Business Model Differences between Open Source and Proprietary SaaS Providers

Jop Bröcker
Business Model Differences between Open Source and Proprietary SaaS Providers

THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Systems Engineering, Policy Analysis & Management

by

Jop Bröcker
Dordrecht, The Netherlands

TU Delft
Section of Information & Communication Technology
Department of Infrastructure Systems & Services
Faculty of Technology, Policy, and Management
Delft University of Technology
Delft, the Netherlands
www.tbm.tudelft.nl
Business Model Differences between Open Source and Proprietary SaaS Providers

Author: Jop Bröcker
Student id: 1006363
Email: jop@brocker.nl

Abstract

This thesis aims to identify the differences between the business models of proprietary and open source ‘software as a service’ (SaaS) providers. Knowledge of these differences aids SaaS providers in their decision whether or not to release their source code to the open source community. To identify possible differences in the business models of open source and proprietary SaaS providers this thesis first gains understanding in the fields of software as a service and open source versus proprietary software. Consecutively various sources of literature on business models have been used to identify possible elements of business models that differ.

With a multiple case study these differences have been analysed for two types of software: Web Content Management software (Wordpress versus Squarespace) and Customer Relationship Management software (SugarCRM versus Salesforce.com). This resulted in four business model elements that differ: Pricing of open source SaaS providers is lower; open source providers have to deploy value activities not based on their open source software; the more the community is involved in the development of software, the more the network complexity increases; development costs can be lowered when the community contributes extensions to the software, or is able to contribute to the core of the software. From these findings it can be said that although development cost can be lowered, leading an open source project also increases complexity of the network of actors which in turn raises costs, possibly outrunning the cost reductions.

Thesis Committee:

Chair: Dr. W.A.G.A. Bouwman, Faculty TPM, TU Delft
University supervisor: Dr. ir. J. van den Berg, Faculty TPM, TU Delft
Committee Member: Dr. ir. A.J.H. Hidders, Faculty EWI, TU Delft
Committee Member: Dr. M.L.C. de Bruine, Faculty TPM, TU Delft
Preface

This thesis is the final element of the completion of my Master of Systems Engineering, Policy Analysis and Management, with which I will end my academic career (i.e. student life) and which enables me to fully focus on my ambitions in the commercial field. The subject of this thesis is closely related to strategic issues encountered at the software company (Yes2web) I have been involved with over the past years. The IT domain within the SEPAM master was a logical theoretical supplement to the practical experience derived from this start-up. Based on the results of the thesis the management of Yes2web can better evaluate the pro’s and cons of a open source versus proprietary strategy (I hope).

First off I would like to thank my graduation committee for their support and feedback on my work, especially Harry Bouwman for the many occasions I have had the possibility to debate my opinion and for sharing his thoughts on business models, STOF models and dual markets. I appreciate the clear instructions I have received to improve my work.

My colleagues at Yes2web are thanked for their valuable input, their interest in my progress and the endless supply of coffee. Special thanks to Hans for reading my abstract and to Gerrit Jan for this beautiful \LaTeX{}template. I look forward to a period of tense collaboration to realise our ambitions.

I would also like to thank my parents, the combination of their patience and —oddly— impatience have pulled me through the completion of this final step in my education. Last, certainly not least, Alice: For correcting and discussing my thesis, for not complaining over the ever shifting deadlines and her patience during the many years it took to finish my study.

Jop Bröcker
Delft, the Netherlands
May 14, 2009
Contents

Preface iii

Contents v

List of Figures vii

1 Introduction 1
   1.1 Background ................................................. 1
   1.2 Research Problem ........................................ 3
   1.3 Research Questions ....................................... 4
   1.4 Research Methods ......................................... 5
   1.5 Outline of thesis ......................................... 5

2 Domain 7
   2.1 Software as a Service (SaaS) ............................ 7
   2.2 Software development: Proprietary versus Open Source .......... 13
   2.3 Commercialising Open Source ............................ 18
   2.4 SaaS Software Applications ............................. 19
   2.5 Conclusion ................................................. 20

3 Theories and concepts 21
   3.1 Business Model Design for Services ....................... 21
   3.2 Service Domain ............................................. 24
   3.3 Technology Domain ........................................ 26
   3.4 Organisation Domain ...................................... 26
   3.5 Finance Domain ............................................ 30
   3.6 Phases of a Service Business Model ...................... 34
   3.7 Propositions ............................................... 35
   3.8 Conclusion ............................................... 35

4 Methodology 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Research Objective and Approach</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>Research Method</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>Case Protocol</td>
<td>38</td>
</tr>
<tr>
<td>5</td>
<td>Case Results</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>Case 1: Wordpress.com</td>
<td>48</td>
</tr>
<tr>
<td>5.3</td>
<td>Case 2: Squarespace</td>
<td>53</td>
</tr>
<tr>
<td>5.4</td>
<td>Case 3: Sugar CRM</td>
<td>56</td>
</tr>
<tr>
<td>5.5</td>
<td>Case 4: Salesforce.com</td>
<td>61</td>
</tr>
<tr>
<td>5.6</td>
<td>Case Comparison 1</td>
<td>64</td>
</tr>
<tr>
<td>5.7</td>
<td>Final Analysis</td>
<td>70</td>
</tr>
<tr>
<td>5.8</td>
<td>Conclusions</td>
<td>74</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions &amp; Recommendations</td>
<td>77</td>
</tr>
<tr>
<td>6.1</td>
<td>Conclusions</td>
<td>77</td>
</tr>
<tr>
<td>6.2</td>
<td>Contributions</td>
<td>79</td>
</tr>
<tr>
<td>6.3</td>
<td>Limitations</td>
<td>81</td>
</tr>
<tr>
<td>6.4</td>
<td>Future Research</td>
<td>82</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>85</td>
</tr>
<tr>
<td>A</td>
<td>STOF Model</td>
<td>91</td>
</tr>
<tr>
<td>A.1</td>
<td>Introduction</td>
<td>91</td>
</tr>
<tr>
<td>B</td>
<td>Case Study Questions</td>
<td>97</td>
</tr>
<tr>
<td>C</td>
<td>Interviews WordPress</td>
<td>101</td>
</tr>
<tr>
<td>C.1</td>
<td>Matt Mullenweg</td>
<td>101</td>
</tr>
<tr>
<td>C.2</td>
<td>Toni Schneider</td>
<td>102</td>
</tr>
<tr>
<td>C.3</td>
<td>Peter Westwood</td>
<td>103</td>
</tr>
<tr>
<td>D</td>
<td>Interview Squarespace</td>
<td>105</td>
</tr>
<tr>
<td>D.1</td>
<td>Dane Atkinson</td>
<td>105</td>
</tr>
<tr>
<td>E</td>
<td>Interviews Sugar CRM</td>
<td>107</td>
</tr>
<tr>
<td>E.1</td>
<td>Chris Harrick</td>
<td>107</td>
</tr>
<tr>
<td>E.2</td>
<td>David Wheeler</td>
<td>109</td>
</tr>
<tr>
<td>F</td>
<td>Interviews Salesforce.com</td>
<td>111</td>
</tr>
<tr>
<td>F.1</td>
<td>Hans Brouwer</td>
<td>111</td>
</tr>
<tr>
<td>F.2</td>
<td>Shelley Gordon</td>
<td>112</td>
</tr>
<tr>
<td>F.3</td>
<td>Transcript of Interview</td>
<td>112</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Research framework ....................................................... 8
3.1 Research framework related to the STOF model ......................... 22
3.2 STOF Model (Bouwman, de Vos and Haaker, 2008) ..................... 23
3.3 Technology Life Cycle (Bouwman et al., 2008) .......................... 34
4.1 Research framework - Case selection ................................. 39
A.1 Service Domain (Bouwman et al., 2008) ............................... 92
A.2 Technology Domain (Bouwman et al., 2008) ............................ 93
A.3 Organisation Domain (Bouwman et al., 2008) .......................... 93
A.4 Finance Domain (Bouwman et al., 2008) ............................... 94
A.5 Phases in business models (Bouwman et al., 2008) ..................... 94
Chapter 1

Introduction

This research thesis will focus on open source software development and Software as a Service (SaaS) in the field of online software. Section 1.1 will introduce the basic concepts of ‘online software’, ‘SaaS’ and ‘open source software’. Section 1.2 introduces the main research problem after which the research questions will be identified in § 1.3. Having discussed the research methods in § 1.4 we will outline this thesis in § 1.5.

1.1 Background

1.1.1 Transition to online software

During the past years the software applications that run from a web-browser (online software applications) have established solid ground in the software market. Example applications are web-mail (e.g. Hotmail or Gmail), direct mailing software, website statistics (e.g. webaliser or, Nedstats), content management systems (used to modify the contents of a website) and lately even office applications have been migrated to the browser (Google Docs). For each of these applications, an offline version (also known as standalone) is also available. It depends both on the preference of the end-user and the relation of the application with the internet whether an online or an offline variant is chosen. E.g.: It is more logical to manage your website through a web-browser, since it is visited by visitors using a web-browser, than it is to use video editing software through your browser, due to the large amounts of data that would have to be transferred over the internet connection.

1.1.2 Software as a Service (Saas)

A revenue model for online software growing in popularity is Software as a Service\(^1\). In this revenue model a periodic fee is paid for the use of the software. This software is centrally hosted and managed and web-based, as described in the section above. No large investment in licences, software installation or hardware is needed and support is provided by the SaaS provider. Since the end-user does not pay directly for the software, this model

---

\(^1\)SaaS can be seen as a successor to ASP, but is different in various aspects. The differences are discussed in § 2.1.4
is also suitable for open source software (Van der Pluijm, 2007). This takes away some burdens of open source software like support, maintenance, installation and hosting, which are often not offered by ‘the open source community’. The SaaS provider could even use open source software not developed by himself, which is freely available.

This does not make the SaaS model less interesting for commercial (closed) organisations based on proprietary software. Examples of successful SaaS providers are Salesforce.com (CRM), Basecamp (project management) and Google Docs (office applications). Without the need for a distribution channel (their clients sign up for the service through their website) they are able provide their services to thousands of customers all over the world.

1.1.3 Transition to openness

Parallel to the online-software trend, some companies have adopted new business models, opposed to the classic model of exploiting intellectual property, keeping all knowledge inside (a closed business model). These companies have introduced products by partnering with another company (or in some cases multiple companies). This model has been described as a more open business model or as Open Innovation (Chesbrough, 2006). Reasons for adopting more openness can be found in the rate in which new product developments are introduced. As product life cycles shorten, time-to-market decreases, putting higher demands on innovation teams. On the other hand, as products become increasingly complex costs of development are rising, making it harder to turn intellectual property into profit. By partnering with specialised firms these hurdles can be overcome, at the cost of sharing some IP outside the company.

This is especially the case for software development, where the rate of new product introductions is very high. Due to the inability to protect software with patents (European Patent Office, 2000), competition can copy a product immediately after its introduction. Adopting a more open approach to software development is one strategy to building a stronger platform (Wonglimpiyarat, 2004) for the introduction of the software innovation.

Open Source Software

The most ‘open’ approach to software development is the adoption of the open source model. The source code is provided to the open source community where everyone is free to use the software and free to contribute to the source code. This has of course its implications for the business, as the software can no longer be sold as a licensed product since the software itself is available for free. There are however a number of other ways to make profit from the software, for example by delivering support and service.

The concepts online software, open source software and software as a service will be further explored in chapter 2.
1.2 Research Problem

The management of a software developing company has to make a strategic decision whether to go open source, or to keep everything closed (i.e. develop everything internally). In the past most commercial software developers rarely considered the release of their software to the open source domain as a serious option. More recently in The Netherlands these companies were forced into a broader mindset through the government, which formulated a dominant preference for open source software and open standards (Ministery van Economische Zaken, 2007). Thus any company targeting the Dutch government has to come with strong argumentation not to make their software available to the open source community.

1.2.1 Open Source implications

There is more to ‘going open source’ than just uploading source-code to a community like sourceforge.com. An open source software development strategy affects the whole way of creating and capturing value, thus affects the business model\(^2\), since it needs to incorporate the open way of innovation (Chesbrough, 2006). Apart from the security implications (source code is immediately available to everyone intending to do harm to the organisation), there are some organisational issues. Not only clients will demand support and feature requests, developers will demand feedback on bug-reports, contributions and questions. The internal R&D needs to restructure the way of programming, to ensure external contributions don’t interfere with internal updates. The software itself is no longer the main intellectual property owned by the company, but knowledge, customer loyalty and trade-secrets become the key to competitive advantage (Chesbrough, 2003).

1.2.2 Objective

This paper tries to identify elements of the business model affected by releasing the software to the open source domain.

Several SaaS providers are known which are considering to release their software to the open source domain. From a managerial perspective, knowledge of the implications for the business model aids SaaS providers in their decision whether to release their software or keep it proprietary and judge whether this will be beneficial to the organisation. Although the voluntary contributors to open source seem attractive to organisations limited on R&D budget, little is known on the differences between open source and proprietary companies. By identifying these differences, the results of this thesis contribute to a better founded decision between the pro’s and cons of this trade-off.

From a scientific perspective the goal is to broaden the understanding of differences in business models for proprietary software and open source software. The goal is not to provide a cost/benefit analysis or a pricing strategy for proprietary versus open source software, as has often been discussed in related literature (Economides and Katsamakas, 2006; Casadesus-Masanell and Ghemawat, 2006). Related research has primarily focused on whether (Tirole, 2003; Hawkins, 2004b) and how (Hecker, 1999; Goth, 2005) profit

\(^2\)The definition of the term textbusiness model is adopted from STOF literature as discussed in § 3.1.1.
can be made off open source and what the motivation is of open source developers which contribute without financial compensation (van Wendel de Joode, 2005).

1.2.3 Scope

For this thesis the explicit choice for SaaS providers is made due to the separation of the revenue model and the underlying software licences: Both proprietary and open source software can be offered through a SaaS business model, without having to revise contracts with its clients. The pricing structure can remain the same and the organisation does not have to look into new ways of making revenue. This provides a form of ‘ceteris paribus’ for the comparison of the business models.

Within the SaaS market, Web Content Management (WCM) and Customers Relationship Management (CRM) will be chosen as applications under study to verify consistency of the findings of the thesis between the different types of SaaS providers. The choice for these two types of SaaS software was made based on serveral arguments: First, the CRM market has some very strong SaaS providers which can be seen as the initiators of the SaaS trend for applications that were traditionally run on local servers. Second, the WCM market differs from CRM in that the application is almost always run online (from a webserver), but only lately providers of SaaS WCM are becoming popular. The latter is also of interest to the author, due to his background in WCM software. The differences between WCM and CRM are further discussed in § 2.4.

1.3 Research Questions

The results of this thesis are intended to guide software developing companies in their decision open sourcing their proprietary software or not. To make this decision differences between business models of open source and proprietary SaaS providers need to be known. For that reason the question to be answered by this research is: What are the differences between the business models of proprietary SaaS providers and open source SaaS providers?

To answer the above question understanding is needed of both SaaS and open source/proprietary software. This will be answered through the question What is SaaS? and What are the differences between proprietary and open source software?

As we are looking at the impact on the business model, What are the elements of a business model? is a relevant question. With the elements of a business model defined, we can look at the expected differences between proprietary and open source for these elements. What elements of the business model of a SaaS provider are expected to differ between proprietary and open source cases? results in a list of propositions which will have to be verified.

To analyse whether these propositions can be funded by findings in practice, the question How are the expected differences best verified? will provide the research method for this thesis.

3Throughout this thesis open-sourcing is the act of releasing previously proprietary software under an open source/free software license.
1.4 Research Methods

Using the STOF model as a starting point, differences between business models of proprietary and open source SaaS providers will be explored using related literature. This results in a list of propositions that indicate a possible difference in the business model.

Using a multiple-case study we gain insight in the business model of real-life cases where the SaaS providers have either not chosen to open source their software, or have indeed made this decision. The case study will be conducted for four cases, consisting of two cases of open source software versus two cases of closed source software. Two providers of Customer Relationship Management (CRM) software and two providers of Website Content Management (WCM) software will be assessed.

1.5 Outline of thesis

The introduction describes the problem area, the intent of the research project, the deliverables and defines the research questions.

Chapter 2 will define the different concepts used in this research project. Chapter 3 will identify the differences between open source business models and proprietary business models, using the STOF model.

The fourth chapter describes the applied methodology and the cases to be used in the case studies are identified.

In chapter 5 the cases will be studied to test whether the propositions found in chapter 3 are indeed also recognised in practice.

The sixth and final chapter provides final conclusions and recommendations.
Chapter 2

Domain

This chapter describes the different aspects of the software market relevant for this thesis. As stated in the introduction we will focus on SaaS, which we will explore in the subsequent section and try to answer the question What is SaaS? To be able to analyse the trade-off between proprietary software to open source business models, understanding is needed of the differences between open source and proprietary software, which is what § 2.2 addresses. This will provide an answer to the question What are the differences between proprietary and open source software? In the case study of the thesis, the differences between open source and proprietary will be analysed for two different types of SaaS offerings. The final section sheds some light on the different types of software applications (WCM and CRM) that will be used in these case studies.

2.1 Software as a Service (SaaS)

As some fields of software are becoming more of a commodity, service and support are gaining importance over intellectual property rights (Goth, 2005). Kakabadse and Kakabadse (2005) show that outsourcing IT becomes more important, to keep focus on the core business. Together with the trend that more and more software runs through a browser as ‘online software’ this resulted in a new revenue model which is called Software as a Service (SaaS). The software vendor offers its software on the internet and the service can be accessed through a web browser only (this is the key characteristic of ‘online software’ and differentiates SaaS from ASP, as will be discussed in § 2.1.4). By signing up for an account at the provider, the customer can make use of the software. SaaS is often offered in different plans, based on the needed capacity. For demanding customers Service Level Agreements can be offered. As Bennett, Layzell, Budgen, Brereton, Macaulay and Munro (2000) predicted software usage (and contracts) will become more ad-hoc. This can also be seen in the plans of SaaS providers, which can usually be canceled per month.

To understand SaaS we will first look into ‘online software’ (the most important characteristic) and discuss some other important characteristics of SaaS.
2.1 Software as a Service (SaaS)

2.1.1 Online Software

The basic idea is that applications will be run from the internet through a web-browser (e.g. Internet Explorer, FireFox), instead of being installed on each individual computer (John Hagel III, 2001). Users are able to access these applications from any computer with an internet connection and a web browser using login credentials. The application is hosted and kept up-to-date by a software provider (the provider often also is the developer of the software). This provider can provide service level agreements on the quality of service (such as up-time\(^1\), back-ups and performance). As the data is also stored on the web at the provider, the end-user does not have to worry about his data in case of hardware failures or theft of the computer.

Web-mail  E-mail was probably one of the first software applications to migrate to the web. A logical step as the e-mails are transmitted over the internet and stored on mail-servers before being downloaded to a local computer. A web-mail application leaves the e-mail on the server and provides an interface to the e-mail comparable to a standalone e-mail client (e.g. Hotmail\(^2\)). The convenience of accessing your e-mail from any internet connection led to large popularity of Hotmail. It reached 8.5 million users in their first two years of operation (Galante, 1997).

Several web technologies have further pushed the success online software, of which we will describe AJAX, Web services and Cloud Computing in the following sections.

AJAX  AJAX (Asynchronous JavaScript and XML) is a term first introduced by Jesse James Garrett which comprises several technologies used by online software applications to enhance the

---

\(^1\) Uptime is a measure of the time a computer system has been providing service without interruption.

\(^2\) Hotmail can be found at: http://www.hotmail.com
interface for the end-user (Garrett, 2005). Instead of reloading the entire webpage every
time a users clicks, only specific parts of the interface are updated with new information
which is loaded in the background, creating a single-page interface. This mimics the way
most standalone applications work, e.g.: If you click elements within a word-processor, only
specific parts are updated or changed. Besides the more intuitive interface, AJAX offers an
other advantage over the traditional ‘full-page refresh’: Since less data has to be transmitted,
AJAX enhanced websites tend to respond quicker. This increased responsiveness further
approaches the user experience of interfaces of standalone applications.

**Google Maps** A good example of a web application using AJAX is Google Maps³. As
you drag or click the map, only those parts of the map are loaded that were not visible
before. Without AJAX, the whole page would refresh and every image of the map and the
layout would need to be retransmitted to the computer.

Google has further extended its portfolio with online applications since. Gmail (Googles
webmail service) utilises a single-page interface, powered by AJAX. Google offers word-
processing, spreadsheet and presentational software, bundled as Google Docs⁴. All online
software applications of Google are bundled into ‘Google Apps for businesses’⁵. As with
most online software, these applications can be used without the need of a system admin-
istrator and without the investment in hardware, as this is handled by the software provider
(in this case Google). Due to the accessibility of the services from any internet connection,
employees can also check their email anywhere, in case of a in-house hardware/software
solution this is often restricted to the office or using complex VPN connections.

**Web services**

Web services enable the communication between software on different computers or servers
over the internet. This way a client can obtain a service from another server. An example
best illustrates the possibilities: When a user visits the website of a travel agency, it would
like information on flight schedules, flight availability, proximity of hotels, availability of
hotels and room rates. For the travel agency its impossible to maintain a accurate database
with such information, due to the high volatility of the data, e.g.: For last-minute flights the
availability can change every minute. The solution lies in web services: When the visitor
requests a specific type of information, the travel agency fetches this information through
a web service at the origin and provides this to the user. The user has no notice of this
background request, the software mimics to be a fully integrated application.

This integration of databases is very useful for online software application vendors as
they can benefit from different sources of accurate data, without the need to maintain the
data (maintenance is done by the web service provider). To the end-user it results in more
accurate data, especially compared to traditional standalone applications where the data was
stored on CD-rom or hard disk, where the data had to be periodically updated.

³Google Maps can be found at: http://maps.google.com
⁴Google Docs can be found at: http://docs.google.com
⁵Google Apps includes Gmail, Calendar, Docs, Chat and Sites. For further explanations of these applica-
tions can be found at: http://www.google.com/a/
Cloud Computing

In the offline environment, there is a risk of hardware failure: the hard-drive or other critical parts could fail and the application will no longer run or worse, data could get lost. This risk still exists with online applications, but as it is centralised it is easier to take preventive measures like backups and redundant web-servers. Cloud Computing is the concept where services and data are no longer hosted on a single server, but spread over many servers (the cloud). If one of the servers fails, others seamlessly take over. If the load on the servers gets too high, more servers are added for increased computing power. Through virtualisation of the operating system, the applications have no notion of the complexity of the underlying hardware, which means little specific adaptations are needed to run an online software application on a cloud.

Amazon Elastic Compute Cloud  Amazon (known best for its online book store) had to switch to a cloud computing platform in order to keep up with the large number of visitors and the increasing demand for up-time. They developed their own cloud computing platform, which they later made available to outsiders known as Amazon Elastic Compute Cloud (EC2).

Online Software Pitfalls

Despite the advantages of these new technological developments, online software also has some logical drawbacks. This thesis does not intend to find solutions to these drawbacks, but we do want to point them out for the sake of completeness.

As noted in the first paragraph of this section everything is done online, so in case of connection loss all online software applications are unreachable. To parry this software vendors have started to work on offline fall-backs for online applications. This essentially turns an online application into a standalone version. As soon as the internet connection is re-established the software resynchronises with the online version. This however, requires the installation of a small software component on the computer, which eliminates an earlier advantage of no installation requirements.

