

# PRICING PATTERNS FOR BUSINESS SAAS APPLICATIONS

---

Master thesis

Dennis Adriaansen

Utrecht University

Faculty of Science

Department of Information and Computing Sciences

*8 August 2012*

**Supervisors**

A. Mars, CFO, AFAS Software BV

Dr. S. Jansen, Utrecht University

J. Kabbedijk, Msc, Utrecht University

## **ABSTRACT**

Software vendors face problems with pricing their business SaaS applications. A typical characteristic of a business SaaS application is that it should be available to different kinds of customers that have their specific wishes. Therefore, business SaaS applications should support configurability and therefore a multi-tenancy architecture is the best solution. Other consequences of provisioning business SaaS applications are that the cost structure depends on the chosen deployment method and that metrics are needed for measuring usage. Pricing problems are identified at the case study company (a large business software vendor in the Netherlands), solutions are developed and experts evaluated the solutions. This resulted in an overview of appropriate pricing patterns for business SaaS applications, describing the problems, solutions, examples and consequences found in this study.

## CONTENTS

Abstract .....	2
Contents.....	3
1 Introduction .....	5
1.1 Problem statement .....	5
1.2 Research questions .....	6
1.3 Research relevance .....	7
1.4 Outline of the research .....	8
2 Research approach.....	9
2.1 Research methods.....	9
2.2 Validity.....	13
2.3 Activities .....	14
3 Business SaaS applications .....	17
3.1 The need for tailoring business software.....	17
3.2 Software characteristics.....	18
3.3 Software as a Service .....	19
3.4 Realizing configurability.....	20
3.5 Chapter overview .....	21
4 Implications of provisioning a business SaaS application .....	22
4.1 Architecture.....	22
4.2 Costs.....	25
4.3 Measuring usage.....	28
4.4 Chapter overview .....	28
5 Pricing patterns.....	29
5.1 Patterns .....	30
5.2 Usage-dependent pricing.....	32
5.3 Situation-based pricing.....	34
5.4 Cost-based pricing.....	35
5.5 Flat fee pricing .....	37
5.6 Two-part tariff pricing.....	38
5.7 Peak load pricing.....	39
5.8 Bundle pricing .....	41
5.9 Dual pricing.....	42
5.10 Chapter overview .....	44
6 Evaluation .....	45

6.1	Usage of the pricing patterns.....	45
6.2	Pattern evaluation.....	48
6.3	Consequences.....	54
6.4	Appropriate pricing patterns.....	57
6.5	Chapter overview.....	59
7	Discussion.....	60
8	Conclusion.....	62
10	References.....	64
11	Appendix.....	68
11.1	Metrics.....	68
11.2	Mail for criteria request.....	71
11.3	Answers from criteria request.....	72
11.4	Screenshots evaluation tool: characteristics.....	73
11.5	E-mail template for first evaluation.....	74
11.6	E-mail template for the second evaluation.....	74
11.7	Results pair wise comparison characteristics.....	75
11.8	Results assessment.....	76
11.9	Points awarded to pricing patterns.....	78

# **1 INTRODUCTION**

An important change to the software industry in the recent years is the emergence of software delivered as a service, this is named Software as a Service (SaaS). SaaS is a deployment method where the software system and the users' data are stored off-site in a central location run by the software vendor. The vendor delivers the bundle of IT infrastructure, software applications and services to users through a network (Ma & Seidmann, 2008).

Many software vendors develop software to support organizations by executing their business processes; this type of software is known as business software. Business software, such as Enterprise Resource Planning (ERP) software, is a configurable information system that integrates information and information-based processes within and across functional areas in an organization (Ward & Peppard, 2002). Accompanied by the SaaS movement, many business software vendors currently offer their software as a SaaS application or they are planning to launch such in the near future.

New possibilities of pricing the software arise when an application is offered as a SaaS application. Since the software runs on servers managed by the vendor, the vendor can precisely track the user's actions and track the usage of the software.

Nagle & Hogan, known for their work on pricing strategies and The Strategic Pricing Pyramid, address the importance of pricing by stating that: "companies operating with a narrow view of what constitutes a pricing strategy miss the crucial point leading to incomplete solutions and lower profits" (Hogan, 2005).

Existing work on pricing software addresses different strategies (Lehmann & Buxmann, 2009), but currently it is unknown what problems business software vendors face and how pricing patterns can solve those problems. The goal of this research is to identify pricing patterns for business SaaS applications.

## **1.1 PROBLEM STATEMENT**

Because of the differences with on-premises installed software, software vendors face new problems because their application is provisioned as SaaS. Established ways of pricing software might not be suitable for business SaaS applications. In addition, differences with consumer SaaS exist because business SaaS applications require configurability to serve different kinds of organizations.

Software vendors face problems with pricing their business SaaS applications. The pricing problems might arise because of the configurability of the software, the need for serving many different customers, difference in usage or another problem caused by provisioning a business SaaS application.

Therefore, software vendors of business SaaS applications should be provided with solutions for pricing their application. Based on these problems, the problem statement is formulated as follows:

*Software vendors face problems with pricing their business SaaS applications,  
and do not know how to solve these pricing problems.*

The purpose of this study is to:

1. Identify pricing problems at a software vendor.
2. Develop solutions for these problems.
3. Evaluate the developed solutions.

Solutions for pricing business SaaS applications are developed, because it is unknown how the pricing problems they face can be solved, so those vendors can be advised which price pattern they should use for their business SaaS application to maximize profit.

## **1.2 RESEARCH QUESTIONS**

The objective of this research is to identify and develop pricing patterns for business software vendors that offer their software as a SaaS application. Therefore, the main research question is formulated as follows:

***What are appropriate pricing patterns for business SaaS applications,  
from the point of view of a software vendor?***

The answer of this main research question is addressed in the evaluation section. To answer the main research question, a number of sub questions are formulated, which are stated below.

### **Sub question 1**      *What is a business SaaS application?*

Characteristics of software and Software as a Service are discussed before a definition of a business SaaS application can be given. These characteristics are found by performing a literature study.

A business SaaS application is different from consumer SaaS applications because there is a strong need for configurability. Configurability in the software is needed in order to serve many different customers and to meet their unique business requirements. When tailoring a software application to fit the customers' needs, the application should be customized or configured.

Characteristics of business SaaS applications and how configurability can be realized in a business SaaS application is addressed in chapter 3.

### **Sub question 2**      *What are the implications of offering business SaaS applications?*

There are several implications when a software vendor offers a business SaaS application. For provisioning a business SaaS application, three elements are assumed to be important: costs, metrics and architecture. The implications of these elements are addressed in Chapter 4 as result of a literature study and interviews with experts from the case study company.

The costs of maintaining a business SaaS application (from the point of the software vendor) are investigated. A document study at the case study company is performed to find out what the costs for software vendors are for provisioning a business SaaS application.

Business SaaS applications are configurable. Consequently, there are many differences between the customers on usage and functionality level. Since a business SaaS applications are configurable solutions for both small and large consumers, it is important to measure usage to differentiate customers based on usage. Therefore, metrics for SaaS applications are gathered by performing a literature study on pricing metrics.

Furthermore, different SaaS architectures are discussed. The possibilities and limitations of these architectures are examined because this is taken into account when developing the pricing patterns. The architecture that is the most suitable for a business SaaS application is extensively discussed by providing a theoretical background on this architecture.

**Sub question 3**      *How can business SaaS applications be priced?*

Design research is the main research method used for answering this sub question. Design research involves the analysis of the use and performance of designed artefacts to improve on the behaviour of aspects of Information Systems (Vaishnavi & Kuechler, 2007).

First, problems related to pricing SaaS are identified at the case study company by performing interviews, a document study and direct observations. Next, solutions are found by performing a literature study on pricing software and microeconomics. To present the solutions for dividing the costs in a unified way, it is chosen to describe them as patterns.

The pricing patterns are discussed in Chapter 5, followed by an evaluation of the patterns in Chapter 6. Surveys and interviews with experts from the case study company are held to evaluate the pricing patterns, followed by statistical analysis on these results.

### **1.3 RESEARCH RELEVANCE**

#### *Scientific relevance*

Literature on the topic of software pricing, such as the book ‘Software Product Management and Pricing’ by (Kittlaus & Clough, 2009) and the ‘Pricing Strategy Guideline Framework for Vendors’ by Abdat (2009) is about pricing SaaS. In general, work on SaaS pricing and in particular, pricing patterns for business SaaS applications is sparse.

None of the current literature includes specific, extensive research on identifying pricing problems and different ways of pricing business SaaS applications. Because of the differences compared to on-premises installed software and consumer SaaS applications, it is not possible to apply those pricing methods to business SaaS applications.

#### *Business relevance*

Because of the benefits of Software as a Service, it is expected that the number of business SaaS applications will increase in the upcoming years. Offering business software as a SaaS application is a relatively new. Therefore, software companies currently do not know how to price their business SaaS applications. A solid pricing method is important because it ensures revenue and profit for the software vendor. Also, the availability and integration of a good pricing strategy as part of the overall corporate strategy for SaaS vendors is essential (Spruit & Abdat, 2012). Pricing takes a central role in the strategy of most companies because it directly determines the turnover level and consequently in the long term also the achieved returns (Lehmann & Buxmann, 2009).

This study is especially interesting for software vendors who plan to release a business SaaS application in the near future. Based on the results, those companies can choose one of the proposed pricing patterns, which is the most suitable for their company. This research can also be relevant for companies that already have released a business SaaS application. They can use the findings from this study to examine their current pricing patterns and improve them.

The benefits in general are that companies do not have to investigate and develop a pricing pattern for their business SaaS application from scratch but base them on the pricing patterns proposed in this study. Because multiple pricing patterns are developed, the results are also usable for different types of companies, which face similar problems with pricing their SaaS applications as identified at the case study company.

#### **1.4 OUTLINE OF THE RESEARCH**

In this chapter the problem statement and the purpose of this study is explained and the research questions are addressed. The next chapter discusses the research approach by addressing the different research methods that are used, the validity and the activities that are performed during the research.

Chapter 3 elaborates on business SaaS applications by describing its specific characteristics and the need for configurability in this type of SaaS applications. The next chapter is about the implications of provisioning a business SaaS application, such as the required architecture, the cost elements and measuring usage.

In chapter 5 the pricing problems that are identified at the case study company, are addressed, solutions are provided and examples of the solutions are given.

Chapter 6 describes a study on whether those pricing patterns are applied at the case study company. In addition, the patterns are evaluated and an overview of the consequences is given. This study ends with a discussion and a conclusion of the findings.



## **2 RESEARCH APPROACH**

This chapter describes the research approach. The first section describes the research methods that are used, followed by a description of the case study company. The next section elaborates on the validity of this study. The remainder of this chapter is a section that describes the performed activities, explained with diagrams and gives an overview of the deliverables.

### **2.1 RESEARCH METHODS**

Several research methods are used in this study. This section gives a description of all research methods that are used and how they are applied in this study. A literature study is used to find relevant background information on business SaaS applications and pricing. Design research is chosen as one of the research methods because solutions for pricing problems need to be developed. Furthermore, case study research is used as research method to ensure a high degree of realism for identifying pricing problems.

#### *Literature study*

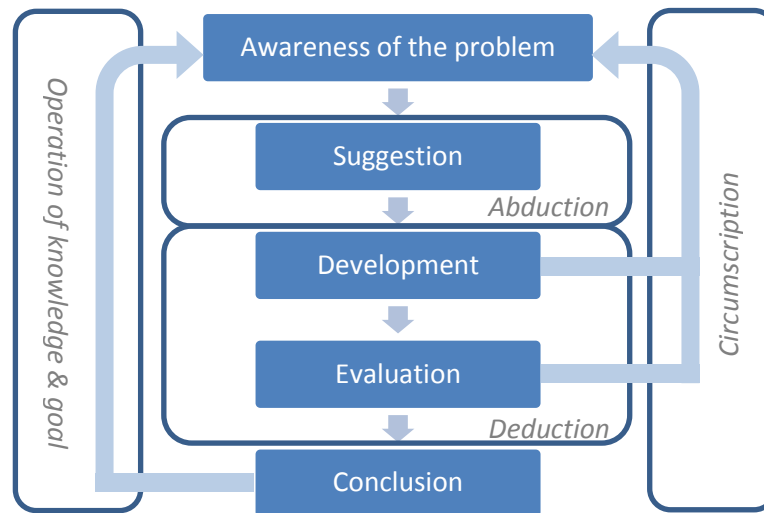
For the literature study, two scholarly search engines were used to search in leading journals: Google Scholar ([scholar.google.com](http://scholar.google.com)) and IEEE Computer Society ([computer.org](http://computer.org)). Pricing models currently described in literature are observed. To accomplish this, a comprehensive literature study had been performed whereby keywords related to pricing software and SaaS were used as search query in the search engines.

#### *Design research*

Design research consists of design cycles that follow a five steps pattern (Takeda & Veerkamp, 1990). The steps are as follows:

1. “Problem awareness – what the problem is and why a solution is needed.
2. Suggestion – in this step, based on desktop or empirical research, a suggestion is made what the solution should be.
3. Development –the artefact proposed as a solution is developed.
4. Evaluation – the process of evaluating how suitable for the problem at hand the solution developed at the previous step is. After the evaluation, the need might arise to return to the suggestion and development steps in order to improve the solution according to the results of the evaluation.
5. Conclusion – the achieved results are summarized and the design research project/cycle is brought to a closure.” (Takeda & Veerkamp, 1990).

The following paragraphs describe how the design cycles from Takeda & Veerkamp (1990) are applied in this research.



**Figure 1 Reasoning on Design Cycle, adapted from Takeda & Veerkamp (1990)**

The general problem (1) is that it is unknown how pricing problems can be solved for business SaaS applications. To identify sub problems for pricing business SaaS, interviews are held with different stakeholders at the case study company. This results in an overview of problems related to pricing a business SaaS application.

Since there are several problems identified it is suggested (2) that for every indicated problem a concrete solution is developed to solve this problem or at least lower the negative impacts. These solutions are based on scientific research and practical usage.

Because there are several problems indicated, multiple solutions for these pricing problems should be developed (3). To develop the solutions a literature study on microeconomics and software pricing is performed. Findings from this literature study and interviews at the case study company are used to develop the solutions for the problems that are identified. These solutions are described as part of a pattern, resulting a structured overview of the problems, the solutions and examples of implementation and usage.

When all patterns are developed, experts of the case study company evaluate these patterns. These experts evaluate the developed solutions on usefulness and applicability. This evaluation results in an overview of the consequences of applying the solutions.

The conclusion (5) includes the final, evaluated pricing patterns. These are depicted in a table with a short description of the problem, the solution, an explaining diagram, an example and an overview of the consequences for using that solution.

### *Case study research*

Case study research is one of the main research methods in this study. Using case study research as research method results in a high degree of realism because it is conducted in a real word setting (Runeson & Höst, 2008). In this section, the case study company is described followed by a description of the research methods that are used within the case study.

### **Case company description**

The case study company is AFAS Software BV. AFAS is an ERP software developer, aimed at the Dutch, Belgian and Caribbean market, located in Leusden, the Netherlands. Currently they have over 10.000 customers, spread over many different market sectors.

AFAS develops software for both business and consumer markets in a socially responsible manner whereby customers, employees and the environment are central. Their product for the consumer market is AFAS Personal, free online house bookkeeping software, recently acquired by buying start-up company Yunoo. AFAS Personal targets the consumer market and therefore it is ignored in this study. Their other software product is AFAS Profit, focusing on the business market. The vision behind the ERP software is that all administrative processes are fully automated in a single software package.

In 2011 they had a turnover of 56 million euro's and a profit of 14.5 million euro's, an increase of over 25% compared with 2010. Their customers can choose between installing the software on premises or access it online.

In 2008 AFAS decided implement their own online solution. Before, they cooperated with other parties to serve customers with online availability of their software, but there were problems with the cooperation. In addition, the customers expected that all responsibility of provisioning software online was taken by AFAS. This resulted in the implementation of AFAS Online whereby AFAS provisioned a complete solution for using the software online, including application management and remote deployment of the software.

From now on 'ERPComp' is used to refer to the case study company AFAS Software BV, and 'ERPOne' is used to refer to the ERP software delivered as SaaS (AFAS Online).

ERPComp is suitable for this study because it is a large software vendor that offers a business SaaS application to 10.000 customers. They are also a typical business software vendor because they develop Enterprise Resource Planning software and they offer it as SaaS.

### **Interviews**

In this study, interviews with experts from ERPComp are conducted to gather information and knowledge for different sections of this study. All interviews are semi-structured so it is possible to anticipate on the answers of the respondents, because it allows to ask new questions resulting from answers from the interviewee.

In the beginning of the research, interviews with product managers are conducted about configuration in SaaS in general and how it is implemented in their own SaaS application. These sessions were merely to gather knowledge from these experts as a starting point for the literature study on tailoring business SaaS applications.

The former manager of ERPOne is interviewed to explore which costs are charged to ERPComp for hosting their SaaS application, to get an idea of the typical costs of provisioning a business SaaS application.

Stakeholders from ERPComp are interviewed to identify problems they currently face or expect to face regarding to pricing their SaaS product. Therefore, exploratory one-hour interviews are conducted to gain knowledge from stakeholders with expertise from different domains as

finance, architecture and product management. The interviewees are asked about the current situation, the problems in the current situation. They were also asked to predict the future in terms of the products that will be launched in the (near) future and what problems then might arise in relation to pricing their software products.

Some follow-up interviews are conducted to verify findings and ask new questions that came up during the research. Especially multiple interviews with the CFO were used as follow-up interviews to elaborate on details. In case of specific questions he could not answer himself, he referred to the responsible expert at ERPComp.

Interviews are also held during the evaluation phase while identifying the usage of pricing patterns at ERPComp. Furthermore a sales manager was interviewed to get a better understanding of the price list and price guidelines. Interviews with the CFO and sales managers are held to identify and verify the appliance of pricing patterns at ERPComp.

### **Survey**

For the evaluation of the pricing patterns, two surveys were made in order to match the requirements for a pricing model to the identified pricing pattern. Because no suitable tool was found for building a sophisticated survey, an online evaluation tool was built including functionality to analyse the results.

The surveys are sent to a selection of employees at ERPComp. The results are statistically analysed to prioritize the requirements and to discover the strength of agreement between the answers from the respondents.

### **Document study**

The document study consists of a selection of documents provided by ERPComp after they were requested. Among these documents were price lists of the products offered by. The price lists gave an overview of all products from ERPComp and the prices for these products. Additionally, they also contained guidelines for sales managers about discount that may be given to customers. These documents were studied to get insight into their current pricing strategies.

Furthermore, ERPComp provided agreements and invoices from several suppliers related to their ERPOnline product, originating from their hosting provider and software supplier. Hereby it was possible to get insight for what kind of things ERPComp is charged and to identify what kind of costs they are.

In addition, access to the intranet and the ERP system of ERPComp was given which was a viable source to gather information by studying documents published on the website and data that was found in the ERP system, such as customer information. An overview of the used documents in this study is given below.

Description	Usage	Source
<b>Price lists/guidelines</b>	Analyse current pricing strategy, identify problems	Provided by CFO
<b>E-mail discussion with customer</b>	Identify pricing problem by analysing a customer's complaint about the price	Provided by CFO
<b>Database size overview per customer</b>	Analyse relationship between number of users and data usage	Provided by manager Customer Service, former manager of ERPOne
<b>Detailed overview of customers</b>	Analyse relationship between number of users and data usage	Intranet, extended with additional information by financial manager
<b>Contracts &amp; Invoices</b>	Analyse payment agreements etc.	Provided by CFO

Table 1 Overview of documents used for the document study

### Direct observations

Observations were made during the attendance at the case study company. A meeting of the board of directors regarding to pricing a new software product was attended. This was a decision making session and both an evaluation of the current pricing strategy. Also, short discussions between employees from the Finance & Controlling department regarding problems with the current pricing strategy were reason to identify problems with the current pricing solutions and used for some example for the pricing patterns.

## 2.2 VALIDITY

It is important to make sure that reliable and valid measures are used in this study. Therefore the following four tests are suggested by Yin (2008).

**Construct validity:** *“establishing correct operational measures for the concepts being studied”* (Yin, 2008).

During the data collection, multiple sources of evidence are used. Interviews are conducted and documents are studied for data collection. Every time when some findings are found at the case study company these were reviewed by a key informant. Usually, this was the CFO, when he was not sure about it he referred to another expert within the case study company. In addition, draft versions of this study were discussed with the CFO.

**Internal validity:** *“establishing a causal relationship, whereby certain conditions are shown to lead to other conditions, as distinguished from spurious relationships”* (Yin, 2008).

Characteristics for pricing models were identified with a group of experts within the case study company. Different experts from the same case study company matched these characteristics to the pricing solutions. For the pricing patterns, the consequences of the patterns (resulted from the evaluation) are matched on the problems that were identified earlier to ensure internal validity.

**External validity:** *“establishing the domain to which a study's findings can be generalized”* (Yin, 2008).

The case study company develops ERP-software, which is a common used type of business software within organizations. It would have weakened the external validity if the case study company developed a specific kind of business software.

