The Rules in the Software Industry 1

1.1 Software and Software Markets: Unique Characteristics of the Software Industry

The software industry is fundamentally different from other industries. This is partly due to the unique nature of software as a product, but also the structure of software markets.

A distinctive feature of *software products* is that they, like all other digital goods, can be reproduced cheaply. In other words, variable costs are close to zero. This cost structure has the result that the licensing side of software providers' business is generally more profitable than the service side—or at least it appears to be, something we will discuss in more detail later. Moreover, software can be copied any number of times without loss of quality. Once a copy is available on the Internet, intellectual property rights are practically unenforceable. This especially applies to easy-to-understand products on business-to-consumer markets. In addition, once a software product has been developed, it is relatively simple to create different versions or packages and sell these to separate groups of customers at different prices.

Software markets also have some unique characteristics. The software industry is more international in nature than practically any other sector. Software can be developed by distributed teams working almost anywhere in the world, and sold over the Internet in seconds, at negligible cost. This has fueled global competition between software providers. In comparison with other industries, providers in many segments enjoy little "home advantage" in their national markets. Moreover, the network effects associated with software often creates winner-takes-all markets. This accounts for the large number of mergers and acquisitions, for example.

These and other specific economic principles and game rules will be discussed later in more detail. After all, they are the backdrop against which software industry players formulate their strategies and business models.

But first, we wish to briefly outline the historical development of the industry.

1.2 The Beginnings of the Software Industry

The software industry is relatively young. Its origins date back to the early 1950s, when it was still common practice to sell software and hardware as a single package. At that time, software was an integral part of hardware, and was still referred to as program code—the term software was first used in 1959 (Campbell-Kelly 1995, p. 80). In the USA, this period saw the establishment of the first small companies to develop software within the scope of specific projects (Hoch et al. 2000, p. 27 ff).

The profile of software rose in 1969, when the US Department of Justice demanded that IBM itemize hardware and software separately on its invoices. During the 1960s, a whole series of companies that focused exclusively on software development sprang up. Of these, Microsoft is most worthy of mention: founded jointly by Bill Gates and Paul Allen, the company began developing programming languages—first BASIC, and later others such as FORTRAN and COBOL—for various processors and computers. Later, in collaboration with IBM, Microsoft developed MS-DOS, which would become the standard operating system, significantly contributing to the spread of the personal computer (Ichbiah 1993, pp. 91–116). Later, the company would offer applications as well, entering into competition with Lotus and the like. As early as 1983, Bill Gates told in Business Week that it was Microsoft's goal to become a one-stop provider of all types of PC software (Ichbiah 1993, p. 141).

At the same time as Microsoft embarked on its long rise, another success story was unfolding in the small town of Walldorf, Germany. Dietmar Hopp, Hans-Werner Hector, Hasso Plattner, Klaus Tschira, and Claus Wellenreuther established a company specializing in the development of software for business applications and processes: SAP AG was born. At first, it developed software for mainframes, later offering applications for client–server environments. Today, SAP is the largest European software company and the world leader in enterprise resource planning (ERP) systems.

These first two examples already reveal a key characteristic of the software industry: often, just one technology or vendor will come to dominate the market. For example, MS-DOS squeezed out the rival CPM operating system, while the Excel spreadsheet program trounced competing products such as Lotus 1–2–3. Microsoft was to become the world's leading provider of office applications, browsers, and operating systems. The market for business software is currently undergoing consolidation (see Sect. 3.1.2). We will return to the special features of these markets—but will now turn our focus to the key players in the software industry.

1.3 Types of Software Provider and Users' Selection Criteria

1.3.1 Software Providers in the Wider and Narrower Sense

In the following section, we will introduce the various types of software providers. A distinction should be made between software providers in the broader and narrower sense.

The role of a *software provider in the narrower sense* is to develop software—and this applies to all types of software. Software itself can be categorized in various ways. A commonly used criterion is how directly it interacts with the hardware (i.e., at what level of the system stack does it operate. According to this method, software comprises system software (e.g., operating systems), middleware at the next level up, and application software (e.g., for word processing or accounting) above that. Software can also be divided into products for commercial and for private users. A third criterion for classification of software, and one that is more pertinent to our discussion, is the degree of standardization. On this scale, custom software and standard software are at the end points.