Besides connection loss the bandwidth of the connection also influences the usability of the application. Slow connections (non-broadband) make the application slow, even when using AJAX, compared to stand-alone applications. In The Netherlands the broadband penetration is relatively high, making it more suitable for online software (OECD, 2008). Countries like Mexico will be less suitable for online software vendors, due to low broadband penetration.

Another problem is the security and availability of the data. The online software provider has access to all data. How can be guaranteed that data is kept confidential? Besides keeping the data confidential, it also has to made sure data is not lost in case of system failure for which cloud computing can prove a solution (as noted before). This makes online software

---

6Amazon EC2 can be found at: http://www.amazon.com/ec2
7Examples are Gears (http://gears.google.com) or Adobe Air (http://www.adobe.com/products/air/)
less suitable for privacy sensitive information.

Having addressed the main concepts of online software, the following sections will highlight some specific characteristics for SaaS.

2.1.2 Multiple tenants

An important characteristic of SaaS is multi-tenancy, i.e. a single instance of the software serves multiple clients (tenants). This enhances economies of scale, as the addition of another client does not require installation of software or hardware, it only requires specification of meta-data (settings) to tailor the software to the needs of the client (Chong and Carraro, 2006).

For traditional online-software vendors to adopt the SaaS model, changes need to be made to both the software and the business model. Chong and Carraro (2006) describe four levels of maturity:

**Level I: Ad Hoc/Custom** The first level resembles most with the ASP software delivery model (Plepys, 2002): A single instance of the software delivers service to a single client. Software is installed per client and can be customised to its needs. This also implicates that each instance needs to be updated individually, which will be hazardous for customised software. Any online software package can be offered in this way.

**Level II: Configurable** The second level still offers a single instance per tenant, but instead of customising the instance, it is adjusted to the clients needs by configuration. All instances are identical, which makes maintenance less of a hassle. To offer this level of SaaS the vendor needs to change the software to be fully configurable. This will always mean that some clients will have to be turned down, as their needs are too specific for the software.

**Level III: Configurable, Multi-Tenant-Efficient** This level requires multiple tenants to be served by a single instance of the software. It inherits the characteristics of the previous level that each tenant is configurable to adjust the software to its needs. The end-user should not notice the difference between this level and the previous, the software should take care that each customers data is kept separate. Upscaling is limited by the limits of a single server. The vendor either has the choice to add servers (and thus add instances) or to progress to the next level.

**Level IV: Configurable, Multi-Tenant-Efficient, Scalable** The final level combines all the characteristics of the previous levels and adds scalability. Where the scalability of the previous level was limited to the computing power of a single server, the vendor serves the customer from a load balanced farm of identical instances (or hosts the service in a cloud, as discussed in § 2.1.1). Without adjusting the software, the capacity can be scaled by adding servers.
The fourth level is not always the ultimate goal for every software vendor, as it trades in flexibility for scalability. The level chosen by a vendor depends on various factors like the demands of users and design of the software.

2.1.3 Versioning

SaaS is often offered in different versions incrementing in level of support and capacity. To lower the barrier of entry a minimalistic free or trial version is often offered, for which support is minimal. An example is Basecamp\(^8\), which offers a free version capable of maintaining a single project. This could be sufficient for single users, but organisations (at which the software is targeted) will want to run multiple projects once acquainted with the free version.

2.1.4 ASP

Critics have called SaaS the new name for ASP (application service provider or provisioning). However, when examined closely there are some key differences. ASP also offers software over an internet connection where hardware and back-ups are managed by the provider, but this is where the resemblance halts: ASP typically runs over a remote desktop connection instead of a browser. The reason for this is the nature of ASP software, most often offline software applications which could also be installed at the end-user. ASP rarely exceeds level 1 of the above described levels: a licence of the software is installed per client. In most cases the software is not developed by the ASP provider and thus not built with multi-tenant scalability in mind (Janssen, 2008). ASP providers are often advanced network operators, not software developers. Almost all of the previously mentioned characteristics of SaaS do not hold true for ASP.

2.1.5 Delineation

To prevent confusion or discussion as to what can be considered SaaS and what can not, for this thesis we will demand the following aspects for a service to be SaaS:

1. The software application must be developed and used as an online application (thus be accessed through a browser), without the need of other technologies such as Java applets\(^9\).

2. The software application must at least have the characteristics of a level 3 SaaS application, i.e. multi-tenancy. As such, the end-user must be able to set up a new instance of the software, without human assistance of the provider. This separates it from ‘traditional’ online software applications which are installed on a per user basis.

3. Transparent versions and pricing without the need to register.

---

\(^8\)Basecamp is a SaaS project management tool. It can be found at: http://www.basecamphq.com/

\(^9\)Java applets are sometimes used to run a traditional software application through a browser
These criteria will be used in the selection of cases in chapter 4.

Having analysed the details on SaaS, the next section will look into the differences between proprietary and open source software.

2.2 Software development: Proprietary versus Open Source

The goal of this thesis is to identify the differences business models of open source software and proprietary software. To be able to conduct this research thorough understanding is needed of both. In the following sections the difference between these worlds will be discussed. The term proprietary software is used instead of commercial software, since it is also possible for a commercial organisation to make profit of open source software (as will be seen in further sections), adding ambiguity to the term ‘commercial’ in commercial software.

2.2.1 Proprietary Software

We start with proprietary software since it is often most common to the reader. Proprietary software is software owned by organisations or individuals. Usage and distribution is restricted to the terms of the owner. The owner/developer is protected by copyright and in some cases by software patents (Tirole, 2003). This type of software is commonly sold in stores or online. Microsoft and Adobe are examples of organisations whose income rely on commercial software like Microsoft Windows, Microsoft Office, Adobe Photoshop and Adobe Reader. Proprietary software can also be distributed for free (like Adobe Reader), but the source code is not made available to the public and making changes to the software is often impossible and prohibited.

The development of the software is done within the organisation itself. This includes designing, programming and testing of the application. After thorough tests the organisation sometimes releases a public beta\(^\text{10}\), in the other cases they immediately release the software and fix bugs by releasing updates or patches. Some companies are open to feedback and incorporate this feedback in future versions, but as the source code is not available, it is impossible for users to suggest specific improvements on the code.

Some proprietary software developers have enabled external developers interact with their software through a Application Programming Interface (API). An API consists of functions and methods which can be invoked from outside the software, making it possible to build additional software which makes use of (specific functions of) the original software (Evans, Hagiu and Schmalensee, 2006, pp. 27). This opens some doors to external developers, but keeps the source code secret. As proprietary software developers provide an API, their audience changes from a single-market (the end-user) to a dual-market (the end-user and the external developers) (Evans et al., 2006, pp53-55). The dimension of the market has effect on pricing strategy, which will be further discussed in § 3.5.1.

\(^{10}\)See for example Internet Explorer 8: http://blogs.msdn.com/ie/archive/2008/08/27/internet-explorer-8-beta-2-now-available.aspx (last visit: 15-10-2008)
2.2 Software development: Proprietary versus Open Source Domain

2.2.2 Open Source Software

Contrary to proprietary software, the source code of open source software is available to the public. Everyone is free to download, use and change (or contribute) to the software, in some cases under certain restrictions. These rights and restrictions are described in the various open source licences which we will discuss below. The Open Source Initiative is the official body recognised by the open source community to approve licences as conforming to the Open Source Definition (Coar, 2006). The development of open source software is organised over the internet, with many individuals contributing for different motives.

The most often used example of open source software is Linux. In landmark article Cathedral and the Bazaar Raymond (1999) used Linux\(^{11}\) and fetchmail\(^{12}\) as examples to advocate why open source development is a good alternative to proprietary software development. Linus Torvalds, leader of the Linux kernel project, is credited by Raymond to have invented the open source development process.

Open source is growing in popularity as the software matures and governments push the use of open source and open standards\(^{13}\). A recent survey conducted by the National Software Industry Survey (2008) shows that 75% of the software companies have some open source activities, which was only 13% in 2000. Little under 50% of the software companies have open source components in the offering (Helander, Rönkkö and Puhakka, 2008). Another research (Bonaccorsi, Piscitello, Merito and Rossi, 2006, p333) in five European countries (Finland, Germany, Italy, Portugal, and Spain) shows that over 30% of the interviewed software firms supplied open source software.

Licencing

There are many different licences available for open source software. In most cases these licences are introduced to prevent parasitism, such that commercial organisations can not copy the source code and sell it to its customers, without contributing to the project.

The GNU\(^{14}\) Public Licence (GPL) is one of the oldest open source licences first written by Richard Stallman of the Free Software Foundation in 1989. A second version was introduced with GNU/Linux in 1992 (Henley and Kemp, 2008). After lots of debate, recently the third version (GPL v3) was released (on 29 June 2007)\(^{15}\). The GNU licence is best known for its copyleft as an opposite to copyright. Copyleft guarantees that works derived from a GPL project should always have a licence with the same rights as a GPL licence would give. A less restrictive variant is the GNU Lesser GPL (LGPL) licence, which was made for situations where, for example, open source drivers had to be used by and included with proprietary software (Henley and Kemp, 2008, p80).

---

\(^{11}\)Linux is an open source operating system initiated and maintained by Torvalds.

\(^{12}\)Fetchmail is a open source mail client developed by Eric Raymond.

\(^{13}\)Many governments have programs that stimulate the use of Open Source and Open Standards, e.g. OSOSS in The Netherlands and COSS in Finland. In (2002) the EU recommends governments to adopt open source and exchange experiences Schmitz and Castiaux.

\(^{14}\)A recursive acronym for: GNU’s not unix (Henley and Kemp, 2008)

\(^{15}\)http://www.fsf.org/licensing/licenses/gpl.html
Other popular Open Source Initiative (OSI) approved licences are the MIT and BSD licences, which we will not discuss here. A full list of available licences can be found at the OSI website\textsuperscript{16}.

**Project management**

An open source project is often controlled by a small group, or in some cases a single person (like Linus Torvalds in the case of the Linux kernel) (Raymond, 1999). In case of a commercial organisation releasing its software as open source, this organisation in most cases stays in control of the development road map. This does not mean the development process remains unaltered, as the next paragraph will show, the project management has to take into account the external developers which will start participating.

**Development**

As everyone is free to contribute to open source projects, the development trajectory differs from the commercial situation. This is an important notion to make when an organisation considers to release its software to the open source community. The organisation has to focus on two sides of the market (a dual-market, as with the API interface of proprietary software): its direct clients remain the same, but on the other side developers will want to interact with the organisation (if they retain the project management). Not serving the developers could result in mutiny of these developers, which could textitfork the project. When forking a project a copy of the current version is made and a new open source project is started with this version, including a new group in charge of project management. If this happens to the open source software project of a commercial organisation they loose all control over the forked version and have in fact created a competitor with their own software.

**Tap into the motivation of developers**

What motivates developers to participate in open source software projects? This question has been the debate of many research articles (i.a. van Wendel de Joode, 2005). If an organisation releases a software project to the open source community with the intention to find interested external developers, it needs to appeal to the motivations of those developers.

Tirole (2003, pp. 214) identifies a *signalling incentive* which consist of a career concern incentive (job offers) and an ego gratification incentive (peer recognition). This signalling incentive is stronger when there is a large ‘audience’ consisting of other programmers. The attraction of programmers depends on the faith in a project.

**Commercial exploitation**

This thesis focuses on commercial organisations that intend to make profit on open source software. There has been a host of research projects on the question how to commercialise open source software (Bonaccorsi, Piscitello, Merito and Rossi, 2006; Fosfuri, Giarratana\textsuperscript{16}http://www.opensource.org/licenses/alphabetical
and Luzzi, 2008; Hawkins, 2004b; Hecker, 1999). Well known (and often cited) examples of commercial open source organisations are MySQL and Red Hat. Such companies do not sell licences of the product, but offer support and tools around the software. In “Setting up shop” Hecker (1999) proposes eight different strategies on commercialising open source software. The main strategies are to exploit intellectual property or deliver service. In case of intellectual property, other assets than the open source software itself have to be used. Examples are trademarks (as RedHat does), add-ons or complements to the software (which in turn are closed source), books (manuals), and hardware specifically built for the software. Examples of service delivery are installation, support, integration with other systems or customisation of the software itself. A specific revenue model built around service provisioning is Software as a Service (SaaS), which we discussed in § 2.1. SaaS is ‘compatible’ with both proprietary and open source licences, as the periodic fee is paid in return for service, not the software licence. We will further explore the combination of SaaS and open source software in § 2.3

**Drawbacks**

One of the most heard complaints on open source software is that it is not designed with a non-technical user in mind, resulting in software with bad usability (Nichols and Twidale, 2003). This is explicable when the motivation of open source developers is taken into account as stated by Raymond (1999): “Every good work of software starts by scratching a developer’s personal itch”. The problem is that the developer is a very technical user, solving a problem for himself, not taking into account the domain of a non-technical user (Nichols and Twidale, 2006; Levesque, 2004). Commercially backed projects (like Ubuntu and FireFox) tend to pay more attention to usability, which shows from their market share numbers.

**2.2.3 Vendor Lock-in**

A characteristic of (proprietary) software is vendor lock-in or path-dependency. This happens when an individual or organisation decides to use a certain software package. After time, it is forced to use the same software package (or a newer version) since all documents are stored in a proprietary format (or a ‘open’ format not supported by any other vendors) (Tiemann, 2006). Platform choice is another example of vendor lock-in. The choice for a specific operating system is difficult to reverse, as the subsequent software packages are not always compatible with other operation systems (Evans et al., 2006). System integration could also be a cause of lock-in: When different software systems are integrated, it is difficult to change vendor, since this affects all software packages integrated.

From the perspective of the vendor this is very beneficial, but as vendor lock-in has received more negative attention lately, vendors are pushed to use adopt open standards, which we will discuss next.
Table 2.1: Open Source software vs. Proprietary software

<table>
<thead>
<tr>
<th>Open Source software</th>
<th>Proprietary software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free usage</td>
<td>Paid usage</td>
</tr>
<tr>
<td>Source code available</td>
<td>Source code protected</td>
</tr>
<tr>
<td>Volunteers contribute</td>
<td>Employees contribute</td>
</tr>
<tr>
<td>Loose structure</td>
<td>Hierarchical management</td>
</tr>
<tr>
<td>Market share = developers</td>
<td>Market share = cash flow</td>
</tr>
<tr>
<td>Lack of Vendor lock-in</td>
<td>Vendor lock-in</td>
</tr>
</tbody>
</table>

Open Standards

Vendor lock-in is not specific to proprietary software and could also happen in cases of open source software. The solution lies in open standards: If vendors agree on a format to store documents in, or how to exchange information between applications, the end-user can change vendor at any moment (Tiemann, 2006). These open standards are of lesser interest to commercial vendors, as lock-in results in ‘loyal’ customers. However, as their clients are becoming more aware (and cautious), they demand applications using open standards for data storage and interchangeability, forcing vendors to adopt open standards.

In the case of open source this is different, if an application does not support an open standard, this still is not as prone to vendor lock-in as proprietary software. First off, anyone can study the source code and thus observe in what format data is saved. The next step is to either build a converter and convert the data to an open standard format. The other option is to extend the source code to support open standards. Although the above examples require technological knowledge, it is not as closed as proprietary software can be (Tiemann, 2006).

The lack of vendor lock-in is one of the key advantages of open source and open standards as opposed to proprietary software. For this reason the EU not only stimulates open standards, but also open source software. For vendors which specifically target at government organisations this is reason enough to release their software to the open source community.

2.2.4 Conclusion

This section focused on the differences between open source and proprietary software. The main differences between open source and proprietary software are summarised in table 2.1. This concerns mostly the traditional differences between open source communities and commercial organisations. For this thesis however, we are interested in commercial organisations offering SaaS using open source software. The next section will elaborate on this specific combination.
2.3 Commercialising Open Source

This thesis is not evaluating open source communities in general, but specifically open source projects lead by commercial SaaS providers. For this reason this section looks at the different strategies used by such SaaS providers to commercialise open source software.

Van der Pluijm (2007) identifies SaaS as a suitable method to commercialise open source software. As the customer is charged for the service delivered (consisting of support, application hosting, data storage, etc), the underlying software can both be proprietary and open source. This means that, to a SaaS vendor, the revenue model or service offering to its customers does not change when releasing its software as open source as opposed to keeping it proprietary. This makes it an interesting field of research on the non-financial consequences for the organisation when it decides to release its software as open source.

When it comes to SaaS and open source, the SaaS vendor has two options, either it develops its own software and releases it to the public domain as open source, or it offers existing open source software to its customers. We will briefly discuss the differences;

**Develop & Release**  The first option provides most flexibility, the software can be developed with the four stages of maturity in mind, aiming at maximum scalability. With a sufficient base the software can be released as open source. As the project is ‘owned’ by the organisation it can assure the scalability during further development. Other open source developers can decide themselves whether to use a single instance for a single tenant or to serve multiple tenants, suiting their needs. The downside of this option is that competitors can also download the open source multi-tenant, scalable SaaS application and offer the exact same features as the ‘owner’. This stresses the need to focus on the delivery of high quality service and support, as this could result to be the single differentiator between two SaaS vendors of the same open source software application.

**Adopt & Deploy**  The second option makes the vendor dependant on existing open source software, but lowers the investment costs. Most web-based open source projects operate between levels I and II of the four levels of maturity, as they often started out as a project to “scratch a developers itch”, so a project commonly starts as a single instance, not designed for multi-tenancy. This implicates that for each customer an instance of the open source software package has to be installed. The vendor is dependant on the direction of the project whether its development team focuses on scalability. This does make the SaaS platform less scalable. When competing with level IV SaaS vendors they will have more overhead costs, making competition difficult.

As this thesis focuses on the business models of organisations which have released their own software to the open source, we presume an all-ready developed proprietary software product, thus deploying an existing open source product will not be taken into further consideration.
2.3.1 Multi-tenant vs. Single-tenant

The previous paragraphs assumed the SaaS software product to be developed with multi-tenancy and scalability in mind. Parties interested in the open source version will for most part be developers and end-users (which only need a single tenant version) and to lesser extent competing SaaS vendors (which need the multi-tenant version). This is an important notion to keep in mind when releasing the software as open source, since it is the competitors who are most interested in the multi-tenant version.

Having identified the characteristics of SaaS software and discussed the differences of open source software and proprietary software, the next section will look into different applications of SaaS software.

2.4 SaaS Software Applications

In the previous sections we have seen examples of various software applications. For this research we will focus on two application types, Web Content Management software \(^{17}\) and Customer Relationship Management software. We will use cases of these types of software to analyse the differences between the business models of open source and proprietary software providers. Figure 2.1 (pp. 8 shows the research framework.

2.4.1 Web Content Management (WCM)

Web Content Management (WCM) software is software to create and maintain web content, which is typically the content of a website. We have chosen to look at the WCM SaaS market, as it has been one of the first dominantly online software applications, due to the nature of the content it addresses. The SaaS variant of WCM is still in its early stages of adoption. The following sections discuss the background of WCM and discuss the website development process, to gain understanding of the context of WCMs.

Background

In the early stages of the internet, web pages often consisted of ‘plain’ HTML pages, edited with a HTML editor like Frontpage and uploaded to the web using a FTP program, requiring experience with HTML. This labour-intensive process was streamlined with the advent of web programming languages like Perl and PHP. Many web development companies developed their content management software so that webmasters could manage their websites through an online interface, without the need of technological knowlegde. Besides commercial variants also some open source systems appeared like Joomla and Drupal. These systems however still require a large amount of effort to ‘build’ a website, thus raising installation costs. This issue is addressed by SaaS web content management systems, where installation of a new website is done by non-technical users.

\(^{17}\)WCM has the specific interest of the author due to his background in web content management software.
2.5 Conclusion

Website Development Process

Although SaaS makes it possible to have a website up and running within minutes, consumers often have an extra demand that needs the input of a graphic designer: a unique look and feel. After this look and feel is designed, it needs to be translated to a template (typically HTML) and integrated with the content management system. These steps require effort of a programmer, again raising the initial costs. These templates are often tailored to the specific WCM system, thus creating a lock-in effect. An open standard for website templates could pare the lock-in effect, but such is not yet available.

WCM SaaS providers try to overcome these initial costs and lock-in effects by providing several ready-made templates the users can choose from to personalise the website. Once the user is acquainted with the service and convinced of its added value, he could decide to have a personalised template developed.

2.4.2 CRM

CRM (Client or Customer Relationship Management) software is software used by organisations to keep track of customer information. The sales department uses the software to track potential new customers and assist them with their purchase. The support department can use the CRM system to monitor the customers and support them if needed.

CRM software makes an interesting case for SaaS, as it lacks the obvious online elements of WCM. CRM traditionally is an offline application, its data often well guarded by the firewall within the organisation, as it is a valuable source of information. This implies an IT department needed to install and maintain the CRM applications. ASP providers offer the service to host these traditional CRM applications outside the organisation which can then be accessed through a remote desktop or a VPN.

Salesforce.com (founded in 1999) was the first to revolutionise this traditional software market by introducing a SaaS offering for CRM.

2.5 Conclusion

This chapter has focused on the SaaS market and identified its key characteristics which separate it from traditional application developers and ASP. Proprietary software and open source software have been described as well as CRM software and WCM software. These are the elements of our research framework.
Chapter 3

Theories and concepts

This chapter aims to identify possible differences between open source SaaS business models and proprietary SaaS business models. This requires understanding of the elements a business model consists of. These elements are found in literature on business model design and analysis. We will look at different views on business model design in § 3.1 and adopt a definition for the term ‘business model’.

For this research project the STOF model is used as a starting point for the analysis of the elements of a business model, exploring the expected differences between proprietary and open source SaaS business models. Figure 3.1 depicts how the STOF model relates to the research framework in this thesis. For each domain of a business model, we will identify differences between open source and proprietary and between CRM and WCM.

Section 3.1.2 first explains the key concepts of the STOF model. The four domains of this model will then be separately discussed using their internal designs and related literature to identify the propositions in sections 3.2 through 3.6 that will be tested in the remainder of this thesis. The propositions found for the different domains are summarised in § 3.7.

3.1 Business Model Design for Services

To design or analyse the business model of a SaaS provider we need both a definition of what a business model is and a framework to guide our analysis. The definition and the framework will be interrelated, since a framework must adopt a definition to be able to break up a business model in parts.

The term business model grew in popularity with the dot com bubble, as ways were needed to explain the “new ways of doing business” without making any money (Hawkins, 2004b). Rayport (1999) noted, “an e-business is just another business”, indicating that these businesses need to create value not only for the customers, but also for themselves. After the bubble has burst, the focus turned to making money; describing what customers value and how to make money providing them that value (Keen and Qureshi, 2006). Porter (2001) emphasises that a viable business model needs to create customer value and charge directly for it. Value is a returning element in business model definition (Hawkins, 2004a, pp. 71), but value alone does not provide us a basis to analyse business models of different
SaaS providers. A number of business models for commercialising open source are given by Goth (2005), however, these do not provide us insight in the elements of a business model. Keen and Qureshi (2006) sharpen the delineation of business model and make a clear distinction between business model and business strategy. They identify underlying concepts, but do not tend to break the business model into essential parts, probably since this is undo-able for all types of business model types equal.

