NTNU – Trondheim
Norwegian University of
Science and Technology

Coordination of Asset Development in Software Ecosystems

Torbjørn Aarlott Aase

Master of Science in Informatics
Submission date: June 2015
Supervisor: Eric Monteiro, IDI
Co-supervisor: Vidar Hepsø, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

This thesis looks into the academic literature of customer co-creation and software ecosystems, in particular the governance of such ecosystems. A privately owned, commercial software ecosystem built around software platform(s) in use by the E&P industry is used as a case for further exploring the usefulness of frameworks and tools found in the academic literature. The ecosystem is analysed both for academic purposes as well as in regards to the particular problem of coordinating asset production within software ecosystems. A proposed solution to this problem is provided towards the end of this thesis, but no technical deployment of it and the relevant tools is done or evaluated. There does, however, seem to be some potential in the application of existing ecosystem literature.

Preface

This thesis is submitted for consideration as the finalization of my informatics master's degree programme at the Department of Computer and Information Science(IDI) at the Norwegian University of Science and Technology(NTNU).

I would like to thank my supervisor at IDI, Eric Monteiro, for his help and guidance through the process. In addition, I would also like to thank Trond Benum and Ahmed Aqrawi at Schlumberger for their assistance and sponsorship of this thesis.

Table of Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Motivation, structure and contribution

1.1 Motivation

Software ecosystems are becoming an ever larger part of the software landscape. From apps on smartphones to plug-ins for your browser, ecosystems facilitate much of the extendible functionality we have come to expect in the technology we use. By providing third party developers the ability to extend functionality and create additional value for their product, companies are seeing increased innovation without needing the proportionate investment and risk in R&D. The third party developers enjoy low development costs and an established customer base, and the users of the product enjoy a customizability and variety not previously possible. All due to the interactions and qualities inherent in software ecosystems that include these parties. Yet, their inner workings, even existence as concepts, has gone largely unnoticed by the general public, and even academia to some extent.

As more and more companies move towards turning their traditional software products into software platforms that allow for ecosystems, such research becomes increasingly important. This thesis aims to contribute to the growing academic field of software ecosystems, as well as make use of the existing literature in an actual industry setting.

1.2 Structure

Following part I, which is this introduction, this thesis goes on to part II, the literature review. This literature review gives an introduction to the concept of software ecosystems and platforms as well as some of the proposed academic models and frameworks that exist. Some familiarity with software development on the part of the reader is assumed, but not strictly required. The literature review also touches on the concept of customer

involvement in product innovation. Part III describes the process that went into the creation of this thesis. Part IV outlines the case of the Ocean ecosystem and problem of duplicate assets. Finally, parts V and VI apply the frameworks and models presented in the literature review to the Ocean ecosystem, and discusses a possible solution to the problem presented earlier.

1.3 Contribution

The academic field of software ecosystems has gained prominence only recently, and can be viewed as fairly young. This is the case in particular for software ecosystem governance. One of the major contributions of this thesis will be to provide yet another case study for the field of ecosystem governance, and utilizing the existing models and frameworks in the literature, adding to their maturity.

The second contribution of this thesis is to propose a solution to the problem of the specific case presented. Ideally, the solution is specific enough to be useful, but also general enough to be able to be applied in similar cases in other ecosystems.

1.4 Research Goal

The research goal of this thesis is to **add a case study to the academic software ecosystem space, while exploring the available academic tools for ecosystem governance and problem solving in an actual industry setting**. gain an increased understanding of software ecosystem governance and its application in an industry setting

From this we derive the research questions:

RQ1: How is governance applied in a real-world setting?

RQ2: What appropriate tools for governance exist in the literature and how do they perform?

Part II

Literature Review

Chapter 2

Software ecosystems

2.1 What are software ecosystems?

In order to discuss software ecosystems, sometimes referred to as platform ecosystems, we first have to establish what is meant by *platform*. In this thesis the definition used is based on the one outlined in Tiwana (2014):

> A software platform is a software-based product or service that serves as a foundation on which outside parties can build complementary products or services.

Such platforms are commonly associated with operating systems for computers and mobile devices, popular examples being Microsoft's Windows, Apple's iOS and Google's Android. In all three cases the platform owner allows third parties to distribute software, commonly known as **apps**, that add functionality to or extend already present functionality in the platform.

The platform and its complementary software, in the form of apps, plug-ins, extensions and so on, make up the two major elements of the *ecosystem*, though the exact definition of what constitutes an ecosystem is debated (Manikas and Hansen, 2013). Within this ecosystem there are three principal sets of actors. The **platform owners**, sometimes referred to as keystone companies, that build a community of **developers** and **end-users** around their core product or product line. As platforms and their functionality grow in complexity, it is also common for some of these developers to create their own internal platform abstraction layers or become platform owners themselves (Romberg, 2007) (Gawer, 2014).

According to Bosch (2012) the ecosystem approach to software development has become more popular recently, with many companies opening up their product or product lines as cores for platforms. These products may be proprietary software, open-source, in-between or in some cases transitional from one state to the other(Kilamo et al., 2012).

Bosch (2012) outlines two main drivers behind a company or organization moving towards a software ecosystem: Firstly, a company realizes that the amount of functionality that needs to be developed to satisfy the needs of their customers is far more than what can be built in a reasonable amount of time and with an R&D investment that offers an acceptable return on investment. Secondly, the mass customization trend drives the need for a significant R&D investment for successful software applications.

2.2 Multi-sided markets and network effects

From an economic perspective the platforms discussed here can be viewed as **multi-sided markets** (Gawer, 2014), indeed this multi-sidedness is one of their key properties (Tiwana, 2014). Multi-sided markets, more specifically two-sided markets in the case of many platforms, are characterized by their ability to facilitate direct contact between two distinct groups of stakeholders, not counting owners. Hagiu and Wright (2011) also differentiates between degrees of multi-sidedness depending on how significant the value created by this facilitation is to the platform owner, in comparison to the value created by other goods or services offered. They go on to name shopping malls, video game consoles and eBay as some examples of multi-sided platforms.

Tiwana (2014) points out that the viability of a multi-sided platform is in part dependent on the degree to which the two sides need to interact. Both could in theory find and trade with each other independently, and this process needs to be a costlier alternative for the platform to be successful.

A large factor in multi-sided markets is the fact that they are subject to so-called **network effects** which arise between the sides of the market (Gawer, 2014). Network effects refer to the degree to which every additional user makes the platform more valuable to each existing user (Tiwana, 2014). A common example of **direct network effects**, also called same-side network effects, is the value of communication channels, such as phones or social media, based on user count. Their value to each user is largely determined by the number of peers using the same system, as a social media account is useless for communication unless other users also have accounts. Each subsequent user with an account then increases value for existing users as well as make the social media site more attractive to other users, triggering a self-reinforcing feedback loop (Gawer, 2014).

In much the same way that direct network effects can make a system more valuable to users based on the number of other users of the same type, in multi-sided platforms **indirect network effects** can also increase the value for users of one side of the platform based on the number of users on the other side. Indirect network effects, also called cross-group or cross-side network effects, come into play when there is an interdependency and complementarity between the demands of the two sides (Gawer, 2014). A common example is smart phone users and developers for smart phone apps. The more developers there are developing complementary apps for a certain mobile platform, the more attractive

that platform becomes for phone users. At the same time, the number of users there are on the platform also influences its attractiveness to developers (Tiwana, 2014).

In addition to the positive network effects mentioned above, there are also negative network effects (Tiwana, 2014). These negative effects exist when each additional user negatively affects the system's value for other users. Examples of this are a large number of users slowing down a network, many cars on a highway making it more difficult for each driver to navigate traffic or market saturation keeping developers away from a platform.

It is also possible for a platform to be affected by network effects from one of its complements. The presence of an app with strong network effects, such as Skype, will make all platforms that support it more attractive to potential users.

According to Gawer (2014) a purely economical perspective is problematic when dealing with technological platforms and their evolution in particular. They argue that in most economic models of two-sided markets, platforms are taken to be both exogenous and fixed, not offering much insight into how or why they evolve. In addition they mention that the nature of the relationship between the platform owner and the two sides gets reduced to a seller-buyer relationship that views both developers and end-users as consumers. Gawer (2014) claims this has the following limiting consequences for further study into platform evolution:

> (a) all forms of competitive interaction between a platform owner and its own complements developers are left unexamined (as outside the scope of these models); (b) the very existence of complementarity-in-demand between the different consumer groups (the foundational network effects) is taken for granted, deemed exogenous, and assumed to be unchanging; (c) the existence of the platform itself is also taken for granted, exogenous and unchanging; (d) the impact of platform design on developers incentives to innovate is left untreated.

2.3 Types and classification of software ecosystems

2.3.1 Taxonomy

Bosch (2009) outlines an early taxonomy intended to help classify software ecosystems. This taxonomy organizes ecosystems in a two-dimensional space, with one dimension referred to as the category or abstraction layer and the other dimension being the underlying platform or base technology. The latter dimension is fairly self-explanatory in its presentation in **Fig. 2.1**, but the former warrants further explanation.

The three categories of ecosystems presented by Bosch (2009) are **operating systems**, **applications** and **end-user programming**.

category / platform	desktop	web	mobile
end-user programming	MS Excel, Mathematica, VHDL	Yahoo! Pipes, Microsoft PopFly, Google's mashup editor	*none so far*
application	MS Office	SalesForce, eBay, Amazon, Ning	*none so far*
operating system	MS Windows, Linux, Apple OS X	Google AppEngine, Yahoo developer, Coghead, Bungee Labs	Nokia S60, Palm, Android, iPhone

Figure 2.1: Software ecosystem taxonomy (Bosch, 2009)

Operating System-centric software ecosystems are characterized by being domain independent and typically being optimized for stand-alone applications to be deployed, with little support for cross-application integration. These applications are the basis for the domain specific uses that make the OS attractive to users, meaning that the ecosystem is very dependent on the number of actors. They are also heavily dependent on the actual devices that have the OS installed, with the exception of web-based ecosystems. Ecosystems based on operating systems for desktops and mobile devices are affected by the number of devices that run the OS, but also in changes to the technology surrounding those devices causing compatibility issues.

Application-centric software ecosystems generally start of as successful domain-specific applications. As the application and its user-base grows, there are an increasing number of requests for increasingly specific functionality. Depending on the volume and nature of these requests, it may not be feasible to develop these features in-house. This might prompt the owners to open up the application by providing APIs to third parties, turning the application into a domain-specific platform around which they can build an ecosystem. Unlike the operating system-centric ecosystems, the application-centric ecosystem is more dependent on the third party developers extending the functionality already present in the platform, as opposed to adding their own specific functionality. Since the original user-base is there because of the already present functionality, many users may not see the merits, or indeed be aware, of the extended functionality. It is therefore important for the platform owner to actively facilitate visibility and interaction between end-users and third party developers, in order to make it an attractive platform to develop for.

End-user programming software ecosystems are fairly small in comparison to operating system-centric and application-centric ecosystems. Most take the form of domain-specific programming languages (DSL), either graphical or textual. They are primarily focused on developers with good domain understanding, but little pro-

gramming skills. As such they are meant to be easy to develop for, but not particularly powerful in the sense of being able to introduce new or advanced functionality.

2.3.2 Classification

Jansen and Cusumano (2012) propose a classification model for software ecosystems based on four classification factors; Base technology, coordinators, extension markets and accessibility.

Base technology: According to Jansen and Cusumano (2012) their survey suggests that all software ecosystems are underpinned by some base technology. The three types of technology identified here are **software platforms**, **software service platforms** and **software standards**.
Most software ecosystems are based on the first type and have a specific software platform as their base technology. On occasion they are underpinned by several platforms, such as in the Microsoft ISV (Independent Software Vendor) ecosystem which is underpinned by a variety of Microsoft platforms (Sharepoint, Exchange...).
A software service platform, the second technology, is any platform based around some sort of online-only service, around which other ecosystem participants can gather, but not host on their own.
The third type of base technology being used is the software standard, as is the case for ODA(Open Design Alliance) which bases most of its technology around the DWG(drawing) standards from AutoDesk.

Coordinators: This classification factor refers to the effective owners and major decision makers of the ecosystem. An ecosystem is either owned by a **community(consortium)** or owned by a **private party**. The Eclipse ecosystem is an example of a community controlled ecosystem, where the Eclipse consortium represents the wishes of its members. An example of a privately owned ecosystem would be the iOS ecosystem, where Apple exerts definite control. Jansen and Cusumano (2012) points out that ecosystems where the base technology is privately owned by a commercial actor, tend to be controlled by that actor.