Custom software is developed, in line with specific customer requirements. It is developed either in-house—generally in the IT department, but in some cases, in relevant user departments—or by an external software vendor. In India particularly, the custom software development industry is booming. A number of successful players with enormous growth rates have emerged, such as Tata Consultancy Services, Wipro Technologies, Infosys Technologies, and Cognizant Technology Solutions. In addition, there are multitudes of smaller or mid-size companies that are still operating nationally or regionally in high-wage countries. By contrast, custom software development companies based in low-wage countries are more likely to operate globally, and so compete against one another. We will examine these nearshore and farshore providers and their impact on the software industry more closely in Sect. 4.6.

Standard software is generally developed for the mass market. As a result, providers address the lowest common denominator in terms of users' needs. In the following pages, we will analyze the rise of, and success factors behind, standard software, using SAP as a case study.

SAP and its R system

SAP is the leading provider of business software in Europe and the fourth largest independent software vendor in the world. The company employs 55,000 people at 75 subsidiaries worldwide. Overall 183,000 customers reap the benefits of SAP's solutions.

Key factors

SAP's success is attributed to three key factors; namely, the idea of standard software, offering integrated solutions, and real-time processing. In 1981, the company put these ideas into practice for the first time with SAP R/2, thereby opting to focus on the development of business software. SAP launched the R/3 system a decade later. Originally, the new architecture was not meant to replace R/2 but complement it by providing a solution for SMEs. However, the general changeover from mainframe solutions to the client–server model, which was underway at that time, made R/3 just as appealing to larger companies and led to its huge popularity. Since then, SAP has extended its product portfolio. We will return to this topic later.

Standard software

In the early days, as mentioned above, software was generally developed within customer-specific projects. This meant that the software and, if necessary, the hardware was tailored exactly to the customer's requirements. By contrast, SAP planned from the outset to develop a standardized system that could be used by multiple customers. SAP was one of the first companies to systematically pursue this approach.

An integrated solution

At the heart of the SAP system is an integrated database that all applications can access. Using an integrated data repository was a completely new approach in the 1970s. At the time, redundant and inconsistent data generated enormous costs. Building on this foundation, SAP gradually developed modules for various corporate functions, such as financial and management accounting, logistics, and human resource management. Initially, these packages were primarily designed for industrial companies and could be purchased individually or as a package which included a database.

In contrast to office applications such as those offered by Microsoft, it became clear that many areas of business had to address industry-specific requirements. To meet this need, SAP now offers a wide variety of industry-specific solutions. This included the service sector, which had received little attention up to that point.

Real-time data processing

By the end of the 1970s, companies generally used punch cards, i.e., data were entered into the computer via punch cards and not processed until later. From the outset, SAP's founders saw real-time processing as a key feature, which was implemented in all systems. This also explains the "R" in the product name, which stands for real time.

Sources Leimbach (2007) Vom Programmierbüro zum globalen Softwareproduzenten. Die Erfolgsfaktoren der SAP von der Gründung bis zum R/3-Boom. In: Zeitschrift für Unternehmensgeschichte 52: 5–34; www.sap.com.

There is no clear distinction between custom and standard software's. Even standard business software can be customized to users' needs to a certain extent. Nonetheless, implementation projects involving software parameterization and/or customization, and, where required, function extensions, frequently cost into the millions. So, it generally makes sense for users to opt for smaller adjustments only: complete modification of standard software, until it is 100 % geared to users' needs, is extremely expensive; it can also create problems when upgrading to new versions. In addition, new approaches, such as service-oriented architectures, offer the possibility (at least theoretically) of selecting software that best matches their

