ebay, which provides a platform that matches buyers and sellers.

3. **Revenue Streams**: these are covered in EU7.

A business model is constructed by choosing one or more combinations among these three business model dimensions. Products and services can be offered standalone, or they can be offered as a bundle. Software vendors are typically offering intangible products and act as a creator and lessor of software, but also offer human services like delivery services, maintenance and support. E.g. a cloud business model offering a service to the customer has the software vendor acting as creator, lessor and broker. In addition, a service to operate the software products is offered. These different products and services are bundled into a cloud service offering, and there usually is a revenue stream compensating for the cloud service offering.

The relationship between the three dimensions and the canvas areas are as follows:

- The relationship between products/services and revenue streams: Single products/services or bundles are compensated by the customer. A compensation can be monetary (revenue streams) or non-monetary (exchange of products, services or information). So for each product there always is a compensation, which in many cases is a (financial) revenue stream.
- Single products/services or bundles have a corresponding value proposition.
- For each type of product/service and for each type of bundle there is a specific cost structure.
- Business models are executed by activities, some of them being key activities. Activities are carried out by resources, some of them being key resources.

The flexibility of software in combination with human services opens up a wide space for creative business model innovation, in particular in connection with the internet and mobile services. This ongoing innovation has increasingly led to companies using different business models for different products.

Literature: *Kittlaus, H.-B. (2022, p. 22 ff); Buxmann, P., Hess, T., Diefenbach, H. (2012); Popp, K.M. (2011a); Popp, K.M., Meyer, R. (2010); Weill, P., Malone, T.W., D'Urso, V.T., Herman, G., Woerner, S. (2005)*

EU3 Customer Segments and Value Propositions

Duration: 2:45 h

Educational Objectives:

EO3.1 Understand the importance of clearly defined target customer segments.
EO3.2 Be able to use appropriate methods to gain a deep understanding of customer needs within a customer segment.
EO3.3 Be able to translate the understanding of customer needs into compelling value propositions.
EO3.4 Be able to develop a positioning statement based on customer segment, value propositions, and understanding of competitive products.
EO3.5 Understand that add-on products, integrations, and services may be needed to actually deliver the compelling value proposition (whole product concept).
EO3.6 Understand key considerations for choosing product names, domain names, and landing page URLs and the importance of trademark research and trademark registration.

Whether evolving an existing business model or creating a new one, the business model generation process often starts with the two canvas segments called “Customer Segments” and “Value Propositions”. These two segments are closely linked through the underlying concept of what customers need or want, specifically their pains, their desired gains, and their “jobs to be done”.

In this chapter, we look at these two canvas segments and their underlying linkage point, the customer needs, all three of which are valuable and important elements of the product strategy. A good basis for this analysis is a strong product vision that helps to convince and engage all stakeholders.

The product vision is the “guiding star” or “North Star” for the strategy and the product team. It outlines in a condensed form:

- conceptual image of what the future product will be,
- the customer value proposition that says why the product is needed and cannot be replaced by an alternative, and
- the business value for the vendor, i.e., why it will be successful.

A product vision describes the future state of the product, to be reached by the end of the strategic time frame, or even later. It is documented in a few sentences or as a relatively short text, no more than one page. The vision needs to be phrased from a marketing perspective, in a style that has a motivating effect on stakeholders inside and outside of the company by painting a desirable, ambitious, but achievable future. The product vision is especially important in the start phase when the first version of the product is conceived and developed.

The elements of the product strategy provide the details that turn the vision into a manageable and executable path into the future. In bigger companies a product vision needs to be aligned with the corporate vision.

Customer Segments

Segmentation can be done based on end users or based on sales/delivery channels. Product managers need to be very specific about the customer segment(s) they are targeting so that they can develop a thorough understanding of customer needs. This understanding in turn is key to developing compelling value propositions that help sell the product. Here methods like the Value Proposition Canvas can help (*Osterwalder/Pigneur (2014)).*

However, this does not mean that customer segments always have to be very narrowly focused – according to Moore it depends on the maturity stage of the respective market how broad or narrow the target segment can be.

For example, when bringing to market a completely new type of B2B product – what Moore calls a new product category – it is often useful to kick-start mainstream adoption by focusing on a small, well-defined market niche (the beachhead segment). The strategy is to expand into adjacent market niches later – one after the other (bowling alley strategy). Once the new product category is better understood in the market and broad adoption sets in at a fast pace (tornado phase), the vendor's top priority needs to be capturing market share. In this stage, an undifferentiated strategy is suitable *(G.A. Moore (2014); G.A. Moore (2004)).*

Value Propositions

Product managers need to build deep customer insight, understanding the problems and the environment in which customers operate (see syllabus "SPM – The Foundation" , EU 3.2). This customer insight is a prerequisite to create value propositions that (hopefully) strongly resonate with the customer segments and help sell the products and services. According to (*Osterwalder/Pigneur (2010)*), the following elements contribute to compelling value propositions:

- **Pain relievers** "eliminate or reduce negative emotions, undesired costs and situations, and risks your customer experiences or could experience before, during, and after getting the job done".
- **Gain creators** "create benefits your customer expects, desires or would be surprised by, including functional utility, social gains, positive emotions, and cost savings".
- **Products and Services** lists the products and services that together deliver the value proposition.

Especially in B2B technology markets, including B2B software, it is very common that core products require add-on products, integration with partner products, and accompanying services as a minimum configuration to deliver a compelling value proposition. Geoffrey Moore (Moore 2014) emphasizes the importance of this – he calls this the *whole product* concept. Based on the value propositions, software product managers need to identify which additional products and services are required to deliver the whole product, and they must make sure these crucial whole product components are actually available to customers – either from the vendor itself or from partners. This needs to be reflected in the Service Strategy. Partners that contribute important whole product components need to show up in the canvas, either in the channels section (see EU5) or in the key partners section (see EU4).

In-depth understanding of customers and strong value propositions also help with positioning: combined with an understanding of competitive offerings (see EU6), they help product managers to arrange for their product "…to occupy a clear, distinctive, and desirable place relative to competing products in the minds of target customers." (*P. Kotler, G. Armstrong (2015)*)

Product Definition

The value proposition needs to have a foundation in a corresponding product definition. The product definition needs to define (see syllabus "SPM – The Foundation"):

- Functional scope
- Quality scope
- Intended use and users
- User experience (UX) design scope
- Offering architecture
- Business architecture (for application software)

Literature: *Christensen, C.A. (2013); Haines, S. (2014, chapter 8); Kittlaus, H.-B. (2022, p. 79 ff); Kotler, P., Armstrong, G. (2015); Moore, G.A. (2014); Moore, G.A. (2004); Osterwalder, A., Pigneur, Y. (2014); Osterwalder, A., Pigneur, Y. (2010); Pruitt, J., Adlin, T. (2006); Dunford, A. (2019); Kittlaus, H.-B. (2019); IL10*

EU4 Partnerships and the Software Ecosystem

Duration 2:30 h

Educational Objectives:

EO4.1 Understand software ecosystems and their structure.
EO4.2 Know the relevant stakeholders of a software ecosystem.
EO4.3 Know typical relationships in software ecosystems, in particular partnerships.
EO4.4 Know strategies of different companies in software ecosystems.
EO4.5 Be able to make decisions based on the company's ecosystem strategy.

Software ecosystems have a significant impact on the work of the product manager. The product manager must conform to the role the software vendor wishes to play in the ecosystem which has direct influence on the positioning of the product, the pricing strategy, the degree to which the product road map and requirements decisions depend on other players in the ecosystem. In addition to this, a product manager can give input as to how the ecosystem strategy works, and suggest opportunities for the evolution of that strategy. In cooperation with Marketing and Development, the product manager may enter partnerships with some players that are or become part of the ecosystem.