Extension markets: Many software ecosystems are centralized around a market of extensions (see section **2.6**) , also known as app stores or app markets. The classification model divides this factor into five possible situations; **No extension market, a simple list of extensions, an actual extension market, a commercial extension market, and multiple extension markets**.
In the case of no extension market, the components may be available through known third parties. As an example of this Jansen and Cusumano (2012) mentions that the Autodesk community site does have a short list of components, but most extensions are available only through third parties that do their own marketing.
In the second case, a list of extensions, a simple list of available extensions can be

accessible through a web page, for instance. Depending on the accessibility of the ecosystem(see below), this list may be curated to some degree.
The third case is where there exists an actual extension market, through which it is possible for developers to distribute and/or monetize their extensions. The Firefox Extensions market is presented as an example of this, where third parties can offer extensions, but without having to pay Mozilla.
Case number four is where a commercial extension market is used. These markets are similar to the previous type of market, but here the owner uses the market to make money. The two major mobile markets, the iOS App Store and Google Play, are examples of this.
The fifth and final case presented in the model is when an ecosystem includes multiple extension markets. An example given of this is the ecosystem around the game World of Warcraft, where the owner did not want to create their own extension store and left it to the community to create solutions for distribution, resulting in multiple markets.

Accessibility: Accessibility refers to the barriers to entry that determine which participants, or types of participants, will have a part to play in the ecosystem. The model presents three possibilities for the accessibility factor: **Open source, screened but free, and paid**.
In an open source ecosystem it is possible to add contributions to a project, create and publish components in the extension market and otherwise be involved, without any barriers.
A screened, but free, ecosystem has some sort of quality control to make sure that contributions are of the right quality and fit.
The most restrictive case is the paid ecosystem, where entry to it requires some form of payment.

Table 2.1 shows the classification of several existing ecosystems using the proposed classification model. Jansen and Cusumano (2012) do acknowledge some potentially important elements are excluded from the model, such as network effects, switching costs, multi-homing and whether or not there are more than two sides to the market.

Name	Base technology	Coordinators	Extension market	Accessibility
AutoCAD plug-ins	platform	privately owned	a list	paid
Ubuntu	platform	consortium	multiple markets	for free
Android	platform	privately owned	a commercial market	screened
iOS	platform	privately owned	a commercial market	paid
Eclipse	platform	consortium	a market	screened
XBMC	platform	consortium	multiple markets	for free
Joomla	platform	consortium	a list	screened
GX	platform	privately owned	a market	screened
Ruby	platform	consortium	multiple markets	for free
Ogre3D	platform	consortium	a list	screened
MS ISV Partners	platforms	privately owned	a commercial market	paid
Wordpress	service platform	consortium	a market	screened
HubSpot	service platform	privately owned	a commercial market	screened
SalesForce	service platform	privately owned	a commercial market	screened
Spotify	service platform	privately owned	a market	screened
World of Warcraft	service platform	privately owned	multiple markets	for free
XBRL	standard	consortium	a list	paid
Open Design All.	standard	consortium	a list	paid
OSGi	standard	consortium	a list	paid

Table 2.1: Examples of classified ecosystems by Jansen and Cusumano (2012)

2.4 Benefits of platforms

Platforms with surrounding ecosystems offer several benefits, not only to the platform owners, but also the developers and end-users (Tiwana, 2014). **The owners** enjoy massively distributed innovation on the platform, while at the same time transferring the financial risk to the developers. Highly specialized software can be made available on the platform without the owner needing to bring in domain experts or even be aware of the niche. The constant influx of outsiders to the development scene also allows the platform long-term sustainability as it will be better able to evolve and respond to the changing needs of the user base. The network effects of users and developers also offer a certain security after the number of users and/or developers has reached a tipping point (Gawer, 2009).

For **the developers** the platform often offers a generic foundation for development. This cuts down on development time that would have been spent on developing this base functionality, and reduces the barrier to entry. It also gives the developer access to an established market, with users already familiar with the base functionality. Because of this, a small developer would be able to reach a disproportionate number of users through the platform, compared to traditional distribution. The platform also usually offers the developer an established infrastructure with payment mechanisms, further reducing necessary development (Tiwana, 2014). A study by Ceccagnoli et al. (2011) also found that while there are benefits for smaller developers opting into an active ecosystem, these benefits are in part dependent on their own capability for downstream marketing and IPRs such as patents and copyrights.

End-users enjoy four distinct benefits from platforms (Tiwana, 2014). Firstly, the platform offers a degree of customizability usually not found in individual products not associated with platforms. A smartphone user, for instance, is able to add/remove functionality to/from their device until it fits their specific needs, be it a scientific calculator or social media software. Secondly, the user can expect their product to evolve and get access to additional features over time, as the developers continue to innovate in the competitive market. The third benefit to the user comes in the form of expected price drop for these features, as both interplatform and intraplatform competition push prices down. Finally, the fourth benefit to the end-users is a reduced search cost - the cost prior to transaction. Software available on the platform is presumed to be screened and certified by the platform owner, and therefore deemed safe. The platform might also offer reviews from other users that have purchased the software, giving the user a quick way of gauging the software's quality or usefulness.

2.5 Challenges and concerns

Romberg (2007) identifies four main forces acting on an established platform, that owners should be aware of. *The base technology*, *competing platforms*, *outside developers* and the *end-users*. The latter two of these factors correspond directly to actors presented earlier, with the *competing platforms* corresponding to actors associated with outside platforms.

There are certain challenges and interactions associated with these forces, as illustrated in **Fig. 2.2**.

The base technology is linked to assumptions made by the platform owner during specification and implementation of the platform. Romberg (2007) points to the progress of the base technology (both hardware and software) as a potential source of disruption, as it can invalidate these assumptions and force the entire technology stack to have to be recreated upwards layer by layer. They mention the earlier generations of video game consoles as examples of this, with radical changes in the base technology between generations. As mentioned earlier, this is also especially important in operating system-centric ecosystems.

Competing platforms can threaten the platform in two different ways. The first, and most obvious, is what Romberg (2007) refers to as *substitution*. This is the traditional and expected competition for users and external developers to grow the ecosystem of their own platform and increase their market share. The second threat that other platforms can pose, is through what is referred to here as *emulation*. This is where the owner of a, usually smaller, competing platform chooses not to engage in a standards battle, but instead emulates that of the larger platform in order to benefit from the already established reputation, technology or base of developers and users (Stanley and Farrell, 1994).

The users and developers, as integral parts of the ecosystem, affect the platform both directly and indirectly. They affect it directly through sheer numbers, as gaining or losing users and developers is inherently impactful to the platform. There is also an indirect effect on the platform through these actors' abilities to bring in more users/developers through network effects. Growing, or at least maintaining, the base of users and developers is therefore a very real challenge and important to keep in mind for platform owners (Bosch, 2009). There is also a very real risk of developers expanding their products to the point where they turn the collaborative relationship with the platform owner into a competitive one (Gawer, 2014).

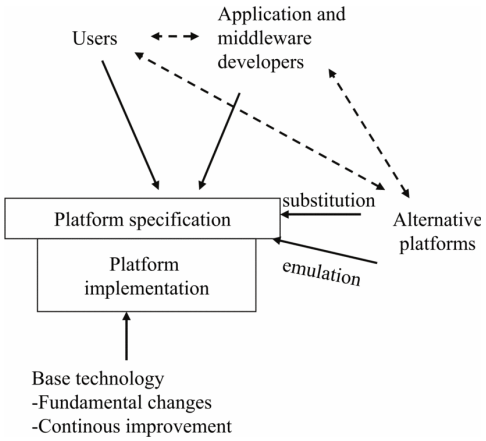


Figure 2.2: Four forces acting on an established platform (Romberg, 2007)

2.6 Distribution

While not strictly required, platform owners will often in some way facilitate transactions between third party developers and end-users (Tiwana, 2014). This often takes the form of a digital marketplace where users can browse and acquire apps. Well known examples include the mobile app marketplaces of Apple and Google; App Store and Google Play, respectively. These marketplaces can vary in functionality and prominence, however a study by Jansen and Bloemendal (2013) found that certain core features are present in the six largest marketplaces:

Core feature	Description
app categories	Apps are listed in categories and subcategories
app listing	Apps are listed with full description, images, etc
app lists	apps are listed, e.g. top selling lists or latest additions
dev app management	Devs can manage their apps in a developer console
dev transaction list	Devs can manage their transactions
distribution integration	Distribution and installation happens through platform
featured apps	Apps can be featured to receive more attention
free revenue model	Apps can be offered for free
paid revenue model	Apps can be sold
pay out methods	Number of pay out methods
payment methods	Number of payment methods
platform comp. filter	Apps have information on their platform compatibility
ratings	Apps can be rated by the user
reviews	Users can read and write reviews of an app
search	Users can search for apps using search keywords

Table 2.2: Core features of marketplaces.

2.7 Platform architecture

Architecture is the conceptual blueprint that describes the components of a complex system, what they do and how they interact. The architecture of a platform or an app can be seen as a high-level description of its building blocks and how they are related to each other, not a working implementation (Tiwana, 2014). Platform architectures are also unique in that they are not only modular, but also structured around a core and a periphery (Gawer, 2014).

The core, in terms of platform architecture, is the set of fairly stable components that make up the platform itself. The periphery is made up of the high-variance complementary components. Mediation between the core and the periphery is done through stable interfaces that divide all information required to build the system into visible and hidden

information. As the complexity of an element grows, this complexity can then be isolated by defining a separate abstraction that has a simple interface(Baldwin and Woodard, 2008).

This makes platforms particularly good at facilitating innovation. Their modularity helps manage complexity by breaking up a complex system into discrete components connected by standardized interfaces. This reduces the scope of information that designers require in order to design their modules and allows a specialization and division of innovative labour (Gawer, 2014). Both innovation within modules and mix-and-match innovation through innovative recombination of modules are facilitated by this (Gawer, 2014).

2.8 Platform governance

As with many aspects of platform ecosystems, the governance of such entities is a fairly unexplored field both academically and in the industry (Jansen and Cusumano, 2012). Governance is an important aspect to consider as it provides direction and coordination to the ecosystem. A platform owner with a good reputation and a strategy that offers stability in the ecosystem has an easier time attracting stakeholders to said ecosystem, and a lack of perceived stability could, conversely, drive them away (Berk et al., 2010).

This section covers some of the literature and proposed tools for academia and industry that have been published on the subject so far.

2.8.1 Governance dimensions

Tiwana (2014) divides platform governance into three dimensions: **Decision rights partitioning**, **control portfolio** and **pricing policies**.

Decision rights partitioning refers to who has the primary authority and responsibility for making certain decisions. Such rights can be centralized, primarily held by the platform owner, or decentralized, primarily resting with the developers. In practice, however, most end up being somewhere along the spectrum, not purely centralized or decentralized. Meaning both owners and developers have some authority and responsibility, but usually leaning more towards one party. Tiwana (2014) further divides rights into categories of strategic or implementation, as well as whether they refer to apps or the platform itself. A structure of rights therefore has four distinct sets of rights: Strategic and implementation rights for the platform, as well as strategic and implementation rights for apps. Note that a platform may use different structures for different apps. Likewise, a developer might have to deal with differing rights on different platforms for the same app.

Control portfolio design is the second dimension of platform governance. Control here refers to the means through which the platform owner ensures that the app developers' work is aligned with what is in the best interests of the platform. Tiwana (2014) identifies four tools, or control mechanisms, that owners use to implement and enforce rules that reward desirable behavior, punish bad behavior, and promulgate standards of behavior among app developers. The control portfolio refers to the combination of mechanisms used by the platform owner over a developer. As with decision rights, it is possible for different apps to be subject to different control mechanisms on the same platform.

The first of these mechanisms is **gatekeeping**, which represents the degree to which the platform owner uses predefined objective acceptance criteria for judging what apps and app developers are allowed into a platform's ecosystem. If entry barriers are set too low, the ecosystem is open to uncontrolled growth and a general loss in the quality of both third party developers and apps. Too high entry barriers will stifle innovation in the ecosystem and scare niche players away. Part of gatekeeping is finding the right balance between innovation and quality (Berk et al., 2010).

The second mechanism is **process control**, the degree to which a platform owner rewards or penalizes app developers based on the degree to which they follow prescribed development methods, rules and procedures that it believes will lead to outcomes desirable from a platform owner's perspective. Desirability here refers to how well apps interoperate with the platform, not how well they do on the market. Process control is sometimes realized through the developer tools provided by the platform owner.

The third control mechanism is the use of **metrics**. This control refers to the degree to which the platform owner rewards or penalizes app developers based on the degree to which the outcomes of their work achieve predefined target performance metrics. Tiwana (2014) notes that in many contemporary platforms, the market itself rewards and penalizes apps through sales, leaving little reason for the platform owner to involve themselves.

The fourth form of control is **relational control**. It is a less formal mechanism and refers to the degree to which the platform owner relies on norms and values that it shares with app developers to influence their behavior. It relies on the owner to provide an over-arching collective goal for the platform ecosystem. As an informal and inexpensive control mechanism, it is popular in open-source platforms.