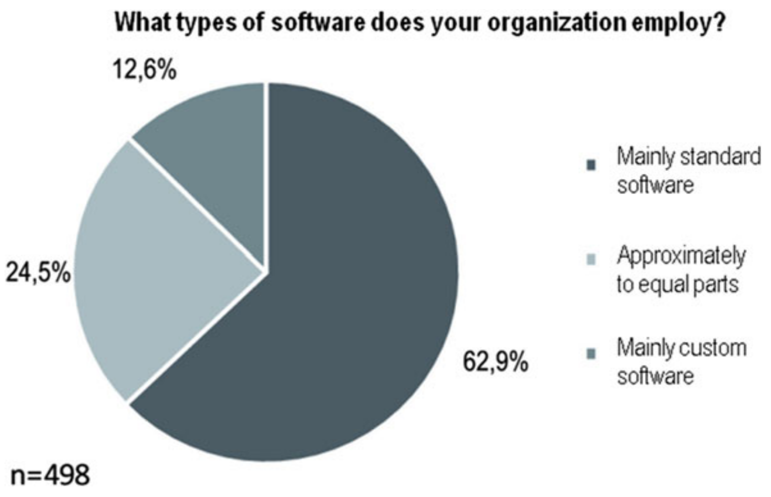

Fig. 1.1 Share of software types

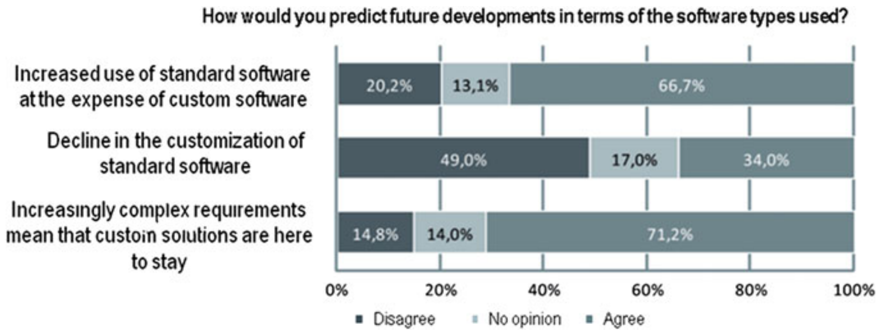

Fig. 1.2 Predicted developments in the share of software types used

requirements in specific areas, and then melding them into a single customized application solution by means of integration software (see Sect. 4.7.2).

The proportion of standardized software solutions in enterprises' portfolios is set to increase, as our survey of 498 German CIOs shows (Buxmann et al. 2010). Total 62.9 % of respondents' companies mostly use standard software solutions, while 24.5 % employ standard and custom softwares more or less equally. Custom software dominates at 12.6 % of enterprises (Fig. 1.1).

In addition to asking about their companies' current use of standard and custom softwares, we also canvassed respondents' opinions as to the future development of these types of software. Almost 66.7 % of the CIOs surveyed agreed with the statement that companies are increasingly employing standard software at the expense of custom software. Out of this 20.2 of companies disagreed, while 13.1 % expressed no opinion (see Fig. 1.2). Total 34 % of the participants

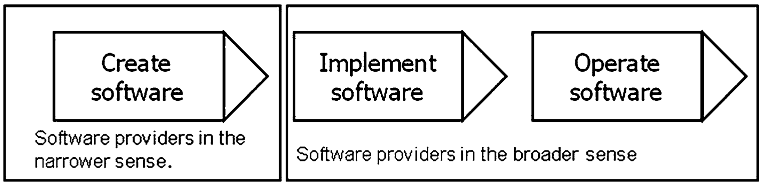
Create software
Implement software
Operate software
Software providers in the narrower sense.
Software providers in the broader sense

Fig. 1.3 Classification of software providers

concurred with the assertion that customizations of standard software would decrease, while almost half (49 %) disputed this assumption. The overwhelming majority of respondents (71.2 %) believed that, due to growing user requirements, custom solutions would continue to be needed sometimes to address specific problems, and would have to be integrated into the overall IT landscape.

In future, therefore, companies will employ more standard software solutions; however, customizations will still be necessary. For specific problems, for which no standard application software is available (as yet), companies will continue to develop individual solutions (either in-house or via outsourcing), and then have to integrate them into their IT environments.

In addition to software providers in the narrower sense—in other words, companies that develop standard or custom software—there are also those that offer services for the later phases in the lifecycle of software solutions. In the following, we have described these organizations as software providers in the broader sense. Their services include both implementation support and ongoing operation. Figure 1.3 depicts our classification of software providers.

Especially, in the case of complex software solutions that are not self-explanatory—in other words, those that are not easy to implement and integrate into an application environment—there is currently great demand for services. As a result, a large number of providers are active in this space. These can be broken down as follows (Lünendonk 2009):

- IT consultancies and system integrators (which offer IT consulting, development of custom software, etc.),
- IT service providers (outsourcing, ASP, user training etc.) and
- Business innovation/transformation partners (management and IT consulting, system implementation).

