Do you mind if I keep that?

Value extraction in software ecosystems

Jakob Buis

Universiteit Utrecht

Final thesis
August 6, 2015

quae non valeant singula, iuncta iuvant

Administrative information

Author:	dhr. Jakob Buis BSc
Student number:	3699080
E-mail:	jakob@jakobbuis.nl
Document version:	2015-08-06 21:09
First supervisor:	dr. Slinger Jansen
Second supervisor:	prof.dr. Sjaak Brinkkemper
Organisation:	Center for Organization and Information Department of Information and Computing Sciences Utrecht University
Address:	Buys Ballotgebouw Princetonplein 5 3584 CH Utrecht

What does the quote on the previous page mean?

It is a legal principle: "What is without value on its own, helps when joined". It is used here in reference to our subject of value in software ecosystems: that what may not be valuable to you, might be of value when shared with others. The quote also applies to the thesis as a text if we adopt its alternative interpretation: "[words] which have no meaning when considered separately, obtain their sense when they are brought in connection with one another".

Acknowledgments

This thesis was not produced in a vacuum and it is all the better for it. A number of people have contributed to the ideas, processes and chapters of this project, and I would like to thank a few of them explicitly. My first supervisor dr. Slinger Jansen, for working with me through this project, providing both critical and supportive comments on every aspect of it. Martijn Feekes and prof.dr. Sjaak Brinkkemper for their critical review of the partial ideas, concepts and chapters that would become a finished thesis. All colleagues at Exact, especially Dimitri, Erich, Jurjen, Martijn, Pieter, Remco and Ruben who offered many new ideas through our discussions and new insights I could never have derived by writing this thesis purely in academia.

Honestly, it is hard to trace every discussion, flash of insight or provoking remark made by a friend, family member or colleague that helped to shape my work; there were so many. If you ever heard me say “wait, why didn't I think of that..” in response to your statement, consider yourself thanked. Please do remind me of it, so that I may properly thank you in person.

Summary

Software companies no longer operate in a vacuum, but share their products and services, knowledge and expertise with other companies in various software ecosystems. The reasons for participating in a software ecosystem are based upon the (business) value the participants receive from these relationships compared to the value they give up. Exchanging value through software ecosystems is not sufficiently studied so far. To cover this research gap, a formal definition of value in software ecosystems is created, together with an overview of the forms of value as they appear in software ecosystems. The Value Exchange Graph (VEGA) is developed to model the movement of value through the software ecosystem. The VEGA can be used to accurately model both the impact of specific actions or strategies, and describe the relationships of a software company in a software ecosystem. The model also supports formal reasoning over its properties to a considerable degree, though applications are limited. The model is validated using scenarios from both scientific literature and case studies of various software companies.

Contents

Chapter 1

Introduction

Actors operate in software ecosystems in conjunction with others. Their actions affect the other actors with which they are in direct contact, but also the contacts of their contacts, and the software ecosystem as a whole. Specific roles can be attributed to the positioning and behaviour of an actor and this has been extensively studied in previous literature. In this research project, the authors study "(business) value" from the viewpoint of an individual actor in a software ecosystem. The actors in a software ecosystem, through their action or inaction, capture value from the software ecosystem for themselves or spread value to other actors. This research project aims to show that modeling these "value flows" can improve our insight into software ecosystems, in both theoretical scenarios and real-life applications. A model is created to depict value flows in a software ecosystems, the Value-Exchange Graph (VEGA). This model is validated using strategies retrieved from scientific literature and case studies of software companies.

In chapter 2 we discuss the existing research in scientific literature which frames our work. The basic concepts and definition of a software ecosystem are described. Additionally, we discuss the concept of software ecosystem health, based primarily on the seminal work by Iansiti and Levien, and discuss the existing literature related to the concept of "value" in software ecosystems. Finally, we discuss the application of strategy in software ecosystems. These sources of literature are modeled in a conceptual model.

This conceptual model is the primary tool to convey the research gap. A number of areas in the conceptual model are not sufficiently studied in scientific literature, and these form the challenges we seek to address in this project. From these triggers, the three main research questions are posed. Chapter 3 provides an exhaustive overview of the research approach taken and its known limitations.

Chapter 4 describes the detailed approach taken in conducting the SLR and the results of it. These results provide the answer to the first and second research questions. Chapter 5

describes the VEGA notation for modeling flows of value in software ecosystems. Based on a formal meta-model, the VEGA notation is used to describe existing software ecosystem strategies retrieved from scientific literature. Chapter 6 explains the concept of using formal reasoning over value flows to evaluate strategies and how this can be used to improve the situation of a company, or adapt existing strategies to better suit the needs of all parties involved. The suitability of VEGAs to describe real-world scenarios is evaluated using a number of case studies in chapter 7. The approach taken in the interviews used to create these cases and the subsequent results are all detailed. Chapters 5, 6 and 7 combined form the answer to the third research question. Finally, the discussion and main conclusions of the project are drawn in chapter 8.

Chapter 2

Theoretical background

2.1 Introduction

In this chapter, we introduce the theoretical foundation of our work, starting with related work on software ecosystems (2.2). The related work is elaborated in a conceptual model (2.3), describing our view of the subject. This model forms the basis of our research project and is directly mapped onto the research triggers (2.4). The chapter concludes with the research questions which this research project will answer (2.5).

2.2 Related work

Software ecosystems is not a new field in scientific research and much has been written on the subject that does not require reiteration here. For a solid overview of the state of the art in software ecosystems research, we refer the interested reader to the literature overview by Manikas and Hansen [50]. We start with a few key points in software ecosystems research that are crucial to our understanding and help explain our point of reference to software ecosystems (2.2.1), introduce the concept of software ecosystem health (2.2.2), discuss the basic principles of value and value extraction (2.2.3), and touch on strategies in software ecosystems (2.2.4).

2.2.1 Basics of software ecosystems

Software is no longer built in isolation, with traditional software companies soliciting requirements from customers, handing these over to development, and passing the resulting

updates and patches back to the customer. Software vendors are increasingly reliant on other companies to work together to integrate product software [42, 43]. The resulting "software ecosystems" have a solid definition by Jansen, Brinkkemper and Finkelstein [43]:

> [A software ecosystem is] a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts.

We recognize several distinctions in the roles companies play in their software ecosystems. The five roles defined by Iansiti and Levien [37]: hubs, keystones, landlords, dominators and niche players; are the most famous and most often used. Hubs occupy central positions in a software ecosystem, with many crucial connections that hold the ecosystem together. The hub role is further divided into three roles, according to the behaviour of the company. Keystones maintain the health of their ecosystems through specific actions that affect their entire ecosystem. Landlords extract too much value from the ecosystem using their central hub position, and do not create or leave sufficient value for others. They extract maximum profit while making the business models of their peers unsustainable. Ultimately, landlords kill off their ecosystem, leaving themselves vulnerable. Dominators seek to control large areas of their ecosystem. It integrates other business horizontally and vertically to maximize the value it captures. Dominators are not necessarily a bad influence: dominating a niche can be effective and can potentially increase the health of the ecosystem as a whole, but dominators in hub positions make the ecosystem unsustainable in the long run. By eliminating the openness that allows others to build upon their product and efforts, dominators essentially reform the ecosystem back into the pre-connected world of closed, monolithic (software) products. Finally, niche players are the natural complement of hubs. Niche players have an average number of connections, none of which are crucial to the ecosystem. The loss of a niche player has no direct effect on the composition or health of the ecosystem. However, niche players make up the bulk of every ecosystem. Their nimbleness at the edges of their ecosystems allows for maximum innovation. [37]

2.2.2 Software ecosystem health

In order to create a software ecosystem that is as effective as possible, network effects are key [31, 37], increasing the value and stability of the network as it grows. Health is a weak metaphor in the context of software ecosystems [31], allowing us only to make relative statements e.g. "this is more healthier than that", but not absolute statements as "this ecosystem is 80% healthy" or "this ecosystem is not healthy". Research on the health of business- and software ecosystems [31, 36, 37] has identified a number of ways to quantify the health of a software ecosystem. Iansiti and Levien identified three determinants of ecosystem health: robustness, productivity and niche creation [36]. Robustness is the resilience of the network

in the face of external changes. Productivity is the efficiency of turning the inputs of the ecosystems into outputs, usually profits. Niche creation is the ability of the network to create and protect profitable niches for niche players to invest.

2.2.3 Value in software ecosystems

Increasingly, businesses are seen as "a major cause of social, environmental, and economic problems" [61], especially if an approach is taken that seeks to exclusively benefit the company in detriment to its environment. While some measure of competition is healthy and inevitable, new work by Porter and Kramer introduces the principle of *shared value*, defined as "policies and operating practices that enhance the competitiveness of a company while simultaneously advancing the economic and social conditions in the communities in which it operates" [61]. For example, using the bargaining power of a large company to lower costs might make the business model of their suppliers unsustainable. This insight forces companies to leave enough value for their partners. Companies should seek to eliminate the divide between economic and social concerns, and focus on creating value for all participants [61].

Value in software ecosystems does not have to be monetary. Innovation is one possibility: the concept of "open innovation" [9] sees outsiders as both a source of new ideas and a means to commercialise them. Research by Von Hippel [33] has even shown that the notion that software is developed by manufacturers and used by consumers is outdated, and lead users can be involved the development of innovative new products and services. Distributing the costs of development over many companies is another option. One might for example desire to avoid incurring large costs for the development of generic customer-relationship management (CRM) functionality and opt for quickly integrating an existing CRM-solution by another party. This lowers the (potential) revenue for the company as customers shift payments the other company to use its CRM-product, but also decreases the cost and risks of development, and shortens the time-to-market. No effective overview of the available forms of value in software ecosystems exists, and this is one of the primary triggers (section 2.4) of this research project.

Research by Lakhani and Von Hippel [48] has shown that participants in software ecosystems are in fact motivated by value. In their research on the motivation of participants in a open source software ecosystem to contribute to the seemingly menial task of answering field-support[1] questions, Lakhani and Von Hippel have shown that the participants perceive a benefit to them in having an active forum filled with information on how to deal with common problems, improve their own work, etc.; and are eager to contribute to this community of practice on their area of expertise to improve the health of the community, even though this benefits them only indirectly.

[1] Field-support is the answering of entry-level questions of end-users and helping them diagnose problems, which - strictly speaking - does not contribute to the community as a whole, as these answers could be readily found in the existing documentation. It is considered "mundane but necessary" work. [48]

2.2.4 Software ecosystem strategy

Adopting an explicit focus on its software ecosystem requires decisions by the company as to which parts of its product(s) are opened up to other parties and to what extent. [43]. Opening a portion of the company can create new opportunities for other parties in the software ecosystem. The Open Software Enterprise Model [45] may be used to guide these decisions. The openness decisions determine the activities, guidelines and standards set by the company in its software ecosystem. Additionally, the company must develop a business model and a strategy to remain profitable in its software ecosystem. [43]

The role of a company influences which strategies and forms of value are available to it. In selecting strategies, companies must not only focus on their own profitability, but also consider the short-term and long-term health of the ecosystem and their position in it. Companies that do not consider the health of their ecosystem in their actions, especially those in hub positions, risk extracting too much value from their ecosystem and ultimately leave it barren. [37]

The choice of appropriate strategy is also dependent on the forms of value that are of interest to the company. Research by Boudreau on fostering innovation [9] indicates that supporting open innovation is primarily useful when the wishes of the final consumers are unclear and thus the risk is relatively high. Boudreau defines two major approaches to software ecosystems: collaborative communities or competitive markets. A prime example of the former is the Linux platform to which innovations are contributed for the benefit of all, and the latter the various game platforms (Xbox, Playstation, Wii) which feature strong competition between the studios that develop games, but who all ultimately benefit from shared innovations as these tend to increase the competitive strength of their platform of choice. The strategy selected varies by the type of ecosystem. This is in line with findings by Bosch [8] encoded in the three-layer product model (3LPM). The 3LPM defines three approaches (called "layers") to software product lines or software ecosystems. The first approach "innovation and experimentation layer" is for products or functionality that are unknown and constitute a high risk. The appropriate approach is to optimise the number of experiments done with partners and consumers to greatly increase the rate of innovation. Over time, functionality migrates downwards to the "differentiating functionality layer" as functionality becomes core to a software product and controlling it provides a competitive advantage to the company. In this type of ecosystem, the functionality must be protected and hurdles erected to counter the threat of new entrants, and the focus of the company changes to supply maximum value to its consumers. Ultimately, the functionality transitions downwards again to the final "commoditized functionality" layer at which point the functionality has become so commonly available that having it is no longer a source of competitive advantage to the company. A good example of this are operating systems, which are required by almost every software product on the market but confer no advantage on the supplier of said product. The appropriate strategy changes once again to minimise the total cost of ownership of the technology.

Berk, Jansen and Luinenburg [5] have developed a software ecosystem strategy assessment model, which provides an overview of various aspects of strategy in software ecosystems. "Sharing" is included as a form of "Lifestyle" on a tactical level, but it it only defined as the dissemination of knowledge. While we recognize that this is quite probably one of the most important forms of value in a software ecosystem, we feel this is ultimately too narrow a view. Value sharing should in our opinion be considered on both a tactical and strategic level. This is another direct trigger for this research project.

2.3 Conceptual model

Figure 2.1 describes our view of the concept of value in software ecosystems. The most important related concepts, apart from value and the software ecosystem themselves, are actors, their role(s) and the strategies they employ. We use an informal syntax to draw this conceptual model, though the notations used should be familiar to those well versed in computer science. Rectangles depict separate concepts. The area of a rectangle does not indicate the relative importance of a concept. Labeled lines depict the key relations between concepts. Open arrows are used to describe an inheritance ("is-a") relationship between concepts to add specialisation where needed.

We define an actor as any party that operates in the software ecosystem in some capacity and has an impact on the software ecosystem. These actors could be anything from a lone developer constructing a open-source module, to a blogger describing a novel approach to a widespread problem, to large multinational companies that build and deploy a new marketplace for applications. In this project, we focus primarily on software companies or their divisions; large entities that have a defined strategy and some semblance of rationality to their actions.

Actors fulfill a role that describes their positioning in the software ecosystem. We adopt the ecosystem roles of Iansiti and Levien [37] but consider them separate from the actor. As described by Iansiti and Levien, an actor can only hold one role at any point in time in a single software ecosystem, but an actor is not its role. The actor can act out different roles in separate software ecosystems at the same time; neither is its role static: through the application of strategy or changing circumstances, an actor might change its role in the software ecosystem. In the latter case the actor remains the same entity in the software ecosystem. For these reasons, we model the role explicitly distinct from the entity that holds it.

Actors define and execute strategies in the software ecosystem, aimed at improving their situation. The importance of strategy in both business and software ecosystems is corroborated by Porter in his quote "the essence of formulating competitive strategy is relating a company to its environment" [60]. This is especially apparent in the study of software ecosystems, which is based largely on the premise that a software company is better off if

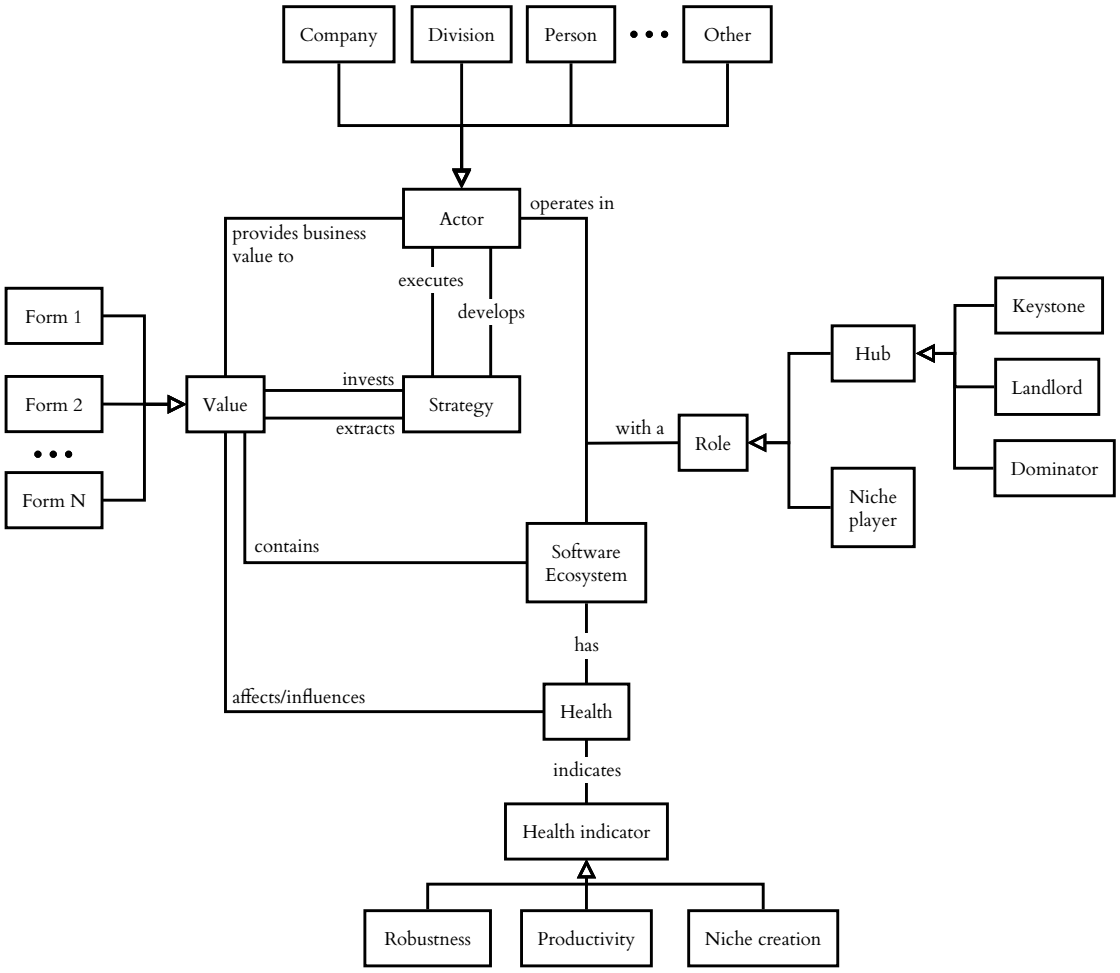

Figure 2.1: Conceptual model of value in software ecosystems

it considers its surrounding software ecosystem explicitly rather than implicitly. Strategy is "the determination of the basic long-term goals of an enterprise, and the adoption of courses of action and the allocation of resources necessary for carrying out these goals" [15]. We consider strategy in this conceptual model to be a coherent plan, aimed at achieving a predetermined goal for the company or entity that defines and executes it. Porter's view of strategy as a "broad formula for how a business is going to compete, what its goals should be, and what policies will be needed to carry out those goals [...] combination of the ends (goals) for which the firm is striving and the means (policies) by which it is seeking to get there" [60], is overly broad for our purpose: we only seek to consider explicit plans that are directly applicable to the software ecosystem and provide some kind of tangible effect, i.e. Porter's "means (policies) by which it is seeking to get there". In this sense, our concept of strategy is more closely related to tactics than grand strategy. Neither do we consider the reaction to unexpected changing circumstances without a predetermined plan as strategy. We're looking for plans, play-books, etc. Taking Mintzberg, Ahlstrand and Lampel's view [53] of strategy as either a plan, a pattern, a position, a ploy or a perspective; we consider strategy only as a plan or ploy, the most concrete instances of strategy that yield a direct impact on the situation of the company.

Strategies wittingly or unwittingly impact the software ecosystem by moving the value in it. In a broad sense, strategies can either extract value from a software ecosystem or invest it. In extraction, value is removed from the software ecosystem as a whole and captured by the actor that executes the related strategy, denying access to it by other parties. Investment is the opposite of extraction, in which an actor applies a strategy that releases some form of value to the software ecosystem, enabling its usage or capture by other parties in the software ecosystem. In this sense, specific instances and forms of value continuously flow between two mutually exclusive states of an internal form in a specific actor or set of actors, and a state of being freely available in the software ecosystem as a "common good". A form of value in the software ecosystem is always a business value to some party in the ecosystem, i.e. there has to be at least a single entity in the software ecosystem which is interested in this specific form of value as a major or lasting advantage to its business. This does not necessarily have to be the same party that applies the strategy which affects this form of value. Actors can release value that they do not value highly (pun intended) and release it to the ecosystem at large to create new opportunities for other players that do consider this form of value as essential to their business and/or consider it a good new opportunity. The area of value in this conceptual model is the focus of the first part of our research project.

One of the primary considerations in the definition and execution of strategy in the software ecosystem is the effect of said strategy on the health of the ecosystem. The impact of investment and extraction of value is expressed in terms of the software ecosystem health indicators of Iansiti and Levien [37], of which there are three. Naturally, executing strategy also affects the bottom line of the company that does so, but this effect is much more clear and has been extensively studied in other scientific fields. We will not consider it further in this study.

As for the central concept of software ecosystem, we adhere to the previously mentioned definition of Jansen, Brinkkemper and Finkelstein of a software ecosystem as "a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts." [43]. In this conceptual model, we take a predominantly economic view of the software ecosystem, focusing more on the common market rather than the common technological platform. The phrase "exchange of information, resources and artifacts" in this definition supports our central tenet that there exist several distinct forms of value in a software ecosystem and these can be moved inside of the software ecosystem.

2.4 Research triggers

From the related work and the conceptual model, a number of problems and deficiencies become apparent. First and most important, no formal definition exists for the concept of value in software ecosystems. Given the central position of this concept in the conceptual model and in our research project, it is crucial that this definition be created and validated. In tandem, we have no clear overview of forms of value in software ecosystems. As depicted in Table 3.1, one can easily come up with a few likely candidates but this is neither a comprehensive nor validated overview of how the concept of value in software ecosystems is applied in practice. Not having this definition and overview is an important gap in our understanding. Creating this definition and overview yields new questions that touch upon the usage of value in software ecosystems. While research on strategy in software ecosystems exists, it does not relate directly to the concept and forms of value. This gap needs to be addressed. These gaps are the primary triggers for our research project which the authors seek to cover with the research questions.

2.5 Research questions

Based on the theoretical background and triggers for this project, four main research questions are identified:

RQ1. How can we define value in software ecosystem?

RQ2. What forms of value exist in a software ecosystem?

RQ3. How can value be extracted from a software ecosystem?