Since the case study company has the same kind of customers as their competitors have, it is assumed that the findings can be generalized to other business software vendors that develop ERP software.

**Reliability:** “*demonstrating that the operations of the study can be repeated, with the same results*” (Yin, 2008).

In order to demonstrate that this study can be repeated, short transcripts of interviews are included. Also, collected data from the survey and the complete results of the statistical analysis are included in the appendix.

When interviews are performed, a second person with a similar role within the company is asked to verify the results from the first interview. Statistical analysis are performed to assess the reliability of the results from the survey. Also, the strength of agreement between the respondents is assessed using a statistical method.

A process delivery diagram (PDD) is created, in which the research activities are depicted. This gives a clear overview of all steps and activities that are performed during this study and how they lead to the results.

## **2.3 ACTIVITIES**

In this section, all activities are described that are required to carry out the research. These activities are modelled in a process delivery diagram (PDD). This diagram consists an activity diagram and a class diagram which are integrated, resulting in a process delivery diagram (Van De Weerd & Brinkkemper, 2008).

### *Process Delivery Diagram*

Below the process delivery diagram (PDD) is depicted so it reveals the relations between activities (the process of the method) and concepts which are the deliverables produced in the process.

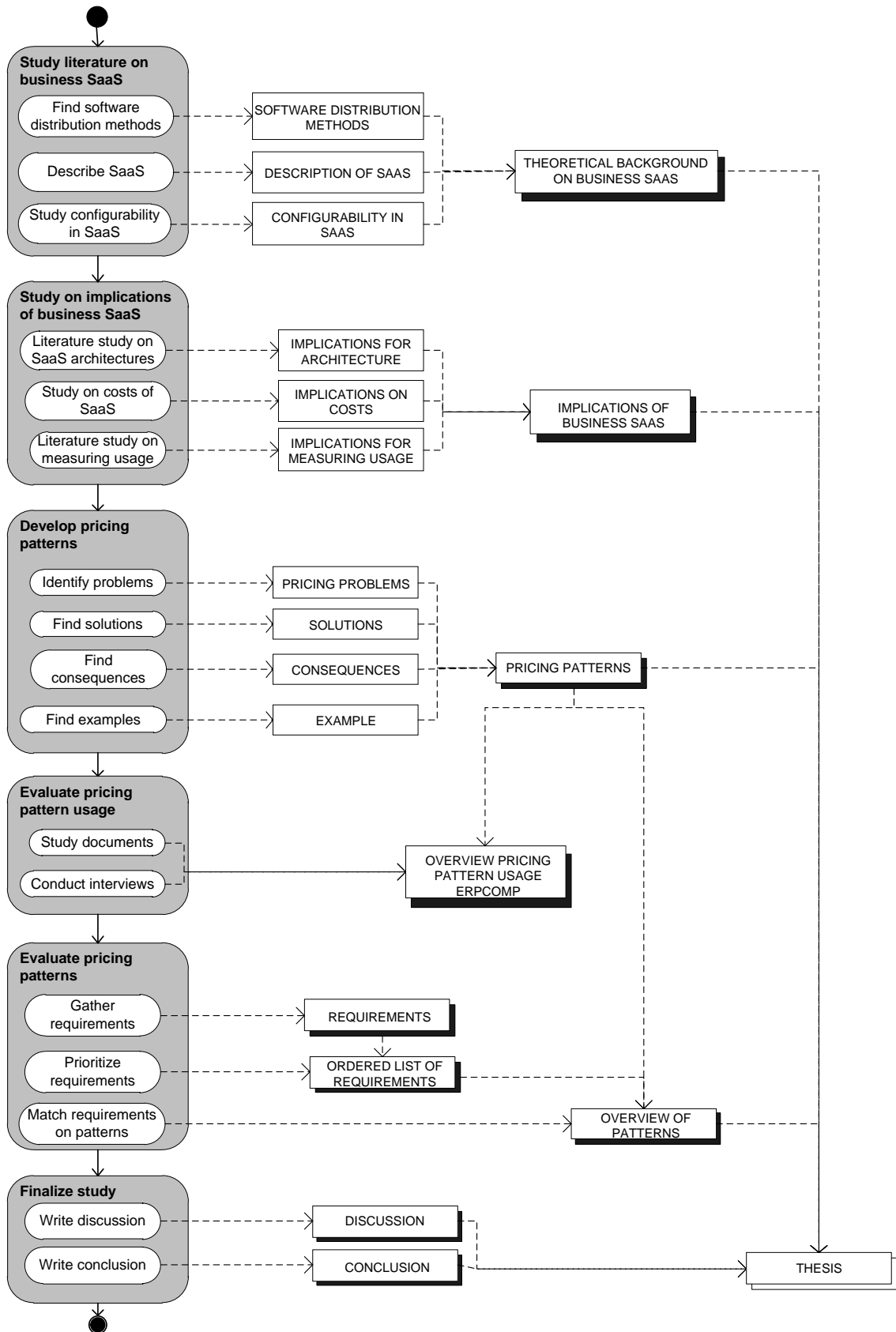


Figure 2 Process Delivery Diagram

*Activity table*

In the table below the activity table is depicted, this table describes all activities mentioned in the PDD (see Figure 2).

Activity	description
<b>Study literature on business SaaS</b>	
Find software distribution methods	Software can be distributed through several channels to the customers.
Describe SaaS	A general description of SaaS is described.
Study configurability in SaaS	Literature study of configurability in SaaS.
<b>Study on implications of business SaaS</b>	
Literature study on SaaS architectures	A literature study is conducted to find different SaaS architectures and to find out which one is the most appropriate for business SaaS.
Studying the costs of SaaS	Interviews and document study at ERPComp are used to find out what the costs or provisioning a business SaaS solution are.
Literature study on measuring usage	A literature study is conducted to find out which metrics can be used of measuring usage.
<b>Develop pricing patterns</b>	
Identify problems	Interviews are held and a document study are conducted to find problems related to pricing the SaaS application of ERPComp.
Find solutions	A literature study on software pricing and microeconomics is conducted to find solutions for each of the identified problems.
Find consequences	Analyse the solution and find consequences using literature.
Find examples	Document study and interviews at ERPComp are conducted to find examples.
<b>Evaluate pricing pattern usage</b>	
Study documents	Documents from ERPComp are studied to find usage of the pricing patterns.
Conduct interviews	Interviews with experts from ERPComp are conducted to verify the findings from the document study and to find other appliance of the pricing patterns.
<b>Evaluate pricing patterns</b>	
Gather requirements	Requirements for a good pricing models are gathered by asking experts from ERPComp by e-mail or by conducting interviews.
Prioritize requirements	The gathered requirements are prioritized by experts from ERPComp with an online evaluation tool.
Match requirements on patterns	Experts from ERPComp match the requirements on the pricing patterns by using an online evaluation tool.
<b>Finalize study</b>	
Write discussion	Describe what the results mean.
Write conclusion	Major findings, outlook of the usage, limitations and further research are described.

**Table 2 Overview of the activities**



### **3 BUSINESS SAAS APPLICATIONS**

This chapter is about tailoring business SaaS applications. The first section elaborates on the need for tailoring, followed by sections about software and Software as a Service. The remainder of this chapter is about realizing configurability in SaaS and discusses terms related to configurability in SaaS.

When tailoring a software application to fit the customers' needs, the application should be customized or configured. In this chapter, the differences between configurability and customizability in SaaS applications are investigated. It is important to get a clear definition of configurability in SaaS because configurability is one of the main aspects in the research. In addition, related terms as customizability and variability are discussed.

In software development there are differences of the term 'tailoring software'. Using the definition above, one would say that if this is applied to standard software it would mean that standard software is configured in such a way that it is right for the particular needs of a customer. It is important to make a clear distinction between these terms, as these definitions are used throughout this research.

In this research the term 'custom-made software' is used to refer to made-to-order system that is specifically made for one customer whereby only one copy is available (Sawyer, 2000) and (Xu & Brinkkemper, 2007). On the other hand, the term 'product software' is used to refer to standard software that is available to many different buyers, following the principle of 'make one, sell many' (Xu & Brinkkemper, 2007). This type of software might be configured to fit to the specific needs of the customer, resulting in 'tailored software'.

A major difference between business-to-business and business-to-consumer software is that the implementation is critical for the customer's business in case of business-to-business software (Xu & Brinkkemper, 2007).

#### **3.1 THE NEED FOR TAILORING BUSINESS SOFTWARE**

Business software, such as Enterprise Resource Planning (ERP) software, is a configurable information system that integrates information and information-based processes within and across functional areas in an organization (Ward & Peppard, 2002).

One of the most important requirements is that it should be configurable to adapt to the different needs of their users across different industries (Klaus, Rosemann, & Gable, 2000). Because customization is an important factor in the adoption of business SaaS (Xin & Levina, 2008), it is important for a software vendor to offer a configurable SaaS application.

This is especially relevant for business software vendors, since variability and extensibility is essential for business applications like CRM and ERP (Aulbach, Grust, Jacobs, Kemper, & Rittinger, 2008).

In a study of configuration and customization perspectives, the following six fundamental causes of needs for tailoring are given (Sun, Zhang, Guo, Sun, & Su, 2008).

- Industry focus differences
- Customer behaviour differences

- Product offering differences
- Regulation differences
- Culture differences
- Operation strategy offering differences

Because every customer is different, each of them has different requirements to the software. Therefore, most business software applications need to be tailored to effectively serve a specific client (Sun et al., 2008). Consequently, in order to serve many different customers and to meet their unique requirements, tailoring of the software is required. A general definition found in a dictionary of the verb to tailor is “to make something so that it is exactly right for your particular needs” (Summers, 2001). Consequently, in relation to software tailoring is making standard software so that it is exactly right for the customer-specific needs.

Although many organizations are using standard software for their customer relationship management and enterprise resource planning, they still have specific wishes concerning the user interface, data, business processes and rules (Nitu, 2009). Accordingly business software vendors should take into account that the same kind of workflow may have different behaviours for different organizations (Nitu, 2009). Also, many customers still ask for variation in functionality according to their specific business requirements (Sun et al., 2008). Therefore tailoring is needed to comply for those customer needs.

### 3.2 SOFTWARE CHARACTERISTICS

From a software development perspective, software is classified in tailor-made software and product software. Product software is defined as a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market (Xu & Brinkkemper, 2007). The differences between product software and tailor-made software are summarized in Table 3.

<i>Software</i>	<i>Typical characteristics</i>
<i>Tailor-made software</i>	
Contractual tailor-made software	Software made for one particular buyer Budget and schedule fixed Penalties for late delivery
In-house tailor-made software	Used to improve efficiency/effectiveness of internal organization Limited number of end-users Possible conflicting interests between IT-department/end-user
<i>Product software</i>	
Business-to-business	Software sold to other business Many different buyers Critical to the buyer’s business
Business-to-consumer	Software sold to individual buyers High volume buyers Market windows and buying seasons Failures can have fatal consequences

**Table 3 Characteristics of tailor-made and product software (Xu & Brinkkemper, 2007)**

From the table above it can be derived that business SaaS applications have many different buyers and is critical to the buyer's business.

Within product software, a distinction can be made between software sold to other business and software to individual buyers (Xu & Brinkkemper, 2007). The focus in this research is on business-to-business software, more specifically on Enterprise Resource Planning (ERP) software.

The traditional approach of distributing software to the user is the "on-premises" model, where the software is installed on the computers on the premises. A company buys applications and deploys them in a data centre that it owns and operates (Aulbach et al., 2008). An organization usually buys a perpetual-use license, without an end date for the use of the bought software product, this is also known as "outright purchase". New product features are released as part of a new version of the software which is usually released once in a few years (Choudhary, 2007), which a customer can acquire by buying a new version of the software.

Within the on-premises model, a distinction can be made between product software and tailor-made software. Product software is available for a wide audience whereas tailor-made software is developed for a specific customer. Product software is sold, leased, or licensed to the general public; offered by a vendor trying to profit from it, available in multiple, identical copies and used without source code modification (Brownsword & Oberndorf, 2000). The implementation of product software involves mainly the adaption of the system to the business needs of the enterprise through system parameterization and the re-engineering of business processes to match system specifications (Stamelos & Angelis, 2003).

### **3.3 SOFTWARE AS A SERVICE**

Due to the opportunities offered by the Internet and consequently the rise of the Software as a Service (SaaS) business model, there is another way to deliver software to the user. SaaS is a deployment technology where the software system and users' data are stored off-site in a central location run by the vendor. The vendor delivers the bundle of IT infrastructure, software applications and services to users through a network (Ma & Seidmann, 2008). Because of the subscription-based model, it is unnecessary to upgrade or make purchases to make use of new features of the software.

Software as a Service (SaaS) is a deployment technology where the software system and users' data are stored off-site in a central location run by the vendor. The vendor delivers the bundle of IT infrastructure, software applications and services to users through a network (Ma & Seidmann, 2008).

In a study on defining Software as a Service the following characteristics of SaaS are identified:

- "Product is used through a web browser.
- Product is not tailor made for each customer.
- The product does not include software that needs to be installed at the customer's location.
- The product does not require special integration and installation work.
- The pricing of the product is based on actual usage of the software."

(Mäkilä, Järvi, Rönkkö, & Nissilä, 2010)

### 3.4 REALIZING CONFIGURABILITY

There are several terms used in the literature related to adapting software to the user's need, this section defines and describes these terms used for tailoring SaaS.

The relation between variability, configurability and tailored software is explained by the figure below. Variability in software leads to configurability, which results in tailored software. To create configurability the software needs variability. When you want tailored software you have to configure it to your specific needs.

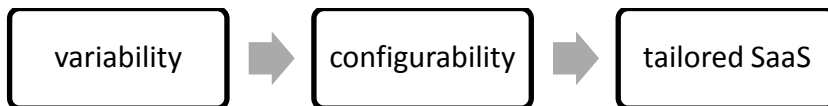


Figure 3 Realizing tailored SaaS

To make a standardized SaaS application suitable to serve specific clients, it should be tailored in to a tenanted SaaS application to meet the customer's unique requirements (Sun et al., 2008).

#### *Variability*

Tailoring software is needed to serve different customers and to meet their business requirements. "In order to serve a lot of different customers in a SaaS environment, software vendors have to comply to a range of different varying requirements in their software product" (Kabbedijk & Jansen, 2011).

Variability is the ability to change or customize a system to use in a particular context (Svahnberg, van Gurp, & Bosch, 2005). Before variability can be built in the software, a variation point should be defined. A variation point represents a point in the software where the variation will arise. Hereby the existence of alternatives is indicated, and each of the chosen alternatives results in a different behaviour of the software (Ghaddar & Tamzalit, 2012).

#### *Configurability*

Configurability allows a unique user experience to each customer of the SaaS application even though the code base is same (Nitu, 2009). Configurability in SaaS aims to provide customers with a many configurable options using a single code base. Hereby it is possible for each customer to have a unique software configuration and experience if it was custom-made (Arya, Venkatesakumar & Palaniswami, 2010).

Tailoring SaaS can be done by configuration, which aims to provide tenants with a multitude of options and variations using a single code base, so each tenant has a unique software configuration (Nitu, 2009). Configuration can support tailoring requirements with a predefined configurable limit (Sun et al., 2008).

Configuration does not involve source code changes of the SaaS application, but it supports variance through setting pre-defined parameters. According to Arja et al. (2010) configurability is the foundation of any SaaS application because without it is a limited ASP.

### *Customizability*

In addition to configuration, there is another way to create a SaaS application to make them suitable for the specific requirements of a customer, which is known as customization. A customized feature is a feature that is tailored to fit the specific needs of a customer (Jansen, Houben, & Brinkkemper, 2010). Different from configurability, customization involves changes to the source code in order to generate functionality that is beyond the options that the configurability offers (Sun et al., 2008).

SaaS providers usually do not offer customizations to their SaaS product. However, organizations require specific functionality to support their business rules and workflow. In other words: they need a highly configurable SaaS application, otherwise standardized software it is not suitable for customer-specific functionality.

## **3.5 CHAPTER OVERVIEW**

By performing a literature study, the need for configurability is addressed. Configurability is needed in order to serve different kind of customers which specific wishes that could be caused by industrial focus differences, customer behaviour differences or culture differences.

Next, software characteristics and SaaS are discussed. From those sections it can be concluded that a business SaaS application is: *administrative software for organizations to support their processes and workflows, with configurability options to adapt the software to the customer's specific needs, provisioned as Software as a Service.*

In order to realize configurability variability in the software is needed which variation points can enable. Furthermore, it was found that customizability involves changes to the source code in order to adapt to the customer's needs and therefore it cannot be used for business SaaS applications.

## **4 IMPLICATIONS OF PROVISIONING A BUSINESS SAAS APPLICATION**

In the previous chapter, the characteristics of a business SaaS application are described. This chapter discusses the implications of business SaaS applications.

The following sections discuss the implications of offering a business SaaS application in terms of costs, architectures and usage. Different SaaS architectures are addressed in this chapter. The possibilities and limitations of this architecture are examined because this taken into account when developing the pricing patterns.

### **4.1 ARCHITECTURE**

This section describes the different SaaS architectures, as found during the literature study. The relationships between the software architectures are described and the benefits and the risks for both the software vendor and the customers are analysed. This section discusses the several SaaS architectures and its characteristics, which is used to define the pricing patterns for business SaaS applications.

To identify the architecture(s) that are feasible for configurability in SaaS, first different SaaS architectures are discussed. Bezemer & Zaidman (2010) distinguish three software architecture principles for the SaaS business model:

- **multi-user:** all users are using the same application with limited configuration options
- **multi-instance:** each tenant gets his own instance of application (virtualization, the easier way of creating multi-tenant like applications)
- **multi-tenant:** each tenant has the possibility to heavily configure the application

#### *Evaluation of SaaS architectures on configurability*

This research focuses on business SaaS applications, therefore the multi-tenant architecture is the most interesting architecture because it supports the possibility to heavily configure the SaaS application (Bezemer & Zaidman, 2010). Multi-tenancy is consequently usually applied to enterprise software such as ERP and CRM (Tsai et al., 2007). Because of the large number of customers, a multi-tenant solution is needed to implement the varying customer's requirements (Kabbedijk & Jansen, 2011).

To offer configurability to a high number of users, multi-tenancy is the most appropriate architecture. A multi-tenant application is an application that enables different customers to use the same instance of a system, without necessarily sharing data or functionality with other tenants. These tenants have one or more users who use the web application to further the tenant's goals (Jansen et al., 2010). It is particularly interesting for business software vendors to offer their application within a multi-tenant environment, because business software should be configurable to accommodate the diverse needs of users across most sectors of the economy (Klaus et al., 2000).

### Types of multi-tenancy

Because the multi-tenant architecture is assumed being the most appropriate solution for business SaaS applications, different types of multi-tenancy are described in this paragraph. Guo et al. (2007) indicate that there are two kinds of multi-tenancy patterns:

- **Multiple instances:** supports each tenant with its dedicated application instance over resources.
- **Native multi-tenancy:** supports all tenants by a single shared application instance of various hosting resources.

It is important to note that the first pattern (multiple instances) by Guo et al. (2007) is not a real multi-tenant pattern according to the definition by Bezemer & Zaidman (2010). There are three approaches to implement multi-tenant databases according to (Jacobs & Aulbach, 2007), these are similar to the multi-tenancy variants as described by (Bezemer & Zaidman, 2010):

1. **Shared machine:** each customer gets their own database process and multiple customers share the same machine.
2. **Shared process:** each customer gets their own tables and multiple customers share the same database process.
3. **Shared table:** data from many customers is stored in the same tables.

The ‘shared table’ approach can be seen as the highest level of multi-tenancy database implementation approaches. Please note that the shared table approach is similar to both the native multi-tenancy pattern described by (Guo, Sun, Huang, Wang, & Gao, 2007) and the ‘pure multi-tenancy’ variant by (Bezemer & Zaidman, 2010). Therefore, it can be concluded that there is an agreement on the highest level of multi-tenancy, since they are all similar. Microsoft (Chong et al., 2006) makes the distinction between:

- **Separated database:** different database for each tenant.
- **Separate schema:** multiple tenants are hosted in the same database, with each tenant having its own set of tables that are grouped into a schema created specifically for the tenant.
- **Shared schema:** the same database and the same set of tables to host multiple tenants’ data.

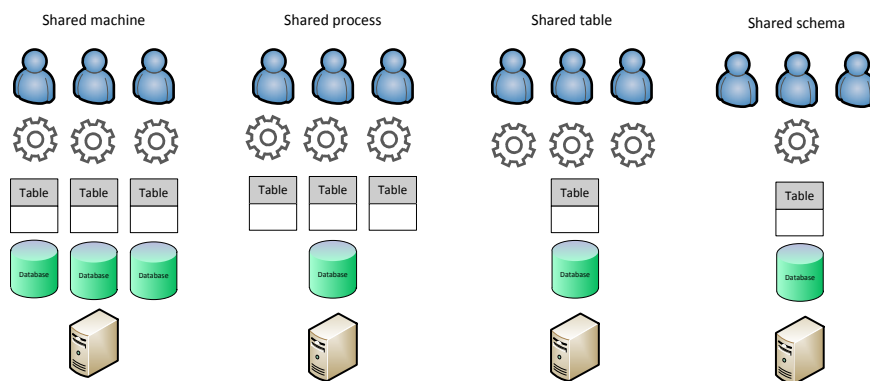


Figure 4 Multi-tenant implementation approaches

The figure above is the result of the combined views of Chong (2006), Jacobs & Aulbach (2007) and Bezemer & Zaidman (2010).