Pricing is the third, and final, dimension of platform governance according to Tiwana (2014). The goal of pricing policies is to create incentives for app developers to make personal investments to ensure the prosperity of their own app offerings, which translates into an increased vibrancy of the ecosystem as a whole. While the pricing and payment model of the individual apps on the market often are decided by the developer themselves, there are still important pricing decisions that the platform owner has to make. One of these decisions is where they wish to profit. It is possible to profit from both sides of a two-sided platform, a symmetrical approach, however in some ecosystems one side is subsidized in the hopes that this has an effect on the other side that turns into a net positive for the platform owners. An asymmetric

approach. In certain cases, a symmetric approach which fails to generate a profit on either side is acceptable, due to alternate revenue streams that depend on the health of the ecosystem. According to Hyrynsalmi et al. (2012) this seems to be the case with the mobile platforms in particular. iOS using the ecosystem to promote sales of Apple's physical devices, Android to extend Google's advertising business into mobile devices and Windows to protect Microsoft's software sales in the ongoing mobile-desktop convergence. Actual revenue directly from app sales appear to be secondary.

2.8.2 Governance analysis framework

Baars and Jansen (2012) presents a framework for analysis of governance and governance structures in software ecosystems. They differentiate between these two concepts. In their paper, governance refers to the processes and procedures by which a company controls their position in the ecosystem, while the governance structure refers to the distribution of rights and responsibilities among all the stakeholders, as well as the rules and protocols that need to be followed in order to make decisions regarding the ecosystem.

The framework sets up a series of concepts surrounding governance and structures. These concepts, when analysed in light of an ecosystem, will yield a yes/no answer or not be applicable. In many cases such a simple answer will be insufficient, in which case the framework will be annotated with an explanation for the result of the analysis of that particular concept. See **Table 2.3** for the full framework. The first section covers processes, procedures and tools used to execute the governance strategy. The second section covers responsibility, control and measurement associated with the governance strategy. A further explanation of the categories of the second section is given:

Explicitness of the ecosystem: Questions that relate to the explicitness of the ecosystem in general. Without an explicit ecosystem, there of course cannot be an explicit governance strategy.

Explicitness of the governance strategy: Questions that relate to the explicitness of the governance strategy. Having an explicit governance strategy lets organizations refer to rules, procedures, protocols and formalized processes when dealing with an ecosystem. This leads to more control over the position in the ecosystem, which ultimately leads to more potential benefit gained from the ecosystem (Baars and Jansen, 2012).

Responsibility: It is important to have appointed members of the organization be responsible for the ecosystem in order to make sure that the governance strategy is is executed correctly. The framework's authors note that ecosystem governance can easily become just a "job on the side", and get neglected in favour of other tasks.

Measurement: The effectiveness of the ecosystem should be measured in some way, in order to determine the organization's benefit from it. Key performance indicators

(KPI) can be applied to various aspects of the ecosystem in order to achieve this. Analysing the current state of the ecosystem and prospecting the future state can lead to higher return on investment (Baars and Jansen, 2012).

Knowledge sharing: Knowledge sharing is not necessarily a vital aspect of a successful governance strategy. The authors of the framework note that it might even be detrimental in some cases for for-profit corporations to actively share knowledge, while for not-for-profit organizations it might be a core aspect.

The purpose of the framework is to allow organizations to be able to get an overview of the state of their current ecosystem governance strategy, as well as allow researchers to better compare different companies with each other, leading to a better understanding of practice and more theoretical completeness. The authors of the framework acknowledge that it is too early to consider it a basis for all research related to ecosystem governance. At the time of their paper on the framework, it had only been applied to two cases. In addition it lacks any expert reviews in order to validate it. The author's point out that another limitation of the framework is that it does not take into account the importance of individual factors, such as knowledge sharing, but merely the existence of such factors.

Category	Governance concept	Result
Partnerships	Creating a partnership network Degree of moderation Degree of division in tiers, levels, etc Acquiring new partners Formalization of entry requirements	
Supplier and customer governance	Coordination of contribution to other ecosystems Setting up new suppliers Changing the ratio of current suppliers Ceasing cooperation with suppliers or customers Using intermediaries	
Development	Creating a development standard Enforcing a development standard	
Partner directory	Creating a partner directory Degree of moderation	
Customer directory	Creating a customer directory Degree of moderation	
User groups	Creating active user groups Degree of moderation	
Licence(s)	Creating reusable software license(s)	
Category	**Governance structure concept**	**Result**
Ecosystem explicitness	Is the ecosystem explicit? Is there documentation describing its current state?	
Governance explicitness	Is the ecosystem governance strategy explicit? Are processes and procedures formalized? Are there formalized and documented rules? How is business strategy formalized to governance strategy?	
Responsibility	Where in the organization does ecosystem governance take place? Who does the decision making unit consist of? Is this decision making unit made explicit? Does the decision making unit report to the Board?	
Measurement	Is the effectiveness of the ecosystem measured? Which parts of it are measured? Which KPIs are used? How are goals defined?	
Knowledge sharing	Does the organization share its knowledge with other companies?	

Table 2.3: Governance analysis framework

2.8.3 Governance tools

Jansen and Cusumano (2012) presents a set of governance tools for the preservation and improvement of ecosystem health. The health of an ecosystem is represented here in three aspects: **robustness, niche creation, and productivity**. Tools are provided for four different types of ecosystems, corresponding to the classification model presented in **subsection 2.3.2**. Based on combinations of the factors *base technology* and *coordinator* these types are: community run software platform, commercially run software platform, community run standard and commercially run standard. For the sake of brevity, only the tools for commercially run software platforms are focused on here, as they will be the most relevant ones later on.

	Software (service) platform		Standard	
	Community	Private Entity	Community	Private Entity
Niche creation	Expand applicability Make strategy explicit Create APIs Do co-development Contrib to comp. platforms	Expand applicability Make strategy explicit Create APIs Do co-development Dev. complementary platforms Develop new business models	Expand applicability Make strategy explicit Form subgroups	Expand applicability Make strategy explicit Form subgroups
Robustness	Form consortium Grow consortium Create subgroups Raise entry barriers Form alliances Stabilize APIs Make consortium explicit Open up governance	Create partnership model Do marketing Grow profits Partner development programs Form alliances Stabilize APIs Raise entry barriers Make partners explicit Propagate operation knowledge	Form consortium Grow consortium Raise memberships Form alliances Make consortium explicit Open up governance Start certification program	Protect the standard legally Do marketing Raise memberships Evolve platform Make partners explicit Start certification program
Productivity	Organize dev days Create knowledge hubs Participate in contests	Organize dev days Collaborative marketing Create sales partner program Create new sales channels	Create showcases Create knowledge hubs	Create showcases Collaborative marketing Create new sales channels

Figure 2.3: Governance model for ecosystem health preservation and improvement (Jansen and Cusumano, 2012)

A privately owned (commercially run) software platform aims to maximize value by penetrating the market as deeply as possible. This is done mostly by associating with domain specific parties that leverage the platform within the ecosystem to create value for customers that would have never been reached without these domain experts (niche players) (Jansen and Cusumano, 2012). As is the case with other types of platforms, its coordinators want to continuously increase the usage of the platform, however they are also focused on maximizing profits as well.

Niche creation: Coordinators have to make sure that sufficient niche creation is happening for other parties to want to join in. Jansen and Cusumano (2012) present several

steps that can be taken here. Creation of APIs for third parties, extending the applicability of the platform (for instance by venturing into new domains), making the strategy of the platform explicit. Making the strategy explicit involves making explicit elements such as the product life cycle, acquisition strategy and ecosystem strategy, allows the niche players to rest assured that their position in the ecosystem remains safe.
Coordinators can also take more direct action by co-developing and co-funding projects with third parties to attract them to the ecosystem. Another direct action is to contribute to complimentary platforms, in the hope that the complimentary platform's growth also benefits the coordinator's platform.
The introduction of new business models for third parties can also create niches and business opportunities in the ecosystem. Extension markets, such as those discussed earlier, are examples of this.

Robustness: In order to increase robustness, coordinators should focus on creating stability and stimulating activity in the ecosystem. One of the first steps to doing this is the development of a partnership model that enables third parties to participate and create value in the ecosystem according to set roles and positions in the ecosystem. Stability of the ecosystem is also increased by doing marketing and growing profits for stakeholders.
Partners with potential or weak partners can also play an impactful role in stabilising or destabilising the ecosystem. Because of this it is suggested that a partner development program be introduced that can strengthen weak participants and bring high potentials closer to the ecosystem.
Robustness of the ecosystem can also be increased by raising entry barriers. This can be done by raising membership fees, raising quality requirements, introducing certification programs, and assigning different levels within the partnership programs. These raised entry barriers will contribute to the growth of a stable core of committed members within the ecosystem.
Finally, stabilizing the APIs creates a consistency within the ecosystem and enables partners to create trustworthy and stable extensions to the platform.

Productivity: Productivity can be fostered in a variety of ways. Coordinators can arrange development days that help raise awareness and activity surrounding the platform. Ververs et al. (2011) found that such events have a clear influence on developer participation in open source communities. Coordinators can also engage in collaborative marketing and sales with third parties, to emphasize that the third party has a respectable relationship with them. Finally, new sales channels can be created to enable more revenue for third parties, presumably leading to increased value creation in the ecosystem.

As is the case with many of the other tools, frameworks and models currently existing in ecosystem literature, the model presented here is in too early a stage to be viewed as complete. Jansen and Cusumano (2012) acknowledge that further evaluation of it is needed.

2.8.4 Decision making

Manikas et al. (2014) presents a framework for defining the decision making strategies in software ecosystem governance. They decompose governance into three main activities: data collection, decision making, and applying actions. These actions form a cycle of constantly gathering data, processing and interpreting the data, and taking action based on the results, followed by again gathering data. The proposed framework, as mentioned, deals primarily with the second step of this cycle, the decision making.

The framework is in part based on six politically inspired archetypes in IT governance outlined by Weill and Ross (2004). Manikas et al. (2014), however, acknowledge that there are some clear differences between IT governance and software ecosystem governance. IT governance has a separation between IT and the product, whereas in a software ecosystem the software is the product itself. There is also a difference in that ecosystems focus on the alignment between independent entities, not entities directly governed by the same organization. Consequently, the framework here differs from the original archetypes, having four instead of six.

The proposed ecosystem governance decision framework consists of five main decision areas that group the main governance decisions of the ecosystem and the four archetypes that describe how decisions are taken for each area. The main decision areas are the following:

Principles: Decisions that address core principles, general values and main directions of the ecosystem. Usually decisions that have a major influence on the entirety of the ecosystem.

Actor Interaction: Decisions that are related to and affect the actors in the ecosystem. Decisions such as total number of actors allowed in the ecosystem, how/if new actors are allowed access, or how actors are allowed to interact with each other.

Software Interaction: Decisions regarding the software interaction and structure of the software component network of the ecosystem, including software release management, software architecture of the ecosystem, and other procedures and rules affecting the software build and distribution.

Platform: Decisions regarding the technological platform and other common technical infrastructure. Due to the platform being such an important and central part of the ecosystem, this decision area is considered distinct from the previous software interaction area. Decisions in this area include management of the platform, platform architecture, commercialization of the platform and the platform openness. The openness here refers to the extent to which different actors are allowed to be involved with the core development of the actual platform itself, not the ecosystem.

Ecosystem business and products: Decisions concerning the business models of the ecosystem, motivation of the actors and distribution/availability of products, such as strategies for the ecosystem's extension market(s) and incentives for actors.

For each such decision area, the framework categorises the way decisions are made, based on the archetypes:

Monarchy: A single actor making decisions for a specific decision area. An example is Apple deciding the principles, as they are the main hardware supplier and orchestrator of the iOS ecosystem.

Collective: Decisions are made through processes involving all the actors, for instance by voting. An example is the Django framework, where developers are asked to vote for which new features should be implemented.

Federal: A number of actors are assigned as representatives to make decisions. An example is the Apache ecosystem where changes in the Apache server repository can only be done by a specific group of actors. Also, in the ODA ecosystem, the founding members can decide to change the way new actors enter the ecosystem.

Anarchy: Each actor makes decisions on their own. An example is the software ecosystem surrounding the game World of Warcraft, where any actor create their own extension market, or in the Ruby on Rails ecosystem where anyone can make commits to the platform.

	Principles	**Actor interaction**	**Software interaction**	**Platform**	**Business and products**
Monarchy	Apple iOS				
Collective			Django		
Federal		ODA		Apache	
Anarchy			Ruby on rails		World of Warcraft

Table 2.4: Decision making framework

An example of the framework being applied to the examples above can be seen in **table 9.1**. It is worth noting that the decision areas in this framework are fairly general and abstract. Some ecosystems might have multiple applicable archetypes for a given decision area. Decision areas might need to be further divided into subcategories or have their classification of archetypes explained in greater detail.

Chapter 3

Customer involvement

The creation of new products has traditionally been viewed as an internal firm-based process, where customers are seen as passive buyers and users. Recently however, businesses are experimenting with a higher degree of customer involvement, letting customers take on a more active co-creator role (O'Hern and Rindfleisch, 2010). In this chapter, we will briefly cover some of the academic literature on customer involvement, more specifically the process of customer co-creation.