These research questions are designed to meet the problems posed by the research triggers. The research questions therefore map directly onto the conceptual model (Figure 2.1). In

Figure 2.2 the conceptual model is overlaid with the areas of application of the main research questions of this project. These indications in red are the areas of our conceptual model that are impacted by the research questions, showing how the model and our knowledge of its concepts is expanded by this project. The first research question RQ1 yields a definition of the concept of value in software ecosystems. Such a definition does not yet exist and is needed to improve our understanding of the concept. Research question RQ2 expands our knowledge of the various forms of value in software ecosystems. This list is important to gaining an understanding how value is applied in practice. Finally research question RQ3 improves the scientific understanding of how strategy is created and applied in the context of software ecosystems, and how this impacts the dissemination of value in software ecosystems.

A fourth research question was originally included as "How much value can be extracted without damaging the health of the software ecosystem?". Unfortunately there was not enough time for a proper in-depth investigation and the research question has been dropped from the project. The concept of health in software ecosystems, appear in a somewhat diminished form in the answers to RQ3, specifically the work on participation in plays (chapter 6).

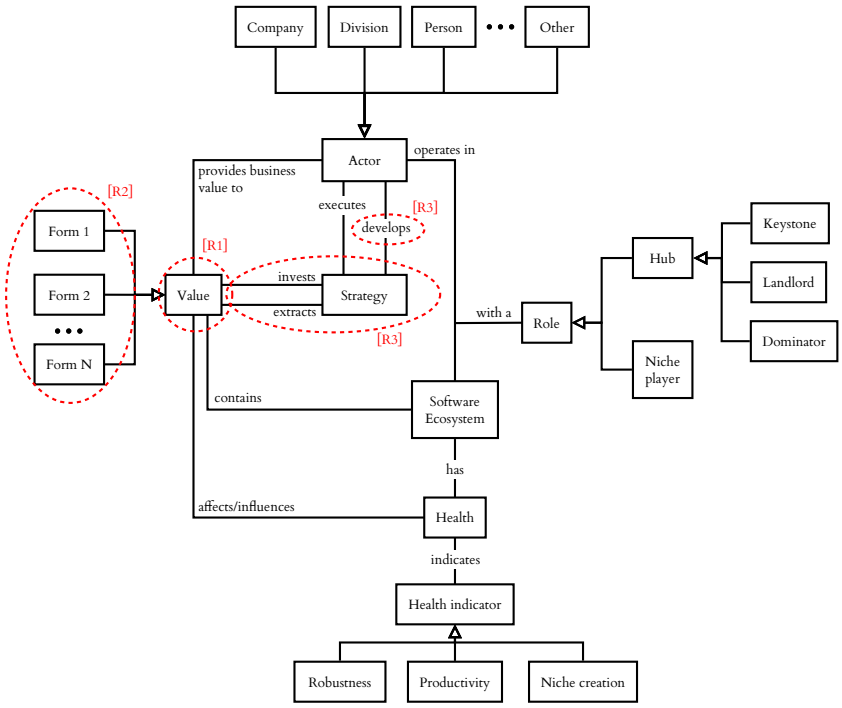

Figure 2.2: Conceptual model of value in software ecosystems, overlaid with the research questions of the research project

2.6 Conclusion

This chapter introduced the key elements of existing research, relevant to our research project. Described are the needed definitions and descriptions of software ecosystems, company roles (hubs, keystones, dominators, landlords and niche players), health indicators (robustness, productivity and niche creation), value in ecosystems and the strategies. This is developed in a conceptual model depicting software ecosystems, actors and their roles in them, health issues, value and its forms and the strategy that affects all by shifting value around, and the key relationships between these concepts. The main conclusion is that companies must pay explicit attention to their positioning in and the health of their software ecosystem, and their strategies must balance their interests with those of others in the ecosystem. This guiding principle (while obvious) is important to our research project. The triggers for research are identified in the conceptual model and the four main research questions are posed upon these gaps.

Chapter 3

Research approach

3.1 Introduction

This research project seeks to first improve our comprehension of value in software ecosystems. We research how value is invested in and extracted from software ecosystems, and provide a new way of modeling value exchanges between companies. The project is organised in several distinct phases. The complete approach is described and the main phases depicted using process-deliverable diagrams (3.2). Challenges to the successful completion of the project are identified (3.3) and the validity of this approach is discussed and substantiated (3.4).

3.2 Research process

The research project is executed in three consecutive phases. To build an thorough understanding of the topic, a structured literature review (SLR) is conducted. This SLR seeks to answer the first research question on the definition of value in software ecosystems and the second research question on the forms of value. Based on existing literature, a number of strategies are selected and used to develop a new artifact to express the concept of moving value within a software ecosystem. Finally, this artifact is validated using real-life case studies based on interviews with key employees at companies in related software ecosystems.

Confronting the first research question, no generally accepted definition of value in software ecosystems exists. We need to create this definition. Suitable definitions do exist for both software ecosystems and (business) value. For the definition of value, we use the short version given as “any desirable result for stakeholder in a context” [13]. A stakeholder in the context of software ecosystems is interpreted as any company that is directly or indirectly affected

by actions taken by the company that is the object of study. Note that this does not preclude competition between companies in the ecosystem. The definition of a software ecosystem we adopt by Jansen et al. [43] is discussed in section 2.2.

Based on a preliminary brainstorm, we have identified a potential set of forms of value in a software ecosystem as a starting point for further research (Table 3.1). These values include concepts like increasing the robustness of the ecosystem, an increase in productivity, fostering innovation, lowering development costs, revenue, increased brand recognition and improved developer relationships. This list of forms of value is by no means exhaustive or conclusive, and provides only a starting point for the SLR.

	Keystones	Landlords	Dominators	Niche players
defensive strength	✓	✓	✓	
increased productivity	✓			✓
increase innovation	✓			✓
lowered development costs	✓	✓	✓	✓
money/fees		✓	✓	
brand recognition	✓			✓

Table 3.1: Concepts of value with interested roles [37] in a software ecosystem

Different forms of value are of interest to different roles in the ecosystem. A company that commands a dominator role in the software ecosystem will have a vested interest in tremendously increasing the defensive strength of the ecosystem to decrease the threat of new entrants, even to the point of deliberately hindering innovation. A niche player that is active in multiple software ecosystems will in contrast be much less reliant on the continued existence of a single ecosystem and may subsequently prefer a more hospitable strategy in the software ecosystem in order to generate maximum innovation and profits. The possible sources of value are laid out in Table 3.1 against the commonly-used ecosystem roles [37]. It is clear from the entries in the table that the interests of companies playing different roles are almost mutually exclusive. Entries in this table are in line with previous findings [37] that successful execution of their strategy by landlords and dominators might harm the interests of niche players, who are dependent on other forms of value that are marginalised by the dominating parties.

The major activities of each phase of the project are depicted as method fragments in process-deliverable diagrams [75] in Figures 3.1, 3.2 and 3.3. Most of the method fragments describe basic processes well known in academia that are not specific to this project. The associated tables with in-depth description of the artifacts and activities are therefore omitted for brevity as these add very little. In addition, connections between artifacts are not labeled in some cases where their interpretation is obvious.

The SLR is executed (Figure 3.1) based on the process as described by Kitchenham [47]. While the need for a SLR is obvious from section 2.4, it is appropriate to explicitly report on this need, including the formal criteria as specified by Kitchenham [47]. After developing a review protocol (which will be peer reviewed), we proceed with gathering the appropriate sources in accordance with it. If needed, the selection criteria will be adapted as described in section 3.3. Afterwards, we report of the results of the SLR based on which a valid definition of the concept of value in software ecosystems is constructed, and the main forms of value identified.

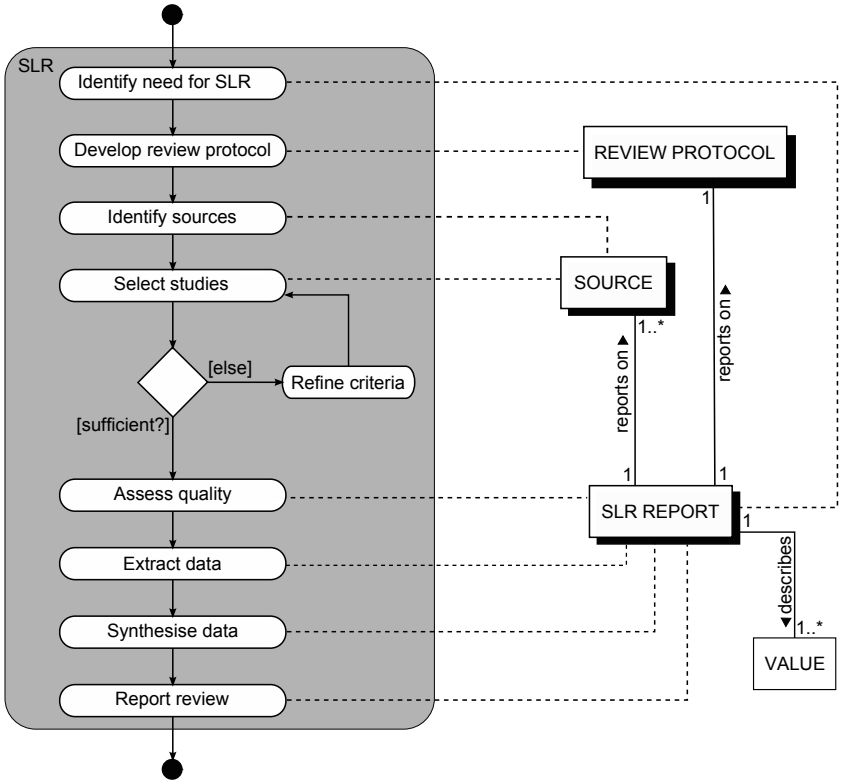

Figure 3.1: Process-deliverable diagram of the first phase (SLR) of the research project

In the second phase of the research project, we start by finding from literature various "strategies" that actors in software ecosystems can use. These strategies are adapted from seminal works on the topic such as Popp's Profit from Software Ecosystems [59] and Iansiti and Levien's book The Keystone Advantage [37], and the results of the previous SLR. From this list, a number of strategies are selected which are suitable to use for expressing the concept of exchanging value in a software ecosystem. For example, some strategies that involve only changes internal to a company, are to be excluded. The authors elaborate on these choices in chapter 5. Based on these selected strategies, the main artifact is developed and the strategies are modeled using the artifact. The resulting models are extensively discussed in this thesis.

The third phase consists of largely two steps: conducting interviews at companies in various

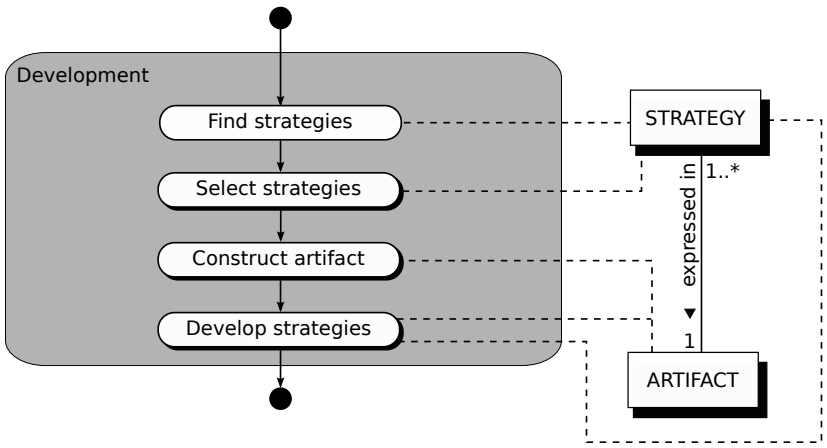

Figure 3.2: Process-deliverable diagram of the second phase (developing artifact) of the research project

software ecosystems, and expressing their situation using the same models used to describe the theoretical scenario's. The interviews are conducted in a semi-structured fashion. The sub-activities are based primarily on the process described by Hove & Anda [35]. Two activities prescribed by Hove & Anda as these are not relevant to this research project. One, the discussions/meetings-activity is not relevant because the interviews are done by a single researcher. Two, transcribing of interviews is excluded because the expected benefits do not outweigh the costs. A single hour of interview can take up to eight hours to fully transcribe [35]. Neither will we analyse the answers in-depth using textual analysis. All interviews will be fully recorded to preserve a proper chain of evidence. Interviewees are asked for their permission prior to starting the interview, and all interviewees gave their explicit consent. After starting the recording, this question and the consent were repeated to be recorded.

Selecting companies is done based on a few hard criteria which are elaborated on in chapter 7, which also contains the interview protocol. Some background information is collected on companies, but this is a limited activity such as studying the website of the company. This is mostly done to ease the conduction of the interview by gathering common knowledge such as company size, main products, etc. so that no time has to be wasted in the interviews on gathering information that is readily available in public sources. The situation of every company is subsequently modeled using the artifact. The resulting model is sent to the interviewee with a few probing questions for validation.

The first phase of the research project provides the answer to the first and second research questions, while the second and third phase uncover the answer to the final research question.

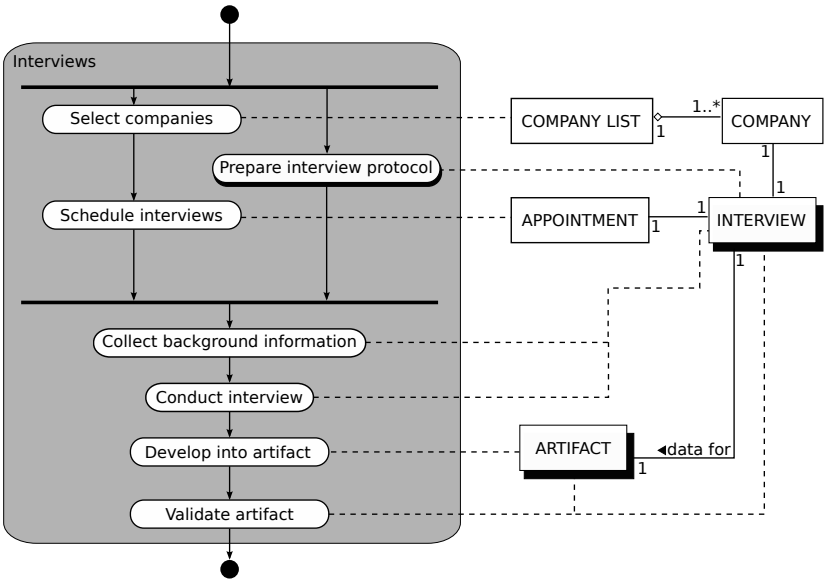

Figure 3.3: Process-deliverable diagram of the third phase (interviews) of the research project

3.3 Challenges

A number of challenges have been identified that constitute realistic threats to the successful completion of the project. We list these issues in this section and discuss the measures implemented for mitigating the effects or preventing these problems altogether.

The main threat to the first step, the SLR, is a potential lack of reliable, published scientific research on the subject. Value (extraction) in software ecosystems is a relatively sparsely populated subject, as a quick search using Google Scholar yielded very little results. We rise to this challenge in two ways. If needed, we adapt the SLR to consider literature on value and value extraction in other types of communities, including but not limited to communities of practice, business ecosystems and (online) social networks. In tandem, we expand the SLR to include non-scientific sources of information such as business publications, whitepapers, press releases, etc. though we should be mindful of and explicitly report on any quality issues. We expect these measures to adequately mitigate the risk of insufficient breadth. While the creation of a definition of value in software ecosystems that is embedded in sufficient literature is strongly desirable, it is not strictly necessary for this research project to proceed. If all else fails, we will construct a usable definition based our own experience with existing studies in this field and subject this definition to expert review. Finally, we aim to use the structured interviews later in the process to both validate and extend the list of forms of value found in the SLR. It is therefore not necessary to create an exhaustive list of forms of value based on the SLR alone.

The challenges in the interviews lie primarily in the reliance on sufficient interest and coop-

eration at the companies. Companies might not be aware of the need of paying attention to their software ecosystem at a sufficiently high level in the company's hierarchy and subsequently not willing to cooperate in the study. We aim to prevent this by relying on warm introductions by dr. Slinger Jansen, Martijn Feekes and other professors/experts, though we are mindful of the selection bias this may incur on the participants. For example, companies who have worked with dr. Slinger Jansen on research in software ecosystems can be expected to have a higher than average understanding of their surrounding ecosystem. A second problem we expect to encounter at case study companies is based on existing research in the field of software ecosystems. Existing literature [5, 31] suggests that most work in this field is based on the financial data of companies. In the latter half of this study, we focus on the strategic aspect of value extraction. These two subjects are both possibly considered confidential by companies, especially if they are publicly traded, or we touch on strategies that have been developed but not yet publicly adopted. We prevent this second problem by explicitly offering to make participating companies anonymous if desired and only publish company information relevant to the study such as the number of employees, turnover, markets, etc. as needed for a clear understanding of its situation.

3.4 Validity

This research project suffers from a few specific concerns related to the validity of results. Academic rigor is applied in all steps by adhering to known best practices in conducting science [35, 47, 76]. To preserve the chain of evidence from the interviews to our analysis, we will ask for permission to record the interviews and we will provide the source material to other researchers upon request.

A number of specific challenges affect the development of the artifact. It might become apparent from the interviews in phase two that the interviewed companies do not execute a pre-defined strategy that is at least closely associated with value extraction in software ecosystems. In this case, we also expect these companies to not have a articulated expectation of value in their software ecosystem, something which would quickly be obvious from the interview. We will adapt as needed, but have no specific strategy in place to mitigate this risk. Second, a potential result of the interviews might be that company strategy related to value extraction in software ecosystems is too dependent on the specific situation of the company and its software ecosystem, with no common themes to be identified. We expect this to be a highly improbable scenario. Nevertheless, if such are our findings, we revert our goal of building a concise model to providing a descriptive overview of current practices in the field. The main challenges in the developing the artifact from existing strategies are twofold: not having a separate test-set and basing the strategies of literature. The main artifact will be constructed based on the set of existing strategies that are retrieved from existing work by relatively famous authors. This approach makes the artifact capable of describing the strategies on which it was based, but it does not imply that the artifact will be able to be properly used on strategies found in other research, or developed in practice. We mitigate

this risk by using three separate sources for strategies to increase the variability found in this set of strategies. This does not mitigate the second concern: basing the strategies only on scientific literature. This validity concern is reduced by the combination with the third phase of the project, in which the same artifact is applied to describe real-world situations of actual software companies.

The main concern for the interviews is the introduction of selection bias, introduced by relying on warm introductions by dr. Slinger Jansen. For example, companies who have worked with dr. Jansen in the past can be expected to have a better than average grasp of their surrounding ecosystem, and thus subsequently employ better strategies in dealing with them. This bias is somewhat lowered by including some companies referred by other colleagues, but cannot be completely excluded.

3.5 Conclusion

The research approach described in this chapter provides a solid basis for the execution of the project. Special care is taken to reduce specific biases. The four phases of the project are based on scientific best practices and validation of created artifacts is included at every step of the way. Challenges to successful completion are identified and specific strategies employed to mitigate their risk to the project.

Chapter 4

Structured literature review

4.1 Introduction

This chapter depicts the structured literature review (SLR) and its results. The first three sections comprise the parts of the review protocol as described by Kitchenham [47]; specifically the rationale for the SLR (4.2), the main questions and related viewpoints (4.3), and the strategies and criteria used (4.4). The review protocol has been peer-reviewed by our colleagues and adjustments have been made based on their feedback. The final sections of this chapter report on the results of the SLR (4.6) and the analysis (4.7). These results serve as the answer to the research questions RQ1 and RQ2 (4.8).

4.2 Rationale

It is clear from existing research on the subject, as discussed in section 2.2.3, that no solid definition of value in software ecosystems is available. To answer the first research question RQ1 of this project, such a definition must be created. Neither exists a clear and exhaustive overview of forms of value in software ecosystems exists. This overview is created to answer the second research question RQ2. This SLR is the first step in closing this research gap. If we relate this reason to the reasons defined by Kitchenham [47] it is most closely related to "provide a framework/background in order to appropriately position new research activities." and to a lesser extent to "assist in the generation of new hypotheses".

The review must be conducted in a *structured* way. A new definition should not be biased by our personal experience. Additionally, the review must be conducted as exhaustive as possible given the constraints of our project to generate a complete and unbiased overview [47].

4.3 Questions

The most important part of a SLR, apart from conducting it in a structured fashion, is asking the right questions. The questions should be relevant to both academia and practice, and feasible [47]. In the case of this specific SLR, this presents somewhat of a problem. We can not search directly for instances of value in software ecosystems as the research gap has shown that such studies do not exist. The concept of value is however closely related to "business value". We can thus study the business models of product software companies and construct the values from them. The questions for the SLR are thus defined as follows:

1. Which business models exist for product software companies?

2. Which factors are identified that influence model performance?

Kitchenham [47] describes valid questions for SLRs from three complementary viewpoints: population, interventions and outcomes. While the viewpoints are not directly applicable to our study, we can view the questions through this lens. The population in the SLR consists of any product software company. The adoption of a new model could be considered the intervention, though we would be hard-pressed to find consistent before-after studies as changes in business models are often coupled with the introduction of new products, services or markets. The outcomes are improvements in the broadest sense to the situation of the company, be it their strategic positioning, revenue, profits or brand recognition.

4.4 Approach

The SLR is conducted exclusively through CiteSeerX[1], despite the prevalent scientific opinion against the practice [7, 26, 39, 47, 55], though not all studies and researchers agree [24, 55].

The criticism of using a single search engine is largely aimed at the suspected search engine framing and reproducibility, poor quality controls and the reliance on a single source. The first concern seems based on only speculation and the unsubstantiated assumption that all search engines operate as Google does. Despite proper testing using different PCs, ISPs and user accounts, any form of search engine framing by CiteSeerX could not be reproduced by the authors. The second concern is valid in our opinion, though solid evidence to support it is lacking and the criticism is mostly leveled at methodological errors in studies in favour of using Google Scholar. We choose to disregard this concern as the potential problems do not outweigh the speed with which we require to execute the SLR. The third concern is not a valid concern in our opinion as the search includes results from multiple authors, countries, affiliations, companies, etc.

[1] `http://citeseerx.ist.psu.edu/`

We have constructed a small tool that allows us to quickly scrape a number of publications based on specific searches, with title, abstract and number of citations included. The search terms and subsequent results are included in Table 4.1.

We then apply some hard criteria. All hits that are not a published, scientific paper, report or thesis are rejected. All publications which have no citations are rejected too. The title and abstract of each paper is than read and judged for inclusion based on the contents. The publications that remain are subsequently read in full. If a publication at this point is not considered relevant or the methodology judged unsound, it is removed from the SLR.

4.5 Limitations of the approach

This SLR is not perfect, and four limitations have been identified.