Definitions

A software ecosystem is defined as a set of businesses functioning as a unit and interacting with a shared market for software and services, while maintaining beneficial relationships. These relationships are frequently underpinned by a common technological platform – what Cusumano e.a. (2019) call innovation platform – and operate through the exchange of information, resources, and artifacts. Organizations can take on different roles in ecosystems, such as keystone, dominator, or niche player. If the platform is owned by a company the success of that company is not only defined by its own success, but by the success of its ecosystem.

Relationships in software ecosystems

Relationships in a software ecosystem describe the interactions between different companies. The companies in the software ecosystem interact with the central platform company or its customers or partners:

- Competitors: they sell products or services to the platform company´s (potential) customers that are competitive to the platform company´s products.
- Acquisition targets: Companies that are potential candidates for acquisition or IP purchase by the platform company.
- All the channel partners whose relationships are described in EU5.
- Other software vendors who provide components, platforms, plug-ins, add-ons etc. (may be partners)

- Influencers like market research companies, journalists, consultants etc.
- Customers, including OEM customers: they license or subscribe to the platform company´s products for internal use or for inclusion in their own products.

Strategies in software ecosystems

Ecosystem strategy defines how a company will deal with its ecosystem. Three specific strategies are popular in ecosystem literature: niche players, and - as platform vendors - keystone players and dominators.

Niche players

In software ecosystems, most companies follow a niche player strategy. They focus their business on critical competencies in narrow areas of expertise, if there is an opportunity to run a profitable business. They usually are smaller companies and outnumber keystone players and dominators. An example for niche players are app developers in the ecosystems of smartphone platforms like Google's Android or Apple's IOS.

Keystone players

Keystone players provide the core of the innovation in an ecosystem. For technology-related ecosystems, this effect of keystone players is a key success factor for survival and adaptability of the overall ecosystem. An example for a keystone player is Google in the Android ecosystem.

Keystone players behave in favour of other players, especially by protecting niche players. The number and diversity of niche players determine the speed and diversity of innovation in an ecosystem, which is an important prerequisite for success.

Dominators

A Dominator is pretty much the opposite of a keystone. A dominator leverages a critical position in the ecosystem to exploit or take over a large portion of the ecosystem.

If a dominator progressively takes over the ecosystem by occupying an ever growing number of niches, this is called physical domination. Dominators typically damage the long-term health of the ecosystem. Especially in ecosystems that are impacted by fast technological changes, a dominator strategy is a questionable strategy. A dominator has to provide all the innovation in all occupied areas which means taking all the cost of innovation without having the diversity and evolutionary power that is created by an ecosystem. There is also the danger that the innovative power of the dominator decreases with the size of a company. An example for a dominator is Apple who controls the IOS ecosystem to a large degree.

Partnerships

Partnerships only make sense when they are beneficial to both parties. The software product manager or a partner management organization need to ensure this win-win-character when designing terms and conditions of a partner program or negotiating individual partnerships. Sourcing decisions can include the selection of partners, both on the development and the marketing side. Partnerships tend to be very important in the software business since few companies can implement the "whole product" approach by themselves, but rather need partners to provide missing elements.

Literature: *Jansen, S., Cusumano, M., Brinkkemper, S. (2013); Kittlaus, H.-B. (2022, p. 111 ff); Kude, T. (2012); Messerschmitt, D.G., Szyperski, C. (2003); Meyer, R. (2008); Popp, K.M. (2010); van Angeren, J., Kabbedijk, J., Popp, K.M. (2013); Schütz, S., Kude, T., Popp, K.M. (2013); Borg, M. e.a. (2019); M. Cusumano et al. (2019)*

EU5 Channels

Duration 1:15 h.

Educational Objectives:

EO5.1 Understand the role of Software Product Management with regard to the channel strategy.

EO5.2 Understand the nature and characteristics of channels.

EO5.2.1 Know the offered "whole product" concept of IT products, the business relationships and delivery models.

EO5.2.2 Know the contract types related to the delivery models.

EO5.2.3 Know the sales channels for IT products, differentiate between human and virtual channel.

EO5.2.4 Be able to develop a comprehensive sales channel strategy.

Channels define the path through which goods and services as well as the compensation are transferred between vendor and customer. Compensation can include financial payment as well as payment in other goods and services, e.g. the disclosure of user data as in Google's search service (see also EU6 – Revenue Streams).

As part of the product strategy work, the software product manager needs to make decisions in cooperation with Marketing and Sales which channels are to be used for the software product. The operational channel management is typically under the responsibility of Marketing and Sales. Required input for the selection of channels is:

- Definition of the "whole product" being offered, i.e. required additional products and services to deliver a complete solution (see section EU3). An example are *Software as a Service (SaaS)/Platform as a Service (PaaS)* offerings that consist of the software plus operations plus maintenance.
- Definition of target market (segments), for example:
  - *Business-to-Consumer* (B2C): A software product is sold by a company to a consumer, e.g. online retail business.
  - *Business-to-Business* (B2B): A company offers its software product to another company, e.g. an HR software product.

  Multi-sided platform: A company provides a platform on which different parties do business or exchange information. In this case, the target segments need to be defined for all parties doing business or exchanging information through the platform.
- Definition of delivery model:
  - On-premise, where the customer acquires a license for the software product and installs and runs it in their own operating environment.
  - SaaS/PaaS, where the responsibility for the operating environment is with the vendor based on a service contract, which includes operations and maintenance.

Characteristics of sales channels for software products

There are two main categories of *sales channels* for software products:

1. *Virtual channels*:

Internet sales has become a major channel, where software products are sold on the web only (e.g. web-based apps, sales through app stores, in-app sales etc.). This works with both license products, SaaS products, and advertising. Customers buy conveniently and easily from the comfort of their home or office at any time.

2. The *physical (human) channel*:

2.1 *Direct sales* means that products are marketed directly by the company to customers, eliminating the need for external intermediaries, such as wholesalers or retailers. An example is a direct sales force, where the company's sales representatives have in-person contact to customers.

In such a setup, the company will often assign key account managers to large existing customers. The key account managers are responsible for cultivating the customer relationship to drive repeat business with those customers. This is called "farming". For global players it can be useful to implement "global account management" to be present wherever the customer is.

In contrast, "hunting" is about recruiting new customers from the defined target market.

Since the customers no longer seek information technology as such, organizations should provide IT solutions that improve the business and describe advantages and positive effects. Direct selling often requires IT sales specialists to communicate the resulting financial benefits to customers.

Telesales (or inside sales) means that a sales person contacts customers to sell software products, either via telephone or through a subsequent face to face or web conferencing appointment scheduled during the call. Telesales is still being used, especially in B2B software sales.

2.2 *Indirect sales* means that selling of products is conducted by partners, e.g., companies such as retail shops or wholesalers. There are different types of partners:

- *Original Equipment Manufacturers* (OEMs): "The term "OEM" was originally coined in the hardware business and later transferred into the software world. It means that one manufacturer sells one of his products to another manufacturer who uses it as a component in one of his products without showing its origin openly" (Kittlaus, Fricker 2017, Ch. 2.2). An example is a database company licensing its software to a vendor of business application software to be included "under the hood" with the business application software. Here, the business application software vendor is the OEM whose brand name is visible, while the brand of the database vendor is less visible or entirely invisible. From the point of view of the database vendor, the business application vendor is one of their sales channels. The relationship can be considered as a type of sales and distribution outsourcing.
- A *service provider* is specialized on human services that he offers to customers and which can be product-related.
- A *reseller* sells software products offered by a software vendor. The reseller has contractual relationships with the customers and with the vendor.
- A *Value Added Reseller/Remarketer* (VAR) is a reseller that offers a software product and adds components and/or services to them, e.g. customer-specific customizing.