As we will be looking specifically at SaaS, we are interested in business model analysis for service provisioning. The earlier mentioned STOF model does provide us the elements a business model consists of, focusing particularly at business models for mobile services (Bouwman et al., 2008). Afuah and Tucci (2001, pp. 49) have identified the building blocks of business models which are specific to the internet. Important to note is that many elements overlap between the two approaches. Ballon (2009) proposes a business model ontology closely related to the STOF model (it inherits the same background), but places it in a broader context. He claims “any ICT-related business model can be expressed as an instantiation of this basic ontology” (Ballon, 2009, pp. 223), but at the same time “It does not, however, provide a tool to assess the viability”. This is in line with what we are looking for, as we are interested in the identification of relevant elements of the business model that differ between open source and proprietary and are not (yet) interested in a comparison of viability. We will use the STOF model in our analysis as a starting point, other literature on business models will be taken into account where applicable.

Before we outline the basic elements of the STOF model, the term business model is first defined.

### 3.1.1 Business Model

The term business model is used in different contexts and uses, varying in definition. To analyse the ‘difference between business models’ a definition for business model is needed. As the STOF model is used to analyse the aspects of a business model, the definition used in literature on STOF is adopted: “A business model is a blueprint for a service to be delivered,
3.1 Business Model Design for Services

describing the service definition and the intended value for the target group, the sources of revenue, and providing an architecture for the service delivery, including a description of the resources required, and the organisational and financial arrangements between the involved business actors, including a description of their roles and the division of costs and revenues over the business actors” (Bouwman et al., 2008, pp. 33).

It is important to note that a business model is not static: It evolves over time, as it needs to adapt to a changing environment of technological innovations, market forces and regulatory changes. For this reason we will also look at influences on business models during different phases in § 3.6.

The above definition covers the four domains defined in the STOF model, which we will discuss next.

3.1.2 STOF model

The STOF model (Figure 3.2) identifies four domains within a business model. It was developed to design and analyse mobile services. As it focuses specifically on services, the model can also be applied to SaaS. We will not discuss the internal design of the STOF model in great detail, as it was developed for mobile services and it could be possible that some elements are not as important to SaaS business models, as it is to mobile service business models.

A full description of the STOF model and its origins are given in appendix A.

3.1.3 Business model domains

STOF is an acronym for the four domains required in a business model: Service Domain, Technology Domain, Organisation Domain and Finance Domain. The main goal of a business model is to create value both for customers/end-users and for the service provider. The four domains contribute to this added value (see § 3.1.1). The domains are interconnected and events within a domain also affect other domains. The service domain describes the offered service and its value proposition. The customers perception of this proposition is
3.2 Service Domain

one of the main drivers behind the output variable “Value for customers”. The technology domain describes the technical architecture behind the service, it specifies all needed technology and data to provide the service. The organisation domain describes the network of actors and their roles needed to deliver the service. The finance domain focuses on the costs and revenues generated by the business model. This domain also provides performance indicators which reflect the performance of the business and it contains financial arrangements which determine how the generated value is distributed over the involved actors.

3.2 Service Domain

In this section we will take the different aspects of the service domain (Appendix A, Figure A.1) as a starting point to analyse possible differences between open source and proprietary SaaS business models. The important output of a service is the generated value for customers, expressed in Intended Value, Delivered Value, Perceived Value and Expected Value. The Intended Value is the starting point and is expressed by the SaaS provider in the Value Proposition. Between the proprietary and open source SaaS providers, there is no reason to assume a difference between business models. In most cases, the Value Proposition is proposed in the early Technology R&D phase, a phase which is usually not performed ‘open source’, as noted in § 2.3 (an open source project often starts with an existent software package).

Afuah and Tucci (2001) have not made the distinction between expected, perceived and delivered value, but have grouped these in customer value. They assume two factors determine customer value: the ability to differentiate or the ability to compete on lower pricing. Differentiation can be done in seven fields; Product Features, Timing, Location, Service, Product Mix, Linkages and Brand-Name Reputation. For an open source SaaS provider it is hard to differentiate on several of these fields: The product features can be copied by competitors, since the source is open, but the open source provider can not copy the features of its competitors, since these are proprietary. The location is often irrelevant in case of SaaS, as the service is provided world wide over the web. To differentiate on service, the SaaS provider will have to deploy value activities that are separated from the technology, we will further discuss these non-technological value activities in § 3.4. Differentiation on the field of timing will be looked at in the context of phases of both the business model and the technology life cycle in § 3.6. When considering brand-name reputation, there could be a difference in the ‘reputation’ of open source software versus proprietary software, this will be discussed in § 3.2.2. SaaS providers do of course have the ability to compete on lower pricing, but this is not restricted to open source or proprietary SaaS providers. We will look into further detail of pricing next.

3.2.1 Pricing

Customers and End-Users have Expected and Perceived Values of a SaaS offering. Since open source is often interpreted and explained as ‘free software’, customers tend to assume that open source software is a lower cost alternative to proprietary software (McGowan,
This is often caused by not having a clear understanding on the total cost of ownership (Bitzer and Schroder, 2007). When the price of open source is equal to a proprietary product or service, consumers expect more features or higher service delivery from the open source provider, since they are not paying for the software, it must be something else they pay for. This could also be formulated the other way around: At an equal price and equal service delivery the perceived value of an open source SaaS service is lower than that of an open source service (in other words, the customer is disappointed that the open source service does not deliver more value than the proprietary SaaS provider).

Resource based view

The above is in line with the theory on the resource based view (RBV) (Barney, 1991). As open source software is not a resource which is in-imitable by competition, it can not be seen as a resource that provides a competitive advantage over competitors, so an open source SaaS provider must find other resources that do provide competitive advantage. We will discuss the focus on other value activities in § 3.4. In de perception of the customer however, the software is no longer the non-substitutable asset, so there must be another unique resource he is paying for to justify an equal or higher priced open source SaaS offering. If this resource is lacking, the provider might adhere to the strategy to lower its tariff which will increase customer value (which was one of the options proposed by Afuah and Tucci (2001) above).

This means that open source SaaS providers charge lower prices: a) to compensate for the (distorted) expectation that open source is a lower cost alternative (self-fulfilling prophecy) and b) to compete on lower pricing (as noted in previous paragraph). We will therefore define the first proposition as follows:

**Proposition 1** Open source SaaS providers charge lower prices

In line with our research framework, it is necessary to question whether this proposition will differ between CRM software and WCM software, but no ground is provided to make such assumption.

3.2.2 Context

The physical context in which the service is used is fairly pre-determined in case of SaaS: within a browser, using a PC, at work or at home. A factor playing a role in the broader socio-economic context are the governmental programmes stimulating the use of open source software and open standards and emphasise the lesser extend of vendor lock-in (§ 2.2.3). One could argue that these positive advocations of open source software contribute to the branding of the product which in turn contributes to the overall value proposition or product differentiation. However, the awareness of these governmental programmes under the public and the assumed positive effect on the image of open source software have not been confirmed as of yet and we therefor should not take these effects on the value proposition into account when analysing differences between business models, but research
3.3 Technology Domain

Of the technological domain the *software application* is the part that can be made open source. Other SaaS specific aspects of the technical architecture like *web-servers, databases, data storage, backbone infrastructure* and *billing platform* are managed by the SaaS provider and its related actors (suppliers). If a part of the development of the applications is done by the open source community instead of the provider itself, this would both add to the *technical functionality* (features), as well as reduce *costs*. How costs can be reduced by the input of external developers will be further discussed in § 3.4 and § 3.5.

Although the software application is the subject of discussion when looking at open source versus proprietary software, there should be no difference in the way the application relates to the technical architecture. The technology domain does not provide other elements of a service business model that we expect to differ between open source and proprietary business models.

3.4 Organisation Domain

The question whether to release the software as open source resides foremost in the organisational domain: It affects almost all elements within the descriptive model for the organisational domain. The STOF model stresses the importance of a network of *Actors* collaborating to deliver the service (the *Value Network*, each offering its *Resources and Capabilities* to the service). In case of SaaS there are different *Roles* in the service delivery which can be fulfilled by different actors (e.g. Hardware platform, billing, support, investors, promotion, etc). Most of these roles are however ‘replaceable’, so the initiator is not dependent on these actors, in contrast to mobile services in which the service provider ‘needs’ the co-operation of certain actors.

Collaboration with other organisations is also advocated by Chesbrough (2003) in his articles on ‘Open Innovation’. We will look into open innovation next to assess whether this delivers expected differences between open source and proprietary SaaS providers.

3.4.1 Open Innovation

Open innovation is a ‘theory’ asserted by Chesbrough (2003) and was embraced and broadened by many others (Fredberg, Elmquist and Ollila, 2008). Due to rapid technological advancement and widespread distribution of knowledge, organisations can no longer expect to keep up with competition when they solely rely on their internal research and development. Instead they should cooperate with other organisations. This is a two-way effect, they should both obtain external intellectual property (IP) by obtaining licences on patents from other companies. On the other side organisations should make their technological know-
Theories and concepts

3.4 Organisation Domain

Closed Innovation Principles

The smart people in our field work for us.

To profit from R&D, we must discover, develop and ship it ourselves.

If we discover it ourselves, we will get it to market first.

If we are the first to commercialise an innovation, we will win.

If we create the most and best ideas in the industry, we will win.

We should control our intellectual property (IP) so that our competitors don’t profit from our ideas.

Open Innovation Principles

Not all of the smart people work for us so we must find and tap into the knowledge and expertise of bright individuals outside our company.

External R&D can create significant value; internal R&D is needed to claim some portion of that value.

We don’t have to originate the research in order to profit from it.

Building a better business model is better than getting to market first.

If we make the best use of internal and external ideas, we will win.

We should profit from others’ use of our IP, and we should buy others’ IP whenever it advances our own business model.

Table 3.1: Differences between closed innovation and open innovation. (Chesbrough, 2003)

As with the STOF model, the business model built around the technology is of large importance in open innovation. Where in ‘closed innovation’ an innovation is brought to market as soon as possible, open innovation focuses on building a better business model before releasing the technology (Chesbrough, 2006). Table 3.1 shows the key differences between closed and open innovation according to Chesbrough (2003).

Although the term ‘open innovation’ may sound as open as ‘open source’, this is far from true. The term ‘open’ was used to oppose it from the ‘closed’ business models in which everything was researched and developed internally. Whilst collaborating with other organisations, open innovation has a strong focus on protecting intellectual property, which we will address after having looked into the combination of software and open innovation.

Software & Open Innovation

The characteristics for a market where open innovation is advised show large resemblance with the software market: High rate of new product introductions, growing investments in R&D and short product life cycles. Although software can not be patented in Europe (European Patent Office, 2000) copyright gives software developers enough possibilities to licence their products. To implement open innovation, organisations need to join forces to innovate, or in software terminology, software developers need to share source code to create innovative software (possibly protected by a non-disclosure agreement). A possible approach is buying or selling (licensing) software components to others. Incorporating other innovations through web services is also a form of open innovation. However the
3.4 Organisation Domain

Theories and concepts

most open form of software development is open source. This section will look into the
relations between open innovation and a commercial open source project.

What happens if the source code is shared with everybody as is the case with open source
software? Chesbrough (2006, pp. 43) asserts “There is no inherent value in a technology
per se. The value is determined instead by the business model used to bring it to market.”,
thus the challenge no longer exclusively lies in the development of the software, but equally
in finding the right business model around the open source software. The business model
has to focus on exploiting all other IP the organisation has to offer: its brand, the customer
base, know-how and trade tricks. SaaS, as discussed in § 2.1, does put the focus on these
assets and provides a revenue model around the software, whether it is open source or not.
As such, SaaS overcomes the paradox identified by West and Gallagher (2006, pp. 327-328)
that open source companies have to create revenue over something that is ‘free’. To better
understand the consequences of this focus on other IP assets, better understanding of IP is
needed. The following paragraphs further elaborate on the management of IP.

Intellectual Property (IP)

In open innovation IP plays an important role. In the past organisation did most of the R&D
internally and some of the innovations were brought to market. Many innovations and
patents never left the building and were kept ‘on the shelf’. Where ‘closed’ organisations
were hesitant to let any innovations fall in the hands of competitors, the open innovation
model values innovations as worthless when not exploited. This means an innovation must
either be used internally, or licensed to outsiders to be of value to the organisation. Besides
from developing innovations internally in the R&D department, innovations can also be
obtained from outside the organisation. Value can be added through the combination of
different innovations. Chesbrough (2006) foresees IP trading markets and intermediaries
that facilitate exchange of IP, as companies become more open and start exchanging IP.

In this light it might seem unwise to give away the most important IP asset (the software)
away for free. Nevertheless this is used as an example of open innovation in many occa-
sions throughout open innovation literature (Chesbrough, 2003; Chesbrough, 2006), often
using IBM as an example. Asserted is the need to have knowledge and trade-tricks about
the open source software as the alternative asset which can be made to value. Chesbrough
(2006) gives the same options as we saw in chapter 2, § 2.2.2; service and support, differ-
ent versions of the software, customisation and integration of the software and providing
proprietary complements.

That said, it is important to note that the GPL licence (see § 2.2.2) has incorporated
several clauses to prevent patents from making it proprietary (for those countries that do
allow patents to restrict development and use of software). The GPL licence i.a. states “Each
contributor grants you a non-exclusive, worldwide, royalty-free patent license under the
contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise
run, modify and propagate the contents of its contributor version.”. This protects the initiator
of the open source project from outsiders claiming a patent on their contributions.

As also mentioned in the service domain section, open source software no longer pro-
vides the inimitable resource protected by intellectual property rights. Thus the open source
SaaS provider can not differentiate itself from competition with its software (as competition can also offer exactly the same software). This means that the value activities not based on open source software are the only means for open source SaaS providers to separate them from competition. Relevant value activities in case of SaaS include assets such as knowledge, trade-tricks, brand name reputation, service quality and customer relationships.

**Proposition 2** The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition

**Governance of Open Innovation**

Incorporating external idea’s in the organisation demands overcoming of the “Not Invented Here” syndrome (Fredberg et al., 2008). Fredberg et al. (2008) argue the need for both “cognitive changes in the mindset of the leader” and “the need for new organisational structures and managerial practises”. It is also noted by them that little research has been done in this field and further research is needed.

Dodgson, Gann and Salter (2006) came across the need for cultural change to adapt to the new “open innovation mindset” when analysing the Connect & Develop programme at Procter & Gamble. This organisational transition took the company “over a decade”. A second conclusion was the notion that external innovations do not replace the need of internal R&D practises.

Although Fleming and Waguespack (2007) uses a different definition for open innovation (taking IP out of the concept), he did identify some interesting characteristics of leaders within open source communities not related to any commercial activities, which might be of use for organisations which plan to release a project to the open source community. To become a leader within such communities both technological and social skills are needed in the form of important technological contributions and collaborative relationships (Fleming and Waguespack, 2007, pp. 178).

Chesbrough (2003, pp. 39) identifies four different types of actors within a value web that generate innovations, of which the innovation architect is the role of the organisation that initiate the software project. They have to fit the pieces of code together and develop a strategy towards a valuable end-product. “Innovation missionaries consist of people and organisations that create and advance technologies to serve a cause.” This shows resemblance with the open source developers, who, as noted before, seek solutions because “it scratches an itch”. Whilst not exclusively referring to open source, Chesbrough (2003, pp. 39) asserts that the architects seek financial profits from their work and missionaries are motivated by their mission.

The above findings indicate that a difference could exists between open source and proprietary SaaS providers in the way the innovation process is managed. The external developers of an open source project might add to the Resources and Capabilities of the value web, but they also add to its complexity (more Organisational Arrangements). As Actors have their own Strategies and Goals, governance of open source software requires more resources of the SaaS provider (i.e. the innovation architect, initiator) compared to proprietary software. This is also includes the necessary internal efforts, mentioned...
above, to incorporate a more open mindset. In the end this would possibly increase the costs made by the SaaS provider.

**Proposition 3** Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project

As also indicated in the Technological Domain and in the previous paragraph; the contributions of external developers add to the technological architecture by providing bug-fixes, improving the software and developing add-ons. If Resources and Capabilities are scarce in terms of availability of developers, the availability an open source community as additional ‘actor’ could be very beneficial and could reduce cost. This reduction of cost is influenced by various factors that will be discussed in the Financial Domain.

### 3.5 Finance Domain

The activities in the other three domains have their impact on Costs and Revenues in the finance domain (see Appendix A, Figure A.4). As noted in the organisation domain, open source SaaS providers could benefit from external developers, reducing cost of developing the Technical Architecture. As discussed in § 2.2, external developers can be seen as a second target group for the organisation. The next section will look into this phenomenon also known as dual markets.

#### 3.5.1 Dual Markets

A dual market (or multi-sided market) is used to describe the situation where an organisation has two types of clients to which the value of the offering depends on the size of the other client base. An example: A credit card company needs both card-holders and shops. Without card-holders, shops are not interested in accepting the card. Without shops that accept the credit card, no consumer will be interested in the credit card. This common ‘Chicken and egg’ problem (or ‘indirect network effect’) has to be addressed by the company in order to make it a success. This involves the pricing structure: which customers should pay for the service: the card-holder or the acceptant?

**Dual Software Markets**

This multi-sided characteristic is also seen in software markets (Evans et al., 2006). Imagine an operating system without any compatible software applications or a game console without any games, they would be of little value to the consumer. This also works the other way around: without consumers these software applications and games would be of no value. These two-sided products are often called software platforms as they form the basis for an environment of software applications (Evans et al., 2006; Economides and Katsamakas, 2006). A good example is the Windows operating system.

The applications built on top of software platforms (such as Microsoft Word built on top of Microsoft Windows) do not necessarily have these dual market characteristics. This
also holds true for SaaS software which is often developed by a single company offered to a specific target group. The end-user of a CRM or WCM SaaS provider does not need additional software applications to be able to benefit from these services.

Software platforms are enabled through an application programming interface. This enables external developers to create applications which extend the functionality of the original application. When a vendor offers an application programming interface (API) to its software, one could speak of a dual market: its clients on one side and the external developers on the other. The developers will only be attracted if there are sufficient end-users that will be interested in the software.

Open source software projects have a form of duality by nature, having developers on one side and end-users on the other. Developers are attracted by a large user base, users are attracted by an active developer community which develops features. The open source project team (whether this is a commercial organisation or not) has to appeal to the needs of both groups in order to make the project a success. This is an implication for commercial organisations which are considering to release their software as open source, the organisation has to anticipate to offer service to the open source developers. This effort has to be taken into account when considering to release the software as open source software and underlines the earlier mentioned proposition on the governance of open source software (§ 3.4.1).

It depends on the type of innovation (in this case type of software) whether it is of interest to others to innovate on top of this innovation to establish a platform (Wonglimpiyarat, 2004). The rubik’s cube is an innovation which can not be extended to a platform, as there is no basis for further extensions or products. Windows OS has shown to be an innovation which provided a good basis for platform development, as the success of it and its many compatible products have proved. Therefore, an important question to ask before building an API interface (or before releasing software to the public) is whether the software product is capable of forming a platform for further technological innovation? If this is not the case, the software vendor will not profit over the effort needed to build the API. Neither will there be a solid ground for an open source version of the software.

Due to the similarity in platform prerequisites (an API versus an open source release), the existence of a community of developers around an API (either at the provider itself, or with a competitor) is an indicator that there are developers interested in the software when this is released as open source. Of course the existence of an similar open source software product is also an indicator there is enough interest. In the cases of CRM and WCM different examples of applications can be found that have shown to provide a platform (of our cases Wordpress, Salesforce and SugarCRM offer many plugins and extensions). This takes away the concern that one of these software applications might not be able to provide a platform. This also means it is difficult to analyse the correlation between the success of API’s and open source projects within a market segment. This is a possible research subject for future research.
3.5 Finance Domain

Pricing strategy

A product (or platform) with a dual market has the possibility to charge these markets independently. Such a two-sided pricing strategy increases profits in cases where there is no subsidies are made to one side, compared to a situation where only one side is charged (Economides and Katsamakas, 2006).

A two-sided pricing strategy is difficult in case of open source software: developers are often volunteers and can not be charged (Economides and Katsamakas, 2006, pp. 1058). The whole idea behind open source is free access to the source, chances are that developers are not willing to pay anything, but expect support and feedback on their work (as noted in § 2.2.2).

The other side, the consumers (or end-users), can not be charged for the a licence of the software as the open source licences prohibit this, but they can be offered support and service in return for money. So although an open source SaaS provider has a two sided market (developers and consumers), it has a single-sided pricing strategy, effectively even subsidising the developers. The developers deliver complements and contributions to the open source software, enlarging the value to end-users in return.

For SaaS providers, the pricing structure does not change when releasing their software as open source. In both cases end-users pay for the service offered, not for the software licence. They might be forced into lowering their overall price (see the first proposition), but the developers do not provide a new revenue source.

3.5.2 Cost

The key difference between a closed source and an open source SaaS-provider lies in the cost structure of the software development. In case of closed source, the software is developed in-house and the development process consists of wages and related costs. External developers can only be obtained by providing an API, which again takes development costs. Even when providing an API, the external developers can not contribute to the core of the application, as this source remains closed, so the developer will always be the only one developing the core of the application. So providing an API does not lower cost, but elevate costs in return of added functionality (if there are sufficient developers).

In the case of open source costs are made to support and manage the open source development process (see § 3.4.1). On the other side the costs are reduced as volunteers add to the development of the software (Wilson and Kambil, 2008). As noted in the organisation, technology and finance domain, the impact of the cost reduction depends on different factors, such as the scarcity of developers, the quality of the contributions and the ability to build a open source community (the software platform). Proposition 4 is the result of this analysis. We will not introduce the ‘higher cost for management of the open source process’ as a separate proposition, as this is the direct result of proposition in § 3.4.1.

Proposition 4 Contributions of volunteers lower development costs for an open source SaaS provider
As argued, the lowered cost will only be possible if there is sufficient interest from external developers in the form of a software platform. For both WCM software and CRM software there have been plenty examples of open source software packages and API contributions to assume that this cost reduction must be possible in both cases.

3.5.3 Risk

The finance domain also indicates that the Market Segment the service is targeted at co-determines the Revenue Sources. As noted in § 2.1, for most SaaS applications an off-line version exists. In case of SaaS, the service is aimed at customers who do not have the capacity/intent to run the software internally. The advantage of SaaS is the outsourcement of the hardware, software and maintenance. With this in mind, users wishing to install and maintain an open source version of the SaaS application for personal use are not part of the Market Segment the service is targeted at, thus should not be interpreted as a missed Revenue Source or as opportunity costs, thus not be interpreted as a risk of releasing the software as open source and as such no proposition will be formulated.

Beside the case above, it is also possible that a competitor will start offering a similar SaaS service using the open source software (mentioned in § 2.3). This is a Risk which does threaten Revenues, as the competitor could charge lower prices or offer better service, without having development costs. This risk contributes to the earlier made proposition that the non-technological value activities are expected to be of greater importance to open source SaaS providers. A crucial part to a SaaS provider is the software that enables it to scale to unlimited tenants (see § 2.1.2. To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed. This would force competitors to develop this crucial part for scalability themselves. Developers or end-users who are interested in a single instance will not miss this software. As noted in § 2.2.2 open source developers often contribute because they are also a user of the software themselves, so in most cases they are not looking for scalability.