### *Benefits and liabilities*

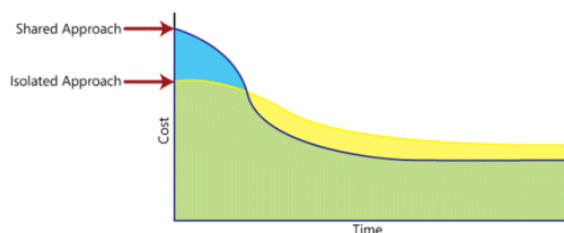
By performing a literature study, the benefits and liabilities of provisioning a multi-tenant SaaS application are discovered. These are categorized in the following four categories: costs, hardware & performance, development & maintenance and security. The next paragraphs discuss the benefits and liabilities for each of these categories.

#### **Costs**

By provisioning a SaaS application in a multi-tenant architecture, a software vendor can reduce costs. Since multi-tenancy requires a small infrastructure, compared with a multi-user architecture, it reduces costs on hardware and infrastructure (Tsai et al., 2007). Similarly, because of the large-scale capabilities of this architecture and the higher utilization of the hardware it results in lower costs for hardware (Hamilton, 2007; Motahari-nezhad, Stephenson, & Singhal, 2009) and hosting (Kwok, Nguyen, & Lam, 2008) .

As the number of instances is much lower in a multi-tenant architecture, the deployment of applications, updates and new versions is much easier and therefore cheaper (Bezemer & Zaidman, 2010). Furthermore, multi-tenancy significantly reduces delivery cost for a large number of customers (Gao et al., 2011), resulting in an increased profit margin for the software vendor (Guo et al., 2007).

A disadvantage of (moving to) a multi-tenant architecture might be the initial and start-up costs for development or reengineering an existing single-tenant SaaS application into multi-tenant application (Bezemer & Zaidman, 2010). The software possibly should be reconstructed because it was actually built for a different architecture. However, since the reduced cost on infrastructure and hardware resources, in the long term it is profitable as illustrated in Figure 5.



**Figure 5 Cost over time for a hypothetical pair of SaaS applications; one uses a more isolated approach, while the other uses a more shared approach (Chong et al., 2006)**

#### **Maintenance & Development**

A multi-tenant SaaS application is considered to be more affordable for customers because of the capabilities for customization and scaling (Kwok & Mohindra, 2008).

Improvements on management efficiency may also be accomplished because of a uniform method for administering the software (Jacobs & Aulbach, 2007). A multi-tenant application may also benefit the customers by saving money and time while having immediate access to the latest software functionality (Kwok et al., 2008). Furthermore, by leveraging a multi-tenant SaaS application, software vendors can significantly ease maintenance operations (Gao et al., 2011).



On the other hand, software vendors are worried that multi-tenancy might introduce additional maintenance problems because these new SaaS applications should be highly configurable and therefore add complexity, which is expected to affect the maintenance process (Bezemer & Zaidman, 2010). Similarly, extensibility support for the SaaS application is, since shared structures are harder to change individually (Aulbach et al., 2008).

Other disadvantages of adopting multi-tenancy as a software architecture might be that the source code should be rewritten (Tsai et al., 2007) and the increased complexity in deployment and management (Guo et al., 2007).

### **Hardware & performance**

Considering the hardware and performance, multi-tenancy improves the utilization rate of the hardware by placing multiple customers on the same server (Bezemer & Zaidman, 2010) and multi-tenancy allows pooling of resources (Jacobs & Aulbach, 2007).

Since multi-tenancy supports many customers on the same hardware resources, the software vendor should be aware that intensive use from one customer does not affect the performance of another customer (Guo et al., 2007)

### **Security**

A multi-tenant architecture can weaken security, because instead of access control on infrastructure level it should be performed at the application level (Aulbach et al., 2008). This is important issue because a security breach may result in the exposure of data from one customer to the other which might be a competitor (Bezemer & Zaidman, 2010). Therefore multi-tenant SaaS applications require “adequate, auditable, protection against the risk of data leakage between customers” (Tsai et al., 2007).

## **4.2 COSTS**

The cost elements of maintaining business SaaS applications, from the point of the software vendor, are investigated. A document study is performed to find what the costs are for software vendors. Interviews with both financial managers and software architects are held to verify the completeness of the costs. At the case study company the costs of deploying their software as a service are investigated.

The research of the costs is done by a document study: invoices and license contracts found in the bookkeeping software or provided by the CFO or financial manager of the case study company are the main source for this study. To ensure the completeness of the document study on the costs, the results are discussed with the CFO and the Director Architecture & Innovation to verify the results and adjusted if necessary.

During one of the interviews with the Chief Architecture & Innovation mentioned that the costs of running the ERP Online product exist of:

- hardware costs
- energy costs
- software licenses

The costs of hardware are relatively low compared to the energy and licensing costs. The latter two have a high impact on the total costs of running ‘ERP Online’. The Chief A&I specially

mentioned that the license costs for their database software are high and the supplier significantly increased the prices recently.

Interviews with the Chief Financial Officer, the Director Architecture & Innovation and the former manager of ERPOnline are conducted to identify the different actors for provisioning a software product as Software as a Service. This resulted that in general there are the following actors when offering SaaS: an infrastructure provider, a platform provider, a software provider and the end-customer.

### SaaS cost structure

This section describes the cost for the software vendor when they provide Software as a Service. The costs might differ depending on the chosen deployment model for delivering SaaS. A distinction is made between software, platform (operating system) and infrastructure (hardware and hosting). These different combinations result in seven different combinations of deployment possibilities as depicted in the table below.

	On premises		Service / XaaS				
	A	B	C	D	E	F	G
<b>Software</b>	User	Vendor	User	User	Vendor	Vendor	Vendor
<b>Platform</b>	User	User	User	Provider	Vendor	Vendor	Provider
<b>Infrastructure</b>	User	User	Provider	Provider	Vendor	Provider	Provider

Table 4 Different combinations of deployment possibilities

The following responsible organizations can be distinguished:

- User: an organization or person who uses the software.
- Vendor: a software vendor that develops the software and offers it to the customer.
- Provider: a service provider facilitate a certain service such as the platform or the infrastructure, but not the software itself

Note that the table describes the responsible organizations, so in case of option E, it could be the case that the vendor hires the platform and the infrastructure from an external party, but offers the whole package (software, platform and infrastructure) to the user of the software. Each of those possible combinations is discussed in the table below.

Option	Description
<b>A</b>	The user develops software in-house and is responsible for both the infrastructure and platform.
<b>B</b>	Software from a software vendor is installed on the platform running on the infrastructure owned by the user on the premises.
<b>C</b>	The software developed by the user, runs on their own maintained platform, but the infrastructure is provided as a service by a provider.
<b>D</b>	The software developed by the user is provided as a software as a service because it runs on an external platform and infrastructure, maintained by a provider.
<b>E</b>	The software vendor developed the software and offers the software on their platform and infrastructure maintained by the vendor.
<b>F</b>	The software and the platform are both maintained by the vendor, but the infrastructure is served by a provider.
<b>G</b>	The software, developed by a software vendor, runs on a platform and infrastructure maintained by an external provider.

Table 5 Description of possible deployment models

### Software ecosystem

A software ecosystem is a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships (Jansen, Finkelstein, & Brinkkemper, 2009).

The cash flow in a SaaS ecosystem is typically as follows: The cloud provider charges the SaaS provider for the usage of the cloud. The SaaS provider charges the SaaS user for the application, but since the SaaS provider is also charged for the cloud usage they will charge the end user for the costs of the cloud usage.

To identify the costs involved in providing a business SaaS application a software supply network (SSN) is modelled, as part of the software ecosystem of the case study company. A SSN is a series of linked software, hardware, and service organizations cooperating to satisfy market demands (Jansen, Brinkkemper, & Finkelstein, 2007). A SSN specifies the suppliers, customers, intermediaries and trade relationships from the perspective of the company of interest. The trade relationships are divided in products, services, finance and content. By modelling a SSN, the costs between the suppliers and the company of interest are clarified, which is part of the research of identifying the costs of providing a business SaaS solution.

Figure 6 depicts the ERPComp software supply network for ERPOne. ERPComp is at the centre of the diagram. ERPComp directly supplies its customers with the ERPOne product (S.2) and the customer pays a fee for it. This fee is usually charged on monthly basis and is based on the number of named users.

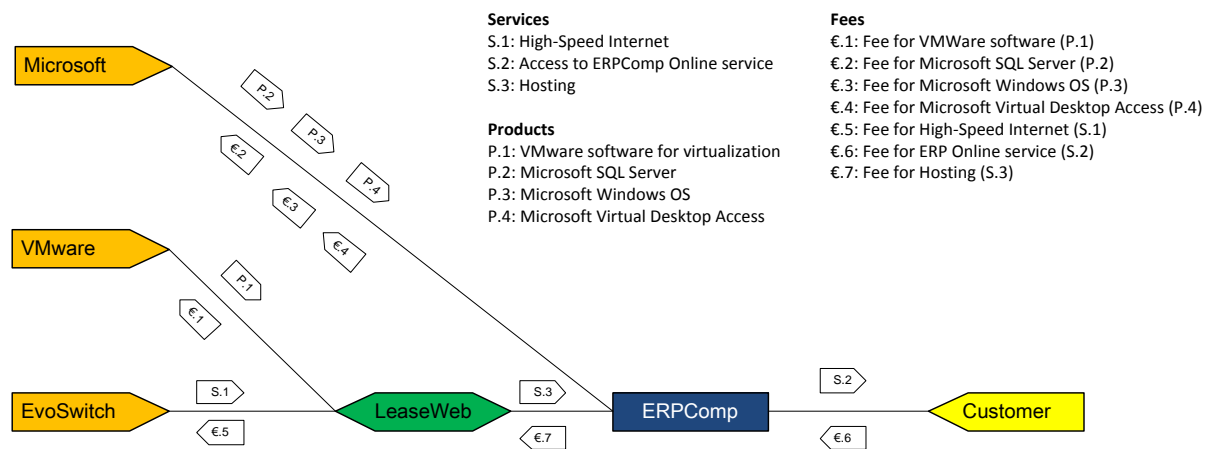


Figure 6 Software Supply Network for ERPOne

### **4.3 MEASURING USAGE**

Business SaaS applications are highly configurable solutions for both small and large consumers, therefore it is important to measure usage. Metrics for business SaaS applications are gathered by performing a literature study on pricing metrics.

According to Kittlaus & Clough (2009), metrics can be many things as long as it tracks to increased customer value: quantity, usage, number of transaction, etcetera. Since there are many different ways of perceiving value to a business SaaS application there are also many different metrics. The metrics that are found during the literature study are included in the appendix.

A distinction can be made between usage-dependent metrics and usage-independent metrics (Lehmann & Buxmann, 2009). Typical usage-dependent metrics are number of transactions, memory requirements and time. For usage-independent metrics named user, concurrent user and number of servers are common used metrics.

Lehmann & Buxmann (2009) mention that it is important that the chosen metric should be strongly linked to what the customer considers to be fair.

Kittlaus & Clough also suggest that the metric that a software vendor chooses, leads to predictable costs for the customer but also for predictable revenues for the software vendor.

Only implementing measurements into the source code it not enough for using metrics in software. Also the back office systems should be support the tracking of the software for invoicing (Kittlaus & Clough, 2009), but also monitoring usage can lead to administrative costs (Lehmann & Buxmann, 2009).

### **4.4 CHAPTER OVERVIEW**

This chapter describes the implications of provisioning business SaaS applications. First different SaaS architectures are identified and evaluated. The multi-tenant architecture is the most appropriate architecture for business SaaS applications since it supports configurability at the highest level.

When studying the costs it turned out that a software vendor will mainly face costs for the hosting and software licenses. For measuring usage of a business SaaS applications metrics are needed in order to price the software accordingly. The main differences between metrics is whether they are usage dependent or not.

## 5 PRICING PATTERNS

This chapter describes the identified problems with pricing SaaS and how these problems are identified. To identify problems with pricing business SaaS applications, a document study, a literature study and multiple interviews are conducted.

Pricing problems are identified during interviews with multiple stakeholders at ERPComp (including the CFO, Director Architecture & Innovation, product managers and financial managers). Since it might be possible that the experts at ERPComp could not think of all possible problems with pricing their SaaS application, a document study at ERPComp is performed. The following definition is used to indicate a pricing problem.

**Pricing problem:** *a problem regarding pricing that software vendors face because they offer a business SaaS application to their customers.*

The price list is studied to identify potential pricing problems. These potential problems were verified with sales managers during an interview. Additionally, a short literature study is conducted to identify additional problems related to pricing business SaaS applications.

A literature study on several topics is conducted to find decent solutions for those pricing problems, such as software pricing. Principles from microeconomics are also adopted in the pricing patterns. A solution is presented for each of the pricing problems, based on the literature study.

**Pricing solution:** *a comprehensive solution for an identified pricing problem a software vendor.*

As there are multiple solutions and therefore there is the need to describe the problems and solutions uniformly. This is done by describing them as patterns. Each solution is described in a structured way starting by addressing the problem, next the solution is explained with a supporting diagram, followed by a real example.

The table below gives an overview of the activities performed to develop the pricing patterns.

Activity	Method	Description
<b>Identifying problems</b>	Document study	Documents including invoices, price lists and price guidelines are studied.
	Interviews	Interviews with experts from ERPComp are conducted to verify findings from document study and to find other problems.
	Direct observations	The study is conducted at ERPComp and direct observations were often used as starting point to identify pricing problems.
<b>Find solutions</b>	Literature study	Literature on microeconomics is studied to find solutions for the pricing problems.
<b>Evaluate solutions</b>	Survey	With two surveys, sent to experts within in ERPComp, the pricing solutions are evaluated.

**Table 6** Used methods for developing pricing patterns

## 5.1 PATTERNS

A pattern is “a description of a solution to a problem found to occur in a specific context” (Meszaros & Doble, 1997). By describing the solutions as patterns results in all solutions are presented in a uniform manner. To compare and evaluate the found solutions, there is a need for presenting the problems, solutions and consequences in a structured way. Therefore, it is chosen to describe them as patterns.

The identified pricing patterns are discussed following a specific template. According to (Wellhausen, 2011), a pattern contains at least five sections, described in Figure 7.

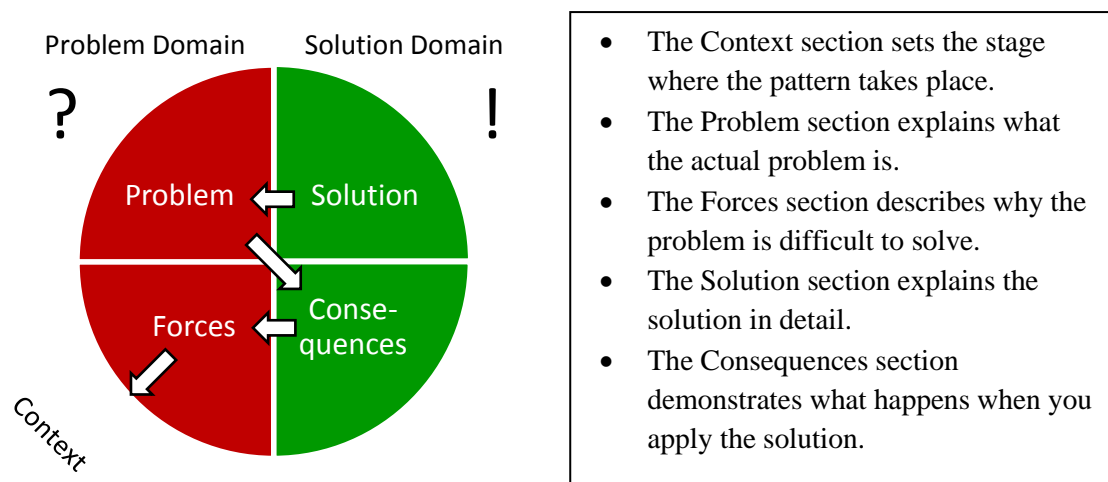


Figure 7 Essential pattern sections and their writing order, adapted from Wellhausen (2011)

For this research, the ‘context’ is considered the same for all pricing patterns, there for the context is not mentioned by each pattern. The context for all pricing patterns is: *A business software vendor that offers a configurable SaaS application, hosted in a private cloud by an external hosting partner.*

First, the problem section describes the problems and how they are identified at ERPComp. The need for a solution is explained as well as a description of what the problem makes a problem. This is followed by a paragraph about the solution. The solution is described, next a diagram is depicted that illustrates the problem and the solution, followed by a supporting text that explains the diagram.

Every pricing pattern ends with an example, whereby real examples and real data from ERPComp are used as much as possible to explain the possible appliance of the solution in a real life setting.

Consequences are also part of a pattern, but these are elaborated in the next chapter where experts from ERPComp evaluate the solutions. Consequently, the different elements of the pricing pattern are discussed separately, but a coherent overview is given at the end of the next chapter. When the consequences are added to the pattern, the pricing pattern is seen as complete and defined as follows.

**Pricing pattern:** *a structured description of a pricing problem, the provided pricing solution, a detailed example and an overview of evaluated consequences.*

This research focuses on pricing business SaaS applications, therefore the patterns (problems, solutions and consequences) are written from a software vendor’s perspective. While developing the solutions for the identified problems, the specific characteristics of business SaaS applications are taken into account, based on the findings in the previous chapters.

The following sections of this chapter describe the pricing patterns for business SaaS applications. For the examples, a description of a known use is given or the example is constructed using data from the case company.

Icons and symbols are used in diagrams to explain the solution for the pricing problem. The table below is a legend for the used symbols.




Symbol	Explanation
\$	fixed price
\$x	variable price
⇒\$	influences price
\$ <sup>2</sup>	different prices occur within one pattern, indicated by the number
	customer of the SaaS application
	usage indicator
	time indicator

Table 7 Legend for used icons and symbols

## **5.2 USAGE-DEPENDENT PRICING**

### *Problem*

Different users of a SaaS application have different usage levels and therefore the costs per customer differ for the software vendor. This is a risk for the software vendor since it might be possible that the intensive use of the SaaS application by some customers, the costs will exceed the (fixed) price. A customer who heavily uses the software pays the same price as a customer that use the software only once in a month.

For example, once a word-processing software program is installed, it does not matter whether a customer processes one or one hundred documents a day, since the onetime fee is already paid. If a company intensively uses the software, the software might be worth more than the price asked by the software vendor. For an occasional user on the other hand, the software is worth less and the (fixed) price is too high.

The willingness to pay a certain price differs among the customers, they have different reservation prices. The reservation price is the highest price a buyer is willing to pay for goods or a service (Mankiw & Taylor, 2006). With a fixed price, there is no relation between the reservation price and the price of the SaaS application.

Since the vendor might have higher costs when usage is higher the software vendor's profit might decrease.

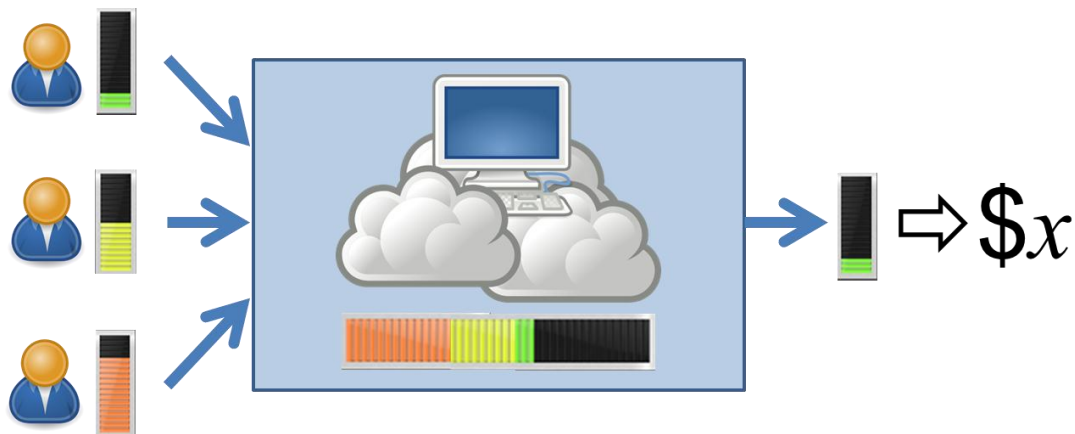
### *Solution*

Usage-dependent pricing is the solution for the problem of different costs per customer for the software vendor, caused by differences in usage.

Since SaaS provides new possibilities to charge customers, it is possible to charge to subsequent use because the software vendor can monitor the usage of the application by the customer. Normally, with on-premises installed software the customer is allowed to use it indefinitely. This is also caused by the fact that the software vendor has little to no possibilities to monitor the usage of the software when it is installed on premises.