- *System integrator* (SI) coordinates and performs the integration of software product components supplied by different vendors and customer-specific software. This includes the responsibility for the overall system design as well as the integration of product and service components. He has to standardize interfaces, integrate different technologies and product supply.
- *Independent Software Vendor* (ISV) develops and distributes software products that may complement another vendor's product. He is independent in the sense that he is not controlled or owned by another vendor.
- *Intermediary* mediates between vendors and customers of software products, with the aim of them signing a contract, e.g. a provider of an internet marketplace.

The advantage of the indirect sales channel is that more potential customers can be reached through the partners and their customer relationships. Decisions are required whether a one- or multi-tier partner model is appropriate, and how channel conflicts (several companies competing to sell the same product to a given customer) can be avoided with a multi-channel approach. On an operational level, the management of the selected partners is part of ecosystem management described in EU8.

A further classification of channels differentiates

- Free versus paid: *Free channels*, e.g. social media or blogging, contain inherent costs (non-zero human capital cost). In contrast, paid channels like search engine marketing require explicit investment.
- Inbound versus outbound: *Inbound channels* rely on being found by the customer (pull messaging, e.g. blogs, e-books, and webinars), while *outbound channels* reach for customers (push messaging, e.g. trade shows, cold calling).
- *Direct on the Internet,* typically through the web page of the software vendor vs. *indirect on the Internet*, e.g. through a market place for software (such as STEAM for games), or through an app store (e.g. iTunes, Google Play)

Sales Channels Selection

The selection of appropriate channels depends on the concept of the offered software products and the definition of the target market.

The sales channels a company uses for a particular product are selected according to specific criteria:

- Target market (segments): Which channel has the highest potential to reach customers in the selected target market?
- "Whole Product" definition: Can partners who provide components act as channels?
- Sales cost: What are the cost and benefits the selection of a particular channel entails.

Sales cost is a key criterion for selection of sales channels. Sales channels with human involvement are more expensive than the virtual (i.e. fully automated) sales channels.

The most expensive sales channel is usally the direct sales force of a software vendor. Therefore, this channel is only financially viable for software products in the high price segment. Software products at lower prices are rather distributed through indirect sales channels with human involvement, or through virtual channels (Internet sales, fully automated).

In addition, other factors should be considered as part of a comprehensive channel strategy:

- the relationship frequency: some channels are used systematically and repeatedly, others opportunistically (one-offs),
- the place of purchase (online retailer versus software retail store around the corner)
- the strategies in software ecosystems, i.e. niche players, keystone players, or dominators (see EU8 Partnerships)
- the purchase frequency or the degree of willingness to buy (impulse buyers versus regular customers)
- the purchase occasion
- the attitude towards the product or the service
- the usage rate/ frequency of the product

The selection of channels usually results in a mix that can include direct sales including telesales and virtual sales, and partner sales. The terms and conditions for the channels need to be defined such that channel conflicts are prevented. The operational responsibility of managing this marketing mix on an ongoing basis is with Marketing and Sales.

Literature: *Doshi, P. (2011, p. 1); Herzwurm, G., Pietsch, W. (2008); Kepes, B., Subramanian, K. (2010, p. 1); Segetlija, Z., Mesarić, J., Dujak, D. (2011); Daviesa, A., Bradyb, T., Hobday, M. (2007, p. 183 f.); Gartner (2013, p. 1); Kittlaus, H.-B. (2022, p. 81 ff); Maurya, A. (2012, p. 17 ff.); Popp, K.M., Meyer, R. (2010); Bech, H.P. (2015)*

EU6 Competition and Other Alternatives

Duration: 1:30 h

Educational Objectives:

EO6.1 Understand the importance of identifying competitive advantage of product/service.
EO6.2 Understand methods for identifying competition.
EO6.3 Understand the impact of findings from competitive analysis on product management decisions, such as positioning, product definition, pricing, and roadmapping.
EO6.4 Be able to apply methods to identify competition.
EO6.5 Be able to develop a competitive strategy.

Competition is not explicitly represented in the business model canvas, but is an important topic for product strategy. Part of forming a successful product strategy requires identifying competitive advantage. A competitive advantage is achieved when the company offers a product that the competition does not, or when the company offers a better solution to the customer than the competition. Providing an answer to the question “How can the product satisfy the needs of potential customers better than competition?” needs to be incorporated in the value proposition. Different methods can be used to identify competition like the Industry Method or the Market Method.

It is important to consider

- Direct competitors who produce a virtually identical product that is offered for sale within the same market (e.g. listed in the same product category by industry analysts or media)
- Indirect competitors who don't necessarily sell the same products, but offer different alternatives to satisfy the same customer need (different product category, solving the same customer problem in a different way)
- Other alternatives customers have, i.e. do-nothing, or do-it-yourself (manual process or home-grown software solution)

Once the alternatives have been identified, gap analysis (surveys with the customers, distributors and partners) can be employed to analyze competitors, e.g. the strategic groups method . However, if the companies want to dramatically improve their value proposition, they need to identify offerings (not competing companies) that fulfil the same need (maybe in a different way).

Unless there are specialists in the organization for market research and competitive analysis, these are part of the product manager’s responsibilities (see SPM - Excellence in Strategic Management). A product manager usually needs needs to cooperate with other roles like senior strategy managers, marketing managers, or sales representatives to analyze competition and define competitive strategy for the respective product.

A competitive strategy is defined as a long-term action plan that is devised to help a company gain a competitive advantage over its rivals. It consists of business approaches to attract customers (by fulfilling their needs), withstand competitive pressure and strengthen market position. A competitive strategy exploits competitive advantage by identifying ways to use resources and capabilities to

differentiate the product from its competitors. It is always work-in-progress, i.e. over time the competitive strategy as well as the product itself need to be changed and adapted in order to maintain or gain competitive advantage.

Regarding competitive strategy, a well-known model identifies five forces that shape competition within an industry (*Porter's 5 Forces model*):

1. Threat of new entrants
2. Bargaining power of suppliers
3. Bargaining power of customers
4. Threat of substitute products/offerings
5. Rivalry among existing competitors

According to Porter, when formulating a competitive strategy, two dimensions need to be considered

- the source of competitive advantage
  - cost leadership (cost! not price) or
  - differentiation
- scope of the target market
  - targeting the broad market or
  - focusing more narrowly on a niche of the market

By combining these two dimensions, there are *three generic competitive strategies* that can lead to success:

- Cost Leadership – addressing the broad market, while having the lowest cost base
- Differentiation – addressing the broad market, with product/offering that is perceived as unique in the industry (for example premium / luxury product)
- Focus strategy – be the niche leader, addressing the specific needs of the niche better than the products targeting the broad market

While Porter's model is widely used, its focus is heavily on the costs of manufacturing products or costs of delivering human services. Therefore, for software markets that are fast-changing and where manufacturing costs are not an issue, other strategy models are needed.

More recent strategy models emphasize the ability of companies to actively shape market boundaries and thus, create new markets. For example the *Blue Ocean Strategy* approach asks vendors to

- Create uncontested market space,
- Make the competition irrelevant,
- Create and capture new demand (innovations),
- Break the cost/value trade-off
- Align the whole system of a company's activities in pursuit of low cost and differentiation.

However, the two approaches are not mutually exclusive and can be combined. For example, by slowing down profit erosion with an effective competitive strategy for an existing market, a company can increase the funds available for Blue Ocean investments and consequently increase its chances of finding an untapped, highly profitable market.