Proposition 5 To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed

A potential third element of Risk can be found in the consequences of the source code made public: Everybody can examine the source code and look for vulnerabilities which can be abused. This risk is reduced by the many volunteers who look for vulnerabilities to be solved. As with the cost/benefit of external developers, the risk of exploiting vulnerabilities depends on the size of the developers community. This risk has already been the subject of prior research; Analysis of software packages in the past has shown that this balance is in favour of the open source software, in which security leaks are found and addressed quicker than in proprietary software (Payne, 2002). No proposition will be formed for this element of risk, as it has already been studied.
3.6 Phases of a Service Business Model

The STOF model uses three phases within the life-cycle of a business model in which the business model adapts to external influences such as technology, market and regulation (see also § 3.1.3 and figure A.5). An external influence that has been mentioned under Risk (the competitor ‘copy-cat’) is the market: especially during the ‘Technology R&D’ and ‘Roll out’ phase, the risk imposed by a competitor offering the exact same service will have a substantial impact on the Revenues. During these phases the SaaS provider is still working on its other IP assets like brand identity and user base. The best phase to release software as open source is therefore within the market phase, when the provider has build up some measures to stand up to competition.

The phases noted in the previous paragraph concern the phases of the business model itself, not the phases of the current technology (this is one of the external influences). Software products evolve through a technological life cycle (figure 3.3) consisting of a emerging, growth, maturity and decline phase (Afuah and Tucci, 2001; Chesbrough, 2006). During the emerging and the beginning of the growth phase, there are relative little competitors of a technology. The product or service is considered ‘High Tech’. As the technology progresses along the growth phase it becomes more of a commodity with competition growing and features adding up (Treloar, 1999).

Afuah and Tucci (2001) have called the capability to maintain its competitive advantage the Sustainability of the business plan. They advise three different strategies; the block strategy which could be seen as the closed business model of Chesbrough, a run strategy which aims to keep innovating quicker than competition and a team-up strategy which shows large resemblance with open innovation and could also be interpreted as open sourcing.
the software in our case. When related to the technological life cycle, growth is a phase where the decision should be made whether to team-up or to block (Afuah and Tucci, 2001, pp. 73-75).

In case of a SaaS provider: If it enters the life cycle at an early stage (emerging phase), releasing its software as open source would commoditise the technology, giving away the advantage of having a disruptive technology. For such organisations it would be wise to wait until the technology has entered the growing stage to consolidate its position during the emerging phase. This results in our final proposition:

**Proposition 6**  
*Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that the competitive advantage is lost*

On the other hand, if the SaaS provider enters a more mature market where growth is stalling, it will have to compete on network externalities (Katz and Shapiro, 1994). This means it needs to gain market share in order to profit from the benefits of a large user base, while it competes with organisations that already have this advantage. We assume that an open source software project is capable of attracting a user base quicker, since it can be adopted by both the users within the market segment it is targeted at, as well as other users and developers not interested in the service, but in the software. Although the non-paying users and developers do not add revenue, they do add to the Value Network. They add contributions to the software and contribute to the brand-name reputation of the software (rumour around the brand), which in turn adds to the customer value (Afuah and Tucci, 2001). So if the technology is already in the growth phase, it should be considered to release the software as open source even before the market phase of the business plan and include open source developers in the development phase. This entry strategy has been thoroughly studied by Bonaccorsi, Giannangeli and Rossi (2006) and although this study was not limited to SaaS providers, it does prove to have been a successful strategy for many open source software developers. We will therefore adopt this entry strategy as a recommendation towards SaaS providers which are considering open sourcing their software.

### 3.7 Propositions

The above review of the STOF model for the SaaS market has given several expected differences between the business models of open source and proprietary SaaS providers. To be able to verify whether these differences exist in practise, variables have to be defined which can be tested through case analysis. The propositions that have been formulated throughout this chapter are summarised in table 3.2.

### 3.8 Conclusion

The analysis in this chapter has provided us the necessary foundation to be able to perform a case study. We have found six propositions which will be tested in these cases. The questions formulated per proposition will provide the necessary guidance in search of the probability of the propositions. The next chapter will describe the research method and the
### 3.8 Conclusion

**Theories and concepts**

Table 3.2: Overview of proposition

<table>
<thead>
<tr>
<th></th>
<th>Open Source SaaS providers</th>
<th>Proprietary SaaS providers</th>
</tr>
</thead>
</table>
| **Pricing** | Proposition 1. Open source SaaS providers charge lower prices  
- Compensate for expectation of customer  
- Compete on lower pricing  
*No proposition within technology domain* | - No need to compensate expectations  
- Compete on software features |
| **Value Activities** | Proposition 2. The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition  
- Software features are copied by competition  
- Focus on knowledge, trade-tricks, brand name, service | - Features can only be replicated, takes more time  
- Focus on capabilities of software |
| **Network complexity** | Proposition 3. Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project  
- Overcoming 'Not invented here' syndrome  
- More management, less internal development | - Only internal developers need to be managed  
- Development team builds up more experience |
| **Cost** | Proposition 4. Contributions of volunteers lower development costs for an open source SaaS provider  
- Volunteers provide bug fixes and contributions to core  
- Cost reduction depends on quality and number | - All development done internally, requires more developers  
- Quality of code determined by level of developers |
| **Risk** | Proposition 5. To protect themselves from competition open source SaaS providers will keep closed the source of the software that provides scalability  
- Competition can not offer identical service  
- Does not affect end-users outside target group | *This proposition only concerns open source providers*  
*This proposition only concerns open source providers* |
| **Risk** | Proposition 6. Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that competitive advantage is lost  
- Competition can copy technology  
- Little other resources to gain competitive advantage | |

Selection of the cases. We will indicate where to obtain the information needed to be able to answer the questions of the propositions.
Chapter 4

Methodology

4.1 Research Objective and Approach

The previous chapters have provided insight in the SaaS market and defined the different fields of our research. The goal of this thesis is to analyse the impact on a business model of the decision to release a proprietary software product as an open source product. SaaS is used as a starting point for this analysis since both open source and proprietary software can be commercialised using this revenue model. This gives us a partial *ceteris paribus* opportunity to focus on all other consequences of the transition from closed to open source, without having to account for adaptation of the revenue model. Using the STOF model and related theories as open innovation and dual markets we identified expected differences (propositions) in the business models between open source SaaS providers and proprietary SaaS providers.

These propositions will be tested to assess whether these are indeed the differences one has to take in mind when considering to release proprietary software to the open source community.

4.2 Research method

Different research method are available, such as case studies, experiments, surveys, multiple histories, and analysis of archival information. For the selection of a research method, one has to look at the context of the phenomenon under study. If it is impossible to isolate the phenomenon from its real-life context, a in-depth case study is often applied (Yin, 2002). Case studies are often performed in situations where the number of known cases is limited and the goal is to gain understanding on ‘how’ and ‘why’ the cases are what they are (Yin, 2002). Case studies rely on multiple sources of data and often specifically select the cases instead of using a random sample as is the done in experiments and surveys.

For this thesis we are interested in a deepened understanding of the business models for both proprietary and open source SaaS providers. Although we are not explicitly answering ‘how’ or ‘why’ questions, we are not only looking whether the formulated propositions exist in our real life cases, but also trying to understand why they exist or not. As a business
model is influenced by external factors (Market, Regulation and Technology, see § 3.1.3),
the real-life context is of great importance. Additionally, case studies are especially useful
when the subject is complex and has not (often) been researched before (Yin, 2002), this is
also applicable to the present study.

An important type specification of case studies is the distinction between single and
multiple-case studies. The evidence of multiple cases is often considered more convincing,
and the overall study is therefore regarded as being more robust (Yin, 2002). For this thesis
the cases are first examined individually. The results of the proprietary cases are compared
with the open source cases afterwards, which results in confirmation or rejection of the
propositions of § 3.7. This is also known as a hierarchic comparative case study (Verschuren
and Doorewaard, 2005, pp. 167).

Contrasting results are expected between the two CRM cases and between the two
WCM cases (theoretical replication) (Yin, 2002), similar results are expected for the open
source cases and for the proprietary cases (literal replication).

4.3 Case Protocol

The case protocol is an instrument used to structure the case research and maintain consist-
ency between case studies, increasing the reliability of the case study as a whole. We will
first discuss the criteria for the selection of cases, followed by general procedures to take in
mind during the study. Next the case questions will be linked to sources of information and
finally an outline for the case study reports is given.

4.3.1 Selection Criteria

The cases need to conform to certain selection criteria to ensure the robustness of the case
study analysis results. The criteria are both based on the aspects of SaaS and different
aspects of open source versus proprietary we explored in chapter 2.

SaaS All cases must be providers of SaaS. This is important since this implies an import-
tant aspect of the business model: it is offered as a service, it is not sold as a product. The
customers should pay a periodic fee and should not have to pay a one-time licence fee. Other
criteria are incorporated from § 2.1.5 where we have determined what is considered ‘SaaS’
for this research. This requires the software to be web-based and multi-tenant capable.

Research framework In chapter 2 we have developed a research framework consisting
of two axes: Proprietary versus open source and customer relationship management (CRM)
software versus web content management (WCM) software. We will choose a case for each
of these combination, resulting in four cases (figure 4.1).

Similar business model elements As we are interested in comparing the results of the
open source cases to the results of the proprietary cases, we should select cases which are
similar on the aspects of those elements of the business model that we do not expect to differ
between open source and proprietary SaaS providers. An example is the market segment the service is targeted at or the technical functionality the service provider offers.

**Other Characteristics** Although it would be convenient when cases are located in The Netherlands, this will not be a leading criterium as it is of greater importance that the other criteria are met.

### 4.3.2 Case Selection

The following paragraphs will look at the four different fields and select a case accordingly. We will take the selection criteria in mind that we have formulated above. We will first look at the open source cases, since these are scarce. We can than find appropriate proprietary cases that match those business model elements that we do not expect to differ, such as the market segment.
Case 1 - Open source WCM

Contrary to proprietary WCM providers, the number of open source SaaS WCM providers is limited. So far three different SaaS providers have been identified:

1. DotCMS onDemand
2. Concrete5
3. Wordpress.com

The difference between these providers is large: DotCMS aims at large organisations and offers a SaaS version of its software (called onDemand) ‘on the side’. They provide little information on their SaaS version and do not provide a trial version. This suggests their intend is to profit from the popularity and attention to SaaS rather than that this is their true business model.

Concrete5 on the other hand shows the typical characteristics of a SaaS application. Set-up is quick and easy, there is a trial version for 15 days and the monthly fee is acceptable. They also display an active open source community, which shows that they indeed try to benefit from external developers and try to profit from the network externalities provided by a large user base. Concrete5 was not build as open source from the start, and as such would provide an interesting case to our study. The only concern is how they position their service: they explicitly call it ‘hosting’. There are no other services offered and they do not position themselves as a provider which takes care of all technological issues. This is in contrast with SaaS, were the provider is expected to be a full-service provider.

Wordpress is the largest provider, and has a huge installed base (both at wordpress.com - the SaaS version - and many downloaded copies of the open source version. A lot of information can be found on Wordpress.com and it has a broad portfolio of service offerings for their software. It adheres to all the criteria required for the cases. Their service is targeted at both individuals and at organisations. An important feature of the software is its ability to provide a weblog.

Wordpress best suits the prerequisites and thus was chosen as the case for the Open Source WCM.

Case 2 - Proprietary WCM

The number of proprietary SaaS WCM providers has grown significantly in the past years. Most of them aim at the small business market, advocating their low cost solution to a content management system for their website. On the other hand the traditional WCM providers have tried to adapt their systems to a SaaS version. These are however easy to filter out, as they are often obscure on pricing information and do not offer instant set-up of an instance (which is a pre-requisite for SaaS). A large player on the WCM SaaS market

---

1 DotCMS onDemand can be found at http://dotcms.org/onDemand/
2 Concrete5 can be found at http://www.concrete5.org
3 Wordpress can be found at http://www.wordpress.com
4 Examples of such large players are Crownpeak and Clickability
that does conform to the prerequisites that we have formulated in § 2.1 is Squarespace\(^5\). The target group shows similarities with Wordpress (individuals and small organisations) and both services offer the ability to maintain a weblog, among other features. This fit is the reason Squarespace was chosen as the proprietary WCM case.

Typepad\(^6\) is one of the larger competitors of Wordpress. The reason Typepad was not chosen as one of the cases, is that the source code of Typepad is partially based on the source code of MovableType, an open source WCM. For this reason Typepad is difficult to analyse, as it will be unclear whether the background with MovableType will have influenced the aspects of the business model.

**Case 3 - Open source CRM**

The selection of open source SaaS CRM systems is limited\(^7\). The search for these providers is thwarted by traditional CRM providers claiming to offer SaaS, though they are in fact ASP’s, offering a traditional standalone application over a remote desktop. The exception is Sugar CRM. Although they do not adhere to a 100% SaaS strategy (besides SaaS they also offer a self-hosted version, customisation, implementation and training), SugarCRM is one of the scarce SaaS CRM providers using an open source software system. Another downside of Sugar CRM is that their software is not 100% open source, some features have been kept proprietary.

Although Sugar CRM is not the ‘perfect’ SaaS provider, they do provide SaaS in their service portfolio. Due to the large popularity of the software, there are many resources of information available. For this and the limited (or non-existent) number of alternatives we have chosen to use SugarCRM as the third case.

**Case 4 - Proprietary CRM**

The best known case for SaaS CRM is Salesforce.com, often identified as the organisation that invented the SaaS model. During the years competitors have tried to access this market, but not with large success. Since Salesforce.com has a long history in the SaaS market, it is an interesting case to our study. Another benefit is the single focus of Salesforce.com. Many other SaaS CRM providers (like Microsoft, Oracle, IBM) originate from traditional software backgrounds and often offer a broad portfolio of software products. This makes it difficult to isolate out the business model related to their CRM offering. This lead to the choice of Salesforce.com as the final case in the case study.

**4.3.3 Procedures**

This section describes the general sources of information, determination of interviewees, methodological concerns and other procedural issues involved in the case study.

---

\(^5\)Squarespace can be found at http://www.squarespace.com
\(^6\)Typepad can be found at http://www.Typepad.com
\(^7\)A search on Google for “open source CRM SaaS -sugarcrm” yields no usable results
4.3 Case Protocol

**Data Collection**

Through desk research we will try to obtain as much information as possible on the topics relevant to this research project. The second step is to confirm findings through interviews and acquire information not discovered by desk research. If it is not possible to interview respondents in real life, the interview will be held through telephone, direct messaging or email.

**Triangulation** For case study research triangulation is a method to improve the reliability of the results. This means that all findings should supported by at multiple independent sources.

**Sources** The first step in data collection is the desk and internet (re)search. Using search engines we will look for interviews, articles and blog posts about the cases. The information obtained is used to prepare interviews. The following sources can be used:

- Annual reports
- Business plans
- Scientific publications
- Press releases
- Interviews done by outsiders

As a final step interviews will be held to obtain missing information and to validate information found.

**Interviewees**

The persons to be interviewed need to have a clear view on various elements of their business model. They should have knowledge on how the business model has come to its current form and what decisions have been made in the past. Other questions concern the technological part of the organisations. It is therefore possible that multiple interviews are needed to cover all topics. Multiple interview also contribute to the triangulation noted above.

**Unit of analysis**

As we are comparing the business models of SaaS providers, the unit of analysis is the business model of a single provider. This business model is influenced by external factors, so the research unit is the company, but from the perspective of the company will we identify these external factors.
4.3.4 Case Questions

This section describes which information is needed to verify the propositions. The questions that provide this information are the end result. From chapter 3 we have adopted the following propositions. The full list of questions which are derived can be found in appendix B.

**Open source SaaS providers charge lower prices**

To analyse this proposition both insight in pricing structure and service offering is needed. If pricing indeed proves to be lower at equal service delivery for open source, the question remains whether this was a deliberate strategic choice of the provider, or that other reasons induced the price difference. Statements made by the provider that refer to the lower cost of open source software indicate such a strategic choice.

**The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition**

In a market where providers compete, they will deploy value activities to differentiate themselves from competition. To assess whether an open source provider deploys more non-software related activities, all value activities need to be inventorised. This is obtained by answering the question: *What value activities are deployed by the provider?*

Besides the activities that are deployed, it is also assumed that an open source provider will emphasise the non-software value activities in its value proposition if these are of greater importance than the software related value activities. The question *How does the provider differentiate itself from competition?* will be used to analyse the strategy of the providers in relation to the deployed value activities.

**Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project**

The management of the more complex organisational arrangements in an open source situation will require support for external developers. This results in a larger support department which is capable of answering more (technically) advanced questions.

An element to take into account is the possibility that the external developers that make use of an API in case of proprietary software. These also need support, so it is interesting to investigate the difference with open source contributors.

**Contributions of volunteers lower development costs for an open source SaaS provider**

For volunteers to be able to lower development costs an important question is whether the contributions can replace the otherwise needed internal developers. This means quality has to be up to the standards of the SaaS provider. Even if this is the case there have to be sufficient developers interested in contributing to be able to replace internal developers. If internal developers were ‘replaced’ by external contributions, this should result in more lines-of-code per employed developer, assuming that the software capabilities are comparable.

**To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed**

This proposition only applies to the open source SaaS providers. If the providers indeed kept the multi-tenant part of the software proprietary it is relevant whether this was done to prevent this risk. If the providers did open source the multi-tenant part of the software the
question arises whether this introduced the predicted risk of competitors offering the same service. Even if this risk became reality, the next step is whether this affects the revenue of the provider.

*Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that the competitive advantage is lost*

This can again only be determined by cases that have already released their software as open source. The first step is to examine in which phase of the technology life cycle the software was released. If this was done during the emerging phase of the cycle, the question arises whether this induced the expected risk of competitors copying (parts of) the technology and whether this risk negatively affected the revenue of the provider. If the provider released the software after the emerging phase, little can be said on the probability of this proposition.

The questions that provide the information needed to confirm these propositions in attached in appendix B. It is important to keep in mind that these are questions posed to the researcher to provide structure in the information retrieval. These questions should not be asked directly at interviewees.

### 4.3.5 Case Study Reports

The case study reports should all follow similar structure. This enhances the analysis done for the cross-case reports. Below we will provide an outline for reports of both individual cases as well as the cross-case analysis.

**Individual Case Studies**

1. General description of service & organisation
2. Findings on Service Domain
3. Findings on Technology Domain
4. Findings on Finance Domain
5. Conclusions

**Cross-Case Analysis**

The two cross-case analyses compares the results of the two CRM cases and the two WCM cases.

1. General comparison of services & organisation
2. Analysis of Findings on Propositions
3. Conclusions
Final analysis

The final analysis is a comparison of the results of the two cross-case analyses. By comparing results we can tell whether findings hold true for a single type of software, or for multiple. If findings occur in only one of the cases we will have to try to understand why this was the case and whether the proposition could be applicable in other cases.

This chapter has described the case study as our research method. A case study protocol was provided to structure the studies to be performed in the next chapter. The cases that will be studied in chapter 5 have been identified in this chapter and are: Squarespace, Wordpress, Salesforce.com and SugarCRM.
Chapter 5

Case Results

5.1 Introduction

The previous chapter discussed the research methodology. For this thesis a multiple case study will be held, in which four cases will be studied which will be cross-examined afterwards. This chapter discusses the results of the case studies and the cross comparisons of the cases. The four case studies are discussed first after a brief introduction. The next sections (§ 5.6 and § 5.6.3) will discuss the results of the two cross-case comparisons. The final section (§ 5.7) will discuss the similarities and differences between the two case comparisons (the literal replication of the case study).

5.1.1 Individual Cases

All case studies were prepared for by setting up a case database in which all information found was entered. The results are combined in case reports which address each proposition of chapter 3 individually. These reports are presented in § 5.2 to § 5.5. Each case study started with a desk research on each of the subjects. By studying information available on the internet more specific questions could be asked during the interviews that were later held. Due to a difference in time zone not all interviews were held directly, some were done by email. The transcripts of the interviews are provided in the appendices.

5.1.2 Cross-case Comparisons

The cross-case comparisons compare information found for the propositions 1 to 4. Based on these findings a conclusion could be drawn for each proposition, valid for the specific type of SaaS software (WCM or CRM). Propositions 5 and 6 were not addressed, as these concern open source providers only, these will be further discussed in the final analysis (§ 5.7).
5.2 Case 1: Wordpress.com

The first case was Wordpress, an open source content management system and service. After a general description of the service and the organisation behind the service the different findings on the proposals are presented. The last section addresses the case report conclusion.

5.2.1 General description of service & organisation

Wordpress is a web-based content management system developed by Automattic\(^1\). The software has become popular under ‘webloggers’, people who publish small articles on a website on a frequent basis. Although ‘blogging’ is an important feature of the software which pushed its popularity, Wordpress is commonly regarded as more than just blog-software, as it offers all features required to maintain a website (Shreves, 2008). It started in 2003 and has since grown to one of the most popular open source projects. Its user base varies from websites for people’s pet, to industry leading websites\(^2\). Although the software project was initiated in 2003, the SaaS version of the software was only introduced in 2005\(^3\). The website of the SaaS service was used as a starting point of the case study.

**Wordpress.com**

Our interest primarily focuses on the SaaS version of Wordpress. As with most SaaS offerings, it is easy to set up a new account. The standard version of the service is free of charge. The counter side is that advertisements are shown on the website powered by Wordpress. This might not be an issue to individuals, but for corporate websites this is unacceptable. Also the website is accessed through an URL consisting of `<name>`.wordpress.com. A personal domain name is one of the other upgrades. Extra data storage and customisation of the look and feel of the website are other premium features\(^4\).

The most basic version is fine for evaluation of the service and provides a low barrier to entry as there is no investment needed to experiment with the software. The advertisements make sure that even the most basic version of the service delivers revenues to the provider.

The VIP services are aimed at organisations with very large websites with a lot of visitors. The pricing starts at $ 500,- a month, which is, compared to other web content management systems, the top of the market\(^5\).

Wordpress currently hosts 3,132,606 websites on its SaaS platform\(^6\). It is unclear from their website how many of these make use of the basic version and how many upgrades are used. Unless stated otherwise, the following information has been obtained from the official website.

---

\(^1\) Automattic can be found on: http://www.automattic.com
\(^2\) E.g. see http://autoshow.ford.com
\(^3\) The SaaS version of Wordpress can be found at: http://www.wordpress.com
\(^4\) Premium features can be found at: http://en.wordpress.com/products/
\(^5\) VIP hosting services can be found at: http://en.wordpress.com/vip-hosting/
\(^6\) As communicated by Automattic on http://en.blog.wordpress.com/2009/01/02/2008-year-end-wrap-up/
Automattic

The organisation behind Wordpress.com is Automattic. Wordpress is not its only project, other software provided includes forum-, poll- and anti-spam software. Automattic currently counts 34 employees, but the website does not provide exact information on who is working on which project. The employees are spread all over the world, collaborating not in a central office, but over the internet. Since the organisation is privately owned, no information is provided on annual statistics, other than information on usage of their services.

5.2.2 Findings in Service Domain

Proposition 1 - Pricing

Open source SaaS providers charge lower prices

The basic version of the service is free, but has advertisements on the website. The features provided in the free version consist of a full functioning WCM system, including 3GB storage and unlimited pages. Users can select a website design from a large database of templates. The basic version is limited to 35 users. The pricing of the service consists of various feature upgrades. The upgrades are given in table 5.1.