The assumption for usage-dependent pricing is that if a customer uses the software more intense, the reservation price will be higher because it has more value to the customer. This type of pricing is illustrated depicted by Figure 8. With usage based pricing a specific price is assigned to each level of usage, whereby the firm does not distinguish between customer types (Sundararajan, 2004).





**Figure 8 Illustration of usage-dependent pricing**

On the left side of the figure above, three customers are illustrated. Each of them have a different usage of the software as illustrated by the coloured bars. Since all customers use the same SaaS application, the sum of all customers results in the total usage. The software vendor might be faced with costs from the hosting provider that depend on total usage of the servers. Since the customers have a different usage level of the SaaS application they are charged according their usage as indicated by the grey bar and the dollar sign at the right end of the illustration. Hereby the customers are charged based on their usage (Miranda, Baida, & Gordijn, 2006).

The price of the product should be higher if the customer is willing to pay more for the product. This willingness might be influenced by the expected usage of the software. This is possible because one customer will use the software more often than another customer. The costs of provisioning SaaS are partly of a variable nature (Lehmann & Buxmann, 2009). Therefore it is reasonable to charge these costs also on a variable basis to the customer.

*Example*

Windows Azure is a cloud hosting platform by Microsoft that is fully usage-dependent. Using a calculator online<sup>1</sup> a customer can indicate the required size of the database, the required bandwidth, the number of instances, etc. There are thirteen of these metrics in total. The customer only has to pay for what is used without any upfront costs. In the table below some examples of variables are given.

Variable	Amount	Price
Storage	1000 GB	\$93.00
Storage transactions	100 million	\$10.00
SQL Database size	100 MB	\$ 5.00

**Table 8 Usage-dependent pricing from Windows Azure**

<sup>1</sup> <http://www.windowsazure.com/en-us/pricing/calculator/>

### 5.3 SITUATION-BASED PRICING

#### *Problem*

With usage-dependent pricing, it is hard to calculate the costs in advance. They are usually less technical and do not how much disk space they are going to use for example.

Because it is hard to calculate usage in advance, the costs are unknown and therefore result in unpredictable expenses for the customer. The usage of a SaaS application can be estimated beforehand but the precise usage is only known after it is actually used.

#### *Solution*

Situation-based pricing solves this problem by using metrics that are not related to usage, to define the price. Therefore, metrics that define the situation for situation-based pricing are needed. Those metrics do not measure the actual usage of the software but they give an indication in what kind of situation the software is used. Examples of metrics for situation-based pricing are given by Lehmann & Buxmann (2009) and are listed in the table below. An extended list of variables that can be used for situation-based pricing found during the literature study is included in the appendix.

Situation-based metrics
Named user
Concurrent user
CPU
Key performance indicators
Machine, server

Table 9 Situation-based (usage-independent) variables according to Lehmann & Buxmann (2009)

Non-technical metrics, related to the customer’s specific situation are used for situation-based pricing. For situation-based pricing, metrics are used that are not related to the actual use of the software (Lehmann & Buxmann, 2009). Note that Lehmann & Buxmann (2009) used the term ‘usage-independent pricing’ to indicate situation-based pricing. In this study it is chosen to use situation-based pricing because usage-independent pricing would implicate all pricing patterns except usage-dependent pricing.

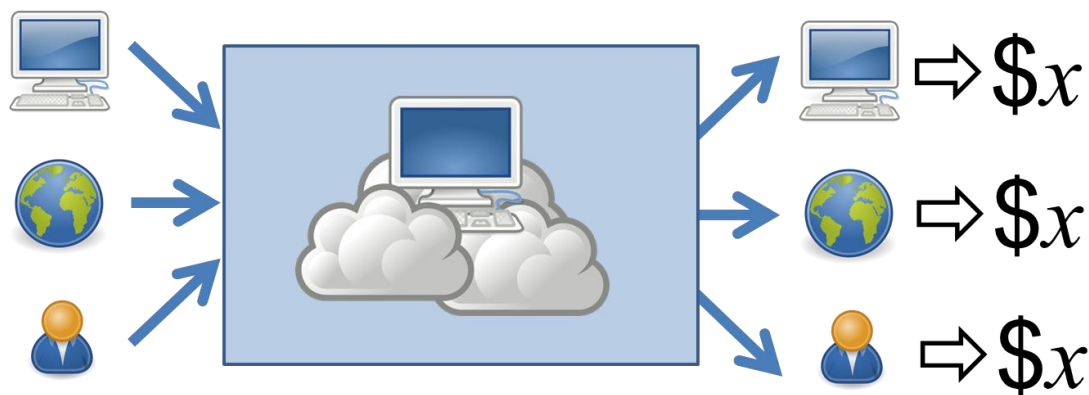


Figure 9 Illustration of situation-based pricing

The illustration above depicts three different customers. Each of these customers have a different ‘situation’, illustrated by the icons on the right side of the user. These customers make use of the SaaS application, illustrated by the blue arrows. The price is defined based on the situation, illustrated by the corresponding icons.

A metric related to the situation of the customer should be chosen. The number of occurrences of that metric times the defined price per metric, result in the total price. Usually the calculation is: ‘situation-based variable’ times ‘number of occurrences’.

### Example

A metric like number of named users can be used for situation-based pricing. Hereby the number of users is leading for defining the price of using the software. The software vendor defines a price per user that the customer has to pay, for example €5,- per month. If the customer’s company has 20 employees, the total price per month is €100,-.

At the moment the number of employees at the customer changes, also the monthly price changes according to the new number of users.

## 5.4 COST-BASED PRICING

### Problem

The costs of developing and maintaining a SaaS application are not equally divided among the customers, when every customer pays the same price.

When the software vendor develops a new feature that requires a lot of development time and therefore money, this functionality becomes available for everyone who uses the software. In that case every customer pays for the development costs, even those customers who do not require and use the new (expensive) functionality.

### Solution

With cost-based pricing the customers are charged based on the costs for the functionality that is needed.

Cost based pricing is different from usage-dependent pricing and situation-based pricing since those prices are based on (perceived) value for the customer. Also, the actual cost of producing the software are taken in to account.



Figure 10 Illustration of cost-based pricing

The diagram above explains cost based pricing. On the left side three different modules are depicted with their corresponding (development) costs. The SaaS application consists out of these three modules. When a customer uses functionality from one of the modules he is charged accordingly.

### Example

The number of development hours can be used to calculate the costs of the development for a specific function. First, make a table with the functions and the required development hours. Second, define the following parameters: margin, break-even time and costs per development hour. To calculate the total development costs, the costs per development hour should be multiplied with the total development hours. When users are charged per month the total costs are divided by the number of break-even months. The values depicted in the table below.

Hence, a customer uses the application according to the data in the table below. Usage can be measured in several ways, i.e. hours or number of transactions. Next, the needed development hours for a function should be multiplied with the usage. For each function the relative usage is used to calculate the costs per month. This is increased with the defined margin resulting in the costs per month.

Function	Description	Dev. hours	Financial	
<b>a</b>	Declaration F-biljet 2012	378	Margin	30%
<b>b</b>	ASite: Document management	892	Costs per development hour	100
<b>c</b>	Wages and Retirements Declarations	134	Break-even months	36
<b>d</b>	Warehouse stock movement	232	Total development costs	€ 163.600,00
<b>Total</b>		<b>1636</b>	Costs per month	€ 4.544,44
			Revenue per month	€ 2.238,89

Customer	Function	Usage		Costs per month		Price	
<b>1</b>	a	5	35	€	194,44	€	252,78
<b>1</b>	c	10	200	€	1.111,11	€	1.444,44
<i>Subtotal</i>				€	<b>1.305,56</b>	€	<b>1.697,22</b>
<b>2</b>	d	3	75	€	416,67	€	541,67
<i>Subtotal</i>				€	<b>416,67</b>	€	<b>541,67</b>
<b>End total</b>				€	<b>1.722,22</b>	€	<b>2.238,89</b>

## 5.5 FLAT FEE PRICING

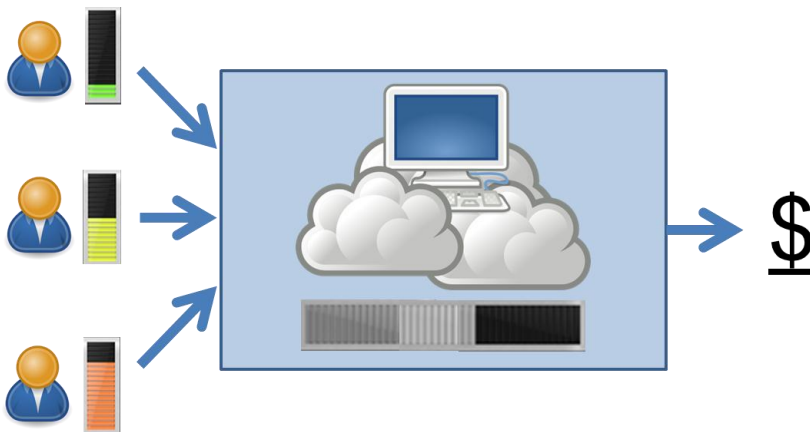
### *Problem*

Calculating and measuring the usage of the SaaS application might be difficult, resulting in unpredictable costs for the customer and unpredictable income for the software vendor.

A problem with a usage dependent pricing pattern is that tracking the usage of the software by the customer might be complicated. In addition, the costs for the customer are unknown in advance and hence the revenue for the provider. Furthermore, the phenomenon 'too cheap to meter' might be a problem to charge accordingly usage. This occurs when the implementation of measurements and metrics is not in line with the costs of using the SaaS application.

### *Solution*

To avoid difficulties with measuring usage a software vendor could opt for flat fee pricing. Hereby the software vendor specifies a price to be paid by the customer in exchange of unlimited usage of the SaaS application (Sundararajan, 2004). This solution is illustrated below.



**Figure 11** Illustration of flat fee pricing

On the left side there is a customer making use of a SaaS application, the usage is irrelevant. Although the software vendor might be faced with additional costs if the customer intensively uses the software, the customer only has to pay a fixed price, there are no transaction costs associated, the customer simply has to pay the pre-specified price (Sundararajan, 2004). A customer has to pay a single, fixed price regardless of usage or any other metric (Kittlaus & Clough, 2009).

A flat fee pricing pattern is available in two flavours: a onetime fee or recurring costs. In both cases, the price is fixed, but customers might have to pay it only once or have recurring payments, for example once a month or once a year. In either case, unlimited use of the service is included.

### *Example*

The usage of two different customers is depicted in the table below. Customer A has used the SaaS application for 2 hours in the last month, while customer B has used the same application for 25 hours. Although there is a significant difference between the usages of these two customers, they have to pay the same price for the SaaS application.

Customer	Hours of usage	Costs	Price
A	2	€2	€20
B	25	€25	€20

Table 10 Example of flat fee pricing

Note that the profit for the software vendor for customer A is €18, while there is a loss for customer B of €5. However, the overall profit is €13.

## 5.6 TWO-PART TARIFF PRICING

### Problem

When applying a pure usage based model the problem arises that the revenues for the vendor are uncertain. Also, in many cases the (potential) customers do not know in advance what their usage of the software will be.

In case of a pure usage-based pricing pattern, it might be that the customer will not promote the use of the software, because the costs are lower as the customer does not use the software.

### Solution

Applying a two-part tariff whereby the customer pays a variable price based on usage upon a fixed price results in less uncertainty about the income/costs.

The customer pays a fixed amount plus an additional usage based charge (Miranda et al., 2006). The illustration below explains this solution. On the left side, there are there customers using the SaaS application on a different usage level.

The price that the customer has to pay is divided in two parts. There is a fixed fee that the customer has to pay on a monthly or a yearly basis. On top of that, there is a fee which is defined by the level of usage from the customer. The fixed fee can be used to recover the costs on the development. Similarly, the usage-dependent fee can be used to charge for the transaction costs and marginal costs, which are caused by usage of the software.

At this moment the most favourable pricing pattern for this type of software is the subscription combined with the usage-based model (Abdat, 2009).

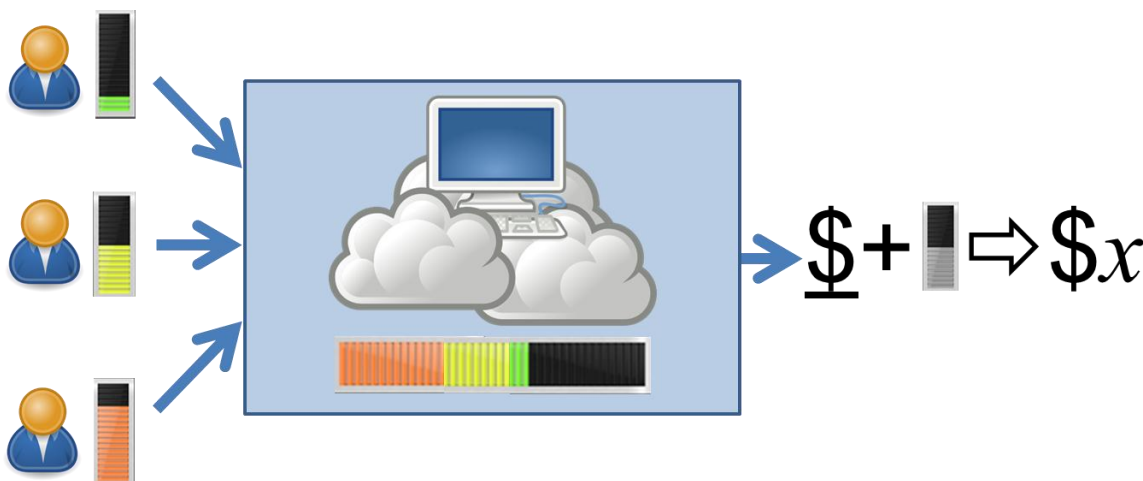


Figure 12 Illustration of two-part tariff pricing

### Example

The software vendor can choose to charge customers a monthly fee of €20 and a usage-dependent fee based on the number of logins. ERPComp itself is charged for using software that facilitates remote access. These charges are based on the number of remote logins on the webserver by the customers from ERPComp. These costs are about €0.20 per login.

In the table below two different customers are depicted. Customer A, has 30 logins because he uses the software every day. Customer B also uses the software on a daily basis, but has five employees resulting in 150 logins.

Customer	Number of logins	Login costs	Fixed price	Total price
A	30	€6	€20	€26
B	150	€30	€20	€50

Table 11 Example of two-part tariff pricing

In this example, the two parts are the login costs and the fixed price whereby the login costs are usage-dependent. Based on the costs of €0.20 per login, customer A is charged €6 and customer B €30 for their logins to the SaaS application. The total costs are mentioned in the table.

## 5.7 PEAK LOAD PRICING

### Problem

A software vendor might face the problem of peak hour usage: a high load on the servers at certain times. Although the high load is only at some specific times, the software vendor should account for this by having enough server capacity. This results that this capacity is only used during peak hours but not during off-peak hours.

During the interviews, both the CFO and the Director Innovation and Architecture mentioned this problem. At some specific points in time there is a high load on the servers because customers use a certain function of the software that has a high impact on the resources. A common example at ERPComp is pay rolling, since a lot of customers do pay rolling for their employees at the end of the month.

In the section about SaaS architectures it was founded that since multi-tenancy supports many customers on the same hardware resources, the software vendor should be aware that intensive use from one customer does not affect the performance of another customer (Guo et al., 2007).

Also Armbrust et al. (2009) explain this problem: “provisioning a data centre for the peak load it must sustain a few days per month leads to underutilization at other times” (Armbrust et al., 2009). From an economic perspective one could say that the customers have different utility functions during peak hours. Since the SaaS provider has to account for the peak hours they should have the capacity always available and running in their ‘private cloud’, they have continuous costs of maintaining these servers, which are also running during non-peak hours.

### Solution

To charge the customers for their peak-hours, it is suggested that the SaaS provider charges different prices for different time segments when using usage-based pricing (Wu, 2010), as illustrated below. On the left side three customers are illustrated, each of them using the SaaS application at specific

times during the day. Since at specific times the SaaS application faces a high load on the hardware, the customers are charged according to the moment they have used the software.

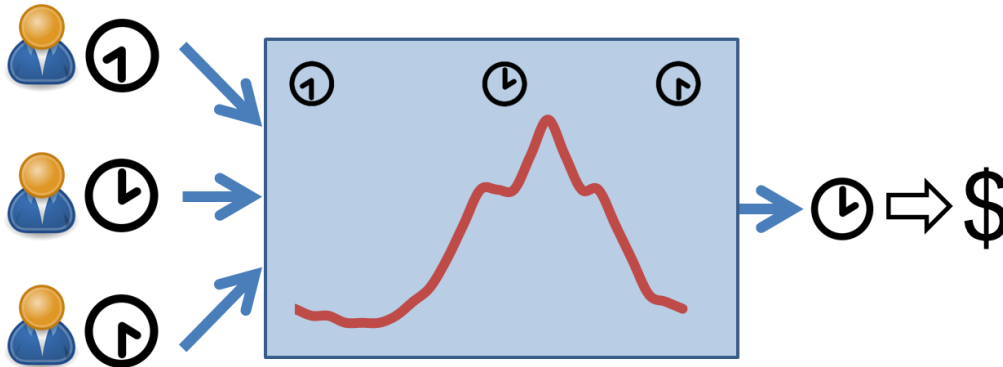


Figure 13 Illustration of peak load pricing

### Example

To apply the peak load charging pattern, a software vendor might want to analyze the usage statistics of the SaaS application. The charts below depicts the number of logins per hour on the ERPOnline application.

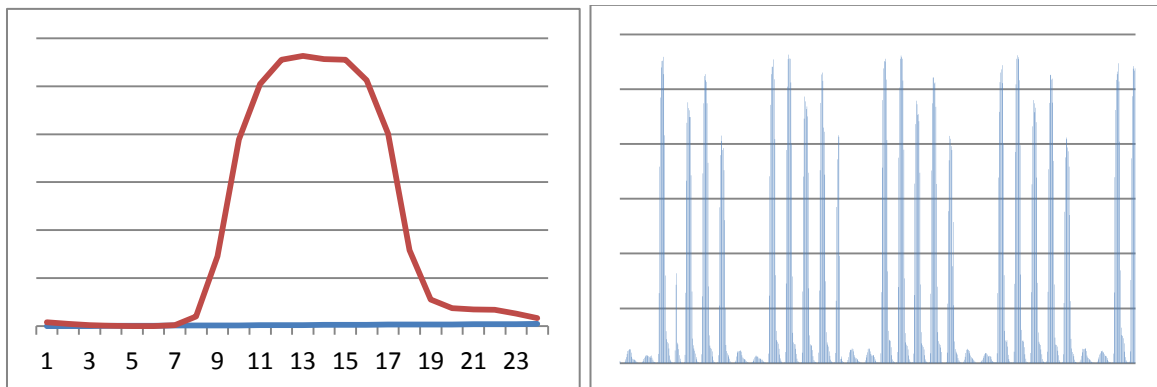


Figure 14 Graph of number of logins per hour, the left graph is during the day, the right graph during a whole month

When analyzing the left chart it can be seen that the usage is the highest between 12h and 15h. From 19h till 8h the usage is significantly lower compared to usage during working hours. From the right chart, that depicts usage during a whole month we see that usage is the highest at working days (Monday till Friday) and extensively lower during the week-ends. When analyzing the usage during working days a small drop can be noticed at Wednesdays and Fridays. The problem is that the needed capacity is equal to the required capacity during peak hours, while the servers are not (fully) utilized during non-peak hours.

Peak load charging provides a solution in here. When the usage of the SaaS application is more expensive during peak hours, the required capacity will be moved to another time of the day, because not all customers are willing to pay more for the usage at specific times.



The peaks in usage are expected to flatten, because it is expensive during peak hours the usage will increase during the non-peak hours. Hereby the available capacity is optimized, since the constant use of the SaaS application. In case peak load pricing does not lead to a decrease in usage, the extra income from higher pricing during peak hours can be used to invest in extra server capacity that is needed to provision the customers.

## 5.8 BUNDLE PRICING

### *Problem*

When a customer buys standard software the problem is that this software usually contains parts of functionality that a customer does not require, but the customers are forced to also buy these not required parts (Katzmarzik, 2011).

However, when it is offered as a whole package the customers does not have the choice to buy only the required parts, therefore the available budget should be spent on both the required and not required functionality (Katzmarzik, 2011).

A software product usually is built upon different modules, which all provide certain functionality. When the software vendor offers the software as a whole package at a single price, all customers have to pay the full price for using the software. It might be that a customer only requires the functionality from specific modules that are available in the whole package. Software companies that develop/offer a large software product usually split it up into several modules.

The problem might also be the other way around when the software vendor sells every module separately and therefore resulting in a lot of fragmentation. The problem is that is difficult for the administration when there are a lot of separately sold modules.

### *Solution*

With bundle pricing it is possible to split up the main software package into separate combinations of modules (bundles) and price them separately. The bundles are made for specific types of customers which are in need of certain combined functionality but do not need all functionality which is available in the main software package.