Literature: *Burke, A., van Stel, A., Thurik, R. (2010); Coulter, M. (2012); Fleisher, C., Bensoussan, B. (2015); Kim, W.C., Mauborgne, R. (2015); Kittlaus, H.-B. (2022, p. 62 ff); Porter, M.E. (2008); Porter, M.E. (1986); Tan, Y.S., Fraser, N.M. (1998); IL5-10*

EU7 Revenue Streams

Duration: 2:30 h

Educational Objectives:

EO7.1 Understand why software product managers have a particularly high degree of flexibility when defining revenue streams.
EO7.2 Understand hybrid revenue models and their use in the software industry.
EO7.3 Understand four key attributes of revenue streams and their definition: compensator, effect, rating, and charging.
EO7.4 Be able to apply the strategic pricing pyramid, in particular: price structure (metrics and fences), pricing policy, and price level.
EO7.5 Understand key static and dynamic concepts for pricing strategies.
EO7.6 Be able to build revenue models based on two different approaches: extrapolation from the past vs. bottom-up model.
EO7.7 Be able to use a bottom-up revenue model to test different pricing strategies, to run what-if-analyses, and to uncover inconsistencies in the business model.

Software products are usually characterized by low variable cost (low cost of revenue). Therefore, software product managers can choose from a much broader selection of business models than product managers responsible for products with high variable costs, such as many physical products or labor-intensive services. This higher degree of freedom also extends to the realm of possible revenue models.

Revenue Model Basics

According to *Popp (2011b)*, a business may combine multiple revenue models, and each revenue model can rely on multiple revenue streams (see Fig. 4). If multiple revenue streams are combined in a revenue model, this is called a *hybrid revenue model*.

Hybrid revenue models are very common in the software industry: for example, the classic software license model often combines a license revenue stream with support and maintenance revenue, plus revenue from other product-related services, such as installation and customization services, often called delivery services.

Revenue streams are characterized by the following four attributes:

- **Compensator**: who provides the compensation?
- **Effect**: the type of compensation, including
  - payment
  - no compensation, e.g. usage of open source software
  - compensation in other goods or services, e.g. a Google user compensates Google for its search service by providing information about his areas of interest

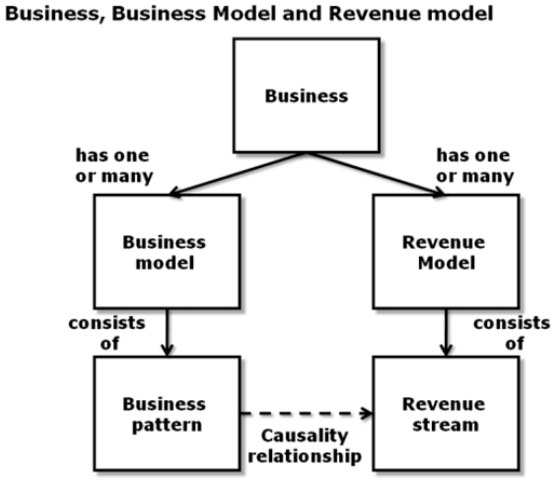

Fig. 4 Business Model and Revenue Model

Fig. 5 Strategic Pricing Pyramid (Nagle, Hogan (2005))

- **Rating**: how is the consumption of goods or services measured?
  - time-based, for example usage for 1 month,
  - usage-based, e.g. gigabytes of storage, number of unique or concurrent users that are permitted to use the product,
  - functionality-based,
    e.g. silver, gold, platinum editions with increasing functional scope
- **Charging**: how is the compensation amount for a certain rating of goods and services determined?
  For example, charging a fixed fee per user per month for using the software.
  This also includes different options regarding the frequency of payment:
  - Recurring revenue from regular business: support services, rental models, IP licensing
  - One-time revenue types from regular business: perpetual license
  - One-off revenue (not from the regular business): selling the IP, spin-offs, …

Strategic Pricing

The concepts presented so far can be used to develop a pricing strategy according to the strategic pricing pyramid (see Fig. 5). Strategic pricing starts with a clear understanding of customer segments and value delivered to the customers (see EU3). Based on that, the metrics used in the price structure can be determined (compensator, effect, rating and charging from the model above). An important consideration is that metrics should mirror the generation of customer value – this is a key factor in increasing the acceptance of value-based pricing.

Strategic pricing also includes processes and policies to ensure the integrity of the price structure in the market, for example fences that prevent abuse of discounts (e.g. student discounts require proof of student status) or criteria for handling “exception requests” in price negotiations (discounting criteria as part of the pricing policy layer in the pyramid).

Regarding overall pricing concepts, we distinguish:

- Static pricing concepts: Price structure and level are not changed over longer periods of time.
- Dynamic pricing concepts: Price changes at high frequency based on current demand, order inventory, etc.

There are a high number of pricing strategies to choose from:

- Premium pricing price strategy (high price, high quality and image) and promotion price strategy (lower price, high quality, e.g. temporary special offers)
- Price differentiation strategy (same product, different prices, e.g. in different market segments – temporary, geographically, personnel, quantitatively)
- Price bundling (single pricing or product package)
- Penetration strategy (low price with the introduction of products for rapidly gaining market share) vs. skimming price strategy (high price for innovative products for compensation of usually high investment)
- Life-cycle-dependent pricing strategy (situation-specific decision if, e.g. in the introduction phase, high or low prices are to be set)
- Yield Management (primarily used with services - by controlling prices and quantities, demand should be smoothed, e.g. early booking discounts)
- Dynamic, non-linear pricing strategy (usage independent component, e.g. fixed charges, and usage based component, e.g. depending on usage)

Building a Revenue Model

The revenue model is a key input for building a comprehensive financial business case for a software product – the other key input being the cost model (see EU8 Cost Structure).

Before starting with the revenue model, software product managers should work with their finance liaison to make sure they understand which revenue recognition guidelines are applied in their organization: when exactly revenue can be recognized is especially tricky for software, varies somewhat between different accounting standards (e.g. IFRS which is heavily used in Europe vs. US-GAAP), and also depends on charging details (e.g. one-time license charge vs. recurring charges for a SaaS offering or a support contract) and bundling (e.g. pure software license vs. software bundled with professional services).

There are two common approaches to build that revenue model: extrapolation from the past vs. bottom-up. Note that extrapolation from the past only works when there is a past to extrapolate from, i.e. either for existing products, or if there is a very close analogon we can use.

The bottom-up model requires a model of how the customer base is building up over time. It uses the established pricing strategy, as well as other input data, such as

- Current user and revenue base (in case of an already existing product),
- Results from market analysis (market size, trends, growth rates, ….),
- Planned sales channels,
- Planned investments in sales, marketing, customer acquisition efforts,

- Experience values regarding channel effectiveness, ROI (return on investment) of customer acquisition expenses, or historic sales ramp-up (adoption curves) for similar products.

A bottom-up revenue model requires more effort to build, but it provides additional benefits: it can turn into a useful tool for assessing the viability of a planned sales approach, and to run what-if analyses for changes in pricing.

For example, for a product that is sold via the web channel, A/B tests may be used to study the impact of different price metrics, discounts, or price levels on buying behavior and conversion rates. Results from these tests can then easily be plugged into the revenue model to determine the financial impact of the pricing changes and changes in conversion rates.