3.1 Customer co-creation typology by O'Hern and Rindfleisch (2010)

O'Hern and Rindfleisch (2010) outline a typology with four types of co-creation and their uses, briefly described here:

Collaborating: In this typology, collaborating is defined as a process in which customers have the power to collectively develop and improve a new product's core components and underlying structure, i.e. source code. Examples of such initiatives are open source software projects, such as Linux, Apache and Firefox. Collaborating offers customers a high degree of latitude to contribute their own improvements, as well as take part in deciding the features of a product. It does, however, also require a less controlled environment, which might turn away IP holders from opening up.

Tinkering: Tinkering is defined as a process in which customers make modifications to a commercially available product and some of these modifications are incorporated into subsequent product releases. It resembles collaborating, however much more control is retained by the firm owning the IP. Tinkering often involves the firm distributing tools to allow customers to make their desired modifications. Examples mentioned by O'Hern and Rindfleisch (2010) are modifications (mods) for computer games and individually tailored web-applications. A possible challenge for

firms that allow for tinkering is to ensure that their newest product iterations actually surpass the functionality offered by a customer modified version of an older iteration.

Co-designing: Co-designing is the process in which a relatively small group of customers provides a firm with most of its new product content or designs, while a larger group of customers helps select which content or designs should be adopted by the firm. It is characterized by a relatively fixed contribution approach, but a high degree of customer autonomy over the selection of these contributions. Co-designing offers an advantage for the firm in that it drastically decreases development costs for new designs and creative content, as well as providing pre-launch evaluation of a product by the customer base. Some challenges to the co-designing approach are actually attracting enough designers to provide sufficient content, as well as maintaining an internal competency and distinctiveness that competitors cannot easily imitate.

Submitting: Submitting is defined here as a process in which customers directly communicate ideas for new product offerings to firm. Submitting is differentiated from traditional forms of customer inquiry(focus groups, surveys, tracking studies) by both the degree of customer effort required and by the nature of the input that customers provide to the firm. It required customers to expend considerable energy developing detailed and tangible ideas for new product offerings, providing well-defined processes, detailed graphic depictions or working prototypes. Firms that make use of this process often actively solicit input from customers, for instance through contests. Submitting resembles co-designing, however differs in that the firm retains full control in the selection. Some advantages of submitting are a reduction in development time, as well as an increase in customer relationships. Compared to the other forms of co-creation, submitting has the lowest level of customer empowerment, which might make it a less attractive option and firms might have difficulty maintaining or increasing participation.

Type of Co-Creation	**Selection Activity**	**Contrib. Activity**	**Key Payoffs**	**Key Challenges**	**Prototypical Application**
Collaborating	Customer-Led	Open	Reduced development costs Continuous product improvement	Protecting intellectual property Attracting a critical mass of collaborators	Open source software
Tinkering	Firm-Led	Open	Enhanced differentiation Virtual test markets for new products	Policing the content of rogue co-creators Creating new competitors	Modified computer games
Co-designing	Customer-Led	Fixed	Reduced development costs Decreased risk of product failure	Attracting a critical mass of designers Defending against new entrants	Online voting on customer-generated content and designs
Submitting	Firm-Led	Fixed	Shortened product development cycles Increased access to novel customer ideas	Acquiring knowledgeable new co-creators Retaining and motivating existing co-creators	Company-sponsored design competitions

Table 3.1: Characteristics of co-creation types, by O'Hern and Rindfleisch (2010)

3.2 Customer Co-creation typology by Piller et al. (2010)

Another typology for co-creation in the innovation process, presented in Piller et al. (2010), identifies three characteristics that form its conceptual dimensions: **The stage in the innovation process**, which refers to when in the development process the customer is involved.

Stage in the innovation process: This refers to when in the development process the customer is involved. Either early in the front end stages of the process (idea generation, concept development) or later in the back-end (product design and testing).

Degree of collaboration: This refers to the underlying relationships between actors. Whether there is collaboration between the firm and one customer at a time or whether there exist networks of customers who collaborate among themselves more or less independent from the firm.

Degrees of freedom: This refers to the broadness of the task assigned to customers. Whether it is a narrow and predefined task or whether it as open and creative task many degrees of freedom.

Using the available combinations of these three dimensions, they further identify eight types of co-creation, visualised in **fig 3.1** and **fig 3.2**:

Idea contest: In an idea contest a firm can post a request to a population of independent agents, such as customers, to submit solutions to a given task within a given time frame. The request can be for a solution that conforms to a specified set of requirements, or be more or less an open call for solutions to a vaguely specified problem. Rewards for providing the best solutions can range from simple monetary prizes or licensing contracts, to non-monetary acknowledgements for marketing purposes. Idea contests aim to incentivize customer involvement early in the innovation process, as customers might not intrinsically see the benefit of their contributions in a short time frame until later stages.

Idea screening and evaluation: Screening and evaluation of ideas is the natural next step after several such ideas have been generated, for instance through an idea contest. Selecting the ideas with the highest potential is a task that can be carried out by a panel of experts from the firm, or by the customers themselves. Depending on their number and their complexity, it might be unreasonable to expect a single customer to evaluate more than a few ideas. Some sort of system for distributing the up-and-coming ideas for evaluation should be implemented.

Product related discussion forums: In these kinds of communities customers primarily exchange usage experiences and support each other in the usage of a product. Innovation and creative activity can sometimes occur, but is not the primary focus.

Communities of creation: These communities differ from the previous discussion forums in that they are primarily concerned with generating novel ideas and concepts. Their innovation productivity is high, and can provide output beyond simple descriptions, such as code and technical drawings.

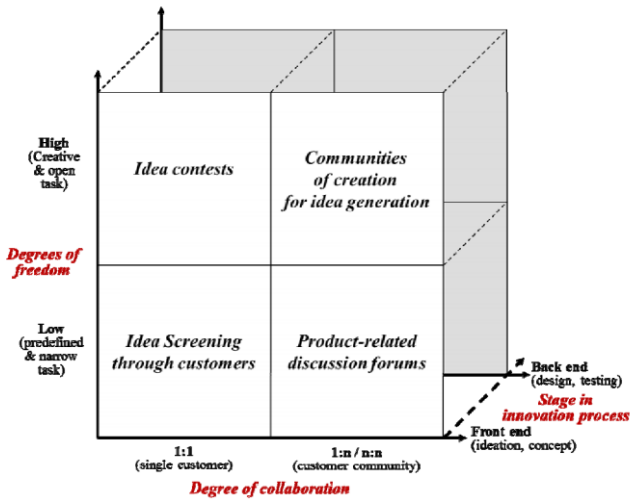


Figure 3.1: Typology at front end of co-creation (Piller et al., 2010)

Toolkits for user innovation: These kinds of toolkits provide customers with a development environment that allows for them to create an improved product for themselves from the manufacturer's standard modules and components, but also lets them experiment with creating entirely new solutions.

Toolkits for customer co-design: These kinds of toolkits are geared more towards product customization and developing variants of existing versions, rather than developing new goods and services. They have a more limited solution space, and modifications are restricted according to the pre-defined "building blocks".

Communities of creation for problem solving: These are an extension of the previously mentioned communities of creation. In this case, however, the scope stretches all the way to the back end of the innovation process where a product reaches its final stages. Open source software projects are the typical examples of such communities, with products being developed through a collaborative effort by the customers/users.

Virtual concept testing: Similar to the screening and evaluation of ideas in the front-end, making use of virtual concept testing at the back-end can allow for customer feedback on more tangible representations of products. This can be in the form of allowing customers access to virtual presentations and test simulations of future products, or in the form of bug fixing activities in open source projects.

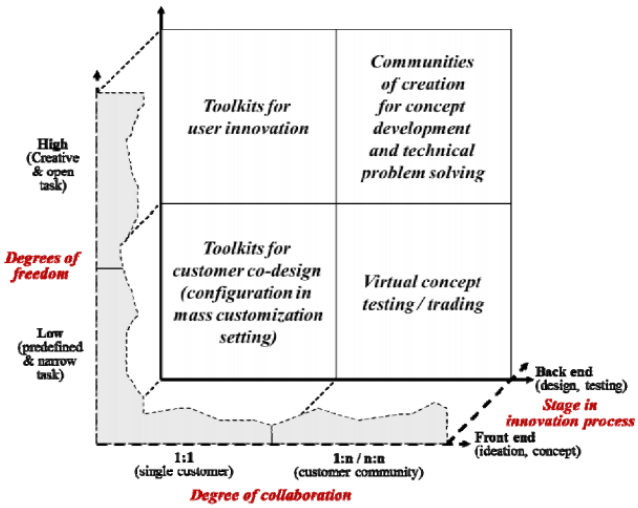


Figure 3.2: Typology at back end of co-creation (Piller et al., 2010)

Part III

Method

Chapter 4

Approach

Over the summer of 2014 it was determined, after conversations with the thesis supervisor, that the theme of the thesis would be software ecosystems. Initial contact was made with Trond Benum at Schlumberger in Trondheim in August 2014, requesting sponsorship for a case study of the Ocean ecosystem. An agreement was made the following September, with Schlumberger agreeing to sponsor visits to their Stavanger offices as well as provide information on request. At this point, relevant academic literature was collected, largely by using **Google Scholar** to find articles, but also physical books. An **exploratory** (Oates, 2005) approach was taken in regards to the literature, as well as the first visit to Schlumberger in Stavanger in December 2014. This allowed for a more defined direction for the thesis to be ready for the start of 2015, after discussing the details of a problem they presented, as seen in part IV of this thesis.

The approach then shifted towards a more **descriptive, short-term, contemporary study** (Oates, 2005), focusing on the case in its current state. Most external interviews were performed in Q1 and Q2 of 2015, interspersed with retrieving literature, as well as staying in contact with Schlumberger primarily through Ahmed Aqrawi at the Stavanger offices. Finally, the literature was applied to the data gathered on the Ocean ecosystem, and the results applied to the case itself.

Chapter 5

Data collection

Data collection for this thesis was primarily done in two ways; interviews and observation. Before any data collection took place, a preliminary outline of the project was sent to the Norwegian Social Science Data Services(NSD), the data protection official, for evaluation. After clearance from NSD, the initial data collection was performed during a week-long visit to Schlumberger's Stavanger offices in early December of 2014. This visit allowed for both semi-structured interviews as well as overt observation of day-to-day operation. In addition, some data was collected through more informal means such as "watercooler talk" and chats during lunch breaks. This visit also established contacts with key personnel at Schlumberger. Finally, a direction for the thesis was determined during this visit based on the interactions had with these individuals.

Contacts at Schlumberger opened up further contact with third parties involved with the ecosystem. These parties were then contacted by e-mail with a request for an interview. As many of them were located outside of Norway, these interviews were primarily performed using VOIP through either Skype calls or Lync meetings. Face-to-face interviews and further observations were performed when possible. Interviews were primarily conducted with third party heads of development and developers, but also with internal developers at Schlumberger. In addition to developers, users of the Petrel platform were also interviewed.

A second visit to the Stavanger offices was made in March 2015, for the purposes of follow-up interviews as well as new, full, interviews.

Interview subjects were chosen from four main categories:

- **Internal developers** at Schlumberger that create plug-ins for commercialization in the Ocean Store (two participants).

- **Third party developers** that create plug-ins for commercialization in the Ocean Store (three participants).
- **Third party developers at E&P companies** that specifically create plug-ins for internal use within their own company (One participant, managerial position).
- **Petrel users** who make use of Petrel plug-ins created with Ocean (One participant, managerial position responsible for Petrel deployment and use).

In addition to these subjects, various staff and managers at Schlumberger were also interviewed during visits in Stavanger (five participants).

Interviews generally lasted for 40 minutes and were transcribed by the interviewer. In addition, four brief follow-up interviews were also performed for the purpose of clarification in some cases.

Part IV

Case

Chapter 6

Background

6.1 What is the E&P sector?

The oil and gas industry is often viewed as a chain involving three major sectors. Upstream, midstream and downstream. E&P is short for Exploration and Production, which is another name for the upstream sector. This sector, as the name implies, is where oil and natural gas is found and extracted. The midstream sector refers to the industry where these commodities are stored and transported after production. Finally, the downstream sector refers to the refineries, petrochemical plants, distributors and retail outlets that terminate the chain. (PSAC, 2015).

An important part of the E&P sector, or upstream sector, is locating new potential sources of oil and natural gas. This process begins with observations by geophysicists and geologists to find areas with a reasonable chance of containing petroleum reservoirs. Once such an area has been found, seismic exploration is performed there to further determine if it is suitable for drilling. Seismic exploration involves using sound waves that travel through the subsurface rocks, is reflected off the subsurface rock layers, and returns to the surface to be recorded. On land, the waves are typically created by use of explosives or mechanical vibrations, while at sea it is often an array of air guns trailing after a ship with sensors (Hyne, 2001). In addition to seismic exploration there is also gravity and magnetic exploration, however seismic is the most common.