The provision of user support for SAP implementation projects has traditionally been a significant market. This is because—as discussed above—these projects involve customizing the software to the specific needs of the user organization. In many cases, companies using SAP software do not have the necessary knowledge and skills, or have insufficient employees with this expertise. Such implementation projects are crucial for the customer to gain the full benefits from the software, as it directly affects internal and inter-company processes. In addition, these projects frequently entail minor programming tasks, such as the development of interfaces between heterogeneous systems.

This business relies like almost no other on a relationship of trust between customer and service provider. Detlev Hoch et al. even describe this relationship in terms of "faith". In their view, the customer must have faith in the outstanding skills of the service provider or the employees assigned to the project, and in their ability to solve the customer's problems as promised (Hoch et al. 2000, p. 160). While users can test-drive products from a standard software vendor, this is simply not possible in the case of bespoke solutions. Often, the customers have no alternative, but to base their decision on the references furnished by the consultants and system integrators. Against this background, these service providers' marketing activities have the primary goal of building trust in their skills and ability to deliver. To this end, the following activities are essential: (Hoch et al. 2000, pp. 162–178):

- Sponsoring IT conferences,
- Discussion groups and forums with leading lights from the IT and software industries,
- Publications in industry media and scientific journals and
- Advertisements and TV commercials.

For customers, the complexity of implementation and integration projects means that selecting the right service provider is crucial. Especially, considering that most projects go over budget—and in many cases, fail altogether. In a study by Standish Group International Inc, just 32 % of the IT projects in their sample were successfully completed within the allotted time and budget (The Standish Group International 2009, p. 2). Overall 44 % far exceeded these targets, while 24 % were never completed at all. Projects assessed as being complex exceeded their budget and planned duration by more than 100 %.

Operating IT solutions has been a key part of the outsourcing business for many years (also see Chap. 4). There are a large number of players on this international and highly competitive market. We include them among our software providers in the broader sense.

In recent years, a new form of IT delivery has emerged: cloud computing, which will be examined in Chap. 5.

1.3.2 The Selection of Software

The business models of software providers (both in the narrower and broader sense) reflect the selection mechanisms and preferences of their customers. The following sections will outline the processes and criteria used for evaluating and selecting software solutions by businesses (Sect. 1.3.2.1) and consumers (Sect. 1.3.2.2).

1.3.2.1 Software Selection by Businesses

Companies generally go through several sequential steps to reach their final choice (see Fig. 1.4). This involves gradually reducing the choice of possible alternatives.

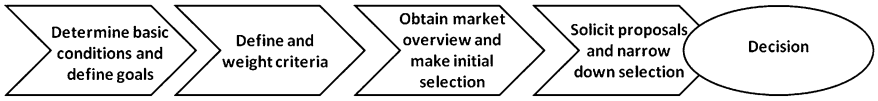
Determine basic conditions and define goals
Define and weight criteria
Obtain market overview and make initial selection
Solicit proposals and narrow down selection
Decision

Fig. 1.4 Software selection by companies

Typical context variables are company-related factors, such as industry, size or business functions to be supported, or environmental variables, such as technological standards. These basic conditions are used to define the goals which purchasing the software must achieve. The next step is usually to define criteria. These are characteristics that describe a software system and that form the basis for evaluation.

Market overviews are used to identify potential options. Typically, this is followed by a two-stage evaluation and selection process. In the first round, the options are assessed according to elimination criteria. Only solutions that meet these criteria are examined in more detail. With the help of a detailed definition of requirements (functional specifications), which is derived from the criteria, the company will then request quotations or additional information on the remaining options. Based on this information, a further round of selection follows, culminating in a selection recommendation. Some approaches also include post-selection steps, such as contract negotiations and functional testing to estimate the customization effort required.

The following section explores the development of a system of goals, the selection and weighting of criteria, and the activities involved in the two selection rounds in more detail.

The formulation of goals that are to be met by introducing or deploying software in a particular area is generally the starting point for analysis, and provides a yardstick for assessing alternatives. A variety of methods can be used; for example, goals can be derived from the top down, i.e., from corporate goals or the IT strategy. For example, high-level corporate goals, such as increasing profits, can be applied to the concrete problem of selecting standard software for a specific purpose. One way of doing this is to breakdown the high-level goals into concrete subordinate goals in terms of an end-means relationship with respect to the selection problem. Examples include reducing throughout, cutting inventory levels, or improving response times.