Literature: *Ries, E. (2011); Sheen, R., Gallo, A. (2015); Popp, K.M. (2011b); Osterwalder, A., Pigneur, Y. (2010); Kittlaus, H.-B. (2022, p. 90 ff); Nagle, T.T., Hogan, J.E., Zale, J. (2014); Schwind, M. (2007)*

EU8 Cost Structure

Duration: 2:00 h

Educational Objectives:

EO8.1 Understand why software product managers need to build business cases.
EO8.2 Understand the structure of an income statement and the major cost drivers for software products.
EO8.3 Understand the concept of “low marginal cost” and how this enables the business models that are unique to the software industry.
EO8.4 Know the target costing approach and when to use it.
EO8.5 Be able to build a financial model.
EO8.5.1 Be able to structure a financial model so that “what-if” analysis can be done.
EO8.5.2 Be able to collaborate with the finance liaison for financial modeling.
EO8.5.3 Be able to collaborate with other functions to build meaningful cost models, e.g. for R&D, marketing etc.
EO8.6 Know several advanced economic efficiency assessments that can be conducted based on the financial model.

Since product managers are expected to manage their respective product as a business, not only the product revenue streams (see EU7 Revenue Models), but also the associated costs need to be managed. Based on understanding both revenue and costs, product managers can build a full business case. By business case we mean a decision support approach in which investments and benefits are quantified and compared. The business case helps justify investment decisions for a new product, and it supports profitability tracking and business model evolution for existing products. A business case can make sense on different levels, from the product level, which is our focus here, down to a requirement level.

The Income Statement Approach

A financial business case for a software product is typically structured similar to the income statement (also known as profit & loss statement, or P&L) for an entire company (see Fig. 7).

For software products, the income statement is typically structured as shown in the picture, with costs broken down into several categories. The first category, costs of revenue, is highly variable. Apart from cost of revenue, we usually distinguish between four other categories of operating expenses that are “fixed”, i.e. they will not move immediately in sync with a short-term revenue spike or revenue decrease:

- Research & development (R&D),
- Sales & marketing,

- General & administrative (G&A),
- Other operating expenses (such as asset depreciation).

Fig. 6 shows the full income statement for a software company (see for example the annual or quarterly reports of companies like SAP, salesforce.com, Adobe, Microsoft, or Google).

Income Statement – Cost Structure

	Revenue	The "top line"
–	**Cost of revenue** or Cost of Goods Sold (COGS)	Revenue-related costs: parts, labor, e.g. support engineers
=	**Gross Profit**	
–	**Other Operating Expenses**	• Research & Development (R&D) • Sales & Marketing • General & Administrative (G&A) • Other operating expenses, e.g. depreciation of assets
=	**Operating Profit**	Profit from ongoing operations
–	**Non-Operating Expenses**	Interest paid or earned, taxes, ...
=	**Net profit**	The "bottom line"

Fig. 6 Income Statement

For an initial assessment of the profit potential for a new product (e.g. for a startup) and for company-internal business cases, it is usually sufficient to stop at the operating profit.

In the company-internal case, that's roughly equivalent to the contribution margin that the product contributes to the larger organization.

In order to comply with company-internal standards for building a financial model, the software product manager needs to work closely with a representative of the company's finance function.

Focus on the Cost Structure

The business model canvas focuses on the ***cost structure***, asking: "What are the major cost drivers that make or break the business case?" Usually, for a software product, this is not the classic cost of revenue, such as labor or parts and materials, but fixed costs, such as R&D and sales and marketing, and in some cases, variable customer acquisition costs (cost of revenue).

In most traditional „brick and mortar" business models, for example a manufacturing business or a retail operation, the cost of revenue (labor, parts, etc.) eats up most of the revenue, often in the 70% range. Only the remaining 30% of gross profit are available to cover the fixed costs listed above. This typically leaves rather small margins for operating profit and net profit. Services businesses, for example support or professional services organizations of software vendors, have a similar cost structure, since their main cost driver is the cost of people delivering the services, and this is classified as "cost of revenue" as well.

However, for pure software products, the income statement looks quite different: small cost of revenue often leaves a gross profit of 70% or more – and this is needed to cover high fixed costs, in particular in R&D and sales & marketing. Usually, there are no expensive assets, so depreciation of assets is not a big concern either. Big cost drivers for a pure software product are (variable) customer acquisition costs, marketing expenses, and people cost in various categories (esp. R&D, marketing, perhaps customer support).

However, many software businesses rely on a revenue mix that combines "pure" software revenue, e.g. from license sales, with revenue from support and professional services. Depending on the revenue mix, the cost structure of these companies falls somewhere in the middle between the two extremes. There can be exceptions regarding low cost of revenue in the form of costs for embedded licenses or intellectual property, or profit sharing with partners.

Considerations for Software Product Managers

Low cost of revenue also means low marginal cost, i.e. once the software has been developed and is up and running, it doesn't cost much more to serve an additional customer.

This is a defining characteristic of digital business models and it gives software product managers a high degree of freedom in business model design and pricing, making it possible to stretch business models to an extent that is unthinkable in traditional industries (see EU2), such as a freemium business model.

This choice in available business models and pricing strategies and the complexity that arises from combining multiple revenue sources (see EU7: Revenue Streams) makes a financial analysis mandatory for evaluating a software product strategy.

Due to the low marginal cost in many software products, the financial analysis often focuses mostly on revenue: costs are treated as unchangeable and the only question is: Can we find a price structure that enables the necessary unit sales in order to recoup our costs and generate a profit?

However, in more traditional industries, this question is often approached the other way around: the target costing method starts with a desired competitive price point for a product and then sets a cost target that allows the company to manufacture and deliver the product and to generate a profit at the desired price. This approach may be useful for certain types of software products as well: for example if the potential market for a product is clearly limited, then development costs cannot be recouped through sales to more customers, and target costing can be helpful.

Building and Using the Financial Model

Especially for a new product or when evolving the business model for an existing product, the main purpose of the financial model is to answer the question: Under which conditions/assumptions can we achieve profitability?

To answer that question, it is recommended to build a financial model that enables product managers to ask "what-if" questions, using the following practices:

- build a multi-year model that reaches beyond the break-even point (3 to 5-year models are common)
- model multiple revenue streams separately
- on the cost side, work closely with liaisons from the various company functions, including R&D and marketing, to build or leverage suitable models that describe costs over time. For example, technical decisions on the R&D side may generate hidden costs or contingency costs later – and they can affect other areas (such as support).
- isolate underlying assumptions as parameters that can be changed easily, e.g. conversion rates from free to premium in a freemium business model
- build multiple scenarios

Once the model has been built, it can be used to study the profitability of the product under various scenarios that use different assumptions.

Literature: *Kittlaus, H.-B. (2022, p. 103 ff); Haines, S. (2014, chapter 6); Institute of Management Accountants (1998)*

EU9 Business Measures, KPIs, and Risk Management

Duration: 1:15 h

Educational Objectives:

EO9.1 Understand the importance of business performance management.
EO9.2 Know techniques for business performance management.
EO9.3 Understand the types of risks involved.
EO9.4 Know techniques to prioritize risks.
EO9.5 Understand how to define and select relevant business measures.
EO9.6 Be able to develop a set of business measures for a defined business model.
EO9.7 Be able to identify risks involved and develop corresponding mitigation strategies.

Business Measures and KPIs

Once the product strategy is defined business measures or key performance indicators (KPIs) are needed for continuous tracking and analysis of the business performance. Ideally, all elements of the product strategy (and business model canvas) are addressed. The measurement results help to learn and improve, and to track whether the product is following or drifting away from the business model and targets.

Fig. 7 shows examples for frequently used measures in four perspectives related to segments of the business model canvas. Often the measures are standardized on the corporate level. If, however, the software product manager has the freedom to define them a trade-off needs to be determined what ought to be measured and what can be measured in a relatively simple and cost-effective manner. An example for a comprehensive measurement approach is the Balanced Score Card (BSC).