The geophysicists and geologists then analyse the data to find evidence of reservoirs or similarities to other areas where petroleum production has already started. If the results are promising, an initial drill site is determined and drilling starts. Throughout the drilling process continuous analysis is performed of the drill's log in addition to rock samples recovered from the well, in order to further determine the profitability of the reservoir. New wells are then drilled at other potential reservoirs in the area, or sometimes multiple wells to the same reservoir, to increase the production of petroleum. Wells deemed to no

longer be profitable, or never profitable in the first place after sample analysis, are shut down and the borehole filled.

6.2 Digital tools in E&P

Until the prominence of digital technology, the representation of data in the E&P sector was mostly done by hand. The transition to digital tools in the 1980s coincides with both the need for 3D representations of data and the capabilities of the new technology to actually provide them (Laver, 2012). A transitional mechanical approach to 3D was attempted in the 80's with the SeisCrop Table, transcribing seismic data to film strips, however it was quickly overshadowed by the introduction of commonplace computers in the workplace (Roth, 2005). In addition to 3D modelling capabilities and interpretation, the functionality required by E&P software soon expanded to include data sharing capabilities between dissimilar tools to better allow geophysicists and geologists to collaborate. As computing power increased, the possibility of running simulations on reservoirs was introduced as well. Today, visualization, modelling and simulation software are in heavy use by geophysicists, geologists and petroleum engineers all over the world to help with optimizing both exploration and production. Recently, a move towards extendible software platforms has taken hold, as opposed to stand-alone products. Figures **6.1** and **6.2** are taken from Welling reports on the subject.

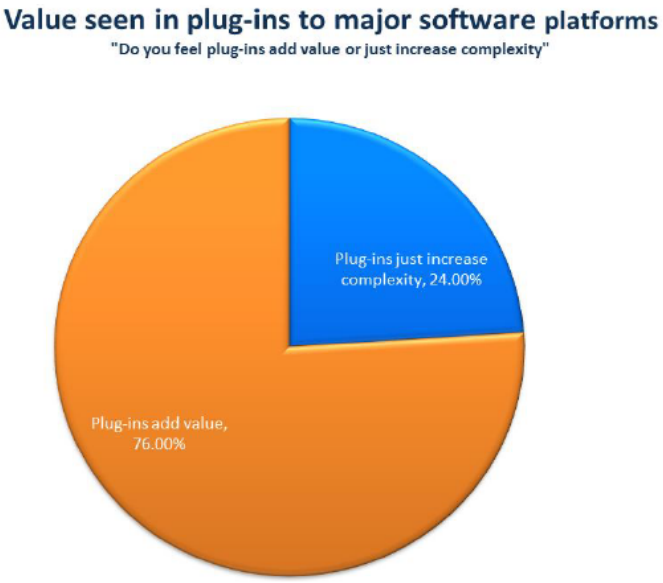


Figure 6.1: Plug-in survey. 2013 Welling Report Geology and Geophysics.

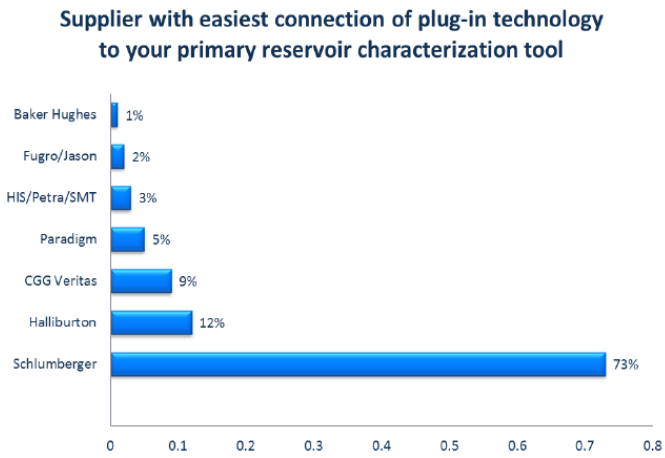


Figure 6.2: Supplier survey. 2012 Welling Report Reservoir Characterization.

6.3 Schlumberger

Schlumberger is the leading supplier of technology, integrated project management and information solutions to the oil and gas industry worldwide. It employs approximately 126,000 people (Schlumberger, 2014a) and had a full-year revenue of $45.27 billion in 2013 (Schlumberger, 2014b).

The precursor to Schlumberger, Societe de Prospection Electrique, or Pros, was established in 1926 by brothers Conrad and Marcel Schlumberger (Schlumberger, 2014d). Pros initially carried out surface prospecting for ores, but gradually extended its activities to exploration of possible oil-bearing structures. They soon pioneered the use of a new technique of using probes lowered into boreholes to measure resistivity. These measurements allowed for improved well logging, where previously samples from the subsurface rock would be required. By 1929 demand for the patented ”electrical coring” was growing rapidly in many countries across the globe.

Further technological innovation and business expansion resulted in the now multinational company Schlumberger Limited being listed on the NYSE in 1962. The following years would see the company furthering research in oil exploration as well as providing sensing and measuring equipment both to deep sea exploration vessels as well as NASA spacecraft during the space race.

Today, Schlumberger is also heavily involved in software for the oil and gas industry through Schlumberger Information Solutions(SIS). Their main software products are the **Petrel E&P Software Platform**, the **Studio E&P Knowledge Environment**, the **Techlog Wellbore Software Platform**, the **Avocet Production Operations Software Platform** and the **Ocean Software Development Framework**, which allows for the development of plug-ins for the platforms. Ocean currently supports development for both the Petrel

platform as well as Studio, with support for Techlog expected in 2015 and Avocet at a later date (Schlumberger, 2014g). The Ocean framework is central to the ecosystem discussed in this thesis and is presented in greater detail below. Petrel and Studio will also briefly be presented, as they are currently extendible by the Ocean framework. Techlog and Avocet do not currently have support in Ocean and so are considered outside the scope of this work.

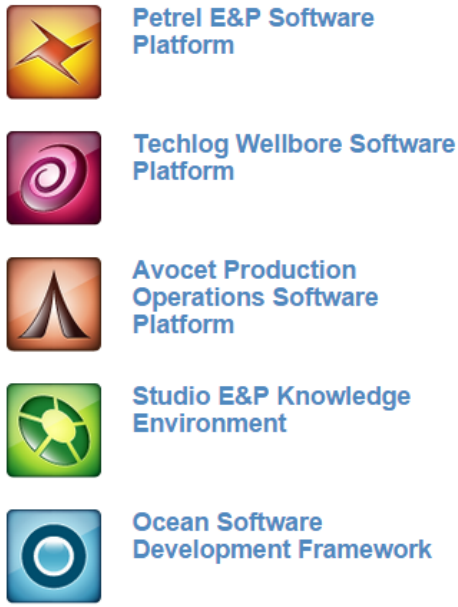


Figure 6.3: Schlumberger platform technologies.

6.3.1 Schlumberger Information Solutions

6.4 Petrel

The Petrel E&P Software Platform is the SIS flagship product. It supports the rapid development and updating of three-dimensional subsurface models, with workflows spanning seismic interpretation through reservoir simulation (Schlumberger and Microsoft, 2014). By offering a wide range of features and perspectives, it facilitates cross-discipline collaboration without the need for disparate applications (Schlumberger, 2014e). The platform also supports automated, repeatable workflows, allowing best practices to be shared across an organization. Additional features in the form of plug-ins can be obtained from the Ocean Store, given that the required so-called ”core” for that module is installed. A Petrel core is a package of additional features intended for a particular domain. The four cores in Petrel are the Combined Core, Geoscience Core, Reservoir Engineering Core and Shale Core.

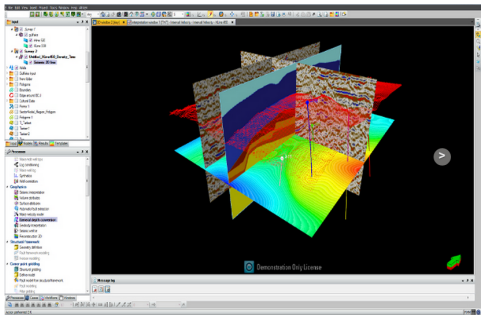


Figure 6.4: Screenshot of Petrel.

6.5 Studio

The Studio E&P Knowledge Environment enhances the Petrel platform by allowing effective capturing and sharing of knowledge. Long term it is planned for all four main products to be integrated through Studio (Schlumberger and Microsoft, 2014).

Studio offers a multi-user, concurrent-access project environment that provides scalability over large sets of data and users (Schlumberger, 2014f). Collaboration is facilitated through contextualized representation of the data for each user, integrated communication-channels with other members of the team, and data management tools for the Studio environment. It's primary purpose is to allow for an easy and accessible way to get an overview of many and large sets of data.

6.6 Ocean

6.6.1 The Ocean Framework

Ocean is an application development framework for developing plug-ins for the SIS software platforms, though currently only Petrel and Studio are supported. Plug-ins are built with Visual Studio for .NET with the help of a platform-specific wizard that takes care of the interaction with the Ocean core as well as some low-level access to functionality provided by the product family.(Schlumberger, 2014c). A plug-in consists of modules, extensions to the product family, that are packaged together along with meta-data to complete and identify the plug-in.

Fig.6.5 shows the architecture of the framework as it is presented in the documentation for the *Ocean for Petrel* API. The Ocean Core serves as the basic infrastructure, managing Ocean modules and registering services, as well as managing data sources provided by the product family or external data sources defined by a module. The Ocean Services are a set of application independent utilities; modules that benefit from being standardized across product families. The product family is the host for Ocean applications and is the environment in which the Ocean module needs to run, for instance the Petrel platform.

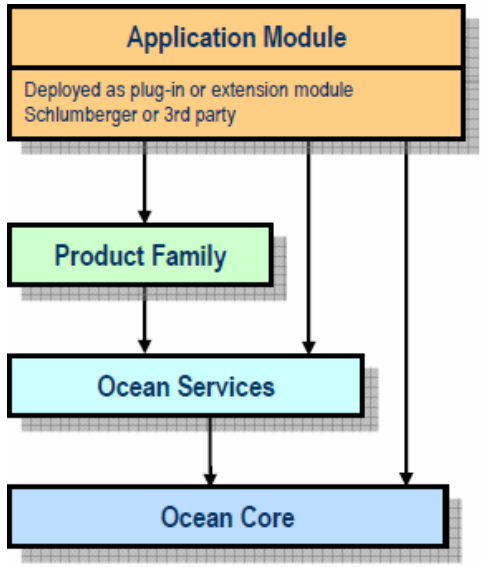


Figure 6.5: Ocean architecture.

6.6.2 The Ocean Store

The Ocean Store enables platform users to browse a catalogue of Ocean plug-ins for the SIS platforms (Schlumberger, 2014g). Most of these plug-ins are developed by third parties using the Ocean framework and offered as licences through the store, though some are developed in-house by SIS. There are currently 138 plug-ins available for Petrel and one for Studio (Schlumberger, 2014c). Compensation for the developers varies, as shown in **Fig.6.6**. Note that only partners and certified partners can distribute commercial plug-ins through the Ocean Store. Certified partners are partners with a proven track record of developing quality plug-ins and are displayed as such in the Ocean Store, along with various other benefits. Some third party developers, remain unconvinced by the partnership program and the Ocean Store however. One developer commented that using their own sales channels were a better option than using the Ocean Store for distribution, and that being a preferred partner seemed to be effectively little more than a marketing gimmick. *"SIS seems more interested in selling copies of Petrel than truly developing a platform."*

6.7 The Ocean Ecosystem

While ultimately all of the SIS platforms might be considered part of the ecosystem, the central unifying piece is the Ocean framework which allows for extensibility in the other

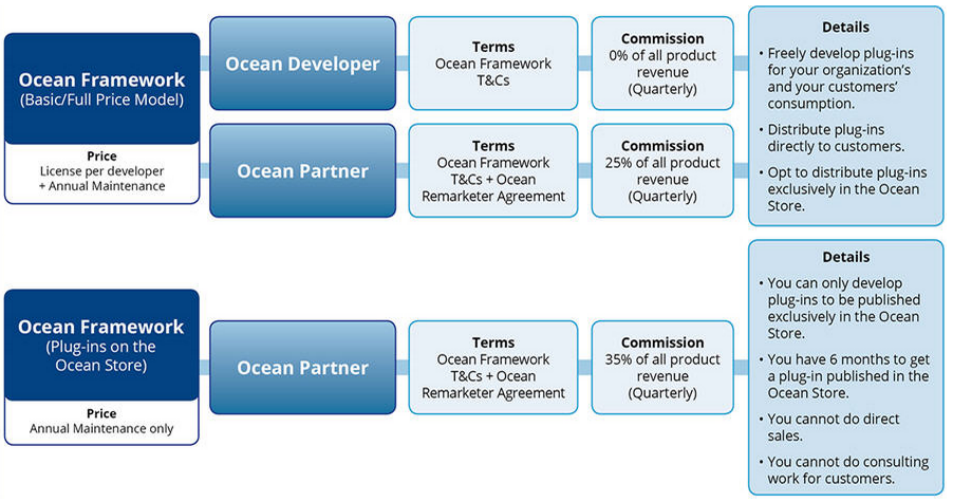


Figure 6.6: Ocean framework payment models.

platforms. For this reason, the ecosystem will be referred to as the Ocean ecosystem from here on in. It is an application-centric software ecosystem, as described in the taxonomy by Bosch (2009), born out of a general desire for extensibility.

