UNIVERSITY
OF TWENTE.

Master of Science in Business Administration

"The price is right? Evaluating revenue models for software components in Identity and Access Management"

Muhammed Bozkurt

A research in collaboration with University of Twente and InnoValor, a consultancy agency specialized in digital trust, business models and agility.

UNIVERSITY
OF TWENTE.

Colophon

Date	:	31 May 2016
Version	:	1.0
Project reference	:	Master thesis
Status	:	Final version

Author	:	Muhammed Bozkurt
Student number	:	s1608010
E-mail	:	m.bozkurt@student.utwente.nl
Education	:	MSc. Business Administration
Track	:	Information management
Institute	:	University of Twente School of Management and Governance Enschede, the Netherlands
Company	:	Innovalor Enschede, the Netherlands
University supervisors	:	Ir. Björn Kijl I. Singaram MSc.
Company supervisor	:	Arnout van Velzen MSc.

Synopsis

In this explorative research, a Software Component Revenue Model Evaluation Framework is developed software component developers in Identity and Access Management can use to evaluate revenue models.

Acknowledgements

I would like to express my sincere gratitude to several people who supported me during the period of this master thesis. Firstly, to Björn Kijl for providing the academically guidance and the inspiring conversations throughout this process. Secondly, to Raja Singaram for his advice and feedback in the final phase of this thesis. Also, a special thanks to Arnout van Velzen for his faithful counsel, patience and confidence in me. As last, to my family for their unconditional support and for keeping their faith in me I would complete this thesis.

Enschede, 31 May 2016

Muhammed Bozkurt

“There is little point in reinventing the wheel”

Jankowicz, 2005

Management summary

The world is changing, and changing fast. From education to healthcare to financial services, there is hardly anything that remains constant. As the world is changing, so is software. Namely, due to the rapid development of computer technology, businesses becoming more networked and products and services are built in collaboration, traditional methods of developing software have been difficult to adapt to present-day complex and changing application requirements. Hence, software component gained increased popularity over the last two decades. Surprisingly, mainly technical aspects of software components is discussed in extant literature. Whereas, the business aspects have received little attention. This master thesis focusses on the business aspects of software components. More specifically, this thesis investigates appropriate revenue models for software component developers in Identity and Access Management. The reason is that, software components have unique properties, for example, they have to be integrated in other applications to create value. This causes specific challenges for software component developers regarding revenue models (e.g. monitoring the usage).

The aim of this study was to create a framework which helps software component developers in IAM to evaluate revenue models. Therefore, the following central research question was applied: *"How can software component developers in Identity and Access Management evaluate revenue models for exploiting software components?"* To guide this research, the concept of business model is used. The reason is that, the following assumption was made; the revenue model choice is influenced by the overall business model. Therefore, based on the Business Model Canvas of Osterwalder and Pigneur (2010), a conceptual framework was developed for analysing revenue models. The current study focusses on software component developer in Identity and Access Management, because many of the software solutions in this industry are provided in the form of a software component (e.g. Google Authenticator).

Due to the exploratory nature of this study, a mixed method qualitative research design was preferred. As a starting point, a review of literature was conducted to gain a thorough understanding of the concepts involved in this study (i.e. software components, business models and software revenue models). Next, an exploratory interview study was conducted. Thereafter, case studies were conducted. During the literature review, four revenue models

were identified as applicable revenue models for software components in Identity and Access Management. The revenue models are one-time-charge, usage-based, subscription and freemium. Subsequently, since the assumption was made that the revenue model choice is influenced by the overall business model, an exploratory interview study was conducted to investigate whether the made assumption could be justified. Through exploratory interviews, it was indeed observed that several business model elements affect the organizations' revenue model choice. Next, through case studies, the list with appropriate revenue models for software components in IAM was reduced to three; one-time-charge, subscription and freemium. In addition, the case study enabled to map out the individual business model elements which affect the viability of the three revenue models.

Based on these findings, the Software Component Revenue Model Evaluation framework was developed. The Software Component Revenue Model Evaluation framework presents the most critical business model factors which impact the viability of the revenue models; one-time-charge, subscription and freemium. Based on this, the following central research question which was applied in this study can be answered: *"How can software component developers in Identity and Access Management evaluate revenue models for exploiting software components?"* The Software Component Revenue Model Evaluation framework can be used by software component developers in Identity and Access Management to evaluate revenue models. By using the framework, software component developers in IAM can make well-considered choices regarding revenue models.

***Keywords**: software components, componentware, reusable software, commercial-of-the-shelf, modified-of-the-shelf, revenue model, revenue stream, business model.*

Contents

1. Introduction 10
2. Literature review 14
2.1 Methodology of the literature review 14
2.2 Software components 15
2.2.1 Software components are on the rise 15
2.2.2 Defining software components 16
2.2.3 Advantages and disadvantages 18
2.2.4 Software component business and it stakeholders 21
2.3 Business Models 24
2.3.1 Distinction between business models, strategy and business processes 24
2.3.2 Defining business models 25
2.3.3 Business Model Frameworks 27
2.4 Revenue models in the software business 31
2.4.1 Revenue model versus business model 31
2.4.2 Revenue models in the software industry 32
2.4.3 Advantages and disadvantages 34
2.5 Conclusion literature review 39
3. Methodology 41
3.1 Research design 41
3.2 Conceptual framework 43
3.2 Data collection 44
3.2.1 Exploratory interviews 44
3.2.2 Case studies 45
3.3 Data analysis 46
4. Data analysis 48
4.1 Exploratory interviews 48
4.2 Within-case analysis 51
4.1.1 Company A 52
4.1.2 Company B 56
4.1.3 Company C 60
4.3 Cross-case analysis 63
4.2.1 Highlights from findings 63

4.2.2 Analyzing individual revenue models 64
5. Results 66
5.1 Revenue models for software components in Identity and Access Management 66
5.2 Advantages and disadvantages of revenue models for software components 66
5.3 Business model elements which impact the viability of software component revenue models 68
5.4 Presenting the Software Component Revenue Model Evaluation framework 70
6. Example case 73
6.1 Background information 73
6.2 Applying the SCRME framework 74
7. Conclusion & discussion 78
7.1 Conclusion 78
7.2 Discussion 79
7.3 Contribution 82
7.3.1 Scientific contribution 82
7.3.2 Practical contribution 82
7.4 Limitations and recommendations for further research 83
Abbreviations 85
Appendices 86
Appendix I: Interview topic list (exploratory interviews) 86
Appendix II: Interview topic list (case studies) 88
Appendix III: Overview revenue models of the case companies 91
Appendix IV: Analysis: one-time-charge revenue model 93
Appendix V: Analysis: subscription-based revenue model 95
Appendix VI: Analysis: freemium revenue model 97
Appendix VII: Analysis: usage-based revenue model 99
Appendix VIII: Topic list interview company X (example case) 100
References 101

1. Introduction

This chapter covers the background and problem area of the research project. It also discusses the research goal and formulates a central research question. Moreover, the relevance of this study is discussed. Finally, a thesis outline is specified.

SOFTWARE COMPONENTS are on the rise (Guntersdorfer, Kay, & Rosenblum, 2000; Ulkuniemi, Araujo, & Tähtinen, 2015). According to a research conducted by Forrester Consulting (2011), where they examined practices in the software industry with respect to managing software quality, security, and safety to identify key market trends and best practices, they found out that the reliance on third-party code (e.g. commercial code, outsourced software and open source code) is increasing. Almost all of 336 organizations in their survey utilizes some form of third-party code, and many (40%) of the respondents rely on software from multiple suppliers. This is not surprising because the usage of software components has multiple benefits, e.g. it can decrease development lead time, transfers risks to suppliers, diminishes maintenance cost, allows organizations to achieve better quality, and enables faster technology adoption and innovation (Helander & Ulkuniemi, 2012; Mingbai, 2011; Raemaekers, van Deursen, & Visser, 2012; Schlauderer & Overhage, 2011; Sridhar, 2015; Ulkuniemi et al., 2015).

The software component industry can be characterized as striving towards a high level of standardization of software interfaces (Helander & Ulkuniemi, 2006). This means, software components are traded like standard items as off the shelf software (Helander & Ulkuniemi, 2012). Despite the increasing adoption of software components in a wide variety of applications (Ayala, Hauge, Conradi, Franch, & Li, 2011), there is no unified definition of software components (Wu & Woodside, 2004). Although, generally speaking, a software component exhibits the following characteristics; it is an independent, compositional and deployable unit which has clearly defined and well documented interfaces interacting with other components (Szyperski, 1998; Wu & Woodside, 2004).

Software components are not a recent phenomenon. The idea of software components was introduced in the nineteen sixties (Heiander, Ulkuniemi, & Seppänen, 2002), but it was never as popular as it is now. Due to the rapid development of computer technology, the

traditional methods of developing software have been difficult to adapt to present-day complex and changing application requirements (Mingbai, 2011). Moreover, businesses are becoming more networked and products and services are built in collaboration (Halinen & Mainela, 2013; Palo & Tähtinen, 2013; Ritter, 2013). Hence, it has been claimed that *"It is becoming not only impractical, but also virtually impossible for mainstream IT organizations to ignore the growing presence of third party software in major segments of the IT industry"* (Gartner, 2008). Thus, it is not surprising that software components have received a lot of attention in academic literature.

However, in extant literature, mainly technical aspects of developing and using software components have been discussed (Helander & Ulkuniemi, 2006). These aspects include, for example, quality assurance and architectural issues in component based software development. Especially, the buyer's or component assembler's perspective has received a lot of attention (Liu & Gorton, 2003). Whereas, the component developer's perspective has not gained as much attention, especially from a business point-of-view (Helander & Ulkuniemi, 2006; Lampson, 2004).

Specifically, little information regarding revenue models for software components can be found in academic literature. Therefore, to contribute to science, revenue models for software components are studied. In this research, the term "revenue model" is used in an operational sense, referring to how a firm collects revenue from its customers. There are two main reasons for focussing on revenue models. First of all, software components have unique properties, for example, they have to be integrated in other applications to create value. This causes specific challenges for software component developers regarding revenue models. For example, when customers pay on a per-use basis, billing can be a major overhead. This is because software components are integrated in other applications, whereby often the component does not remain at the developer's side, which means the component developer has difficulty monitoring the usage. Moreover, the emphasis is on revenue models because a well-defined revenue model is a critical part of commercializing products and services. A revenue model impacts the overall business model, from customers to channels and from sales force to back office (Kittlaus & Clough, 2008). So, this research fills the scientific gap by researching revenue models for software component developers.

The goal of this study is to contribute to knowledge on revenue models for software components. More specifically, this research investigates appropriate revenue models for exploiting software components. The end-result of this study is to create a Software Component Revenue Model Evaluation framework, which software component developers in Identity and

Access Management can use to evaluate revenue models. In this way, this research will serve as a companion for businesses who provide software component solutions for appropriate revenue model decisions. Therefore, aside from the academic relevance, the practical relevance of this study is that it will help software component suppliers to make well-considered choices regarding revenue models.

In this research, the following central research question was applied: *"How can software component developers in Identity and Access Management evaluate revenue models for exploiting software components?"* Viability in this research is defined as "*the ability of the revenue model to compete effectively on the market and to make profit*". Whereas, exploiting meant *to benefit from.* To provide guidance, sub-research questions are formulated. The answers to these sub-questions contribute to providing an answer to the central research questions which is stated above. The sub-research questions are formulated as follows:

1. What are the predominant revenue models for software components?
2. Which advantages and disadvantages can be identified for these revenue models from the perspective of a software component developer?
3. What are the most critical business model element(s) which impact(s) the viability of revenue models for software components?

The first sub-question will help to provide an overview of revenue models for software components. The second sub-question will help to map out the advantages and disadvantages of the revenue models from the software component developers' perspective. The last sub-question will help to create an understanding of which revenue models software component developers prefer under which (business model) circumstances.

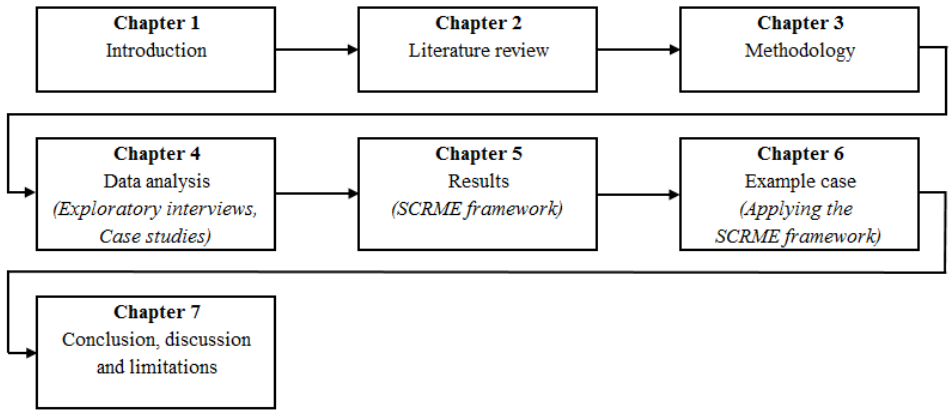
To guide this research, the concept of a business model is used. More specifically, the advantages and disadvantages of revenue models and the reasons of which revenue models software component developers prefer under which circumstances are looked at through the lens of a business model. The reason for using the business model concept is because a business model is a powerful tool to analyse an organization (Magretta, 2002; Osterwalder & Pigneur, 2010). Besides, a revenue model is an element of the whole business model (Magretta, 2002; Osterwalder & Pigneur, 2010; Teece, 2010). For this reason, the assumption is made that the revenue model choice is influenced by the overall business model. As a limitation to the study, other possible influences regarding the revenue model choice (e.g. social-psychological influences, environmental factors, competition etc.) are beyond the scope of the study.

Since this study aims to yield insights into a relatively unexplored topic, this study is classified as explorative research (Babbie, 2013). Due to the exploratory nature of this research,

a qualitative mixed method research design was preferred because a combination of different data from different methods leads to a comprehensive understanding of complex problems (Creswell & Clark, 2007). The current study focusses on software component developers in Identity and Access Management (IAM). According to Gartner (2015), IAM is the security discipline that enables the right individuals, at the right time, to access the right resources for the right reasons. IAM software provides a single identity for users so they can access a computer system or network and manages the rights and permissions a user has so that the user is only able to do those things she or he has been authorized for (Bloor, Baroudi, & Kaufman, 2007). IAM solutions are usually integrated in software for business processes. Therefore, many of these solutions are provided by suppliers in the form of a software component (e.g. Google Authenticator). This makes the IAM industry an attractive context for this research.

This paper is structured as follows. Firstly, a literature review is presented wherein relevant concepts are discussed. More specifically, the concept of software components, its advantages and disadvantages, and its stakeholders are described. Moreover, it will present the concept of business models, including business model frameworks and the Business Model Canvas by Osterwalder and Pigneur (2010). Subsequently, attention is given to revenue models for software components. In the next chapter, the methodology for conducting exploratory interviews and case studies is described. Thereafter, the results are presented, which includes the Software Component Revenue Model Evaluation framework. The SCRME framework helps software component developers in IAM to evaluate revenue models. Next, to illustrate the SCRME framework and how it may be used, an example case is presented based on a software component developer which delivers a solution for Identity and Access Management. In the next chapter, a conclusion is drawn and a discussion is presented. Lastly, the limitations of this study and some directions for future research are included. Figure 1 presents the overview of the structure of this research.

Figure 1: Overview of the structure of the thesis



2. Literature review

This section starts with the methodology of the literature review (2.1). Subsequently it discusses relevant concepts for this study, namely: Software components (2.2), Business Models (2.3) and Revenue models (2.4). Lastly, a conclusion of the literature review is presented (2.5).

2.1 Methodology of the literature review

As Jankowicz (2005) stated: *"There is little point in reinventing the wheel. Whatever your epistemology, the work that you do is not done in a vacuum, but builds on the ideas of other people who have studied the field before you. This requires you to describe what has been published, and to marshal the information in a relevant and critical way"* Therefore, the first part of this study is based on a comprehensive review of literature, which provides the foundation on which this research is build (Saunders, Lewis, & Thornhill, 2009).

The literature review was conducted between September 2015 and February 2016. To find relevant articles the following search engines were accessed: Google Scholar, Scopus and Science Direct. No publication date limits or language restrictions were applied. There are quite a few interchangeable terms that refer to software components, such as componentware, reusable software, commercial of the shelf and modified of the shelf. For this reason, a varying combination of the following key words were used in searches: "software component" OR "componentware" OR "reusable software" OR "commercial of the shelf" OR "modified of the shelf" AND "business model" OR "business strategy" OR "business" AND "revenue model" OR "revenue stream" OR "revenue strategy" OR "revenue". At the same time, the extracted publications were reviewed in terms of included references to other potentially relevant publications.

Unfortunately, no paper was identified which discusses revenue models for software components. Therefore, the literature review was divided into three parts: software components, business models and (software) revenue models were discussed individually. Subsequently, the findings were combined and a first overview of appropriate revenue models for software components was mapped out.

2.2 Software components

This part describes the concept of software components. It starts by giving a brief history of software components (paragraph 2.2.1). In paragraph 2.2.2 a definition of software component is given. Subsequently, paragraph 2.2.3 introduces arguments that motivate the use of software component by presenting the advantages and disadvantages. Finally, relevant stakeholders in the software component industry (paragraph 2.2.4) are presented where the focus will be particularly on component developers.

2.2.1 Software components are on the rise

In 1968, it was realized that software was getting very complicated and too large to manage and that there was a need for different ways to handle it. Hence the concept of software components was introduced in a paper by Doug MCIlroy (1968), who described software components as useful fragments of a software system that can be assembled with other fragments to form larger pieces or complete applications. Software components are based on the idea of Service-Oriented-Architecture (SOA). SOA is considered as the infrastructure supporting communications between services (Shi et al., 2015). As Bloor, Baroudi and Kaufman (2007, p. 12) described in their book, SOA is like the *form* of a dance: "*If you dance any kind of formal dance, from the cha-cha to the waltz, you know that form matters. The form is what allows you to dance with someone you've never met. When both partners truly know the form, they move in tandem, are flexible, and navigate with ease and grace".* This example perfectly shows the idea behind SOA; SOA is the form of a dance, whereas software components are the dancers. With SOA, loose coupling is enabled, here is where software components fit in. Loose coupling is an approach to interconnect components in a system to provide a service, whereby the components are not dependent on each other. Because the components are not dependent on each other, they can be mixed and matched with other component services as needed. With other words, new pieces of software (software components) can be plugged and unplugged in systems or applications. This allows software services to be more flexible. Moreover, it offers new business models by enabling organizations to combine different software programs to create new products/services. So, SOA is not only a technical approach (of how to develop software), it also changes the way of how organizations deliver services. Further on in this paper these aspects are discussed in greater detail.