Perspective	**Business canvas segment**	**Sample measures**
Financial	Revenue streams and cost structure	Revenue (monthly recurring revenue (MRR), average revenue per user (ARPU) and customer lifetime value (CLTV)), profit and loss (P&L) metrics such as gross margin and EBIT
Customer / Market	Value proposition, customer relationships, customer segments	Customer satisfaction score (CSAT), net promoter score (NPS), customer perceived value, customer retention rate and churn rate, number of registered users; market share, share of (market) growth, share of (customer) wallet, customer acquisition

		cost (CAC), share of market spend
Internal business / Organizational	Key activities, key partners, channels	Process quality, process cycle time; speed metrics such as time-to-market, time-to-revenue, customer time-to-value (TTV); cost metrics such as cost of quality (COQ) and customer onboarding cost
Innovation and learning	Key resources	Human capital measures

Fig. 7 Mapping of four perspectives to business canvas segments and sample measures

The choice of measures has to be a careful one since the chosen metrics should enable effective decision support. While so-called vanity metrics make a company feel good, they do not provide actionable insight. Thus, a product manager needs to select actionable metrics (that provide insight and support decision-making). For example, during the growth phase of a paid web service, it is recommended to continually increase the budget for customer acquisition initiatives to take advantage of the growth opportunity. However, in that situation, focusing solely on the number of registered users in most cases is a vanity metric – what really matters in that situation is the number of users relative to effort/acquisition costs. If the number of users is increasing by 50%, it might look good at first sight (vanity metric), but if the average acquisition cost per user shows a trend of increasing heavily over time, it is actually not a good development. There is a risk that the product will run into a situation where acquisition costs per user are getting higher than the average life time value of a new user. That is clearly not sustainable - every new customer will increase the company's losses. In this example, the standalone "number of users" metric is not really actionable for product managers, as it is not telling them whether their product is moving in the right direction or not. Some of the ways to obtain actionable metrics:

- Split tests
- Per-customer metrics
- Funnel metrics and cohort analysis
- Keyword metrics

With software, product usage can be measured in ways that no other product area allows. For a web-based environment, the Lean Startup movement provides a lot of information on how to track users and develop meaningful metrics. This data allows highly valuable insight into how individual product features are being used or not used.

Risk Management

The capability to mitigate risks effectively at all the different product management and development stages is critical for building a successful software product. Three types of risks can be identified, and corresponding mitigation strategies need to be devised:

- Product risks: relate to getting the product right. In the canvas it relates to the unique value proposition, key activities and key resources, partners, cost structure (resources) and revenue streams (pricing).
- Customer risks: relate to building a scalable path to the customers. In the canvas it relates to the customer segments and channels.
- Market risks: relate to building a viable business. In the canvas it relates to business measures and intersection of cost structure and revenue streams segments (determining the product's margins).

Depending on the product's life cycle stage, risks will differ. It is important to focus on addressing the right risks at the right time to minimize waste since incorrect prioritization of risks is one of the top reasons for waste.

Literature: *Croll, A., Yoskovitz, B. (2013); Neef, D., Siesfeld, G.A., Cefola, J. (1998); Kaplan, R.S., Norton, D.P. (1996a); Kaplan, R.S., Norton, D.P. (1996b); Pritchard, C.L. (2015); Kittlaus, H.-B. (2022, p. 129 ff)*

EU10 Legal Aspects

Duration: 1:15 h

Educational Objectives:

EO10.1 Understand the legal differences between a software license contract and a service contract.
EO10.2 Understand the importance of protecting intellectual property.
EO10.3 Know intellectual property protection mechanisms.
EO10.4 Understand the legal aspects of open source software.
EO10.5 Understand the legal aspects of data protection.

Software product managers need to consider several legal aspects related to software products. Details are typically handled by legal experts (e.g. counsels), but product managers need to have an overview of the legal risks. In addition to the four legal areas this chapter focuses on, product managers also need to consider supply-chain, antitrust restrictions, compliance requirements, product liability, export controls, e.g. blacklisting of countries for specific software components etc. which are increasingly relevant (see for instance the growing impact of export control laws). Companies usually intend to achieve legal compliance which means that all relevant legal requirements are addressed and fulfilled in all relevant areas of jurisdiction, i.e. wherever the company does business.

Contracts

The contract by which software is "acquired", may be negotiated individually or, particularly in the mass-market, based on so-called "standard terms and conditions," which describe the generally applicable legal terms, including:

- Scope of the license or service
- Warranty / SLA
- Transferability
- Type of charges
- Liability
- Maintenance provisions (a separate maintenance agreement may be concluded)
- Miscellaneous legal provisions (e.g. set-off, default, dispute resolution, governing law, severability clause)

A license describes to which extent and under which conditions the licensee can use software, which is subject to intellectual property rights, in particular copyrights, and possibly trademarks and patents. These rights of use are granted by the licensor who may be the owner of all rights in such item, i.e. the software company, or may be an entity authorized to grant the license, e.g. a reseller. The term "license contract" means an agreement between the licensor and the licensee about all terms in connection with a license. With software, the term "license" usually describes the scope of rights of use which the licensor grants to the licensee. In the case of SaaS or other software-intensive services, the service provider, i.e. the entity which makes the service available through the internet, needs to either own the IP rights in the relevant software itself or has to conclude license contracts with the owner of the

software which explicitly allow this kind of use, in other words a SaaS-license. Customers of such a service do not need a license, but only service contracts with the service provider, as the software code, e.g. object code, is not running on their system, but only on the service provider's infrastructure (whether owned or leased). Please note that this applies to browser-based SaaS-scenarios where a specific client application does not need to be installed on the customer's computer system. In IaaS-scenarios the license requirements may be different: the IaaS-provider or the customer may need a license for the software installed that runs on the infrastructure provided by the IaaS-provider.

Protection of Intellectual Property

Since the development of software requires significant investment, and software can be easily copied, it is of utmost importance to the investor that – from a legal perspective – the intellectual property resulting from the investment is protected, and – from a practical perspective – actual access to the software, especially the source code, is controlled. There are four fundamental legal constructs for the protection of intellectual property:

- Copyright: Protection against copying of software code (as specific expressions of an idea or way of doing something) and product material such as manuals, brochures and presentations. This is the main way software is protected. The algorithm or idea behind software is not protected under copyright law.
- Trademark: Protection for the names of brands, i.e. brands do not apply to the software itself but only to the brand under which it is marketed.
- Trade Secret: Protection of company-internal knowledge (primarily against employees). This protection is exercised by restricting knowledge and access to a very small number of people and by using non-disclosure agreements. In most jurisdictions, trade secrets are only protected under unfair competition laws.
- Patent: Protection of the specific technical concept or idea. In most jurisdictions, patent protection can only be obtained for software which is integrated into a technical solution to a problem.

Open Source

Open Source Software is frequently an (important) part of software development projects and may help to reduce development cost and time. However, as all software, open source software is subject to copyright protection. Thus, the developer of an open source software component is owner of the pertaining copyrights and disposes of them by offering the module free of charge under an open source license agreement. Consequently, a company which uses open source software for its own development processes has to comply with its license terms. An according license contract is concluded (implicitly) when downloading or installing the open source component. The variety of available open source licenses is manifold. From the perspective of the user of open source, the so-called free licenses are not causing significant problems. These licenses simply allow any kind of use of the open source software free of charge without stipulating any further restrictions, apart from uncomplicated formal requirements (e.g. Creative Commons' CC0, BSD License, MIT License). More difficulties can be caused by the so-called copyleft licenses which oftentimes govern the use of open source components (e.g. the GNU-Licenses (GPL, LGPL and GFDL)). The underlying idea of copyleft licenses is that the open source software is made available free of charge and thus any further

developments accomplished on the basis of such copyleft software may only be distributed on the basis of the copyleft license. In practice, this means that the source code of software which was developed by using open source software must be offered free of charge – at least to anyone who acquires a copy of the machine-readable code (object code), sometimes also to the general public. Dangerously, it does not matter how significant the open source code was in developing software. If only an open source code snippet is used, the respective copyleft license applicable to such snippet applies to the entire software (so-called viral effect). As long as such software is used for mere internal purposes of a corporation, the copyleft license effect does not apply. As soon as the object code is sold (stand-alone or as part of a hardware product), the obligations of the copyleft license apply, in particular the above mentioned duty to make the source code available, and formal requirements, such as the obligation to mention the applicable open source license in the source code. This may also apply if the object code is made available as part of a SaaS offering, or the code is made available within a group of companies. If open source code from various origins has been used, multiple open source licenses may apply in parallel – making it impossible to comply with all of them (in case of distribution). In case of an infringement, the owner of the rights (as a rule, the developer(s) of the respective open source code) has the right to request halting the use of the software and to claim damages. Damages may e.g. be calculated by the amount of profit made by distributing the software in violation of the open source license.