While in the process of renewing their older flagship product, GeoFrame, the idea of extensibility and openness inspired work on Ocean to start in 2001. Shortly after, in 2003, the Petrel platform was acquired when Schlumberger bought Technoguide. Based on the existing reputation of Petrel, it was decided that it and Ocean combined would gradually replace GeoFrame. Work on the new GeoFrame product, called iGeoFrame, ceased and development resources were moved to the new products. Since then the Ocean framework has been closely tied to the Petrel platform and vice versa, despite some initial concerns about giving up potential revenue streams by allowing functionality to be commercialized by external parties. Interviews with Schlumberger staff reveal that there was also some contention between those who wanted Ocean to remain independent of Petrel and those who wanted the integrated solution that is in use today. An interviewed representative from a large E&P company with several years of Petrel and Ocean experience regarded this to be an important move, citing that Ocean was *"really lacking a killer-app up until that point"*.

The Ocean ecosystem is currently focused around Ocean and the extensibility it offers to the Petrel platform, as well as the Studio platform to a lesser extent. Long term, however, the plan is for the ecosystem to encompass all five of the SIS platform technologies. This means adding Ocean support for Techlog, which is focused on the detailed well level, and Avocet, which is focused on the business level. The intention is for the Ocean ecosystem to be host to a holistic platform that integrates all the functionality needed by E&P companies, with easy interaction between its sub-platforms as visualized in **Fig.6.7**.

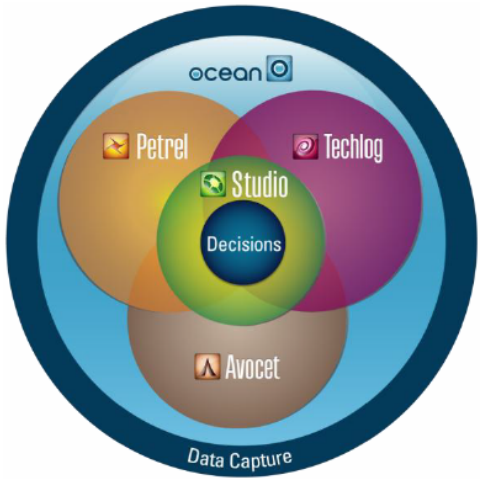


Figure 6.7: Schlumberger's vision for an integrated E&P plaform.

6.7.1 The actors in the Ocean ecosystem

The owner of the ecosystems and its platforms is Schlumberger, through its SIS segment.

The developers in the ecosystem consist of a mixture of traditional software development companies, developers specializing in creating plug-ins for the SIS platforms, in-house development teams from E&P companies as well as SIS themselves. In addition to this is the development of plug-ins that are never released on the Ocean Store, but created for usage within certain companies or in an academic context. There are however instances of the latter being commercialized later with assistance from SIS.

The end-users of the ecosystem are E&P companies of varying sizes and prominence that make use of the platforms, primarily Petrel.

6.7.2 Governance

SIS has purposely opted for a low involvement approach to the ecosystem, claiming openness and a free marketplace to be important. An analysis of the governance of the Ocean ecosystem using the dimensions outlined in Tiwana (2014) gives us the following:

Decision rights in the ecosystem are left fairly decentralized. Certainly for plug-ins. Developer's develop the functionality that they believe will generate a profit, with little involvement or direction from SIS. In the platform category there are a number of efforts made towards making the ecosystem more open. Input from users is encouraged on ideas for improvements to both Petrel and Ocean, as well as maintaining an active dialogue between themselves and SIS. Annual week-long events for users and partners are held to further discuss the long-term plans for Petrel and Ocean, as well as allow for voting on

changes or new features to prioritize. The actual prioritization and impact of requests are not known publicly, however.

The control portfolio of the Ocean ecosystem primarily consists of strict gatekeeping of plug-ins. All plug-ins must satisfy certain criteria in regard to technical quality, marketing and documentation. Only plug-ins that satisfy this acceptance test laid out by SIS are approved to be available on the Ocean Store. There is also a certain amount of process control, although it is more a case of encouragement than enforcement: Online instructional resources provide some best practices, as well as the Ocean user group involving developers with presentations and workshops. Long term adherence to best practices and creating high quality plug-ins opens up the possibility of a developer getting status as a preferred partner. Preferred partners get improved visibility in the Ocean Store, as well as other benefits such as information on upcoming releases of the platforms in the ecosystem. In addition SIS frequently recommends their preferred partners to E&P companies looking for a developer to develop a specific solution. This creates incentives for developers to follow best practices, without anything being directly enforced.

The pricing dimension of the Ocean ecosystem follows an asymmetrical approach. While the profits generated from the sale of plug-ins on the Ocean Store are not inconsequential to SIS, the main source of revenue from the ecosystem comes from the sale of platform licenses. As such, the main purpose of the Ocean ecosystem becomes to add value to the existing platforms. Developers are subsidized in part through the Ocean framework and its APIs, online resources, as well as the user group and other community events. Pricing models for plug-ins on the Ocean Store are restricted to licensing, though actual prices can be set by developers. An Ocean representative noted, however, that they see it as preferable that prices stay at a reasonable level as it *"sets a standard for what plug-in technology is worth and, consequently, reflects on the value of Ocean."*.

Chapter 7

Case presentation

7.1 Domain objects

7.1.1 Description

The term ”domain object” is used in the context of Ocean and its associated platforms to refer to a class of data. This domain object is often a representation of a physical entity, such as a geological horizon, surface, reservoir and so on. These objects have properties which in turn are also technically domain objects, however cannot exist on their own as they have to be linked to a parent object. Properties such as height fields along a surface or porosity distribution in a reservoir. Both entities and their properties are stored in Petrel projects as domain objects. There are several kinds of domain objects native to each platform, however the Ocean framework also allows developers to create their own custom domain objects (CDO) to be used in their plug-ins. These CDOs can also be used to extend native domain objects to be used in a plug-in.

7.1.2 CDO Usage

Not every plug-in requires the use of CDOs. The developers interviewed seemed to tie the usage of CDOs to how well established the domain of the plug-in is. A plug-in dealing with cutting edge processes in electromagnetism in oil exploration, for instance, might not have sufficient native objects in Petrel, and one or more CDOs might be needed. Developers generally expressed that they preferred to use native objects as much as possible, only using CDOs as a last resort. *”Our policy is to try to not re-invent the wheel, so we try to make sure we don’t develop features that we know are coming to Petrel soon”* one developer commented. However, due to the constant advances in the E&P industry, it was often the case that CDOs were a necessity.

Occasionally functionality from a plug-in may later be included wholly or partially in a Petrel release or in a Petrel core, which also creates a native version of any CDOs required

for that functionality. One developer commented that in their case this was not a problem, as their plug-in was competitive in price compared to the core, and also that the lack of compatibility with a user's stored work done in their plug-in caused a lock-in effect where users continued to use the plug-in rather than switch to the native solution and have to redo work.

7.2 The problem

Most domain objects used in the platforms are native to the platform. As mentioned above, however, developers can include their own custom domain objects in their plug-ins. In itself, this does not pose a problem. A developer using a custom domain object in their plug-in to better represent the user's data should, after all, be a good thing. The issues arise when the user runs two or more plug-ins, presumably from different developers, that both represent the same data in different custom domain objects. This results in the same data being represented twice or more without benefit to the user. Users of Petrel report frustration with having to interact with these multiples of what is, to the user, essentially the same object. In addition to the inconvenience described by end-users of the software, data managers have also described the multiples of data as a problem for them.

As an example, let us assume that a pioneering new technique *"new-tech"* for determining the location and amount of oil in a given area has been developed. Since this is an entirely new concept, no support for it exists in Petrel. Developers Alice and Bob each decide to take advantage of this by creating plug-ins that provide features for E&P companies looking to make use of this new type of exploration. The two developers create `plug-in A` and `plug-in B`, respectively, each with some distinct features. Both plug-ins use the same algorithm to create a CDO for an area's initial new-tech outline from already present data that they then do work on, however Alice and Bob develop and commercialize their plug-ins separately.

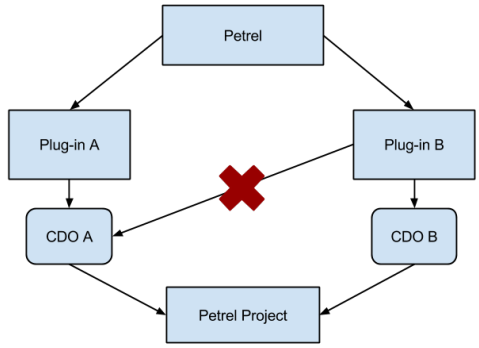


Figure 7.1: Two plug-ins create different CDOs

A Petrel user, Carol, works in an E&P company interested in using new-tech exploration. Both `plug-in A` and `plug-in B` have features that Carol thinks would be useful, so she licenses and installs them both. They both generate their own new-tech outline of a geographical area that Carol is working on, as the CDOs `CDO A` and `CDO B`. Initially these two CDOs are identical, since the plug-ins happen to generate them the same way, and the fact that there are two different CDOs in use by the plug-ins is not apparent to Carol. She then uses one of `plug-in A's` features which does some work on the outline and alters it. After the changes have been made, Carol wants to make use of some of `plug-in B's` features in order to do some work on the updated outline. Now she notices that there is a discrepancy. This is because what to her was the same business object, was represented by two distinct objects by the plug-ins. `Plug-in B` cannot access the updated `CDO A`, and is only able to do work on `CDO B`, which it created. Carol now realizes that she is stuck with two new-tech outlines for the same area in her project, if she wants to continue using the features of both plug-ins. Additionally, each feature set will only work with one of the outlines and updates to them will be done individually.

7.2.1 Impact on users

See above. In addition, SIS has received reports from those users responsible for data management that it adds unneeded complexity to their work.

7.2.2 Impact on developers

Besides the fact that it can sometimes mean ”re-inventing the wheel” and causing unnecessary work, overlapping CDOs can also have an impact on developers by limiting the interoperability between plug-ins. This impact is not necessarily a negative one, however. One developer noted that it could have a *”sometimes desirable lock-in effect for us”*, while still acknowledging it as an issue. Another noted that they indeed think it would be preferable to be rid of the problem altogether and had no problem with potentially sharing their CDOs, if able: *”We usually find that our important IP is in the actual business logic. We would not be opposed to others being able to make use of CDOs we create. Moving maintenance over to SIS would benefit everyone.”*. Implied here is that SIS recreate the CDO as a native DO in Petrel.

A related issue that also came up during interviews was the inability to reach native DOs that were children of CDOs from different plug-ins, i.e. instantiated as properties of a CDO.

7.2.3 Impact on owner

As these overlapping domain objects create usability problems for end-users, this, in turn translates into a problem for SIS as the platform owner through a potential loss of customers.

7.3 What is being done

Interviews with SIS staff reveal that they currently have a, self-described, ”reactive approach”. *”We can’t just pre-emptively create objects”*, one Ocean representative remarked, as the workload on the Ocean developer team is already high enough in addition to the potential bloating it might cause. If problems of this nature arise, the response from SIS is usually to implement their own solution after the fact and attempt to get everyone on board. It also takes time before objects can be added natively. This means that the developers of the plug-ins with the offending domain objects are encouraged to migrate to the SIS solution. They are not forced to migrate, meaning it is up to the developers themselves to decide how they would like to proceed. On the one hand code migration can potentially be very costly for these developers, on the other it can mean future savings in terms of reduced code maintenance. Several developers noted that they would prefer for SIS to carry the responsibility and centralize maintenance.

Part V

Discussion

Chapter 8

Reasoning

This part of the thesis examines the Ocean ecosystem closer by applying some of the frameworks already presented in the literature review, for the purpose of providing an overarching analysis of the ecosystem both for future academic use and in relation to the case. Due to the nature of the case, extra attention is paid to the governance of the ecosystem and the collaborative aspects. The reason for this is that an expected solution would need some measure of customer involvement in such a decentralized environment, and the orchestration of such a solution would inevitably fall within the purview of the ecosystem's governance.

The models and frameworks covered here are:

- Ecosystem taxonomy by Bosch (2009)
- Ecosystem classification model by Jansen and Cusumano (2012)
- Ecosystem governance analysis framework by Baars and Jansen (2012)
- Ecosystem governance model and tools by Jansen and Cusumano (2012)
- Decision making framework by Manikas et al. (2014)

- Co-creation typology by O'Hern and Rindfleisch (2010)
- Co-creation typology by Piller et al. (2010)

Chapter 9

Ecosystem

9.1 Applying frameworks

In order to investigate possibilities surrounding the case outlined in the previous chapter, we further analyse the Ocean ecosystem. In this section, ecosystem frameworks and models from the literature review are applied to the Ocean ecosystem, in particular as they relate to ecosystem governance. This also serves the purpose of expanding the, somewhat lacking, number of cases available in academic literature on the subject.