Getting back to the history of software components, in the late 1970s, the first software reuse project called DRACO had started (Neighbors, 1989). The next burst of activity regarding software components was in the mid-1980s. A book on object oriented programming was

published by Brad Cox, where he introduced the idea of 'software integrated circuits' (Cox, 1985). Software integrated circuits are software that can be plugged and unplugged, replaced and reconfigured, and which can be traded on the open market as any other standard product, for example like trading car parts. These integrated circuits are today's software components. Subsequently, in the nineteen nineties, intra-organizational software had emerged and in the late nineteen nineties software components gained wide acceptance. Although, despite the fact that the concept of software components already was 30 years old, it was still in an early stage of development (Heiander et al., 2002).

Nowadays, software components are more popular than ever (Guntersdorfer et al., 2000; Ulkuniemi et al., 2015). First of all, this is due to the industrialization of software (Bloor et al., 2007), which was made possible through the emergence of standards (e.g. web services interfaces and XML) to interconnect software components. Moreover, businesses becoming more networked, products and service being built in collaboration, and a trend towards outsourcing is understandable (Halinen & Mainela, 2013; Palo & Tähtinen, 2013; Ritter, 2013). For example, combining software components from third-parties through the use of APIs to create new products/services has triggered a new type of economy; the API-economy. (Gat & Succi, 2013; Janes, Remencius, Sillitti, & Succi, 2014). In fact, numerous businesses are taking advantage of the economic opportunities offered by exposure of their proprietary code to external software developers through APIs, such as Amazon, Google, Facebook, Foursquare and many others, to become industry platform leaders. Industry platforms are services, products or technologies which are developed by one or several businesses, and which serve as foundations upon which other developers can build complementary products, services, or technologies (Gawer, 2009).

2.2.2 Defining software components

Building systems from components stems from other engineering disciplines, such as electrical or civil engineering (Crnković, Sentilles, Vulgarakis, & Chaudron, 2011). However, due to the fact that software is in its nature different from physical products, a direct translation from the classical engineering disciplines into software engineering is not possible (Crnković et al., 2011). For example, in contrast with classical engineering disciplines where the understanding of the term component has never been a problem, there has been much debate on the notion of software components. Besides, software components are called by many names which causes confusion, such as programs, applications, modules, functions, dynamic link libraries, subroutines, and classes, yet they all refer to software components (Bloor et al., 2007).

Nevertheless, much effort has been devoted to defining and describing the terms and concepts involved. According to Szyperski (1998) software components have the following characteristics; 1) *a component is a unit of independent deployment*, 2) *a component is a unit of third-party composition*, and 3) *a component has no persistent state*. Subsequently, the following definition of a software component is formed by Szyperski (1998, p. 34): "*A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties".* Whereas, Helander and Ulkuniemi (2006, p. 3) use another definition: *"A software component is a reusable computer program that is integrated into a larger software based system solution as an individual operational part, and that is not valued by the end customer as a standalone application".* Moreover, other researcher use the following definition: *"A software component is an independent and reusable computer program which is accessible through specified interfaces"* (Brereton & Budgen, 2000; Broy et al., 1998; Councill & Heineman, 2001).

From the characterizations above, the following definition of a software component is formed: "*A software component is an independent and reusable computer program that is integrated into a larger software-based solution and is accessible through specified interfaces. Moreover, it is subject to composition by third parties and it is not valued by the end customer as a standalone application*". In this definition of a software component, both modified-off-the-shelf (MOTS) software and commercial-off-the-shelf (COTS) software are included. MOTS software is offered by the supplier with services for tailoring the software to specific requirements of the acquirer, whereas COTS products are standardized software components, to be used without any or little modifications by the customer (Heiander et al., 2002). However, in both cases the idea is the same: a piece of software is sold to customers as a building block for other software-based solutions. Nevertheless, modifications of MOTS software components most likely will result in additional or even substantial development costs to the seller or buyer, compared to COTS software components.

2.2.3 Advantages and disadvantages

So, why are software components on the upswing? What is the rationale behind the adoption of software components? Software can be broadly be defined in two categories (Szyperski, 1998). In one case, an application is developed entirely from scratch (custom-made software). In the other case, software is bought and implemented in a solution without the need of tailoring the software or limited adaptation (standard software). Custom-made software has significant advantages: it can be optimally adapted to the business processes of the applying party and it can take advantage of any in-house proprietary knowledge or practices. However, custom-made software also has its disadvantages. Producing software from scratch is a very expensive undertaking. Besides, compatibility with other software is often a burden. For example, integrating custom-made software with software of business partners or clients is very difficult, because there is a big chance their software/interface have other requirements. As a result of this, many projects fail completely to meet project requirements.

Standard-software, in the form of software components, has various advantages over custom-made software. According to Szyperski (2000), three sets of arguments are in favour of software components:

(1) *Baseline argument:* solutions that are component-based can combine acquired and bespoke software. By integrating already existing software components, costs are reduced by focusing on core competencies and by avoiding excessive reinvention of the wheel (Raemaekers et al., 2012; Sridhar, 2015; Szyperski, 2000; Ulkuniemi et al., 2015). Hence, non-strategic software components are bought, and strategic software components are made. In this way, businesses can maintain their competitive position while taking advantage of existing software components.

(2) *Enterprise argument:* when software component factoring is done skilfully, then several products of a product line can be offered by configuring a set of components, which subsequently makes product creation largely an issue of configuration. In addition, software components make version management less complex, because when a particular component is upgraded, no major release changes and upgrading of entire systems are needed.

(3) *Dynamic computing argument:* software components makes flexibility possible and meets a demand for flexibility as with the open and growing set of content types to be processed, more software systems are increasingly challenged to be dynamic. A web

browser is a good example. If it is well designed, it can be dynamically extended to meet new requirements on demand.

So, building new solutions by combining bought and made software components improves quality and moreover supports rapid development, which leads to a shorter time to market. Besides, adaptation to changing requirements can be achieved by investing only in key changes of a software-component-based solution, rather than under taking a major release change. Because once a system is modularized into components, there is less need for major release changes and upgrading of entire systems. Another important factor for adopting software components, is the scarcity of software developers leading software providers and clients without in-house capabilities (Ulkuniemi et al., 2015). Based on these arguments, software components are expected to be the corner store of software in the years to come (Ulkuniemi et al., 2015).

However, even though the use of software components is being blazoned as the way forward, software components also have downsides. Standard software, in the form of software components allow competitors to have access to them as well, hence limited competitive edge may be achieved by using standard software components. Moreover, producing reusable software designs is very expensive. According to Lampson (2004) it costs up to two times as much to build a module with a clean interface that is well-designed for integration into a system than just writing code without an interface. Whereas a reusable component costs three to five times as much as just to write the code, so the development costs are usually very high. The extra costs are due to; 1) *Generality*: the reusable software must meet the needs of a wide range of clients, so designing software in such a way is often very hard and therefore time-consuming, 2) *Simplicity*: the interface needs to be simple so clients can understand it easily, 3) *Customization*: in order to make the software general enough, it needs to be customizable, which often involves special-purpose programming, 4) *Testing*: clients use the software in different ways and assuring a high quality is very important, therefore the reusable software must be tested thoroughly, 5) *Documentation*: developing reusable software involves a lot of documentation, and 6) *Stability*: the reusable software has to be stable, which means it must cope with the changing application requirements of today's world.

Besides, standard software is not under local control of the buyer, so it might not adapt quickly enough to suit a changing need. So, as a business grows and expands, the existing features of the software component may not be scalable and therefore may no longer work. In

other words, the lack of the customers' control over the current and future functionality of the software component present some potential risk from the buyer's side. Also, the standard software might not be incompatible with the buyer's application (Pour, 1998; Sedigh-Ali, Ghafoor, & Paul, 2001). Furthermore, several risks are associated with the inexperience of the integrators and users of software components, which can lead to high costs. Besides, when a software system is developed from components, there might arise security issues for the developed system (Nazir, Shahzad, Nazir, & Rehman, 2013). This is because software often contains bugs, which can cause a "leak" in security for the developed application. Table 1 summarizes the advantages and disadvantages of software components.

Table 1: Advantages and disadvantages of software components

Advantages	Disadvantages
• Reduces costs (Raemaekers et al., 2012; Sridhar, 2015; Ulkuniemi et al., 2015)	• High initial development costs (Lampson, 2004)
• Decreases development time; shorter time to market (Heiander et al., 2002; Helander & Ulkuniemi, 2012; Raemaekers et al., 2012; Schlauderer & Overhage, 2011)	• Lack of customer control over the current and future functionalities (Vitharana, 2003)
• Allows organizations to achieve better quality (Heiander et al., 2002; Mingbai, 2011; Sridhar, 2015; Vitharana, 2003)	• Incompatibility with the buyer's application (Pour, 1998; Sedigh-Ali et al., 2001)

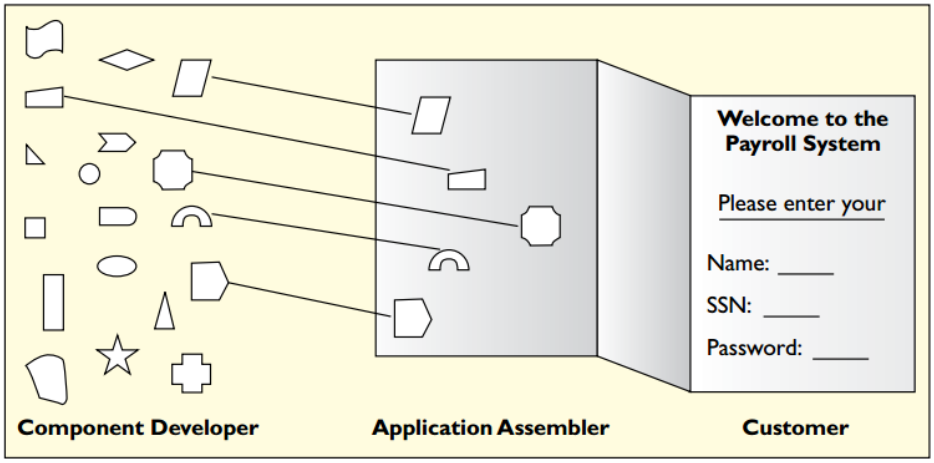
• Less need for major release changes (Szyperski, 2000)	• Security issues (Nazir et al., 2013; Vitharana, 2003)
• Makes product creation largely an issue of configuration (Szyperski, 2000)	
• Faster technology adoption and innovation (Mingbai, 2011)	

2.2.4 Software component business and it stakeholders

So, software components are on the upswing and it looks like they are here to stay. However, before we dive into the software component business, an overview of the characteristics of the general software industry is given to show that the software industry differs fundamentally from other industries. According to Popp and Meyer (2010) this may be ascribed to the specific characteristics of the product on the one hand and to the structure of the software markets on the other hand. A unique characteristic of software products, as any other digital good, is possibility of replication against negligible costs, because variable costs tend to be zero. Another characteristic of the software business is internationalization. Software is designed and developed on a global basis, and it can be distributed via the internet. As a result, competition between software providers evolves globally.

The software component industry has its own characteristics. First of all, software components are not valued as a standalone product, therefore they have to be integrated into other software applications or systems to create value. Hence, three primary stakeholders can be identified in the software component industry (Bergner, Rausch, Sihling, & Vilbig, 1998; Vitharana, 2003); 1) *Component developers*; these are freelance developers, Information Systems departments, and firms specializing in component fabrication, 2) *Application assemblers*; they locate suitable software components and assemble them into applications to satisfy customer requirements; and finally 3) *Customers*; they use the component-based application to perform certain tasks .

Figure 2: Stakeholders in the software component business (Vitharana, 2003)



As noted in the introduction of this paper, the focus in this study is on the first group of stakeholders, the *component developers*. The uniqueness of software components is that they have to be integrated in other software applications or systems to create value. This has consequences for the business model of software component developers. The reason is that, the value of the software depends on the software system or application it is integrated into. Moreover, the software systems which the software component is integrated into might limit the distribution channels of the software component suppliers. For example, in the software industry, common revenue models include usage-based and pay-per-use models (Kittlaus & Clough, 2008). This is enabled by today's technology, which support the development of virtual integrated systems wherein the software component functionality remains on the provider's servers, and client systems can access them when required (Kittlaus & Clough, 2008). For example, Software-as-a-Service (SaaS), which is one of the Cloud Computing service models, is getting more popular (Luoma, 2013). However, in the software component industry, usage-based revenue models can cause major overhead regarding billing. For example, when a software component needs to be integrated in a secured environment, the application assembler might not want his data to be stored at the software component provider's side. Therefore, he might request an on-premise implementation rather than a SaaS solution. In this case, monitoring the usage of the software component, necessary for billing, can bring difficulties for the software component developer. The system which the software component is integrated might not allow to communicate with external applications. So, the software component developer cannot monitor the usage directly. Of course, with a usage-based on-premise solution the component developer can request a monthly report of the usage, however in this scenario the application assembler can commit fraud easily. For example, when a foreign

customer, let’s take a customer in Japan, has many installations of the software component around the world, it is almost impossible for the component developer to obtain the knowledge of what and where the software component is installed. So, as this example illustrates, there might be a relationship between several components of a business model, as in this case; the trustworthiness of customers, the distribution channels and the revenue model.

However, other aspects of the business model can also play a substantial role regarding the revenue model choice. For example, how quick wants a component developer to recoup his development costs. If a component developer wants to recoup his development costs as soon as possible, he might choose for perpetual licensing. As these examples illustrates, several components of a business model might influence the revenue model choice. Therefore, in the next section, business models will be discussed comprehensively.

2.3 Business Models

In this section the concept of business models is discussed. It starts by explaining the differences between business models, business strategy and business processes (paragraph 2.3.1). Next, in paragraph 2.3.2, an overview of the most used definitions of business models is presented and subsequently a commonly used definition is adapted. Finally, Business Model Frameworks (paragraph 2.3.3) are discussed and an appropriate business model framework is chosen for this study.

2.3.1 Distinction between business models, strategy and business processes

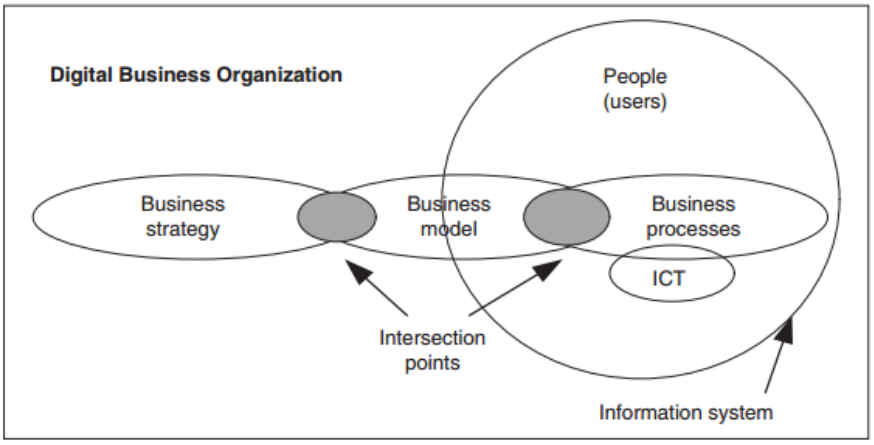
The business model concept was introduced in 1975, however it became popular during the last twenty years (W. Bouwman et al., 2012; Ghaziani & Ventresca, 2005; Zott, Amit, & Massa, 2011). The business model concept became prevalent with the advent of the internet in the mid-1990s, and since then is has been gathering momentum (Zott et al., 2011). From that time on, ideas revolving around the business model concept have increased tremendously with scholars and business practitioners. Basically, a good business model is crucial for every successful organization, regardless whether it is a new entrant or an established firm (Magretta, 2002). However, despite the increased interest in the concept, there is no widely accepted definition of a business model (W. Bouwman et al., 2012; Magretta, 2002; Zott et al., 2011). Moreover, the term is often confused with business strategy (Osterwalder, Pigneur, & Tucci, 2005).

Because the business model concept is relatively young, the role and place of it in the organization is still subject to debate. This has led to a lot of discussions in literature, regarding the positioning of business models with respect to business processes and strategy (Al-Debei & Avison, 2010; Osterwalder et al., 2005). However, there is already some consensus regarding the differences between business models and business processes (Morris, Schindehutte, & Allen, 2005). According to Osterwalder et al. (2005) the business model concept is generally understood as a view of the firm's logic for creating and commercializing value, whereas the business process model is more about how a business case is implemented in processes. To be more specific, the meaning of business processes are "*the clear translation of the mission and the structure of the business model into operational terms*" (Bouwman, De Vos & Haaker, 2008, p. 35). So, generally speaking, business models focus on 'what' a firm does, whereas business processes focus on 'how' firms work on operational level.

On the other hand, there is no consensus regarding the distinction between business models and business strategy. Some researchers are using the terms "strategy" and "business

models" interchangeably (Magretta, 2002). Other researchers explicitly state that the business model is not a strategy, and at the same time they include the strategy and/or parts of its elements within the business model (e.g. Shafer, Smith & Linder, 2005). Some researchers suggest an alternative way of looking at the business model concept. These researcher (Morris et al., 2005; Osterwalder et al., 2005) see the business model as an interface between the business strategy and the business processes. So, business models are acting as a sort of glue between business strategy and business processes. As illustrated in figure 3, Al-Debei and Avison (2010) show how business model intersects with business strategy and business processes. However, in the scope of this research, the focus is primarily on business models. Therefore, business strategy and business processes are not taken into account explicitly.

Figure 3: Business model intersection points (Al-Debei and Avison, 2010, p. 370).



2.3.2 Defining business models

Basically, business models are stories that explain how organizations work and it answers questions like; "Who is the customer?", "What does the customer value", and "How do we make money in this business?" (Magretta, 2002). They provide powerful ways to analyse, understand, communicate and subsequently manage strategic-oriented choices (Osterwalder et al., 2005; Shafer et al., 2005). However, there is no academic consensus about the definition of business models (W. Bouwman et al., 2012; Magretta, 2002; Zott et al., 2011). Surprisingly, in a research done by Zott et al. (2011), where they reviewed 103 business model publications, they found out that one-third (37%) of these publications did not define the concept at all, taking its meaning more or less for granted. Whereas 44% explicitly defined or conceptualized the business model. The other 19% publications referred to the work of other scholars in defining the concepts.

According to Al-Debei and Avison (2010) there are three main reasons for the murkiness around business models: *1) the youthfulness:* the business model concept is relatively young, 2) *multidisciplinary nature:* the business model concept comes from diverse disciplines such as eBusiness and eCommerce, strategy, business management, economics and technology and 3) *the newness of sectors within which the business model concept is investigate.* To show the complexity of the business model concept, the most prevalent and widely cited definitions are presented in Table 2.