<table>
<thead>
<tr>
<th>Features</th>
<th>incl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic storage 3GB</td>
<td>$ 30/year</td>
</tr>
<tr>
<td>Ad removal</td>
<td>$ 15/year</td>
</tr>
<tr>
<td>Custom CSS</td>
<td>$ 15/year</td>
</tr>
<tr>
<td>Domainname</td>
<td>$ 30/year</td>
</tr>
<tr>
<td>Unlimited users</td>
<td>$ 20/year</td>
</tr>
<tr>
<td>5GB Extra Storage</td>
<td>$ 50/year</td>
</tr>
<tr>
<td>VIP hosting</td>
<td>$ 500/month</td>
</tr>
</tbody>
</table>

Wordpress clearly presents itself as “the underdog” compared to Blogger (a service by Google) and Typepad (Automattic, 2009c). They underline the free services they offer and promise these features will remain free in the future. In this light it is relevant to look at the pricing model of these main competitors. Schneider (Appendix C) commented that Wordpress.com chose to offer the basic version for free to target the audience of blogger, even though Wordpress offers more features.

Blogger  

Blogger is a tool offered by Google which enables individuals to post articles to their personal website. It is targeted at the individual primarily, not at corporate websites. It offers foremost blog functionality and little web content management functionality. This (and the business model of Google in general) explains why there are no paid services and Blogger relies on an ad-based revenue model.

8Blogger can be found at: http://www.blogger.com
5.2 Case 1: Wordpress.com

**Typepad** Typepad\(^9\) is a proprietary WCM SaaS provider with a focus on blogging (like Wordpress and Squarespac). It is targeted at both organisations and individuals. There is no free version and pricing ranges from $5,- per month to $15,- per month.

### 5.2.3 Findings in Organisation Domain

#### Proposition 2 - Value Activities

_The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition_

We will look at both the free version as well as the upgrades when determining the value activities of Wordpress.

**Free version** The value proposition of Wordpress is expressed on several pages of their website. Notable is the lack of a ‘feature comparison’ with competitors, which is often seen on the websites of SaaS providers (see the other cases). The feature page emphasises the free service and the included capacity, like 3GB storage and the different designs that can be chosen. These are clearly not determined by the software, but are non-software activities that provide value. Of the 18 free features, only eight features concern the open source software that is used in the service. Others are related to their community, the quality of the hosting and the lack of vendor lock-in.

**Feature Upgrades** All of the upgrades are upgrades to non-software features. This can be seen from the features given in table 5.1. They all concern the capacity of the service and do not add software features which were not available in the free version.

As all of the software-related value activities are ‘free’, it can be stated that the customer only pays for non-software features. Schneider

#### Proposition 3 - Network Complexity

_Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project_

Wordpress.org is dedicated to supporting developers interested in (and contributing to) the open source software WordPress. It provides a support forum and extensive documentation to both novice and experts.

**Developer Support** There are two members in charge of “support and documentation” (WordPress.org, 2009a), one of which is employed by Automattic. It is not possible for developers to contact the Automattic employees directly. Support has to be sought through the support forums of Wordpress. Another option is to use the Wordpress IRC channel or the mailinglists. The Automattic members also use the IRC channel for their own internal communication (Automattic, 2009b). When looking at the support forums, the mailing list and the IRC channel it is remarkable that almost every question is handled by people not

---

\(^9\)Typepad can be found at: http://www.typepad.com
related to Automattic. This was confirmed by Schneider, commenting that all forum activity is powered by volunteers. The job description of Mark Riley suggests that an important task is the administration of the documentation of both the SaaS version of Wordpress and the open source version.

Although many developers seem to help each other, the effort of Wordpress put into the platform that enables all communication between developers should not be neglected. Every Wordpress employee has to do customer support for three weeks (Schneider, Appendix C).

Software Development The Wordpress team consists of 34 employees (36 according to Schneider). As noted earlier the Automattic website does not provide exact information on who is working on which project, although some job descriptions do provide an indication. The Wordpress.org ‘About’ page contains a list of contributors and lead developers (WordPress.org, 2009a). When comparing this list to the team list on the Automattic website, we find that three out of five lead developers and five out of six contributors are employed by Automattic (Automattic, 2009a). This was confirmed by multiple interviewees. With this, Automattic has held a strong grip on the development of Wordpress. Although Wordpress is open source, this does not mean anybody can edit the core of the software. This privilege is restricted to the core developers, others can only supply ‘patches’ which can be adopted by the core developers if found useful (Westwood, Appendix C). The question remains how much time is consumed by reviewing all the patches provided by the community. As this is mostly done by Automattic employees, this has an impact on the capacity of developers. Lead developers Peter Westwood stated that 80% of his time is consumed reviewing and rewriting contributions. This is handled through the ‘trac’, where all bugs are reported. Each bug or feature is uploaded, tested and in the end committed. Different people have to go through this process, it is not uncommon for bugs to take ‘months’ before they are committed to the source code.

5.2.4 Findings in Finance Domain

The finance domain concerns the final three propositions of this case study.

Proposition 4 - Cost

Contributions of volunteers lower development costs for an open source SaaS provider

The lower development costs should result from contributions made by external developers. These contributions consist of two types: patches to the source code and added functionality through plug-ins or add-ons. Westwood stated that almost all patches through the bug tracking system are contributed by volunteers. The lead developers and contributing developers are either reviewing these contributions, or working on major parts of the source code. The interviewees were unable to estimate what percentage of the code was developed by ‘volunteers’, as they liked to consider all contributors as volunteers, including Automattic. Automattic sees themselves as part of the community. The independence of Wordpress from Automattic shows from the lead developers of Wordpress that are not related to Automattic. In the end, Mullenweg (Appendix C) stated that at least 40% of the code was
made by developers outside Automattic. This means that although Wordpress.com depends on the functionality of Wordpress, Automattic has to collaborate at equal level with the non-Automattic lead developers.

Besides the many contributions to the core of Wordpress, the community has also created over 4,000 plug-ins\(^\text{10}\). Plug-ins are developed by the community, without interference of any of the Wordpress developers. These plug-ins offer functionality not included in Wordpress and since most are available for free, these add to the technological functionality of Wordpress. This means that Wordpress.com could offer these features, without having to develop them themselves. This is a clear reduction of the development costs.

**Proposition 5 - Risk, Release of scalability enabling software**

*To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed*

The first aspect of risk involves the release of the software that enables the scalability of the SaaS platform of Wordpress. Wordpress.com does not make a secret of this piece of software. A website is dedicated to the support and distribution of Wordpress MU\(^\text{11}\). It is provided under the same licence as the normal Wordpress version (GNU GPL v2) and thus free to use, adapt and distribute\(^\text{12}\). There is no restriction on commercial organisations using it as competitor to Wordpress.com.

This raises the question whether the software is indeed used by competitors to offer the same service as Wordpress.com. Searching for ‘wordpress’ on Google does not yield any relevant results. Even the search result advertisement do not offer a similar service to Wordpress.com. Most of the results and ads concern ‘themes’, either provided for free or designers offering their service. A search for ‘wordpress hosting’ leads us towards a webpage on Wordpress.org, where hosting providers are suggested for users looking for hosting for their Wordpress installation (WordPress.org, 2009\(^b\)). These hosting providers offer hosting that meets the requirements for Wordpress, but do not use the Multi-User version of Wordpress to provide this. Some hosting providers offer a “one-click WordPress installation”. Although this looks like SaaS, it’s really a hosting solution with a setup-script. Support, automatic updates, etc are not supported. The user has access to all Wordpress files and can alter these, which is never the case on true SaaS platforms. Searching for ‘Wordpress SaaS’ did not yield any relevant results or ads. An explanation could lie in the unfamiliarity with the term SaaS under the target audience.

The showcase on Wordpress.org\(^\text{13}\) does show some websites that use Wordpress MU to host websites, but in almost all cases these websites are targeted at a niche. None of the featured providers offer the upgrades that are available on Wordpress.com (as described in table 5.1). “It could be possible that some Wordpress.com users of the free edition have switched to these niche providers, but this does not affect the revenue’s of Wordpress.com” (Mullenweg, appendix C). This was confirmed in the interview with Schneider.

---

\(^{10}\)The plug-in database can be found at: http://wordpress.org/extend/plugins/

\(^{11}\)MU for Multi-User. Can be found at: http://mu.wordpress.org

\(^{12}\)The licence can be found in the downloaded zip of the software: http://mu.wordpress.org/latest.zip

\(^{13}\)Showcases can be found at: http://wordpress.org/showcase/
The conclusion is that some hosting providers offer tools to a quick Wordpress installation, but they do not make use of the WordPress MU software to provide this. Although an experienced Wordpress user might prefer this over the ‘controlled’ SaaS environment of Wordpress.com, these hosting providers can not be seen as direct competitors, they serve a more demanding target audience. This is probably the reason these providers are suggested on the Wordpress website itself. This means that although Wordpress has released the ‘multi-tenancy’ software behind Wordpress.com to the public, this has not yet initiated any direct competitors to Wordpress.com.

**Proposition 6 - Risk, Timing of Release to Public Domain**

*Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that the competitive advantage is lost.*

The second aspect of risk lies in the timing of the release of the software to the open source community. Wordpress started in 2003 immediately as open source software (WordPress.org, 2009a). At that time several other WCM providers existed, such as Greymatter and Movable Type, the latter was introduced in 2001 (Movable Type, 2009). Open source WCM system Mambo has been around since 2000 and was released as open source in april 2003 (Mambo Foundation, 2007).

With the above in mind, it can not be said that WordPress entered an emerging market. Several software solutions were available at the time it entered the market. Since Wordpress did not start as a SaaS provider (this was only introduced in 2005), it is hard to tell whether competitors offered the same service. However, as Salesforce introduced the SaaS concept in 2000, WordPress certainly was not an early player in the SaaS market.

The above shows that Wordpress did not release their software in an emerging market.

### 5.2.5 Case Conclusions

Automattic has used the benefits of open source to thrive the growth of their user base. Automattic does not try to control Wordpress

### 5.3 Case 2: Squarespace

For this case we will look at Squarespace, a proprietary web content management service. After a general description of the service and the organisation behind the service the different findings on the proposals are presented. The last section is the case report conclusion.

#### 5.3.1 General description of service & organisation

Squarespace\(^{14}\) is a commercial web-content management SaaS provider located in New York, founded by Anthony Casalena. It adheres a true SaaS strategy in which it offers the service solely on-line through their website and does not offer external installation. There is a free trial version which lasts 14 days, plans afterwards range from $ 8,- to $ 50,- a month,

\(^{14}\)Squarespace can be found at: http://www.squarespace.com
with the average customer paying $20.- per month (Appendix D). In an interview with Stern (2007) Casalena stated that his “company is poised to take percentage points from Wordpress.com”, which we discussed in appendix 5.2. At time of the interview (2007) Squarespace had “thousands of paying customers”. This makes Squarespace a clear underdog of WordPress, which hosts a multitude of this number. Atkinson (Appendix D) confirmed Wordpress, together with Typepad as most important competitors.

5.3.2 Findings in Service Domain

Proposition 1 - Pricing

Open source SaaS providers charge lower prices

The pricing model of Squarespace is organised in five incremental tariffs. The free trial is identical to the basic plan of $8.- per month. Each incremental step is an upgrade in both features and capacity. The basic version includes a full content management system which is capable of managing pages, blog posts, news articles and calendars. It also includes a statistics package and a program to design the graphical part of the website. This last feature is especially useful to website owners not wanting to hire a designer.

An overview of the pricing structure is given in table 5.2. A description of each feature can be found on the website.

Table 5.2: Pricing structure of Service (Squarespace, 2009b)

<table>
<thead>
<tr>
<th>Editions:</th>
<th>Basic</th>
<th>Pro</th>
<th>Advanced</th>
<th>Business</th>
<th>Community</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pricing</td>
<td>$8/mo</td>
<td>$14/mo</td>
<td>$20/mo</td>
<td>$30/mo</td>
<td>$50/mo</td>
</tr>
<tr>
<td>Custom URL</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Storage</td>
<td>1 GB</td>
<td>2 GB</td>
<td>3 GB</td>
<td>4 GB</td>
<td>5 GB</td>
</tr>
<tr>
<td>Bandwith</td>
<td>75 GB/mo</td>
<td>125 GB/mo</td>
<td>175 GB/mo</td>
<td>300 GB/mo</td>
<td>400 GB/mo</td>
</tr>
<tr>
<td>Custom Audiences</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Members</td>
<td>-</td>
<td>250</td>
<td>250</td>
<td>Unlimited</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Editors</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>Unlimited</td>
<td>Unlimited</td>
</tr>
</tbody>
</table>

Software Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Basic</th>
<th>Pro</th>
<th>Advanced</th>
<th>Business</th>
<th>Community</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Shortcuts</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Member Subscriptions</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Drop Box Module</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Per-Page Stylesheets</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HTML Injection Points</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SSL Security</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FAQ Builder</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Email Template Control</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Form Builder</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Public Registration</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Squarespace gives an explanation why it does not offer a free plan on its pricing webpage: “[...] Because we charge for our service and have a stable business model, we don’t
Case Results

5.3 Case 2: Squarespace

need to make decisions later on that would compromise our integrity in order to make money from our users [...]” (Squarespace, 2009b). It further states that if the cost of time of the end-user is taken into account, no web content management system is free.

The website does not compare its pricing with its main competitors. Although Casalena refers to WordPress in an interview with Allen Stern, stating that “Squarespace is a sophisticated publishing tool. It handles more than say WordPress, but [...] ”, he does not compare his pricing to open source initiatives. Atkinson (Appendix D) does state that they have deliberately charged 20% higher prices compared to direct competitors, since they felt the software was of better quality.

5.3.3 Findings in Organisation Domain

Proposition 2 - Value Activities

*The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition*

The value activities deployed by Squarespace are represented in table 5.2. Software extensions are annotated with an asterisk, these extend the functionality of the software. The other options are capacity increases and do not add extra functionality to the software. This shows that Squarespace clearly puts a price at the proprietary software; if specific advanced software features are needed, a more costly plan is needed.

In its differentiation with competition Squarespace does not proclaim its (extended) features over competition, but focusses on the user experience and the stability and scalability of their hosting platform (Squarespace, 2009a). The latter is an important argument for customers questioning why the software can not be downloaded and self-hosted.

Proposition 3 - Network Complexity

*Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project*

Support Squarespace does not offer an API to external developers to build third party extensions, which means they do not have to deliver support to these external developers. There is no documentation on how to write plug-ins or how to extend functionality, as this is not possible.

Concerning the lay-out of the website, end-users have the choice to either make use of the available templates or design their own. Squarespace offers extensive documentation on the latter, as it is impossible to design the unique templates for all of their clients. This however, should not be seen as support for the development community, as these designs are website specific and do not become part of Squarespace itself.

Software Development The Squarespace team consists of seven people (Squarespace, 2009a). Their job descriptions do not provide information on how many developers are
working full-time on the project, apart from the CEO and Support Staff. The small project team keeps the network complexity to a minimum.

5.3.4 Findings in Finance Domain

The finance domain concerns the final three propositions of this case study.

Proposition 4 - Cost

Contributions of volunteers lower development costs for an open source SaaS provider

The development costs of Squarespace are limited to its internal developers. It is possible for end-users to report bugs, but they can not suggest code specific solutions, as they do not have access to the source code. The developers do not have to argue with external developers over ‘optimal solutions’, but do have the obligation to provide feedback on the status of the bugs.

The service blog\(^{15}\) reports of updates and extensions multiple times a month, varying from once to three times a month.

An interesting aspect of the cost structure is that Squarespace does make use of open source software as a building block of their own software to reduce cost.

Risk

The propositions concerning risk are specific to the open source cases of this thesis. As no source code is released to the public, Squarespace is not affected by risks imposed from competition offering identical services.

5.4 Case 3: Sugar CRM

For this case we will look at Sugar CRM, an open source customer relationship management system and service. After a general description of the service and the organisation behind the service the different findings on the proposals are presented. The last section is the case report conclusion.

5.4.1 General description of service & organisation

SugarCRM\(^{16}\) was founded in 2004 as an open source project to provide customer relationship management software. The software has been downloaded over 5 million times since and has a user-base of over 500,000 end-users. The company currently holds over 150 employees (SugarCRM, 2009a). Sugar CRM offers both a SaaS solution (called On-Demand) and a self-hosted version (called On-Site). We will focus this case study on the On-Demand version as we are primarily interested in SaaS. At the end of 2006 the company reached 1,000 customers (Beal, 2006). The website currently claims to have 4,000 customers, but it

\(^{15}\)Can be found at: http://service.squarespace.com/

\(^{16}\)SugarCRM can be found at: http://www.sugarcrm.com/
Case Results

5.4 Case 3: Sugar CRM

does not state how many of these are users of the On-Demand version (SugarCRM, 2009a). CEO John Roberts revealed in an interview that 30% of the revenues are generated by the On-Demand version of the software (Datawire, 2007). In a more recent interview Harrick (interview, appendix E) explained that this percentage has grown to 50% of the revenues.

There are three different versions of the SugarCRM software: Sugar Enterprise, Sugar Professional and Sugar Community edition. The community edition is the open source version of the software, offering the smallest feature set, the other versions offer some proprietary functionality not included in the open source version. Sugar offers an On-Demand version of the community edition, called Sugar Express. The naming of the On-Demand professional and enterprise editions do not differ from the On-Site versions.

SugarCRM.com

SugarCRM.com is the website for the commercial products and services of Sugar CRM. It targets most attention to the services the organisation offers. Besides the SaaS versions of the software, the company also offers training, workshops, consultancy and customisation of Sugar CRM. It offers extensive support to end-users through their Support Portal. The open source version can be downloaded, but support for open source developers is not provided, this is managed through SugarForce.org.

SugarForce.org

SugarForce.org is the website dedicated to the community (open source) edition of Sugar CRM. The open source software can be downloaded here and instructions for installation are provided. Users can ask the community for help in the forums. The site hosts the many extensions developed by the community and provides the documentation how to build extensions. A bug tracker is used to report and search bugs.

5.4.2 Findings in Service Domain

Proposition 1 - Pricing

Open source SaaS providers charge lower prices

The versioning structure used by SugarCRM makes the analysis of this case more complex than the other cases, as in two cases the monthly fee is not solely based on the service (in case of the professional and enterprise versions), but partially also on some extra proprietary functionality not available in the open source version.

Table 5.3 illustrates the differences between the open source SaaS version (Express) and the SaaS versions that contain proprietary upgrades (Professional & Enterprise). Since the professional and enterprise versions can also be purchased as an self-hosted version, we can estimate the cost of the SaaS service: The On-Site professional version costs $ 275/year/user or $ 23/month/user. This means that the open source SaaS service, without the extra features would cost approximately $ 17/month/user.

Sugar CRM clearly positions itself as the cheaper alternative to Salesforce.com (its main competitor), which was also confirmed by Harrick (interview, appendix E). The website lists
Table 5.3: Pricing structure of Service\textsuperscript{17}

<table>
<thead>
<tr>
<th>Editions:</th>
<th>Sugar Express</th>
<th>Sugar Professional</th>
<th>Sugar Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pricing</td>
<td>$7/month/user</td>
<td>$40/month/user</td>
<td>$75/month/user</td>
</tr>
<tr>
<td>Max users</td>
<td>10</td>
<td>300</td>
<td>500</td>
</tr>
<tr>
<td>Min users</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

**Software Features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Sugar Express</th>
<th>Sugar Professional</th>
<th>Sugar Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard CRM functionality</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Dashboards</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Custom Modules</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Customer support</td>
<td>Limited</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Reporting</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Workflow management</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Plugins for MS Office</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Advanced charts</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Access control</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Self-service portal</td>
<td>-</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>Oracle support</td>
<td>-</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>Offline client</td>
<td>-</td>
<td>-</td>
<td>yes</td>
</tr>
</tbody>
</table>

an endorsement of a SugarCRM client stating “We moved from salesforce.com to SugarCRM because Sugar gives us twice the functionality at half the price.” (SugarCRM, 2009d). The company does not contribute this to being open source. The advantages of open source that are expressed on the website do not include lower costs, but claim to deliver a better product. The company claims to spend less time on sales and more time on R&D compared to proprietary SaaS providers. This was also the argument Harrick gave. Lack of vendor lock-in and many extensions are two other claimed advantages of open source software (SugarCRM, 2009c).

5.4.3 Findings in Organisation Domain

**Proposition 2 - Value Activities**

*The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition*

Proposition 2 emphasises the need for value activities not based on open source software. Sugar CRM addressed this from two angles: On one side by introducing some software features which have not been released as open source (see table 5.3). On the other side by stressing the importance of a secure hosting environment and the flexibility of SaaS.

The proprietary additional features give Sugar CRM an advantage over competition. Even if competitors offer the Sugar community edition (Sugar Express) as SaaS, they would not be able to offer all the features that Sugar Professional offers, without paying a licence fee to Sugar CRM.

The open source version is aimed at providing an introduction with the software for
small organisations. SugarCRM assumes the customer will outgrow this version, after which the end-user is expected to engage a commercial relationship with the organisation.

The differences between the Professional and Enterprise edition only concern proprietary features and to some extent capacity enlargement (the limit of maximum users is raised from 300 to 500 users).

Proposition 3 - Network Complexity

Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project

Sugar CRM has split its target audiences in two groups, serving the commercial end-users at SugarCRM.com and the open source end-user and developers at SugarForge.org. This gives a clear distinction between what is needed to support the commercial users versus the service and support needed by the open source community.

Support

The resources offered on SugarForge.org to support the open source community include installation guides, usage guides, development guides, a wiki, forums, FAQs and web casts. All these sources of information can not be edited by end-users (except for the wiki and forums), so these must be maintained by the employees of Sugar CRM.

There is no way to directly contact Sugar CRM employees for support, but there is an IRC channel where employees sometimes are available. According to Harrick a team of 10 employees are dedicated to the community by providing assistance through the forums.

Software Development

Sugar CRM does almost all of its development on the software itself. There is a way for external developers to contribute code, but this is hidden far away in the bug tracker\textsuperscript{18}. After the code has been submitted through the bug tracker, another form has to be filled in to notify the developers of Sugar that a contribution has been posted which will be reviewed once a week by Sugar Developers (SugarCRM, 2009b). Unsurprisingly Sugar would not comment on the number of contributions to the core made by external developers.

External developers are not able to get an overview of unsolved bugs or feature requests other than searching through the bug tracker. Using the search, it is impossible to only list feature requests. There is little or no communication on the progress of bugs and how they are solved.

Developers are foremost encouraged to extend the functionality by development of extensions which do not alter the core of the software, but use its API to deliver functionality (plug-ins). Although not mandatory most of these extensions are open source and developers can contribute to extensions of others. These extensions can provide value to the end-users of all Sugar versions, as they are compatible with each one. Sugar does not review or moderate these extensions, so this does not directly affect the resources of the organisation.

\textsuperscript{18}The bug tracker can be found at: http://bugs.sugarcrm.com/
5.4.4 Findings in Finance Domain

The finance domain concerns the final three propositions of this case study.

Proposition 4 - Cost

*Contributions of volunteers lower development costs for an open source SaaS provider*

As noted above, Sugar CRM does not attribute lowered development costs of the software to one of the benefits of open source software. The benefits of open source are explained in terms of software quality, feature richness (through the installation of extensions), the lack of vendor lock-in (SugarCRM, 2009c) and the lowered sales and marketing costs (Harrick, appendix E).