The main limitation is the inherent bias in the selection criteria. To limit the result set to top quality sources, the results of the scraper are ordered by citation count. As the used search engine returns only the top 500 results for any query, this automatically serves to exclude less cited studies from the SLR. However, citation counts are primarily a measure of impact, not quality of a given publications. Additionally, this approach tends to exclude newer studies with potentially more recent insights in favour of the 'classics' of the field. Additionally, we have not corrected for self-citations, papers published by authors who refer to their own previous work. A manual inspection of a random sample (n=20) of the publications was conducted and concluded that this problem is probably not an issue as it appears very little in our data-set.

Three additional limitations have become apparent from the inspection of the results. It is hard to adopt a consistent, clear and complete definition of a business model. Various studies have differing opinions on what elements are included and excluded, and the wording is making things worse. Some papers indicate a strategy to enter a new market as a business model, while others discuss price discrimination strategies and name that as a business model. Ultimately, we have identified four common elements that are broadly accepted and which are discussed in the results.

A lot of early stage e-commerce business models we encountered in the SLR are basic adaptations of existing, pre-digital business model with superficial changes, described in research papers mainly concerned with improved logistics, made-to-order products and increased price competition. These business models are largely not applicable to the software industry, do not yield any meaningful contribution to our study and are thus excluded from the analysis.

Finally it is very tough to find a set of descriptions of business models. Nearly every author seem more than happy to talk about "business model-models", and tooling to design business

models is abundant and omni-present with the Board of Innovation[2] and Business Model Canvas[3] as the most well-known examples. The simple descriptions of business models are a lot less common. Due to this effect, the analysis of business models is limited to bits and pieces retrieved from other studies, and it is hard to paint a complete overview of current practices in the field.

4.6 Results

The selection criteria used in the SLR (see 4.4) result in the exclusion of a number of studies from our initial set of publications. The numerical effects of these filters are described in Table 4.1. The table lists the total number of hits on a given query and how many hits were retrieved (title & abstract). The third column "exclude technical" lists the number of hits rejected for technical reasons. There are three reasons entries can be rejected on technical reasons: it is a duplicate entry that appears under multiple keywords, it is not a published scientific paper, or it has never been cited. The next column "exclude contents" counts all papers which are rejected as not relevant to our study upon reading. The final column lists the total count of papers which were included in the SLR. Note that the results do not cite all 123 papers, as we only report on the most significant findings in the SLR.

Search query	*Hits*	*Retrieved*	*Exclude technical*	*Exclude contents*	*Included*
"business model" software	29.812	500	1	444	55
"software ecosystem" "business model"	171	171	152	7	12
"software ecosystem" strategy	382	382	325	44	13
company platform "business model"	178.368	500	22	446	32
company platform strategy	537.907	500	39	450	11
Totals:	746.640	2.053	539	1.391	123

Table 4.1: Search keywords in the SLR with number of results and inclusion/exclusion counts

A number of effects are immediately apparent from reading the included studies. First of all a large disagreement in literature on what a business model is, and what it is comprised of. A large variation of different approaches and definitions is encountered. Value propositions, specific go-to-market strategies and organisational values are all referred to as "business models". Ultimately a widely accepted definition is not found, neither a sliver of consensus on the subject. However through the SLR we have identified four main elements that are most often referred to as central tenets of a business model. These elements are best framed as questions and are each discussed in turn below: what is sold? how is it sold? how is it paid for? and what affects the performance of the model? The former three elements relate closely to

[2] `http://www.boardofinnovation.com/`

[3] `http://www.businessmodelgeneration.com/`

the first research question of the SLR (which business models exist?) while the latter relates to the second question (what affects the performance?).

4.6.1 What is sold?

What is being sold comprises the nature of products changing hands and services being delivered. It is naturally the category with the largest diversity. However, even with the wide variety found in the studies studied, a number of clusters can be identified which are relevant to a (SaaS) software ecosystem.

Starting from simply selling a product or service [1, 6, 16, 18, 25, 28, 38, 44, 49, 51, 52, 63, 66–68, 71], a number of papers cite added products or services as a potential source of sales. For example consulting work on implementations of product software [1, 16, 18, 21, 40], with outsourcing tangibly related [29, 30, 44, 52], or even selling improved software on hardware built by direct competitors [16]. Additionally plugins, mods and extensions are well-known and widely used [40, 41, 44, 57]. This serves as a strong indicator that complementary opportunities are important in building succesful sofware. However, some parties prefer to erect some barriers to cooperation, employing paid access to APIs, source code or documentation [38, 40] and preferred partners, often in a tiered system [40]. This creates a direct revenue stream in lieu of reduced entrance of new partners.

Online marketplaces and platforms have an increased role due to lowering transactions costs, resulting in some novel approaches such as vulnerability marketplaces [2], reselling user-generated content [22], virtual gifts exchanged [54] or companies that aggregate pricing information of various markets and suppliers [34].

Finally, we are left with a significant group of high-level strategies we collectively refer to as "open source-plays". Companies have various reasons to participate in open source projects or even open up parts of their codebase, such as adopting an open source-freemium model [72] with an open source, limited (both in license and features) product often referred to as a "community edition" and a premium offering for commercial usage. It could be applied to deny new entrants access to the market by offering feature-rich software for free [11]. The model is equally applicable to services [23, 32, 73], offering premium support, consultancy and implementations of their open source product. These strategies can even be pursued over different stack levels. The software package is released as open open source, increasing its adoption compared to priced offerings. Said software subsequently performs best on hardware sold by the company, boosting revenue from hardware sales. Companies might seek to lower costs by outsourcing some of the development to the greater community [23], using bug bounties [4] or public roadmap funding [23, 40] to steer development. In a reversal of this model, the company accepts donations from the community to fund its contributions to an open source project [23].

4.6.2 How is it sold?

How it is sold concerns the various methods by which the products and services above are sold to end-users, or how the market on which they are traded functions. A number of variations are apparent from the SLR.

The payer does not have to be one and the same with the end-user. A company can adapt and change the payer by for example including advertising [14, 18, 22, 29, 51, 54, 58, 62, 65, 69, 78] and reselling data of end-users to third parties [22, 51] resulting in a free or cheaper product for end-users.

SaaS, and the variants PaaS and IaaS, are the dominant delivery model for software with respect to the scientific literature [6, 18, 27, 44, 51, 52, 66–69]. SaaS allows companies to fund IT as an operating expense instead of a capital investment. Ultimately in the fullness of time this might result in the promise of "IT as a utility" [14, 67] in which IT resources and software are delivered in a way similar to electric power is.

Finally, a number of publications write on the adoption of new markets and platforms. In these scenarios, companies build a (N-sided market) platform [14, 38, 41, 64, 69, 77], or create a new clearing house [2, 14, 19, 28, 34, 40, 51, 56, 65, 70] for either a new market or an existing market that benefits from improvement in various ways, and capture a share of the traded value in exchange. Auctions are a closely related concept, though mostly adopted in e-business and advertising [14, 18, 22, 29, 51, 54, 58, 62, 65, 69, 78]. Customers might also band together to group-buying concepts [14, 25, 46, 56] in which customers but in bulk to profit from volume discounts.

4.6.3 How is it paid for?

Value that is spread to customers and end-users nearly always has a return component. While this is usually a form of monetary compensation, especially in simpler business models, other models exist which alter the exchange. In the result set of this SLR, these adaptations either delay the pricing or change it.

Delaying pricing reduces the profit from the initial value exchange, in exchange for (the expectation of) higher future profits. Examples are the razor-blade model [16, 64] and the freemium model [17, 18, 28, 32, 51], the former selling the initial artifact at a price lower than the market will bear. The latter offers a less powerful version of the product, in hope of enticing companies to upgrade later when they have bought into the value proposition of the product or service. In more extreme cases, it is a "loss leader" [64] in which the initial value is exchanged at a loss, forcing the company to recoup it through later transactions.

Pricing is changed when either the payer or the payee of the money flow is adapted. The prime example, advertising [14, 18, 22, 29, 51, 54, 58, 62, 65, 69, 78] is one of the main

pillars upon which IT is built, especially when sold to consumers as opposed to companies. This can also include reselling usage data to advertisers [22, 51]. It allows for free delivery of value to end-users in exchange for their (presumed) attention to advertisements. An alternative change is bundling [6, 18, 25, 28] in which the product is combined with other complementary products and sold as one package. A bundle might be offered for a reduced price, some products being offered cheaper and subsidised by its complements [28]. In some cases unbundling a monolithic product might also be favourable [72]. The payee is changed much less often than the payer, as any company that is not a charity will need to reap some benefit from delivering its products. Adopting a kickback-scheme [49] is such an option, paying a fee to a third-party for delivering sales opportunities.

4.6.4 What affects the performance?

A large number of factors can be identified which influence the performance of a business model. Some of those are obvious from the definition of the business model itself such as the performance of a vendor lock-in play being affected by the incompatibilities with competing products. Some factors are more broadly applicable and a number of results from the SLR will be discussed here. As there are uncountable variants of business models, each having one or more properties that affect its performance, there is too an inexhaustible supply of performance factors. We naturally do not endeavour to create an even remotely complete list.

First up are the direct financial incentives. Strong price competition [67, 70] erodes margins and might force companies towards alternative schemes such as freemium and loss-leader plays. Research shows that the decisions of buyers to invest in software are for example heavily influenced by potential indirect cost savings rather than price [6, 12].

Curated markets and N-sided market platforms are strongly affected by both the level of customer lock-in [6, 67, 72]. However, the customer' perception of the level of vendor lock-in reduces the inflow of new entrants [70, 72]. For solid performance, the products must further lock-in, but it mustn't look it. Related to these factors are the reduction of initial adoption problems [40] and low search costs for buyers [70], which both serve to improve the influx of new customers.

A large number of factors in the SLR deal with complementary opportunities. It is important for products to have many opportunities for complementary products [6, 65]. Adopting interoperable standards [6, 40, 72] might be a good way to stimulate this. Creating modular products may help too [63]. Network effects [6, 20] which serve to attract new customers, are strengthened by these factors, even more so when taking into account network externalities [72]. Congestion (reverse network effects) [6] can dampen the network effects when present, or mitigate them altogether.

4.7 Analysis

From the business models and factors (section 4.6), a number of clusters become apparent that could entice companies to join a software ecosystem. These clusters are displayed in figure 4.1. These clusters serve as the final answer to research question 2 (section 2.5).

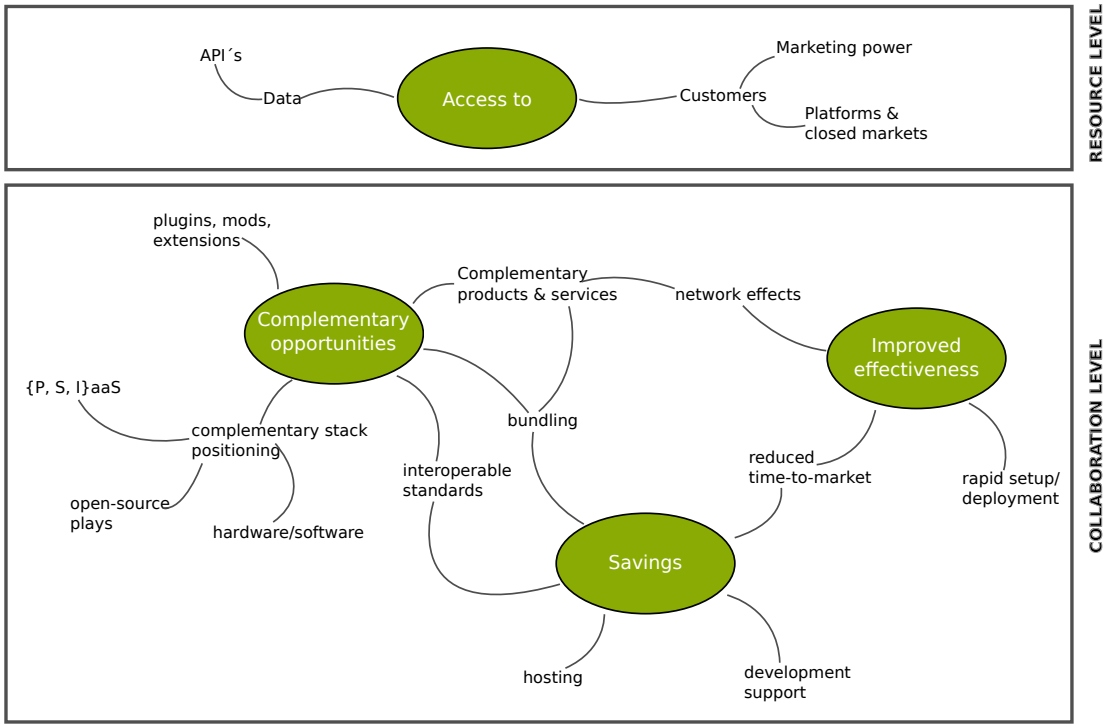

Figure 4.1: Main clusters of results from SLR

We divide the value in the software ecosystem in two separate levels: resource and collaboration. The resource level comprises reasoning in which joining a software ecosystem is not optional. A company (usually a niche player) that needs to use another company's (usually a hub) customer data or gain access to its customer base, has no choice but to join the ecosystem. The resulting relationship largely operates as a "two-sided market platform" [64]. Accepting the terms and conditions that the hub sets, is a *conditio sine qua non* for joining. The performance of this business model from the perspective of the host is mediated by the balance of negotiating power between companies. Having very useful data available raises the price the hub can charge, while having another suitable alternative available, lowers it dramatically. The joining companies essentially pay the hub for having access to its data, commonly through (metered) APIs, or alternatively pay for its market reach, essentially buying the right to contact and sell to the hub's customers. If the hub can allow partners to reach its customers in a way that is mutually beneficial, it can extract hefty fees from the other companies while preserving its relationship with its current customers.

The second "collaboration level" consists of the other forms of business value found in the SLR. These are the forms of business value with which a company might help to support other companies. This form of business value in a software ecosystem is more optional than those in the resource level in which participation is an obvious requirement for gaining access. Companies seeking value on the collaboration level can freely choose to engage in the ecosystem if it is beneficial to them. It usually employs a more tailored approach to the relationship, instead of the hub or platform setting the terms for participants indiscriminately. The forms of value are defined in three main opportunities for companies: savings, improved effectiveness or complementary opportunities, though all three clusters are closely related. Savings describe the business value that reduces the development costs or operating costs of partner companies. For example, a company with an API might develop a software development kit (SDK) for a programming language that is popular with its partner companies. An SDK helps partners with their development efforts by containing common, boilerplate setup logic and abstracting common functionality of the API. This reduces the development costs of using the API for partners, saving money while it might also contribute to a faster time-to-market of innovative solutions. This in turn makes the ecosystem more attractive to new partners as productivity is a primary indicator of software ecosystem health, attracting more partners to the more healthy ecosystem, strengthening network effects present in the ecosystem. The second cluster "improved effectiveness" is the natural opposite of savings, generate more revenue faster for partners, rather than saving them costs. The aforementioned SDK that helps to attract new partners also helps these partners to get setup faster and hit the ground running in development. This reinforces the network effects present even more. The final cluster "complementary opportunities" comprises all business value which enables partner companies to build complementary, not competing products, services and artifacts in the software ecosystem. This cluster ties back strongly to the other main health indicator of software ecosystems "niche creation". One prime example of a complementary opportunity is a complementary stack position between companies. A company that makes a product sold to consumer, end-users can collaborate more closely with a company that develops server hardware, an operating system or anything else of the myriad of technologies that are involved in delivering the final user-facing product to the end-user, without becoming competitors. The same principle applies to makers of complementary products and services that offer more benefit from integration than direct competition. An example of the former is the case of an monolithic enterprise resource planning system, involved in any and all details of the business that adopts it, compared to a suite of specialised products that operate in tandem, creating a tightly-integrated, custom solution for a customer that allows multiple companies to thrive albeit with a smaller slice of the complete pie for each. Companies can explicitly support these complementary opportunities by for example supporting plugins, extensions or mods[4] and essentially adopting an platform approach for its product. Finally open-source plays have been extensively discussed in section 4.6.1, and bundling and interoperable standards in section 4.6.4.

[4]Short for "modification", a term used specifically in gaming to describe third-party plugins that alter the aspects of a game to improve it or create a new game altogether.

Based on these clusters we can construct our definition of value in software ecosystems, answering research question 1 (section 2.5):

> *Value in software ecosystems is any product, service, artifact, improvement or right, which is enabled or exchanged through the software ecosystem and provides a tangible benefit to a participant.*

This definition closely covers all aspects we have identified in the analysis of the SLR. It considers the fact that the reason for joining a software ecosystem might be immaterial, such as having the right to market to a hub's customer base. Value can be exchanged between parties in software ecosystem, for example in the form of an artifact in the construction of an SDK. Value is not limited to being exchanged and can also be derived by a company itself, for example simply having more partners or customers on either side is of value to a N-sided platform, but this is not in the strict sense value that is provided by the specific partner as a party in the software ecosystem. The latter case is covered by the inclusion of "which is *enabled* [...] through the software ecosystem". Value does not need to reciprocated by the receiving party, but it does need to be of a concrete benefit (read: business value) to a party in the software ecosystem to be considered value. "Participant" is to be interpreted as "participant in the software ecosystem", but this extended description is not included as to not extend the definition needlessly.

The authors recognize that the inclusion of "artifact" in the definition is technically superfluous as artifacts could always be considered a "product" and vice-versa. However, we choose to make the distinction between products that are exchanged for payment to consumers or companies in a economic transaction, as opposed to artifacts which are usually distributed free from direct payment and which realise mostly indirect business value. Informally, read Microsoft Office and bananas as examples of what we consider to be products, and SDKs, marketing materials and data-sets as artifacts.

4.8 Conclusion

The results of the SLR serve as the answers to both research questions RQ1 and RQ2. Based a structured search through scientific literature, the main forms of value in software ecosystems are identified as the creation of complementary opportunities, realising savings in both time-to-market and costs, and other measures that make the business of partners more effective. If one party controls access to a needed resource such as access to existing customers or their data, these benefits are sidelined by the need to have access to that resource. Based on these SLR results, the definition of value in software ecosystems is constructed as "any product, service, artifact, improvement or right, which is enabled or exchanged through the software ecosystem and provides a tangible benefit to a participant".

Chapter 5

Value Exchange Graphs

5.1 Introduction

Confronting the third research question in this project (how can value be extracted from software ecosystems?) describes the way a company might capture value from its ecosystem. We adopt the viewpoint that value generally flows through a software ecosystem between the companies operating in it. A company might release example code of the intended usage of its APIs to teach partners the proper way to use the API. This provides partners with an improved understanding of the thinking behind the API, reducing the occurrence of potential issues in development. In return, the company supplying the API receives less support requests, reducing its operating costs. In this fashion the two partners help each other succeed. The business value, which is different for each partner, flows between them. Companies can influence how business value flows between them by adopting different business models, changing tactics or strategies in a coherent fashion. We refer to these actions as "plays". Variations of plays that do not change the general idea of a play but change its parameters, are named "scenarios".

In this chapter we show that the changing flow of business value when affected by a play, can be modeled in a visual way by drawing a Value Exchange Graph (VEGA). The chapter begins with depicting the formal meta-model for a VEGA (5.2). We describe three separate variants of software ecosystems and a set of plays that can be effective in changing the flow of business value in them (5.3). These plays are developed in VEGAs and the implications extensively discussed (5.3.1 through 5.3.9). Finally, we discuss the main limitations of applying VEGAs (5.3.10) and conclusions to the research questions (5.4).

5.2 VEGA meta-model

Value Exchange Graphs (VEGAs) are directed, unweighted graphs with actors as vertices and exchanges of business value as its edges. The VEGA is described in a formal meta-model depicted in Figure 5.1. Note that the meta-model does not prescribe a visual style for the elements of the diagram, which are left to the discretion of the reader. We adopt the visually simplest style that covers our use case, but other users may feel the need to add further distinctions such as borders around actors, etc. They are free to do so.

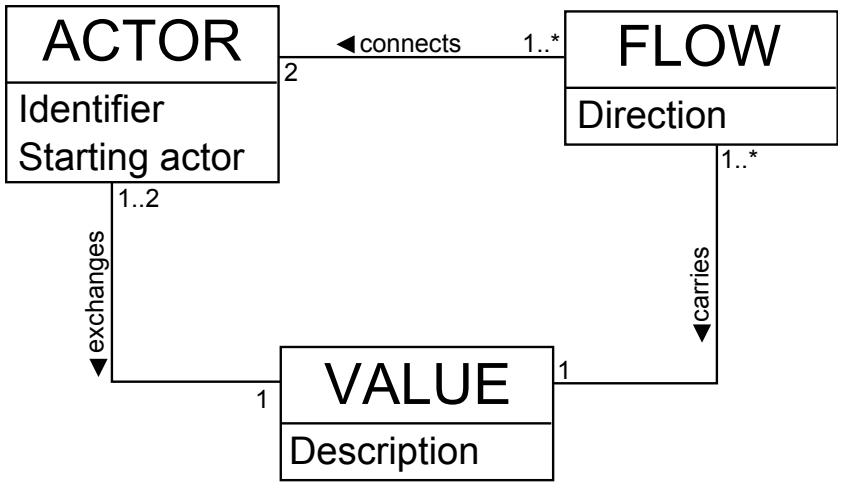

Figure 5.1: Meta-model of the business value flow diagram

An actor (vertice) represent a participant in the software ecosystem. It is usually a company or a single developer, but no explicit constraints are set. The various types of actors (companies, developers, business units, etc.) are not of interest here and thus not distinguished in the notation. An actor has an identifier, which is usually the name of the company when modeling a specific play or a role (partner, customer, etc.) when describing a more general scenario. One actor in the diagram is the "starting actor", depicted here as a boolean attribute of the Actor-concept. This is the actor which adopts the play and executes it. It is purely a semantic distinction, indicated in our diagrams by a slightly different colour. Flows (edges) connect actors. A flow always runs from an actor to an actor. It may connect from an actor to the same actor. For example, development costs to execute the play are usually indicated as a flow from the main executing entity to itself. An actor must have at least flow to participate, though it can be a passive participant, i.e. having one or more inbound flows, but no outbound flows. Flows may not flow through an actor and continue to another one: these must be distinguished into two separate flows. In our experience, wanting to run a flow through an actor usually indicates another deeper, semantic problem with the description of the business value. Business value (edge value) is exchanged by actors and carried over flows. It has a textual description to indicate its contents. A value may appear on more than one flow, while it should obviously appear at least once.

5.3 Software ecosystem plays

To show how a business value flow diagram can be used to develop a clearer understanding of plays in software ecosystems, a number of existing plays from literature are expressed in business value flow diagrams. We begin by searching for applicable plays in software ecosystems in existing literature. These plays are primary adapted from Popp [59] and Iansiti [37] and to a somewhat lesser degree the results of the SLR in chapter 4. The plays are listed as rows in table 5.1, with unique IDs and the source indicated.