Table 2: Selected business model definitions

Author(s) year	Definition	Times cited in Google Scholar
Timmers (1998)	*The business model is "an architecture for the product, service and information flows, including a description of the various business actors and their roles; and a description of the potential benefits for the various business actors; and a description of the sources of revenues" (p. 2).*	2715
Chesbrough and Rosenbloom (2002)	*The business model is "the heuristic logic that connects technical potential with the realization of economic value" (p. 529).*	2690
Margretta (2002)	*Business models are "stories that explain how enterprises work. A good business model answers Peter Drucker's age old questions: Who is the customer? And what does the customer value? It also answers the fundamental questions every manager must ask: How do we make money in this business? What is the underlying economic logic that explains how we can deliver value to customers at an appropriate cost?" (p. 4).*	2300
Morris et al. (2005)	*A business model is a "concise representation of how an interrelated set of decision variables in the areas of venture strategy, architecture, and economics are addressed to create sustainable competitive advantage in defined markets" (p. 727).*	1416

Teece (2010)	*"A business model articulates the logic, the data and other evidence that support a value proposition for the customer, and a viable structure of revenues and costs for the enterprise delivering that value" (p. 179).*	2042
Osterwalder and Pigneur (2010)	"A business model describes the rationale of how an organization creates, delivers, and captures value" (p. 14).	2809

The broad range of definitions supports the lack of consensus about the business model concept and represents a potential source of confusion (Zott et al., 2011). In this research, the widely used business model definition of Osterwalder and Pigneur (2010, p. 14) is used: "*A business model describes the rationale of how an organization creates, delivers, and captures value"*. The choice for the definition of Osterwalder and Pigneur (2010) is justified with the selection of a business model framework below.

2.3.3 Business Model Frameworks

Chesbrough and Rosenbloom (2002) provided one of the first business model frameworks. A business model framework describes the concept of a business model by the functions it fulfils. According to Chesbrough and Rosenbloom (2002) the functions of a business model are to: articulate the *value proposition*, identify *market segments*, define the structure of the *value chain* within the firm, estimate the *cost structure* and *profit potential*. Furthermore, it describes the position of the firm within the *value network* linking suppliers and customers, and it formulates the *competitive strategy* by which the innovating firm will gain and hold advantage over rivals.

Whereas Hedman and Kalling (2003) propose another business model framework, which is drawn both on strategy theory and related business model research. They propose the following framework consisting of: customers and competitors, offering (generic strategy), activities and organization (the value chain), resources, and supply of factor and production inputs, as well as longitudinal process component. Another framework is provided by Osterwalder (2004) which contains nine elements, the so-called building blocks. These elements are: 1) *value proposition*, 2) target customer, 3) distribution channel, 4) relationship, 5) value configuration, 6) capability, 7) partnership, 8) cost structure and 9) revenue model. For this research, the business model framework of Osterwalder (2004) will be used due to its

popularity. Namely, Osterwalder's business model framework found ground-breaking resonance and was cited by 1497 academic publications (Google Scholar, 2015). Osterwalder's and Pigneur's (2010) handbook *Business model generation: a handbook for visionaries, game changers, and challengers*, in which the Business Model Canvas is developed, was sold over one million times and the Business Model Canvas template is downloaded over 5 million times (Upward & Jones, 2015). Besides, the framework is in line with Osterwalder's (2004) definition of a business model which is used in this research.

2.3.3.1 Business Model Framework of Osterwalder

The business model framework of Osterwalder (2004) was created through two steps. First, Osterwalder identified four major areas that constitute a business model Osterwalder (2004). These areas are (Osterwalder, 2004, p. 42): 1) *Product*: what business the company is in, the products and the value propositions offered to the market: 2) *Customer interface*: who the company's target customers are, how it delivers products and services, and how it builds a strong relationship with them; 3) *Infrastructure management*: how the company efficiently performs infrastructural or logistical issues, with whom, and as what kind of network enterprise; and 4) *Financial aspects*: what is the revenue model, the cost structure and the business model's sustainability. Subsequently, these four areas (pillars) are split into nine interrelated building blocks, which are explained in table 3 below.

Table 3: The nine business model building blocks by Osterwalder (2004)

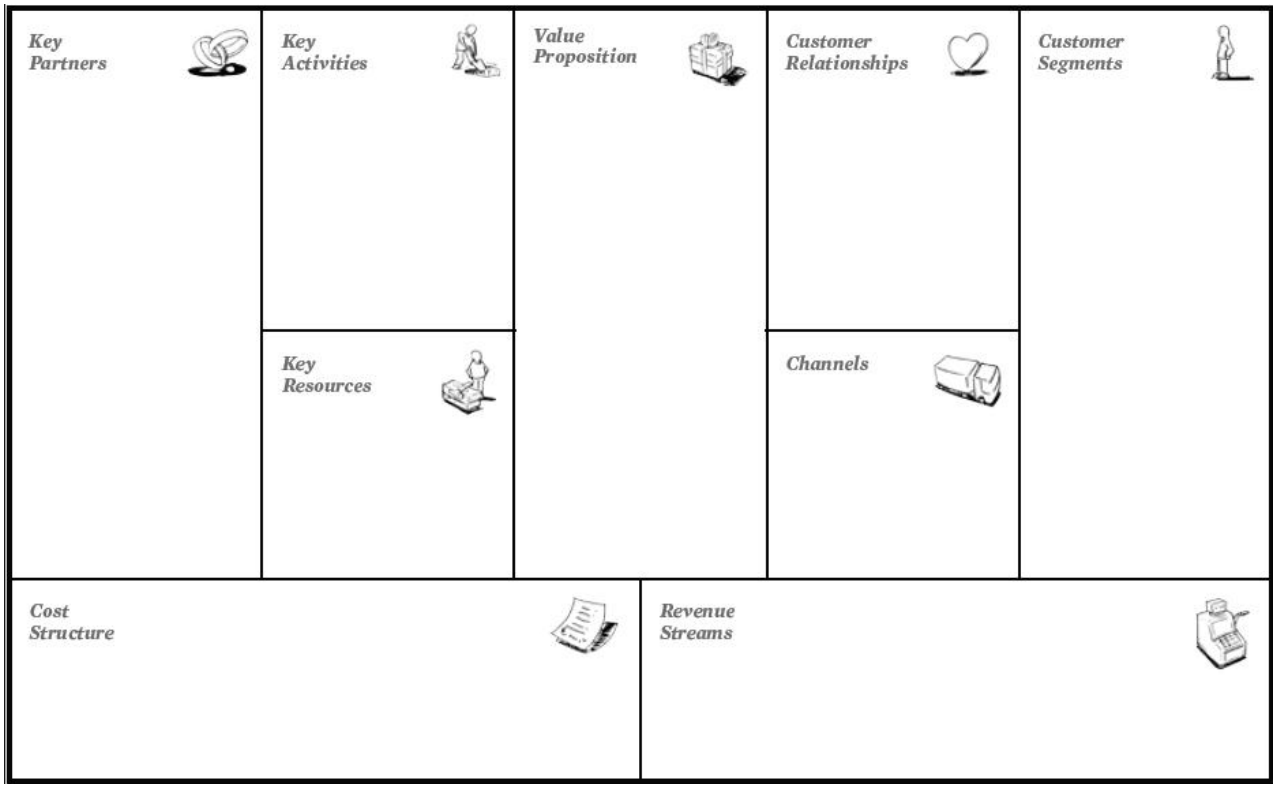
Pillar	**Building block**	**Description**
Product	**Value proposition**	A Value Proposition is an overall view of a company's bundle of products and services that are of value to the customer.
Customer interface	**Target customer**	The Target Customer is a segment of customers a company wants to offer value to.
	Distribution channel	A Distribution Channel is a means of getting in touch with the customer

	Relationship	The Relationship describes the kind of link a company establishes between itself and the customer.
Infrastructure management	**Value configuration**	The Value Configuration describes the arrangement of activities and resources that are necessary to create value for the customer.
	Capability	A capability is the ability to execute a repeatable pattern of actions that is necessary in order to create value for the customer.
	Partnership	A Partnership is a voluntarily initiated cooperative agreement between two or more companies in order to create value for the customer
Financial aspect	**Cost structure**	The Cost Structure is the representation in money of all the means employed in the business model.
	Revenue model	The Revenue Model describes the way a company makes money through a variety of revenue flows.

2.3.3.2 Business Model Canvas

Based on the Osterwalder's (2004) Business Model Framework, Osterwalder and Pigneur (2010) developed the Business Model Canvas tool. The Business Model Canvas tool allows to describe and think through the business model of an organization, competitors, or any other enterprise. Moreover, it allows to easily describe and manipulates business models to create new strategic alternatives (Osterwalder & Pigneur, 2010). To summarize, this tool can be used to create understanding, discussion, creativity, and analysis. Since this research aims to create an *understanding* of revenue models in the software component business, the Business Model Canvas by Osterwalder and Pigneur (2010) is used. The Business Model Canvas is displayed in Figure 4.

Figure 4: The Business Model Canvas (Osterwalder & Pigneur, 2010)



2.4 Revenue models in the software business

As discussed in the introduction of this research, in the academic literature no paper was identified that discusses appropriate revenue models for software components. Therefore, in this section the most common revenue models in the software business are identified. Subsequently, the identified software revenue models are analysed as to whether they are appropriate revenue models for software components. Finally, the advantages and disadvantages of the revenue models, which are found appropriate for software components, are discussed. However, before we dive into revenue models for software, this sub-section starts by giving a clear distinction between revenue models and business models

2.4.1 Revenue model versus business model

The terms "revenue models" and "business models" are often confused (George & Bock, 2011). This is not accurate, because "a business model describes the rationale of how an organization creates, delivers, and captures value" (Osterwalder and Pigneur, 2010, p. 14), whereas a revenue model describes the revenue flow. In this research, the term "revenue model" is used in an operational sense, referring to how a firm collects revenue from its customers. This means, it relates to the various payment options that a firm might offer to customers who want to buy its software components. A company can create a separate revenue model for each of its products and services. Moreover, each revenue model might have different pricing mechanisms (Osterwalder & Pigneur, 2010). As presented in the Business Model Canvas by Osterwalder and Pigneur (2010), a revenue model (revenue stream) is viewed as an important element of a business model. So, a revenue model does not define how a company creates value by itself, nevertheless it is clearly an important component of a business model.

According to Osterwalder and Pigneur (2010, p. 5) a business model can involve two different types of revenue streams: 1) *Transaction revenues resulting from one-time customers payment* and 2) *Recurring revenues resulting from ongoing payments to either deliver a value proposition to customers or provide post-purchase customer support*. In order to choose the right type of revenue stream, a company has to ask itself questions, questions like; "How are customers currently paying?" and "How would they prefer to pay?" Answering these questions successfully allows organizations to implement one or more revenue streams for each customer segment they have defined. Although the presented revenue streams by Osterwalder and Pigneur (2010) are clear and well adapted, they are not specific for the software industry.

Therefore the next subsection, will present an overview of common revenue models in the software industry.

2.4.2 Revenue models in the software industry

As discussed in the introduction, no software component specific revenue models were found in literature. Therefore, in this subsection first an overview is given of common revenue models for software-based business models. Subsequently, these revenue models are evaluated as to whether they are appropriate for software components. The uniqueness of software components is that they have to be integrated in other applications to create value. For this reason, to locate suitable revenue model(s) for software components, the following criterion is used: *the revenue model has to be viable when the software component is integrated in a system/application.* In the software industry the most common revenue models are: 1) one-time-charge, also known as perpetual licensing, 2) subscription fees, 3) usage-based revenue model, 4) freemium-models and 5) advertising-based revenue model (Cusumano, 2008; Kittlaus & Clough, 2008; Ojala, 2013; Ojala & Tyrväinen, 2012).

The *one-time-charge* revenue model is the most common revenue model in the software industry. When the one-time-charge revenue model is used, customer can buy a perpetual license which allows them to use the licensed software indefinitely. In addition to the initial fee, maintenance, support and updates are usually provided at an additional charge (Ojala & Tyrväinen, 2012).

A remarkable trend in the software industry is the transition from large up-front perceptual licenses fees (one-time-charge revenue model) to alternative stretch payments over a period of time. This trend has led to another often used revenue model in the software business; *term licensing/subscription*. With the subscription-model, customers do not own the software/product, but pay an annual/monthly fee to use the software. When a subscription is sold, in contrast to the one-time-charge revenue model, the annual/monthly fee often includes; maintenance, support and updates.

Due to the advent of SaaS solutions, software companies have the ability to measure resource usage at the level of individual users or systems (Bala & Carr, 2010). This enables the implementation of *usage-based* revenue models. With the usage-based revenue model, software firms have great flexibility. Namely, the measured usage may be based on, for example, pages printed, number of transaction, CPU consumed or any number of different metrics.

Aside from the usage-based revenue model, the *freemium model* is spreading quickly, especially among web start-ups (Miller, 2009). The idea of Freemium is based on providing a basic version of a product for free and to charge a premium for the full version (Günzel-Jensen & Holm, 2015; Niculescu & Wu, 2011; Pujol, 2010).

The last revenue model commonly applied in software business models is the *advertisement-based* revenue model. For publishers on the internet, advertising has become the dominant revenue generator (Kittlaus & Clough, 2008). With this revenue model, the software vendor gets paid by allowing advertisers to advertise within their software application, and offers the software service for free or at a discount.

Table 4: Common revenue models in the software business

Revenue model	**Description**
One time charge	"a method of charging in which a fee is charged initially for the use of the license and that the customer has the right to use the license in question for the capacity and quantity that is specifies, with no further payments" (Kittlaus and Clough, 2008, p. 130) .
Term licensing / Subscription-based	"an entitlement to use a software product over a specific period of time. Such a license is usually offered at a fixed price which often takes the form of annual payments for a fixed number of years, after which the customer must stop using the product, renew his term license, or license the product under other available terms" (Kittlaus and Clough, 2008, p. 132).
Usage-based	"charging for a software product based on some measurable defined metric" (Kittlaus and Clough, p. 134).

Freemium	The idea of Freemium is based on providing a basic version of a product for free and to charge a premium for the full version (Günzel-Jensen & Holm, 2015; Niculescu & Wu, 2011; Pujol, 2010).
Advertising-based	With the advertisement-based revenue model, the software vendor gets paid by allowing advertisers to advertise within their software application (Kittlaus & Clough, 2008).

The first four revenue models (one-time-charge, subscriptions, usage-based and freemium) do not impact the application which the software component is integrated in. For this reason, the first four revenue models are seen as appropriate revenue models for software components. The advertisement-based revenue model is not seen as an appropriate revenue model for software components. The reason is that, with the advertisement-based revenue model, the component developer earns money when ads are shown. Since, a software component is integrated in another application, the software component developer cannot force the *application assembler* to show ads in his software. So, the advertisement-based revenue model is left out of scope for this research. The advantages and disadvantages of the first four revenue models are discussed in the next sub-section.

2.4.3 Advantages and disadvantages

One time charge, subscription, usage-based and freemium revenue models were determined as possible appropriate revenue models for software components in the previous section. In this section, the advantages and disadvantages will be discussed per revenue model. However, before we dive into each individual revenue model, the notion is made that a software component developer has great flexibility in constructing his revenue model. In other words, several revenue models can be used at the same time. Also, revenue models can be used in combination as well. For example, a subscription revenue model may or may not be usage-based or when a perpetual license is sold, an additional "maintenance, support and updates" plan might be offered in the form of an annual subscription.

2.4.3.1 One time charge

An advantage of OTC is that it is a closed transaction. So, once it is done, there is nothing for the vendor to keep track of, the revenue is bookable and the customer has his license. These benefits help to cover the software development costs, and they do so in a short time in contrast to other revenue models, for example subscriptions (Ojala, 2013). Moreover, a high license fee also increases switching costs for the customer, so if the software is appropriate it increases customer loyalty (lock-in effect). Also, OTC makes it possible for customers to store and secure the data within the firm's own data centre. Another advantage of OTC is that it lends itself to discounting and promotions. However, OTC also has it downsides. Except for possible maintenance charges, OTC has no recurring revenues. Another disadvantage is when such a license is acquired, and a customer upgrades his hardware, he often is required to upgrade his license as well. But with upgrading to a newer version, usually hefty fees are involved, which poses challenges for the sales department. Moreover, in some cases the end-customer can sell the software when he does not want to use it anymore. This has an impact on the sales opportunities and customer base of the software vendor. Besides, since the software is installed on the customer's premises, misuse of the software or direct software piracy is more likely (Ojala, 2013). Besides, when a new version of the software becomes available, customers who have previously purchased the software have the choice between spending more money to purchase the upgrade or to use the existing software. Hence, customers usually do not upgrade, unless the new software provides substantial benefits (Choudhary, 2007). If there are various versions available, this causes challenges for the customer support department as well, because they have to provide support for each version.

2.4.3.2 Subscription-based

A big advantage of the subscription-based revenue model is that it leads to recurring revenues. When the customer stops paying, he loses the right to use the product. So, the subscription-based revenue model generates income as long as customer use the software. In contrast to OTC, where customer often have to pay a high license fee, with term licensing customers do not need to pay a huge amount upfront. So, for the customer a major advantage is that they have no major initial investment and can usually cancel their use of the product within a period of prior notice, which gives great flexibility. Also, since no high initial fees are involved, the software becomes interesting for organizations with less resources as well. In this way, a subscription-based revenue model can diversify the customer base. Besides, since the

subscription-based revenue model causes recurring revenues, it is likely to attract investors because investors are always looking for steady incomes (Kittlaus & Clough, 2008).

Aside from the various advantages, the subscription-based revenue model also has its disadvantage. Since, no large up-front licenses fees are involved, development costs are not recouped quickly. It can even take up to several years before the development costs are recouped.

2.4.3.3 Usage charging

Usage charging is "*charging for a software product based on some measurable defined metric, often the actual during a period*" (Kittlaus and Clough, p. 134). The measured usage may be based on amount of storage maintained, numbers of transactions, pages printed, or any number of different metrics. According to Kittlaus and Clough (2008) many customer prefer the usage charging, because many believe that they are light users of some programs and so it will give them a better price. From the seller's perspective, a usage charging mechanism can make their software available to customers who might not have the financial resources to buy software at an OTC, so it lets vendors to expand and diversify their customer base. Moreover, for customers, it has the advantage that they pay only when they are using the software, so it has an advantage over other models if customers need the software only occasionally. Nevertheless, the usage charging mechanism has three main disadvantages for software vendors. First of all, in this mechanism, where initial incomes are low and uncertain, recouping the software development costs is riskier than with traditional licensing. Moreover, since customers do not sign long-term contracts, they can switch to alternatives. Finally, as stated before, billing can be a major overhead in the software component industry. With usage charging it is key for component developer to monitor the use of their software by customers precisely. If the software component is not a SaaS solution, it is very difficult to monitor the usage, because when the software component is integrated in a secured environment often the component developer cannot access the system directly.