9.1.1 Taxonomy

In the taxonomy proposed by Bosch (2009) it is clear that the Ocean ecosystem can be described as an **application-centric software ecosystem**. As with many other such ecosystems, it had its start with a successful domain-specific application, in this case, Petrel. Its history further conforms with the expected development of such an entity. The volume and nature of new requests were too large to be feasible to develop in-house, and the application became a domain-specific platform with a surrounding ecosystem complete with development tools for third parties, the Ocean framework.

To reiterate some of what Bosch (2009) mentioned regarding pitfalls for this kind of ecosystem, the original user base of the application, now platform, are users of it primarily due to the existing functionality. They may not be aware of or realize the extended functionality available to them. Actively facilitating the interaction of users and developers, as well as improving the visibility of extensions available, therefore becomes important in order to attract developers.

9.1.2 Classification

Using the classification model proposed by Jansen and Cusumano (2012), the Ocean ecosystem is described in terms of the four classification factors:

Base technology: The Ocean ecosystem is underpinned by several **platforms**, although it could be argued that, at present, Petrel is the actual basis for the ecosystem. The Studio, Techlog and Avocet platforms might become more prominent in the future, if the plans for integration are realized. In which case, the new unified platform might be considered the base technology as a single platform.

Coordinators: The ecosystem is **privately owned** by Schlumberger, more specifically SIS. They are the owners and decision makers of the ecosystem. This is to be expected, as they are also the owners of the base technology being used.

Extension markets: The classification model calls for this factor to fit into one of five cases: No extension market, a simple list of extensions, an actual extension market, a commercial extension market, or multiple extension markets. The Ocean Store, operated by SIS, is clearly a **commercial extension market**, with SIS taking a portion of the revenue from sales. It is however worth noting that not all plug-ins are distributed through the Ocean Store. Many plug-ins are created through agreements made between two ecosystem actors directly. Some E&P companies also develop plug-ins themselves, for internal use. As such, the truth might lie somewhat outside the description of just a commercial market, as distribution of plug-ins exists not just within the Ocean Store.

Accessibility: **Paid**. In order to develop usable plug-ins using the Ocean framework, a paid license is required. In addition, there is a screening process in order for a developer to be allowed purchase such a license. Distributing a plug-in through the Ocean Store requires further quality control, and that a portion of the revenue goes to SIS.

Name	Base technology	Coordinators	Extension market	Accessibility
Ocean ecosystem	platforms	privately owned	a commercial market	paid

9.1.3 Governance analysis framework

When applying the governance analysis framework proposed by Baars and Jansen (2012), we get the following table. Some concepts are **marked with ***, meaning that a yes/no answer was deemed insufficient and that an explanation can be found below.

Partnerships: The Ocean ecosystem does operate with a partnership model. Partnership in the ecosystem involves being an Ocean developer with at least one commercial plug-in on the Ocean Store. This means that the developer must have produced a plug-in that passed the quality control for the Store. The partnership model is tiered, with the title of certified partner being reserved for partners that can demonstrate consistent adherence to the acceptance criteria in the quality control, and generally create high quality software. Status as a certified partner is reviewed annually, and as such, it is possible to lose this status.

Category	Governance concept	Result
Partnerships	Creating a partnership network	Yes
	Degree of moderation	*
	Degree of division in tiers, levels, etc	*
	Acquiring new partners	*
	Formalization of entry requirements	Yes
Supplier and customer governance	Coordination of contribution to other ecosystems	N/A
	Setting up new suppliers	N/A
	Changing the ratio of current suppliers	N/A
	Ceasing cooperation with suppliers or customers	N/A
	Using intermediaries	N/A
Development	Creating a development standard	Yes
	Enforcing a development standard	Yes
Partner directory	Creating a partner directory	Yes
	Degree of moderation	*
Customer directory	Creating a customer directory	No
	Degree of moderation	N/A
User groups	Creating active user groups	Yes
	Degree of moderation	*
Licence(s)	Creating reusable software license(s)	No
Category	**Governance structure concept**	**Result**
Ecosystem explicitness	Is the ecosystem explicit?	Yes
	Is there documentation describing its current state?	No
Governance explicitness	Is the ecosystem governance strategy explicit?	Yes
	Are processes and procedures formalized?	Yes
	Are there formalized and documented rules?	Yes
	How is business strategy formalized to governance strategy?	*
Responsibility	Where in the organization does ecosystem governance take place?	*
	Who does the decision making unit consist of?	*
	Is this decision making unit made explicit?	No
	Does the decision making unit report to the Board?	No
Measurement	Is the effectiveness of the ecosystem measured?	Yes
	Which parts of it are measured?	*
	Which KPIs are used?	*
	How are goals defined?	*
Knowledge sharing	Does the organization share its knowledge with other companies?	

Development: Partners, and certified partners in particular, are expected and required to conform to a certain level of quality in their finished products. The actual development process, however, is not regulated by SIS, beyond an imposed time limit for developing a marketable plug-in.

Partner directory: Partners and certified partners are listed on the Ocean Store and on the website for the Ocean framework itself. The list is curated by SLB.

User groups: There is an official user group for Ocean, with annual events. These events include presentations by SIS and partners, as well as workshops. They are meant to promote interaction and best practices. In addition there are message boards available where developers can interact. There are also several events for Petrel customers. Ranging from small local events, to bi-annual summits that are invite-only for the 30 largest customers. There is also the more, marketing-oriented SIS global forum, with a 1000+ attendants.

Governance explicitness: Rules, procedures and guidelines for the actors in the ecosystem are clearly explained in their individual contracts. The stated purpose of the ecosystem is to increase competency of all actors and to facilitate distributed innovation, with the goal of increasing the value of the platforms involved.

Responsibility: Decisions regarding the individual platforms are made by the portfolio organization for that product. This includes the portfolio manager, legal team, partner manager, marketing team, product champions and support teams. For the ecosystem as a whole, this effectively means the portfolio organizations for Petrel and Ocean. These organizations then report to SIS management.

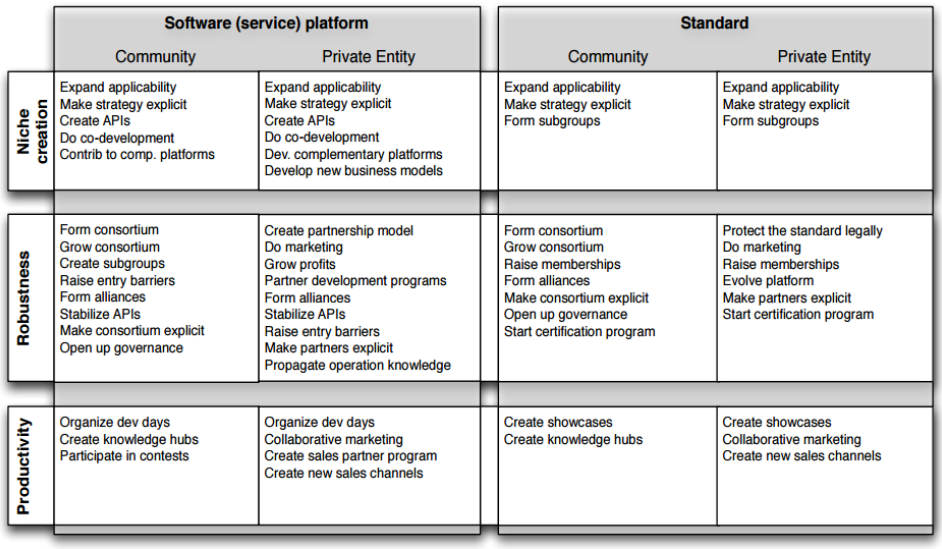
Measurement: The ecosystem is measured primarily through number of agents involved and the number of plug-ins that are distributed. As the ecosystem's purpose is to increase the value of the existing platforms, the actual revenue from plug-in sales through the Ocean Store is not as important as the number of sales. Customer satisfaction gathered from surveys is also used as an indicator. Goals for the ecosystem and the platforms themselves are generally given as percentage-based increases in certain indicators, though occasionally they are also in the form specific goals, such as increasing sales in a given region.

Knowledge sharing: Through openness in the ecosystem, SIS aims to promote distributed innovation and draw in outside expertise. Knowledge is shared to as high a degree as possible within the limits of not violating IP rights. Certain information regarding the ecosystem and it's partners, such as financial records, are also not shared.

9.1.4 Governance tools

Based on the results from **9.1.2**, the Ocean ecosystem is clearly based on privately owned software platforms. In this subsection we will go through the governance tools for such ecosystems as outlined in Jansen and Cusumano (2012) and compare them to the Ocean ecosystem.

Niche creation: SIS facilitates niche creation largely through the Ocean Framework, aiding and allowing third parties to contribute extended functionality through plug-ins for the platforms. Niche creation is further enhanced by the creation of new business models through the existence of the Ocean Store, which allows for distribution



	Software (service) platform		Standard	
	Community	Private Entity	Community	Private Entity
Niche creation	Expand applicability Make strategy explicit Create APIs Do co-development Contrib to comp. platforms	Expand applicability Make strategy explicit Create APIs Do co-development Dev. complementary platforms Develop new business models	Expand applicability Make strategy explicit Form subgroups	Expand applicability Make strategy explicit Form subgroups
Robustness	Form consortium Grow consortium Create subgroups Raise entry barriers Form alliances Stabilize APIs Make consortium explicit Open up governance	Create partnership model Do marketing Grow profits Partner development programs Form alliances Stabilize APIs Raise entry barriers Make partners explicit Propagate operation knowledge	Form consortium Grow consortium Raise memberships Form alliances Make consortium explicit Open up governance Start certification program	Protect the standard legally Do marketing Raise memberships Evolve platform Make partners explicit Start certification program
Productivity	Organize dev days Create knowledge hubs Participate in contests	Organize dev days Collaborative marketing Create sales partner program Create new sales channels	Create showcases Create knowledge hubs	Create showcases Collaborative marketing Create new sales channels

Figure 9.1: Reminder of the governance model by Jansen and Cusumano (2012)

and sale of plug-ins. Explicitness of strategies involved in the ecosystem is achieved through clear rules and regulations that members have access to, as well as certain members getting access to early information on the direction of the Petrel platform and the Ocean Framework. One might argue the existence of the development of complementary platforms in the Ocean ecosystem, as all the involved SIS products are eventually meant to be interconnected. However, as neither of these are external to the ecosystem, they are only complementary to each other, not the ecosystem itself. Co-funding and co-development (not to be confused with co-creation, discussed in chapter 10) by SIS is done to a certain degree through the sponsoring of academic endeavours, with competitions and donated licences.

Robustness: As mentioned previously, the Ocean ecosystem does operate with an explicit partnership model for third party developers, which also them to be marketed in the Ocean Store. Access requirements to this partnership and its different levels also serves as an entry barrier and, as such, should contribute to the growth of a stable core of committed members. As the Ocean Framework is its own commercial product, APIs created by SIS in the ecosystem are well maintained.

Productivity: The Ocean user group's annual meetings contain workshops in addition to presentations, but designated dev. days are not organized by SIS. A form of collaborative marketing is done in the form of the perks for preferred partners, as they are featured more prominently on the Ocean Store and can be presumed to have a respectable relationship with SIS.

9.1.5 Decision making framework

Here we will apply the decision making framework presented by Manikas et al. (2014) to the Ocean ecosystem. To quickly recap: the framework defines decision making strategies for an ecosystem based on five decision areas and four archetypes. Below we will go through each of these decision areas and identify the corresponding archetype for the Ocean ecosystem, describing how decisions are made.

Principles: Core principles, values and the main direction of the ecosystem is controlled by SIS, as they are the proprietors of the platforms involved. As such, the most appropriate archetype for this decision area is **monarchy**.

Actor Interaction: Much like the previous decision area, access to the ecosystem and the rules governing actor interaction is controlled by SIS alone. **Monarchy**.

Software Interaction: In this decision area, the ecosystem begins to lean in the direction of a collective. Requests for upcoming features for the SIS products involved in the ecosystem are gathered, and in some cases voted for. The extent to which an actor has influence over these decisions can vary and is unknown, as the final say rests with SIS. Because of this, it would be more correct to also classify the decision making process in this area as a **monarchy**, as well.

Platform: See above.

Ecosystem business and products: The Ocean Store is controlled and curated by SIS, however decisions regarding distribution outside this market can be made by individual actors. Decision making in this area then falls somewhere between monarchy and anarchy. Due to the degree of autonomy that exists in regards to business conducted outside the Ocean Store, however, it can be argued that **anarchy** is the most fitting archetype.