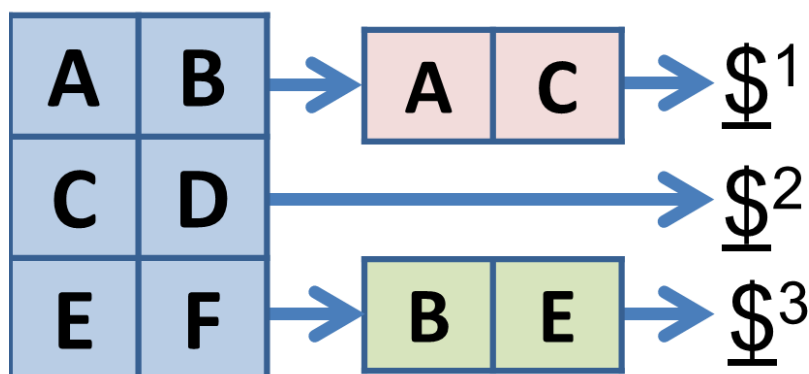


Figure 15 Illustration of bundle pricing

In the figure above, a standard software package is illustrated in blue. This standard software consists of six modules, named from A to F. Assuming that these modules also function as standalone

modules; they can be bundled in separate packages. Since these separate packages contain less functionality than the whole package they are cheaper. Because functionality differs between the bundles, also the price between the bundles can differ, as depicted by the dollar sign.

### *Example*

Bundling several modules should be done carefully to prevent a gap between the bundles. This problem is illustrated by the following example.

*“The small company bundle has a lot of functionality but is relatively cheap. For associations it would be nice to have a membership record functionality. This functionality not available in the ‘small company bundle’ but it only available in the much more expensive ‘big company bundle’. Another disadvantage is that the ‘big company bundle’ contains a lot of functionality that is not needed by the societies. To serve society customers it would be better to include the membership functionality in the ‘small company bundle’.”*

The fragment above is based on a discussion by employees of the finance and controlling department, indicating that bundling software should be done carefully.

## **5.9 DUAL PRICING**

### *Problem*

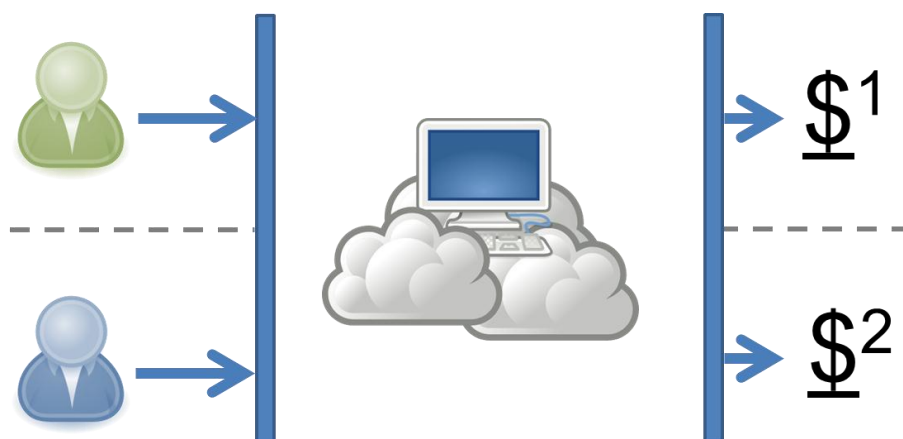
A software vendor might have different types of customers that have a different valuation of the software. In other words, the reservation prices among the customers differ.

When a software vendor offers a SaaS application which is suitable for many different markets, it might be that the organizations from one market segment have different reservation price, compared to organizations from another market segment. When the price of the software is lowered for all customers this would mean a decrease in revenue.

At ERPComp this situation occurred in the healthcare market where a competitor offers a similar product at a lower price than ERPComp.

### *Solution*

Applying different prices for different types of customers can be used to solve this problem. These different types of customers be bundled in groups of customers who are all operating in the same market. However, also other characteristics of a customer can be used for categorization in customer groups that have different prices for using the same software compared to other customer groups.



**Figure 16 Illustration of dual pricing**

This solution is depicted in the diagram above, two different customer types; making use of the same Software as a Service, pay a different price for using that same service.

The different types of customers are illustrated as two users coloured in blue and green. The difference between these customers is also illustrated by the dotted line, which is not depicted in the illustration of the SaaS application (‘computer-cloud’). Although they make use of the same SaaS application, they are charged differently, illustrated by \$<sup>1</sup> and \$<sup>2</sup>.

### Example

An example of a customer complaint found in an e-mail discussion between the CFO and a customer is depicted below.

*“We want a web portal for our customers, therefore we can choose between your software: Esite or another software package. The problem is that your price (€2.50 per contact) is too high, while we have about 15.000 customers. Many of those customers are once-only customers. We want to service these customers; this price of this product is too high since it won’t lead to direct sales or more profit.*

*At this moment, the pricing is such that I won’t buy the software, it is a missed opportunity if we have to develop a custom-made web portal which is cheaper, while your software is meant to be a web portal.”*

Esite is a new software product from ERPComp whereby it is possible as a customer to login and for example make a reservation for a car at car rental service. This request is automatically linked to the ERP software from ERPComp. Currently, ERPComp customers have to pay €2,50 per contact *they* have. The problem is that a business-to-consumer company usually has a lot one-time customers, who will occasionally use Esite. Therefore they don’t want to pay as much as a business-to-business company that usually has a fixed amount of customers that will often use the software.

This example illustrates that the current pricing pattern that ERPComp has applied for this software package does not fit for all customer types. In the example above, the complaining customer is a business-to-consumer company, often dealing with one-time customers who will eventually make use of the software only once and for this type of companies the software is too expensive at this moment.

## 5.10 CHAPTER OVERVIEW

The previous paragraphs gave an indication of the problems that a software vendor for business SaaS applications might face. For each of the problem a solution is presented based on a literature study. In the table below an overview is given of the solutions and the problem that the solution solves.

<b>Solution</b>	<b>Problem</b>
<b>Usage-dependent pricing</b>	Intensive customers are charged the same as less intensive customers
<b>Situation-based pricing</b>	Usage-dependent metrics are hard to calculate in advance
<b>Cost-based pricing</b>	Unequal division of costs
<b>Flat fee pricing</b>	Measuring usage is difficult
<b>Two-part tariff pricing</b>	Revenues are uncertain when usage based pricing is applied
<b>Peak load pricing</b>	Extra capacity is needed for peak hour usage
<b>Bundle pricing</b>	The main software package contains modules that are unneeded by some customers
<b>Dual pricing</b>	Different types of customers have different valuations for the same software

Table 12 Overview of the identified problems and the corresponding solutions

The next chapter evaluates the solutions as proposed in the previous sections.

## 6 EVALUATION

This chapter describes the evaluation of the pricing patterns. The goal of the evaluation is to find consequences of the pricing patterns described in the previous chapter.

The first part describes the application and usage of the pricing patterns within the case study company to find practical usage of the pricing patterns. The second part is the evaluation of the pricing patterns. This begins with gathering requirements for pricing patterns, followed by prioritizing these requirements. Next, the requirements are matched to the pricing patterns as identified in the previous chapter. The chapter concludes with an overview of all pricing patterns (pattern thumbnails), shortly describing the problem, solution and consequences.

### 6.1 USAGE OF THE PRICING PATTERNS

The usage of the pricing patterns at ERPComp is identified to find out how the suggested pricing patterns are used in a 'real world' situation.

Sales managers and financial managers are interviewed to find usages of the pricing patterns. Also, a document study of the price lists and price guidelines is conducted to gather information about the pricing patterns that are currently used for ERPOnline. With the gathered information, the pricing patterns from the previous chapters are evaluated.

The following paragraphs describe the usage of each pattern at ERPComp in detail as the result of the interviews and document study.

#### *Usage-dependent pricing*

Usage-dependent pricing is not a common used pricing pattern within ERPComp. It is used as a secondary pattern in addition to pricing based on named users.

For the customers of ERPOnline, 1Gbyte of data storage is included in the price. If a customer exceeds this limit, an additional fee has to be paid for the extra storage. This extra storage is available in blocks of 2Gbyte. Currently, about 700 of the 13.000 environments (including test environments) are currently using more than 1Gbyte of data storage.

#### *Situation-based pricing*

Situation-based pricing is one of the main pricing pattern at ERPComp. As described in the previous chapter there are several metrics available for situation-based pricing. The metric used for this pricing pattern at ERPComp is 'named user'. The number of named users registered at ERPOnline multiplied with the monthly fee per user define the total monthly fee that has to be paid by the customer.

Also at a lower level situation-based pricing is somewhat applied. In the internal pricing guideline for sales managers, the following remark was found at the bottom of the pricelist: *"In case of a considerable difference between office employees and other employees, different discount of an offer may be applied, in consultation with a sales manager."* This remark was the reason for further investigation, resulting in an interview with the responsible sales manager.

When asking the sales manager it turned out that discount is given to certain types of companies. These are companies with a relatively small amount of office workers. This is usually the case for companies with more than 25 employees or employment agencies. Accordingly, ERPComp offers

discount to companies that have a lot of ‘other employees’ besides office employees. Office employees include sales, management, human resource management, etcetera. The ‘other employees’ do not need all the functionality offered within the main product. Therefore, certain functionality is blocked and a discount is offered to those customers.

This happens in at least 50% of the implementations, according to the sales manager. In fact this is usage dependent pricing because a customer has to pay less for employees that do not often make use of the software. A weakness of this way of pricing is that the percentage of discount that is given to the customer is the decision of the sales manager, there are no guidelines.

### *Cost-based pricing*

Cost-based pricing is not one of the mainly used pricing patterns. ERPComp does not make comprehensive calculations for defining the price for the software, because the margins are currently high enough to cover the costs. The major impact on the costs are the development costs, but they are seen as sunk costs by ERPComp.

Instead of looking at the real costs, ERPComp mainly focusses on the following aspects when defining the price:

- Price of competing products
- Customer value
- Development costs

ERPComp focusses on a companywide profit, instead of calculating prices based on costs, according to the CFO. Therefore, it is possible that they make losses on some customers.

### *Flat fee pricing*

Flat fee pricing is not applied at ERPComp. The price of ERPOnline is based on the number of users and the needed storage. Customers have to pay extra if they are in need of more data storage.

On the other hand, customers have unlimited access to the functionality of the SaaS application (when it is part of their chosen bundle). Since access is given per named user, it is merely a situation based pricing that is applied than a flat fee pricing pattern. Consequently, flat fee pricing is applied in terms of unlimited use of functionality but not for the used data storage.

### *Two-part tariff pricing*

It depends on the type of customer whether two-part tariff pricing is applied. Customers that are satisfied with the included 1Gbyte data storage have no additional costs besides the monthly fixed price. For those customers the situation based pricing pattern is applied.

On the other hand, there are also customers that are in need of extra data storage on top of the standard provisioned 1Gbyte. Those customers are additionally charged by ERPComp for using the amount of data storage that exceeds the 1 Gbyte. For this type of customer the two-part tariff pricing pattern is applied.

Because it depends on the type of customer (in need of extra data storage or not), whether two-part tariff pricing is applied, this pattern is not assumed as one of the main pricing patterns at ERPComp, because it does not apply for all customers. Therefore two-part tariff pricing is assessed as secondary pricing pattern at ERPComp.

### *Peak load pricing*

Peak load pricing is not applied at ERPComp. They do face the problem of peak load usage, especially during the end of the month when pay rolling occurs since it is usual for companies to pay their employees at the end of month. To solve the problem of peak load usage at certain time in a month, they make use of a batch server. Tasks that result in a high server load are postponed to be executed during the night by the batch server.

Currently all customers accept this solutions since no complaints are known from customers that do not agree with this solution. However, this might change in the future and therefore ERPComp has to consider alternative solutions for the peak load problem, whereby peak load pricing might be the right solution.

Instead of solving the problem of peak hour usage with a pricing pattern, ERPComp changed functionality by using a batch server to avoid peak loads.

### *Bundle pricing*

Bundling modules of the whole software package is one of the main pricing patterns used by ERPComp. The main product from ERPComp consists of several modules such as ‘human resource management’, ‘payrolling’, ‘workflow management’ and ‘customer relationship management’.

Since not all customers need all modules for their business, ERPComp made a pre-selection of several related modules that are presented as a bundle. Each of these bundles has a focus on a specific market. Consequently, there are bundles for education, health care and accountancy.

### *Dual pricing*

There is no official form of dual pricing at ERPComp, there are different customer types but they have chosen to differentiate them by mean of a different package as described in the bundling paragraph. ERPComp indicates that there are different customer types, but instead of offering them different prices for the same product, they pre-configured the standard product to serve specific customer and consequently have different prices for each of the (pre-configured) packages.

In the price list, there are two similar bundles: ‘HRM/Payroll’ and ‘HRM/Payroll for healthcare and education’. There is not a significant difference in price between these two bundles. Nevertheless, for the latter bundle there are more software extensions available in the bundle. Therefore, the bundle for healthcare and education does not differ in functionality except for the extensions that are included in the package.

During an interview with the Chief Financial Officer, he mentioned that ERPComp chose to offer a bundle with more functionality at a lower price to the health healthcare and education markets to compete and gain market share. But also to keep up with the price of the competitors which are also operating in those markets.

### *Overview*

Based on the findings as described in the previous paragraphs, a table is made to give an overview of the usage of the pricing patterns at ERPComp. For each pricing pattern, there are three options to indicate the degree of application of the pricing patterns. The following definitions of the degrees are used:

- Primary: The evaluated pricing pattern is (one of) the mainly used pricing patterns.
- Secondary: A used pricing pattern but not one of the main patterns, it is used as an addition to the primary pricing pattern.
- Not applied: The pricing pattern is not applied.

The results of the evaluation regarding the usage of the pricing patterns at ERPComp are summarized in Table 13.

	Primary	Secondary	Not applied
Usage-dependent pricing		X	
Situation-based pricing	X		
Cost-based pricing		X	
Flat fee pricing		X	
Two-part tariff pricing		X	
Peak load pricing			X
Bundle pricing	X		
Dual pricing		X	

**Table 13 Application matrix of the pricing patterns at ERPComp**

From the table it can be derived that ‘situation-based pricing’ and ‘bundle pricing’ are the two primary pricing patterns. Five of the remaining pricing patterns are used as secondary pricing pattern at ERPComp, only ‘peak load pricing’ is not applied.

The evaluation of the pricing patterns showed that at ERPComp there are multiple pricing patterns are used on top of each other for pricing ERPOnline. For example it is possible that customer choses for a specific bundle with certain modules in it (bundle pricing), for every named user a monthly fee has to be paid (context pricing), if this customer is in the education sector, he receives a discount that is not available for other customers (dual pricing). If this customer needs more data storage than the standard included 1 Gbyte they have to pay an additional fee per extra Gbyte (usage-dependent pricing). Because the fee that a customer has to pay consists out of two parts: the monthly fee and (eventually) for the data storage if it exceeds 1 Gbyte, also two-part tariff pricing is applied at ERPComp for the same product. ERPComp applies five different pricing patterns for one product.

## 6.2 PATTERN EVALUATION

This section of the evaluation of the pricing patterns begins with gathering requirements for pricing patterns, followed by prioritizing these requirements. Next, the requirements are matched to the pricing patterns which are identified in the previous chapter. This chapter concludes with an overview of all pricing patterns (pattern thumbnails), shortly describing the problem, solution and consequences.

### *Gathering requirements*

In order to validate the pricing patterns on goodness and correctness, an expert validation with multiple employees of the case study company is carried out.



## Approach

For this expert evaluation the following approach is used:

1. The requirements of a good pricing model are gathered.
2. The requirements are prioritized.
3. The requirements are matched to the pricing patterns.

For gathering the requirements, the following question is asked in an e-mail to the sales managers: “*What are important criteria for a pricing model for SaaS?*”. Note that the term ‘pricing model’ is used instead of ‘pricing pattern’ since this is much more common and this term is used within the case study company. A template of the e-mail sent to the financial and sales managers is included in the appendix. One of the approached sales managers preferred a face to face conversation about the criteria for a good pricing model, instead of answering by e-mail.

## Results

Five financial and sales managers from the case study company are e-mailed, four of them responded. The results are summarized in the table below. Since the respondents used different terms for the same definition, similar requirements are merged. The characteristics are listed below, in the columns on the right it is depicted which expert mentioned which characteristic as important.

	Expert 1	Expert 2	Expert 3	Expert 4
transparent		X	X	
relation between usage and price		X		
easy for administration	X			
predictable			X	
recurring costs		X		
easy to calculate			X	
usable to serve multiple markets				X
clear insight for the customer	X		X	X

**Table 14** Overview of requirements mentioned by the respondents

## Problems

Two of the answers are omitted because they are not useful for pricing models: ‘no space for discount’ and ‘consistent with market demand’, because these are not requirements for a pricing model as used in this research. This results that eight characteristics are used for the prioritizing evaluation.

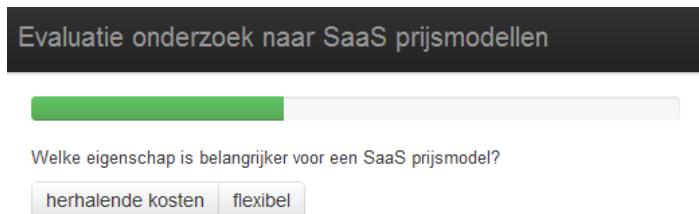
### *Prioritizing the requirements*

It is expected that not all requirements are of equal importance, therefore the requirements should be prioritized. This is done by using the pair wise comparison method, which already has proved to be a useful in an evaluation of methods for prioritizing software requirements (Karlsson, 1998).

#### **Approach**

No suitable free software was found to carry out a pair wise comparison with multiple stakeholders. Therefore an online tool is developed with the use of PHP and MySQL. PHP is the programming language and MySQL is database software used to retrieve the requirements and store the results.

One database table contains all identified characteristics. When a respondent started with the survey, all possible combinations are calculated based on this table. These combinations are saved in a separate table. The application randomly selects one of these combinations and shows it to the respondent, accompanied with the question ‘which characteristic is more important for a SaaS pricing model?’ followed by two characteristics where the respondents should choose the most important one as can be seen in Figure 17.



**Figure 17 Screenshot of the evaluation tool for prioritizing the characteristics (in Dutch)**

The statements are presented till the respondent answered all combinations. A progress bar gives an indication of the progress to the respondent. Clicking one of the characteristics saves the answer in the results table and a new pair wise comparison is randomly selected and presented to the respondent. Once the respondent answered all questions, a ‘thank you’ page is showed with the possibility to leave an e-mail address to receive the results of this research. This survey is sent to eight people within ERPComp, including the CEO, CFO, sales managers and account managers, seven of them completed the survey.

#### **Results**

To analyse the results of the pair wise comparisons the analytical hierarchical priority (AHP) method is used. The AHP-method allows to prioritize pair wise comparisons. Usually, the AHP-method requires that the respondent give a weight to each answer. Since this weight was not asked in the survey, all weights are given 5 points on a scale of 1 to 9, to equally asses the pair wise comparisons. The results of the prioritization using the AHP-method are depicted in the table below. The normalized principal Eigenvector is a percentage indicating the priority of the corresponding characteristic. The higher the percentage, the more important it is.

Matrix	Transparent	Relation between usage and price	Easy for administration	Predictable	Repeating costs	Easy to calculate	Usable for different markets	Clear insight for customer	normalized principal Eigenvector
Transparent	1	2	5	5	5	3 1/6	2	4/5	24,0%
Relation between usage and price	1/2	1	2	2	3 1/6	3 1/6	2	4/5	16,2%
Easy for administration	1/5	1/2	1	1 1/4	1 1/4	4/5	1/3	1/3	5,9%
Predictable	1/5	1/2	4/5	1	1 1/4	1/5	4/5	1/3	6,1%
Repeating costs	1/5	1/3	4/5	4/5	1	1/3	1/2	1/3	5,1%
Easy to calculate	1/3	1/3	1 1/4	5	3 1/6	1	4/5	1/3	8,7%
Usable for different markets	1/2	1/2	3 1/6	1 1/4	2	1 1/4	1	4/5	12,1%
Clear insight for customer	1 1/4	1 1/4	3 1/6	3 1/6	3 1/6	3 1/6	1 1/4	1	21,9%

**Table 15 Matrix with the normalized principal Eigenvector as part of the AHP-method**

When ordering the normalized principal Eigenvector percentages, the top 8 of most important characteristics is as follows:

Position	Characteristic
1.	Transparent
2.	Clear insight for the customer
3.	Relation between usage and price
4.	Usable for different markets
5.	Easy to calculate
6.	Predictable
7.	Easy for administration
8.	Repeating costs

**Table 16 Overview top 8 characteristics**

The characteristics ‘transparent’ and ‘clear insight for the customer’ are significantly more important because of their high Eigenvector percentages (24.0% and 21.9%).

### Problems

A problem occurred when one of the respondents could not retrieve the webpage. This was possibly due some downtime of the web server, or a lacking internet connection of the respondent. However, he tried again the next day, which worked fine. The respondents mentioned no other problems.

### Matching the requirements to the pricing patterns

Since the pricing patterns and pricing characteristics are developed and identified independently, they are not yet related to each other. Therefore, the requirements should be matched to the identified pricing patterns.