2.4.3.4 Freemium

As stated, the freemium model is spreading quickly, especially among web start-ups (Miller, 2009). Freemium models lends itself for software, because nowadays software is built using a modular architecture, which enables grouping, separating or locking certain features. Moreover, software has negligible costs and it can be distributed relatively easily via online distribution channels. Besides, freemium models are great for software, because software

belongs to the category of experience goods (Niculescu & Wu, 2011). So, with a freemium model, potential customers can first experience the product before they have to pay for it. There are different types of freemium models. For example, there is the *feature-limited-freemium*, which involves offering a basic version of the product with limited functionality for free while charging for additional features. Besides, also *time-limited-freemium* models exists, which allows users for a limited period of time free access to the full version of a product (e.g. 30-day free trials). The main benefit of a freemium model is that it lowers the barrier for potential customers to use software. Thereby, with a freemium model, different kind of customers can be reached, from small start-ups to big organizations. So, a freemium model can diversify the customer base. However, there are also disadvantages, for example users might not upgrade so development costs will not be recouped. Moreover, since the users do not have to make high investments in monetary terms, it is difficult to create a lock-in effect. Besides, if a sales process is involved, it will cost serious money to only get potential customers to use the (free) software. Therefore, the freemium-model might be a more appropriate revenue model for COTS than MOTS products, because with COTS products less costs are involved regarding the implementation of the product.

Table 5: Comparison revenue models from the software component developer's viewpoint

Revenue models	Advantages	Disadvantages
One time charge	• Helps to recoup development costs quickly • Creates a lock-in effect (switching costs are high for customers)	• No substantial income after initial purchase • High initial costs involved for potential customers, which limits the customer base

Subscription-based	• Causes recurring revenues • Diversifies the customer base • Increases profit when customers remain loyal • Attracts investors, since it causes recurring revenues	• Risk of not recouping development costs • Relatively low switching costs for customers
Usage charging	• Diversifies the customer base • Increases profit when customers use the software above average	• Risk of not recouping development costs • Relatively low switching costs • Less profit when customers use the software occasionally • Monitoring usage-metrics effectively
Freemium	• Easy to get customers on • Diversifies the customer base	• Risk of not recouping development costs • Risk of not turning free users into paying customers • Relatively low switching costs for users (difficult to create a lock-in effect)

2.5 Conclusion literature review

The literature review was meant as a starting point for this research. The literature review was divided into three parts; software components, business models and revenue models. Table 6 provides an overview of the core concepts in this research.

Table 6: Definitions of core concepts

Concept	Definition
Software components	*A software component is an independent and reusable computer program that is integrated into a larger software-based solution and is accessible through specified interfaces. Moreover, it is subject to composition by third parties and it is not valued by the end customer as a standalone application.*
Business Model Canvas	*"The Business Model Canvas tool allows to describe and think through the business model of an organization, competitors, or any other enterprise. Moreover, it allows to easily describe and manipulates business models to create new strategic alternatives"* (Osterwalder and Pigneur, 2010 p. 15).
Revenue model	*Revenue models are referring to how a firm collects revenue from its customers*

The first part of the literature review focussed on software components. In this part, the unique properties of software components were discussed and subsequently the following definition of a software component was given; "*A software component is an independent and reusable computer program that is integrated into a larger software-based solution and is accessible through specified interfaces. Moreover, it is subject to composition by third parties and it is not valued by the end customer as a standalone application*". This definition shows the unique property of a software component; it has to be integrated in another application to create value. Since it has to be integrated in another application, the revenue model of the software component developer might be limited by this.

However, before we dived into revenue models, the concept of business models was discussed. The reason is that, the assumption in this research is made that choosing an appropriate revenue model depends on the context which the business operates in. Eventually, it was decided that the Business Model Canvas of Osterwalder and Pigneur (2010) was the best business model tool/framework to come to an understanding of the context that a business operates in. In other words, the BMC tool will help to get a better understanding of which business model aspects influence the revenue model choice.

Finally, revenue models for software components were studied. No specific revenue models for the software component industry were found in extant literature. Therefore, common software revenue models were mapped out. Subsequently, these revenue models were analysed as to whether they are appropriate for software components. Eventually, four revenue models, namely one-time-charge, subscriptions, usage-based and the freemium revenue model were found appropriate revenue models for software components. These four revenue models form the basis for the rest of this study.

So, the literature review formed the basis for answering the first two sub-research questions; 1) "What are revenue models for software components?" and 2) "Which advantages and disadvantages can be identified for these revenue models from the perspective of a software component developer?" Next, the focus is on investigating the most critical business model element(s) which impact(s) the viability of revenue models for software components in Identity and Access Management. This is done by conducting an exploratory interview study and case studies, which will be elaborated upon in the next chapter.

3. Methodology

In this section, the research design (3.1), conceptual framework (3.2), data collection (3.3) and the data analysis methods (3.3) are discussed.

3.1 Research design

Since this study focusses on relatively new topics of interest, this study is classified as explorative research. Due to the exploratory nature of this research, a mixed method qualitative research design was preferred on the premise that a combination of different data from different methods leads to a comprehensive understanding of complex problems (Creswell & Clark, 2007). The overall research process is reported as follows.

Phase 1. *Literature review*

The literature review was meant as a starting point for this research. A review of literature was carried out in order to gain a thorough understanding of the concepts involved in this research (i.e. software components, business models and software revenue models). Thereby, it formed the basis for answering the first two sub-research questions; 1) "What are revenue models for software components?" and 2) "Which advantages and disadvantages can be identified for these revenue models from the perspective of a software component developer?"

Phase 2. *Exploratory interviews*

In addition to the literature review, an exploratory interview study was conducted to 1) gain a deeper understanding of the empirical context of the study – software component developers in Identity and Access Management -, 2) check in an early stage if the made assumption which was made in this study holds (i.e. the revenue model choice is influenced by the overall business model), 3) see if the revenue models which were identified in the literature review for software component developers in IAM were really used in practice and 4) select suitable case companies for a more in-depth inquiry.

Phase 3. *Conceptual framework*

In the literature review the Business Model Canvas of Osterwalder and Pigneur (2010) was identified as a useful framework to analyse the business model elements which impact the

revenue model choice. Through exploratory interviews, it was indeed observed that the business model impacts the revenue model choice of software component developers. Based on these findings a conceptual framework was developed.

Phase 4. *Case studies*

In phase 4, case studies were conducted to answer the third sub-research question, which states as follows; "*What are the most critical business model element(s) which impact(s) the viability of revenue models for software components?"* The case study approach was chosen because it is well established in academics and is a preferred way of investigating real-life phenomena (Tsang, 2014). Moreover, according to Eisenhardt and Graebner (2007), case studies provided many answers to management problems. For these reasons, and due to the lack of prior research, the case study approach seemed to be an appropriate way to investigate this research topic.

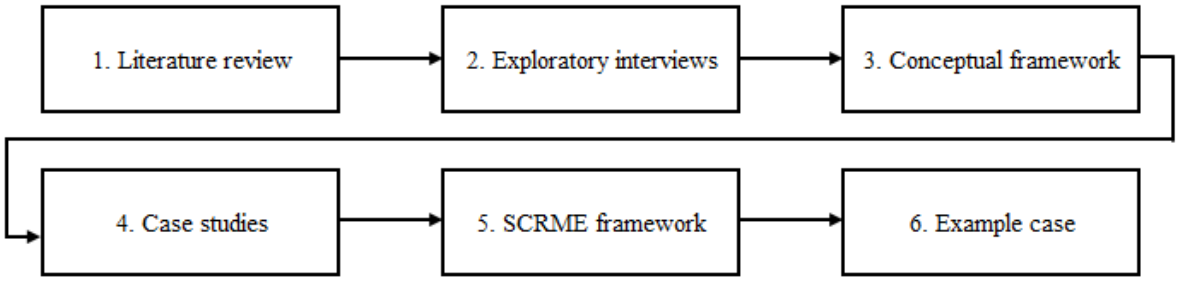
Phase 5. *Software Component Revenue Model Evaluation framework*

Based on the literature review, exploratory interviews and through case studies, the Software Component Revenue Model Evaluation framework (SCRME framework) was designed (see chapter 6.4).

Phase 6*. Example case: applying the SCRME framework*

To illustrate the Software Component Revenue Model Evaluation framework and how it may be used, an example case is presented based on a software component developer which delivers a solution for Identity and Access Management (chapter 7).

Figure 5: Overview research process

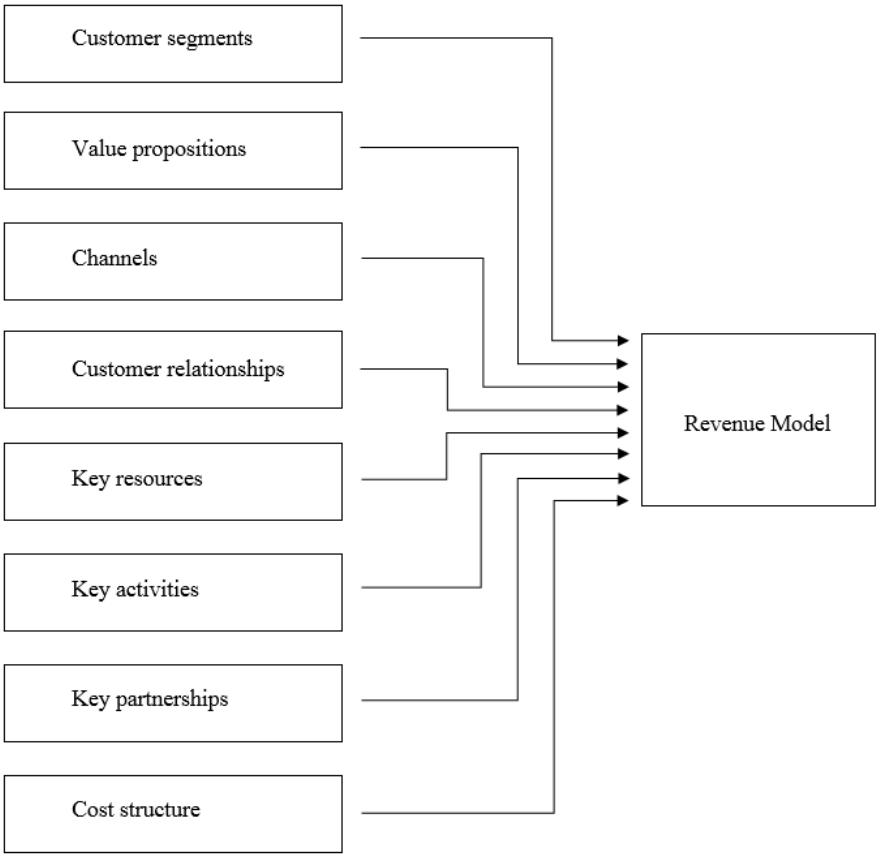


3.2 Conceptual framework

In this study the assumption is made that the revenue model choice is influenced by the overall business model. During the literature review, the Business Model Canvas of Osterwalder and Pigneur (2010) was identified as a useful tool for analysing the business model elements which impact the revenue model choice of software component developers. Based on the Business Model Canvas of Osterwalder and Pigneur (2010), a conceptual framework was established for analysing revenue models in IAM. As can be seen, the conceptual framework illustrates the business model elements and their interconnectedness with the revenue model (figure 6).

As the arrows indicate, the revenue model in this situation is the dependent variable which might be influenced by one or more business model elements (i.e. customer segments, value propositions, channels, customer relationships, key resources, key partnerships and/or cost structure). One has to take into account that there are no scores assigned to these relationship and it is not examined whether there is a causal relationships between the dependent (i.e. revenue model) and independent (e.g. customer segments) variables.

Figure 6: Conceptual framework



3.2 Data collection

3.2.1 Exploratory interviews

In the first part of the empirical analysis, an initial exploratory interview study was conducted to gain a deeper understanding of the empirical context of the study – software component developers in Identity and Access Management. Besides, conducting exploratory interviews, helped in an early stage to confirm that the made assumption in this research (i.e. revenue model choice is influenced by the overall business model) holds. Besides, during the interviews, it was confirmed that the revenue models which were identified in the literature review as applicable revenue models for software components in IAM are really used in practice. In addition, by creating a deeper understanding of the empirical context of the study, it helped to select suitable case companies for a more in-depth inquiry. To select companies for the exploratory interview study, the following criteria was used: the company had to provide a software component solution for Identity and Access Management.

Interviews as a research technique was chosen because it provides richness of data and an interview allows the interviewee to speak freely on a topic, so the interviewer gets new ideas and insights he otherwise would not have thought of (Galvin, 2015). Therefore, interviews are one of the most important data gathering tools in qualitative research (Myers & Newman, 2007). There are different types of interviews, e.g. structured interviews, unstructured or semi-structured interviews, group interviews etc. (Myers & Newman, 2007). Structured interviews are complete scripts which are prepared beforehand which leaves no room for improvisation, therefore this type of interviews is often used in surveys. Whereas, semi-structured interviews leave room for the interviewee to improvise and ask follow up questions. Semi-structured interviews are usually open-ended questions, with follow up questions emerging from the answer (DiCicco-Bloom & Crabtree, 2006). There are also group interviews, these are interviews with two or more people, which can be structured or semi-structured. In this research, semi-structured interviews were conducted because this gives the opportunity for the interviewer to explore particular responses further. Moreover, they are used the most in qualitative research (Myers & Newman, 2007).

A key approach in limiting bias, is to interview highly knowledgeable informants who view the focal phenomena from diverse disciplines (Eisenhardt & Graebner, 2007). Therefore, gatekeepers in the interviewed companies were contacted, who recommended individuals who were particularly knowledgeable about the research topic. Per company, one interview was conducted. This included two companies in the Netherlands and one in the United States. For

this reason, interviews were held in Dutch and in English. Although preferably held at the interviewee's office, for logistical reason one interview was conducted by telephone. To maintain the level of accuracy and richness of the data, the interviews were audio-recorded. Issues which could influence the quality of the recorded interviews, like excessive background noise, was prevented.

The exploratory interviews were guided by a semi-structured topic list (see appendix I) that enabled follow-up questions. Since, one of the goals of the exploratory interview study was to confirm the made assumption in this research. The questions were mainly business model related. Subsequently, the focus was on revenue models and the (business model) motivations behind choosing a specific revenue model. In this way, the researcher could analyse whether the business model influenced the revenue model choice of the companies. The interviews ended with a snowball sampling technique, which was used by the researcher, in order to identify potential case companies.

3.2.2 Case studies

In literature single and multiple case studies are distinguished (Yin, 2003). Single case studies are usually chosen in order to explore a significant phenomenon under rare or extreme circumstances (Eisenhardt & Graebner, 2007). Whereas, multiple case studies are often conducted to generalize findings (Yin, 2003). In this research, a multiple case study approach was used, because with multiple case studies the theory is better grounded and more accurate (Eisenhardt & Graebner, 2007). To select case companies, the following criteria was used: the company had to provide a software component solution for Identity and Access Management. Eventually, three case studies were conducted. To ascertain validity and reliability multiple sources of information were used to gather data for each case. Available data such as company websites and documents (e.g. presentations) were consulted. In addition, interviews were conducted, which were the main form of data collection. Interviews as a research technique was chosen for the same reasons as the exploratory interview study. Also, during the case studies, semi-structured interviews were conducted.

Again, to limit bias, highly knowledgeable informants were interviewed who viewed the focal phenomena from diverse disciplines (Eisenhardt & Graebner, 2007). Therefore, gatekeepers in the case companies were contacted, who recommended individuals who were particularly knowledgeable about the research topic. Per case, one interview was conducted. This included two companies in the Netherlands and one in Luxemburg. Therefore, interviews were held in Dutch and in English. For logistical reasons, the interviews were conducted by

telephone or via Skype. To maintain the level of accuracy and richness of the data, the interviews were audio-recorded. Issues which could influence the quality of the recorded interviews was prevented.

The semi-structured interviews were guided by a semi-structured topic list (see appendix II) that enabled follow-up questions. In each case study, the emphasis was on one specific software component solution within the case company's portfolio. The interviews started with general questions about the case company, the interviewee's position and a specific product within the case company's portfolio that was chosen. Subsequently, since the aim was to identify the most critical business model elements which impact the viability of revenue models for software components, the questions were business model related. Next, it was examined how the business model elements influenced the revenue model choice of software component developers. Lastly, the four revenue models (one-time-charge, subscription-based, usage-based and freemium), which were identified in the literature review as appropriate revenue models for software components, were discussed.

3.3 Data analysis

Data analysis is the heart of building theory from case studies (Eisenhardt, 1989). Qualitative data analysis is defined by Bogdan and Biklen (1982, p. 145) as follows; "*working with data, organizing it, breaking it into manageable units, synthesizing it, searching for patterns, discovering what is important and what is to be learned, and deciding what you will tell others"*. The challenge of qualitative data analysis is to place the raw data into logical, meaningful categories in order to examine them in a holistic fashion and subsequently to find a way to communicate this interpretation to others (Hoepfl, 1997).

The first step of analysing the data starts with identification of themes emerging from the raw data, also referred to as "open coding". During this process conceptual categories must be identified by the research to the perceived phenomena. The goal of this step is to form a preliminary framework for analysis. The next step is to perform an "audit trial", which is a scheme for identifying the data according to a person or context. The third step is called "axial coding". This step involves combining and comparing, the categories identified in open coding, in new ways in order to see the big picture, which helps to acquire new understanding of the phenomenon of interest. During this process, the researcher is also responsible for shaping the conceptual model. Finally, the researcher translates the conceptual model into a story that

resembles reality closely. Although the phases are discussed sequentially, in practice they may occur simultaneously and repeatedly.

The data analysis started with a within-case analysis. With the within-case analyses, the focus was to concentrate on the individual case companies. First, the recorded interviews were transcribed. Next, the interviews were summarized. When summarizing the interviews, it was necessary to go back and forth between data in order to develop a cohesive story. The summarized interviews were supplemented with information from additional sources to create a more complete picture. Subsequently, the basis for the data analysis was formed.

The next step was conducting a cross-case analysis. The first aim was to identify the most critical business model elements which influenced the revenue model choice. As was explicitly asked during the interviews, which business model building block(s) influenced the revenue model choice, evidence was found that various business model building blocks influenced the revenue model choice of the case companies. When a specific business model building block influenced one case company's revenue model, all other cases were scanned whether the business model building block influenced their revenue models as well. By the use of this approach, it was possible to identify differences and similarities between the case companies. Next, the focus was on the four revenue models which were identified in the literature review as possible appropriate revenue models for software components. To be more specific, the researcher aimed to map out the business model elements which impacted the viability of each revenue model. If a particular revenue model (one of the four revenue models) was not used by the case company, the reasons behind the absence were investigated.