Data Protection

European Union (EU) data protection law, primarily the General Data Protection Regulation (GDPR), applies to all companies and branch offices within the European Economic Area (EEA) and also to other companies to the extent they collect, process or use personal data with regard to services offered to or monitoring of EEA data subjects.

The EU data protection law has served as a model for many other jurisdictions. Consequently, various other countries have enacted data protection laws which are meeting the EU data protection requirements: e.g. Switzerland, Canada, Israel, Argentina, Australia etc. However, in many other countries, e.g. in the USA, the approach to data protection is fundamentally different. Whereas in the EU any kind of personal data (i.e. any data which relates or can be related to an individual) is protected irrespective of its sensitivity, in the US numerous special data protection laws apply in certain areas, e.g. for health data or credit card data; outside those areas the general principle of privacy applies which only grants protection to sensitive data from private spheres. As a result, e.g. personal data of employees is subject to the data protection laws in the EU, but not, at least in principle, in the US. However, the California Consumer Privacy Act (CCPA) now provides for some data protection rules similar to the GDPR.

Under the GDPR the company running software to process personal data, e.g. ERP software, CRM software etc., is responsible for data protection compliance as the so-called controller. Hence, the software-developing company is not directly in charge of data protection compliance. However, customers are, and will become more and more, aware of data protection issues and will thus likely request software which takes into account data protection principles and requirements, such as data minimization including pseudonymization wherever possible, access by users on a need to know basis only (roles) and data security. Moreover, Art. 25 GDPR codifies the principle of privacy by design and

default. Such regulation requires controllers to purchase and develop software which is compliant with GDPR requirements.

The situation is fundamentally different if the software company runs the software and makes it available to its clients under a SaaS distribution model. In such case, the client's personal data are processed on the company's server infrastructure. Nevertheless, the customer remains the controller of its data who remains responsible to ensure data protection compliance. As a rule, the SaaS provider is considered to be a commissioned data processor under Art. 28 GDPR. In such case, the controller and the SaaS provider have to conclude a commissioned data processing agreement according to which the SaaS provider commits to comply with the directions of the controller, allows regular checks and controls and implements adequate technical and organizational measures for data security. Special restrictions may apply in certain areas, e.g. telecommunication, health data, insurance data, tax data etc.

To the extent that personal data is transferred from the EEA to recipients outside of the EEA, SaaS requires additional precautions if the recipient's country has not been approved by the EU commission as a safe country. The typical solution is that the controller and the SaaS provider agree on the so-called EU model clauses which oblige the data importer (i.e. the SaaS provider) to comply with the fundamental principles of EU data protection law. For recipients in the USA, the privacy shield rules used to help to comply with GDPR requirements but the European Court of Justice has held them to be unlawful. Based on this ECoJ decision (Schrems II), the EU Model Clauses alone are not deemed to be sufficient to justify a transfer of personal data to the US.

Literature: *Kittlaus, H.-B. (2022, p. 117 ff); Carey, P. (2009); Klemens, B. (2006); Vasudeva, V.N. (2014); Popp, K.M. (2015)*

Bibliography

This literature has been used by ISPMA as the scientific basis for this syllabus. It is not required reading for course participants.

Allen, P. (2006): Service Orientation – Winning Strategies and Best Practices, New York.

Bech, H.P.: Building Successful Partner Channels, TBK Publishing, 2015

Bekkers, W., van de Weerd, I., Spruit, M., Brinkkemper, S. (2010): A Framework for Process Improvement in Software Product Management, in: Communications in Computer and Information Science - Systems, Software and Services Process Improvement, 99, 2010, 1, pp. 1-12.

Borg, M., Chatzipetrou, P., Wnuk, K., Alégroth, E., Gorschek, T., Papatheocharous, E., Shah, S.M.A., Axelsson, J. (2019): Selecting component sourcing options: A survey of software engineering's broader make-or-buy decisions, Information and Software Technology 112

Burke, A., van Stel, A., Thurik, R. (2010): Blue Ocean vs. Five Forces, Harvard Business Review.

Buxmann, P., Hess, T., Diefenbach, H. (2012): The Software Industry. Springer.

Carey, P. (2009): Data Protection: A Practical Guide to UK and EU Law, Oxford University Press.

Chen, S., Cheng, A., Mehta, K. (2013): A Review of Telemedicine Business Models. Telemedicine and e-Health 19(4):287-297.

Christensen, C.A. (2013): The Innovator's Solution, Harvard Business Review Press.

Coulter, M. (2012): Strategic Management in Action (6th ed.). Upper Saddle River, NJ: Pearson Prentice Hall.

Croll, A., Yoskovitz, B. (2013): Lean Analytics: Use Data to Build a Better Startup Faster, Lean (O'Reilly).

Cusumano, M., Gawer, A., Yoffie, D.B.: The Business of Platforms, Harper Business, 2019

Daviesa, A., Bradyb, Z., Hobday, M. (2007): Organizing for solutions: Systems seller vs. systems integrator, in; Industrial Marketing Management, 36, 2007, 2, pp. 183–184.

Doshi, P. (2011): Seven software delivery models and why software as a service (SaaS) is the future! (http://insightextractor.com/2011/09/05/seven-software-delivery-models-and-why-software-as-a-service-saas-is-the-future/).

Dunford, A.: Obviously Awesome – How to Nail Product Positioning so Customers Get It, Buy It, Love It, Ambient Press, 2019

Fleisher, C., Bensoussan, B. (2015): Business and Competitive Analysis: Effective Application of New and Classic Methods, 2nd edition, Pearson Education.

Gartner (2013): IT-Glossary: Technology Defined (http://www.gartner.com/it-glossary).

Haines, S. (2014): The Product Manager's Desk Reference, 2. ed., McGraw Hill.

Herzwurm, G., Pietsch, W. (2008): Guidelines for the Analysis of IT Business Models and Strategic Positioning of IT-Products, 2nd International Workshop on Software Product Management (IWSPM'2008), Barcelona, pp. 1-8.

Iansiti, M., Levien, R. (2004): The Keystone Advantage – What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability. Harvard Business School Press.

Institute of Management Accountants (1998): Tools and Techniques for Implementing Target Costing, ISBN 0-86641-272-7; (http://www.imanet.org//-/media/745cee1e7db74276b6fef4d0c4072396.ashx?as=1&mh=200&mw=200).

Jansen, S., Cusumano, M., Brinkkemper, S. (2013): Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar Publishers

Kaplan, R.S., Norton, D.P. (1996a): Linking the balanced scorecard to strategy. California management review, 39, no. 1, pp. 53–79

Kaplan, R.S., Norton, D.P. (1996b): Using the balanced scorecard as a strategic management system, Harvard Business Review 74, no. 1, pp. 75–85 (Reprint #96107).