	Principles	**Actor interaction**	**Software interaction**	**Platform**	**Business and products**
Monarchy	X	X	X	X	
Collective					
Federal					
Anarchy					X

Table 9.1: Decision making framework applied to the Ocean Ecosystem

Chapter 10

Co-creation

10.1 Applying frameworks

Taking part in a software ecosystem, regardless of position, can be seen as an ultimately collaborative effort where one actor's actions affect the others. Considering the case presented in this thesis in particular, some manner of orchestration is called for. In this section, the co-creation typologies from the literature review are applied to the Ocean ecosystem. In applying these typologies in this setting, the roles of firm and customers no longer fit in the traditional sense. Instead, we here deal with platform owners and third party developers, as well as end users to a lesser degree.

10.1.1 Co-creation typology 1

The first typology is the one proposed by O'Hern and Rindfleisch (2010), which defines four types of co-creation. **Collaborating**, **tinkering**, **co-designing** and **submitting**. The Ocean ecosystem does, of course, allow for the extension of functionality provided by the platforms. Full-on, unrestricted access to the source code is, however, not provided. As such, there is no collaborating in the ecosystem, as precisely defined in this typology. The type of co-creation which most fittingly describes the Ocean ecosystem, is **tinkering**. SIS retains control over their IP and product, however they provide the tools for extending this functionality through the Ocean framework.

One could argue that there are elements of the co-designing and submitting types of co-creation, as well, although these may not be differentiated enough from traditional forms of customer inquiry such as focus groups and surveys. These elements can be found in the requests that are routinely accepted by SIS for improvements to both the platforms and the Ocean framework, which are a form of **submitting**. These requests are taken into account by SIS, and a backlog of requested features is built up. The prioritization of which features are to be included, and when, is completely up to SIS, however. One developer noted, for example, that they were mostly happy with the Ocean framework,

but felt that it was *"difficult to get our requested improvements implemented by SIS"*. An Ocean representative explained that incoming requests had to be weighted according to feasibility, as well as the size and importance of the customer making the request. As such, not all requests can be implemented. A more direct use of submitting can be seen in the Ocean Academic Competition, where participants submit working plug-ins that aim to solve specific challenges.

10.1.2 Co-creation typology 2

The second typology in this chapter is the one presented by Piller et al. (2010) for co-creation in the innovation process. To quickly recap, the types here are based on a set of three characteristics; At what stage in the innovation process the co-creation occurs, the degree of collaboration between actors, and the degree of freedom that the actors have. With two alternatives for each of these characteristics, the possible combinations give us eight different types of co-creation within the typology. These are: Idea contest, idea screening and evaluation, product related discussion forums, communities of creation, toolkits for user innovation, toolkits for customer co-design, communities of creation for problem solving, and virtual concept testing.

The Ocean ecosystem is heavily based around two of these types of co-creation, namely the toolkits. While the Ocean framework does mostly fall into the **toolkit for customer co-design** type, due to the restrictions inherent in the available APIs, it could also be argued that it also serves as a **toolkit for user innovation** in accordance with the definitions provided in the typology, as the framework does allow for experimentation with original "building blocks". This is one of the key features of a user innovation toolkit, and so I would argue that the Ocean framework has elements from both of the co-creation toolkit types.

We also find the use of **product related discussion forums** in the online forums hosted by SIS themselves, as well as some use of Stack Overflow with the "ocean" or "petrel" tags. These forums see little activity by the larger development houses, however. Developer's interviewed expressed that they felt little need to use them, preferring to ask/answer questions internally. Somewhere between these online forums and the **communities of creation** type of co-creation, we find the Ocean user group which meets twice per year with presentations and workshops. These meetings are hosted by SIS, but also feature presentations by the third party developers in the community.

The **Idea contest** type of co-creation is also present in the ecosystem, through the regional Ocean Academic Competitions for academia held by SIS each year. It should be noted, however, that these contests are more in-depth than what is defined in the typology, as they require participants to submit working code.

Chapter 11

Proposed improvements

11.1 Possible solution

The interactions between actors in the Ocean ecosystem appear to be effectively dyadic, that is to say 1:1. Either between SIS and the individual actors, or between individual third party developers and individual end-users of the platforms. Other than the Ocean user group meetings, there appears to be little interaction between third party developers. In order to reach a solution to the case presented, however, it would seem reasonable that a more community-oriented approach might be suitable, if SIS is to retain a hands-off approach to the development process of plug-ins in terms of governance.

Looking to the typologies in chapter 10 we identify that **submitting** and **communities of creation** both facilitate innovation between actors towards a common goal even at a fairly late stage in the process.

Taking a cue from other online communities, a possibility could be to open an online space for submissions of CDOs. If a developer suspects that a CDO in their up-coming plug-in might be useful outside their plug-in, and therefore be likely to be reproduced in some way by a different developer, they can submit it here. Interviews suggest that developers do not consider the CDOs themselves to be important IP, and have few reservations about sharing them. This online service would be open to all licensed Ocean users (third party developers) to submit their CDOs to. These submissions can then be voted on by other Ocean users, if they see it as a useful addition, bringing in an element of screening and evaluation done by the community. Submitted CDOs can also be edited by other users, effectively becoming open source within the ecosystem, and letting the CDO be applicable to more plug-ins.

By letting the community share in the development and maintenance of the CDO, the time between iterations is kept low, and more importantly, it can see use by multiple plug-

ins without having to wait for a new release from SIS in order to be centralized. When the time does come for a new release by SIS, the highest voted submission(s) can be adopted natively and SIS takes over the responsibility for maintenance of the CDO. This removes the workload previously shouldered by the Ocean community, while still keeping the CDO available for use. All without needing to retroactively coordinate and promote the usage of the centralized CDO, as it is already in use where needed.

A possible challenge in getting this solution to work, is to get and keep the community motivated enough sustain the co-creation. As opposed to traditional open source ecosystems, privately owned commercial ecosystems do not have the same ”sense of community”. Ocean users will therefore have to be convinced that actively participating in the co-creation is something that will ultimately be beneficial, or be incentivized through other means. Making further use of the partnership model might be an alternative, by letting preferred partners be more influential in the system. This would provide added value to the status of preferred partner, as well as push decision making in a more federal direction, keeping more in line with the stated ”hands-off” approach of SIS. A similar challenge that is also a possibility is a lack of activity due to an inability to reach ”critical mass”. The possibility that there simply aren't enough developers affected by the issue to justify co-creation, and the initiative fades. This might lie outside the possibility for governance or orchestrating co-creation to mend, as it relies on the inherent value of the proposition itself.

Given that these hurdles are overcome, however, a successful implementation might resolve the current issue, while also starting to foster a stronger community, which in turn is associated with an increase in overall innovation and niche production in the ecosystem.

Part VI

Conclusion

Chapter 12

Conclusion

This thesis has taken a closer at some of the existing literature in the field of software ecosystems and customer involvement. We have also looked closer at a software ecosystem surrounding multiple software platforms in use by the E&P industry today. The existing literature in the areas of software ecosystem, in particular governance, and customer co-creation have been presented and frameworks from this literature has subsequently been applied to the ecosystem. Several tools have been found in academia for analysis of software ecosystems, however studies actually making use of these tools are sparse. This has hopefully been a useful addition to the, presently, small number of cases in the growing academic field of software ecosystems and governance.

A proposed solution to the particular problem of duplicate assets being developed in an ecosystem has also been provided, grounded in the academic literature presented and field work performed as part of this thesis. The solution is presently untested, and so the efficacy of the literature it is built on remains uncertain. If successful, this solution could also be applicable to other software ecosystems. The potential seems present for ecosystem literature to provide value to owners of ecosystems, although further research should continue to be done in the field in order to gather empirical data to confirm efficacy and improve.

12.1 Limitations of this work

One major limitation of this work is the inability to fully deploy and test the proposed solution, due to time and scope constraints. Even more interview subjects could also have been useful, in order to gain more perspectives on the situation. This study has also been purely contemporary, and so does not offer insight into long-term effects of any subjects discussed.

12.2 Future work

A deployment of the proposed, or similar, solution combined with a longitudinal case study to assess its usefulness over time would be a natural follow-up to the work presented in this thesis. Care should be taken to record both the response of the actors in the ecosystem, as well as the gathering of quantitative data that might suggest the value created by the solution.

Bibliography

Baars, A., Jansen, S., 2012. A framework for software ecosystem governance. Software Business, 168–180.

Baldwin, C., Woodard, C., 2008. The Architecture of Platforms: A Unified View. Harvard Business School Working Paper No. 09-034.

Berk, I., Jansen, S., Luinenburg, L., 2010. Software ecosystems: A software ecosystem strategy assessment model. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. Copenhagen, Denmark.

Bosch, J., 2009. From software product lines to software ecosystems. In: SPLC '09. Proceedings of the 13th International Software Product Line Conference. San Francisco, California, the United States of America.

Bosch, J., 2012. Software ecosystems: Taking software development beyond the boundaries of the organization. Journal of Systems and Software 85 (7), 1453–1454.

Ceccagnoli, M., Forman, C., Huang, P., Wu, D., 2011. Co-creation of value in a platform ecosystem: The case of enterprise software. MIS Quarterly.

Gawer, A., 2009. Platforms, Markets And Innovation. Edward Elgar Publishing Limited.

Gawer, A., 2014. Bridging differing perspectives on technological platforms: Toward an integrative framewok. Research Policy 43 (7), 1239–1249.

Hagiu, A., Wright, J., 2011. Multi-Sided Platforms. Harvard Business School Working Paper No. 12-024.

Hyne, N., 2001. Nontechnical Guide to Petroleum, Geology, Exploration, Drilling, and Production. PennWell Corporation.

Hyrynsalmi, S., Makila, T., Jarvi, A., Suominen, A., Seppanen, M., Knuutila, T., 2012. App store, marketplace, play! an analysis of multi-homing in mobile software ecosystems. Jansen, Slinger, 59–72.

Jansen, S., Bloemendal, E., 2013. Defining app stores: The role of curated marketplaces in software ecosystems. Software Business. From Physical Products to Software Services and Solutions, 195–206.

Jansen, S., Cusumano, M., 2012. Defining software ecosystems: A survey of software platforms and business network governance. In: Proceedings of the Forth International Workshop on Software Ecosystems. Cambridge, MA, USA.

Kilamo, T., Hammouda, I., Mikkonen, T., Aaltonen, T., 2012. From proprietary to open source - growing an open source ecosystem. Journal of Systems and Software 85 (7), 1467–1478.

Laver, R., 2012. How geoscience support software tools have evolved and what it means for e & p companies. First Break 30 (6), 121–125.

Manikas, K., Hansen, K., 2013. Software ecosystems - a systematic literature review. Journal of Systems and Software 86 (5), 1294–1306.

Manikas, K., Wnuk, K., Shollo, A., 2014. Defining decision making strategies in software ecosystem governance.

Oates, B., 2005. Researching information Systems and Computing. Sage.

O'Hern, M., Rindfleisch, A., 2010. Customer co-creation. Review of marketing research, 84–116.

Piller, F., Ihl, C., Vossen, A., 2010. A typology of customer co-creation in the innovation process.

PSAC, March 2015. Industry overview.
URL http://www.psac.ca/business/industry-overview/

Romberg, T., 2007. Software platforms – how to win the peace. In: HICSS 2007. 40th Annual Hawaii International Conference on System Sciences. Waikoloa, Hawaii, the United States of America.

Roth, M., 2005. Automating petrotechnical workflows. Hart's E & P 78 (12), 75–77.

Schlumberger, November 2014a. Corporate profile.
URL http://www.slb.com/about/who.aspx

Schlumberger, November 2014b. Financial news.
URL http://investorcenter.slb.com/phoenix.zhtml?c=97513&p=irol-resultsNewsArticle&ID=1891675&highlight=

Schlumberger, November 2014c. Getting started with ocean.
URL http://www.ocean.slb.com/Docs/gettingstarted/GettingStartedWithOcean-2014-1.pdf

Schlumberger, November 2014d. History.
URL http://www.slb.com/about/history.aspx

Schlumberger, November 2014e. Petrel ep software platform.
URL http://www.software.slb.com/products/platform/Pages/petrel.aspx

Schlumberger, November 2014f. Studio product sheet.
URL http://www.software.slb.com/lists/salesandmarketingdocuments/attachments/426/studio-ep-environment.pdf

Schlumberger, November 2014g. What is ocean.
URL http://www.ocean.slb.com/Pages/ocean-what-is.aspx

Schlumberger, Microsoft, November 2014. Uniting the business and petrotechnical worlds.
URL http://www.software.slb.com/Lists/SalesandMarketingDocuments/Attachments/233/5102%20Schlumberger%20and%20Microsoft%20White%20Paper%20%281%29.pdf

Stanley, M., Farrell, J., 1994. Choosing how to compete: Strategies and tactics in standardization. The Journal of Economic Perspectives 8 (2), 117–131.

Tiwana, A., 2014. Platform ecosystems: aligning architecture, governance, and strategy. Elsevier.

Ververs, E., Bommel, R., Jansen, S., 2011. Influences on developer participation in the debian software ecosystem. Proceedings of the International Conference on Management of Emergent Digital EcoSystems, 89–93.

Weill, P., Ross, J., 2004. IT governance: How top performers manage IT decision rights for superior results. Harvard Business School Press.

Appendix