4. Data analysis

In this chapter the findings of the exploratory interview study (4.1) is discussed. Subsequently, the within-case analysis is presented (4.2). Lastly, a cross-case analysis (4.3) is conducted.

4.1 Exploratory interviews

The exploratory interview study was conducted with three companies which are active in the software component business for IAM. The interviewed companies include large software component developers who are active in this business for over 20 years with a worldwide presence to a start-up company. In table 7 an overview of the sample size is given.

Table 7: Overview exploratory interviews

	Company A	**Company B**	**Company C**
Years of founding	1993	2013	1998
Location	United States	Netherlands	Netherlands
Interviewee's Position	VP Marketing	Director	COO
Type of interview	by telephone	face-to-face	face-to-face
Length of interview	44 minutes	68 minutes	40 minutes
Product	user-authentication	user-authentication	user-authentication

The *first objective* of the exploratory interview study was to gain a deeper understanding of the empirical context of the study, namely; software component developers in IAM. During the interviews, the business models and revenue models involved in this industry were discussed which helped the researcher to create a better understanding of the software component industry. Also, the interviews confirmed, as assumed, that the Identity and Access Management industry was a good focus to study software components. Namely, Identity and Access Management solutions are designed to be integrated in other software applications or systems. Moreover, these solutions are not valued by the end customer as a standalone application. As this holds with our definition of a software components, namely "*A software component is an independent and reusable computer program that is integrated into a larger software-based solution and is accessible through specified interfaces. Moreover, it is subject to composition by third parties and it is not valued by the end customer as a standalone application*", it can be concluded that it is justified to refer to IAM solutions as software components.

The *second objective* was to investigate whether the made assumption in this research (i.e. revenue model is influenced by the overall business model) holds. Therefore, firstly, the business model of the interviewed companies were discussed. Subsequently, the attention was drawn to the influence of the business model elements which influence the revenue model choice. During the interviews, it was observed that there was a relationship between the business model and revenue model choice. For example, the interviewee from company A stated that the segmented target group influences the revenue model choice. For instance, it was mentioned that large organizations usually request a perpetual license. In contrast, a subscription-based model was usually preferred by small and medium organizations. Whereas, company C mentioned that channels also influence the revenue model choice. For example, when the customer request an on-demand solution, the subscription-revenue model is offered. In contrast, when an on-premise solution is requested, the one-time-charge revenue model is used by the company. Based on these findings, evidence for justifying the made assumption was found.

The *third objective* was to investigate whether the identified revenue models for software component developers during the literature review were really used in practice. At least one of the following revenue models were identified within the interviewees companies; one-time-charge, subscription and the freemium revenue model. The usage-based revenue model was not used by the case companies. The reason is that, a usage-based revenue model is usually provided when a SaaS solution is provided. However, according to the interviewees, in

the IAM software component industry SaaS solutions are not feasible due to security reasons. For example, company B mentioned "*our customers do not want to rely on public cloud services*". The other revenue models; one-time-charge, subscription and freemium, were identified within the interviewees' companies'. The one-time-charge revenue model and the subscription revenue models were used by all three companies. Whereas, the freemium revenue model was used by one company. Despite several revenue models are used by the companies, the one-time-charge revenue model is the most popular. This revenue model is usually used in combination with an additional "maintenance and updates" plan. Other revenue models were not identified within the interviewees' companies'. Based on the findings, the third objective of the exploratory interview study was achieved.

The *fourth* and last objective of the exploratory interview study was to help the researcher to select suitable case studies for a more in-depth inquiry. This objective was also achieved. Namely, the interviews did not only helped the researcher to create a better understanding of the empirical context of this study, also one interviewee mentioned a platform named "Platform Identity Management Nederland" (PIMN) which helped the researcher to locate suitable software component developers in IAM. More specifically, consulting PIMN helped the researcher to gain a list of software component developers which develop solutions for IAM.

4.2 Within-case analysis

In this section, the case companies are analysed individually (see table 8 for an overview of the case companies). Each case study is described as follows, firstly, it starts with a brief overview of the company and the software product that is discussed. Subsequently, the business model of that particular product is presented. Next, attention is given to revenue models.

Table 8: Overview case companies

	Company A	**Company B**	**Company C**
Years of founding	2005	2010	2005
Location	Netherlands	Luxemburg	Netherlands
Interviewee's position	Marketing & sales manager	CEO/CTO	CEO
Type of interview	via Skype	via Skype	via telephone
Length of interview	55 minutes	50 minutes	50 minutes
Product	multi-factor authentication	user-authentication	single sign-on
Customers	4000+	3000+	50+
Target market	government, healthcare, financial institutions, small and medium businesses and industrial companies	government, healthcare, financial institutions, industrial companies and small and medium businesses	governmental organizations
Revenue model	one-time-charge, subscription and time-limited-freemium model	one-time-charge, subscription and support-and-feature-limited-freemium model	subscription and support-limited freemium
Additional data sources	company website, company documents	company website, company documents	company website, company documents

4.1.1 Company A

Company A is specialized in adaptive multi-factor authentication, improving enterprise security and productivity. Their software authenticates users through their mobile devices. In this way, it helps IT managers address evolving business needs with mobile security and cloud applications by dynamically authenticating users based on geo-location and login behavior patterns. The multi-factor authentication software looks at multiple factors surrounding each login. Factors such as geolocation, network-IP and the type of system being accessed are looked at. By using these factors, a context is created that helps determining the level of trust and whether a user should be authenticated or blocked.

Business Model

Company A has thousands of customers world-wide. They have a very diverse customer base, which includes customers in sectors such as, government, healthcare, education, small and medium businesses and financial institutions. More or less, they target businesses of every size who have a need to protect their IT-applications in a secure, flexible and convenient way. Dependent on the security policy of the customer, the product can be provided as an on-premise or an on-demand solution. Regarding the on-demand solution, the software is not a typical SaaS solution. The reason is that, according to the interviewee, a typical SaaS solution is not feasible, because customers do not want to have a copy of their data hosted in the cloud. Moreover, he states that *"having a copy of this kind of data in the cloud is a direct security-breach"*. Therefore, the company provides on-demand solutions via a private cloud. Although the company offers two solutions, customers usually request on on-premise solution (in 80% of the cases) due to security-reasons. Irrespective of the provided solution, the software is easy to integrate and customers are usually ready to go within one hour. In case customers need help with the implementation, they can hire consultants from Company A for an additional fee. Besides selling the software directly to customers, Company A also works together with resellers. Resellers get a certain percentage of the sale. These resellers have deep and specialized knowledge of Company B's product, therefore they are considered by Company A as a key resource. Besides, the intellectual property of the software and the employees of Company A are considered as a key resource as well. The company's main activity, besides marketing and sales, is to constantly develop and update their product. An overview of the whole business model of Company A is presented in figure 7.

Figure 7: Business Model Canvas of Company A

<table>
<tr><td rowspan="2">Key Partners

- resellers</td><td>Key Activities

- development

- maintenance and support

- marketing</td><td rowspan="2">Value proposition

Confirm identity of users</td><td>Customer Relationships

- Strong relationships (high service level)</td><td rowspan="2">Customer Segments

- government

- healthcare

- financial institutions

- small and medium businesses

- industrial companies</td></tr>
<tr><td>Key Resources

- intellectual property

- highly trained resellers

- employees</td><td>Channels

- direct

- indirect (via partners)

In both cases the customer can request two solutions: 1) on-premise or 2) on-demand</td></tr>
<tr><td colspan="3">Cost structure

- training resellers

- development

- marketing and sales</td><td colspan="2">Revenue streams

- licensing-model (per user)

- monthly based subscriptions (per user)

- freemium (30 day free trial)</td></tr>
</table>

Revenue models

Company A has a hybrid revenue model which means they have several revenue models. The revenue models are 1) Term licensing, 2) Subscriptions and 3) Freemium. Each revenue model will be discussed individually.

One-time-charge

A customer can buy a perpetual license, which gives him the rights to use the software for ever. However, in order to get the newest updates, maintenance and support the customer has to get an additional "software assurance" subscription which is terminable on yearly basis. The one-time-charge revenue model is offered because Company A target large organizations. Large organizations usually prefer such a perpetual license for two main reasons. First of all, when a perpetual license is acquired it involves a relatively high initial fee, however on the long-term the total costs are less compared to other models (e.g. subscription). Second, large

organizations such as banks, do not want to have their data stored somewhere in the cloud. By buying such a perpetual license, customers can implement the software within their own secured IT-environment (on-premise). The main advantage of this revenue model from Company A's viewpoint, is that this licensing model is a one-time-sale, which helps to recoup development costs very quick. Moreover, such a one-time-sale is very easy, because usually after sales services do not have to be provided.

Subscription

With the advent of cloud-models, Company A has seen a changing market demand from licensing-models to monthly subscriptions in the last two years. The company offers subscriptions which are user-based and terminable on yearly basis. In the subscription-model, customers only pay for the months they really use the software. So, it is more or less a combination between a regular subscription and the usage-based revenue model (where customers pay per usage). The subscription based revenue model is only possible in combination with the on-demand solution.

The subscription revenue model is offered because, besides large organizations, the company targets small and medium businesses who do not have the resources to buy a perpetual license. Also, small sized organization do not always have their own dedicated IT-infrastructure. Therefore, the subscription is sold in combination with an on-demand solution (private cloud). So, the subscription-based revenue model diversifies the company's customer base. Another benefit of this revenue model is that is causes recurring revenues. Since, the company is actively developing their product, this kind of revenue is very important. Moreover, in the long-term, customers pay up to 4 to 5 times the amount of they would have paid if they chose to license the software. Besides, since this plan is very flexible, it also attracts customers who want to use the software on a project basis, for example for a project of 6 months. However, there are also disadvantages. There is no initial fee involved, so when the software is implemented and the customers decide to not use the software, the company will not earn any money (recurring revenues are not constant). Moreover, if a customer only uses it for a year, the total revenue is less than when a customer would have gotten a license.

Freemium

The freemium-model is also used by Company A. They offer a 30-day free trial for customers to test the software. The free trial is offered for several reasons, first of all, Company A is convinced that customers who get a free trial will upgrade to a payed subscription/license.

Moreover, the cost for offering a 30-day free trial is negligible, because the software is very easy to implement for customers and therefore usually no assistance from Company A is needed. The main benefit of the freemium model is that it lowers the barrier for potential customers to use the product. However, since the software is easy to implement, customer usually do not always consult Company A for advice. Therefore, in some cases, customers do not exploit the full potential of the software.

Usage-based

The usage-based revenue model is not used by Company A. The main reason is because a SaaS solution is not provided by the case company. This is because, SaaS solutions are not feasible in the software component business for Identity and Access Management. The reason is that, their customers do not want to have their data stored in a public cloud.

4.1.2 Company B

Company B provides security solutions designed for modern enterprise technologies. To be more specific, they are specialized in next-generation user authentication. Their solutions are used by thousands of companies world-wide in all businesses, ranging from companies in IT, financial, government to healthcare. Their customers include companies in the fortune 100 as well. The goal of Company B is to ensure that businesses requiring IT-security gets cost-effective leading technology into their infrastructure with a professional service. In this case study, the revenue model of a product within their portfolio is analyzed which makes single-factor and multi-factor authentication possible.

Business Model

Company B has customers around the world. Their solution is used by thousands of customers in more than 40 countries. They do not target a specific customer segment, because their solution is suitable for every business who wants to secure corporate access. The software of Company B is provided as an on-premise solution. The main reason is, because customer generally do not want to rely on cloud services for security reasons. For this reason, an on-premise solution is provided, so customers can integrate the software within their own secured IT-environment. Since the software is very easy to integrate and in most cases is implemented in just one hour, usually support from Company B is not asked. However, in case support is needed, the support is directly done by R&D. Since, support is done by R&D, the provided support is very high (technical knowhow), but also very costly. The biggest expense is product development. Namely, the company spends a lot of their budget on product development in order to release new versions, which is done internally by R&D. Therefore, the employees of Company B are seen as a key-resource. Another key-resource, besides the software itself, is the network that Company B has. They have very good contacts within their local market. For example, they have good contacts with the ministry of economics and large banks within their home country. These contacts with the local market helps them a lot for the outside market. Aside from these networks, they also work with partners who resell their software. For example, they just started a partnership with a large US company, which has a very large presence in the States. An overview of the business model of Company B is presented in figure 8.

Figure 8: Business Model Canvas of Company B

<table>
<tr><td rowspan="2">Key Partners

- resellers

- government</td><td>Key Activities

- marketing

- product development</td><td rowspan="2">Value proposition

- securing identities</td><td rowspan="2">Customer Relationships

- very strong</td><td rowspan="2">Customer Segments

- Mass markets

- customers with less than 40 users

- customers with more than 40 user</td></tr>
<tr><td>Key Resources

- product (software)

- employees

- network</td></tr>
<tr><td colspan="3">Cost structure

- product development
- employees</td><td colspan="2">Revenue streams

- licensing model (per user)

- subscription model (per user)

- freemium (free up to 40 users).</td></tr>
</table>

Revenue model

Company B uses several revenue models, which includes: 1) Freemium, 2) Licensing and 3) Subscriptions. Each revenue model is discussed individually.

Freemium

The software is free for customers who have less than 40 users. If these customers face any issues with the software, they can consult open forums. However, if there is an issue they cannot solve, Company B gives support at an additional fee. So, the support-limited freemium model is used. The freemium model is used for several reasons. First of all, the software is relatively easy to integrate and the costs for Company B providing this solution is nearly negligible. Secondly, the product they offer used to be completely open-source and free at the beginning. Besides, the company believes that there should be a solution for companies who cannot pay for security. The company believes that companies with less than 40 users, might not have enough resources to pay for security. Therefore, they still offer the product for free.

Licensing

As described, the product used to be completely open source and free at the beginning. However, over time, when companies started to use the software, customers wanted something more professional. At first, the company tried to keep the software for free and asked only additional fees for support and integration services. This business model did not work, because it was very difficult to recoup development costs. For this reason, a second revenue model was introduced; licensing. Customers buy a license for each user that uses the software. The licensing-model is introduced, because large organizations usually demand a perpetual license. When a license is acquired, the customer can use the product for a life time. However, if customers want to receive the newest updates and support, they have to purchase an additional annual maintenance/support contract per user.

Subscriptions

Besides selling licenses, Company B also sells subscriptions. As in the licensing model, customers pay per user. The subscription is terminable on yearly basis, which means that customers have to pay each year to use the software, otherwise they lose the right of using the software. With the subscription model, maintenance, software updates and support is included. Since, in this plan software updates are involved, customers expect the product to be up-to-date at all time. Therefore, continually developing and improving the product is very important.

The subscription model is introduced for two main reasons. First of all, there has been a growing demand for this kind of models in the last two years. Second, buying a perpetual license involves a relatively high fee, and not all the customer have the resources to acquire such a license. Therefore, since no initial fees are involved with the subscription-model, the software also becomes interesting for small-medium sized organizations with less resources. So, this revenue model diversifies the customer base for Company A.

A big advantage, mentioned by the interviewee, is that companies usually forecast their expenses for three years. Usually, in the first three years, the total costs of a subscription is less than the total costs of a license. For this reason, when a subscription-model is offered, the barrier for customers to use the software is lowered immensely. Another big advantage of the subscription-model is that it is the most profitable in the long-term, since this revenue model causes recurring revenues as long as customers use it. According to the interviewee, the total revenue of a subscription-based model, if a customer used the software for a life-time, will be 5 times as high as when the software would be sold as a perpetual license. Also, the interviewee

mentioned that, the subscription-based revenue model works especially well when a company is financially stable to finance software development, because in the beginning revenues will be low. This is because, development costs are not recouped quickly. Since software development is very important, the development process needs to be financed in other ways.

Usage-based

The usage-based revenue model is not used by the company. As with Company A, this is because a SaaS solution is not provided by the case company.

4.1.3 Company C

Company C is specialized in software component solutions for Identity and Access Management. The business has been active in this field for over 10 years. It also won several awards in this field. Company C provides an Identity and Access Management solution which manages identity and secures access to IT-applications. It provides secure authentication and authorization services. Their solution is a software component which can be integrated in other applications. Moreover, it includes Single Sign-On, which is a process that permits a user to enter a username and a password in order to access multiple application at once.

Business Model

Company C is located in the Netherlands. They are active in the B2B (business to business) market and mainly provide solutions for governmental organizations. The reason is because in an early stage they focused on a Dutch identity and management platform called DigiD. This platform is used by government agencies in the Netherlands to verify the identity of Dutch residents for access to digital government services. Besides governmental organizations, the company also sells the software to other software vendors, who integrate the software solution of Company C into their own application to subsequently sell as a total package. These software vendors are more seen as partners rather than customers. Remarkably, the company does not do any kind of active sales or marketing to acquire new clients.

The software solution Company C provides is on-premise. They do not provide a SaaS solution, because there are several security reasons which withholds them to provide such a solution. Moreover, their customers do not want to have their data stored somewhere in the cloud, but rather in their own datacenter. Since their solution is very unique, in most cases their expertise is required when the solution has to be implemented at the client's side. Depending on the clients IT-infrastructure the implementation is usually done within two days. Since they offer a software solution, continuous software development is very important. They actively bring out new patches which improves the security of the software. However, new functionalities are only brought to market if customers demand it. For example, when a customer wants a new functionality they develop this for an additional fee. When the new functionality is developed, it becomes free for all their current customers. An overview of the business model of Company C is presented in figure 9.

Figure 9: Business Model Canvas of Company C

<table>
<tr><td rowspan="2">Key Partners

- software developers</td><td>Key Activities

- product development

- implementing the product

- keeping up to date regarding the latest trends in digital security</td><td rowspan="2">Value proposition

-Providing secure authentication</td><td>Customer Relationships

- strong relationships (paying customers)

- weak/automated (not paying customers)</td><td rowspan="2">Customer Segments

- governmental organizations</td></tr>
<tr><td>Key Resources

- product knowledge

- employees

- partners</td><td>Channels

- direct

The software is an on-premise solution</td></tr>
<tr><td colspan="2">Cost structure

- employees

- product development</td><td colspan="3">Revenue streams

- freemium (support limited)

- subscription (maintenance, support & updates)</td></tr>
</table>

Revenue model

One revenue model Company C uses is the freemium-model. Customers of Company C do not pay at all for the software itself. They can download the full product for free and install the software themselves. However, if they need any expert assistance from Company C they have to pay per hour for support. Subsequently, company C helps them with the implementation which usually costs around two working days. After implementation, a customer can get a support and maintenance contract which is terminable on a yearly basis, against is a fixed amount (regardless of the amount of user). This gives the customer the right to get updates, support and maintenance. If the customers chooses not to get such a support and maintenance subscription, he loses the right to receive the newest updates and support from Company C. So, besides the freemium-model, Company C also introduced the subscription-model.