No strategy will be applicable in any and all situations in a software ecosystem. To make the intentions of each play more clear, we distinguish three variants of software ecosystems which occur in practice. These variants are defined between them by the primary unit of exchange in them (flow) and the primary motivation for parties to participate in the ecosystem (counter-flow). These variants are included as columns in table 5.1. For each variant, we indicate which plays might be used by adding a ✓ in the appropriate field in table 5.1. The first variant, named "Platform", are the N-sided platforms as described by Armstrong [3]. Companies exchange access to their users and their data on these customers through the market platform. The primary motivation for joining these ecosystems is the money to be made and the complementary opportunities to be had. The second variant, named "Supply Chain", is of the traditional supply chains. A company that buys a Microsoft server license does not join a two-sided market platform: it simply uses the software to execute its primary goals. However, it is locked in to some degree to the supplier as it incorporates the technology into the business and uses larger portions of it. The primary unit of exchange in this variant are the products and services traded in the ecosystem, and the opposing payments are the primary motivation for companies to participate. The third variant, named "Community", comprises the open source ecosystems in which companies share open source code, use it in the development of their products, or sell services based upon it. The primary unit of exchange is thus "code", though usually exchanged in coherent higher-level units such as modules or products. The main motivation for participation is the recognition it brings to the company (or a single developer for that matter) when it contributes to an open-source product, and to a somewhat lesser degree the complementary opportunities it creates.

Not all of these plays are suitable to be expressed in a VEGA in the timespan allotted to our project. Additionally, some of these plays are either too obvious or too vague to be suitably expressed in a VEGA. To explain and prove the concept of applying a VEGA to We select a number of plays based on personal interest and experience to be modelled in VEGAs. Specifically, the plays numbered 1, 4, 6, 7, 10, 13, 15, 16 and 34 are included. These plays are depicted and described into VEGAs in the following sub-sections.

Id	Platform	Platform	Supply Chain	Community	Source
1	Support partner developers / Optimise partner productivity	✓	✓	✓	[37, 59]
2	sell a premium product			✓	[23, 32, 59, 73]
3	sell premium support / implementation / maintenance / consulting			✓	[23, 32, 59, 73]
4	public roadmap funding			✓	[23, 40]
5	Accept donations			✓	[23]
6	Adopt bounties			✓	[4]
7	Sell complementary stack products		✓	✓	[28]
8	Release open source as a defensive entrenchment		✓		[11]
9	adopt usage-based pricing		✓		[59]
10	subsidize one-side, extract the other	✓			[3, 64]
11	stimulate innovation	✓	✓		[59]
12	open source parts of the product	✓	✓		[72]
13	bundling		✓		[6, 18, 25, 28]
14	maximize retention through product investment	✓			[59]
15	Eat or destroy standards	✓			[37]
16	Create, support or adopt a public standard	✓	✓		[37]
17	Cull threatening partners	✓			[37]
18	Erect barriers to entry to promote stability	✓			[37]
19	Create high-value, sharable assets	✓			[37]
20	Leverage direct customer connections	✓			[37]
21	Create and manage physical information hubs	✓			[37]
22	Create, package and share state-of-the-art tools and building blocks for innovation	✓			[37]
23	Build or acquire financial assets for operating leverage	✓			[37]
24	Reduce uncertainty by centralizing and coordinating communication	✓			[37]
25	Avoid dominating and landlording	✓	✓		[37]
26	Keep API's consistent to promote stability	✓	✓		[37]
27	move towards borders of the ecosystem	✓			[37]
28	Specialize in unique capabilities	✓	✓		[37]
29	Leverage complementary capabilities from keystone	✓			[37]
30	Sustain innovation	✓			[37]
31	Adopt tight coupling with keystone	✓			[37]
32	Adopt loose coupling from keystone	✓			[37]
33	Use mobility as collective bargaining power	✓			[37]

Table 5.1: Collected plays in software ecosystems

5.3.1 Play #1: support partner productivity

A successful strategy for any hub is to optimise the productivity of its partners. This improves to the overall productivity of the software ecosystem and thus contributes to the health of the software ecosystem [37]. A hub can for example create systems for developers to reuse code from other developers [37] or deliver tools (free or paid) that generate compliant code to reduce the amount of 'boiler-plate' that developers need to write over and over again. The resulting VEGA is depicted in Figure 5.2.

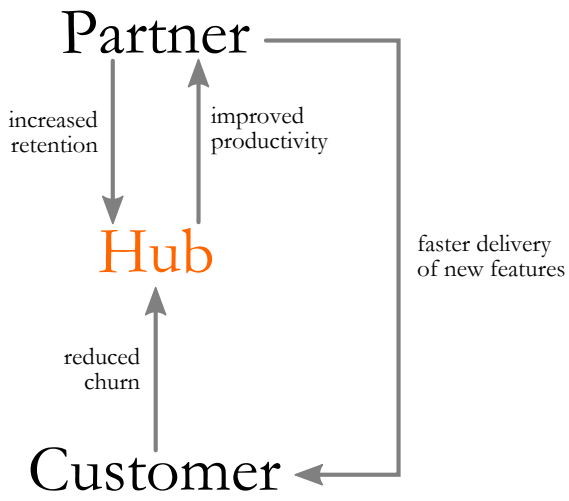

Figure 5.2: VEGA of play #1 optimising partner productivity

The hub is in the center of the graph. In this instance, we assume that the customer buys a software product from both the hub and the spoke. The hub incurs a cost for developing the tools and systems needed to support its partners. Even if these tools and systems exist for internal usage, these probably must be adapted before release outside the company. The partners benefit by using these tools and systems, presumably resulting in greater productivity of their engineers and a subsequent lowering of development costs. This results in them delivering new features faster to their customers. The play does not result in a direct benefit to the hub. It is not paid for supplying these tools or at least not enough to recover its investment in development. However, the company benefits in more easily retaining its partners and customers. Reduced churn of customers raises the company's profitability, while increased retention of partners offers other intangible benefits such as increased attractiveness to new customers, or a more innovative and resilient software ecosystem.

It should be noted that we can devise a lot more flows and business value that result from the flows depicted here. It is up to the user to include extra flows if needed. In general as the flows extend further and further their connection to the original play becomes increasingly tenuous and unreadable. We thus usually include only a single 'roundtrip' of a play so as not to over-complicate the graph.

5.3.2 Play #4: public roadmap funding

Public roadmap funding gives users of the software the opportunity to put their money where there mouth is. The software vendor defines a initial roadmap with the features, bug fixes and changes it intends to build in the specified time period. As customers use the software product in different ways, they will invariably have different wishes, concerns, etc. The vendor allows customers to pay extra (apart from the regular software pricing) to influence the product roadmap to cater to their wishes. A customer could create an incentive for the vendor to pull a specific feature forward on the roadmap, or add a new feature or change altogether. The main concern in executing this play is balancing the changes when two or more customer express conflicting priorities or wishes. The resulting VEGA is depicted in Figure 5.3.

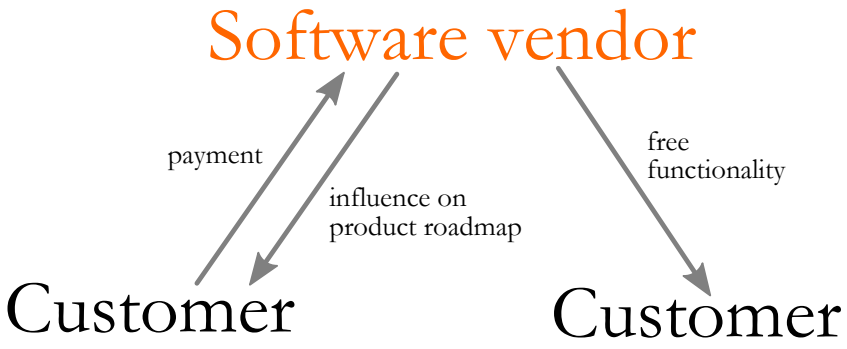

Figure 5.3: VEGA of play #4 public roadmap funding

The paying customer derives business value from having the influence it desires on the product. The software vendor is compensated for this by direct payments. The other, non-paying customers benefit too in two ways: a) as the company receives more revenue, it can presumably execute its roadmap faster, resulting in earlier delivery of new features, and b) a customer that has an active interest in a change in the roadmap for which another customer already has paid, does not need to add to the pile. Essentially, as long its wishes are common among other paying customers, the vendor will deliver its desired functionality early, for free.

This public roadmap funding as described is a relatively simple scenario. It can however be extended quite easily into a two-sided market platform for open source software. Suppose a developer creates an open source module. It has seen some adoption, though lately development has been slow as the developer has been busy working on other, paid projects needed to make a living. A hub could allow customers to pool voluntary contributions to help fund development. Customers can pledge money to a specific feature, for which developers can sign up to build it, earning the pledges when the feature is delivered. The hub could extract a small percentage as fees, while the largest contributors get the largest say in the roadmap for the product. The paying customers gain influence on the ordering of the roadmap, while

all customers receive the fruits of increased development efforts. Developers are funded collectively to work on the product, while the hub extracts a small fee for matching supply and demand on its platform. The VEGA of this adaptation of the original play is depicted in Figure 5.4. From this major change, it is thus apparent that VEGAs are inherently flexible and we can easily adapt them to describe changing circumstances.

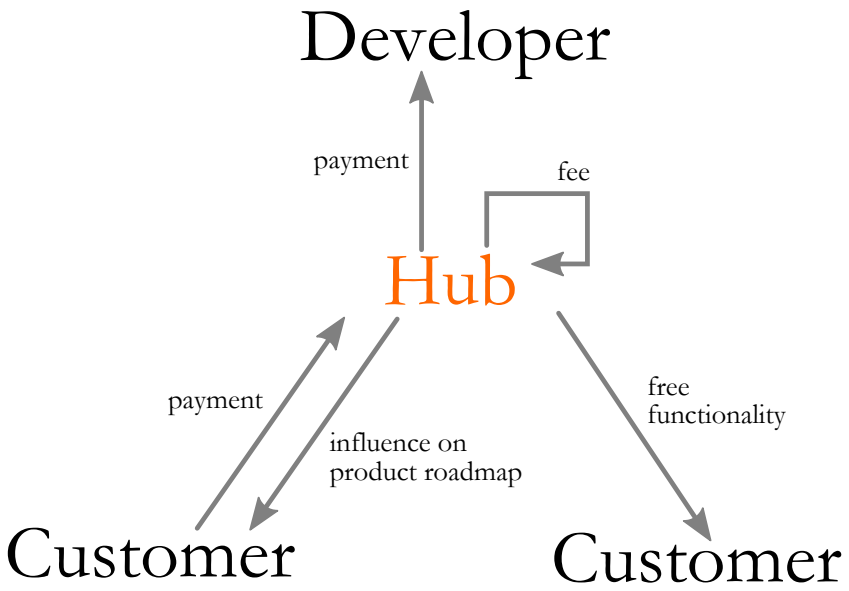

Figure 5.4: VEGA of an adaption of play #4 (public roadmap funding) to a two-sided market platform

5.3.3 Play #6: bug bounties

In a reversal of play #4, a software vendor may offer bounties for contributing to its software, essentially outsourcing a specific part of its operations to the public at large. The public could contribute to documentation, tutorials or anything else that the vendor values to be available but is not important to build right now. This play is currently most known for "bug bounties" in which users and security researchers are paid for bugs reported to the company. The resulting VEGA is depicted in Figure 5.5.

The software vendor benefits by receiving more bug reports of problems in its offerings. The customers that submit bug reports, receive the bounty payment in return. The other customers benefit from using improved software, due to having more bugs caught earlier, while the software vendor benefits by having less frustrated customers and thus less churn.

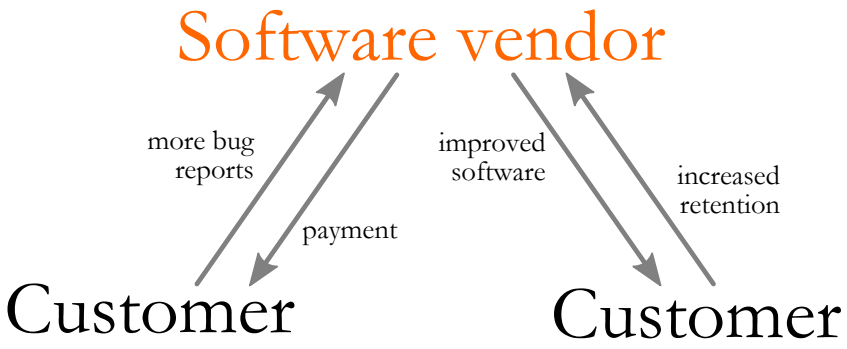

Figure 5.5: VEGA of play #6 bug bounties

5.3.4 Play #7: building complementary products

One of the primary ways to operate successfully is to create as many as possible complementary products and services. Having a complementary product strengthens the offering of both parties, while the customer enjoys a benefit due to improved interoperability. The play could be executed in various ways, such as creating an entirely new product, adding features to make the products more compatible or even removing a part of the product entirely, but we assume that in any case a change in the product is necessary. Suppose that we were to consider the scenario in which a smaller company adapts an existing product it offers to better complement an existing, different product by a much larger party, called the hub. The company can now offer its product in conjunction with the hub. The resulting VEGA is depicted in 5.6.

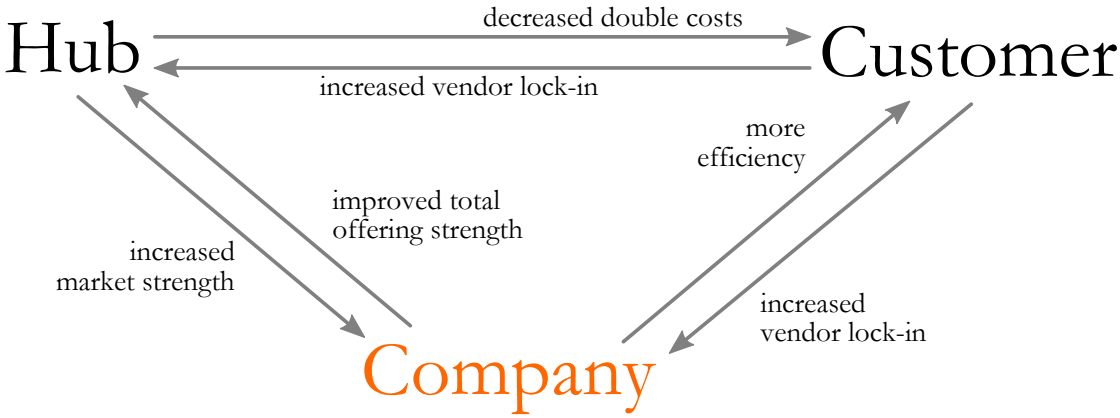

Figure 5.6: VEGA of play #7 building complementary products

The hub benefits by having an improved total offering to sell to its customers. Even if there is no formal integration between the products, simply mentioning that compatible products are available for the customers other problems, could help win sales pitches. In return, it offers the company improved access to the larger market of its own customers. The

company helps its existing customer by making the integration of data from the two products more efficient. Customers will increasingly integrate their data, increasing its value and enhancing vendor lock-in. Finally, the customer benefits by not having to pay for identical functionality twice though this connection is tenuous at best, as the other participants would probably not reduce prices voluntarily. In return, the hub too enjoys increased vendor lock-in as it is now the only party offering the functionality to the customer, rather than two companies competing.

5.3.5 Play #10: subsidise one side, extract the other

Play 10 is the general underlying principle that underpins the "two-sided market platforms" [3, 64], i.e. connect two sides of a market through your platform, identify the side which benefits the most from having access to the other side, and charge that side a premium (called the "money-side"), while subsidising the product for the other side (called the "subsidy-side") to attract maximum number of customer on both sides to your platform. The most famous examples are the large console gaming platforms such as the Xbox, for which Microsoft sells the console as a loss-leader, luring customers to the platform attracted by its (relatively) low initial costs. As triple-A game development is expensive, multihoming by developers tends to be limited. Developers will join the platform that promises the largest potential market of consumers. Microsoft subsequently extracts a fee from each game sale and earns back its investment through these cutbacks. Reversals of this play occur too, in which consumers pay a premium to access suppliers and products that are normally unavailable to them. The generic variant of this play can be expressed in a VEGA and it is in Figure 5.7.

Figure 5.7: VEGA of play #10 subsidise one side, extract the other

In this VEGA, the customer is considered the subsidy-side and the partner the money-side. The customer benefits by receiving a product or service from the platform at a lower price than would normally be considered. In exchange, the platform benefits by locking its customers into its product. The partner benefits by having access to the platform, and compensates the platform owner by handing over a percentage cut of total sales.

5.3.6 Play #13: bundling

Bundling is the combination of several (software) products into a single, presumably coherent package that is sold as one to a customer. The primary goal for the company which creates the bundle could be to strengthen its own product which are not of significant value in and of itself, and command a premium price point. The VEGA of this play is depicted in 5.8.

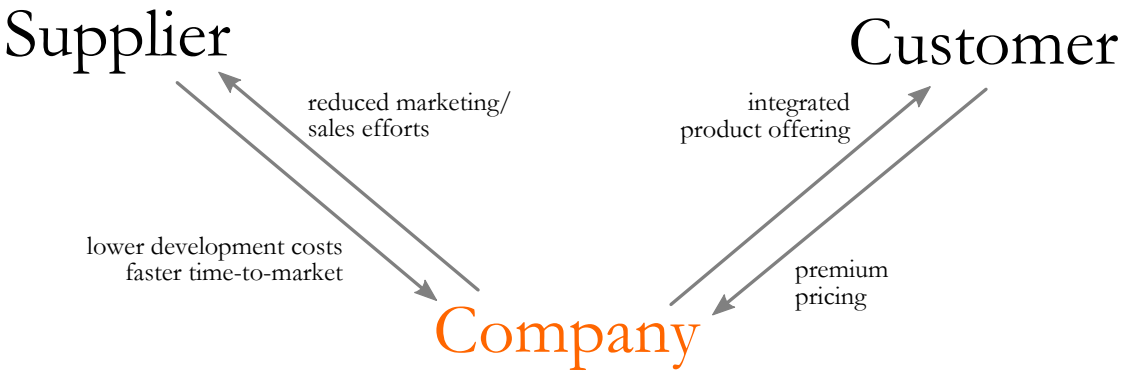

Figure 5.8: VEGA of play #13 bundling

The customer benefits by having an integrated product offerings to cooperates better than the separate products would. In return, the company adopts a higher price for the combined offer than the products would command when sold separately. The supplier benefits by having to spend less time/energy/money marketing and selling its products as this burden is carried by the company. Depending on the arrangement, in for example a white-labeled bundle, this could also include lower costs for supporting end-users, as these costs too shift to the company. In return, the company benefits by having lower product development costs and a faster time-to-market as its products gain functionality it doesn't have to build itself.

5.3.7 Play #15: disregard standards

Play #15 and #16 are two sides of the same coin. Either deliberately disregarding relevant standards that are in place (#15) or adopting them (#16). A company might consider explicitly not adopting a common standard that is in place if it improves its position in the market. The basic aim of the play is apparent from the major example of Microsoft's “embrace, extend, extinguish” strategy, in which Microsoft publicly adopted (“embrace”) common and open standards, added incompatible features and improvements to them (“extend”), thereby making Microsofts products compatible with the competition, but not vice-versa reducing the level playing field (“extinguish”) [74]. We will review the scenario or variant of this play in which an existing standard is in place. The company adopts the standard, but extends it with performance enhancements incompatible with the original standard. The resulting VEGA, which only considers the customer side for reasons of brevity, is displayed in Figure 5.9. A careful reader will note that this is an example of a purely non-monetary play, as it has no flows on which the business value consists of either payment or savings.

The customer benefits by having improved performance in using the product. However, in exchange it must incur a greater level of vendor lock-in as it can no longer exchange its data easily with competing products. In addition, the perception of vendor lock-in harms the interest of the company as it will reduce inflow of new, potential customers [70, 72].

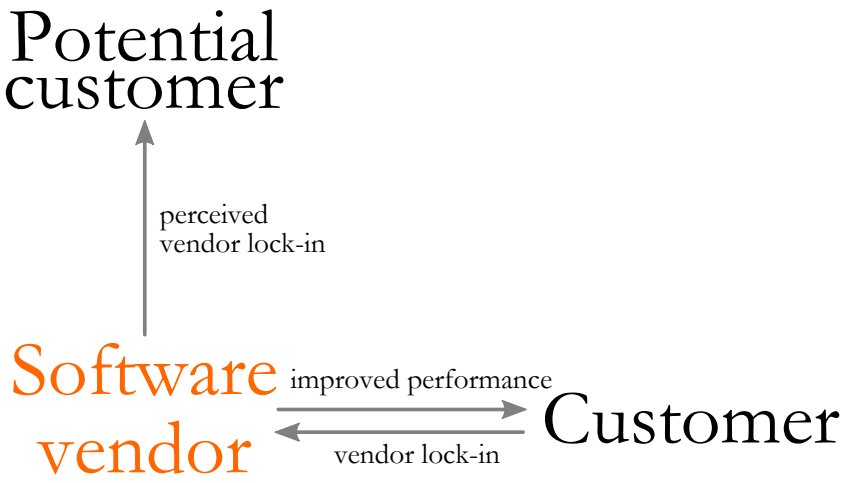

Figure 5.9: VEGA of play #15 disregard standards

5.3.8 Play #16: adopt standards

In a reverse of play #15 a company can adapt its products to adhere to new public standards which are not under the control of any one company. The primary goal of the play is to make its products more attractive to the consumer by promising a reduced level of vendor lock-in. Additionally, the company might promote a common standard to lower R&D costs for its products. Both considerations are expressed in the VEGA in Figure 5.10.

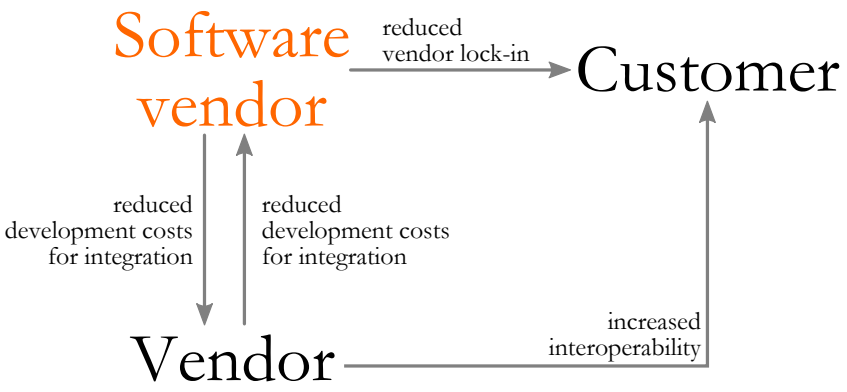

Figure 5.10: VEGA of play #16 adopt standards

Suppose we were to consider adopting a common standard which makes exchanging data from several applications easier by specifying a common format and its formal interpretation. These applications might or might not have overlapping functionality. The customer benefits and the company suffers from having less vendor lock-in. Both the company and the other vendors profit from reduced R&D costs for making their products successfully interoperable. Finally, this increased interoperability again benefits the customer in having less

issues working with data stored in multiple systems from different vendors.