#### Approach

A similar tool as used for prioritizing the requirements is used to for matching the requirements to the pricing patterns. The pricing patterns, including a short description are stored in a table.

There are eight characteristics identified from gathering the requirements and there are eight pricing patterns. Since each of the characteristics should be matched with each pricing pattern, it results that there are 64 statements that should be assessed by the respondents. A five point Likert scale is used to assess statements, with the following format:

1. Strongly disagree
2. Disagree
3. Neither agree nor disagree
4. Agree
5. Strongly agree

For matching the requirements to the pricing patterns, 12 account managers and 2 sales managers were asked to participate in this research by sending them an e-mail. The e-mail template is included in the appendix. For each possible match between a pricing pattern and a requirement a question is asked to the participant using the format: *name of pricing pattern is name of characteristic*. An example is given in Figure 18.

Beoordeel de volgende stelling:

peak load prijzen is eenvoudig voor de administratie

1 2 3 4 5

\* 1 = oneens, 5 = eens

peak load prijzen: prijs hangt af van het moment van gebruik, op drukke momenten is de prijs hoger

Figure 18 Screenshot of the evaluation tool used to match the requirements to the pricing patterns

#### Results

In total, 8 respondents from ERPComp assessed the statements. Not all respondents made it through all 64 statements, therefore some statements are answered by a lower number of respondents than other statements. Since the questions are randomly presented to each respondent it differs which statements are assessed less often.

One respondent answered all statements but afterwards he mailed that he speedily answered all questions and in fact needed more time to accurately assess each statement, resulting in unreliable answers. To omit that this affects the reliability of the results, his answers are deleted and not part of the analysis.

As multiple respondents assessed the statements there was a difference in the number of points given to a statement. The Fleiss' kappa is used as a statistical measurement to assess the reliability of the agreement between the respondents. With the Fleiss' kappa it is possible to measure the scale agreement among multiple raters (Fleiss, 1971).

The table below depicts the top 10 statements with the highest strength of agreement. The full results of the assessment are included in the Appendix.

		1	2	3	4	5	$P_i$
<b>Id</b>	<b>Statement</b>						Fleiss kappa
24	cost based pricing results in clear insight for the customer	1	5	1	0	0	0,476
32	flat fee pricing results in clear insight for the customer	0	0	1	1	5	0,476
48	peak load pricing results in clear insight for the customer	5	1	1	0	0	0,476
18	cost based pricing has a relation between usage and price	4	0	3	0	0	0,429
63	dual pricing is usable to serve multiple markets	0	0	0	3	4	0,429
17	cost based pricing is transparent	3	0	3	0	0	0,4
27	flat fee pricing is easy for administration	0	0	1	1	4	0,4
28	flat fee pricing is predictable	0	0	1	1	4	0,4
43	peak load pricing is easy for administration	4	1	1	0	0	0,4
25	flat fee pricing is transparent	0	0	2	1	4	0,333

Table 17 Top 10 results

For the interpretation of the Fleiss' kappa results the interpretation table by Landis and Koch (1977) is used which is depicted below. Although these divisions are subjective, they do provide valuable benchmarks (Landis & Koch, 1977).

<b>Kappa Statistic</b>	<b>Strength of agreement</b>
< 0.00	Poor
0.00-0.20	Slight
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Substantial
0.81-1.00	Almost perfect

Table 18 Interpretation of the Kappa statistic for the strength of agreement (Landis & Koch, 1977)

In this research, a Kappa statistic of minimal 0.21, corresponding with at least a fair agreement, is considered as enough agreement between the respondents. Because all statements are formulated 'positive', also statements with a low average score are useful because when they are formulated 'negative' they do apply to the pricing pattern in the statement.

Hence the statement 'peak load pricing is clear for the customer' has an average score of 1.4 on the five point scale, with a moderate strength of agreement ( $K = 0.48$ ). This implies that the respondents would have agreed with the statement 'peak load pricing is *not* clear for the customer'. Therefore, in case of a low average score, the statements are rephrased to a negative statement so they can be matched to the pricing pattern.

When all statements with a low strength of agreement are ignored, the characteristics can be matched to the pricing patterns as depicted in the table below. Following the criteria of at least 0.21 for the

Kappa statistic and rephrase statements with a low score, it results that the statements depicted in the table below can be matched to the pricing patterns.

<p><b>usage dependent pricing:</b></p> <ul style="list-style-type: none"> <li>- is not transparent</li> <li>- is usable to serve multiple markets</li> </ul>	<p><b>two part tariff pricing:</b></p> <ul style="list-style-type: none"> <li>- is not transparent</li> <li>- is not easy for administration</li> <li>- is usable to serve multiple markets</li> </ul>
<p><b>situation-based pricing:</b></p> <ul style="list-style-type: none"> <li>- has recurring costs</li> <li>- is usable to serve multiple markets</li> </ul>	<p><b>peak load pricing:</b></p> <ul style="list-style-type: none"> <li>- is not transparent</li> <li>- is not easy for administration</li> <li>- is not predictable</li> <li>- is not easy to calculate</li> <li>- has no clear insight for the customer</li> </ul>
<p><b>cost based pricing:</b></p> <ul style="list-style-type: none"> <li>- is not transparent</li> <li>- has no relation between usage and price</li> <li>- is not easy for administration</li> <li>- is not usable to serve multiple markets</li> <li>- has no clear insight for the customer</li> </ul>	<p><b>bundle pricing:</b></p> <ul style="list-style-type: none"> <li>- is easy for administration</li> </ul>
<p><b>flat fee pricing:</b></p> <ul style="list-style-type: none"> <li>- is transparent</li> <li>- has no a relation between usage and price</li> <li>- is easy for administration</li> <li>- is predictable</li> <li>- is easy to calculate</li> <li>- is usable to serve multiple markets</li> <li>- has clear insight for the customer</li> </ul>	<p><b>dual pricing:</b></p> <ul style="list-style-type: none"> <li>- is not predictable</li> <li>- has recurring costs</li> <li>- is usable to serve multiple markets</li> </ul>

**Table 19 Overview of requirements that are matched to the pricing patterns**

### Problems

During the evaluation one of the respondents mentioned that the matching of the characteristics sometimes resulted in a statement that was not right. The respondent had a problem with the statement “situation-based pricing has repeating costs”. He commented that “the phrasing of the question is not right or at least it is unclear for me” and therefore stopped with evaluation after two questions because he wanted to keep the results of the evaluation accurate.

This could have been solved by reviewing all statements at first; however this might have resulted in subjective deletion or acceptance of statements by the researcher. A better solution might have been to give the respondents the option ‘not applicable’ next to the choices 1 to 5.

Some of the respondents seem to have stopped with the evaluation half way. Since they did not mention why they stopped (and the respondent could not be retrieved because it was anonymous) and the statements were presented random for all users, the results of the unfinished are included in the results.

## 6.3 CONSEQUENCES

In this section the consequences of using the pricing patterns is elaborated. This includes results from the literature study on the consequences as well as the results from matching the requirements to the pricing patterns.

### *Usage-dependent pricing*

This pricing pattern allows the software vendor to set the prices in a way that always cover the costs of ownership. The software product becomes also accessible for customers that will use the software less often and therefore might not want to pay the initial market price. There will be a decrease of the consumer surplus.

Another benefit of a usage-dependent pricing pattern is that the company can serve customers that are operating in different markets (result from evaluation with experts). The price of the software becomes variable and is more dedicated to the customer's usage. The strength of agreement among the respondents about this statement was not high enough to accept that statement, but it was not rejected either.

Usage-dependent pricing can be applied to decrease the consumer surplus by pricing the product related to the usage by the buyer. It is expected that the consumer surplus will decrease in favour to the software vendor. By applying a usage-dependent pricing pattern, the software vendor will gather a larger part of the consumer surplus, resulting in extra turnover and eventually in increasing profit. (Mankiw & Taylor, 2006).

A liability of this pricing pattern is that the risk shifts to the software vendor, since it is unknown in advance what the exact usage of the software will be. Consequently, this implies uncertainty about the earnings and the vendor is faced with unpredictable costs. Since usage can vary by time, the costs will also vary. It is also unknown to the customer what the costs are in advance, resulting in a non-transparent pattern according to the respondents.

Furthermore, the metrics of measuring the usage can be tricky to build. This depends on the chosen metric, some as used bandwidth can be easily calculated while measuring usage on a functionality level might more complicated. The level of usage of the software will be by the customers is unknown in advance.

### *Situation-based pricing*

Since the price is based on the customer's situation, this pricing pattern is able to serve multiple markets (according to the respondents). With this pattern is possible to charge recurring costs.

The customers also have to pay for their users/employees that do not often use the SaaS application because the same price should be paid per user, regardless usage. Since the price is determined beforehand and the customer's situation will not change often, both the software vendor and the customer now in advance what the expenses or income will be.

### *Cost-based pricing*

The costs are fairly distributed among the customers with cost-based pricing. If there is a complex function built (a lot of development hours results in higher costs) than these costs are not distributed among all customers, but only among the customers who actually use that software.

A disadvantage is that if nobody uses the software, the cost per month will be very high for those who have used the software. Other notable liabilities are that is not easy for administration and by the lacking transparency it is no clear insight for the customers. It also expected that it is hard to include all costs, because it is hard to identify hidden costs.

### *Flat fee pricing*

When flat fee pricing is applied, the customer knows the costs in advance. Consequently, the income for the vendor are predictable in case of flat fee pricing. Other advantages of flat fee pricing are the transparency and the relatively easy implementation (“Internet Economics,” 1998). Because a fixed fee has to be paid, no extensive calculations are needed to define the price and therefore the costs are also predictable for both the software vendor as the customer.

Customers are usually willing to pay more for the option of unlimited usage of a SaaS application (Sundararajan, 2004).

A difficulty with flat fee pricing is the missing relation between usage and price. Another liability is that the fixed price regardless usage might be too high to some customers resulting in less customers for the SaaS application (Kittlaus & Clough, 2009).

It might be that the software vendor makes losses on certain customers, while making high profits on other customers. Therefore, overall profit is important with flat fee pricing.

### *Two-part tariff pricing*

An advantage compared to a usage dependent model is that in case of two part tariff pricing, the software vendor is guaranteed a minimum income because of the recurring subscription fee. But the software vendor is also faced with a higher income if the usage is higher. The risks for the software vendor are covered.

Two-part tariff pricing cannot be seen as transparent based on the evaluation results. Consequently, it might also have difficulties for the administration. Since the final price is based on two separate prices the pricing might become more difficult to understand by the customer and more complex for administration by the software vendor. In a research to which pricing schemes is the best for a monopolist providing information services, it resulted that either flat fee pricing or the two-part tariff is the optimal pricing scheme (Wu, 2010).

### *Peak load pricing*

Customers that do not care when they use the SaaS application benefit from peak load pricing, because they can use it when the price is the lowest. Other customer will probably want to know the costs in advance, while the time of peak usage is not always predictable and at the same time.

Many liabilities of peak load pricing are identified during the evaluation: it is hard to calculate and has no clear insight for the customer. Furthermore it lacks transparency and predictability.

### *Bundle pricing*

From the evaluation with experts no explicit statement, apart from ‘easy for administration’, are accepted because of low strengths of agreement and the average scores that are assessed by the respondents.

In the case that first the software consisted of separate modules which are now bundled, than it won’t be possible anymore to see the usage of those separate modules based on the sent invoices. Therefore if the vendor wants to measure usage, metrics should be built into the software. This problem occurred at ERPComp when someone from the product development department wanted to know how many



customers a certain module from a bundle are using. Because that module is part of bundle and is not sold separately it is difficult to see how often that is used.

By splitting up the large software package into separate modules with specific functionality the required effort might lead to increasing costs and therefore also results in higher prices per piece of functionality (Katzmarzik, 2011).

An advantage is that by offering separate bundles, the customizability and flexibility increase. Therefore also the sales volume will increase because it becomes interesting for new customers that otherwise would never have bought the complete software package (Katzmarzik, 2011). By offering more variations of a product (for example by configuration), the product becomes more attractive to more customers because since the variations it is more likely it will suit their needs.

Compared to a situation where the customer can create his own bundles by choosing from all available modules, the administration is easier with bundle pricing.

### *Dual pricing*

With dual pricing the distinction between different customer types might not be exclusive. This is similar to bundle pricing, where the software vendor should be aware that all different types of customers can be served.

When this is done carefully dual pricing is especially usable to serve multiple markets, according to the evaluation results. The predictability is less according to the evaluation results.

## **6.4 APPROPRIATE PRICING PATTERNS**

Now all that all consequences of the pricing patterns are gathered, it is possible to select the most appropriate pricing patterns.

Points are awarded to assess whether a pricing pattern is appropriate. The number of points awarded is the normalized principal Eigenvector percentage resulted from analysing the pair wise comparisons results. By using the percentages as points to award the pricing patterns, also the weight of the characteristics is taken into account.

<b>Position</b>	<b>Characteristic</b>	<b>Points</b>
<b>1.</b>	Transparent	24,0
<b>2.</b>	Clear insight for the customer	21,9
<b>3.</b>	Relation between usage and price	16,2
<b>4.</b>	Usable for different markets	12,1
<b>5.</b>	Easy to calculate	8,7
<b>6.</b>	Predictable	6,1
<b>7.</b>	Easy for administration	5,9
<b>8.</b>	Repeating costs	5,1

**Table 20 Number of points to award**

When a pricing pattern is assessed to be transparent, 24 points are awarded, if the pricing pattern has repeating costs, 1 point is awarded. When the negative statement is accepted the points are subtracted. Every pricing pattern begins with zero points. The results of awarding the points is summarized in Table 21, a complete overview is included in the appendix.

<b>Position</b>	<b>Pricing pattern</b>	<b>Points</b>
1.	flat fee pricing	62,5
2.	situation based pricing	17,2
3.	dual pricing	11,1
4.	bundle pricing	-5,9
5.	usage dependent pricing	-11,9
6.	two part tariff pricing	-17,8
7.	peak load pricing	-66,6
8.	cost based pricing	-80,1

**Table 21 Points awarded to pricing patterns**

In the table above the pricing patterns are prioritized based on the awarded points. According to this study flat fee pricing is by far the most appropriate pricing pattern for business SaaS applications. At some distance, situation based pricing and the dual pricing pattern complete the three most appropriate pricing patterns for business SaaS applications. Cost based pricing and peak load pricing are the pricing patterns with the least awarded points.

## 6.5 CHAPTER OVERVIEW

This table gives an overview of all identified pricing patterns, shortly describing the problem, solutions and consequences. The number left to the name of each pricing pattern is the result of awarding points to assess what the most appropriate pricing pattern is for business SaaS applications.

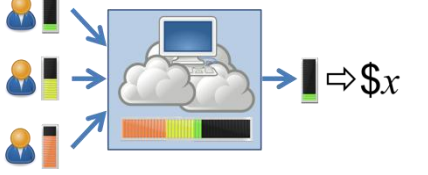
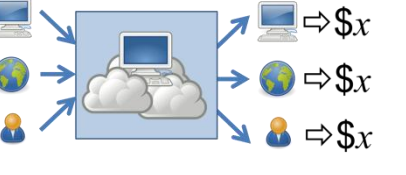


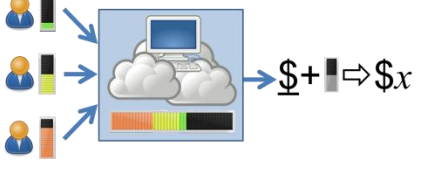
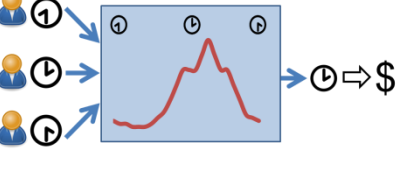
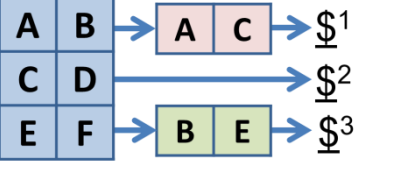

<p><b>⑤ Usage-dependent pricing</b></p> 	<p><b>② Situation-based pricing</b></p> 	<p><b>③ Cost-based pricing</b></p> 	<p><b>① Flat fee pricing</b></p> 
<p>P: intensive customers are charged same as less intensive customers S: the more a user uses, the more he pays. C: not transparent</p>	<p>P: usage metrics are difficult, hard to calculate in advance S: use metrics not related to usage, but to the situation C: serve multiple markets</p>	<p>P: unequal division of costs S: pricing based on the costs C: not transparent, no price-usage relation, no clear insight for customer</p>	<p>P: difficult to measure usage, need for simplicity S: fixed price regardless usage C: transparent, predictable, easy for administration</p>
<p><b>⑥ Two-part tariff pricing</b></p> 	<p><b>⑦ Peak load pricing</b></p> 	<p><b>④ Bundle pricing</b></p> 	<p><b>③ Dual pricing</b></p> 
<p>P: revenues uncertain when usage based charging is applied S: fixed fee plus additional usage based fee C: not transparent, not easy for administration</p>	<p>P: extra capacity needed only for peak hour usage S: charge different prices for different times C: not transparent, unpredictable for customers</p>	<p>P: software contains modules not needed by customer S: split up in bundles: combinations of modules C: easy for administration</p>	<p>P: customers have different valuations for same software S: apply different prices for different kind of customers C: usable to serve multiple markets</p>

Table 22 Overview of all patterns, P: Problem, S: description of the solution, C: consequences, ⑥ position in ranking of appropriate patterns

## **7 DISCUSSION**

The research into the costs of provisioning SaaS shows that there are recurring costs for the software vendor. Every month or every year the software vendor should pay the infrastructure provider. Assuming that a software vendor wants to make profit by developing and selling software, a single purchase of the software by the vendor's customers is not suitable for SaaS.

Single purchase is often used for on-premises installed software usually including unlimited usage for unlimited time. This has the consequence that only bug fixes and small updates are for free, but an additional payment has to be paid for large updates or new versions of the software product. Because a software vendor that develops software that should be installed on-premises does not have recurring costs, the pricing is less complex because only the initial development costs should be covered.

Making the choice between charging users based on their usage or a flat fee model is in fact making a choice about risk spreading. If a company chooses to charge the customers based on the number of users, the risk of heavy use of the software, and the corresponding higher costs, is for the SaaS vendor. With a fixed contract for several years, it can cause financial problems for the vendor, because the price cannot be changed if it does not appear to be cost-effective.

On the other hand, if the user is charged based on the usage of the software, then the financial risk of high usage of the software is on the customer's side, because the customer must pay more because of the high usage level. In case of a usage-dependent price model the vendor can always be cost effective by pricing the product at a higher price than the costs.

Instead of solving a problem with applying a certain pricing pattern, this study showed that it is also possible to solve a problem technically. This was found with peak load pricing where ERPComp has chosen to solve the peak load usage problem by moving intensive tasks to a batch server that executes those tasks at night.

At ERPComp the SaaS application is priced according to named user (situation-based pricing) but when the included 1Gbyte storage is exceeded, the customers have to pay according used data storage (usage-dependent pricing). This shows that it is possible to combine or stack different pricing patterns, but that there are only one or two primary patterns.

After prioritizing the requirements and matching them on the pricing patterns the main findings of the study were found. The most appropriate pricing patterns for business SaaS applications are flat fee pricing, situation based pricing and dual pricing.

### *Limitations*

In this study the customers are not asked but since the sales managers and account managers have close contact with customers it is expected that they have accounted good enough for the customer's wishes when evaluating the pricing patterns.

Another limitation is that this study is carried out at one case study company. Therefore only problems that the case study company faced are identified, while other companies might face different or additional problems. Therefore the identified pricing patterns might not be complete.

A limitation of this research is that it is validated at only one company. Although it weakens the external validity, this method was chosen because it was possible to perform a deep insight into important documents such as agreements between ERPComp and suppliers and other internal documents. This won't have been possible if the study was conducted at multiple companies, because it is expected that if the results are shared among the competitors the companies' willingness to participate in this research would have been very low. Therefore, it is chosen to perform the research at one company where it was possible to extensively interview experts and gain insight by document studies.

### *Further research*

This study might not provide solutions for specific problems that other software face. Therefore similar research might be performed at other software vendors to identify possible other problems that those companies face with pricing SaaS. This research could be extended by interviewing people from other software vendors to identify their SaaS pricing problems. Next, it should be researched whether one of the pricing patterns as proposed in this research can solve those problems or additional pricing patterns are needed.

Further research could be studying software vendors that offer a SaaS application and to verify how they have priced their software. This information can be used how often certain pricing patterns are used in the field and possibly to identify whether pricing patterns for SaaS are used that are not identified in this research.

Also, research could be done by studying other types of (SaaS) software. This research focuses on business SaaS applications, which has the typical characteristic that it must be very configurable to meet the needs of the customers. For consumer software, it is expected that it should be less configurable, but still some configuration is needed. Therefore further research could focus on other types than business SaaS applications whether the pricing patterns developed in this study are also applicable or those types have different problems and need other pricing patterns.

### *Outlook of the usage*

Currently the price list and price strategy is under revision. Based on the results from this research some changes are made. One of the changes is that the separate 'HRM/Payroll' bundles for education and healthcare institutions are going to be merged in one bundle that has exactly the same functionality for every customer. But since competitors in the healthcare and education markets offer their product at a lower price, ERPComp is planning to apply dual pricing for this bundle. These types of customers have to pay a lower price for the same bundle.