A bottom-up approach is also possible. The goals derived in this way have a very functional focus with respect to the specific selection problem and/or to existing structures (such as the IT architecture). Examples include goals in the form of general characteristics of software, such as independence from database or hardware vendors, compatibility with existing versions, or specific application-related functionality, such as automatic allocation of part numbers with checksum or broadening the information base. These characteristics often correspond to concrete requirements rather than being goals in the narrow sense, which are pursued for their own sake. In practice, to determine the goals needed as a basis for the criteria, most companies employ a combination of top-down and bottom-up methods.

Table 1.1 Typical selection criteria in the selection of ERP systems

Selection criteria (software)	Selection criteria (implementation/provider)
Functionality	Ability of software to be integrated into existing IT architecture
Cost	Implementation time/Customization costs
User friendliness	Support from provider
Reliability	Reputation of provider

If criteria are derived from the goal system, they generally represent the lowest level of this system. As a result, the effect of a given system property on the desired goals is clearly visible. Selection criteria are often subdivided into software, implementation, and provider-related properties to reflect the goal hierarchy. Besides the characteristics of the software itself, implementation criteria also play a key role, especially in complex systems. This is because over and above the procurement costs, implementation work can significantly impact the software's total cost. The implementation of complex software systems is often carried out with the support of software providers or implementation partners, and generally speaking, their expertise and professionalism should also be evaluated before selecting software. In addition to purely rational criteria, the role of political factors (e g., informal agreements, nepotism, and internal power struggles) should not be underestimated. These can have a significant influence on the selection process (Howcroft and Light 2006).

Table 1.1 shows which selection criteria for software, implementation, and providers are actually used in practice in the selection of ERP systems (Keil and Tiwana 2006; Jadhav and Sonar 2009).

In terms of weighting criteria toward the overall decision, it is generally assumed that these criteria are already, or will be fulfilled. However, it is also possible for a criterion to be partially met. In this case, it is necessary to specify a desired goal threshold or evaluate various thresholds. For example, a criterion dubbed "database independence" does not make it clear whether compatibility with two, three, or ten database management systems is required. This can be important to ensure solutions are not overrated, for example, where the threshold is set relatively low. With some goals, on the other hand, a certain minimum level needs to be met before an option will even be considered or investigated further. Such knockout (or killer) criteria make the selection process more efficient. However, they also pose the risk of ruling out options that only narrowly miss the minimum requirement.

A holistic approach is usually employed to weight criteria, for example, the kind familiar from cost-benefit analyses. Weightings are generally allocated by a direct comparison of criteria. For example, empirical studies on the selection of ERP systems have shown that functionality, reliability, and cost are weighted more heavily, while user friendliness and implementation effort are regarded as less important (Keil and Tiwana 2006). However, problems arise with this approach when a highly differentiated system of weightings is used. Especially with complex selection problems, which owing to the large number of evaluation criteria, include the selection of software, it is only possible to rank options according to an ordinal scale. More precise scores imply (pseudo-) accuracies that seldom reflect with the decision maker's actual preferences.

In practice, drawing up market overviews, shortlisting and evaluating alternatives are often interdependent tasks that are carried out at the same time. Whether or not a system is included in a market overview is a form of selection in itself. Due to the wide array of offerings available, this first selection step is commonly shaped by heuristic approaches such as:

- Inclusion of systems employed by competitors,
- Inclusion of systems currently receiving widespread coverage in industry media (e.g., Google Apps Premier, SAP ERP),
- Inclusion of systems that employees already have experience with,
- Inclusion of systems that, if selected, the consultant hopes will lead to a follow-up contract at the implementation stage and
- Inclusion of systems based on a random selection, e.g., through a visit to a trade show.

As explained above, evaluation and selection are an interconnected process, in which knockout criteria can reduce the number of alternatives. The following methods are typically employed:

- Evaluation of system descriptions,
- Evaluation of potential providers' bids in response to tender documents or requirements catalogs and
- Presentation of the system and/or concrete functionality based on defined application scenarios.

To evaluate certain criteria, a three-point scale comprising the values "fulfilled", "partially fulfilled", and "not fulfilled" is frequently used. Finally, individual evaluations need to be aggregated to form an overall assessment.