The freemium model is used, because the costs for providing the software for free (without support) is nearly negligible. Moreover, since the software solution they offers is so unique, Company C is confident that each customer will need their support eventually. Besides, since they offer a security solution, customers always want to have the newest updates in order to stay up to date. At the end of the day, customers get the product to secure their data and it would make no sense if they would invest time and money in implementing a solution which gets outdated very quickly (without updates).

One-time-charge

The one-time-charge revenue model is not used by Company C. The reason is that, their software is developed on the basis of open-source software (source code is free). Therefore, to keep the open-source mindset, they do not sell perpetual licenses. Instead, users/customers can download the software for free. However, if users/customers do want additional services, such as receiving maintenance or support, they have to pay a fee for it.

Usage-based

The usage-based revenue model is not used at all, this has several reasons. First of all, a SaaS solution is not provided by the case company, which makes it very difficult to monitor the usage. Moreover, the interviewee mentioned that, it is very difficult to implement a usage-based mechanism. Besides the technical part of developing such a model, it is quite difficult to determine when the software is actually used (e.g. is it in terms of attempts or succeeded logins?). Moreover, the software solution is usually integrated on-premise, in a secured IT-environment which they cannot access. This means, in most cases the usage cannot be monitored at all. However, the interviewee mentioned that, if SaaS models become accepted within Identity and Access Management, the company might use such a revenue model

4.3 Cross-case analysis

In this sub-section, firstly, key findings are presented. Subsequently, per revenue model, the key business model building blocks which impacts the viability of the revenue model are discussed.

4.2.1 Highlights from findings

As noted in the literature review, software (component) vendors have great flexibility in constructing their revenue model. This is immediately noticed when we compare the revenue models of the case companies. Namely, all the case companies have hybrid revenue models, which means they use several revenue models. Besides, using multiple revenue models, case companies also make, per revenue model, a distinction between; user-based vs. not-user based pricing. Two of the three case companies based its revenue model on user-basis, whereas one company's revenue model is not user-based (one total price). When we look at the revenue models, three revenue models; *one-time-charge*, *subscription* and *freemium*, which were identified in the literature review as appropriate revenue models for software components, are used by the case companies. The one-time-charge revenue model is the most popular, however the case companies mentioned that the demand for subscriptions has been on the rise for the last two years. In addition, the consulting revenue model is used as well. With the consulting revenue model, the consultant (software component developer) charges an hourly rate for providing advice to a customer.

Interestingly, the usage-based revenue model is not used at all by the case companies. The usage-based revenue model is not used for several reasons, first of all, a usage-based revenue model is usually provided when a SaaS solution is provided. However, in the Identity and Access Management software component business, SaaS solutions are rarely offered for security reasons. The reason is that, customers want a high level of security and using a public cloud to store sensitive data is not found secure enough. It was even mentioned by a case company that *"storing this kind of sensitive data in a public cloud would be an outright security breach".* Besides not providing a SaaS solution from a security-viewpoint, the case companies also mentioned that it is also very difficult to implement a usage-based mechanism. Thirdly, even if the case companies succeeded in building a usage-based mechanism (without providing a SaaS solution), it would still be difficult to monitor the usage because usually the software component is installed on-premise at the client's side in a secured IT-environment (in almost 80% of the cases, the software is provided as an on-premise solution). Despite SaaS solution are not being used, on-demand services are still provided. These on-demand solutions are

offered via a private cloud, which minimizes the security concerns. Since, the usage-based revenue model is not found appropriate, for the rest of this research, it is left out of scope.

4.2.2 Analyzing individual revenue models

In this sub-section, the most critical business model elements which impacts the viability per revenue model are described (see appendices III-VII for the cross-case analysis). The revenue-models: one-time-charge, subscription and freemium are discussed individually.

The viability of the one-time-charge revenue model is impacted by two main business model elements. First of all, *customer segments* is a critical business model elements which influences the viability of the one-time-charge revenue model. Namely, observed from the case studies, when large organizations are targeted the viability of this revenue model increases. The reason is that, large organizations usually prefer a perpetual license. Moreover, acquiring such a license involves high fees for the customer and large organizations usually have the right resources to acquire such a license. Another important business model element which impacts the viability of the one-time-charge revenue model is *channels*. To be more specific, channels in terms of providing the software as an on-premise vs. on-demand solutions. This is because large organizations usually want to implement the software in their own secured IT-environment (on-premise). Therefore, when the one-time-charge revenue model is used, providing an on-premise solution is a key determinant for the success of the one-time-charge revenue model. So, targeting large organizations and providing the software as an on-premise solution are important elements which impact the viability of the one-time-charge revenue model.

The viability of the subscription revenue model is also influenced by several key business model elements. First of all, where large organizations usually demand a perpetual license, a subscription based model is usually demand by small and medium sized organizations. The reason is that, 1) small and medium organization do not always have the resources to acquire a perpetual, which involves high initial fees, 2) small and medium organizations do not always have their own on-premise IT-infrastructure, but make use of third-party cloud services. As can be seen, with this revenue model *customer segments* again plays a crucial role. Also, *channels* plays a crucial role, namely when the software is provided as an on-demand solution, it has a positive impact on the viability of the subscription revenue model. As with the case companies, the subscription-plan includes support, maintenance and updates. Therefore, *key activities* in terms of providing support and developing the software is very

important. Since, providing (personal) support and developing the software is quite expensive, *key resources* also play a crucial role regarding the viability of the subscription-based revenue model. The reason is that, when a subscription-model is used, the earnings in the beginning for the component developer is relatively low. This means, the development costs are not recouped. Therefore, in order to finance software development, key resources, in monetary terms, is very important. For example, when a company is in a start-up phase, and it needs to recoup the development costs quickly in order to finance new developments, then the subscription-based revenue model might be not that appropriate. So, *customer segments, channels, key activities and key resources* are important business model elements which impact the viability of the subscription-based revenue model.

The viability of the freemium revenue model is mainly impacted by *customer relationships* and the business model element *key resources*. Namely, as the case companies mentioned, low marginal costs are essential to make this model work since no revenue is generated directly. Therefore, it is important that the software is relatively easy to integrate so no expert assistance is needed. Also, providing non-personal (automated) customer service is very important. The reason is that, providing personal customer support is quite expensive and in most cases costs a lot of time, which usually is not affordable. *Key resources* also impacts the viability of the freemium revenue model. The reason is that, in IAM it is important that the software is up-to-date at all time since the component developers offer security solutions. Therefore, software development needs to be done continuously, which costs money. Since a freemium model does not generates revenue directly, it is important the software component is financially stable to fund new developments.

5. *Results*

First, in this chapter, the sub-research questions are answered (5.1 – 5.3). Based on these answers, a framework is presented which helps to evaluate revenue models for software components.

5.1 Revenue models for software components in Identity and Access Management

In extant literature, no papers were identified which discuss revenue models for software components. Therefore, as a basis, to locate revenue models for software components, common revenue models in the software business were mapped out. The uniqueness about software components is that they have to be integrated in other applications to create value. For this reason, to locate appropriate revenue models for software components, the following criteria was used: *the revenue model has to be viable when the software component is integrated in a system/application*. Finally, on the basis of the literature review, four revenue models, i.e. one-time-charge, subscription, usage-based and the freemium-model, were found appropriate revenue models for software components.

However, during the case studies, it was observed that the usage-based revenue models was not used by software component developers in Identity and Access Management. This is because, a usage-based revenue model is usually provided in combination with a SaaS model. However, in the Identity and Access Management software component business, SaaS solution are not used for security reasons. The reason is that, customers want a high level of security and using a public cloud to store sensitive data is not found secure enough. For this reason, the usage-based revenue model was left out of scope for the rest of this study. So, the list with appropriate revenue models was reduced to: one-time-charge, subscriptions and the freemium-model.

5.2 Advantages and disadvantages of revenue models for software components

On the basis of the literature review and through case studies, the advantages and disadvantages of the revenue models; one-time-charge, subscription and the freemium-model were mapped out. The *one-time-charge* has several advantages for the component developer. First of all, a one-time-charge is a closed transaction. So, once it is done, there is nothing for the software

component vendor to keep track of, the revenue is bookable and the customer has his license. In this way, selling a perpetual license helps to recoup development costs in a short time in contrast to other revenue models. Moreover, since high initial fees are involved for the customer switching costs for customers are also increased, in this way it can create a lock-in effect for customers. However, there are also disadvantages which the component developer should take into account. The main disadvantage of the one-time-charge revenue model is that it does not provide a substantial income after initial purchase. Moreover, offering a perpetual license involves high initial fees for the customers which might limit the target group.

When a *subscription revenue model* is used, it has several advantages. First of all, it diversifies the customer base, because no high initial fees are involved. Therefore, the software becomes interesting for small organizations as well. Moreover, since no high initial investments are done, the barrier for potential customers is relatively low. Besides, the subscription-model generates income as long as customers use the software. In general, on the long-term the total revenue of a subscription-based revenue model is up to 4 to 5 times as high compared to selling a perpetual license. In addition, a subscription-based revenue model, causes recurring revenues. Despite the various advantages, software component developers also have to take the disadvantages into account as well. With the subscription-model, no initial fees are involved, so when the software is implemented and the customer stops using the software, let's say after one year, the development costs are not recouped.

When a *freemium revenue mod*el is used, it offers various advantages for the component developer as well. First of all, it lowers the barrier for potential customers to use the software, because no monetary investments are involved. Thereby, it diversifies the customer base since the software becomes interesting from small-startups to large organizations. Moreover, a freemium-model can also create a lock-in effect. Namely, the case companies mentioned that the lock-in effect is not necessarily in monetary-terms, but also in the technology. The freemium model also has its disadvantages. It does not generate any revenue directly, so development costs are not recouped. Besides, there is a chance that free-users are not converted into paying-customers. Moreover, when personal customer service is not involved, usually the customers do not exploit the full potential of the software.

5.3 Business model elements which impact the viability of software component revenue models

First, to answer this question, a business model tool framework had to be chosen to analyze a company's business model. As stated before, the business model concept was used because it is a powerful tool to analyze an organization (Magretta, 2002; Osterwalder & Pigneur, 2010). Therefore, in the literature review several business model frameworks were analyzed (i.e. Chesbrough and Rosenbloom, 2002., Osterwalder and Pigneur, 2010., Hedman and Kalling, 2003.). On the basis of the literature review, it was decided that the Business Model Canvas of Osterwalder and Pigneur (2010) was the best business model tool/framework to use. Subsequently, through case studies, the most critical business model elements which impact the viability of revenue models were mapped out.

The viability of the revenue models; one-time-charge, subscriptions and usage-based, are impacted by, one or more, of the following business model elements; customer segments, channels, customers relationships, key activities and/or key resources. *Customer segments*, in terms of the target market (small-medium vs. large organizations), impact the viability of the one-time-charge and subscription-based revenue model. When large organizations are targeted it increases the viability of the one-time-charge revenue model. In contrast, when small and medium sized organizations are targeted, the viability of the subscription revenue model increases.

Channels, in terms of providing an on-demand vs. on-premise solution, is another important business model element which impacts the viability of the revenue models; one-time-charge and subscription. The on-premise solution is usually demanded by large organizations such as banks. The reason is that, large organizations usually want to implement the software in their own secured IT-environment. Also, when the software is installed on-premise, the one-time-charge revenue model is the most popular. This is because, large organizations usually have resources to buy a perpetual license. In contrast, small-medium organizations usually demand an on-demand solution, because small and medium organizations do not always have their own (on-premise) IT-infrastructure, so the software cannot be installed on-premise.

Customer relationships also play an important role when constructing a revenue model. A distinction should be made by the level of customer support between paying and non-paying customers. This is because, providing personal support is very time consuming and costly.

Therefore, in case of paying-customers, personal customer service is provided, whereas in case of not-paying-customers (freemium-users), customer service is automated. So, for example, when personal support is needed, the freemium revenue model is most likely not an appropriate freemium model.

Moreover, *key resources* are also of importance when constructing a revenue model. Especially financial resources are of importance when constructing the subscription and freemium revenue model. This is because, since the software component developers provide a security solution, the software needs to be up-to-date at all times. Therefore, continuous software development is very important, however software development is very expensive. Since the revenue models subscription and freemium, do not recoup the development costs quickly, financial resources are a key determinant for the viability of these revenue models. When enough financial resources are present to fund software development, the viability of both revenue models will increase. When development costs need to be recouped quickly in order to fund product development, the one-time-charge is more appropriate.

Customer support and software development represent the main key activities of component developers. As discussed above, both business model elements impact the viability of certain revenue models. Therefore, *key activities*, are considered as an important building block which affects the revenue model choice. The other business model building blocks (cost structure and value proposition) are interrelated with the described business model building blocks.

5.4 Presenting the Software Component Revenue Model Evaluation framework

As previous research has shown, the revenue model is interconnected with various elements of a business model (Magretta, 2002; Osterwalder & Pigneur, 2010; Teece, 2010; Timmers, 1998). This study reaffirmed that there is a link between the overall business model and choosing a revenue model. In other words, an organization should construct a revenue model which fits well within the overall business model. However, choosing a revenue model remains quite difficult (Kittlaus & Clough, 2008; Lambrecht et al., 2014). One reason might be the lack of a tool for software component developers in IAM to evaluate revenue models. This void in literature is addressed in this study by developing the SCRME framework (an acronym for Software Component Revenue Model Evaluation), which is presented in figure 10. The SCRME framework presents the most critical business model factors which impact the viability of the revenue models; one-time-charge, subscription and freemium. At the level of practice, the SCRME framework helps software component developers in IAM to evaluate revenue models. Whereas, at a conceptual level, the SCRME framework helps scholars to link the world of business models to the world of revenue models.

The SCRME framework is based on the answers of the sub-questions. Namely, the answers on the sub-questions helped to 1) identify applicable revenue models for software components, 2) map out the advantages and disadvantages per applicable revenue model and 3) map out the most critical business model elements which impact the viability of revenue models for software components. As illustrated in figure 10, in the first column the most critical business model factors which impact the viability of revenue models are presented. These business model factors are based on the most critical business model elements which impact the viability of revenue models for software components in IAM. Subsequently, in the next columns, the revenue models which were identified as appropriate revenue models for software components are listed.

Per business model factor the impact on each revenue model is presented. To show the level of impact, the following scales are used; *probably positive*, - *(no influence/neutral)* and *probably negative*. *Probably positive* indicates that the business model factor has most likely a positive effect on the viability of the revenue model. In other words, the revenue model is seen as an appropriate revenue model for the business model factor in question. *No influence/neutral* indicates that the business model factor does not of nearly impact the viability of the revenue model. Whereas, *probably negative* indicates that the revenue model is most likely not a good fit for the business model factor in question. When using the framework, one has to take into

account that the business model factors are listed without any order of priority or causal relationship. Therefore, one business model factor might indicate that a specific revenue model is probably appropriate, whereas another business model factor indicates that the same revenue model is not appropriate. Also, there is no difference in importance regarding the different business model factors.

To increase the SCRME frameworks' usability, a guideline is defined. The first step is to mark the business model elements which apply for the organization in question. Next, when the impact of a business model factor has a *probably positive* impact on the viability of a revenue model, that particular box should be given a color (e.g. green). In contrast, if a business model factor has a *probably negative* impact on a specific revenue model then, for example, a red color should be given. In this way, the usability of the SCRME framework increases.

Elaborating on the business model factors

The first two business model factors refer to the *size* of the customers. In this study a definition of a *large organizations* neither of a *small-medium sized organization* is given. In the viewpoint of two case companies, organizations such as banks and governmental organizations are referred to as large organizations. Using their definition of a large organization, can create obscurities for users of the SCRME framework. Therefore, this definition is not applied in this study. Another case company, classified a customer as a "large organization" when a customer has more than 40 potential users. This definition is found more applicable when using this framework. So, one might use this definition to distinguish large organizations from small-medium sized organizations. In other words, when a customer has more than 40 potential users, it can be classified as a large organization. Whereas when a customer has less than 40 potential users, it can be classified as a small-medium sized organization.

The third and fourth business model factors refer to the type of solution (on-premise vs on-demand). In this research, on-premise software is referred to software which is installed and runs on the premises (building) of the organization using the software. Whereas, on-demand software is referred to software which is managed and deployed via a cloud computing infrastructure (public as well as a private clouds) and accessed by users over the internet. The fifth and sixth business model factors are regarding the type of customer service. With customer service, non-technical (e.g. providing information) as well as technical services (e.g.

implementing the software) is meant. The last business model factor, as presented, is regarding the necessity of recouping developing costs in order to fund software development.

Figure 10: Software Component Revenue Model Evaluation framework

Business model factors	**Revenue model**		
Factors	One-time-charge	Subscription	Freemium
Customers are large organizations	probably positive	probably negative	-
Customers are small-medium sized organizations	probably negative	probably positive	-
The software is provided on-premise	probably positive	probably negative	-
The software is provided on-demand	probably negative	probably positive	-
Customer service is automated	-	-	probably positive
Customer service is personal	-	probably positive	probably negative
Development costs do **not** have to be recouped quick in order to fund software development	-	probably positive	probably positive

6. *Example case*

In this chapter an example case is presented, which is based on a real company who is active in the software component industry for IAM, to illustrate the SCRME framework and how it may be used.

To illustrate the Software Component Revenue Model Evaluation framework and how it may be used, an example case is presented based on a software component developer which delivers a solution for Identity and Access Management. The case company is chosen, because the company is in their start-up phase and has not defined a definitive revenue model yet. To apply this framework, information regarding the case company business model had to be gained. For this reason, an interview with the CEO of the company was conducted (see appendix VIII). For confidentiality reasons, the concerned company is called company X and the software they provide as software X.

6.1 Background information

The software prevents identity fraud by verifying and reading personal information from passport chips with a smartphone. Identity documents, such as, identity cards and passports contain nowadays a contactless (RFID) chip, which can be used to verify the authenticity of these identity documents. This enables to use modern smartphones, which have NFC-technology, as a tool to verify identity documents. For example, this technology enables police offers to verify identity documents in the street by using their mobile phones. Another example is, the technology can be integrated in mobile apps to use identity cards for second-factor authentication. As these examples illustrate, the software component can be used in various applications and could be interesting for a wide customer base.

However, customers of company X include mainly large and governmental organizations, such as banks and the Dutch police. Small and medium organizations are not targeted, because the software is quite expensive. For example, the software will cost an average customer around €60.000 a year. The high costs are mainly due to the high development costs and the amount of personal support company X has to provide. For example, implementing the software at the client's side takes on average three to four working days.