For pricing the products in the upcoming years and the composition of the pricing list for the next year, the CFO will make use of the findings in this research. At the evaluation section there are some suggestions made for ERPComp that they can adopt. The CFO acknowledged that there are some limitations and weaknesses in the current price list and will consider the proposed solutions in composing new pricing lists and guidelines.

Similarly, other companies that currently offer a business SaaS application can benefit from the results of this study by evaluating their current pricing pattern(s). Also companies that will launch a business SaaS application in the near future can anticipate on the problems as identified in this study by choosing a pricing pattern that is suitable for them.

## **8 CONCLUSION**

The main purpose of this study was to find appropriate pricing patterns for business SaaS applications, from the point of view of a software vendor. In order to find those pricing patterns the following research questions were formulated:

*What are appropriate pricing patterns for business SaaS applications, from the point of view of a software vendor?*

1. *What is a business SaaS application?*
2. *What are the implications of offering business SaaS applications?*
3. *How can business SaaS applications be priced?*

Based on the results from the literature on business software and SaaS it can be concluded that a business SaaS applications is administrative software for organizations to support their processes and workflows, with configurability options to adapt the software to the customer's specific needs, provisioned as Software as a Service.

When offering a business SaaS application, the software vendor requires a multi-tenant architecture in order to support configurability in the SaaS application. Furthermore, the costs elements differ from on-premises installed software by consisting out of software licences and hosting costs. Furthermore, metrics are required in case a software vendor want to price the application according the customer's situation or usage.

Business SaaS applications can be priced using one of the following pricing patterns: usage-dependent, situation based, cost-based, flat fee, two-part tariff, peak load, bundling or dual pricing. This study shows that there are many different ways of pricing a business SaaS application. Problems regarding pricing were identified at the case study company and solutions were found for these problems. An evaluation with experts from the case study company resulted in an overview of requirements for pricing models, which were matched to the pricing patterns using a survey and statistical analysis.

Based on the results of the prioritization and matching the requirements to the pricing patterns, the most appropriate pricing patterns for business SaaS applications could be selected. The three most appropriate pricing patterns are:

- flat fee pricing
- situation based pricing
- dual pricing

This resulted in the main deliverable of this study: an overview of all pricing patterns, shortly describing the problem, the solution, an illustration of the solution, the consequences and the result from assessing the appropriateness of the pricing pattern.

## **9 ACKNOWLEDGEMENTS**

I would to thank my supervisors Jaap Kabbedijk and Slinger Jansen. Jaap provided me detailed feedback while being constructive and realistic, but also coming up with new ideas during valuable meetings. Also special thanks to Slinger for providing me extensive feedback and introducing me at people at AFAS, the company where I spent most of my time working on my thesis.

Furthermore, I would to thank the AFAS employees who participated somehow in my research, whether as a survey respondent or as an interviewee for being open and transparent. I want to thank especially, the ‘colleagues’ from Controlling for having a good time at AFAS and CFO Arnold Mars for introducing me to people at AFAS to conduct interviews and for providing me valuable data.

Special thanks for my girlfriend Queeny for supporting me during the whole project and saving me time by cooking nice meals after a long day working at my thesis. But also for keeping me motivated and for reviewing the thesis.

Last but not least, I would to think my friends and family for their support and patience, because I could not spent a lot of time with them in the last months.

## 10 REFERENCES

- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., et al. (2009). Above the Clouds : A Berkeley View of Cloud Computing Cloud Computing : An Old Idea Whose Time Has ( Finally ) Come. *Computing*, 7-13.
- Arya, P. K., Venkatesakumar, V., & Palaniswami, S. (2010). Configurability in SaaS for an electronic contract management application. *Proceedings of the 12th international conference on Networking VLSI and signal processing (2010)*, 210-216. Retrieved from <http://dl.acm.org/citation.cfm?id=1820538.1820574>
- Aulbach, S., Grust, T., Jacobs, D., Kemper, A., & Rittinger, J. (2008). Multi-tenant databases for software as a service: schema-mapping techniques. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08* (p. 1195). New York, New York, USA: ACM Press. doi:10.1145/1376616.1376736
- Bezemer, C. P., & Zaidman, A. (2010). Multi-tenant saas applications: Maintenance dream or nightmare? *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)* (pp. 88–92). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1862393>
- Brownsword, L., & Oberndorf, T. (2000). Developing new processes for COTS-based systems. *Software, IEEE*, (August). Retrieved from [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=854068](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=854068)
- Chong, F., Carraro, G., Wolter, R., Corporation, M., Architecture, A., & Architecture, R. M.-tenant D. (2006). Multi-Tenant Data Architecture Three Approaches to Managing Multi-Tenant Data. Retrieved from <http://msdn.microsoft.com/en-us/library/aa479086.aspx>
- Choudhary, V. (2007). *Software as a Service : Implications for Investment in Software Development* The Paul Merage School of Business. Sciences-New York, 1-10.
- Gao, B., An, W. H., Sun, X., Wang, Z. H., Fan, L., Guo, C. J., & Sun, W. (2011). A Non-intrusive Multi-tenant Database Software for Large Scale SaaS Application. *2011 IEEE 8th International Conference on e-Business Engineering*, (1), 324-328. Ieee. doi:10.1109/ICEBE.2011.23
- Ghaddar, A., & Tamzalit, D. (2012). Variability as a Service: Outsourcing Variability Management in Multi-tenant SaaS Applications. *Advanced Information Systems ...*, 1-15. Retrieved from [http://subversion.assembla.com/svn/simon\\_papers\\_repo/trunk/Ali/CAISE-2012-Accepted Paper/CAISE-PAPER.pdf](http://subversion.assembla.com/svn/simon_papers_repo/trunk/Ali/CAISE-2012-Accepted Paper/CAISE-PAPER.pdf)
- Guo, C. J., Sun, W., Huang, Y., Wang, Z. H., & Gao, B. (2007). A Framework for Native Multi-Tenancy Application Development and Management A Native Multi-tenancy Enablement Framework Challenges of the Native Multi-tenancy Pattern.
- Hamilton, J. (2007). On designing and deploying internet-scale services. *LISA07 Proceedings of the 21st conference on Large Installation System Administration Conference*, 1-12. USENIX Association. Retrieved from <http://portal.acm.org/citation.cfm?id=1349444>
- Hogan, J. (2005). What is Strategic Pricing? Strategic Pricing Group Insight.



- Internet Economics. (1998). MIT Press Books. Retrieved from <http://econpapers.repec.org/RePEc:mtp:titles:0262631911>
- Jacobs, D., & Aulbach, S. (2007). Ruminations on Multi-Tenant Databases. (A. Kemper, H. Sch Ning, T. Rose, M. Jarke, T. Seidl, C. Quix, & C. Brochhaus, Eds.) BTW Proceedings, 103(Btw), 514-521. GI. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.6429&rep=rep1&type=pdf>
- Jansen, S., Brinkkemper, S., & Finkelstein, A. (2007). Providing Transparency In The Business Of Software: A Modeling Technique For Software Supply Networks. IFIP Advances in Information and Communication Technology, 243, 677-686. doi:10.1007/978-0-387-73798-0\_73
- Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). A sense of community: A research agenda for software ecosystems. 2009 31st International Conference on Software Engineering - Companion Volume (pp. 187-190). IEEE. doi:10.1109/ICSE-COMPANION.2009.5070978
- Jansen, S., Houben, G.-J., & Brinkkemper, S. (2010). Customization Realization in Multi-tenant Web Applications: Case Studies from the Library Sector. In B. Benatallah, F. Casati, G. Kappel, & G. Rossi (Eds.), (Vol. 6189, pp. 445-459). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-13911-6
- Kabbedijk, J., & Jansen, S. (2011). Variability in Multi-tenant Environments: Architectural Design Patterns from Industry. Advances in Conceptual Modeling. Recent, 151-160. doi:10.1007/978-3-642-24574-9\_20
- Karlsson, J. (1998). An evaluation of methods for prioritizing software requirements. Information and Software Technology. Retrieved from <http://www.sciencedirect.com.proxy.library.uu.nl/science/article/pii/S0950584997000530>
- Katzmarzik, A. (2011). Product Differentiation for Software-as-a-Service Providers. Business & Information Systems Engineering, 3(1), 19-31. doi:10.1007/s12599-010-0142-4
- Kittlaus, H.-B., & Clough, P. N. (2009). Software Product Management and Pricing. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-76987-3
- Klaus, H., Rosemann, M., & Gable, G. G. (2000). What is ERP? Information Systems Frontiers, 2(2), 141-162. Springer. doi:10.1023/A:1026543906354
- Kwok, T., & Mohindra, A. (2008). Resource calculations with constraints, and placement of tenants and instances for multi-tenant SaaS applications. Service-Oriented Computing—ICSOC 2008, 633–648. Springer. doi:10.1007/978-3-540-89652-4\_57
- Kwok, T., Nguyen, T., & Lam, L. (2008). A Software as a Service with Multi-tenancy Support for an Electronic Contract Management Application. 2008 IEEE International Conference on Services Computing, 179-186. Ieee. doi:10.1109/SCC.2008.138
- Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. Biometrics, 33(1), 159-174. International Biometric Society. doi:10.2307/2529310

- Lehmann, S., & Buxmann, P. (2009). Pricing Strategies of Software Vendors. *Business & Information Systems Engineering*, 1(6), 452-462. Gabler Verlag. doi:10.1007/s12599-009-0075-y
- Ma, D., & Seidmann, A. (2008). The Pricing Strategy Analysis for the “ Software-as-a-Service ” Business Model. (J. Altmann, D. Neumann, & T. Fahringer, Eds.) *Grid Economics and Business Models*, 5206, 103-112. Springer Berlin Heidelberg. doi:10.1007/978-3-540-85485-2
- Mankiw, N. G., & Taylor, M. P. (2006). *Microeconomics* (p. 474). Cengage Learning EMEA.
- Meszaros, G., & Doble, J. (1997). A pattern language for pattern writing, 529-574. Retrieved from <http://dl.acm.org/citation.cfm?id=273448.273487>
- Miranda, B. D., Baida, Z., & Gordijn, J. (2006). Modelling Pricing for Configuring e-Service Bundles, (Grönroos 2000), 1-13.
- Motahari-nezhad, H. R., Stephenson, B., & Singhal, S. (2009). Outsourcing Business to Cloud Computing Services : Opportunities and Challenges Outsourcing Business to Cloud Computing Services : Opportunities and Challenges. *Development*, 10(4), 1-17. Retrieved from <http://www.hpl.hp.com/techreports/2009/HPL-2009-23.pdf>
- Mäkilä, T., Järvi, A., Rönkkö, M., & Nissilä, J. (2010). How to Define Software-as-a-Service – An Empirical Study of Finnish SaaS Providers. *Lecture Notes in Business Information Processing* (pp. 115-124). doi:10.1007/978-3-642-13633-7\_10
- Nitu. (2009). Configurability in SaaS (software as a service) applications. *Proceeding of the 2nd annual conference on India software engineering conference - ISEC '09* (p. 19). New York, New York, USA: ACM Press. doi:10.1145/1506216.1506221
- Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), 131-164. doi:10.1007/s10664-008-9102-8
- Sawyer, S. (2000). Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems*, 9(1), 47-58. doi:10.1057/palgrave.ejis.3000345
- Spruit, M., & Abdat, N. (2012). The Pricing Strategy Guideline Framework for SaaS Vendors. *International Journal of Strategic Information Technology and Applications (IJSITA)*, 3(1), 38-53. doi:10.4018/jsita.2012010103
- Stamelos, I., & Angelis, L. (2003). Estimating the development cost of custom software. *Information & ...* Retrieved from [http://www.winfobase.de/lehre%5Civ\\_materialien.nsf/intern01/C760C9B911EAEAD5C1256E6D004B9D41/\\$FILE/estimating the development cost of custom software.pdf](http://www.winfobase.de/lehre%5Civ_materialien.nsf/intern01/C760C9B911EAEAD5C1256E6D004B9D41/$FILE/estimating%20the%20development%20cost%20of%20custom%20software.pdf)
- Summers, D. (2001). *Longman Dictionary of Contemporary English: Plus New Words (LDOC)* (p. 166901). Longman. Retrieved from <http://www.amazon.com/Longman-Dictionary-Contemporary-English-Words/dp/0582456398>

- Sun, W., Zhang, X., Guo, C. J., Sun, P., & Su, H. (2008). Software as a Service: Configuration and Customization Perspectives. 2008 IEEE Congress on Services Part II (services-2 2008), 18-25. Ieee. doi:10.1109/SERVICES-2.2008.29
- Sundararajan, A. (2004). Nonlinear Pricing of Information Goods. *Management Science*, 50(12), 1660-1673. JSTOR. doi:10.1287/mnsc.1040.0291
- Svahnberg, M., van Gurp, J., & Bosch, J. (2005). A taxonomy of variability realization techniques. *Software: Practice and Experience*, 35(8), 705-754. doi:10.1002/spe.652
- Takeda, H., & Veerkamp, P. (1990). Modeling Design Processes. *AI Magazine*, 11(4), 37-48.
- Tsai, C.-H., Ruan, Y., Sahu, S., Shaikh, A., Shin, K., Clemm, A., Granville, L., et al. (2007). Virtualization-Based Techniques for Enabling Multi-tenant Management Tools. In A. Clemm, L. Z. Granville, & R. Stadler (Eds.), *Managing Virtualization of Networks and Services* (Vol. 4785, pp. 171-182). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-75694-1
- Vaishnavi, V., & Kuechler, W. (2007). Design Research in Information Systems. *Order A Journal On The Theory Of Ordered Sets And Its Applications*, 48(2), 133-140. Auerbach. Retrieved from <http://desrist.org/design-research-in-information-systems>
- Van De Weerd, I., & Brinkkemper, S. (2008). Handbook of Research on Modern Systems Analysis and Design Technologies and Applications. (M. R. Syed & S. N. Syed, Eds.) *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications* (p. 35). IGI Global. doi:10.4018/978-1-59904-887-1
- Ward, J., & Peppard, J. (2002). Strategic planning for information systems (3rd ed., p. 624). John Wiley & Sons, Ltd. Retrieved from [http://books.google.com/books?hl=en&lr=&id=Y-djKt6DaV8C&oi=fnd&pg=PR5&dq=strategic+planning+for+information+system&ots=oFriK9D7Ht&sig=sJv4RXsSwUF\\_zxZ\\_22x2f7XUc-Y](http://books.google.com/books?hl=en&lr=&id=Y-djKt6DaV8C&oi=fnd&pg=PR5&dq=strategic+planning+for+information+system&ots=oFriK9D7Ht&sig=sJv4RXsSwUF_zxZ_22x2f7XUc-Y)
- Wellhausen, T. (2011). How to write a pattern ? Retrieved from <http://www.tim-wellhausen.de/papers/HowToWriteAPattern.pdf>
- Wu, S.-yi. (2010). Best Pricing Strategy for Information Services. *Journal of the Association for Information Systems*, 11(6), 339-366. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.168.2501&rep=rep1&type=pdf>
- Xin, M., & Levina, N. (2008). Software-as-a-Service Model: Elaborating Client-Side Adoption Factors. *SSRN Electronic Journal*. SSRN. doi:10.2139/ssrn.1319488
- Xu, L., & Brinkkemper, S. (2007). Concepts of product software. *European Journal of Information ...*. Retrieved from <http://www.ingentaconnect.com/content/pal/0960085x/2007/00000016/00000005/art00003>
- Yin, R. K. (2008). Case Study Research: Design and Methods. (L. Bickman & D. J. Rog, Eds.) *Essential guide to qualitative methods in organizational research* (Vol. 5, p. 219). Sage Publications. doi:10.1097/FCH.0b013e31822dda9e

# 11 APPENDIX

## 11.1 METRICS

Metric	Description	source
<b>transaction</b>	The price depends on the number of transactions executed by the software. This can be a technical assessment base (e. g. Web Service invocation) or an assessment base with regard to contents (e. g. number of crawled delivery items)	Lehmann & Buxmann, 2009
<b>memory requirements</b>	The price is ascertained by units of memory requirements (e. g. per GB).	Lehmann & Buxmann, 2009
<b>time</b>	The amount of the price is determined by the actual duration of software usage (e.g. price per minute)	Lehmann & Buxmann, 2009
<b>named user</b>	The right to use the software is bounded to a specific person and therefore the price refers to a defined user.	Lehmann & Buxmann, 2009
<b>concurrent user</b>	This pricing unit allows the simultaneous use of the software with a pre-defined number of users.	Lehmann & Buxmann, 2009
<b>server/machine</b>	Customers will be charged for each server or machine. The rights of use are bounded to this server or machine.	Lehmann & Buxmann, 2009
<b>CPU</b>	The price of the software refers to the amount of CPUs in use.	Lehmann & Buxmann, 2009
<b>master data</b>	The price of the software refers to the number of entered master data (e. g. customers, suppliers, employees, inventory, rental units, land parcels, managed assets).	Lehmann & Buxmann, 2009
<b>locations</b>	The price applies per location. This includes special forms of locations (e. g. mines).	Lehmann & Buxmann, 2009
<b>produced amount</b>	The software is priced according to the production of the customer (e. g. produced barrels oil per day).	Lehmann & Buxmann, 2009
<b>key performance indicators</b>	The price refers to Key Performance Indicators (e. g. revenue, expenses, budget)	Lehmann & Buxmann, 2009
<b>Committed Monthly Recurring Revenue (CMRR)</b>	Also referred to as the Monthly Committed Recurring Revenue (MCCR). The total of committed subscriptions for the period.	Dunham, 2009
<b>Retention Rate (RR)</b>	Also known as Churn. The percent of customers that renew their subscription at the end of the term. This includes “automatic” renewals, even though they may be a separate class in some implementations because they can always “opt-out” – although the tendency is less than in manual renewal systems.	Dunham, 2009
<b>In Trial</b>	The number of prospects (not users in a company or organization subscription model) subscribed in a trial period. SaaS ISVs offering trial subscriptions should also track Average Users per Customer during trial (to answer: how large are the trial prospects? Do they cover a full Line of Business? Might they need a Sales Engineer to help them evaluate?). The Average Trial Period is also of interest and depending on the sales model can be different from Time to Close.	Dunham, 2009

<b>Average Size (ADS) or the Average Revenue Per Client (ARPC)</b>	Deal or Revenue	the average Committed Monthly Recurring Revenue (CMRR) from a customer.	Dunham, 2009
<b>Average Revenue Per User (ARPU)</b>		Like the ARPC but broken down per user instead of per client. When the size of client implementations vary broadly across your customer base, this helps to establish your average user load and revenue. The impact of this metric depends on the subscription or revenue model of the application.	Dunham, 2009
<b>Time to Close</b>		The time it takes for an identified prospect to become a subscribing customer. This can be easily monitored for trial customers but requires CRM to monitor for a sales team.	Dunham, 2009
<b>Close Rate</b>		The percentage of In Trial customers that convert to subscribers. This can be aggregated with sales team prospects from CRM but it wise to also follow it separately to gauge pure web-based marketing and sales.	Dunham, 2009
<b>Cost to Maintain (CtM)</b>		The cost of services required to maintain customer instances. This should include hosting charges, hardware and software renewal, support, staff operations and outside services required to maintain customer instances outside of sales, marketing, R&D, and product development. This is an important metric because it also provides a window into Contingency Costs.	Dunham, 2009
<b>Cost to Acquire (CtA)</b>		The average of the cost of sales and marketing activities in a period divided by the number of customers acquired in the following period. The period is usually adapted from the Time to Close so that costs and the customer acquisition relationships are realistic.	Dunham, 2009
<b>Customer Acquisition Cost Ratio (CAC)</b>		Customer Acquisition Cost Ratio (CAC) – A key indicator of how long it will take a customer to “payback” their Cost to Acquire. The ratio is developed by dividing the Annualized Net Gross Margin added during a quarter by sales and marketing costs of the previous quarter. Example: CAC Ratio (Q408) = $\frac{[GM(Q408)-GM(Q308)]*4}{\text{Sales \& Marketing Costs (Q308)}}$	Dunham, 2009
<b>Customer Lifetime Value Ratio (CLV)</b>		Customer Lifetime Value Ratio (CLV) – The net present value of Annual Recurring Revenue from a customer less the Cost to Maintain and Cost to Acquire. A lifetime “horizon” needs to be assumed – usually a 3-5 year window is used unless the company has been in business long enough to make assumptions based on Churn.	Dunham, 2009
<b>Software to Services Ratio</b>		In a subscription only model, professional services may supplement income. In a mixed model, subscriptions can be supplemented with “value-add” services that can be transaction or on-demand based and professional services. In either case, this is a ratio of subscription revenue to services and is important when there is a low margin remaining after Cost to Maintain and Cost to Acquire are taken out. A high services ratio can compensate for the low margin if the services are considered to be high value by your	Dunham, 2009

		customers and in high demand.	
<b>Largest Customer %</b>		Knowing the percentage of gross income your largest customer provides helps to assess risk. In a new company or a vertically-based SaaS, it might be also wise to assess the percentage of gross income provided by your top ten (largest) customers. The point is to assess your risk if they drop their subscription and leave you in a position of supporting sales, marketing and operations with too low a margin.	Dunham, 2009
<b>Packaged Trial</b>	<b>Perpetual</b>	Single license purchased for a single user or machine. Traditionally sold as out-of-the-box software. Permanent licenses purchased upfront. Examples include node-locked, user-locked, or unlocked. Users able to try software before purchasing	Ferrante, 2006
<b>Server (per CPU)</b>		Number of processors running determines number of licenses purchased.	Ferrante, 2006
<b>Network-based</b>		Uses a centralized system to distribute licenses to network users.	Ferrante, 2006
<b>Subscription-based</b>		License purchased for some time period.	Ferrante, 2006
<b>Utility-based</b>		Customer charged according to time product is used (pay per use).	Ferrante, 2006
<b>Access</b>		A company pays for access to the application and data, regardless of how many users there are or how much usage is made. This metric makes sense for applications with a large differentiated value that have relatively few users or intermittent use, and whose value generation depends on relatively unfettered access. A refinement of this is to breakdown pricing by function.	Forth, 2010
<b>Users</b>		The number of users that can access the application is a common metric, this can be the absolute number of users or the number of named users (when people need to personalize their data). Per user metrics make sense when the value created by the application is primarily at the individual level or there are operating costs that scale with the number of users. This is the most common pricing metric for SaaS vendors and is used for many services by industry leaders	Forth, 2010
<b>Usage</b>		In some cases it is how often the software is used, or how many resources that are consumed, that maps best to value or cost. This approach is especially relevant to commoditized services such as cloud computing.	Forth, 2010
<b>Performance</b>		The ultimate pricing metric is to link price to outcomes. Typical examples are a percentage of savings, or a percentage of new sales, or a percentage of margin improvement. These pricing metrics are rare because outcomes usually depend on multiple inputs and it can be very difficult to untangle these.	Forth, 2010