A system's suitability does not depend on how closely it meets functional requirements, but also on its costs. Accordingly, cost factors are often incorporated into the evaluation process by way of a cost-benefit coefficient. Ultimately, the software selection decision is made by comparing relevant quantitative and qualitative factors.

1.3.2.2 Software Selection by Consumers

While selection in companies is usually a group decision, consumers mostly make a choice alone. Much the same as in a business context, consumers typically go through four phases when selecting any product, including software (Blackwell et al. 2003). These are shown in Fig. 1.5.

Once a consumer has become aware of the need to buy new software (Phase 1: Determination of need), he or she will look for suitable information on available products that could satisfy the need. In doing so, the consumer, more or less

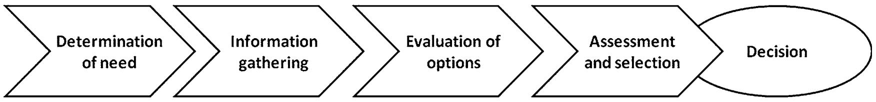

Fig. 1.5 Software selection process by consumers

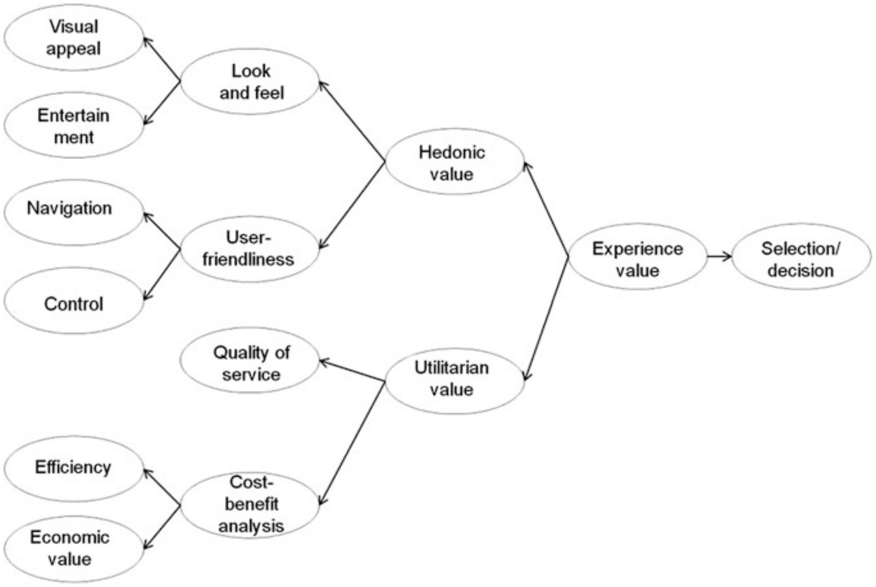

Fig. 1.6 Factors influencing the experience values of digital goods (following Mathwick et al. 2001)

consciously, identifies key subjective and objective criteria for decision making (Phase 2: Information gathering). After information gathering, the consumer processes the available data and evaluates the options in light of his or her personal selection criteria (Phase 3: Evaluation of options). Finally, the consumer compares and evaluates the available options and decides which product to purchase (Phase 4: Assessment and selection).

In contrast to selection decisions made by enterprises, consumers' choices are influenced not only by purely objective criteria (such as cost or functionality), but also individual or internal, i.e., personal (e.g., their personality, emotions, impressions, or lifestyle) and situational or external factors (e.g., culture or mouth-to-mouth propaganda). To help providers better understand which key criteria consumers use to make their selection, these factors will be examined in more detail below.

As mentioned above, in addition to purely rational selection criteria, soft factors play an important role in consumer decision making. A comprehensive model, which aptly reflects the special characteristics of digital products, is that of the experience value of digital goods as described by Mathwick et al. 2001 (see Fig. 1.6). It reveals that digital products (e.g. computer games) have an experience value that is (consciously or unconsciously) evaluated before purchase. The experiential value is a combination of utilitarian and hedonic factors.

Utilitarian factors are overwhelmingly objective in nature and typically reflect cost-benefit calculations. This cost-benefit mindset, in turn, is fed by mainly rational evaluation criteria, such as functionality, compatibility, or the software's procurement costs. In addition, there is the service provided with the software, including the support, and guarantees (e. g., a hotline or warranty) the user will obtain on purchasing the software.