5.3.9 Play #34: divestment

Divestment is the 'extreme' version of play #7 (complementary products, section 5.3.4). The company removes a portion of its product to create space for partners to operate in. A VEGA of such a play is expressed in Figure 5.11.

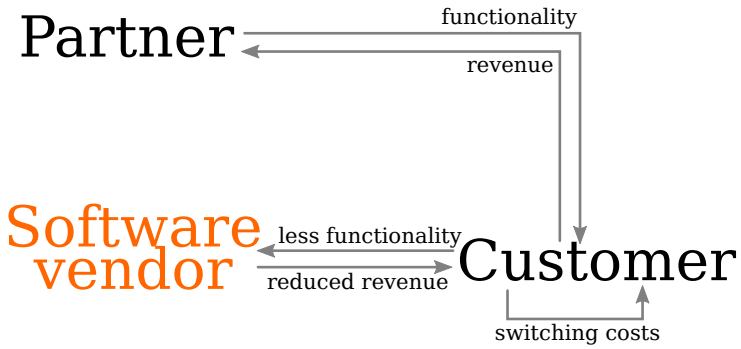

Figure 5.11: VEGA of play #34 divestment

In this scenario, we assume the software vendor sells a products consisting of a few modules that are perceived as distinct to some degree and are configurable, i.e. the product can be operated and sold with an incomplete set of its modules. Take for example an enterprise resource planning (ERP) product with human resource management (HRM) and customer relationship management (CRM) modules. The CRM module is small, limited in functionality and has seen less proactive development over the years. Partners of the software vendor build a superior product, and sales of CRM have been declining. The software vendor must nevertheless incur maintenance costs to support existing clients. The vendor can now improve its situation by divesting its CRM module. It delivers less functionality to its clients (and thus lower costs) in return for reduced revenue, assuming it has adopted per-module pricing or gives its customers some discount. The customer must migrate to a new CRM product, losing its value and costs to the original software vendor, replacing it with the utility of the new product. In addition, the customer incurs switching costs for migrating data, re-training employees, etc. which will dampen its enthousiasm for participating in this the play.

We have stacked the table heavily in the scenario to make this play work, in having given the company an ulterior motive (module is outdated) to divest, which is not explicitly modeled in the VEGA. If this assumption is removed, the play becomes irrelevant. In addition the

customer will not be happy with this play by the software vendor if we take it that the utility the customers derives from the new product is the same as from the old module. It is thus obvious that this play is not optimal and could be significantly improved. We explore these improvements to this and other plays in the next chapter (chapter 6).

5.3.10 Limitations

A number of limitations have become apparent in the development of various plays, including both those mentioned here and even more so other less successful ones. These limitations make drawing VEGAs less or even not appropriate in some scenarios. We discuss these situations in this section.

Drawing VEGAs does not work properly in situations in which very little value flows are shared, i.e. exchanged between two participants. Drawing a flow from "nowhere" to a single participant in which the resulting flow has no beginning party or no end, is currently not supported, but might be needed in the future. An example of this are the flows that appear in some plays or our plays that are essentially a form of "costs incurred to execute this play". These flows now flow from one participant to itself, but this is essentially a stop-gap measure that does not work out very well, especially once one starts reasoning on these flows (chapter 6). It is currently preferred over having flows end in nothingness, as having these circular flows is more fitting to the actual situation. Having a flow end in the void, suggests that value would be permanently destroyed, which is in our experience almost never the case. There are other complicating factors to this problem. Some flows that enter from nowhere and end in a participant that is not the main actor of the play, are not usually exclusive to the play at hand, and can be reached by the participant that receives them in other ways, making analysis of them more complicated.

A natural limitation of applying VEGAs is that it only considers the exchanges between companies. A valid strategy in a software ecosystem is to create "shareable assets" (play #22) which are assets that the company holds that are applicable to the problems of many of its customers [37], essentially high-tech building blocks of customised solutions, comparable to the software product line approach [8]. A VEGA of this "creating shareable assets strategy" is included as Figure 5.12. While this play benefits the company, its customers and the software ecosystem as a whole (by increasing productivity) and it is a very important strategy within some large, successful software ecosystems; and while the depicted VEGA here is a complete and correct depiction of the resulting flows of this play, it is also completely pointless.

Finally, the "state" of a partner is modeled awkwardly and should be improved in the future. It is currently indicated on the description of the actor, e.g. "partner" versus "potential partner", and "customer" versus "potential customer". This could be significantly be improved.

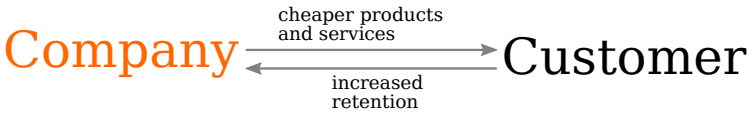

Figure 5.12: VEGA of play #22 creating shareable assets

5.4 Conclusion

The introduction of the VEGA and its demonstration serves as the first half of the answer to our third research question. We have shown that we can create an effective, graphical way to express the flow of (business) value through an software ecosystem and the associated capture/release of it by partners. We will pursue this concept further in the next chapter (6) to show that VEGAs can be used in formal reasoning to enhance our interpretation of the effects of plays on the flow of value.

Chapter 6

Reasoning over Value Exchange Graphs

6.1 Introduction

Expressing a strategy in a VEGA allows for structured reasoning over its flows. This enables us to clearly indicate potential changes to plays to improve them. In this chapter, we explore this approach. We explain the general principles and the assumptions we make (6.2). By discussing some of the plays of the previous chapter, we show that we can use structured reasoning to i.a. make it clear who benefits from plays (6.3), to improve suboptimal plays (6.4), and to guide pricing decisions (6.5). Unfortunately, this approach is not universally applicable and we discuss its limitations (6.6), and finally the most-important conclusions (6.7).

6.2 Reasoning on plays

The primary underlying idea of a Value Exchange Graph is that value is exchanged between parties in a software ecosystem. This allows us to define the utility of a participant in a play. The generic value function

$$v(x, f)$$

is the amount of value that participant x perceives to receive or give away through flow f. As the formulas can grow to be very long with extensive description of value, we use

a shorthand version $v(f)$ in cases in which the participant is clear from the text, and we might abbreviate the description of flows in some cases. Note that this function expresses not the business value received by x, but x's appreciation of it. While for some forms of business value that are expressed on an integer scale, most notably money-based forms, we can assume that two participants will value it equally, it can *not* be assumed in general that $v(x, f) = v(y, f)$ if $x \neq y$.

The utility of a play to a participant p can than be expressed as the sum of all inbound value, minus the sum of all value the participant loses through the play:

$$u(p) = \sum v(p, \text{in}) - \sum v(p, \text{out})$$

If we assume that participants in a software ecosystem operate as rational actors, we can define the precondition for joining a play as such that a participant must derive some utility from the play: $u(p) > 0$. This naturally expands into

$$\sum v(p, \text{in}) > \sum v(p, \text{out})$$

The precondition must hold for a rational actor to participate in the play willingly, without being forced by some outside condition. We can use it to reason about the implications of executing a play.

In the adoption of the precondition we make a few assumptions. First that the participants operate as rational actors. This is a reasonable assumption to make as we study companies, not individuals and companies to some degree always operate as rational entities. Additionally, we assume that the existing situation, i.e. the exchange of business value before adopting the play, is stable and willing. "Stable" means that the actors and flows can be considered *ceteris paribus*, and the only change is wrought by the adoption and execution the play. "Willing" means that the participants have joined the existing situation willingly, not forced by some outside mechanism. This means that for every participant p in a play the assumption $u(p) > 0$ and thus $\sum v(p, \text{in}) > \sum v(p, \text{out})$ is true in the initial situation.

6.3 Who benefits?

Structured reasoning over flows can be used to show who stands to benefit from a play. We will illustrate this using play #16 from the previous chapter. The VEGA is repeated here for the reader's convenience as figure 6.1 and is identical to the original one depicted in figure 5.10.

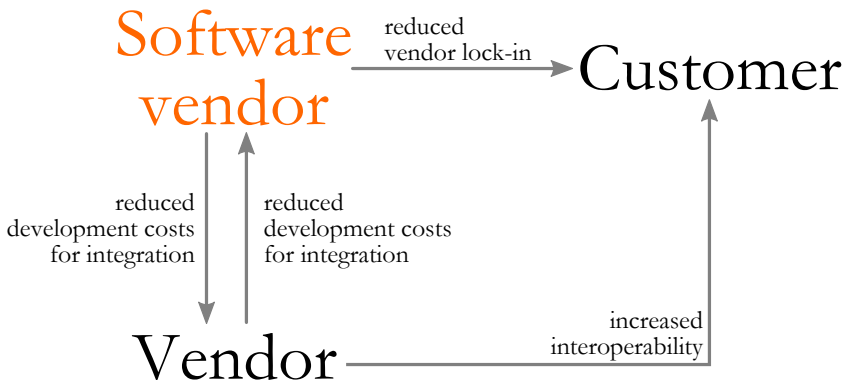

Figure 6.1: VEGA of play #16 adopt standards

The flows between the software vendor and the secondary vendor are an equal exchange so we assume a reasonable

$$v(\text{software vendor}, \text{reduced development}) = v(\text{vendor}, \text{reduced development})$$

That makes this part of the exchange neutral with respect to the value exchanged. As we can reliably assume that the customer will appreciate these forms of value, we take $v(\text{customer}, \text{reduced vendor lock-in})$ and $v(\text{customer}, \text{increased interoperability})$ to both be positive and it becomes clear that the customer will always join the play willingly:

$$\sum v(\text{customer}, \text{in}) > \sum v(\text{customer}, \text{out})$$
$$v(\text{customer}, \text{reduced vendor lock-in}) + v(\text{customer}, \text{increased interoperability}) > 0$$

The utility of the software vendor is

$$v(\text{costs for integration}) > v(\text{reduced vendor lock-in}) + v(\text{costs for integration})$$
$$0 > v(\text{reduced vendor lock-in})$$

from which it is clear that the software vendor only stands to lose from the play. It thus becomes clear when the play is suitable for the software vendor. It should only adopt a common standard in two conditions. One, if it desires rapid integration with many partner vendors more than it wants to protect its own interests

$$v(\text{software vendor}, \text{reduced costs}_{in}) > v(\text{software vendor}, \text{reduced vendor lock-in}) + v(\text{software vendor}, \text{reduced costs}_{out})$$

Or alternatively, when there is some ulterior motive that makes the play desirable outside the scope of direct business value, such as pressure from consumers to adopt standards, or a desire of the software company to portray a more favourable image of itself.

6.4 Improving a play

Reasoning over plays can also be used to construct improvements to a play in a structured fashion. We discuss this point at the hand of play #34 from the previous chapter, reproduced here in figure 6.2. The current basic play has two subtle flaws that could be improved. The first flaw becomes obvious if we take that the replacement product by the partner is perceived by the customer as equal both in price and functionality to the original product. Expressed in the value-function, we take $v(functionality_{out}) = v(functionality_{in})$ and $v(revenue_{out}) = v(revenue_{in})$ and it becomes clear that the customer will not be happy about the change as he/she only incurs the switching costs and gains no benefit:

$$u(customer) = \sum v(in) - \sum v(out)$$
$$u(customer) = v(switchingcosts)$$

The second flaw relates to the value exchange of the partner. The utility of the partner

$$u(partner) = v(revenue) - v(functionality)$$

results in an apparent positive utility $u(partner) > 0$ in any scenario. If the utility were to become negative, it means that the partner sells its product at a loss, which is not useful[1]. However, nothing in this play forces the customer to buy the replacement product from a partner of the software vendor, and if the customer buys another product from another company that is not a partner of the software vendor, the company's partner is essentially excluded from the play.

Now that the weak points of the play are made explicit, we can adapt the play as needed to improve it. The software vendor agrees with a specific partner to have it act as a “preferred (replacement) supplier”. The company makes the customer's data (with the permission of the customer) available to this partner, while the partner pays a lead fee in exchange for every customer that switches to its offering. The customer incurs no switching costs to the partner as its data is already automatically imported and made available. To model these changes, we remove one flow and add a new one. The VEGA of the improved play is included as figure 6.3.

[1]The authors recognize that in some business models such as loss-leader plays, selling products at a loss is perfectly fine. However, this scenario would necessitate adding extra flows, making the play less clear and befuddle the main point. We take these scenarios as out of scope, but the reader is invited to construct them him/herself if desired.

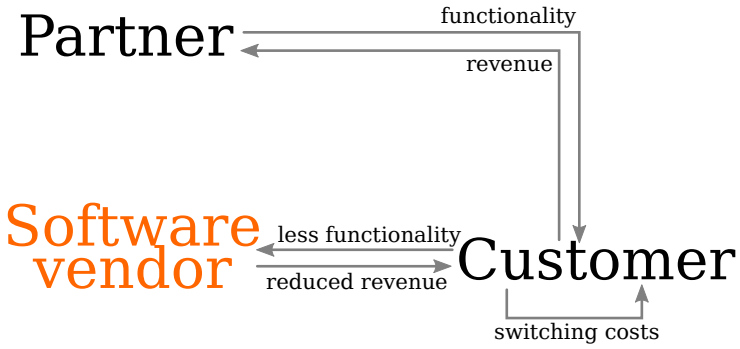

Figure 6.2: VEGA of play #34 divestment

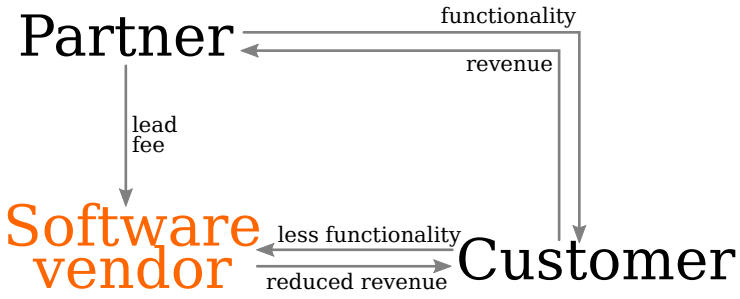

Figure 6.3: VEGA of the improved scenario play #34 divestment

The play has now markedly improved for all parties. The customer's utility is zero, as its inbound and outbound flows are identical while it no longer incurs switching costs. While the customer is not getting a great deal per se, its utility has improved from the original, negative utility. The utility for the partner is still positive, provided that the lead fee is lower than the margin it makes on its product:

$$v(fee) < v(revenue) - v(functionality)$$

Last but not least the utility of the software vendor has improved by $v(fee)$, as it captures some of the upside from the partner's new turnover, receiving a new lead fee while leaving its other flows untouched.

6.5 Determining a price point

Formal reasoning can also be employed to determine a price point, to some degree. While determining an exact price point using this method requires resolving the value function, which is hard or impossible (section 6.6), this structured approach can aid in determining key factors in pricing decisions.

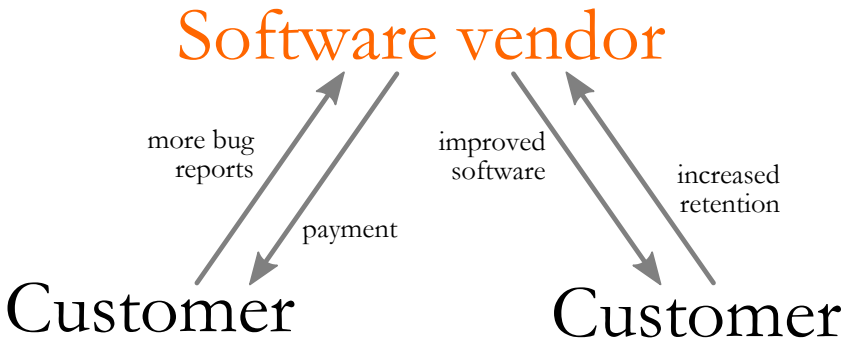

Figure 6.4: VEGA of play #6 bug bounties

Recall the VEGA of the bug bounties play (#6), reproduced here as figure 6.4. How much should the company offer as a bounty? To answer the question, we can split the value function of the customer into two parts to make it more clear:

$$v(\text{more bug reports}) = v(\text{finding bugs}) + v(\text{reporting bugs})$$

We cannot model this change directly in the VEGA without using the value function, as the two parts of the equation are intertwined from the perspective of the company. The

company derives no value from the customer finding bugs without reporting them, while the customer cannot report bugs it hasn't found.

Finally, we take it that the customer does not care about its retention as it does not incur a cost to be retained so $v(customer, retention) = 0$. The utility for the customer has now evolved to

$$\begin{aligned} u(customer) = & v(\text{improved software}) + v(\text{payment}) \\ & - v(\text{cost of finding bugs}) - v(\text{cost of reporting bugs}) \end{aligned}$$

If we rewrite the equation for $u(customer) > 0$:

$$v(\text{improved software}) + v(\text{payment}) > v(\text{cost of finding bugs}) + v(\text{cost of reporting bugs})$$

the key scenarios for the decision become abundantly clear as we solve for $v(payment)$. If the company wishes to attract professional bug hunters that are not using its product, for which $v(\text{improved software}) = 0$, it must adopt

$$v(\text{payment}) > v(\text{cost of finding bugs}) + v(\text{cost of reporting bugs})$$

If it only wishes to entice current customers who use the product to report bugs that they stumble upon in their daily work, the company adopts

$$v(\text{payment}) = v(\text{cost of reporting})$$

If the software is perceived as critical by its customers, we can safely assume that any bug will be promptly reported (and patches demanded), regardless of the costs incurred by the customer: $v(\text{improved sofware}) > v(\text{more bug reports})$ and the company adopts

$$v(\text{payment}) < 0$$

which essentially translates to “do not adopt this play”.

6.6 Limitations

Reasoning over VEGAs is not without its quirks and limitations and the authors recognize that the system could be significantly improved in further iterations. We discuss the four main concerns here.

The most obvious limitation is that actors are not rational. While we assume $u(p) > 0$ as the precondition for a partner p to join the play willingly, it is more likely in practice to be $u(p) > V$ for some undefined value of V which varies per participant, per situation, per play, and with the weather. Additionally, partners might not appreciate the changes a play brings, even as their resulting utility is still positive, as a way of sticking to acquired rights.

Second, the value function is too abstract for interpretation. Recall that the value function describes the amount of business value that an actor perceives to receive from a flow. Not only is it difficult to resolve the value function to an actual hard number for yourself, but resolving it for other parties is extremely hard as it is skewed by the asymmetric appreciation of value. Furthermore, partners and customers have vested interest in negotiation to maximise their own utility, making fact-finding hard. The main benefit of the value-function lies in the structured approach of the thinking instead of the actual values, revealing the key determinants of the answer, if not the exact number.

The third limitation of using the value function is that some of the results are with hindsight obvious. For experienced entrepreneurs and , expressing plays in VEGAs and value functions derives only insights that are obvious and could be more easily reached through simpler methods.

Finally, these approaches do not account for time and competition aspects. Plays in general take time to work, and $1 of business value now is better than $1.05 of value in a year. Neither do we model the effects of competition exhaustively as these tend to make the plays quite complicated. In combination with the first limitation, utility functions with six or more distinct forms of business value tend to be unresolvable.

6.7 Conclusion

The main conclusion of this chapter is that value-exchange graphs can be used in formal reasoning to determine key determinants of a play that influence its performance, though the reasoning has several important limitations and applications of it in practice are expected to be of limited value. This provides the second half of the answer to our third research question.

Chapter 7

Case studies

7.1 Introduction

Expressing theoretical plays in VEGAs and reasoning on them, as we did in the previous two chapters, is not enough to prove the actual worth of this approach. In this chapter, we survey established companies on their relationships with other participants in their respective software ecosystems. Based on interviews with senior leadership at these companies, VEGAs are drawn and shown to the interviewees for validation. Four sub-groups can be identified in the set of case studies. The results of this process serves as a partial answer to the third research question on how value in software ecosystems can be modeled.

This chapters begins by describing the approach and process of the interviews (7.2). The four subgroups are depicted and all individual cases discussed (7.3). Finally, we discuss the results of this validation phase (7.4) and the main conclusions (7.5).

7.2 Approach

Companies are selected on wide selection criteria. As we seek to show that VEGAs can be applied to real-world software ecosystems and produce a solid overview, we do not impose artificial limits on what types of software ecosystems these companies operate in. We do limit the survey to companies and thus to "business software ecosystems" in a sense that we exclude single developers operating in the context of an open source project. The full survey contains cases on ten companies.

The interviews were conducted over Skype with one employee of the studied company. These employees were selected by the company based on the subject matter, with minor

suggestions from the researchers in some cases in which we were previously familiar with the company. A majority of the interviewees held senior roles in partner management such as “international partner manager” or a related role as “marketing manager”. In two cases in which senior personnel were unavailable, we have interviewed junior employees. One company (Exact) was not interviewed, as the authors spend eight months working on this research project embedded at Exact's headquarters as a student employee. Any formal interview would be superfluous.

Interviewees were informed of the subject matter and a rough outline of the interview in advance. All interviewees gave explicit, verbal consent to the interview being recorded. This consent was repeated for the record directly after the recording started. All interviewees were given the explicit choice whether to protect the identity of their company by including only anonymised data in this thesis. None of the interviewees requested that their company remain anonymous.

The interviews are recorded in full with audio of both the interviewer and interviewee, with video added where possible. All e-mails regarding the interviews and subsequent VEGAs are exported to PDF and archived with the footage.[1] This provides a clear chain of evidence to back up the research claims.

The interviews are conducted in a informal manner, with few questions set in advance. The interviews are are roughly structured around two main points: first, the interviewee describes his/her employers, its size, main products and markets, etc. Second, we discuss which partners are important to the company and how the company cooperates with them.

The authors subsequently depicted these relationships in a VEGA and e-mail this depiction to the interviewee. No further explanation or interpretation was included, not even a general overview of the concept of a VEGA so that interviewees would not be influenced in the slightest. The interviewee is asked to answer two final questions by responding per e-mail:

1. Is this an accurate depiction of the situation of your company? Are there any elements which you find to be inaccurate?