**Table 23 Overview of metrics**

## **11.2 MAIL FOR CRITERIA REQUEST**

### **Dutch:**

*Beste [voornaam],*

*Mijn naam is Dennis Adriaansen en ik doe momenteel onderzoek bij AFAS naar prijsmodellen voor SaaS als onderdeel van mijn afstuderen aan de Universiteit Utrecht voor de studie Business Informatics.*

*Als onderdeel van de evaluatie van mijn onderzoek wil ik in kaart brengen welke criteria van belang zijn voor een goed prijsmodel. Vandaar mijn vraag aan jou als [functie binnen ERPComp]:*

*Wat zijn volgens jou de eigenschappen/criteria van een goed prijsmodel?*

*Hierbij kun je zowel rekeninghouden met je functie als [functie binnen ERPComp], als het algemene AFAS 'belang'.*

*Een korte opsomming is voldoende, maar een korte motivatie mag er natuurlijk bij.*

*Alvast bedankt voor de medewerking.*

*Groeten,*

*Dennis Adriaansen*

### **English:**

*Dear [first name],*

*My name is Dennis Adriaansen and I am currently performing research on the topic of pricing models at AFAS as part of my graduation at the Utrecht University for the master of Business Informatics.*

*As part of the evaluation of my research I want to gather criteria that are important for a solid pricing model. Therefore I have the following question for you as [..].*

*What are the characteristics of a good pricing model according to you?*

*Hereby you can take into account both your responsibilities as a [function] as the general ERPComp interests.*

*Thanks in advance for your cooperation.*

*Best regards,*

*Dennis Adriaansen*

### 11.3 ANSWERS FROM CRITERIA REQUEST

#### Chief Financial Officer (Expert 1) during an interview.

*“A good pricing model is:*

- 1. competitive*
- 2. easy for administration*
- 3. allowing clear insight for customer”*

In Dutch:

*“Een goed prijsmodel is:*

- 1. concurrerend*
- 2. eenvoudig om te verwerken*
- 3. inzichtelijk voor de klant”*

#### Sales Manager (Expert 2) during interview

This person preferred a face to face meeting instead of answering the criteria by e-mail.

*“A good pricing model takes into account that it should serve for multiple, different markets. It should also serve for different types of customers.*

*Cheap/competitive.*

*Prefers recurring costs instead of one time investment costs.”*

#### Sales manager (Expert 3) by e-mail

*“A good pricing model:*

- 1. Consistent with the market demand*
- 2. Is flexible enough in terms of the balance between use vs. price to pay*
- 3. Is transparent and without hidden costs*
- 4. Gives little to no space for discounts*
- 5. Is simple enough to communicate to prospects without complex calculations.”*

In Dutch:

*“Een goed prijsmodel:*

- 1. Sluit aan bij de vraag uit de markt;*
- 2. Is voldoende flexibel als het gaat om de balans tussen gebruik vs. te betalen prijs (klanten kopen immers ‘gebruiksrecht’*
- 3. Is transparant en zonder verborgen kosten (kosten die in de toekomst waarschijnlijk gemaakt moeten worden maar voor de klant bij aanschaf niet te overzien zijn)*
- 4. Geeft weinig tot geen ruimte voor kortingen / stunts*
- 5. Is simpel en zonder al teveel voorkennis te communiceren met prospects zonder hier al teveel ‘berekeningen’ op los te hoeven laten”*

#### Response Sales Manager (Expert 4)

*“A good pricing model is simple, predictable and prevents misuse.”*



## 11.4 SCREENSHOTS EVALUATION TOOL: CHARACTERISTICS

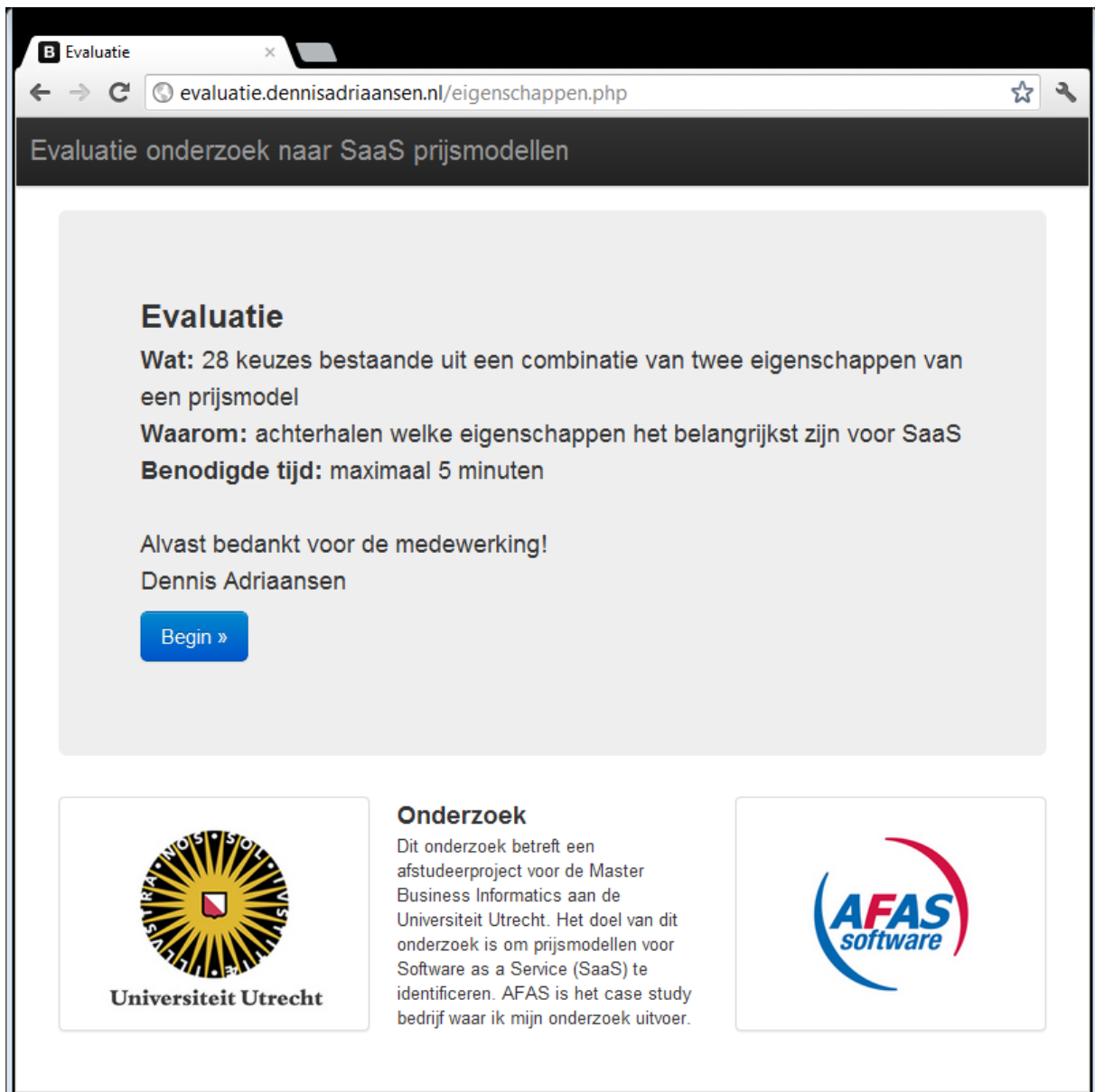


Figure 19 Screenshot of start screen

## **11.5 E-MAIL TEMPLATE FOR FIRST EVALUATION**

Beste [first name],

Mijn naam is Dennis Adriaansen en ik doe momenteel onderzoek bij AFAS naar prijsmodellen voor SaaS als onderdeel van mijn afstuderen aan de Universiteit Utrecht voor de studie Business Informatics.

Als onderdeel van mijn evaluatie wil ik graag de acht eigenschappen van een goed prijsmodellen prioriteren. Hiervoor heb ik een website gemaakt waarin telkens twee van de genoemde eigenschappen naast elkaar worden gezet waarbij de keuze moet worden gemaakt welke van de twee het belangrijkste is.

De website is: <http://evaluatie.dennisadriaansen.nl/eigenschappen.php>

Het betreft een relatief korte evaluatie die binnen vijf minuten uit te voeren is, waarbij 28 keer wordt gevraagd welke eigenschap belangrijker is. Zou je deze evaluatie willen invullen?

Alvast bedankt!

Groeten,

Dennis Adriaansen

## **11.6 E-MAIL TEMPLATE FOR THE SECOND EVALUATION**

Beste [first name],

Mijn naam is Dennis Adriaansen en ik doe momenteel onderzoek bij AFAS naar prijsmodellen voor SaaS als onderdeel van mijn afstuderen aan de Universiteit Utrecht voor de studie Business Informatics.

Tijdens mijn onderzoek zijn er meerdere manieren om SaaS te prijzen naar voren gekomen. Het doel van deze evaluatie is om na te gaan in hoeverre bepaalde eigenschappen van toepassing zijn op deze prijsmodellen. Hierbij kan ik de ervaring van een accountmanager natuurlijk goed gebruiken.

De evaluatie is uit te voeren op de volgende webpagina:

<http://evaluatie.dennisadriaansen.nl/prijsmodellen.php>

Er worden 64 stellingen voorgelegd waarbij op een schaal van 1 tot 5 beoordeeld moet worden of je het eens bent met de stelling of niet. Deze evaluatie zal zo'n 5 tot 8 minuten in beslag nemen.

Ik hoop dat je de evaluatie wil invullen, zodat ik binnenkort mijn resultaten aan Arnold kan presenteren.

Alvast bedankt voor de medewerking!

Groeten,

Dennis Adriaansen

## 11.7 RESULTS PAIR WISE COMPARISON CHARACTERISTICS

characteristic A	characteristic B	Preference per respondent						
		4	5	6	21	22	24	27
transparant (zonder verborgen kosten)	een relatie tussen gebruik en prijs	A	A	B	A	A	B	A
transparant (zonder verborgen kosten)	eenvoudig voor de administratie	A	A	A	A	A	A	A
transparant (zonder verborgen kosten)	voorspelbaar	A	A	A	A	A	A	A
transparant (zonder verborgen kosten)	repeterende kosten	A	A	A	A	A	A	A
transparant (zonder verborgen kosten)	eenvoudig te berekenen	B	A	A	A	A	A	A
transparant (zonder verborgen kosten)	geschikt om verschillende markten te bedienen	A	A	B	B	A	A	A
transparant (zonder verborgen kosten)	duidelijk voor de klant	B	B	A	A	A	B	B
een relatie tussen gebruik en prijs	eenvoudig voor de administratie	B	A	A	B	A	A	A
een relatie tussen gebruik en prijs	voorspelbaar	A	B	A	A	A	A	B
een relatie tussen gebruik en prijs	repeterende kosten	A	A	A	B	A	A	A
een relatie tussen gebruik en prijs	eenvoudig te berekenen	A	A	A	A	A	A	B
een relatie tussen gebruik en prijs	geschikt om verschillende markten te bedienen	B	A	A	B	A	A	A
een relatie tussen gebruik en prijs	duidelijk voor de klant	B	A	A	A	B	B	B
eenvoudig voor de administratie	voorspelbaar	A	B	B	A	A	A	B
eenvoudig voor de administratie	repeterende kosten	A	B	B	A	A	B	A
eenvoudig voor de administratie	eenvoudig te berekenen	A	B	B	A	A	B	B
eenvoudig voor de administratie	geschikt om verschillende markten te bedienen	B	B	B	A	B	B	B
eenvoudig voor de administratie	duidelijk voor de klant	B	B	B	A	B	B	B
voorspelbaar	repeterende kosten	B	A	A	B	B	A	A
voorspelbaar	eenvoudig te berekenen	B	B	B	B	B	B	B
voorspelbaar	geschikt om verschillende markten te bedienen	B	A	B	B	B	A	A
voorspelbaar	duidelijk voor de klant	B	A	B	B	B	B	B
repeterende kosten	eenvoudig te berekenen	B	B	B	A	B	B	B
repeterende kosten	geschikt om verschillende markten te bedienen	B	B	B	A	B	A	B
repeterende kosten	duidelijk voor de klant	B	B	B	A	B	B	B
eenvoudig te berekenen	geschikt om verschillende markten te bedienen	B	A	B	B	B	A	A
eenvoudig te berekenen	duidelijk voor de klant	B	B	A	B	B	B	B
geschikt om verschillende markten te bedienen	duidelijk voor de klant	B	B	A	A	A	B	B

**Table 24 Results of the pair wise comparisons**

## 11.8 RESULTS ASSESSMENT

		1	2	3	4	5	$P_i$
Statement id		nr of assessments					
1	usage dependent pricing is transparent	2	2	3	0	0	0,238
2	usage dependent pricing has a relation between usage and price	0	1	3	1	2	0,190
3	usage dependent pricing is easy for administration	2	3	1	1	0	0,190
4	usage dependent pricing is predictable	2	2	2	0	0	0,200
5	usage dependent pricing has recurring costs	2	2	3	0	1	0,179
6	usage dependent pricing is easy to calculate	2	1	2	0	1	0,133
7	usage dependent pricing is usable to serve multiple markets	1	1	1	1	4	0,214
8	usage dependent pricing is clear insight for the customer	1	2	3	2	0	0,179
9	situation-based pricing is transparent	1	0	1	3	1	0,200
10	situation-based pricing has a relation between usage and price	0	2	1	3	1	0,190
11	situation-based pricing is easy for administration	0	1	2	3	1	0,190
12	situation-based pricing is predictable	1	1	1	2	2	0,095
13	situation-based pricing has recurring costs	0	1	1	4	1	0,286
14	situation-based pricing is easy to calculate	0	1	1	2	2	0,133
15	situation-based pricing is usable to serve multiple markets	0	2	2	1	4	0,222
16	situation-based pricing is clear insight for the customer	0	1	2	2	1	0,133
17	cost based pricing is transparent	3	0	3	0	0	0,400
18	cost based pricing has a relation between usage and price	4	0	3	0	0	0,429
19	cost based pricing is easy for administration	2	4	1	1	0	0,250
20	cost based pricing is predictable	2	2	2	0	0	0,200
21	cost based pricing has recurring costs	1	2	3	0	1	0,190
22	cost based pricing is easy to calculate	2	2	2	0	0	0,200
23	cost based pricing is usable to serve multiple markets	1	2	3	0	0	0,267
24	cost based pricing is clear insight for the customer	1	5	1	0	0	0,476
25	flat fee pricing is transparent	0	0	2	1	4	0,333
26	flat fee pricing has a relation between usage and price	3	0	2	2	0	0,238
27	flat fee pricing is easy for administration	0	0	1	1	4	0,400
28	flat fee pricing is predictable	0	0	1	1	4	0,400
29	flat fee pricing has recurring costs	0	0	2	2	2	0,200
30	flat fee pricing is easy to calculate	0	0	2	2	4	0,286
31	flat fee pricing is usable to serve multiple markets	0	0	2	3	3	0,250
32	flat fee pricing is clear insight for the customer	0	0	1	1	5	0,476
33	two part tariff pricing is transparent	3	1	2	0	0	0,267
34	two part tariff pricing has a relation between usage and price	0	1	3	1	2	0,190
35	two part tariff pricing is easy for administration	2	3	3	0	0	0,250
36	two part tariff pricing is predictable	0	2	2	2	0	0,200
37	two part tariff pricing has recurring costs	0	1	3	2	1	0,190

38	two part tariff pricing is easy to calculate	1	2	1	2	0	0,133
39	two part tariff pricing is usable to serve multiple markets	0	1	4	0	2	0,333
40	two part tariff pricing is clear insight for the customer	3	1	1	3	0	0,214
41	peak load pricing is transparent	4	2	1	1	0	0,250
42	peak load pricing has a relation between usage and price	1	1	2	1	1	0,067
43	peak load pricing is easy for administration	4	1	1	0	0	0,400
44	peak load pricing is predictable	4	1	1	0	1	0,286
45	peak load pricing has recurring costs	2	1	2	1	1	0,095
46	peak load pricing is easy to calculate	4	2	0	2	0	0,286
47	peak load pricing is usable to serve multiple markets	2	1	3	1	0	0,190
48	peak load pricing is clear insight for the customer	5	1	1	0	0	0,476
49	bundle pricing is transparent	1	0	1	3	2	0,190
50	bundle pricing has a relation between usage and price	1	2	0	4	0	0,333
51	bundle pricing is easy for administration	0	1	1	3	3	0,214
52	bundle pricing is predictable	1	0	1	2	3	0,190
53	bundle pricing has recurring costs	1	1	1	1	2	0,067
54	bundle pricing is easy to calculate	0	1	1	2	3	0,190
55	bundle pricing is usable to serve multiple markets	2	0	2	3	1	0,179
56	bundle pricing is clear insight for the customer	0	1	2	1	2	0,133
57	dual pricing is transparent	2	1	2	1	0	0,133
58	dual pricing has a relation between usage and price	1	2	1	1	2	0,095
59	dual pricing is easy for administration	1	0	3	2	0	0,267
60	dual pricing is predictable	1	2	3	0	0	0,267
61	dual pricing has recurring costs	0	0	3	2	2	0,238
62	dual pricing is easy to calculate	1	2	3	2	0	0,179
63	dual pricing is usable to serve multiple markets	0	0	0	3	4	0,429
64	dual pricing is clear insight for the customer	2	1	1	3	1	0,143
	<b>Total</b>	82	76	115	88	81	
	$p_i$	0,186	0,095	0,155	0,141	0,151	
	K	0,141					

Table 25 Results of the assessment

## 11.9 POINTS AWARDED TO PRICING PATTERNS

Pricing pattern / characteristic	Points
<b>usage dependent pricing:</b>	<b>-11,9</b>
- is not transparent	-24
- is usable to serve multiple markets	12,1
<b>situation-based pricing:</b>	<b>17,2</b>
- has recurring costs	5,1
- is usable to serve multiple markets	12,1
<b>cost based pricing:</b>	<b>-80,1</b>
- is not transparent	-24
- has no relation between usage and price	-16,2
- is not easy for administration	-5,9
- is not usable to serve multiple markets	-12,1
- has no clear insight for the customer	-21,9
<b>flat fee pricing:</b>	<b>62,5</b>
- is transparent	24
- has no a relation between usage and price	-16,2
- is easy for administration	5,9
- is predictable	6,1
- is easy to calculate	8,7
- is usable to serve multiple markets	12,1
- has clear insight for the customer	21,9
<b>two part tariff pricing:</b>	<b>-17,8</b>
- is not transparent	-24
- is not easy for administration	-5,9
- is usable to serve multiple markets	12,1
<b>peak load pricing:</b>	<b>-66,6</b>
- is not transparent	-24
- is not easy for administration	-5,9
- is not predictable	-6,1
- is not easy to calculate	-8,7
- has no clear insight for the customer	-21,9
<b>bundle pricing:</b>	<b>-5,9</b>
- is easy for administration	-5,9
<b>dual pricing:</b>	<b>11,1</b>
- is not predictable	-6,1
- has recurring costs	5,1
- is usable to serve multiple markets	12,1

**Table 26 Overview of all points awarded to the pricing patterns for selecting the most appropriate pricing patterns for business SaaS applications**