Cost reduction is achieved through the extensions developed by external developers. This add to the benefits of Sugar CRM, without the investment of their own developers. The extensions are offered in two forms. The SugarForge website lists all ‘projects’, which facilitates projects of external developers. SugarForge currently counts 638 projects, which can all be downloaded and tested. The nature of these projects varies from localisation to added functionality. The Sugar Exchange lists all extensions which are developed by third parties, including the production-ready projects of SugarForge. This website currently counts 118 applications. Many of the SugarForge projects are not listed on Sugar Exchange due to the “listing fees” that apply.

Proposition 5 - Risk, Release of scalability enabling software

*To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed*

Traditionally the Sugar CRM software was build as an on-line application, not with multi-tenancy in mind. As Sugar CRM started offering an On-Demand version, they initially kept the software providing this scalability proprietary. It was only until 2007 that this functionality was added to the open source edition of the software with version 5.0 (Datawire, 2007).

When searching for Sugar CRM providers, most providers either offer custom Sugar CRM implementations or consultancy, or offer hosting for Sugar Community Edition. The latter occasionally offer the full service package, including taking care of updates and delivering support. These could be seen as competitors to the On-Demand services offered by SugarCRM.com, but they can not offer the additional features of the Professional edition once the customer outgrows the most basic version.

Another type of providers encountered when searching for competitors are in fact resellers. They offer On-Demand versions of the Professional and Enterprise editions using the fairly new SugarCRM Datacentre Edition (Kanaracus, 2009). These can not be seen as competitors to Sugar CRMs own On-Demand service, as the providers have to pay a licence fee to Sugar CRM for the use of this edition.

---

19 Using the terms ‘SugarCRM’ and ‘SugarCRM hosting’ on Google
20 E.g.: http://www.optimusrcrm.com/hostingas_nuoma.php?lang=en [last visit: 19-3-2009]
The above shows that while Sugar CRM has released its software which enables scalability, it has used its licencing structure to prevent competitors offering the same service with its own software.

**Proposition 6 - Risk, Timing of release to public domain**

*Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that the competitive advantage is lost*

Sugar CRM was developed as an open source project from the start. This was done in 2004, 6 years after Salesforce.com first started offering SaaS CRM software. Many others followed the SaaS initiative of Salesforce.com and ‘traditional CRM vendors’ transformed their standalone software into online versions (Oracle\(^{21}\), SAP CRM On-Demand\(^{22}\)) (Anderson, 2009).

With the above in mind, Sugar CRM’s first version can not be considered disruptive. It entered a mature market with large players and used open source as a vehicle to quickly expand its brand. With the release of their On-Demand version, they also could not be considered pioneers, as this was almost 8 years after Salesforce.com initial launch.

Since Sugar CRM did not release its software during the emerging phase of the technology life cycle, nothing can be said on the risk of doing so.

**5.4.5 Case Conclusions**

Sugar CRM has taken a less open approach to open source software. The development of the core of the software is strictly separated from the community and all done by Sugar CRM internally. Sugar benefits most from the plug-ins and extensions developed by volunteers.

**5.5 Case 4: Salesforce.com**

This case will study Salesforce.com, a proprietary SaaS customer relationship management provider. After a general description of the service and the organisation behind the service the different findings on the proposals are presented. The last section is the case report conclusion.

**5.5.1 General description of service & organisation**

Salesforce.com was founded in 1999 by Marc Benioff. It started offering their on-demand software (SaaS) in Februari 2000. The company employs 3,566 fte as of Januari 31st, 2009 and serves over 55,000 customers (Salesforce.com, 2009). 72% of its revenue originates from the USA, 17% from Europa and 11% from Asia. The company had her IPO on june 23, 2004.

Salesforce.com was not the first to introduce CRM SaaS, but quickly gained a leading position and retained that position over the years (Anderson, 2009).

\(^{21}\)Available at: http://crmondemand.oracle.com/en/index.htm

\(^{22}\)Available at: http://www.sap.com/solutions/business-suite/crm/crmondemand/index.epx
As of the beginning of 2008 Salesforce.com is trying to expand its services beyond the CRM market by offering a platform that others can use to build On-Demand applications for, known as force.com (Benioff, 2008). This should be seen as another product of the company and therefore falls beyond the scope of this case study, as we are explicitly examining the CRM software.

### 5.5.2 Findings in Service Domain

#### Proposition 1 - Pricing

*Open source SaaS providers charge lower prices*


<table>
<thead>
<tr>
<th>Editions</th>
<th>Group</th>
<th>Professional</th>
<th>Enterprise</th>
<th>Unlimited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pricing</td>
<td>$99/user/year</td>
<td>$780/user/year</td>
<td>$1500/user/year</td>
<td>$3000/user/year</td>
</tr>
<tr>
<td>Max users</td>
<td>5</td>
<td>unlimited</td>
<td>unlimited</td>
<td>unlimited</td>
</tr>
<tr>
<td>Storage</td>
<td>1 GB</td>
<td>20 MB/user</td>
<td>20 MB/user</td>
<td>120 MB/user</td>
</tr>
<tr>
<td>Premier Support</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Features</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard CRM functionality</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Analytics</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Google Analytics integration</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Callcenter</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Innovation management</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Advanced CRM functionality</td>
<td>-</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Workflow management</td>
<td>-</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>API integration</td>
<td>-</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Mobile</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 5.4 provides an overview of the differences between the versions. The benefits Salesforce.com claims, are foremost benefits of SaaS, which means they target their service at users of traditional CRM applications. This also shows from the 2008 annual report, where IBM, Oracle and SAP are mentioned first as competitors, other On-Demand providers (including SugarCRM) are mentioned second.

When we look at the pricing of SaaS competitors, most SaaS providers are in the same price category as Salesforce (Oracle CRM at $70/user/month, SAP CRM at $75/user/month). Lower cost alternatives are Microsoft Dynamics and SugarCRM. As a pioneer of CRM SaaS, Salesforce has not yet adjusted its pricing (Gordon, Appendix F).

---

24 Salesforce lowered their pricing in response to SugarCRM Express Edition introduced on May 29th, 2009
25 Pricing obtained from sales representatives
5.5.3 Findings in Organisation Domain

Proposition 2 - Value Activities

The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition.

To differentiate between the editions offered (as displayed in table 5.4) Salesforce.com has made a mix between capacity upgrades (in terms of users, support level and storage space) and additional software features. The unlimited edition offers almost no extra software features, but upgrades most capacity limits. Independently of the different editions extra software functionality can be purchased, such as a customer portal and a loyalty system.

This pricing structure enables small companies to enter a low level edition at low cost. When the company grows it is forced to adopt a more expensive edition, either because of specific feature needs, or capacity need.

Besides capacity upgrades and additional features, Salesforce.com also offers different levels of service as added value activity.

Unique selling point As claimed by Salesforce.com, the key advantages of Salesforce’s services are given in the Annual Reports:

- Secure, reliable, scalable delivery platform
- Rapid deployment
- Ease of integration and development
- Lower total costs of ownership
- High levels of user adoption

Although their software is proprietary, they do not claim to provide functionality that others do not.

Proposition 3 - Network Complexity

Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project.

The ‘network’ of Salesforce is relatively straightforward, there is a R&D department that develops the software, there is a support department that supports users and there is a large sales department (Benioff, 2008). The support department is split up in a division that supports the end-users and a division that provides technical support to developers.

Salesforce.com has a dedicated R&D department which develops new features and tests the software. External developers only have the API at hand to build extensions.

Salesforce.com offers an API which external developers can use to customise their Salesforce CRM installation and to develop add-ons. The support of these developers is
handled through developer.force.com, a separate website. Developers can not contact sup-
port directly (unless they have Premier Support), but there are Discussion Boards where
many Salesforce.com employees interact with the developers.

As external developers can not look into the source code behind Salesforce, there is
no way to contribute code. It is possible to report bugs, but developers can not suggest
solutions.

5.5.4 Findings in Finance Domain

The finance domain concerns the final three propositions of this case study.

Proposition 4 - Cost

Contributions of volunteers lower development costs for an open source SaaS provider
Salesforce.com currently has an internal R&D department consisting of 8% of the total
expenses or $ 64 million in 2008 (Benioff, 2008). As these costs consists mostly out of
salaries, approximately 285 employees are working on the software. These employees both
work on the native Salesforce software, as on the platform that enables scalability of the
software and the API to external developers. According to Gordon (Appendix F) this cal-
culation is too simple, the actual number of developers would be higher, but she could not
comment on the exact number.

Although the external developers do not contribute to the core of the Salesforce.com
software, they do develop extensions which are sold to other Salesforce.com users through
the AppExchange. These extensions add to the offering of Salesforce.com indirectly and
shows resemblance with the plug-ins of Wordpress and projects of SugarCRM. The 2009
Annuals state that 800 applications have been developed by ‘third party developers’ (Benioff,
2009, pp. 4). The website states that customers of Salesforce.com have made over 80.000
customisations using the API. These customisations have thus not been shared with others.

Risk

As Salesforce.com did not release their software as open source, the final two propositions
are not relevant for this case study.

5.6 Case comparison 1

This chapter concerns the theoretical comparison between the two case studies on WCM
SaaS providers located in § 5.2 and § 5.3. First the providers of the cases are compared,
so differences in environment and background can be accounted for when analysing
the propositions. For each proposition a comparison is made between the case study results and
whether this is confirms or invalidates the proposition.

All data is obtained from the appropriate cases, no new information is introduced in this
case comparison.
5.6.1 WCM Providers

Both Wordpress and Squarespace started the development of their current software offering (the core of their service) in 2003. Wordpress has since become best known, as it is a very popular open source project. Both organisations do not provide any information on revenues, so a comparison of size has to be made on their number of clients. Because they have different pricing structures, the companies cannot be compared on absolute numbers either. Wordpress counts just over a million blogs, but does not state what percentage of these has actually paid for any of the upgrades or how much advertisement revenue this has yielded. Squarespace has “thousands” of customers, paying an average of $20 per month resulting in a “multi-million” annual revenue (exact numbers were not provided).

The number of employees might provide some basis for comparison: Wordpress counts 36 employees where Squarespace counts 15 employees.

5.6.2 Analysis of Findings on Propositions

Proposition 1 - Pricing

Open source SaaS providers charge lower prices

The pricing structure differs greatly between the two WCM providers, which frustrates one on one comparison. Both software applications also differ in feature set, so it is not possible to compare one price plan with another (Table 5.1 and Table 5.2). When taking the perspective of the end-user a bare minimum is an ad-free website with an unique URL. These upgrades would cost $45,- per year with Wordpress and $168,- per year with Squarespace.

In this plan, Squarespace has no limit on users, which would cost $30,- per year extra with Wordpress.

At first sight, the difference is a factor 2 in pricing between comparable features. That said, Squarespace indicated that they deliberately set their pricing 20% higher than direct competitors (like Typepad), since they felt they had a better product. This still leaves a difference in price of 30%.

There is one big difference between the two that does not read from the feature list: Both organisations offer a forum for support, but Squarespace also offers the option to open a support ticket through the administrative interface. This form of personal support is not offered by Wordpress natively. The level of support desired is very personal. For this proposition it can be said that the open source alternative is priced lower, comes at the cost of less support.

Proposition 2 - Value activities

The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition

As suggested by the proposition, Wordpress does not charge any money for any of its software related features, but limits this on aspects that provide a greater capacity of data storage and users, or services like domain name and layout customisation. It does list its software features explicitly, but emphasises that these features are all included in the free edition.
Wordpress does use “open source” as an argument for their product over that of competitors.

Squarespace on the other hand has made a mix of most upgrades also offered by Wordpress, combined with software features that are added each step. The need for a specific feature (like a FAQ builder) can force a user in a higher price plan. Atkinson states the product is more user-friendly which is clearly a feature of its software they use to separate themselves from competition.

The above shows clearly that Wordpres.com does promote the features of Wordpress (the software), but makes money (thus differentiates) on capacity upgrades on their service. Also premier support is used as a value activity to differentiate from competition. This is in line with the proposition.

Proposition 3 - Network Complexity

Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project

The complexity of the network of a WCM SaaS provider consists of the number of actors it has to interact with. The difference between open source and proprietary software providers was expected to lie in the support and communication with external developers for open source projects.

Developer Support  
Wordpress has dedicated Wordpress.org to the support of external developers. This website includes many pages describing each function in the code base through a wiki. There is an extensive forum where developers can ask for help and help each other. It also provides access to the IRC channel, the mailing lists and the bug tracker.

Squarespace has a FAQ and forum supporting its end-users, not developers. The only thing external developers can change is layout, not functionality. There is a developer forum, but this mostly concerns design issues.

What can be concluded from the above is that the amount of data Wordpress has to manage is a great deal larger than Squarespace. It can not neglect its external developers, as this is an important target group for Wordpress.

Software development  
Squarespace has a very straightforward organisation of their software development team. The team consists of ten members, each responsible for their own aspect of the software. As all are working in the same office the communication lines are short. End-users have the possibility to report bugs, suggestions and request features. These are assigned to the appropriate developer. Squarespace has a special blog where they announce feature additions and maintenance schedules.

Wordpress’ organisation of software development is more complex. Lead developers of both Automattic and external developers are working together. It is important for Automattic to have a strong voice in decisions made in Wordpress development, as this same software runs on Wordpress.com. An error in the open source version results in an error on the SaaS platform. To maintain quality standards, peer control is used to check every change made to the source code (the process is explained in § 5.2.3). This reduces the productivity
of the developers. In the end, only lead developers can commit any of the changes. There are many numbers of users adding bugs to the tracker that are not bugs and suggestions are sometimes rejected.

The big difference in this process is that the Squarespace developers have a final word on everything. They decide whether a feature is added and if not, no-one will know. Within Wordpress this is more of a discussion. Everybody can suggest solutions and argue its benefits. The analogy of the Cathedral and the Bazaar of Raymond (1999) is very visible. When looking at complexity; which is easier to manage, the cathedral and its audience, or the bazaar with all its people?

Looking at the above differences, it becomes clear information management is very important to support the information exchange between developers in an open source environment. The complexity of an open source project shows from the large amount of information to support developers and the large amount of communication between these developers.

**Proposition 4 - Cost**

*Contributions of volunteers lower development costs for an open source SaaS provider*

The lower development costs should result from contributions made by external developers. These contributions consist of two types: patches to the source code and added functionality through plug-ins or add-ons. The latter is not unique to open source software, but can also be achieved through an API in commercial software.

Wordpress both receives many contributions for the source code and offers a large list of plug-ins. A lot of the functionality offered by Squarespace is not included in Wordpress by default, but can be added by plug-ins. This functionality would otherwise have to be developed by Automattic to be able to compete with Squarespace.

When comparing the size of the two projects, Squarespace counts a lot more lines of code. This could be explained with the native functionality that is placed in for Wordpress. Automattic estimated that at least 50% of the code was written by others, Squarespace has written everything themselves, apart from the few open source elements they use within their software.

**Proposition 5 - Risk: Release of scalability enabling software**

*To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed*

This proposition can not be compared between Wordpress and Squarespace, since it concerns open source software only.

**Proposition 6 - Risk: Timing of release to public domain**

*Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that the competitive advantage is lost*

This proposition can not be compared between Wordpress and Squarespace, since it concerns open source software only.
5.6 Case comparison 1

5.6.3 Case Comparison 2

This chapter concerns the theoretical comparison between the two case studies on CRM SaaS providers located in § 5.4 and § 5.5. First the providers of the cases are compared, so differences in environment and background can be accounted for when analysing the propositions. For each proposition a comparison is made between the case study results and whether this is confirms or invalidates the proposition.

All data is obtained from the appropriate cases, no new information is introduced in this case comparison.

5.6.4 CRM Providers

The CRM providers of the cases that are compared differ on several aspects. The first aspect is their age. Where Salesforce entered the CRM SaaS market in 2000, SugarCRM entered the CRM market in 2004 and started as a traditional CRM application (not SaaS). It started offering a SaaS version in 2006. This reflects in the number of clients: SugarCRM has 4000 clients, where Salesforce has reached 55,400 clients. SugerCRM counts 150 employees, Salesforce employs 3566 fte.

5.6.5 Analysis of Findings on Propositions

Proposition 1 - Pricing

*Open source SaaS providers charge lower prices*

The comparison of pricing and features is straightforward: Sugar charges $7,-; $40 and $75 per month for its Express, Professional and Enterprise editions and Salesforce charges $8,25; $65 and $125 per month respectively. SugarCRM has chosen these number to be the lower cost alternative to Salesforce.com and even displays testimonials of customers claiming this cost reduction. Sugar does not confirm that this was also expected by customers, but claim it results from the lower costs of customer acquisition.

Proposition 2 - Value activities

*The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition*

The value activities used by SugarCRM to differentiate from others who offer the open source (Express) edition of SugarCRM come from the proprietary features included in the Professional and Enterprise Editions. This tactic shows resemblance with Salesforce, which offers exactly the same editions. To differentiate from Salesforce, Sugar CRM offers lower pricing. Both Sugar CRM and Salesforce promote the benefits of SaaS to differentiate themselves from traditional CRM applications.

Salesforce clearly targets their service at those organisations currently using a traditional CRM package, while Sugar CRM aims at both traditional CRM users as Salesforce’s clients.
Case Results

5.6 Case comparison 1

Proposition 3 - Network Complexity

Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project

When the complexity of the networks of Salesforce and Sugar CRM are compared, the large number of websites maintained by Sugar CRM attracts attention. These are aimed at supporting developers which want to develop extensions for Sugar CRM. The emphasis lies in developing extensions, not contributing to the core of the source code. In this respect having released the source code serves as an transparent alternative to offering an API to interact with the Sugar CRM software. As developers can see how the code works, it is easier to write extensions to this code that interact with it.

In this respect Sugar CRM and Salesforce.com are very similar; they both encourage external developers to extend functionality of their software, but maintain the core themselves. Both offer an API and documentation how this is done. In case of Sugar CRM, the benefit of open source software lies in the transparency, the developer can see with his own eyes what happens in the source code.

The reason that Sugar CRM hosts a lot more websites than Salesforce.com lies in the nature of open source projects. As a host of an open source project the communication between the open source users and developers has to be facilitated to make it a success. Not only the API has to be documented, but the whole source code has to be documented. Open source projects not only attract developers, but also end-users wishing to install the open source version on their own server. These have to be supported. Although Sugar CRM does not offer direct support to these end-users, Sugar CRM does have to facilitate methods for end-users and developers to help each other.

Sugar CRM is not interested in community involvement around the development of the core of the software. There is little space do discuss new features or bug issues. External developers can submit code, but whether these contributions make it into the software depends on the grace of the Sugar CRM developers, which is far from ‘democratic’. This reduces the complexity as the number of interactions with externals is reduced.

Proposition 4 - Cost

Contributions of volunteers lower development costs for an open source SaaS provider

Both Sugar CRM and Salesforce.com benefit from extensions developed by external developers. Although Salesforce.com is a factor 10 larger than Sugar CRM (both in revenue and number of clients), the number of applications developed are very close: 800 versus 638 for Salesforce.com and SugarCRM respectively. This shows that an open source project attracts more external developers than proprietary projects. Although many download the open source software without contributing, the effort put into the software by those that do results in both a reduction of cost (e.g. Sugar CRM did not localise their software themselves, Salesforce.com did) and a more attractive product (most of the apps offered by Salesforce.com’s App Exchange are not for free).

The development costs of Sugar CRM spent on the core applications are not lowered by contributions, as Sugar CRM puts little effort in community involvement in the development of the core of Sugar CRM software.
There is another aspect of costs (not development costs) that are affected by the open source nature of Sugar CRM. Salesforce.com spends 50% of all revenue on marketing and sales of their product and 9% on R&D. Sugar CRM spends 50% on R&D. Sugar claims this is possible, because an open source product sells itself. People are free to try and test the open source product. After convincing themselves of the quality of the product, they change to one of the more advanced editions offered by Sugar CRM (either because they need specific features, or they want the service of the On-Demand version.

**Proposition 5 - Risk: Release of scalability enabling software**

*To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed*  
This proposition can not be compared between SugarCRM and Salesforce.com, since it concerns open source software only.

**Proposition 6 - Risk: Timing of release to public domain**

*Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that the competitive advantage is lost*  
This proposition can not be compared between SugarCRM and Salesforce.com, since it concerns open source software only.

## 5.7 Final analysis

In this final analysis the propositions are evaluated with the findings from the individual cases and cross-case analyses. The focus is to find any differences between the CRM and WCM cases, which indicate uncertainty in the findings on the propositions. If propositions are supported by both CRM and WCM findings, more robust conclusions can be drawn on the differences between the business models.

### 5.7.1 Proposition 1 - Pricing

*Open source SaaS providers charge lower prices*  
In both WCM and CRM situations the open source providers charged significantly lower prices than the proprietary providers. In both cases the open source providers argued that this was a strategic choice, one of the advantages of open source is that costs are lowered and this should reflect in the pricing. The argument that competitors could offer the same service using the open source software at a lower price did not influence their prices, as they thought they had a branding advantage over these third party providers.  
Although it remained unclear whether the customers also expected the service to be less expensive, the case studies showed that the providers indeed charged lower prices than their proprietary competitors.
5.7.2 Proposition 2 - Value activities

The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition.

The theory behind this proposition is that open source software features can not be monetised, since competitors offer exactly the same features by using the publicly available software. Unique selling points must thus lie in other features.

When comparing the value activities of the four cases it becomes clear that proprietary providers do see their software features as a competitive advantage at put a price at it accordingly. For Squarespace this was reason to raise their pricing compared to competitors. They also reckon the importance of non-software value activities, like the stability and performance of the hardware platform the service is provided on.

The open source providers have solved the problem in different ways. Wordpress.com does promote its software features, but they are all included in the free version, so they do not monetise these features. The value activities that are offered as an upgrade are all non-software upgrades, which is in line with expectations. Sugar CRM tackled the problem by introducing software features that are not included in the open source version. These proprietary features prevent competitors offering the exact same service.

The conclusion is that the proposition is supported by our case studies, but there are different ways to manoeuvre around the problem. The cases have shown that open source SaaS providers can use any of the following strategies:

- Introduce capacity limitations
- Offer different levels of support
- Introduce proprietary software features on top of the open source software
- Offer a high level of service (expressed in up-time, response-time, etc)

5.7.3 Proposition 3 - Network complexity

Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project.

In both WCM and CRM cases the complexity of the networks of the providers were compared. The analysis shows that complexity is inherent to open source software projects. The number of actors to cooperate with increases with the success of the open source project. The cases show that the open source provider has to facilitate the communication between these actors, which consist of end-users of the open source software, external developers, clients of the SaaS provider and developers of the SaaS provider. Both open source providers have to maintain a multitude of websites providing documentation, discussion platforms and support. The result is that many actors engage in dialogue and help each other, without the interference of the open source provider. The benefit of this effort is a very involved community.

The difference between Wordpress and Sugar CRM is also visible: Wordpress gets the community really involved, in fact, some of the lead developers are not related to the SaaS
offering on Wordpress.com and only focus on the open source software. Sugar CRM has
shared its source code, mainly to enable developers to build applications as extensions to
Sugar CRM, but they do not let the community work on the core. While this reduces com-
plexity for Sugar CRM (they do not have to manage the discussion with external developers
on the core software), it also reflects in the involvement of the community; discussion on
IRC channels is ample compared to Wordpress and the discussion boards have many un-
solved questions. Of course the fact that Wordpress is a far more popular and extensive
open source project also contributes to this difference.