Kepes, B., Subramanian, K. (2010): Software Delivery Models - Comparing Approaches and Debunking the Myths (http://diversity.net.nz/wp-content/uploads/2010/07/Software-Delivery-Models.pdf)

Kim, W.C., Mauborgne, R. (2015): Blue Ocean Strategy: How to Create Uncontested Market Space and Make the Competition Irrelevant. Extended edition. *Boston: Harvard Business School Press*

Kittlaus, H.-B. (2022): Software Product Management – The ISPMA-Compliant Study Guide and Handbook, 2nd Edition, Springer.

Kittlaus, H.-B. (2020): Platform – What Do We Mean?, https://www.linkedin.com/pulse/platform-what-do-we-mean-hans-bernd-kittlaus/

Kittlaus, H.-B. (2019): Customer-Specific Tailorability of Your B2B Software Product – the Good, the Bad and the Ugly, https://www.linkedin.com/pulse/customer-specific-taylorability-your-b2b-software-product-kittlaus/

Kittlaus, H.-B.,Clough, P. (2009): Software Product Management and Pricing – Key Success Factors for Software Organizations. Springer.

Klemens, B. (2006): Math You Can't Use – Patents, Copyright, and Software, Brookings.

Kotler, P., Armstrong, G. (2015): Principles of Marketing – Global Edition, 16th ed., Pearson.

Kude, T. (2012): The Coordination of Inter-Organizational Networks in the Enterprise Software Industry: The Perspective of Complementors . Peter Lang.

Maurya, A. (2012): How to create your lean canvas (http://leanstack.com/LeanCanvas.pdf, pp. 17-19).

Messerschmitt, D.G., Szyperski, C. (2003): Software Ecosystem – Understanding an Indispensable Technology and Industry, MIT Press.

Meyer, R. (2008): Partnering with SAP. Books on Demand.

Moore, G.A. (2014): Crossing the Chasm, 3rd ed., Harper Business.

Moore, G.A. (2004): Inside the Tornado: Strategies for Developing, Leveraging, and Surviving Hypergrowth Markets. Harper Business.

Nagle, T.T., Hogan, J.E., Zale, J. (2014): The Strategy and Tactics of Pricing – A Guide to Growing More Profitably, 5th revised edition, Pearson Prentice Hall.

Nagle, T.T., Hogan, J.E. (2005): What Is Strategic Pricing?, in Strategic Pricing Group Insights (Monitor Group).

Neef, D., Siesfeld, G.A., Cefola, J. (eds., 1998): The Economic Impact of Knowledge, Elsevier.

Osterwalder, A., Pigneur, Y. (2014): Value Proposition Design. Wiley.

Osterwalder, A., Pigneur, Y. (2010): Business Model Generation. Wiley.

Parker, G.G., Van Alstyne, M.W., Choudary, S.P.: Platform Revolution, W.W. Norton, 2016

Peppard, J., Rylander, A. (2006): From Value Chain to Value Network: Insights for Mobile Operators, European Management Journal (DOI:10.1016/j.emj.2006.03.003)

Popp, K.M.: Best Practices for Commercial Use of Open Source Software, Books on Demand, 2015

Popp, K.M. (2012): Leveraging open source licenses and open source communities in hybrid commercial open source business models, Proceedings of the International Workshop on software ecosystems (IWSECO 2012), Cambridge (http://ceur-ws.org/Vol-879/).

Popp, K.M. (2011a): Software Industry Business Models, IEEE Software, vol.28, no.4, pp.26-30.

Popp, K.M. (2011b): Hybrid revenue models of software companies and their relationship to hybrid business models, Proceedings of the International Workshop on Software Ecosystems (IWSECO 2011), Brussels (http://ceur-ws.org/Vol-746).

Popp, K.M., Meyer, R. (2010): Profit from Software Ecosystems: Business Models, Ecosystems and Partnerships in the Software Industry, Books on Demand.

Popp, K.M. (2010): Goals of software vendors for partner ecosystems – a practitioner´s view, published paper. In: Proceedings of Software Business: First International Conference, ICSOB 2010, Jyväskylä, Finland, June 21-23, 2010. Springer.

Porter, M.E. (1986): Competitive Strategy. Harvard Business School Press.

Porter, M.E. (2008): The Five Competitive Forces That Shape Strategy, Harvard Business Review 86, no. 1, pp. 78–93.

Pritchard, C.L.: Risk Management: Concepts and Guidance, 5th ed., CRC Press, 2015

Pruitt, J., Adlin, T. (2006): The Persona Lifecycle: Keeping People in Mind Throughout Product Design, Elsevier.

Ries, E. (2011): The Lean Startup, Crown.

Ries, E. (2009): Minimum Viable Product: a guide, http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html

Schmidt, M. (2002): The Business Case Guide. Solution Matrix.

Schütz, S., Kude, T., Popp, K.M. (2013): The Impact of Software-as-a-Service on Partner Management. In: Herzwurm, Margaria (eds.): Proceedings of Software Business. From Physical Products to Software Services and Solutions: 4th International Conference, ICSOB 2013, Potsdam, Germany, June 11-14. Springer.

Schwind, M. (2007): Dynamic Pricing and Automated Resource Allocation for Complex Information Services, Springer.

Segetlija, Z., Mesarić, J., Dujak, D. (2011): Importance of Distribution Channels – Marketing Channels – for National Economy, in D. Križman Pavlović, D. Benazić (eds.): Proceedings of 22nd CROMAR Congress: Marketing challenges in new economy, Pula, 2011., ISBN: 978-953-7498-45-0, pp. 785 – 809.

Sheen, R., Gallo, A.: HBR Guide to Building Your Business Case, Harvard Business Review Press, Boston, 2015

Sodhi, M.N., Sodhi, N.N. (2007): Six Sigma Pricing: Improving Pricing Operations to Increase Profit, Pearson FT Press.

Tan, Y.S., Fraser, N.M. (1998): The modified star graph and the petal diagram: Two new visual aids for discrete alternative multicriteria decision making, Journal of Multi-Criteria Decision Analysis 7, no. 1, pp. 20–33.

van Angeren, J. Kabbedijk, J., Popp, K.M. (2013): Managing Software Ecosystems through Partnering. In: S. Jansen, M. Cusumano, S. Brinkkemper (2013).

Vasudeva, V.N. (2014): Open Source Software and Intellectual Property Rights, Klüwer Information Law Series

Waltl, J. (2013): Intellectual Property Modularity in Software Products and Software Platform Ecosystems, Books on Demand.

Weill, P., Malone, T.W., D'Urso, V.T., Herman, G., Woerner, S. (2005): Do Some Business Models Perform Better than Others? A Study of the 1000 Largest US Firms. Boston : MIT Center for Coordination Science, 2005. Working Paper No. 226.

Internet Links:

IL1: http://www.slideshare.net/minivation/business-model-canvas-12516531

IL2: http://www.startupaddict.com/startups/using-the-business-model-canvas/4710

IL3: http://canvanizer.com/how-to-use/business-model-canvas-tutorial

IL5: http://www.slideshare.net/conniemkwan/marketing-dealing-with-competition

IL6: http://steveblank.com/2013/11/08/a-new-way-to-look-at-competitors

IL7: http://www.wisegeek.org/what-are-indirect-competitors.htm

IL8: http://www.fourhourworkweek.com/blog/2009/05/19/vanity-metrics-vs-actionable-metrics/

IL9: http://ec.europa.eu/justice/data-protection/index_en.htm

IL10: http://en.wikipedia.org/wiki/Persona_(user_experience) (Wikipedia entry on "Persona (user experience)" in July 2014)