By contrast, *hedonic factors* appeal to the personal and situational needs of customers. Consumers' esthetic preferences play a key role, as does user friendliness. Both factors include more concrete subcriteria, e. g., visual attractiveness (such as colors and 3-D effects) or the way the software is navigated (e. g., mouse, keyboard, and joystick), which address consumers' specific personal purchasing motives.

Put together, these subcriteria influence the product's hedonic and utilitarian factors and so the entire experiential value of the software, which in turn impacts the consumer's software selection and ultimate purchase. These subcriteria can be influenced to a lesser or greater extent by software providers. The design of business models for software vendors is the subject of the following section.

1.4 Business Models for Software Companies

A business model describes the essential decisions that influence the economic performance of a company (cf. Osterwalder 2004). It outlines the products or services offered by the company as well as its revenue and distribution model. Furthermore, it addresses the embedding of a company in the value chain.

In the software industry, it all begun with companies that develop software by direct order of their customer. Such a software company gets in exchange directly paid by the ordering customer, either a fixed sum for the product or accordingly for the employed resources. The business model of the individual software manufacturer was born.

Relatively fast it became clear, that some software could be used in a variety of companies. The idea of standard software was formed, first for mainly system software but soon also for application software. It led to the business model of the standard software manufacturer. In this model, software is no longer developed for a specific customer—but the software now is developed for an anonymous market. However, this does not exclude pilot projects in individual companies. Well-known suppliers in this segment are Oracle (for system software) and SAP (for application software). In this model, a user company typically buys a license to use the software. The price depends on the numbers of installations or other indicators for usage of the software. Often the license fee also includes costs for maintenance and incremental upgrades of the software. Up to now, the user companies use software mainly to support their processes. Increasingly, however,

Table 1.2 Three generic business models of software companies

	Supplier of Individual Software	Supplier of Standard Software B2B	Supplier of Standard Software B2C
Offering	Software, developed for the specific needs of a company	Software, developed for a mass market on the business side	Software, developed for a mass market on the consumer side
Revenue Model	Payment for completed solution or used resources	One-time license fee plus maintenance fee	One-time license fee
Distribution Model	Direct contact with customers	Direct contact with customers	Digital or physical retail or in exceptions preinstallation on computer hardware

the software becomes part of the company's product offering (e.g., in cyber-physical systems of cars).

In the 1990s, the rising demand of private customers (B2C) enlarged the existing customer base (B2B) to a sizeable extend. Suppliers of standard software begun to serve this market also, but employed a modified business model than in B2B. Typically in this model, the private customer pays an up-front fee for an unlimited usage license of the software and receives incrementally upgraded versions as well. Distribution takes place directly through the Internet or indirectly through computer hardware retail shops, in exceptions also through the suppliers own brick and mortar shop.

Table 1.2 shows the three traditional business models (see also Valtakosi/Rönkkö 2009) once again in an overview.

Two specific models have to be taken in account. First, an attractive model for software companies is the preinstallation of software on end-user devices. This enables the supplier to gain a revenue share from the sold bundle of hardware and software. Bundles like this are offered to private customers. A second special case occurs when the software can be purchased over a platform which is located between the software company and the user. This setting becomes problematic for a software company if the platform is the only possible way to install the software on the end-user device and the company is not the platform provider. Such a position was established by Apple for their temporarily market dominating Smartphone, the iPhone.

Suppliers of software were often keen to extend their share of the value chain by offering complementary services to their products. In the case of suppliers of individual software solutions, this is often directly part of the customer's order. Suppliers of complex standard software in the B2B market offer sometimes operational or technical services that support the implementation of the software in the customer's organization. Therefore, they stand in partial competition with specialized service providers.