Moreover, despite software X is standard software, customers usually want the software to be tailored to their specific needs. In this case, the implementation phase takes even more time. Besides, after the software is implemented, customers need to be trained in order to exploit the full potential of the software, which again is quite expensive. The reason is because software X is a very complex piece of software. Another reason that a lot of personal support is provided is due to the lack of manuals on how to use and implement the software. The main reason for the high development costs is because the company offers two types of solution. Customers can choose between an on-premise and on-demand version. The high development costs are mainly funded via side project that the company does. Namely, besides software x, company X generates an income by providing consultancy services. For this reason, the development costs of the software does not have to be recouped quickly in order to fund software development.

6.2 Applying the SCRME framework

In this part the SCRME framework is applied for software X. Per revenue model the business model factors are examined and subsequently is concluded which revenue models are applicable. However, before evaluating the revenue models, the guidelines which were defined for using the SCRME framework are applied.

The first step was to mark the business model factors which apply for company X. The business model factors that apply are underlined. Next, when the impact of a business model factors has a *probably positive* impact on the viability of a revenue model, that particular box is colored green. If a business model factor has a *probably negative* impact on a specific revenue model, then that particular box is colored red. When a business model factor has *no influence* on a revenue model, that particular box in the SCRME framework is left blank. In figure 11, the filled in SCRME framework for company X is presented.

Figure 11: Filled in SCRME framework for software X

Business model factors	**Revenue model**		
	One-time-charge	Subscription	Freemium
Customers are large organizations	probably positive	probably negative	-
Customers are small-medium sized organizations	probably negative	probably positive	-
The software is provided on-premise	probably positive	probably negative	-
The software is provided on-demand	probably negative	probably positive	-
Customer service is automated	-	-	probably positive
Customer service is personal	-	probably positive	probably negative
Development costs do not have to be recouped quick in order to fund software development	-	probably positive	probably positive

One-time-charge

When we look at the one-time-charge revenue model, we see that the two business model factors who have a probably positive effect on the viability of this revenue models, 1) customers are large organizations and 2) the software is provided on-premise, are present. The business model factors who have a negative impact on this revenue model do not apply. For these reasons, the one-time-charge revenue model is seen as an appropriate revenue model for software X.

Subscription

Customers of company X mainly include large organizations. As the SCRME framework illustrates, the business model factor *customers are large organizations*, has probably a negative impact on the viability of the subscription revenue model. Also, the business model factor *customers are small-medium sized organizations* is not present within company X's business model. So, at a first glance, the framework indicates that the subscription revenue model is probably not an applicable revenue model for software X.

However, three other business model factors who have a probably positive effect on the viability of the subscription revenue model are present within company X's business model. Namely, 1) *the software is provided on-demand,* 2) *the development costs do not have to be recouped quickly in order to finance software development* and 3) *customer service is personal*. Since the technical condition (providing an on-demand solution) is met by company X, the subscription revenue model can be used. However, the notion is made that it is not likely that customers will prefer this revenue model.

Usage-based

According to the SCRME framework, when *customer service is personal*, it has a negative impact on the viability of the freemium revenue model. As discussed in the previous section, implementing software X involves a lot of personal support (i.e. implementing the software takes on average three to four working days). In addition, customers of company X need to be trained in order to exploit the full potential of the software. So, per customer, the investments, and thereby the costs, for company X is relatively high. For this reason, the freemium model is not seen as an applicable revenue model for software X.

Conclusion

Based on the SCRME framework, the one-time-charge revenue model is seen as an appropriate revenue model for software X. The subscription revenue model can also be used, however the notion is made that it is not likely that customers will prefer this revenue model. Whereas, the freemium revenue model is seen as an inappropriate revenue model for software X. So, the example case illustrated how revenue model can be analyzed. It also showed that it is a useful tool to evaluate revenue models.

However, the example case also pointed out that the SCRME framework has its shortcomings. Namely, it showed that depending on the business model of a software component developer, the importance of certain business model factors in the SCRME framework may differ. Also, one business model factor can have a positive effect on the viability of a revenue model and another business model factor can have a negative impact on the viability of the same revenue model. Since the SCRME framework does not give an overall score which decides whether a specific revenue model is applicable or not, the SCRME framework can create obscurities for someone who uses the framework. Therefore, it might have been better to make the SCRME framework in the form of a decision tree, which is a tool that shows decisions and their possible consequences. However, due to the time restrictions and the scope of this master thesis, it was not possible to develop such a decision tree. Namely, not enough data could be gathered for developing a decision tree. Also, in this research, it was note examined whether there was a causal relationship between the dependent variable (i.e. revenue model) and independent (e.g. customer segments) variables. Nevertheless, the SCRME framework remains a useful tool which helps software component developers in IAM to make well-considered changes regarding revenue models.

7. Conclusion & discussion

In this chapter a conclusion (7.1) is presented and the findings are discussed (7.2). Next the contribution to science and practice is described (7.3). Lastly, limitations and some directions for further research (7.4) are included.

7.1 Conclusion

The aim of this study was to create a framework which helps software component developers in Identity and Access Management to evaluate revenue models. On the basis of the literature review, four revenue models were identified as applicable revenue models for software components in Identity and Access Management. The revenue models are one-time-charge, usage-based, subscription and freemium. Each revenue model has its own advantages and disadvantages, which were mapped out throughout this research. Next, since the assumption was made that the revenue model choice is influenced by the overall business model, an exploratory interview study was conducted to investigate whether the made assumption could be justified. Through exploratory interviews, it was indeed observed that several business model elements affect the organizations' revenue model choice. Subsequently, through case studies, the list with appropriate revenue models for software components in IAM was reduced to three, namely: one-time-charge, subscription and freemium. In addition, the case study enabled to map out the individual business model elements which affect the viability of the three revenue models. Based on these findings, the following central research question which was applied in this study can be answered:

"How can software component developers in Identity and Access Management evaluate revenue models for exploiting software components?"

The Software Component Revenue Model Evaluation framework, which is presented in figure 11, can be used by software component developers in Identity and Access Management to evaluate revenue models. The SCRME framework shows how key business model factors impact the viability of the revenue models one-time-charge, subscription and freemium. The framework indicates whether the impact of a business model factor on the viability of a revenue model is *probably positive*, *- (no influence/neutral)* or *probably negative*. In this way, the SCRME framework helps software component developers in IAM to make well-considered choices regarding revenue models.

7.2 Discussion

This study corroborated the premise that the overall business model influences the revenue model choice. Namely, empirical evidence was found that various elements of a firm's business model impacts the revenue model choice of software component developers in IAM. So, our results suggests that when choosing a revenue model, the overall business model has to be taken into account. This finding corresponds with previous research, which has shown that a firm's business model is an important element which affects the revenue model choice (Rajala, Nissilä, & Westerlund, 2006). One of the key contribution that this study makes lies in mapping out the most critical business model elements which impacts the viability of several revenue models. Moreover, this study contradicts the statement of Lampson (2004) who claimed in his essay that software components are unlikely to work in future, because he mentioned that there is no business model for it (i.e. too expensive to develop software components) and due to the lack of standard interfaces. However, in contrast, this study showed that software components are very successful and in today's world a market without software components in IAM is even unthinkable.

As a result of this study, the SCRME framework was developed, which helps software component developers in IAM to evaluate revenue models. To develop the SCMRE framework, the Business Model Canvas of Osterwalder and Pigneur (2010) was used. As widely acknowledged, the Business Model Canvas is a very useful tool to analyze an organization (Upward & Jones, 2015). However, this study also identified its major shortcoming. Namely, the Business Model Canvas mainly focusses on endogenous elements, e.g. key resources, key activities and channels. Whereas the exogenous factors, e.g. competition and other environmental factors, are ignored. When evaluating the SCRME framework, another point of critique has to be mentioned. Namely, the SCRME framework shows per business model factor, which revenue models are probably applicable and which not. So, one

business model factor can indicate that a revenue model is probably applicable and another business model factor can indicate that the same revenue model is probably not applicable. Since the SCRME framework does not give an overall score which decides whether a specific revenue model is applicable or not, the SCRME framework can create obscurities for someone who uses the framework. For this reason, it might have been better to make the SCRME framework in the form of a decision tree, which is a tool that shows decisions and their possible consequences. However, this was not possible due to the time restrictions and the scope of this master thesis. Namely, not enough data could be gathered for developing a decision tree. Also, in this research, it was note examined whether there was a causal relationship between the dependent variable (i.e. revenue model) and independent (e.g. customer segments) variables. Nevertheless, the SCRME framework remains a useful tool which helps software component developers in IAM to make well-considered changes regarding revenue models.

Also, this study showed that there are quite a few similarities between the software component industry and the regular software (custom-made) industry. Therefore, since various papers discuss that the technical aspects of developing software components requires a fundamentally different approach than developing custom-made software (Bloor et al., 2007; Lampson, 2004), the question arose whether the business aspects of software components, especially regarding revenue models, also differ fundamentally from custom-made software? Well, a major difference has been observed between both industries in this research. When we look at the applied definition of a software component, "*A software component is an independent and reusable computer program that is integrated into a larger software-based solution and is accessible through specified interfaces. Moreover, it is subject to composition by third parties and it is not valued by the end customer as a standalone application*", it immediately shows the difference between a software component and custom-made software. Namely, a software component has to be integrated in another system or application to create value. Since it has to be integrated in another application or system, the revenue model choice of the software component developer is limited by this. For this reason, several software revenue models were not found applicable for software components (e.g. advertising revenue model). So, certainly, there are differences between both industries. However, this study also observed similarities between both industries. For example, in both industries hybrid revenue models are used. Also, in both industries, several revenue models are combined (e.g. one-time-charge for the license and a subscription revenue model for additional services). Besides, despite the software component industry is usually referred to as the "plug and play" business,

as in the regular software business, implementing the software at the clients' side involves a considerable amount of time and customizations. So, differences between both industries have been identified, which implicate that evaluating revenue models for software components require a different approach than evaluating revenue models for custom-made software. However, since also similarities occur in both markets, software component developers probably do not have to reinvent the wheel again when it comes to revenue models and might learn from best practices in the regular software business, and vice versa.

In addition, when interpreting the results of this study, one key finding has to be interpreted with caution. Namely, based on the case study, SaaS solutions were not found feasible in the software component industry for IAM. Therefore, since the usage-based revenue model is mainly used in combination with a SaaS model, the usage-based revenue model was not found an appropriate revenue model in this study. However, the notion has to be made that the customers of the case companies included mainly large organizations, such as banks and governmental organizations, who do not want to rely on public cloud services from a security viewpoint. So, if other customer segments were targeted by the case companies, e.g. repair-shops, it might be the case that these customers are willing to use SaaS solutions. In that case, the usage-based revenue model would be an applicable revenue model. Also, the notion has to be made that SaaS solutions are becoming more acceptable, so it might be the case that organizations, such as banks, are willing to use SaaS solutions in future.

7.3 Contribution

7.3.1 Scientific contribution

This study contributes to science in several ways. First of all, as stated before, while the technical aspects of software components is widely debated in academic literature, there is very little empirical research available regarding the business aspects, especially regarding revenue models, of software components. So, this study contributes by filling the literature gap concerning software component revenue models and can serve as a starting point of future research on software component revenue models. Moreover, this study identified that the business challenges, especially regarding revenue models, of software components differ from custom-made software. In this way, this study acknowledges the need that more research is needed regarding the business aspects of software components. In doing so, it hopes to interest other researchers for this topic.

In addition, by having mapped out the most critical business model elements which influence the revenue model choice of software component developers. This research contributes to science with a first draft to study key business model determinants for choosing a revenue model for exploiting software components. Moreover, the finding of this research, i.e. various business model elements impact the viability of revenue models for software components, could be possibly interesting for other research topic areas, besides software components, as well. Namely, it is not unthinkable that the same business model elements have an influence on the viability of revenue models within other industries. Lastly, the used cases in this research contributes to literature as well. Namely, the cases used are described in-depth and can be used by other researchers for probably other purposes.

7.3.2 Practical contribution

Besides contributing to science, this study also contributes to practice. The Software Component Revenue Model Evaluation framework helps software component developers in Identity and Access Management to evaluate revenue models. So, this study serves as a tool for software component developers to make well-considered choices regarding revenue models. Also, this study revealed that SaaS solutions are not feasible in the software component industry for IAM. So, this study does not only help software component developers in IAM in evaluating revenue models, but also provides useful information on what types of solutions to offer (on-premise and on-demand via a public cloud). Besides, by having mapped out the advantages and disadvantages of standard-software (software component) vs. custom-made

software, it can assist software vendors and customers to evaluate whether to buy or develop standard software or custom-made software.

7.4 Limitations and recommendations for further research

As with any study, this thesis also has its limitations. First of all, as it is typically with qualitative research, this study is hard to generalize. The results can be only generalized to a certain extent. The findings should only be applied in an appropriate context, namely the software component industry for Identity and Access Management. Regarding the case studies, despite no major problems have emerged, some side notes can be made. The scope of this study only allowed three cases within the resources and time available, which hampers the external validity of this thesis. Also, only one person per case company was interviewed. Therefore, single response bias might have occurred. However, in an attempt to overcome these constraints, different types of data (e.g. interviews, company websites, company documents) were triangulated, which contributed to the validity of this research. A last generalization impediment is that the case companies are not very diverse, in terms of their targeted customer segments. Namely, the case companies' customers mainly include large organizations. These organizations do not want to rely on public cloud services and therefore the usage-based revenue model was not found applicable for software component developers. However, it might be the case that other customer segments, such as repair shops, are willing to use SaaS solutions.

According to Eisenhardt (1989), for external validity it is important to follow replication logic, because the generalization of findings is not automatic. Therefore, in an attempt to solve the problems of generalization, similar research with more case companies should be conducted, preferably including case companies who also target other customer segments that the current case companies. Also, not only a qualitative research design, but a mixed method approach, including quantitative data (e.g. surveys), might gain more significant results and generalizations. After replications have been made and the results of this study are confirmed, the findings might be accepted for generalization.

Another limitation of this study is that it solely focusses on the business model of a firm as the determinant for choosing a revenue model. The empirical observation from the exploratory interview study and case studies, indeed indicate that the revenue model choice is affected by the overall business model. However, it might be the case that other possible important determinants for choosing a revenue model (e.g. competitors) are missed. Therefore,

a different possibility for further research would be to study the influence of other factors, such as competition, on the revenue model choice of software component developers in IAM.

In addition, due to time constraints, it was not possible to include the other two actors in the software component business (i.e. application assembler and end-customer) in this study. If both actors were included in this research, it might have helped to create a more complete picture of the software component industry. Besides, it might have revealed insights, which could have contributed to the SCRME framework. Therefore, a possibility for further research is to involve the two other actors more in the research.

Also, it has to be noticed that the study was restricted in time and scope which for instance only allowed to use the most common revenue models in the software business as a basis to identify applicable revenue models for software components. Therefore, besides the one-time-charge, subscription and freemium revenue models, also other software revenue models could be potentially applicable for software component developers in IAM. For this reason, a point for further research is to analyze and evaluate more revenue models than solely the most common software revenue models in the software industry.

Another focus for further research could be to test the SCRME framework extensively in practice. Testing the SCRME framework gives organizations the chance to evaluate its usability and to encounter potential drawbacks. Moreover, it provides a chance to expand the SCRME framework by adding more business model factors. In this way, the SCRME framework can be optimized. Also, since the current SCRME framework does not give an overall score which decides whether a specific revenue model is applicable or not, an interesting possibility for further research is to investigate whether it is possible to assign scores to the specific business model factors. When assigning scores to the business model factors, it becomes easier for the user of the framework to evaluate revenue models.

Abbreviations

API	:	Application Programming Interface
B2B	:	Business to Business
BM	:	Business Model
BMC	:	Business Model Canvas
CEO	:	Chief Executive Officer
COO	:	Chief Operations Officer
COTS	:	Commercial off the shelf
CTO	:	Chief Technology Officer
IAM	:	Identity and Access Management
MOTS	:	Modified off the shelf
PIMN	:	Platform Identity Management Nederland
R&D	:	Research and Development
SC	:	Software component
SCRME framework	:	Software Component Revenue Model Evaluation framework
VP marketing	:	Vice-President marketing

Appendices

Appendix I: Interview topic list (exploratory interviews)

Interview protocol

Date	:
Time	:
Interviewee	:
Company	:
Job description	:
Product/service	:
Sector	:
Confidentiality	:
Data recording	:

Elaboration of the purpose of the research

The goal of this study is to contribute to knowledge on revenue models for software components. More specifically, this research investigates appropriate revenue models for exploiting software components. The end-result of this study is to provide a Software Component Revenue Model Evaluation framework which software component developers in Identity and Access Management can use to evaluate revenue models.

The goal of this interview is to discuss revenue models for software component suppliers. First, the aim is to create an overview of your business model. Subsequently, we will discuss the revenue models you use and the business model reasons behind it.

The interview starts with some general questions regarding your company and your positions. Subsequently, questions will be asked regarding the revenue model of a specific product within your company's portfolio.

1. **General questions**
    - Confidentiality issues (permission to record)
    - Company specific
    - Job description of the interviewee
    - Product specific

2. **Business Model**
    - Create an overview of the business model
        i. Customer segments
        ii. Value proposition
        iii. Channels
        iv. Customer relations
        v. Key resources
        vi. Key activities
        vii. Key partners
        viii. Revenue streams

3. **Revenue model**
    - Map out the revenue models which are used
    - Business model reasons behind using the revenue models
    - Discuss each revenue model individually

4. **Ending the interview**
    - Comments / questions
    - Request additional information/documents
    - Asking for potential case study companies

Appendix II: Interview topic list (case studies)

Interview protocol

Date	:
Time	:
Interviewee	:
Company	:
Job description	:
Product/service	:
Sector	:
Confidentiality	:
Data recording	:

Elaboration of the purpose of the research

The goal of this study is to contribute to knowledge on revenue models for software components. More specifically, this research investigates appropriate revenue models for exploiting software components. The end-result of this study is to provide a Software Component Revenue Model Evaluation framework which software component developers in Identity and Access Management can use to evaluate revenue models.

The goal of this interview is to discuss revenue models for software component suppliers. First, the aim is to create an understanding of which business model aspects influence the revenue model choice of software components. Subsequently, we will discuss four revenue models for software components, whereby the aim is to get insights for the perceived advantages and disadvantages from a software component developer view-point.

The interview starts with some general questions regarding your company and your positions. Subsequently, questions will be asked regarding the revenue model of a specific product within your company's portfolio.