Interesting to see is that offering an API also increases complexity, which was the case
in the CRM studies. By offering an API Salesforce.com also has to support external devel-
opers. This results in a complete website dedicated to developer support, as is also seen with
Sugar CRM. The main difference is that Salesforce.com does not have to facilitate a group
of non-paying end-users (the ones that download the open source version of Sugar CRM).
The counter side is that there is no real community of developers helping each other, so
Salesforce.com has a dedicated support division for developers only. Although both Sugar
CRM and Salesforce.com attract third party developers, it shows that those working with
open source software are more eager to share their solutions. This results in a higher num-
ber of third party applications (both free and proprietary). This could also be encouraged
by the large user-base; Sugar CRM has ‘4.000’ paying customers, but millions of users of
the open source version.

With respect to the proposition, the cases show that the governance of an open source
project do consume resources of the provider. The difference between an open source
project and a proprietary project offering an API is a lot smaller and might in some cases be
marginal.

5.7.4 Proposition 4 - Development costs

Contributions of volunteers lower development costs for an open source SaaS provider
The fourth proposition determines whether the effort taken by open source SaaS providers
as described in the previous proposition results in lower development costs. For this to be
the case the external developers of open source projects had to contribute in such a way that
these contributions could replace efforts of the providers themselves.

The contributions of external developers result in two forms of added value:

- Developers can contribute to the core of the software, directly aiding the provider in
development, or;

- developers can develop extensions to the software which add to the attractiveness of
the product, indirectly contributing to the provider.

With respect to the first aspect Wordpress and Sugar CRM have taken a different ap-
proach. Where the community contribution process in Wordpress is fully ‘open’, Sugar
CRM’s process is less open. Although this reduces complexity (as discussed above), it also
results in less contributions and thus almost all development on the core has to be done by
Sugar CRM themselves. This clearly shows that an open source provider itself controls how much external involvement it accepts.

When comparing this with the proprietary providers, it becomes clear that the open approach of Wordpress reduces most cost. Most bugs are reported including a fix, where the feedback received by Squarespace and Salesforce.com consists mainly of feature requests. The approach of Sugar CRM lies somewhere in the middles, where is remains unclear whether development costs are reduced.

The second aspect results in add-ons or extensions which are beyond the control of the open source provider. These extension do often contain solutions to problems encountered by other end-users. An example is the localisation; Sugar CRM has been translated in many more languages than Salesforce.com. Although an API at an proprietary provider also results in such extensions, an open source projects seems to attract more developers willing to share their code. These extensions are often not critical to the application, but do provide value to end-users, making the SaaS offering more interesting.

The conclusion is that development cost are indeed lowered, especially with extensions which elaborate the functionality. Provider can try to attain this by offering an API, but the CRM case has shown this to be less effective. The costs of the development of the core of the application can only be lowered when the provider collaborates with the community, which was the case with Wordpress.

Based on the above we expect development cost to be lowered, but this does not mean that all cost of a open source provider are lowered. As seen for proposition 3, the increased complexity will result in higher cost for certain divisions of the provider. This thesis does not provide the net cost/benefit analysis to draw these conclusions.

### 5.7.5 Proposition 5 - Risk: Release of scalability enabling software

To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed

In both cases the open source providers also released the software that enables the scalability of the SaaS service.

In case of Wordpress this means that competitors could use this software to offer exactly the same service. The analysis looked for evidence proving that the risk expected from the proposition was encountered, but no competitors offering a service identical to Wordpress.com were found.

Due to the proprietary software features that were used by Sugar CRM to distinguish their commercial versions from the open source version, competitors could only offer a SaaS version of the open source edition. Since Sugar CRM does not offer a SaaS version of their open source software, these providers using the Sugar CRM software form no threat to Sugar CRM, since they target a different group (not interested in the features Sugar CRM offers in its SaaS version).

The conclusion is that the expected risk was not encountered in our cases. Due to the unusual construction Sugar CRM, we have only one case that fully rejects this proposition. For this reason this conclusion needs more cases to be strengthened.
5.7.6 Proposition 6 - Risk: Timing of release to public domain

Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that the competitive advantage is lost.

Both open source providers entered the market of WCM systems and CRM systems in a phase after the emerging phase of the product/technology life cycle. For this reason no conclusions can be drawn on the probability of this proposition.

5.8 Conclusions

Of the six propositions that were studied, four were confirmed by the case studies. Their support by the case study is summarised in table 5.5. The propositions are discussed next.

Table 5.5: Outcome of case study

<table>
<thead>
<tr>
<th>Proposition</th>
<th>WCM Case 1 vs. 2</th>
<th>CRM Case 3 vs. 4</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposition 1</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Proposition 2</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Proposition 3</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Proposition 4</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Proposition 5</td>
<td>Rejected</td>
<td>Not applicable</td>
<td>Partially Rejected</td>
</tr>
<tr>
<td>Proposition 6</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

5.8.1 Differences in Business models

Proposition 1 - Pricing

The first proposition argues that the pricing of open source SaaS providers is lower than proprietary providers. This was confirmed by the analyses. Several possible reasons were given in chapter 3. However, it remained unclear whether consumers expect lower pricing or that pricing was lowered to provide a competitive advantage.

Proposition 2 - Value Activities

The case studies also indicate that open source SaaS providers rely on value activities that are not based on open source software to distinguish themselves from competitors. The features incorporated in the software do not form a unique selling point, as other can offer exactly the same features.

Proposition 3 - Network complexity

To facilitate an open source project means that the provider has to put effort in the communication between developers and support these appropriately. This increased number of actors the provider has to cope with consumes resources which are not needed in case of a proprietary provider.
A related finding is that the proprietary SaaS provider that offers an API also encounters this increased number of actors, along with the extra attention these demand. Nevertheless does an open source project also attract non-paying customers that need support (those that download the software), so an open source project will still require more resources than a proprietary project with an API.

In § 3.4.1 another reason was also given for this proposition: the need to change the mindset of the organisation. This reason was not encountered in our cases, since both open source organisations have been open source since the start.

**Proposition 4 - Development costs**

The lowered development costs were expected to consist of two dimensions; contributions to the core and extensions to the software through plug-ins and add ons (see § 3.5.1). Contributions do prove lower the development cost, but the amount depends on the openness of the SaaS provider. Providers collaborating with external developers will see more contributions to the core of the software. More closed organisations will find most contributions in the form of extensions. A prerequisite for the latter is of course that the software sufficiently provides a platform as discussed in § 3.5.1.

When a provider aims to attract external developers with an API, it should consider open source as an alternative as this attracts more developers willing to share and contribute.

**Proposition 5 - Risk: Release of scalability enabling software**

A source of risk was expected when open source providers release the software that enable the scalability of their SaaS platform. Although this was done in one case, this did not result in competitors offering exactly the same service. Since this was not proven by multiple cases, the conclusion on this proposition is not as solid as the other propositions. With regard to the theory in chapter 3, this further illustrates the quote “There is no inherent value in a technology per se”, emphasising the need of a business model and added value to sell a service.

**Proposition 6 - Risk: Timing of release to public domain**

A second source of risk was expected when SaaS providers release their software to the open source domain whilst in the emerging phase of the technology life cycle. None of our cases released their software in this early phase, so no conclusions can be drawn on this final propositions. This does not mean the risk does not exist, other cases that have released their software in this phase could still confirm this proposition.

**5.8.2 Other affected elements of the business model**

Besides the four elements identified by this case study, a benefit of open source over proprietary software shows when comparing the market share of Wordpress versus Squarespace, this confirms the findings of Bonaccorsi, Giannangeli and Rossi (2006) that open source is a good strategy to benefit from network externalities in a mature market. Of course this
could also be contributed to other factors, but these cases do not conflict with the findings of Bonaccorsi.

In § 3.2.2 we suggested that brand-reputation could play a role in the decision process of end-users. Although no proposition was formulated on brand-reputation, Wordpress suggested that this is one of the reasons end-users favour the SaaS service at Wordpress.com over competing providers offering identical Wordpress software.

The results of the case studies show that the main differences are found in the organisational domain of the business model, although these have their impact on the different other domains in the business model design. The finance domain is affected on both cost and revenues. Cost are lowered by contributions, but also raised by the complexity of the value network and the resulting interactions with actors. The lowered pricing affects revenues, but increased market share from network externalities could result in higher revenues. The service domain and specifically the delivered value is affected by the value activities and the technological functionalities. The latter are a result of external developers building add-ons and plug-ins that add to the functionality of the service.

These findings show that the decision whether to release software as open source software is a complex question which involves at least four elements of the business model. The next chapter will form the conclusion of this thesis.
Chapter 6

Conclusions & Recommendations

This final chapter concludes this thesis. In § 6.1 the general conclusions will be drawn, § 6.2 provides the recommendations for software providers that are evaluating whether to release their software to the open source domain. § 6.3 contains the recommendations for future research.

6.1 Conclusions

This thesis started with a problem that SaaS providers are facing: whether or not to release their software to the open source domain. At the time of writing it was unclear which elements of the business would be affected by such a decision. The goal of the thesis was to gain understanding so that decision makers at SaaS providers can better trade-off pros and cons of the decision on the release of their software to the open source domain.

From a scientific perspective the results of this thesis will contribute to a better understanding of business model of both proprietary and open source SaaS providers. Where prior research has mainly focused on how to monetise open source software, this thesis focused on effects on the business model.

The basis of the research project was formed with the central question *What are the differences between the business models of proprietary SaaS providers and open source SaaS providers?*

To answer this question the answers to several sub-questions were needed. The thesis started with the question *What is SaaS?* Chapter 2 provided information on this new form of software provisioning and described technologies needed to provide SaaS. In § 2.1.5 a delineation was given to draw a line on what is considered SaaS for this thesis. This was needed to be able to select cases in chapter 4.

The next question *What are the differences between proprietary and open source software?* was discussed in § 2.2. Understanding these differences provides a basis for chapter 3, where the differences were mapped to business models of both open source SaaS providers and proprietary SaaS providers. The differences are foremost found in licensing and copyright. An commercial organisation which leads an open source software project can be compared with an organisation developing proprietary software with an API, as both...
have to service clients and developers.

What are the elements of a business model? Before being able to identify differences between business models, agreement has to be established on the building blocks of a business model. In § 3.1.1 the definition for a business model was adopted as used in the STOF model literature: “A business model is a blueprint for a service to be delivered, describing the service definition and the intended value for the target group, the sources of revenue, and providing an architecture for the service delivery, including a description of the resources required, and the organisational and financial arrangements between the involved business actors, including a description of their roles and the division of costs and revenues over the business actors” (Bouwman et al., 2008, pp. 33).

The STOF model was used as a theoretical starting point from which several elements of business models were identified which were expected to differ between the business models of open source and proprietary SaaS providers. For each of these elements a proposition was given which defined how this element was expected to differ. This answered the question What elements of the business model of a SaaS provider are expected to differ between proprietary and open source cases? The expected different elements found in chapter 3 are: Pricing, Value activities, Network complexity and Cost. Two other propositions were defined, not containing an expected difference but concerning risks that could possibly affect open source SaaS providers.

The multiple case study was chosen as research method to analyse whether the expected differences in business model were to be encountered in selected cases. A case study provides the depth of analysis to gain understanding how the business model of an organisation is constructed. The case study consisted of four cases, two open source cases (WCM and CRM) which were compared to two proprietary cases (again WCM and CRM). In the final analysis the results of the WCM business model differences is compared to the CRM business model differences to obtain more robust results. This type of case study research is called a hierarchic comparative case study.

The answer to the main question What are the differences between the business models of proprietary SaaS providers and open source SaaS providers? results from chapter 5 with the conclusion of the case studies. The results show that four of six propositions were indeed confirmed by the case study analysis. The differences in business model were found in the following elements of a business model: Pricing, Value activities, Network Complexity and Cost. The differences are presented below.

**Pricing** The results of the case studies indicate that open source SaaS providers have chosen to charge lower pricing than their direct competitors. This was a deliberate strategy of the management of these providers.

**Value activities** Open source SaaS providers have to deploy other value activities than their open source software to be able to differentiate from providers offering the exact same service based on the same open source software. These value activities can both lie in capacity upgrades, or in software features that are excluded from the open source version.
Network complexity As the number of actors increases with an open source SaaS provider, the complexity of its network can be said to increase. This results in more communication and interaction with developers outside the organisation (volunteers). The provider can control the level of added complexity by reducing the collaboration with external developers on the core of the software.

Cost Open source SaaS providers have lower development costs, since external developers contribute to the core (if made possible) and extend functionality by developing add-ons and plug-ins.

The two propositions that were not confirmed by the case studies concerned the risk element of the business model. It was expected that the release of the software that enables multi-tenant scalability would lead to competitors offering the same service. This was not the case in one of our cases. The other case protected itself from such competitors with proprietary features, thus did not provide any added value to this proposition. The other risk factor was expected to result from a release of the software to the open source community in the emerging phase of the technology life cycle. Both open source cases did not release their software in this phase, so no conclusions could be drawn on this proposition.

6.2 Contributions

The introduction proposed the intended contributions of this thesis to both the scientific field, as to managers of SaaS providers. Both will be discussed here.

6.2.1 Scientific Contributions

The goal of this thesis was to broaden the understanding of differences in business models of open source and proprietary organisations. With the identification of four elements that differ, this goal was achieved. This thesis shows that the difference is not a mere cost issue, but requires a transformation of the whole organisation.

The STOF model was used as a starting point of the identification of possible different business model elements. The STOF model was originally targeted at mobile services. Although some STOF elements are not as important in SaaS business models as in mobile business models, the STOF model did not conflict with the findings in other literature and the results of this thesis.

6.2.2 Managerial Contributions & Recommendations

This section discusses the recommendations that result from this thesis, targeted at the problem owner: the SaaS provider that evaluates the trade-off whether or not to release its software to the open source community. To assess this trade-off the differences between the business models of open source and proprietary SaaS providers have been identified. It is important to acknowledge the scope of these elements, they do not exist on their own, but have an effect on other elements of the business model (as discussed in § 5.8.2). The
following paragraphs discuss the consequences that these differences have on the different departments of a Saas provider;

Cost & Benefits

An important factor in the decision making process will be the cost/benefits of the release. It is important to not only consider lower development cost. The added complexity of the value network will raise costs; additional support staff is needed to support developers. Lower pricing will affect revenues, but on the other hand the network externalities will contribute to the revenues.

Research & Development

Besides the cost/benefits to take into account, one must also consider the changing role of the R&D department. The focus will be less on internal development and more on collaboration and reviewing. This requires both a different type of employees (more senior developers over junior developers) and a different mindset (see also the ‘not invented here’ syndrome discussed in § 3.4.1). It is up to the company management to assess whether the open source philosophy is ‘compatible’ with the culture of the organisation.

Sales & Marketing

The difference between open source and proprietary is also felt in the sales and marketing department. Since the software features are no longer a unique selling point (anyone can copy these), the focus has to change to other value activities like quality of service, stability of the platform and the level of support offered.

Support

For an open source Saas provider the support department is of greater importance than for proprietary providers. Its position is more important in the value activities offered, as the features have become less of an unique selling point. On the other side the support department has to offer support to developers, which was not the case before.

Open source versus API

One method for proprietary Saas providers of attracting external developers is by offering an API. However, the cases have shown that an open source project attracts more developers willing to share their add-ons and plug-ins. The developers for a proprietary API often use their code internally, or charge a price for them when offered to the public. The add-ons and plug-ins that are shared contribute to the technological functionality of the Saas provider and hence to the value proposition. For this reason, Saas providers considering to offer an API to external developers should take into consideration to release the software to the open source domain, as this has the same (or possibly better) effect.
Whether the above sufficiently aids organisations in their trade-off varies per organisation. Open source is not ‘suitable’ for every organisation due to cultural differences. That said; the results have already aided the organisation that has had access to the results of this thesis. This organisation decided not to release its software, due to the needed increase in support facilities. The focus will first be to improve the support to end-users, before adding support to external developers.

6.3 Limitations

In the design and execution of the research, disturbing factors have been controlled as much as possible to increase the quality of the research and the reliability of the results. Nevertheless, as with all research, this study has limitations, i.e. some weaknesses. These will be addressed in this section.

The goal of this thesis was to identify differences in business models with the question in mind what would happen if a SaaS provider released its software to the open source community. The four identified differences between open source and proprietary business models do not rule out other differences which were not detected. It is possible (though not very likely) that there are other differences that have not surfaced, but which have a larger impact on the organisation than those found. The propositions that were rejected by the cases of this thesis can also not be ruled out, it might be possible that the risks have occurred at other open source organisations. Some propositions could not be validated since the chosen cases did not conform to the prerequisites (such as proposition 6). It is advisable to find specific cases that do conform to the prerequisites of this proposition and analyse whether these risks were existent.

Even if all four differences are only taken into account the trade-off still results in an “it depends” answer when questioning which strategy would benefit the organisation most. It depends on the amount of collaboration the organisation is willing to accept, it depends on the type of organisation, it depends on the complexity of the software, it depends whether the software is capable of providing a platform to external developers, etc. This might be too vague to organisations not willing to invest time to gain understanding of their position in the trade-off. These organisations would benefit from a list of questions which results in an advice for their specific situation. This step was not taken due to lack of time in the research project.

Another limitation can be found in the research method used. Multiple-case study findings in general are not easy to interpret and the sample is far lower than for qualitative research. Therefore, one should be very careful in projecting the qualitative findings to a larger population. Although two different types of SaaS offering have been analysed, the cases have focussed on business-to-business cases. It is possible that business-to-consumer services might differ in business model and thus also in elements affected. To broaden the applicability of the results in other software fields the differences should also be studied in other SaaS markets. Even within the CRM and WCM software market, external factors could influence the business models of these organisations. To further prove the differences found in this thesis, it would be useful to execute a follow-up quantitative research.
Most of the data was gathered by desk research and interviews. Although the interviewees formed essential sources of information, they might also be biased in their perception of their own business. Especially privately owned companies are reluctant to share every bit of information on their business model and might not tell the whole truth. This was partially cancelled out by interviewing multiple people, but it could still be that some bias remained.

In this thesis the STOF model is applied in the context of SaaS business models, but originally the STOF model was aimed at mobile services. It is unsure to what extend the STOF model is completely compatible with SaaS business models. To overcome this uncertainty other literature on business models was also analysed. To organisations using a different definition of a business model, the outcome of this thesis would be of little value. To understand the results of this thesis, the organisation would first need to define their own business in the elements used in this thesis in order to understand what effect these differences have on their business.

### 6.4 Future Research

This section proposes several subjects for future research on the subjects of SaaS and open source software development.

One of the outcomes of this thesis is that open source SaaS providers charge lower prices. It has remained unclear whether this is something that is also expected by the customers of an open source SaaS provider, or that this is only a strategic choice made by the provider to gain competitive advantage. Additional research would be needed to understand whether customers would be willing to pay the same price for a service, regardless whether the underlying software is open source or proprietary. This would take a survey held with customers of SaaS providers.

From the conclusions of this thesis it shows that development cost can be lowered by open source SaaS providers, but the increased complexity raises costs. A future case study looking only at costs and revenues would be interesting to organisations looking to release their software as open source. This cost/benefit analysis should consider the lower pricing and elevated complexity of open source projects, versus the reduced costs of community developed functionality.

In chapter 3 no expectations were expressed in the technology domain. In hindsight however, it became clear that the element technical functionality plays a larger role in open source software than expected. Most contributions results in functionality not provided by the provider, but by extensions. An interesting subject for a research project would be the analysis of feature richness at open source providers and proprietary providers. In this analysis a distinction should be made between proprietary providers that do and do not offer an API to external developers.

In § 3.2.2 the expectation was expressed that the promotion of open source by governmental programmes have a positive influence on open source software. This expectation was not translated into a proposition, as this positive image was insufficiently supported by prior research. A survey would be needed to assure this relation. If this survey confirms the positive image of open source software, further research would be required to assess
whether this affects the business model of open source SaaS providers.

This thesis has focused on SaaS providers, since both open source and proprietary providers can use the same pricing structure. This provides a convenient way to compare the business models of both types of organisations. An important question is: To what extend do the findings of this thesis only apply to SaaS providers? Research in traditional software markets is needed to answer this question.

Since both open source cases in the case study did not release their software during the emerging phase, no conclusions could be drawn on proposition 6 on the risk when the software is released too early. Specific cases which did release their software during this phase have to be studied to examine whether this risk exists. An example of a SaaS provider that released its software is Scalr\(^1\).

For this thesis the STOF model was used as a starting point for the identification of relevant elements of business models. Originally the STOF model was developed with mobile services in mind. This explains the focus on multiple stakeholders in the value web of the business model. Future research could focus at the investigation where the STOF needs adaptation to use with services not provided over a mobile network. This would require the analysis of business models of different services and assess to what extent the elements can be found in these business models and the significance of these elements within the business model. Based on this thesis it would be expected that for SaaS services the value network will not be as important as in mobile services.

\(^1\)See http://scalr.net/


Automattic, How we work [online], 2009. Available from: [http://automattic.com/about/how-we-work](http://automattic.com/about/how-we-work) [cited 10-3-2009].


Beal, B., Open source crm hits milestone, still trails saas [online], 12 2006. Available from: [http://searchcrm.techtarget.com/topics/0,295493,sid11_tax306375,00.html](http://searchcrm.techtarget.com/topics/0,295493,sid11_tax306375,00.html) [cited 16-3-2009].


McGowan, D., There’s no such thing as free software (and it’s a good thing, too) [online], 10 2006. Available from: http://firstmonday.org/issues/special10_10/mcgowan/index.html [cited 18-02-2009].

Ministry van Economische Zaken (2007), ‘Actieplan ‘nederland open in verbinding”.


OECD (2008), ‘Broadband growth and policies in oecd countries’.


Salesforce.com (2009), Salesforce.com’s fourth quarter fiscal 2009 financial results, Quarterly results 4, Salesforce.com.


Appendix A

STOF Model

A.1 Introduction

The STOF model (Figure 3.2) and STOF method are part of the ‘STOF approach’, which are the result of research effort for the Freeband Communication Project Frux (Bouwman et al., 2008). It is aimed at “providing a structured approach to designing viable business models for electronic services” (Faber and de Vos, 2008).

STOF is an acronym for the four domains required in a business model: Service Domain, Technology Domain, Organisation Domain and Finance Domain. The main goal of a business model is to create value both for customers/end-users and for the service provider. The four domains contribute to this added value.

A.1.1 Domains

The Service Domain describes a number of elements of importance when defining the offering (the service), which can be found in figure A.1. The central factor in this figure is the Perceived Value as it is the result of many variables including the Expected Value and Intended Value.

The Technology Domain comprises the technical architecture which is specified by the service. The technical architecture consists of different design variables needed to deliver the intended service offering. Figure A.2 depicts these variables. Some of these are typical to mobile services, such as Access Networks and Backbone Infrastructure. In case of SaaS an internet connection is needed, but this is something the provider has no or little impact on. What is of importance for the provider is the platform the service is build upon, i.e. the choice whether to use cloud computing. Devices also play a bigger role in mobile services than in SaaS. A PC with web browser will be the most common device for SaaS users, however mobile browsers could play a bigger role in the future.