2. Are there any improvements you would like to add?

Based on the answers to the questions, the authors updated the VEGAs before including them into this thesis. Minor problems such as spelling mistakes in names were repaired in the VEGAs. Major problems such as fundamental issues in the approach taken or important relationships missed are not repaired, and are included in the discussion of the cases below. These major issues show the limitations of this approach.

[1] The authors make these records available upon request. For contact details, see the administrative information table on page i.

7.3 Cases

Ten companies were interviewed and included as a case. While each of these companies has some unique aspect to it, upon studying the resulting VEGAs some common traits emerge. Dividing the group of companies along the common traits, four main groups emerge that fall somewhat along the lines of Iansiti & Levien [37]. We find these four patterns in the group: supply chains, hubs/keystones, niche players, and networked companies. The hubs/keystones and niche players are well-known distinctions from Iansiti & Levien which the authors expect the reader to be familiar with. The group "supply chains" contains two companies that have primarily 'traditional' relationships with other companies, exchanging solely products and services for money. Networked companies are best described as "multihoming niche players". We will discuss each group and their common elements in this section.

7.3.1 Supply chains

Two companies appear to be "traditional supply chains" in the sense that products and services are exchanged for money. These companies are iHomer[2] and Datafox Benelux[3].

iHomer is a software company based in Etten-Leur, The Netherlands, employing approximately 19 FTE. It deploys developers on secondment at other companies, specialising in software development for electric cars, customs, healthcare, retail and crowdfunding. Unique aspects of iHomer are that its employees are expected to work from home four days a week, only meeting in person one day a week, combined with an extreme degree of freedom for individual employees. The VEGA of iHomer is depicted in Figure 7.1.

iHomer supplies its customers with outsourced development services in exchange for payment. It acquires a number of goods and services from other companies, all in exchange for payment. For example, it acquires hosting services from a ISP, and common modules from other software vendors to reduce rework. Staff services such as administration, sick leave and all kinds of legal paperwork are handled by a dedicated service provider. When design services are requested by the client, iHomer includes a preferred supplier in its quotation.

Upon being presented with the completed VEGA, the interviewee of iHomer remarked that the model presented a very simplified picture of the situation at iHomer, though it was largely correct. He remarked that iHomer cooperates more frequently with the design agency, often sharing leads and collaborating on new development trends. This is mentioned in the interview but was not accurately reflected in the VEGA. iHomer also collaborates with its customers on new pricing models rather than the work-for-hire model reflected in the

[2] `http://ihomer.nl/`
[3] `http://www.datafoxbenelux.nl/`

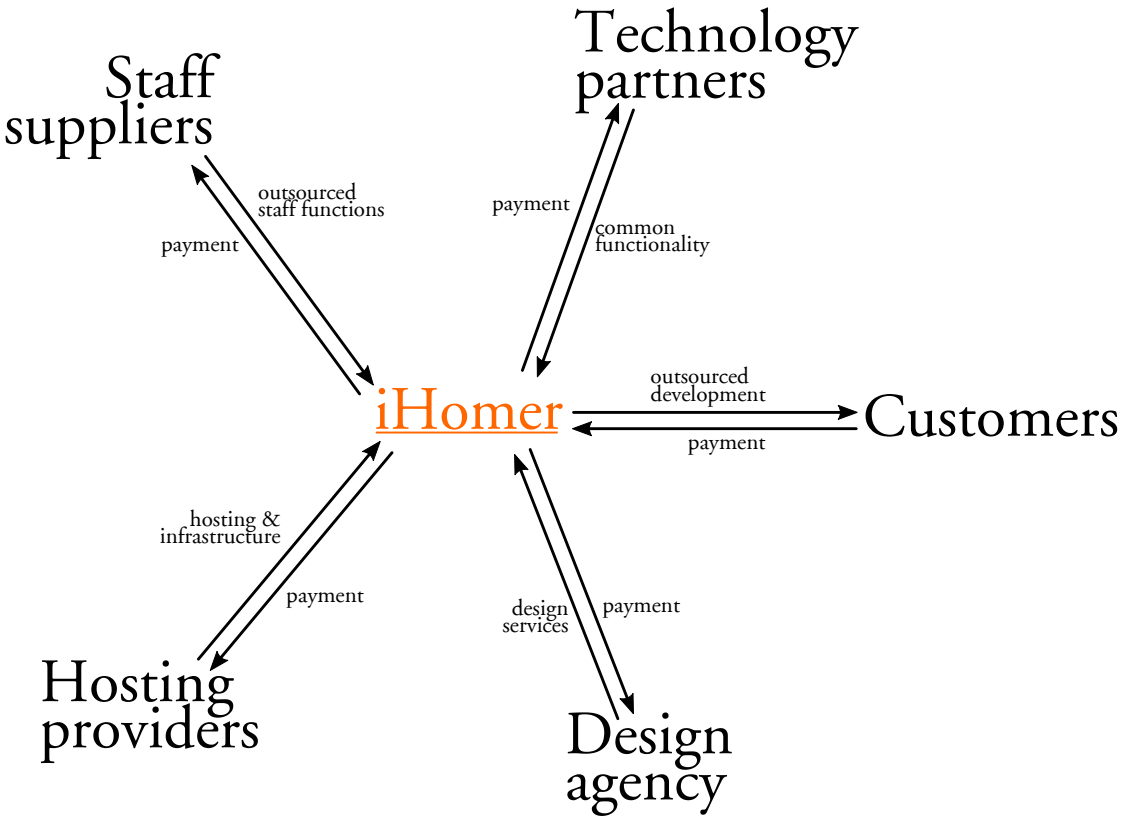

Figure 7.1: VEGA of iHomer

VEGA. This aspect was touched upon in the interview, but the authors believed it to be of minor concern. These issues show that the completeness and quality of the resulting VEGA is very dependent on a thorough analysis of the company.

Datafox Benelux is the exclusive importer of products of Datafox Gmbh in Belgium, The Netherlands and Luxembourg. Datafox Benelux is an independent company, not a part of Datafox Gmbh and employs 3 FTE. Datafox's main product are physical terminals for time registration (punch clocks). These terminals are often used for shift registration of employees, access control and billing purposes. The VEGA of Datafox Benelux is depicted in Figure 7.2.

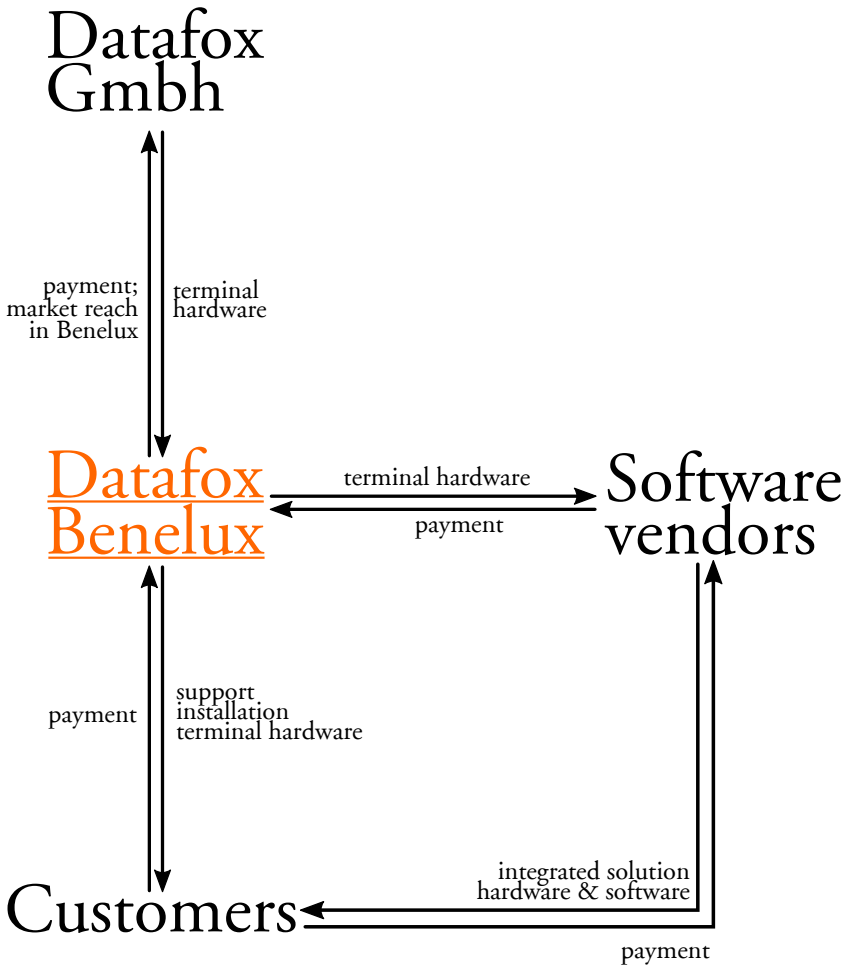

Figure 7.2: VEGA of Datafox Benelux

Datafox Benelux simply operates as an exclusive reseller in the Benelux. It is a member of various software ecosystems in the sense that most of its offers are sold directly to software vendors which bundle the terminals with their tightly integrated software and offer a complete, fully integrated solution to customers. Occasionally Datafox Benelux also sells directly

to end-users based on leads from a software vendor, though this occurs infrequently. Finally, end-user support and regular maintenance on the terminals is also provided by Datafox Benelux to the customers.

Reflecting on the finished VEGA, the interviewee at Datafox judged the VEGA to be an accurate representation of the company. He noticed that installation services were missing from the graph. This service was mentioned in the interview, though the authors considered it a part of the support flow, which was included. The installation flow was added to the VEGA to clarify this relationship.

From these two VEGAs, it becomes abundantly clear that drawing VEGAs is not a very suitable approach when modeling business relationships that effectively amount to traditional supply chains. In both VEGAs, nearly all flows are of the type "<some product X> in exchange for <payment>". This is correct in the sense that the customer apparently needs that product or service, and thus the fact of having the product or service *is* the business value. However, if one were to draw these relationships in a software supply network, the relationships would be exactly the same, save for some slightly different wording.

Additionally, it is apparent from these VEGAs that it becomes very difficult to determine where to stop. Modern supply chains can be very large and very complex. Almost no employee cleans his own desk, should cleaning services be included in the VEGA? Catering? One could argue that the Staff supplier should be removed from the iHomer VEGA (Figure 7.1) as it is simply an outsourced support function of the business, not related in any way to the core service which iHomer delivers. It is however one of the unique aspects of iHomer as a company, akin to how it operates without a single office. Strongly related to this issue is the problem that it is currently not possible in this notation to account for the importance of the relationship. This problem is apparent in the VEGA of Datafox Benelux (7.2). If one were to only read the VEGA, one might very well conclude that Datafox Benelux sells its products in two markets: directly to end-users and to software vendors to resell in integrated, full-stack solutions. While this is correct in theory, in practice Datafox sells almost exclusively to the software vendors, only occasionally selling directly to end-users and derives a large portion of its revenue from its relationships with the software vendors. This makes the relationship with the software vendors much more important to Datafox Benelux than those with its direct customers, but this fact is not apparent from the VEGA.

7.3.2 Hubs

Two companies fulfill the traditional role of a hub in their respective ecosystems: they are an order of magnitude larger than their typical partner, hold crucial advantages, in these cases important financial customer data that other parties desire to have access to. At the same time, these hubs must attract enough niche players to their ecosystem to be competitive and thus have a vested interest in the productivity, health and resilience of their respective ecosystems.

Two companies are included in this group: Exact[4] and AFAS[5]. Both are major product software companies in The Netherlands, headquartered in Delft and Leusden respectively.

Exact employs approximately 1.600 FTE worldwide and is active in numerous countries in Europe, Americas and Asia. Its main products are focused on ERP systems and financial administration for small and medium businesses. The VEGA of Exact is depicted in Figure 7.3.

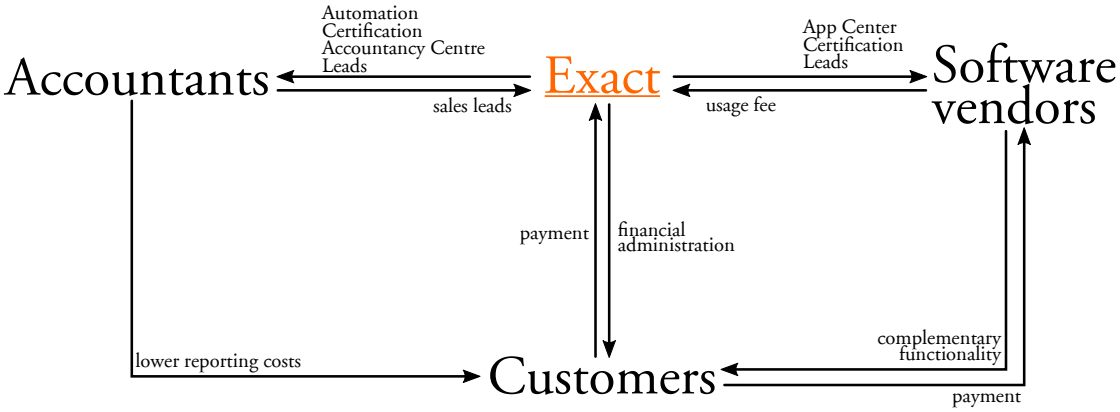

Figure 7.3: VEGA of Exact

Exact sells its software products directly to its customers. Through its Accountancy Programme and its Exact App Center, is has two important groups of partner companies. Accountancy firms are an important part of Exact's go to market strategy. Accountants can follow training courses on Exact's product suite and become “certified cloud accountants”. Accountants who pass the certification are added with special distinction to the “Accountancy Center”, essentially a website of all accountancy firms who have experience working with Exact. Customers of Exact can use this website to find an accountant experienced with their internal systems. This lowers the costs customers of Exact incur for reporting to their accountant. Exact provides special versions of its software for accountants to manage large numbers of customer administrations quickly and other power user features. In some cases, accountancy firms adopt Exact's software as their preferred solution for their existing and prospective customers, generating more sales (leads) for Exact. Using the Exact App Center, Exact offers other software vendors the opportunity to display their products in an App Center. This too is a website listing software for Exact customers which integrates perfectly with their data in its administration. Exact certifies these solutions through a standards inspection process, including marketing and technical reviews of the integration points. Exact has recently concluded that because of its market size in comparison to the average partner is lopsided. Exact sends much more leads to its partners than it receives in return. This unequal exchange is especially apparent in markets such as The Netherlands in which Exact

[4] http://www.exact.com
[5] http://www.afas.nl/

has a firmly established market and is considered a well-known company amongst its target customers. In response, Exact has adopted a usage fee for its partner software vendors, charging a (arguably small) monthly fee per customer for which the partner retrieves data.

At 350 FTE, AFAS is a smaller company compared to Exact though it develops its ecosystem in much the same way as Exact, though it focuses on different aspects of its ecosystem. The VEGA of AFAS is depicted in Figure 7.4.

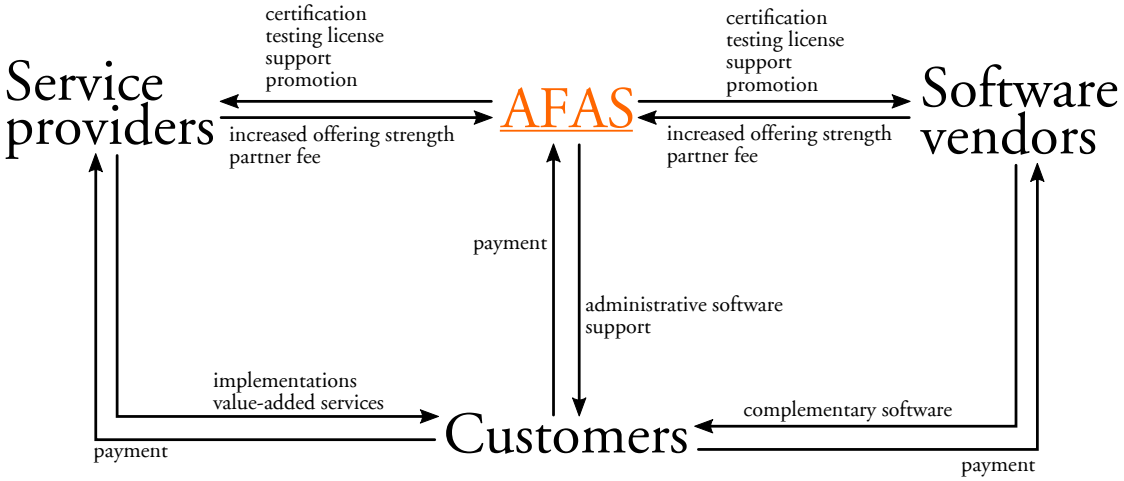

Figure 7.4: VEGA of AFAS

AFAS supply its customer with its software product for financial administration. AFAS works together with two main sets of partners: service providers and other software vendors. Service providers function much like Advisie (section 7.3.3) though Advisie in this case works primarily with Exact, not with AFAS. These service providers are listed on the website of AFAS, who supports them with additional support and a certification and includes them in marketing materials. Attracting these service providers helps AFAS increase the strength and size of its ecosystem, which makes its products more attractive to its customers. These service providers primarily sell advice, custom implementations and integrations (though AFAS does not allow them to change or adapt its core software, nor simply resell the product), and provide some services to clients that AFAS does not such as configuring the software for specific needs. The software vendors operate much in the same way as with Exact, and in fact some companies in this category such as Xpenditure (section 7.3.4) work with both AFAS and Exact as a complementary vendor.

When discussing the final VEGA, the interviewee argued that AFAS does not supply licenses to partners for the purpose of reselling. However, included in the recording of the interview is a segment on which a partner license is discussed, though not for resale through partners. AFAS supplies its partners with a single license for testing the integration of their product or service with the software, and it is this license that is added to the VEGA. More clarification would have been needed in this case and the flow could have been labeled more

appropriately as “testing license”. This problem was also appears in the flow “promotion” to partners, which the interviewee believed should have been directed at the customers, as AFAS promotes its partners to its customers through the website, marketing materials, etc. However, it is the partners who benefit from this action and thus receive business value from it. Again, more clarification here would have prevented this misunderstanding. The interviewee also indicated that the most important components of the partner program are certification and technical support for partners, and therefore this licenses-flow was dropped altogether. With regards to customers, the feedback indicated that AFAS naturally also directly supports its customers, rather than only its partners and this flow was added to the VEGA. Service providers are not allowed by the AFAS partner program to execute implementations of AFAS' software at customers and this label should have been removed. An interesting problem surfaced when adapting the VEGA to the feedback. It is discussed in the interview that partners of AFAS pay a €1.500 partner fee for participating in the partner program. This flow was missed from the initial VEGA where it definitely should have been added. Interestingly enough, the interviewee did not mention this omission. The missing flow was added to the final VEGA.

From this comparison between AFAS and Exact it quickly becomes apparent that the application of VEGAs is limited in this aspect. Modeled in these VEGAs are the most important relationships that interviewed employees of the companies described. The VEGAs are not complete: AFAS too sells some of its software through accountancy firms, as does Exact implements its software in conjunction with some service providers. Mapping a complete set of relationships for a company of this size would present an overwhelming, clew of arrows. In addition, what we refer to here as Exact is largely focused on the Cloud Solutions business unit. While other business units at Exact adopt some if not most of the same practices, differences remain. The VEGA of AFAS completely ignores AFAS Personal, a personal finance platform for consumers, as does the VEGA of Exact not include specific details on Exact's JobBOSS, Macola, MAX, AEC, Insights, Financials, Globe, Manufacturing, Wholesale, etc. A VEGA is insufficient to describe a complete company in a single graph. The analyst must choose what to include and what to exclude, and potentially draw multiple VEGAs for a single companies to make a clear point.

7.3.3 Niche players

Three case studies are conducted that the authors classify as niche players. These companies are nétive[6], Prepped[7] and Advisie[8]. We consider these companies niche players because they operate on the basis of a platform which they do not own, but is developed by a much larger company in terms of revenue, market reach, number of customers and FTEs employed.

[6] http://www.netive.nl/
[7] http://www.prepped.nl
[8] http://www.advisie.nl/

Additionally, these niche players supply products or services that are not included by the hub-company, and are much more specialised in nature. These characteristics of niche players are in line with the classification by Iansiti & Levien [37].

nétive is a product software company with 20 FTE that builds and sells a platform for hiring specialists to fulfill specific roles. The product is sold to large companies which use it to manage resumes and job descriptions, and offers the opportunity to create a buy-side marketplace on which freelancers may bid on open project proposals of the company. The VEGA of nétive is depicted in Figure 7.5.

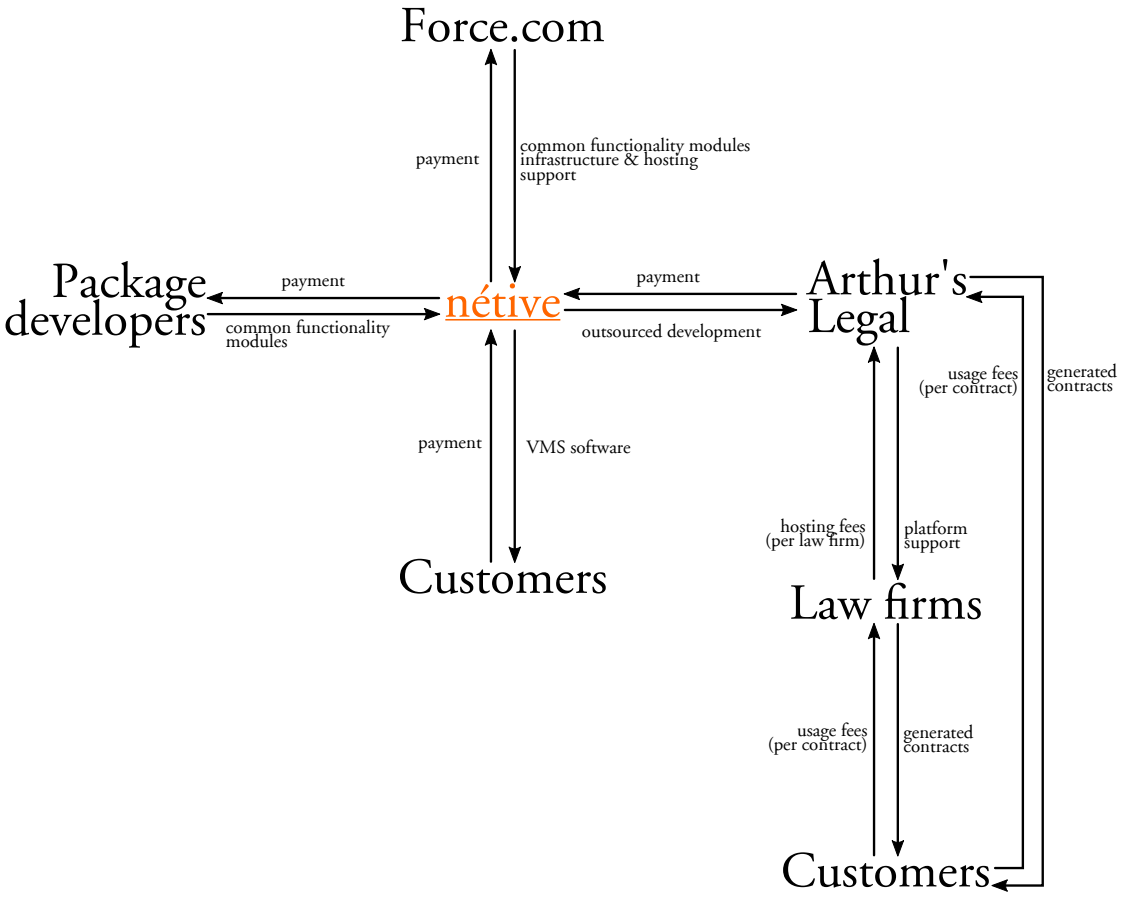

Figure 7.5: VEGA of nétive

nétive builds its software on the Force.com platform, a software development platform built by Salesforce.com (sic.). The platform offer tight integration with salesforce deployments at customers and common functionality such as user authentication and fine-grained authorisation. It also hosts the developed software of nétive and provides support to its developers. In exchange, nétive pays Salesforce.com a usage fee. nétive acquires software with common functionality such as optical character recognition (OCR) from other software vendors which it uses in its products. The finished product (VMS) is delivered to customers in return

for payment. Unfortunately, nétive did not respond on time for its feedback to be included in this report.