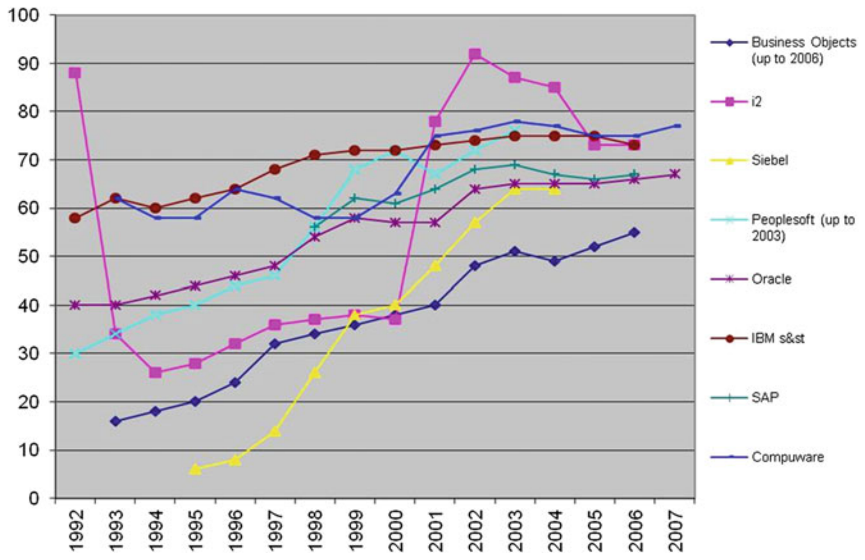

Fig. 1.7 Share of total sales generated by services, large standard software vendors (own calculations, building on Cusumano 2004, p. 37)

Equally, strong growth can be observed in the business customers' demand for the provisioning of services for data processing center. Specialized service providers in this segment are, e.g., T-Systems or IBM, but suppliers of software also try to enter in this business model. Recently, they started to offer Software-as-a-Service over the Internet and simplify the company-specific adjustments of software through, e.g., easy to use toolkits. SAP is currently developing their offers into this direction. Suppliers of that kind reach their turnover usually through charging monthly usage fees.

1.5 Revenue from Services in the Software Industry

In addition to incomes from licenses, software providers increasingly earn revenues from services. This is a source of revenue not just for custom software developers and software providers in the broader sense. It also offers standard software providers a wealth of opportunities to generate sales through services such as consulting and maintenance.

While consulting fees are commonly based on working days (and occasionally, on results), maintenance services generally involve the user paying the provider a percentage of the licensing fees annually. This percentage varies from provider-to-provider—a typical amount would be around 20 %/year. That makes it clear just how important this source of revenue is for software providers: If we assume that a software solution is used for an average 7–10 years before, it is replaced by a new version—or, less commonly, a different product—we can see that the revenues from maintenance generally exceed those from licensing; this applies even after discounting revenues. Furthermore, this business model also ensures a regular income from maintenance fees, in contrast to licensing fees, providing a relatively steady source of revenue that software providers can factor into their planning. Figure 1.7 shows that, over the last few years, most major standard software vendors were able to significantly increase the share of their total sales generated by services.

But although revenue from services is significantly higher than income from licensing, this does not mean that providing services is necessarily a more attractive business than selling licenses. This is shown by comparing the profitability of the various sources of revenue. Licensing generally makes a 90–100 % profit (Cusumano 2004, p. 43 ff.). Variable costs, such as for data media, manuals, services such as logistics service providers or packaging, are mostly low—and in the case of sales over the Internet, they are negligible. In contrast, the profit margins for consulting and maintenance services are considerably lower.

The profitability of these business models is pivotal for investors, who are often more interested in rates of return than in absolute monetary values. This often makes software providers who primarily focus on selling products and licenses more attractive for investors than software providers in the broader sense. The main reason for this is that providers of products can grow faster than companies whose sales largely consist of services. As a result, many software companies lack a consulting arm, e.g., Oracle. On the other hand, software providers who generate a large share of their sales through services have the advantage that, as noted above, they gain relatively constant revenues, which is especially significant during economic downturns, when there is little new business to be won.

It should also be noted that, the revenues from licensing and services discussed in this section are not independent of each other. Rather, the amount of revenue from licensing and providers' pricing strategies significantly influence income from services. In fact, a business model in the software industry can entail selling software at low prices—or even giving it away. This is often a rewarding and even necessary strategy, to gain an early foothold in the market or take market share from an established competitor. And in any case, it can be worthwhile to offer services in conjunction with low cost or free software, and generate profits entirely or largely from the sale of these services.

This kind of business model is especially suitable for companies that offer digital goods, such as software or music. In addition to building up revenue, there are a few other defining characteristics that differentiate software from other goods. In the second chapter, we will explore the economic background and the consequences for software markets.