To deliver a service more is needed than technology. The Organisation Domain addresses organisational issues related to technology, marketing and finance. Even in situations where the service is offered by a single organisation, it is often the case that this organisation has to collaborate with others to be able to offer the service. As noted in the description of ‘business model’, a viable service business model needs to create value for
all actors involved, thus the organisational domain of the STOF model includes the roles and value activities of the actors which make up the value network (Figure A.3). Mobile services typically need to have more actors involved to deliver the service. Not only the service provider plays a role, but network operators, phone manufacturers and content providers are often involved. In the case of SaaS the number of actors can be as large, but the role of the actors other than the initiator will be of less influence. Besides the initiator a hosting provider and a billing partner are often involved. These are however contributing partners and can be replaced if needed, without having impact on the delivered service.

Finally, the **Finance Domain** aims to design a business model in which costs and revenues result in net benefits for the actors involved. The defined performance indicators provide a means to evaluate the financial arrangements. As can be seen in figure A.4, the risk in the other domains is aggregated in the finance domain, as all risks have their impact on the revenues. Allthough mobile services are often charged differently (through the network operator), the finance domain has no elements that are not applicable in the case of SaaS.

These four domains form the essential parts of a service business model according to the STOF model.
Figure A.2: Technology Domain (Bouwman et al., 2008)

Figure A.3: Organisation Domain (Bouwman et al., 2008)
A.1 Introduction

STOF Model

Figure A.4: Finance Domain (Bouwman et al., 2008)

Figure A.5: Phases in business models (Bouwman et al., 2008)

A.1.2 Phases

A business model is subject to change over time, influenced by the drivers ‘market’, ‘technology’ and ‘regulation’. This continuous process is depicted in figure A.5. The phases used in the STOF model, which the business model encounters, are ‘Technology R&D’, ‘Roll out’ and ‘Market’ (Bouwman et al., 2008).

A.1.3 STOF method

The STOF method aids the design process of developing a viable business model for a ICT service. The method consists of four steps leading from idea to business model. Using check-lists, it ensures the enclosure of all four domains in the business model description. The authors emphasise its importance in the early stages of business model development, but can be used to increase market potential in a later stage (Bouwman et al., 2008, pp. 115).
The check-lists used by the method are a useful means to thoroughly analyse the cases in our case studies in Chapter 5.
Appendix B

Case Study Questions

For the questions acquired in § 3.7 we will determine the possible sources of data.

The following abbreviations have been used:

- I = Interview
- A = Annual/Quarterly report
- B = Business plan
- S = (Scientific) Articles
- W = corporate website
- D = Software download
- G = Search engines

B.0.4 Service

*Open source SaaS providers charge lower prices*

- What level of services are offered at what price? W, I
- How were these prices determined? I, B
- Did the provider look at prices of competitors? I, B
- Does the provider compare its prices with competition on its website? W
- Did the open source provider take the lower cost of open source software into account whilst setting their prices? I, B

B.0.5 Technology

*The value activities not based on open source software are the only means for open source SaaS providers to separate them from competition*

- What value activities are deployed by the provider? W, A
  - Are these software related or non-software? W
Case Study Questions

- How are the value activities priced? \( W \)
- How does the provider differentiate itself from competition? \( W, I \)
- What are the advantages of the offered service as claimed by the provider? \( W \)
- What are its acclaimed USP’s? \( W \)

*Governance of an open source software requires more resources of the SaaS provider compared to a proprietary project*

Support

- Is there a department that supports external developers? \( W, I \)
- Of how many employees does this department consist? \( W, I \)
- Do external developers deliver support to each other? \( W \)

Software Development

- How is the development of software organised? \( W, B, I \)
- How much time is consumed by reviewing contributions? \( I, S \)

B.0.6 Finance

*Contributions of volunteers lower development costs for an open source SaaS provider*

- What do the development costs consist of? \( W, I \)
- How many lines-of-code does the current version of the project count? \( I, D \)
- How many developers are employed? \( A, I \)
- How many developers are developing new features? \( I \)
- How many developers are assigned to test the software? \( I \)
- How many developers are assigned to fix bugs? \( W \)
- How many contributions are made per month? \( W, I \)
- Does the quality of the contributions reflect the standards demanded by the organisation? \( I, S \)

*To protect themselves from competition the open source SaaS providers should keep the source of the software that provides scalability closed*
Case Study Questions

- Did the provider release the multi-tenant part of the software as open source? *W, I*
  In case it was not released:
  - Have competitors developed this software themselves? *G*

  In case the software was released:
  - Is the software being used by competition to offer the same service? *G*

- Are there competitors offering the same service? *G, I*

- Do these competitors serve the same market segment? *W*

- Have clients switched to the competition? *I*

- If so, what was the reason? *I*

Releasing the software to the open source community in the ‘emerging phase’ of the technology life cycle imposes the risk that the competitive advantage is lost

- In which phase of the technology life cycle was the software released?
  This can be determined by:
  - How many equivalent services were offered at that time? *I, S*
  - What kind of users used the service? Early adopters? *I, S*
  - Did the service offer something unique (disruptive) over the predecessors? *I, S*
  - Did the provider change relatively high prices compared to current pricing? *I, W*

- Did the release of the software enable competitors to offer the same service? *G, I*

- Did competitors copy solutions used in the software? *G, I*

- Did average prices drop after release to the open source community? *S, I*
Appendix C

Interviews WordPress

C.1 Matt Mullenweg

Matt Mullenberg is the founder of Automattic and ‘architect’ of Wordpres. The interview was held between February 12th, 2009 and March 19th (several e-mails have been sent). The transcript is a summary of these emails.

C.1.1 Transcript of interview

Brocker: The goal of this interview is to gain understanding of business models behind SaaS providers that are based on open source software. The questions will focus on the business behind the service (the provider, in this case Automattic) and the software itself.

C.1.2 Software Development

Brocker: How is the software development and developer support organised within Automattic and Wordpress?

Mullenweg: *It’s a flat organisation. Developers have their speciality and there is no hierarchy, no senior versus junior developers.*

Brocker: The support you offer to external developers working on Wordpress is extensive. Do you have employees dedicated to supporting these external developers? If so, how many?

Mullenweg: *Our support is dedicated to Wordpress.com and the VIP clients. Our developers work with other developers and provide help if needed, but no-one is dedicated.*

Brocker: Could you make an estimate what percentage of support is delivered by Automattic and what factor is done by volunteers helping each other?

Mullenweg: *Almost all is done by volunteers*

Brocker: The Wordpress.org website shows most lead developers and contributing developers are working for Automattic. How much time of lead developers is consumed reviewing the contributions added to the trac?

Mullenweg: *It differs. Sometimes a developer is working on a specific project, neglecting the trac for days, other times he’s hunting bugs in the trac full-time*
Brocker: Does the quality of these contributions meet the quality standards as demanded by the lead developers, or does a lot of the code have to be rewritten to meet the standards?

Mullenweg: Of course some has to be rewritten, but a lot is contributed by experienced developers that have made many contributions to Wordpress. Automattic hires those that do best.

Brocker: Could you make an estimate how much of the current source code of Wordpress is produced by Automattic team members and how much is obtained from volunteer contributions?

Mullenweg: It's very hard to draw a line. Someone at Automattic working on his own plug-in for his own blog at night is also a volunteer. There's no clear line, were all one big community ;)

Brocker: Any rough estimate will do...

Mullenweg: Ok, at least 40%, probably more

Brocker: You have also released the software behind Wordpress.com (Wordpress MU).

Are there any competitors of Wordpress.com offering the exact same service using the Wordpress MU software that you know of?

Mullenweg: There are some

Brocker: If so, do you know of any Wordpress.com users having switched to these competitors?

Mullenweg: It could be possible that some Wordpress.com users of the free edition have switched to these niche providers, but this does not affect the revenue's of Wordpress.com

C.2 Toni Schneider

The interviewee is Toni Schneider, CEO of Wordpress. The interview was held on April 10th, 2009 by email.

C.2.1 Transcript of interview

Brocker: The first aspect is the pricing of the service on Wordpress.com. You charge for several upgrades on the free account. How were the prices of these upgrades determined?

Schneider: Mostly by looking at other, comparable online services.

Brocker: Did the ‘free’ aspect of open source play a role when you decided that the basic Wordpress.com blogs should be free?

Schneider: Not really. It had to do with us being able to do it for free (and somewhat with Blogger being free as well).

Brocker: What separates Wordpress.com from your competitors (both competitors offering hosted Wordpress and SaaS providers like Blogger and TypePad)?

Schneider: More features, less spam, better support, better SEO, bigger community.

Brocker: How is the software development and developer support organised within Automattic and Wordpress?

Schneider: It's a flat organization.

Brocker: How many people does Automattic count? And how many of those are working on Wordpress
Schneider: Wordpress is 90% of our business. We currently hold 36 people all over the globe.

Brocker: The support you offer to external developers working on Wordpress is extensive. Do you have employees dedicated to supporting these developers, or is this part of the job of the developers with Automatic?

Schneider: Automattic developers contribute to WordPress just like anyone else, so we have plenty of incentive to help developers. It’s part of our on-going work at Automattic.

Brocker: Could you make an estimate what percentage of support is delivered by Automattic and what factor is done by volunteers helping each other?

Schneider: Do you mean WordPress.org forums? That’s all volunteers.

Brocker: The Wordpress.org website shows most lead developers and contributing developers are working for Automattic. How much time of lead developers is consumed reviewing the contributions added to the trac?

Schneider: Not that much.

Brocker: Does the quality of these contributions meet the quality standards as demanded by the lead developers, or does a lot of the code have to be rewritten to meet the standards?

Schneider: Low quality code gets rejected.

Brocker: Could you make an estimate how much of the current source code of Wordpress is produced by Automattic team members and how much is obtained from volunteer contributions?

Schneider: Not sure.

Brocker: You have also released the software behind Wordpress.com (Wordpress MU). Are there any competitors of Wordpress.com offering the exact same service using the Wordpress MU software that you know of?

Schneider: Several people have tried, but no one has gotten very big. There are some niche focused ones like edublogs.org that are doing well.

C.3 Peter Westwood

Peter Westwood, Lead developer of Wordpress, not employed by Automattic. Interview was held on April 1st, 2009, through IRC. The transcript is not edited.

C.3.1 Transcript of interview

Brocker: If you had to make a rough estimate, what percentage of the code base of WP is developed by the community and what % is developed by Automattic?

Westwood: I see automattics contributions as part of the community [wink]

Westwood: It is hard to tell realistically. 3 out of the 5 lead developers work for automattic but I wouldn’t necessarily class everything they do as developed by automattic

Westwood: it’s not like they clock off and don’t contribute outside office hours

Brocker: ok, but outside of the lead developers, do you get many contributions of external developers?

Westwood: yes we get a lot. a good portion of the lead developers time is spent on reviewing and committing changes made by others
Brocker: good portion would be about what %?
Westwood: *for me over 80% of the time I spend on WP is reviewing, updating, commenting, committing code the community has written*
Nicola Greco: *more are the contributions, more lead developers spend time on reviewing the code. That’s the gpl power*
Brocker: ok, cool
Westwood: *If you look at the commit messages you will find a high proportion have a comment like Props/hattip and the username of a community member who has contributed to the change by providing a patch*
Brocker: yep, saw that... and its only the lead developers that can commit?
Westwood: yes
Westwood: *we have had branches in the past where we have opened up access*
Westwood: *but we don’t have any at the moment*
Brocker: ok, did that work? Was the quality up to standards or did you have to rewrite a lot?
Westwood: *there is also the plugins repo which has a lot more contributors*
Westwood: *the purpose of the branch was to develop functional UI mockups for A <> B testing and most of the core was merged with trunk later*
Brocker: Peter, thanks for your info!
Westwood: *NP!*
Appendix D

Interview Squarespace

D.1 Dane Atkinson

The interviewee is Dane Atkinson, CEO of Squarespace. The interview was held on April 3rd, 2009.

D.1.1 Transcript of interview

Brocker: The goal of this interview is to gain understanding of the relevant elements of business models of SaaS providers. This interview is part of a case study on Squarespace, one of the four case studies of the thesis.

Pricing

The first aspect of the service is the pricing, how did Squarespace determine the pricing of the service? Did you look at any specific competitors?

Atkinson: *Yes we reviewed the pricing of who considered competition and then raised our 20%+. We offer a better premium product and wanted that reflected in the price as well, without being crazy about it.*

Brocker: Who do you consider your main competitors?

Atkinson: *We enable online publishing and cloud reliability. That means the guy down the street, razorfish, typepad, wordpress, dreamweaver are all competition. In the blog vertical it is really typepad and wordpress.*

Brocker: What are the USP’s of Squarespace that separates it from its direct competitors?

Atkinson: *Our product is simpler and allows all skill levels to achieve better results then any of our competitors from a technology standpoint*

Brocker: Which price plan counts most of your customers?

Atkinson: *Our average customer pays 20$ a month*

Software development

Brocker: How is the software development organised within Squarespace?
Atkinson: Each engineer is responsible for their aspect of the application, that direct ownership breeds responsibility and care.

Brocker: Apart from the support delivered to your customers, do you also deliver support to external developers (or is there no way they can interact with your software)?

Atkinson: 20% of our customers have multiple accounts we service designers and developers. They have an online community in addition to our existing support.

Brocker: How many employees are dedicated to support and how many are allocated to the development of Squarespace?

Atkinson: 4 are support and 10 are app building

Brocker: Could you make an estimate on how many man hours have gone into the development of the software behind Squarespace.com?

Atkinson: 10 man years

Brocker: Could you make an estimate on how many lines of code the software consists of?

Atkinson: 100s of thousands

Brocker: Do you get a lot of feedback from users on bugs, suggestions, etc (in numbers per week)?

Atkinson: Yes, mostly suggestions, but all total 50 a week that make it through.
Appendix E

Interviews Sugar CRM

E.1 Chris Harrick

The interviewee is Chris Harrick, Vice President of Corporate and Product Marketing. The interview was conducted over Skype on April 30th, 2009. The questions were sent to the interviewee in advance.

E.1.1 Transcript of Interview

Brocker: The goal of this interview is to gain understanding of business models behind SaaS providers that are based on open source software. Although Sugar CRM offers different deployment options, these questions focus on the on-Demand version of Sugar CRM, the organisation behind the service and the software itself.

Pricing

Brocker: The first aspect is the pricing structure of Sugar CRM On-Demand Professional and Enterprise. How was the pricing of Sugar CRM On-Demand determined? Was this influenced by the pricing of Salesforce?

Harrick: SugarCRM pricing was determined by an analysis of the price it takes to acquire and support customers as well as in-depth market research into competitive pricing of a variety of vendors.

Brocker: Did Sugar put a lower price on its services than e.g. Salesforce because it is open source and is therefore ‘expected’ to be a lower-cost alternative?

Harrick: SugarCRM is able to offer lower prices because our acquisition costs are lower due to our open source model (pull versus push marketing).

Brocker: Who are Sugar On-Demands main competitors?

Harrick: Salesforce.com, Microsoft and Act!

Brocker: Why doesn’t Sugar offer an On-Demand version of the Community Edition?

Harrick: We just announced Sugar Express, which is Sugar Community Edition plus MSFT Office Plug-ins and Support On-Demand. It took significant investment in our on-demand services to be able to offer this product at scale and a reasonable price point.
Brocker: What percentage of revenue is generated by the On-Demand editions?
Harrick: *About 50%*

Brocker: The website lists 150 employees, how many of these are dedicated to software development? And how many are dedicated to offering support to On-Demand customers?
Harrick: *45 Engineers, 20 customer support*

**Software development**

Brocker: SugarCRM maintains a large community (sugarforge.org) of both end-users and volunteer developers of the open source edition. How many employees are devoted to the support and maintenance of sugarforge.org, the wiki, the forums and the Sugar Developers zone?

Harrick: *Our core engineering team work on the CE product and a team of 10, consisting of engineering, product management and support live on the forums and forge.*

Brocker: Could you make an estimate what percentage of support of open source users and developers is delivered by Sugar and what factor is done by volunteers helping each other? (this does not concern the support to professional and enterprise end-users and resellers)

Harrick: *All support for CE users is offered through the Forums. I do not have an exact breakout % of forum support threads.*

Brocker: Are all of the developers that have rights to commit changes to the source code of the core employed by Sugar CRM?

Harrick: *Yes, or they sign away their rights to the software so that we can offer our customers full copyright protection.*

Brocker: You provide a form\(^1\) to external developers wishing to contribute to the core of the community edition, how many contributions are made by external developers (on a weekly basis)?

Harrick: *Most developer contributions are through extensions and there are more than 600 of them.*

Brocker: How much time is consumed reviewing these contributions?

Harrick: *This is done through extensions on the Forge these are reviewed by our team but are not included in the core product distribution.*

Brocker: Is the quality of these contributions up to the standards of the lead developers, or do most of them have to be rewritten?

Harrick: *It is a meritocracy. There will always be good and bad extensions. The former rise to the top while the latter run out of momentum.*

Brocker: Per saldo, could you say that these contributions lower the development cost? Or do the benefits of open source lie in other fields (like lack of vendor lock-in and higher rates of adoption)?

Harrick: *Extensions enrich the Sugar ecosystem which creates value. It also lowers of distribution and marketing costs because people can evaluate the software without running through the normal resources of an enterprise software sales cycle.*

\(^1\)http://developers.sugarcrm.com/contribute.php
Brocker: Do you know of any functionality in SugarCRM that has been copied by competitors like Salesforce?
Harrick: Yes, they copied our Forums with Idea.salesforce.com, our Forge with AppExchange and our AJAX UI (though this was a larger market trend). I am sure there are many more.

Brocker: Did SaaS competitors with proprietary software lower their prices after Sugar introduced the On-Demand version?
Harrick: They definitely discount more heavily when competing against us. And MSFT just did announce a price cut for their CRM product.

Brocker: Thanks for your time.

E.2 David Wheeler

Software Engineer, SugarCRM. Interview held on March 26th, 2009 on the IRC channel of SugarCRM

E.2.1 Transcript of interview

Brocker: I have some questions on the software development within Sugar for the thesis i’m writing for my master of science. Could I direct them at you?
Wheeler: good question, i’ll give it a try

Brocker: How many employees are devoted to the support and maintenance of sugarforge.org, the wiki, the forums and the Sugar Developerszone?
Wheeler: I think you might need to email contact@sugarcrm.com for those type of questions

Brocker: I have
Brocker: but maybe you could answer this more practical question: are there any developers not employed by Sugar that have the right to commit changes to the community edition?
Wheeler: Community Contributions are handled through http://developers.sugarcrm.com/contribute.php

Brocker: And do you have any number on the quantity of contributions that are made?
Wheeler: I’m sorry, we do not provide that kind of information

Brocker: Ok, thanks for your time
Appendix F

Interviews Salesforce.com

F.1 Hans Brouwer

Interviewee: Hans Brouwer, General Manager Salesforce Benelux. This interview was held on April 6th, 2009.

F.1.1 Transcript of Interview

Brocker: The goal of this interview is to gain understanding of the relevant elements of business models of SaaS providers. This interview is part of a case study on Salesforce.com, one of the four case studies of the thesis.

Pricing

Brocker: Who are the main competitors of Salesforce.com?

Brouwer: That would be both traditional CRM products as any On-Demand providers

Brocker: Has lower (Microsoft or SugarCRM) or higher (Sap or Oracle) pricing of competitor SaaS providers influenced the current pricing of Salesforce.com?

Brouwer: As far as I know Salesforce has not adjusted its pricing the past several years

Brocker: What is the most sold edition of Salesforce?

Brouwer: I can only speak for the Benelux, where is is the Professional Edition. You might find information on this in the SEC filings

Brocker: Is the On-Demand version of SugarCRM of significant threat to Salesforce?

Brouwer: Any CRM product is a threat. But Sugar seems to be targeting at a different target group. We don’t encounter them very often in the Benelux

Developers

Brocker: How many developers does the R&D department count?

Brouwer: All development takes place in the US, you would have to inquire them for details

Brocker: How is the R&D department organised?

Brouwer: Same story

Brocker: How many feature requests or bug reports does the R&D department receive from end-users (in numbers per week)?

Brouwer: I have no exact notion of those numbers for the Benelux, but I do know these are periodically shared with the R&D in the States

Brocker: Besides the traditional support to end-users of the application, you also of-
fer support to external developers developing applications and extensions. How many employees are dedicated to support these external developers (both direct support as support through the discussion boards)?

Brouwer: We only deliver direct support to the end-users, developer support is only done in english and handled in the States

Brocker: How many employees are allocated to the maintenance of all API documentation and the wiki?

Brouwer: I don’t have any information on that

F.2 Shelley Gordon

The interviewee was Shelley Gordon, referred to by Gordon Evans (Public Relations Salesforce.com). The interview was held on April 8th, 2009 by telephone.

F.3 Transcript of Interview

Brocker: The goal of this interview is to gain understanding of the relevant elements of business models of SaaS providers. This interview is part of a case study on Salesforce.com, one of the four case studies of the thesis.

Pricing

Brocker: Who are the main competitors of Salesforce.com?

Gordon: The main competitors are of course listed in our annual reports, but when it comes to ranking them in importance, the On-Demand providers are most important.

Brocker: Has lower (Microsoft or SugarCRM) or higher (Sap or Oracle) pricing of competitor SaaS providers influenced the current pricing of Salesforce.com?

Gordon: Of course we keep an eye on competition, but I wouldn’t say that we let this directly influences our own pricing. We have been very consistent with our pricing over the years which show the believe in our philosohy. I think our pricing is competitive and I assume competitors have adjusted their pricing to ours.

Brocker: What is the most sold edition of Salesforce?

Gordon: That would be the Professional edition. But keep in mind, there’s the very big contracts with multinationals that get SLA’s that don’t fall into one of our standard editions.

Brocker: Is the On-Demand version of SugarCRM of significant threat to Salesforce?

Gordon: I could say we occasionally run against Sugar on deals, but most of the time they don’t form a threat. We have several advantages over them that come into play on a business deal. We have revolutionised the industry and have been in business 10 years. Sugar is pretty young, i think they are only 5 years old. Organisations often want a big partner, Sugar is still pretty small. But of course, there are plenty of organisations that favor the lower pricing or the open source nature, you can’t have em all!

Developers

Brocker: How many developers does the R&D department count?

Gordon: I can’t make any statements on such kind of figures. It would be too easy to assume that 9% of budget also means 9% of all personel. We’ve hired 1400 people last year and a great deal is on R&D.

Brocker: How is the R&D department organised?
Gordon: The R&D department is split up in small clusters that work on small problems. We use the SCRUM or agile development method to be able to deliver new versions 4 times a year.

Brocker: Does the R&D department receive many feature requests, suggestions or bug reports?

Gordon: Loads! We’ve got a website, ideas.salesforce.com where our customers can enter suggestions and vote for the ones they would like to see implemented first. Our developers can comment on which ideas are taken into consideration. We believe listening to our customers is crucial to the success of our service.

Brocker: Besides the traditional support to end-users of the application, you also offer support to external developers developing applications and extensions. How many employees are dedicated to support these external developers (both direct support as support through the discussion boards)?

Gordon: Again, I can’t comment on exact numbers. But support has two divisions, one for developers, the other for end-users. We try to service both at our best.

Brocker: How many employees are allocated to the maintenance of all API documentation and the wiki?

Gordon: I really don’t know. It is organised with the support department and the development department, but wouldn’t be able to give you any numbers on that.