The right side of the VEGA of nétive is a special case of historic purposes. A few years ago, when nétive went through a downturn, it started a new project for a law firm named Arthur's Legal. For this law firm, nétive builds a a custom Force.com module with which customers of law firms can generate standard contracts for situations. This is much cheaper than having a lawyer draw up a contract manually and saves the lawyer's time for more interesting work. In the user interface, a single lawyer is indicated as the expert on each type of contract, and he/she can be quickly contacted for questions, or checking or adapting the contract to the situation of the client. Arthur's Legal uses this module on its own Salesforce.com deployment for its current customers, and sells the module to other law firms for use with their customers.

Modeling this extra chain through Arthur's Legal raises some questions on the appropriate depth to model in a VEGA. The product is an important, stable source of revenue for nétive because it is billed as time and materials, and the product is in a mature state and generally needs only perfective maintenance. This enables nétive to work on its own products and when orders are slow, shift more of its developers to the legal module to provide a steady stream of income for the company to last through tougher times. It is there clear that the relationship with Arthur's Legal should be included in the VEGA. However, should the relationships of Arthur's Legal with the law firms and customers be included too? In general, only direct relationships of the central company are modeled in a VEGA. The authors have added these relationships here because it adds to our understanding of the product and to demonstrate the trade-off to be made.

Prepped is a small company (7 FTE) based in Delft, The Netherlands. Its main service is creating custom modules for Sharepoint, a CMS platform created by Microsoft. These modules are not sold as product software, but rather specialised solutions and integrations built for a single customer. The VEGA of Prepped is depicted in Figure 7.6.

Prepped's relationship with its hub is similar to that of nétive though it is more detached. Microsoft provides training and certification to developers of Prepped. This provides value to Prepped as it allows Prepped to prove to its customers that its consultants and developers are well acquainted with the Sharepoint platform. In contrast to nétive, Prepped does not offer to host Sharepoint implementations for its customers. All customers have a Sharepoint installation available, either a hosted solution by Microsoft or a local deployment on their own servers. Prepped acquires common, existing solutions from other software vendors to prevent reimplementation of basic functionality, as does nétive. Unfortunately, Prepped did not respond on time for its feedback to be included in this report.

Advisie is a systems integrator headquarted in Soest, The Netherlands. It employs around 62 FTE. Advisie helps its customer in implementing large software systems centered around administration. Its main partner is Exact, an product software vendor which develops a large software product for financial administration. Additionally, it resells a number of products

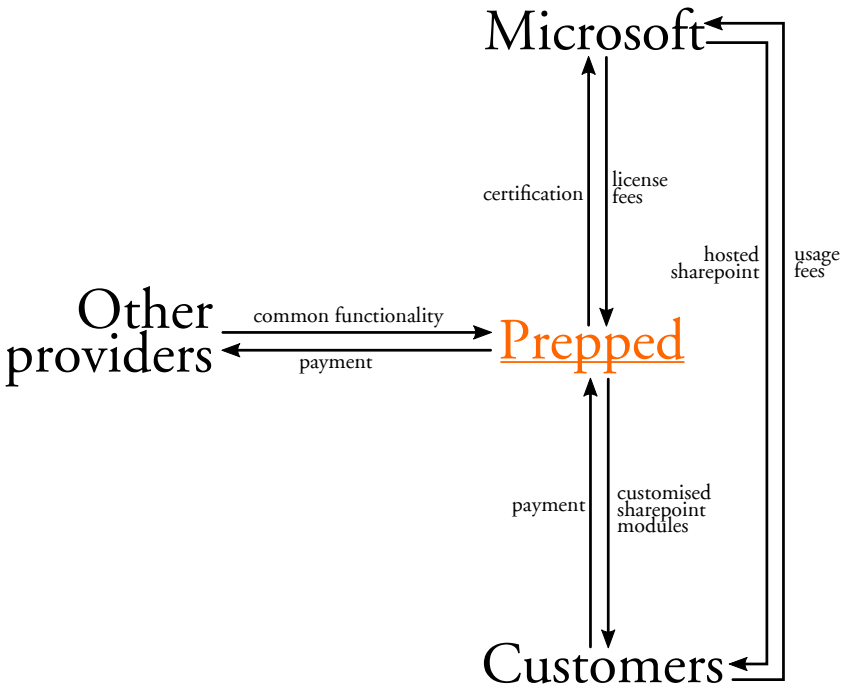

Figure 7.6: VEGA of Prepped

from other vendors such as webshops, OCR, document management, management dashboards and time registration software. Advisie sells and customises these product for its customers, focusing on customer intimacy. A large part of its activities are integrating disjunct software products to increase the efficiency for end-users. Advisie has recently begun to extract common elements from various customers and used these to start creating its own product software solutions. The VEGA of Advisie is depicted in Figure 7.7.

It should be readily apparent to the reader that this VEGA is very similar to that of Prepped (Figure 7.6). Again, Adivisie acquires many customers on the merits of its certification, proving to its potential customers that it has expert know-how of the solutions it resells. Advisie too acquires common functionality from other software vendors and sells custom solutions to its customers.

Responding to the final VEGA, the interviewee of Advisie noticed that the flow of leads is wrong and should be reversed. Leads far more often flow from Exact to Advisie than the other way around. This is in line with the VEGA and analysis of Exact, and the flow has been updated in the VEGA. Missing from the VEGA too is the implementation and customised services that Advisie delivers to its customer. The authors included this in the in the flow "customised software", meaning a collection of software packages that is customised and tightly integrated for a customer and subsequently deployed at the customer. The authors agree in retrospect that this description is woefully inadequate and would have needed a clear explanation at the least to be a clear representation of the situation. As this is considered a

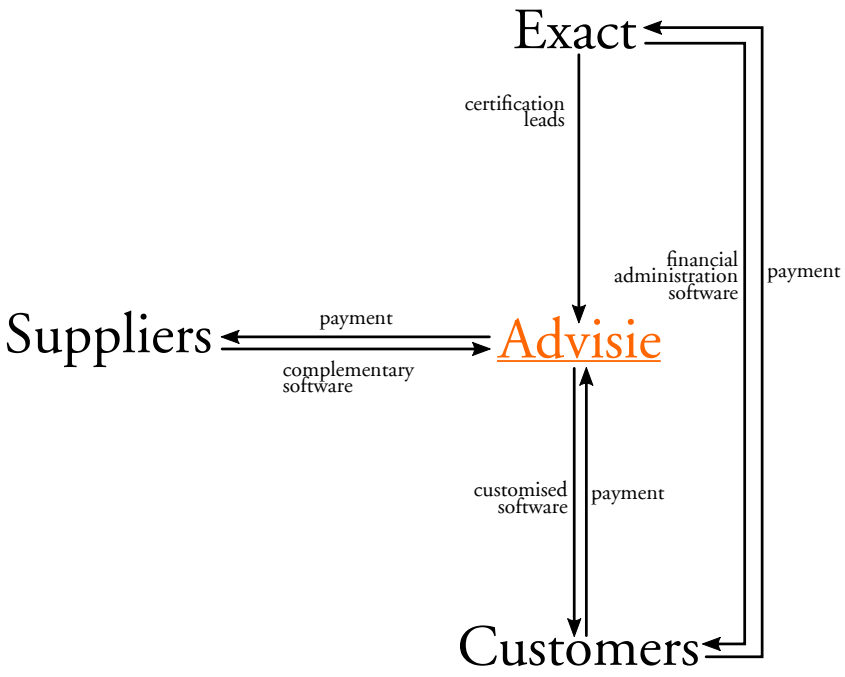

Figure 7.7: VEGA of Advisie

major problem, the VEGA is not updated to reflect it. Overall, the interviewee found this graph to be an accurate representation of the cooperation with Exact, though she remarked that Advisie was more complex than just this relationship.

7.3.4 Networked companies

Networked companies are product software companies which combine properties of both hubs and niche players. The cases included are roos[9], Xpenditure[10] and Nmbrs[11]. These companies develop a software product which integrates with a significant number of partners. In contrast to the niche players, these companies are not tightly dependent on a single, much larger company in a hub-role, instead integrating their product with several competing hubs or a number of partners of roughly the same size in terms of revenue and market reach. The distinction between networked companies and niche players (section 7.3.3) is somewhat arbitrary: no company in these cases is truly dependent on a single other company, and most companies rely on others in varying degrees and in different ways, so there is more of a gray area rather than a black-and-white distinction. One could properly argue that roos should be considered a niche player, and Advisie a networked company. The

[9] `https://halloroos.nl`
[10] `https://xpenditure.com`
[11] `http://nmbrs.com`

authors draw the line here at the impact losing the hub would presumably have on the company. Networked companies are considered to be more resilient to losing a partner than niche players are.

roos is startup based in Amsterdam, The Netherlands and currently consists of a single founder. It builds a platform that offers consumers the option of setting an 'alarm' for a recurring contract or payment. The software than reminds the user at the appropriate date that the contract is up for renewal and that the time to cancel is now, should the user wish to do so. These reminders are coupled with personalised offers based on the information supplied by the user. Users do not pay for the service. The VEGA of roos is depicted in Figure 7.8.

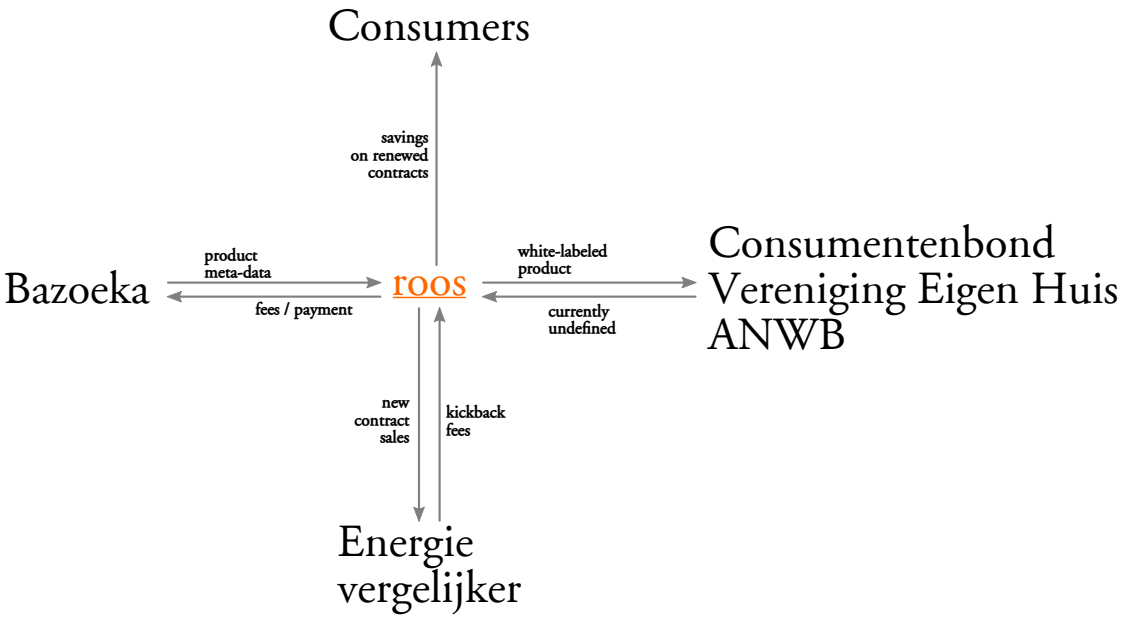

Figure 7.8: VEGA of roos

Consumers benefit from roos, saving money by preventing silent extensions of contracts they intended to cancel. No payment is required to use the service, so no return-flow is modeled. Additionally, the suggestions roos sends to consumers benefit them by presumably utilising savings or bonusses on new contracts of which they were unaware. roos uses two intermediaries BaZoeka and Energievergelijker for the mobile phone industry and consumer energy contracts respectively. These companies aggregate offers from various suppliers through an API. New contracts are closed through these companies, which also pay out the kickback fees to roos. Finally, Roos is in talks with various interested parties such as the Consumentenbond (Consumers Association), Vereniging Eigen Huis (Homeowners Association) and the ANWB (Motorists Association) amongst other parties, to sell the service as a whitelabeled platform for these associations to use. These talks are currently in an early stage and naturally a somewhat sensitive subject, so no further information could be shared

on the exact flows in these future partnerships.

Commenting on the final VEGA, the interviewee at Roos noticed a few spelling errors in company names which were fixed in the final VEGA. The set of partner companies had changed in the meantime so the ANWB was replaced with AEGON, a major insurance company in The Netherlands. The relationship with these partners was judged to be incomplete. Roos receives access to the installed market of these partners, receives data on their users, shares the kickback fees with the partner, and is included in the marketing materials. Apart from this relationship, the VEGA is a very good representation of the software ecosystem around Roos.

Xpenditure is a product software company based in Mechelen, Belgium. It sells a SaaS platform for companies to speed up managing expenses. Expense receipts can be scanned or photographed by employees and imported to Xpenditure's service through various means. These receipts are subsequently processed and can be exported to various administration packages by other suppliers for inclusion in the (financial) administration of the company. The VEGA of Xpenditure is depicted in Figure 7.9.

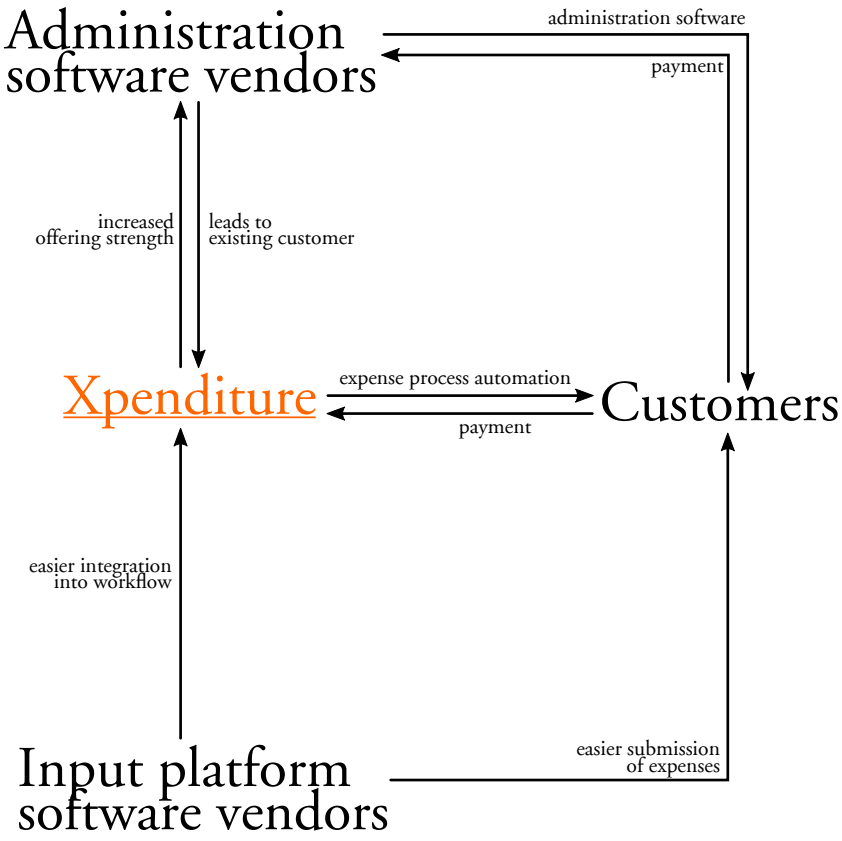

Figure 7.9: VEGA of Xpenditure

Xpenditure helps its customers with speeding up the expense management process, creating

insight and improved handling of expenses and having employees enter receipts while still away and not having to wait to return the office to claim their restitution. It has two main categories of partners to work with: input and administration. “Input partners” are software products/companies that are used to submit receipts to Xpenditure. Xpenditure currently supports submitting receipts through Dropbox[12] and Evernote[13]. These companies have made APIs available that are used by Xpenditure to quickly integrate these platforms for its users. It also supports the import of bank statements from various financial institutions such as American Express, Mastercard, citi, etc. On the other side, Xpenditure integrates its platform with various suppliers of administrative and financial software. Xpenditure allows employees to categorise expenses directly on its platform and enables administrative personnel to export expenses to its financial administration, supporting Quickbooks, Sage one, Freshbooks and Exact, amongst many others. Xpenditure receives many inquiries and sales leads through its presence on the website of these partners, where existing users of the administration software can find its product.

Regarding the VEGA, the interviewee at Xpenditure concluded that the VEGA was an accurate representation of the situation. A single comment was entered on how Xpenditure is attempting to create a bundled offering together with several partners. This was not included in the recording of the interview, though there had been some discussion in broad strokes on working towards a better integration with partners as a major goal. This aspect was subsequently not added to the VEGA.

Nmbrs operates on a similar basis as Xpenditure. It builds a software platform for the administration of HR. Having employees creates a lot of paperwork and administration to process such as sick leave days, vacation, salary slips, documentation, taxes, etc. Nmbrs is active in several European countries, with its headquarters in Amsterdam employing around 60 FTE. It sells its software both directly to companies using it for their HR administration, and to accountancy firms which use it to form the HR administration of third-parties who have outsourced their HR administration to these accountancy firms. The VEGA of Nmbrs is depicted in Figure 7.5.

Nmbrs sells software directly to its customers in exchange for payment for using the software, usually based on the number of employees administered. These customers in the VEGA include both the direct sales and the accountancy firms. Nmbrs offers a number of partners to integrate its software with, which provides the customer with additional benefits such as automatically importing aggregated expense claims into its financial administration. Nmbrs shares sales leads with these partner vendors, with both benefiting. With some partner vendors, Nmbrs forms a more concrete arrangement, paying its affiliates a lead fee for some qualified leads. These arrangements are custom made agreements per partner.

Evaluating the VEGA for Nmbrs indicated that two partners of Nmbrs were missing: integration partners which help customers to integrate existing systems and products with

[12] `https://www.dropbox.com/`
[13] `https://evernote.com/`

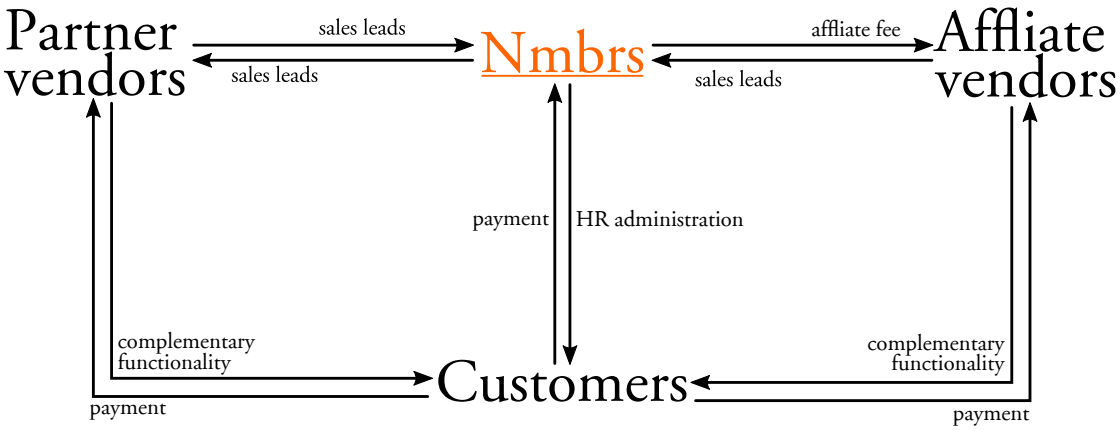

Figure 7.10: VEGA of Nmbrs

Nmbrs, and development partners which help customers to implement Nmbrs in their organisation, converting and importing existing data to Nmbrs. These partners were not discussed in the interview, and are thus not added to the VEGA.

7.4 Results

Ultimately, out of a great many e-mails, ten interviews were conducted with key personnel at these companies. Of the VEGAs drawn, eight are evaluated through the comments of the interviews. From these responses, it becomes clear that while VEGAs can be used to properly model a portion of a software ecosystem, it comes with some challenges of its own. These issues require the analyst to make trade-offs and are mostly considered judgment calls. There is not one correct answer that is appropriate, but the approach taken will vary based on the situation.

Four main issues and/or problems on the application have been discovered in these cases. These are: what to model, limitations on flows, modeling complex VEGAs, and creating clear and concise models.

Choosing what to model in a VEGA is a somewhat difficult and arbitrary choice. It is clear from the case of Datafox Benelux and iHomer that using a VEGA to clearly represent a traditional supply chain is not beneficial. While it can be done and will create an accurate assessment of the relationships of the company and their flows, there are other tools better suited to the task. Most if not all of the flows end up as a variant of "<some product X> in exchange for <payment>". The reader would probably be better off using for example Software Supply Networks [10]. From the nétive case, it is clear that one must make a choice on how far to model the relationships and its flows. In this thesis, the authors usually only model direct relationships of a company, but indirect connections can be included if it

helps the understanding of the reader. Rather than strictly limiting the depth of relationships depicted, the analyst must make a choice on what to include or exclude. Outsourcing of staff business functions such as HR in the iHomer case are not of particular interest to the analysis of a software company, but rather a fundamental choice on how the company operates by its management or owners. One could argue that these should be excluded.