1. **General questions**
    - Confidentiality issues (permission to record)
    - Company specific
    - Job description of the interviewee
    - Product specific

2. **Business Model**
    - Create an overview of the business model
        i. Customer segments
        ii. Value proposition
        iii. Channels
        iv. Customer relations
        v. Key resources
        vi. Key activities
        vii. Key partners
        viii. Revenue streams

    - Discuss elements which influences the revenue model

    - Discuss the challenges, emphasize:
        i. Integration
        ii. Dependent on clients business model
        iii. Distribution
        iv. Security

3. **Revenue model**
    - One time charge
        i. Explain the revenue model
        ii. Advantages vs. disadvantages (why)
        iii. Conditions to make it work
        iv. When would you use this revenue model

- Term licensing
    i. Explain the revenue model
    ii. Advantages vs. disadvantages (why)
    iii. Conditions to make it work
    iv. When would you use this revenue model

- Usage based
    i. Explain the revenue model
    ii. Advantages vs. disadvantages (why)
    iii. Conditions to make it work
    iv. When would you use this revenue model

- Freemium
    i. Explain the revenue model
    ii. Advantages vs disadvantages (why)
    iii. Conditions to make it work
    iv. When would you use this revenue model

- Other revenue models
    i. Explain the revenue model
    ii. Advantages vs disadvantages
    iii. Conditions to make it work
    iv. When would you use this revenue model

4. Ending the interview

- Comments / questions
- Request additional information/documents

Appendix III: Overview revenue models of the case companies

Revenue model	**Company A**	**Company B**	**Company C**
One-time-charge	License (initial fee) - optional: software assurance*; costs are **per-user**, per year. **software assurance: plan to get maintenance, support and updates.*	License (per-user) - optional: plan for maintenance, support and updates; costs are: **per-user**, per year	
Subscription	The subscription-plan includes:, maintenance, support and updates - Per-user (monthly-based*) **customers (users) only pay if they use the software in a particular month*	The subscription-plan includes: maintenance, support and updates - Per-user	The subscription-plan includes: maintenance, support and updates
Freemium	Time-limited freemium (30-day free trial)	Support & feature limited* **Freemium users do not get personal support and the software is only free up to 40 users*	Support limited* **with the freemium-model no support, maintenance or updates are delivered*
Usage-based		This is not used, because a SaaS	

		model is not offered. Besides, implementing a usage-based mechanism is very difficult and complex	
Other			Consulting revenue model; an hourly rate is charged for providing consultancy services

Appendix IV: Analysis: one-time-charge revenue model

With-in case analysis			**One-time-charge**
Components	Company A	Company B	Company C
Why is it used? Or not?	This revenue model is used, because large organizations are targeted. It stems from the traditional software market	Large organizations usually demand a perpetual license customers want to implement the product in their own IT-environment	the one-time-charge model is not used, because the software itself is distributed for free, the main reason is because the company uses open-source software
Business model determinants	customers: large organizations are targeted customers: large organizations have the resources to buy a perpetual license channels: customers who buy a perpetual license, usually want to implement the software on-premise customer relationships: support after implementation is only provided at an additional fee	customers: large organizations are targeted customers: organizations, such as banks, want to implement the software on-premise channels: when a perpetual license is sold, almost every customer wants to install the software on-premise customer relationships: after implementation, support is only provided at an additional fee (also includes for maintenance and updates)	

Advantages	one-time-sale, easy to recoup development costs it is easy for the software-vendor it can create a lock-in effect	it helps to recoup development costs quick	
Disadvantages	no recurring revenues” high initial costs for potential customers, which limits the target group	no recurring revenues	

Appendix V: Analysis: subscription-based revenue model

With-in case analysis			**Subscription**
Components	Company A	Company B	Company C
Why is it used? Or not?	the last two years, customers want to be more flexible, therefore the demand for subscriptions has been increased we target small to large organizations, therefore we want to offer a solution for every customer segment	there is a trend from traditional software-licensing (one-time-charge) to subscriptions in our industry, therefore we offer this solution	on-going services are provided, such as; support, maintenance and updates, therefore this revenue model is used
Business model determinants	customers: usually small-medium sized organizations demand a subscription-based model channels: the subscription-based revenue model is usually sold in combination with an on-demand solution key-activities: constantly developing to stay up-to-date	key-activities: when a subscription is sold, customers expect the product to be up-to-date at all time, so continuous development is very important key-resources: this plan works especially well when a company is financially stable, because in the beginning the revenues will be low customer relationships: support is included,	

	Key-resources: development costs are not recouped quick, so future investments needs to be funded in other ways customer relationships: support is included in the subscription-plan, therefore good customer services has to be provided	therefore a high customer service level has to be reached	
Advantages	customer does not become owner of the software with the subscription-plan, customer do not have to budget, therefore it lowers the barrier in the long-term, total revenues are 4 to 5 times as high compared to selling a perpetual license it diversifies the customer base	it diversifies the customer base companies do not hesitate, because companies only have to pay for just 1 year, so the risk in monetary terms is relatively low” forecast their expenses for three to five years maximum. The subscription-plan	it causes recurring revenues” do not have to count the amount of users
Disadvantages	it takes longer to recoup development costs	it takes longer to recoup development costs	if customers use it a lot, it costs us more money

Appendix VI: Analysis: freemium revenue model

With-in case analysis			**Freemium**
Components	Company A	Company B	Company C
Why is it used? Or not?	99% of the freemium-users are turned into paying-customers”	there should be a solution for companies who cannot pay for security	we are confident that free-users will turn into paid-users
Business model determinants		customer relationships: support is automated (we cannot afford to provide personal support for not paying customers	customer relationships: no (premium) support is given”
Advantages	it lowers the barrier for potential customers to use the product almost all freemium-users turn into paying customers	it lowers the barrier for customers to use the software without a freemium-model our business would never be this successful	helps to create a lock-in effect, the lock-in effect is in the technology rather than the time or money spent on the software solution diversifies the customer base lowers the barrier for potential customers
Disadvantages	we do not always get the possibility to show the freemium-user the full potential of the software		

	we have to send every freemium user a license-file, which costs time		
Conditions		support needs to be automated, otherwise it costs too much time and money Implementing the product should	

Appendix VII: Analysis: usage-based revenue model

With-in case analysis			**Usage-based**
Components	Company A	Company B	Company C
Why is it used? Or not?	Due to security reasons, no SaaS model is offered. Therefore, a usage-based model is not offered	Usage-based revenue models is not used, because we do not offer a SaaS solution	it is too complex to create a usage-based mechanism” “the software is usually implemented in the client’ secured IT-environment, therefore we cannot monitor the usage of the software
Business model determinants			
Advantages			
Disadvantages			
Conditions			

Appendix VIII: Topic list interview company X (example case)

1. **General questions**
    - Confidentiality issues (permission to record)
    - Company specific
    - Job description of the interviewee
    - Product specific

2. **Create an general overview of the business model**
    - Customers
    - Customer relationships
    - Key resources
    - Value proposition
    - Channels
    - Customer relations
    - Key resources
    - Key activities

3. **SCRME framework related questions**
    - Who are your customers?
        i. Large organizations vs. small-medium organizations
    - Is the software provided on-premise or on-demand?
    - How is customer service organized?
        i. Automated vs. personal support
    - How are the (development) costs financed?

4. **Ending the interview**
    - Comments / questions
    - Request additional information/documents

References

Al-Debei, M. M., & Avison, D. (2010). Developing a unified framework of the business model concept. *European Journal of Information Systems, 19*(3), 359-376.

Ayala, C., Hauge, Ø., Conradi, R., Franch, X., & Li, J. (2011). Selection of third party software in Off-The-Shelf-based software development—An interview study with industrial practitioners. *Journal of Systems and Software, 84*(4), 620-637.

Babbie, E. (2013). *The basics of social research*: Cengage Learning.

Bala, R., & Carr, S. (2010). Usage-based pricing of software services under competition. *Journal of Revenue & Pricing Management, 9*(3), 204-216.

Bergner, K., Rausch, A., Sihling, M., & Vilbig, A. (1998). *A componentware development methodology based on process patterns.* Paper presented at the Proceedings of the 5th Annual Conference on the Pattern Languages of Programs.

Bloor, R., Baroudi, C., & Kaufman, M. (2007). *Service oriented architecture for dummies*: John Wiley & Sons.

Bouwman, H., De Vos, H., & Haaker, T. (2008). *Mobile service innovation and business models*: Springer Science & Business Media.

Bouwman, W., De Reuver, G., Solaimani Kartalaei, H., Daas, D., Haaker, T., Janssen, W., . . . Walenkamp, B. (2012). *Business models: Tooling and a research agenda.* Paper presented at the 25th bled econference eDependability: Reliable and Trustworthy eStructures, eProcesses, eOperations and eServices for the future, 17-20 juni 2012, Bled, Slovenia.

Brereton, P., & Budgen, D. (2000). Component-based systems: A classification of issues. *Computer, 33*(11), 54-62.

Broy, M., Deimel, A., Henn, J., Koskimies, K., Plášil, F., Pomberger, G., . . . Szyperski, C. (1998). What characterizes a (software) component? *Software-Concepts & Tools, 19*(1), 49-56.

Chesbrough, H., & Rosenbloom, R. S. (2002). The role of the business model in capturing value from innovation: evidence from Xerox Corporation's technology spin-off companies. *Industrial and corporate change, 11*(3), 529-555.

Choudhary, V. (2007). *Software as a service: Implications for investment in software development.* Paper presented at the System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on.

Councill, B., & Heineman, G. T. (2001). Definition of a software component and its elements. *Component-based software engineering: putting the pieces together*, 5-19.

Cox, B. J. (1985). Object oriented programming.

Creswell, J. W., & Clark, V. L. P. (2007). Designing and conducting mixed methods research.

Crnković, I., Sentilles, S., Vulgarakis, A., & Chaudron, M. R. (2011). A classification framework for software component models. *Software Engineering, IEEE Transactions on, 37*(5), 593-615.

Cusumano, M. (2008). The changing software business: Moving from products to services. *Computer, 41*(1), 20-27.

DiCicco-Bloom, B., & Crabtree, B. F. (2006). The qualitative research interview. *Medical education, 40*(4), 314-321.

Driver, M. (2008). Predicts 2009: The Evolving Open-Source Software Model. *Gartner Group, Inc*.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of management review, 14*(4), 532-550.

Eisenhardt, K. M., & Graebner, M. E. (2007). Theory building from cases: opportunities and challenges. *Academy of management journal, 50*(1), 25-32.

Galvin, R. (2015). How many interviews are enough? Do qualitative interviews in building energy consumption research produce reliable knowledge? *Journal of Building Engineering, 1*, 2-12.

Gartner. (2008). Predicts 2009: The Evolving Open-Source Software Model. *Gartner Group, Inc*.

Gartner. (2015). Identity and Access Management (IAM). Retrieved from: http://www.gartner.com/it-glossary/identity-and-access-management-iam/.

Gat, I., & Succi, G. (2013). A Survey of the API Economy. *Cut. Consort*.

Gawer, A. (2009). Platform dynamics and strategies: from products to services. *Platforms, markets and innovation, 45*, 57.

George, G., & Bock, A. J. (2011). The business model in practice and its implications for entrepreneurship research. *Entrepreneurship theory and practice, 35*(1), 83-111.

Ghaziani, A., & Ventresca, M. J. (2005). *Keywords and cultural change: frame analysis of business model public talk, 1975–2000.* Paper presented at the Sociological Forum.

Guntersdorfer, M. S., Kay, D. G., & Rosenblum, D. S. (2000). Using Software Patents To Support The Business Model Of Software Components.

Günzel-Jensen, F., & Holm, A. B. (2015). Freemium business models as the foundation for growing an e-business venture: a multiple case study of industry leaders.

Halinen, A., & Mainela, T. (2013). *Process Research in business networks: a review of longitudinal research methods.* Paper presented at the IMP Conference, Atlanta.

Hedman, J., & Kalling, T. (2003). The business model concept: theoretical underpinnings and empirical illustrations. *European Journal of Information Systems, 12*(1), 49-59.

Heiander, N., Ulkuniemi, P., & Seppänen, V. (2002). Understanding software component markets: The value creation perspective *Software Quality—ECSQ 2002* (pp. 256-266): Springer.

Heineman, G., & Councill, W. Component-based software engineering: putting the pieces together. 2001: Addison-Wesley: Reading, Massachusetts.

Helander, N., & Ulkuniemi, P. (2006). Marketing challenges in the software component business. *International Journal of Technology Marketing, 1*(4), 375-392.

Helander, N., & Ulkuniemi, P. (2012). Customer perceived value in the software business. *The Journal of High Technology Management Research, 23*(1), 26-35.

Hoepfl, M. C. (1997). Choosing qualitative research: A primer for technology education researchers.

Janes, A., Remencius, T., Sillitti, A., & Succi, G. (2014). Towards Understanding of Structural Attributes of Web APIs Using Metrics Based on API Call Responses *Open Source Software: Mobile Open Source Technologies* (pp. 83-92): Springer.

Jankowicz, A. D. (2005). *Business research projects*: Cengage Learning EMEA.

Kittlaus, H.-B., & Clough, P. N. (2008). *Software product management and pricing: Key success factors for software organizations*: Springer Science & Business Media.

Lambrecht, A., Goldfarb, A., Bonatti, A., Ghose, A., Goldstein, D. G., Lewis, R., . . . Yao, S. (2014). How do firms make money selling digital goods online? *Marketing Letters, 25*(3), 331-341.

Lampson, B. W. (2004). Software components: Only the giants survive *Computer Systems* (pp. 137-145): Springer.

Liu, A., & Gorton, I. (2003). Accelerating COTS middleware acquisition: The i-Mate process. *Software, IEEE, 20*(2), 72-79.

Luoma, E. (2013). Examining Business Models of Software-as-a-Service Firms *Economics of Grids, Clouds, Systems, and Services* (pp. 1-15): Springer.

Magretta, J. (2002). Why business models matter.

McIlroy, M. D., Buxton, J., Naur, P., & Randell, B. (1968). *Mass-produced software components.* Paper presented at the Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany.

Miller, C. C. (2009). Ad revenue on the Web? No sure bet. *New York Times http://www. nytimes. com/2009/05/25/technology/start-ups/25startup. html*.

Mingbai, F. (2011). *The Study on Software Reuse Based on SOA Architecture.* Paper presented at the International Conference on Measurement and Control Engineering 2nd (ICMCE 2011).

Morris, M., Schindehutte, M., & Allen, J. (2005). The entrepreneur's business model: toward a unified perspective. *Journal of business research, 58*(6), 726-735.

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and organization, 17*(1), 2-26.

Nazir, S., Shahzad, S., Nazir, M., & Rehman, H. U. (2013). *Evaluating security of software components using analytic network process.* Paper presented at the Frontiers of Information Technology (FIT), 2013 11th International Conference on.

Neighbors, J. (1989). DRACO: A Method for Engineering Reusable Software Systems. 1989 ACM, Inc: Addison-Wesley Publishing Co., Reading MA.

Niculescu, M. F., & Wu, D. J. (2011). When should software firms commercialize new products via freemium business models. *Under Review*.

Ojala, A. (2013). Software-as-a-Service Revenue models.

Ojala, A., & Tyrväinen, P. (2012). *Revenue models in cloud computing.* Paper presented at the Proceedings of 5th Computer Games, Multimedia & Allied Technology Conference (CGAT 2012).

Osterwalder, A. (2004). The business model ontology: A proposition in a design science approach.

Osterwalder, A., & Pigneur, Y. (2010). *Business model generation: a handbook for visionaries, game changers, and challengers*: John Wiley & Sons.

Osterwalder, A., Pigneur, Y., & Tucci, C. L. (2005). Clarifying business models: Origins, present, and future of the concept. *Communications of the association for Information Systems, 16*(1), 1.

Palo, T., & Tähtinen, J. (2013). Networked business model development for emerging technology-based services. *Industrial Marketing Management, 42*(5), 773-782.

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

Pour, G. (1998). *Moving toward component-based software development approach.* Paper presented at the Technology of Object-Oriented Languages, 1998. TOOLS 27. Proceedings.

Pujol, N. (2010). Freemium: attributes of an emerging business model. *Available at SSRN 1718663*.

Raemaekers, S., van Deursen, A., & Visser, J. (2012). *An analysis of dependence on third-party libraries in open source and proprietary systems.* Paper presented at the Sixth International Workshop on Software Quality and Maintainability, SQM.

Rajala, R., Nissilä, J., & Westerlund, M. (2006). *Determinants of OSS revenue model choices.* Paper presented at the ECIS.

Ritter, D. (2013). Towards a Business Network Management *Enterprise Information Systems of the Future* (pp. 149-156): Springer.

Saunders, M., Lewis, P., & Thornhill, A. (2009). *Research Methods for Business Students. Harlow, England: Pearson Education*. Retrieved from

Schlauderer, S., & Overhage, S. (2011). *How perfect are markets for software services? an economic perspective on market deficiencies and desirable market features.* Paper presented at the ECIS.

Sedigh-Ali, S., Ghafoor, A., & Paul, R. (2001). *Metrics-guided quality management for component-based software systems.* Paper presented at the Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International.

Shafer, S. M., Smith, H. J., & Linder, J. C. (2005). The power of business models. *Business horizons, 48*(3), 199-207.

Shi, H., Li, T., Liu, R., Chen, J., Li, J., Zhang, A., & Wang, G. (2015). A service-oriented architecture for ensemble flood forecast from numerical weather prediction. *Journal of Hydrology, 527*, 933-942.

Software integrity risk report. Technical report. Forrester Consulting, 2011.

Sridhar, S. (2015). A Review on Reuse of Software Components for Sustainable Solutions in Development Process. *IJITR, 3*(2), 1998-2001.

Szyperski, C. (1998). Component Software: beyond object-oriented software. *Reading, MA: ACM/Addison-Wesley*.

Szyperski, C. (2000). Component software and the way ahead. *Foundations of Component-Based Systems*, 1-20.

Teece, D. J. (2010). Business models, business strategy and innovation. *Long range planning, 43*(2), 172-194.

Timmers, P. (1998). Business models for electronic markets. *Electronic markets, 8*(2), 3-8.

Tsang, E. W. (2014). Case studies and generalization in information systems research: A critical realist perspective. *The Journal of Strategic Information Systems, 23*(2), 174-186.

Ulkuniemi, P., Araujo, L., & Tähtinen, J. (2015). Purchasing as market-shaping: the case of component-based software engineering. *Industrial Marketing Management, 44*, 54-62.

Upward, A., & Jones, P. (2015). An Ontology for Strongly Sustainable Business Models Defining an Enterprise Framework Compatible With Natural and Social Science. *Organization & Environment*, 1086026615592933.

Vitharana, P. (2003). Risks and challenges of component-based software development. *Communications of the ACM, 46*(8), 67-72.

Wu, X., & Woodside, M. (2004). *Performance modeling from software components.* Paper presented at the ACM SIGSOFT Software Engineering Notes.

Yin, R. K. (2003). Case study research design and methods third edition. *Applied social research methods series, 5*.

Zott, C., Amit, R., & Massa, L. (2011). The business model: recent developments and future research. *Journal of management, 37*(4), 1019-1042.