Two limitations of flows are apparent through the Datafox Benelux case. Datafox sells its hardware to its customers via two channels: directly and via third party software vendors, but it never sells its hardware to a customer through both channels. While this may seem obvious, this distinction is not clearly deduced by simply reading the VEGA. Additionally, the indirect channel is much larger and thus far more important to Datafox Benelux than its comparatively minor direct sales channel, but this too is lost in the VEGA as we lack a way to indicate the importance of a flow. Adding this is not as simple as drawing thinner and thicker lines in the VEGA. Partners can have a different appreciation of the relationship, especially when the flow is between partner of unequal size such as nétive and Salesforce.com, or Xpenditure and Dropbox. It also begs the question of what the unit of importance is, and whether the importance can even be quantified directly. Finally, when including very tight relationships in which companies cooperate on many aspects such as in the case of roos with the Consumentenbond/Aegon/Vereniging Eigen Huis and the case of AFAS with its partners, it becomes quite difficult to judge which flows to include and which to ignore. If all flows are to be included, this creates an issue of completeness (are you sure you have got them all?) and, in conjunction with the issue of not indicating the importance of a flow, makes the VEGA more unclear as undue weight is given to very minor flows.

In more complex cases, such as the cases of AFAS and Exact, capturing all flows in a single VEGA becomes nigh impossible. Especially when companies grow to a certain size, its actions might no longer be coherent. For example, two business units of a company might treat the same partners very differently. A VEGA does not support accurately modeling this aspect, except for adding a partner or the company twice, which tends to get out of hand quickly. It is advisable in those case to model the VEGA on a smaller level of detail, perhaps taking as a basis either a business unit of the company, a single product or a product line.

Finally, clearly depicting the situation of a company in a VEGA requires two aspects to be in place: a thorough analysis of the company and additional explanation of key terms. As seen in the iHomer case, the quality of a VEGA is strongly dependent on the thoroughness of the analysis. Creating a suitable VEGA based on a single interview, some background research and other sources is still quite hard to get it right on the first try. From the case of AFAS and Adivisie it is clear that displaying a VEGA without extra explanatory text to further specify the labels of flows is in some cases not sufficient. Only when the situation is relatively simple, and the flows clearly labeled such as in the case of Datafox Benelux, one can expect to not have any confusion as to the interpretation of the VEGA flows.

7.5 Conclusion

VEGAs can be used to sufficiently model real-world scenarios, though some restrictions apply. VEGAs are not ideally suited for describing traditional supply chains. Often, the ideal representation of the situation is not clear and developing a VEGA will require trade-offs or adjustments to create a clear picture of the situation. Combined with the previous two chapters on VEGAs, applications in theory and reasoning over them, this chapter forms the answer to the third and final research question of this research project.

Chapter 8

Discussion and Conclusion

This research project aimed to find look deeper at the concept of "value" in a software ecosystems. Based on an overview of existing literature, a conceptual model was developed to combine scientific views in a comprehensive overview. This conceptual model combines previous work on a variety of topics in software ecosystems such as software ecosystem health, actor roles and relationships, indications of value in a software ecosystem, and strategy for participants in an ecosystem. The model clearly shows that additional research is needed on defining what "value" in software ecosystems actually means, in what forms we could find it, and how value is moved through a software ecosystems by parties through their actions. Three research questions were answered in this project to address these research gaps.

The second research question aims to create a comprehensive overview of what forms of value that could be found in software ecosystems. Searching directly for a vague concept like value, without a solid definition of it, would not yield useful results. To answer this research question, a structured literature review was conducted to gather all kinds of business models in software ecosystems from existing scientific literature. Based on common keywords, a clear search strategy and predefined acceptance criteria, 123 scientific publications were studied for descriptions of business models. Two important limitations apply to this SLR. One, the SLR is not exhaustive and it is possible and even likely that some business models have been missed in the search. Second, there are considerable differences of opinion in scientific literature on what constitutes a business model, which limits the analysis to bits and pieces found in various studies. From the publications, the forms of value were extracted and clustered.

The results clearly clustered around 4 main forms of value on two levels. On the first level, named the resource level, companies cooperate because they require access to each others resources. Those resources are commonly access to the market of a company, or the valuable data it has on its users. On the second level, named the collaboration level, companies work together because it benefits their work. Three main clusters of value appear in this level. First,

complementary opportunities describe the instances in which companies can cooperate to create larger value for their customers. By building new software product for a well-known hardware platform, customers benefit in having more or better functionality available to them, and companies reap the benefits of a greater installed base. In developing a mod for a well-known game, a developer gains access to an existing player base instead of having to built such a base from scratch, and the players gain new content and more playtime from the game they already own. The second and third cluster are named "savings" and "improved effectiveness". These cover the economies of scale that companies can reach by cooperating with each other. These clusters include values such as network effects of a system, reduced time-to-market, and faster development of software resulting from having the right APIs available or adopting common standards. The resulting model (Figure 4.1, page 27) is the answer to the second research question.

The first search question addresses the need for a solid definition of value in software ecosystems. Based on the forms of value found in the structured literature review and the clusters made in the model of the second research question, a definition is constructed. This definition clearly incorporates the key points of value in a software ecosystem, i.e. that is must be of value to at least one actor, and that it must be moved through the software ecosystem. The final definition provides the answer to the first research question:

> *Value in software ecosystems is any product, service, artifact, improvement or right, which is enabled or exchanged through the software ecosystem and provides a tangible benefit to a participant.*

The third research question considers how value is moved through a software ecosystems. As depicted in the conceptual model, the actors in the software ecosystem move value between them by their actions. These actions are guided by the strategies that the actors adopt. To answer this research question, the authors have created new model: the Value Exchange Graph (VEGA). This model is a directed graph structure with the actors as vertices and flows of value between actors as directed edges. A formal meta-model (Figure 5.1, page 31) is defined for the VEGA model with some additional business logic. To evaluate the VEGA model, three steps are taken: 1) the VEGA model is used to explain strategies retrieved from scientific literature, 2) the authors show that the properties of a flow can be reasoned over and 3) the VEGA model is applied to describe the real-world situations of various software companies.

First, existing scientific literature is used to create a list of common strategies applied in software ecosystems (Table 5.1, page 33). Strategies that are suitable for expression in VEGAs are selected and subsequently modeled and discussed. This step shows that VEGAs can be used to model how some theoretical strategies move value through a software ecosystem. Some limitations have become apparent in the execution of this step. Using VEGAs does not work for every strategy; if a strategy does not change any value flow, VEGAs are not a proper choice to model the impact of this strategy. Additionally, some strategies have large effects on an ecosystem in a way that is not apparent from the flow as its core flow is not changed. These strategies too should not be analysed using VEGAs.

Second, the authors show that the flows of a VEGA can be used to reason on these strategies. Using formal reasoning over VEGAs can be used to determine a price point, develop an improved version of a play, and determining which actors stands to gain the most of a play. The application of this reasoning is hampered by requiring a number of assumptions that are not particularly realistic in practice such as perfectly rational actors, a value-function that is hard to resolve for other actors. Even with these assumptions, formal reasoning tends to create insights that could have been reached through other, simpler means.

Third, employees at ten companies in various software ecosystems are interviewed using a semi-structured setup on how their companies operate, especially with respect to their partners. The situation of these companies are depicted into VEGAs by the authors and shown to the interviewees for comment. It emerges that VEGAs can be a good approach to depict real-world situations of companies in software ecosystems, though some limitations become apparent. From the analysis of the cases, it is also clear that applying VEGAs to situations suffers from some limitations. In some, more complex scenarios VEGAs require additional clarification to avoid confusion, and significant trade-offs in how deep and detailed the VEGA is modeled. Most importantly, clearly depicting a complex situation has proven to be a challenge, and exhaustive VEGAs for larger companies could not be drawn.

Together, these three steps show that VEGAs can be applied to clarify the exchange of value in software ecosystems, both in theory and in practice, and serve as the answer to the third research question.

The definition of value in software ecosystems, the forms of value found in theory and the development of the VEGA come together to address the research gap as defined by the conceptual model. By addressing the research gap with the execution of this research project, the authors hope to have contributed to the field of software ecosystems research, and eagerly await the day to see it used in practice or improved upon.

Bibliography

[1] Thomas a. Alspaugh, Hazeline U. Asuncion, and Walt Scacchi. "The role of software licenses in open architecture ecosystems". In: *CEUR Workshop Proceedings* 505 (2009), pp. 4–18. DOI: 10.1109/FLOSS.2009.5071361 (cit. on p. 24).

[2] Ross Anderson and Tyler Moore. "The economics of information security." In: *Science (New York, N.Y.)* 314.5799 (2006), pp. 610–613. DOI: 10.1126/science.1130992 (cit. on pp. 24, 25).

[3] Cm Armstrong. "Competition in Two-Sided Markets". In: 37.3 (2006), pp. 668–691. DOI: 10.1111/j.1756-2171.2006.tb00037.x. URL: http://discovery.ucl.ac.uk/97844/ (cit. on pp. 32, 33, 38).

[4] David F. Bacon et al. "A market-based approach to software evolution". In: *Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications - OOPSLA '09* (2009), p. 973. DOI: 10.1145/1639950.1640066. URL: http://dl.acm.org/citation.cfm?doid=1639950.1640066 (cit. on pp. 24, 33).

[5] I. van den Berk, S. Jansen, and L. Luinenburg. "Software ecosystems: a software ecosystem strategy assessment model". In: *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM. 2010, pp. 127–134 (cit. on pp. 7, 18).

[6] U C Berkeley et al. "Previously Published Works UC Berkeley". In: (2010). DOI: http://dx.doi.org/10.1289/ehp.1002503 (cit. on pp. 24–26, 33).

[7] Martin Boeker, Werner Vach, and Edith Motschall. "Google Scholar as replacement for systematic literature searches: good relative recall and precision are not enough". In: *BMC medical research methodology* 13.1 (2013), p. 131 (cit. on p. 21).

[8] J. Bosch. "Achieving Simplicity with the Three-Layer Product Model". In: *Computer* 46 (11 2013), pp. 34–39 (cit. on pp. 6, 42).

[9] K. J. Boudreau and K. R. Lakhani. "How to manage outside innovation". In: *MIT Sloan Management Review* (2009) (cit. on pp. 5, 6).

[10] S. Brinkkemper, I. van Soest, and S. Jansen. "Modeling of product software businesses: Investigation into industry product and channel typologies". In: *Proceedings of the Sixteenth International Conference on Information Systems Development* (2007) (cit. on p. 68).

[11] Alan W Brown and Grady Booch. "Reusing open-source software and practices: The impact of open-source on commercial vendors". In: *Software Reuse: Methods, Techniques, and Tools* 2319 (2002), pp. 123–136. DOI: 10.1007/3-540-46020-9_9. URL: http://link.springer.com/chapter/10.1007/3-540-46020-9%5C_9 (cit. on pp. 24, 33).

[12] Gary Burtless et al. "Information technology, workplace organization, and the demand for skilled labor: firm-level evidence* t". In: February (2002) (cit. on p. 26).

[13] International Institute of Business Analysis. *What is Business value?* http://www.slideshare.net/IIBA/what-is-business-value. May 2012 (cit. on p. 13).

[14] R. Buyya, D. Abramson, and J. Giddy. "A case for economy grid architecture for service oriented grid computing". In: *Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001* (2001). DOI: 10.1109/IPDPS.2001.925033 (cit. on p. 25).

[15] A. Chandler. *Strategy and Structure: Chapters in the history of industrial enterprise*. Doubleday, New York, 1962 (cit. on p. 9).

[16] Henry Chesbrough and Richard S Rosenbloom. "The role of the business model in capturing value from innovation: evidence from Xerox Corporation's technology spin-off companies". In: *Industrial and corporate change* 11.3 (2002), pp. 529–555 (cit. on pp. 24, 25).

[17] Alina M Chircu and Robert J Kauffman. "Reintermediation strategies in business-to-business electronic commerce". In: *International Journal of Electronic Commerce* (2000), pp. 7–42 (cit. on p. 25).

[18] Michael a. Cusumano. "The changing software business: Moving from products to services". In: *Computer* 41.1 (2008), pp. 20–27. DOI: 10.1109/MC.2008.29 (cit. on pp. 24–26, 33).

[19] Qizhi Dai and Robert J Kauffman. "Business models for internet-based e-procurement systems and B2B electronic markets: an exploratory assessment". In: *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*. IEEE. 2001, p. 10 (cit. on p. 25).

[20] Pedro Domingos and Matt Richardson. "Mining the Network Value of Customers". In: *Proceedings of the Seventh {ACM} {SIGKDD} International Conference on Knowledge Discovery and Data Mining* (2001), pp. 57–66. DOI: 10.1145/502512.502525. URL: http://doi.acm.org/10.1145/502512.502525 (cit. on p. 26).

[21] Omar a El Sawy et al. "It-intensive value innovation in the electronic economy: Insights from marshall industries". In: *MIS Quarterly* 23.3 (1999), pp. 305–335. DOI: 10.2307/249466. URL: http://www.jstor.org/stable/249466 (cit. on p. 24).

[22] Joan Feigenbaum et al. "Privacy Engineering for Digital Rights Management Systems". In: *Security and Privacy in Digital Rights Management* 2320 (2002), pp. 1–30 (cit. on pp. 24–26).

[23] Karl Fogel. "How To Run A Successful Free Software Project - Producing Open Source Software". In: (2009). URL: http://www.amazon.com/How-Successful-Free-Software-Project/dp/1441437711?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1441437711 (cit. on pp. 24, 33).

[24] J. Gehanno, L. Rollin, and S. Darmoni. "Is the coverage of Google Scholar enough to be used alone for systematic reviews". In: *BMC medical informatics and decision making* 13.1 (2013), p. 7 (cit. on p. 21).

[25] George M. Giaglis, Stefan Klein, and Robert M. O'Keefe. "The role of intermediaries in electronic marketplaces: Developing a contingency model". In: *Information Systems Journal* 12.3 (2002), pp. 231–246. DOI: 10.1046/j.1365-2575.2002.00123.x (cit. on pp. 24–26, 33).

[26] D. Giustini. *Sure, Google scholar is ideal for some things*. http://blogs.ubc.ca/dean/2010/05/sure-google-scholar-is-ideal-for-some-things/. May 2010 (cit. on p. 21).

[27] N Gold et al. "Understanding service-oriented software". In: *Software, IEEE* 21.2 (2004), pp. 71–77. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=1270766 (cit. on p. 25).

[28] J. Gordijn. "Value-based Requirements Engineering: Exploring Innovative e-Commerce Ideas". PhD thesis. Vrije Universiteit Amsterdam, 2002 (cit. on pp. 24–26, 33).

[29] Jim Gray. "Distributed Computing Economics". In: March (2004), p. 6. DOI: 10.1145/1394127.1394131. arXiv: 0403019 [cs]. URL: http://arxiv.org/abs/cs/0403019 (cit. on pp. 24, 25).

[30] Paul Grefen, K Aberer, and Heiko Ludwig. "CrossFlow: cross-organizational workflow management in dynamic virtual enterprises". In: *Computer Systems Science & Engineering* 15.5 (2000), pp. 277–290. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=4092175$%5Cbackslash$nhttp://infoscience.epfl.ch/record/54419/files/CSSE%5C_00.pdf (cit. on p. 24).

[31] E. den Hartigh et al. "Measuring the health of a business ecosystem". In: *Software Ecosystems: Analyzing and managing business networks in the software industry*. Ed. by S. Jansen, S. Brinkkemper, and M. A. Cusumano. Edward Elgar, Cheltenham, UK, 2013 (cit. on pp. 4, 18).

[32] Richard E Hawkins. "The economics of open source software for a competitive firm". In: *NETNOMICS: Economic Research and Electronic Networking* 6.2 (2004), pp. 103–117 (cit. on pp. 24, 25, 33).

[33] E. Von Hippel. *Democratizing innovation*. MIT press, 2005 (cit. on p. 5).

[34] Donna L. Hoffman, Thomas P. Novak, and Patrali Chatterjee. "Commercial Scenarios for the Web: Opportunities and Challenges". In: *Journal of Computer-Mediated Communication* 1.3 (2006), pp. 1–21. DOI: 10.1111/j.1083-6101.1995.tb00165.x. URL: http://doi.wiley.com/10.1111/j.1083-6101.1995.tb00165.x (cit. on pp. 24, 25).

[35] S. E. Hove and B. Anda. "Experiences from conducting semi-structured interviews in empirical software engineering research". In: *Software Metrics, 11th IEEE International Symposium*. IEEE. 2005, pp. 10–23 (cit. on pp. 16, 18).

[36] M Iansiti and R Levien. "Keystones and dominators: Framing operating and technology strategy in a business ecosystem". In: *Harvard Business School, Boston* (2004) (cit. on p. 4).

[37] M. Iansiti and R. Levien. *The keystone advantage: what the new dynamics of business ecosystems mean for strategy, innovation, and sustainability*. Harvard Business Press, 2004 (cit. on pp. 4, 6, 7, 9, 14, 15, 32–34, 42, 54, 61).

[38] Marco Iansiti and Roy Levien. "Keystones and dominators: Framing operating and technology strategy in a business ecosystem". In: *Harvard Business School, Boston* (2004) (cit. on pp. 24, 25).

[39] Jacqueline. *No, Google Scholar Shouldn't be Used Alone for Systematic Review Searching*. https://laikaspoetnik.wordpress.com/2013/07/09/no-google-scholar-shouldnt-be-used-alone-for-systematic-review-searching/. 2013 (cit. on p. 21).

[40] S. Jansen, S. Brinkkemper, and A. Finkelstein. "Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems". In: *Proceedings of the first International Workshop on Software Ecosystems*. 2009 (cit. on pp. 24–26, 33).

[41] S. Jansen and M Cusumano. *Defining software ecosystems: a survey of software platforms and business network governance*. Edward Elgar Pub, 2013 (cit. on pp. 24, 25).

[42] S. Jansen and M. A. Cusumano. "Defining software ecosystems: a survey of software platforms and business network governance". In: *Software Ecosystems: Analyzing and managing business networks in the software industry*. Ed. by S. Jansen, S. Brinkkemper, and M. A. Cusumano. Edward Elgar, Cheltenham, UK, 2013 (cit. on p. 4).

[43] S. Jansen, A. Finkelstein, and S. Brinkkemper. "A Sense of Community: A Research Agenda for Software Ecosystems". In: *Software Engineering-Companion*. IEEE, 2009 (cit. on pp. 4, 6, 10, 14).

[44] S. Jansen, a. Finkelstein, and S. Brinkkemper. "A sense of community: A research agenda for software ecosystems". In: *2009 31st International Conference on Software Engineering - Companion Volume* (2009), pp. 2–5. DOI: 10.1109/ICSE-COMPANION.2009.5070978 (cit. on pp. 24, 25).

[45] S. Jansen et al. "Shades of gray: Opening up a software producing organization with the open software enterprise model". In: *Journal of Systems and Software* 85.7 (2012), pp. 1495–1510 (cit. on p. 6).

[46] R. J. Kauffman and Bin Wang Bin Wang. "New buyers' arrival under dynamic pricing market microstructure: the case of group-buying discounts on the Internet". In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences* (2001), pp. 1–35. DOI: 10.1109/HICSS.2001.927065 (cit. on p. 25).

[47] B. Kitchenham. *Procedures for performing systematic reviews*. Tech. rep. Keele University, July 2004 (cit. on pp. 15, 18, 20, 21).

[48] K. R. Lakhani and E. Von Hippel. "How open source software works:"free" user-to-user assistance". In: *Research Policy* 32.6 (2003), pp. 923–943 (cit. on p. 5).

[49] David Lucking-Reiley. "Auctions on the Internet: What's Being Auctioned, and How?" In: *Journal of Industrial Economics* 48.3 (2000), pp. 227–252. DOI: 10.1111/1467-6451.00122 (cit. on pp. 24, 26).

[50] K. Manikas and K. M. Hansen. "Software ecosystems--a systematic literature review". In: *Journal of Systems and Software* 86.5 (2013), pp. 1294–1306 (cit. on p. 3).

[51] Jonathan R. Mayer and John C. Mitchell. "Third-party web tracking: Policy and technology". In: *Proceedings - IEEE Symposium on Security and Privacy* (2012), pp. 413–427. DOI: 10.1109/SP.2012.47 (cit. on pp. 24–26).

[52] David G Messerschmitt. "Marketplace issues in software planning and design". In: *Software, IEEE* 21.3 (2004), pp. 62–70 (cit. on pp. 24, 25).

[53] H. Mintzberg, B. Ahlstrand, and J. Lampel. *Strategy Safari: A Guided Tour Through The Wilds of Strategic Mangament*. FreePress, New York, 2005 (cit. on p. 9).

[54] Tyler Moore. *Economics of Information Security and Privacy*. 2010 (cit. on pp. 24, 25).

[55] MSU. *PubMed, Web of Science, or Google Scholar? A behind-the-scenes guide for life scientists. : So which is better: PubMed, Web of Science, or Google Scholar?* http://libguides.lib.msu.edu/pubmedvsgooglescholar. Feb. 2015 (cit. on p. 21).

[56] M. P. Papazoglou and D. Georgakopoulos. "Service-oriented computing". In: *Communications of the ACM* 46.10 (2003), pp. 24–28. DOI: 10.1109/WISE.2003.1254461. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=1607964 (cit. on p. 25).

[57] Sloan Working Paper, Eric Von Hippel, and Karim Lakhani. "MIT Sloan School of Management". In: May (2000), pp. 1–39 (cit. on p. 24).

[58] Edieal J. Pinker, Abraham Seidmann, and Yaniv Vakrat. "Managing Online Auctions: Current Business and Research Issues". In: *Management Science* 49.11 (2003), pp. 1457–1484. DOI: 10.1287/mnsc.49.11.1457.20584 (cit. on p. 25).

[59] K.M. Popp and R. Meyer. *Profit from Software Ecosystems*. Synomic Gmbh, 2010 (cit. on pp. 15, 32, 33).

[60] M. E. Porter. *Competitive strategy: Techniques for analyzing industries and competitors*. 1980 (cit. on pp. 7, 9).

[61] M. E. Porter and M. R. Kramer. "Creating shared value". In: *Harvard business review* 89.1/2 (2011), pp. 62–77 (cit. on p. 5).

[62] Danny Quah. "The weightless economy in economic development". In: *CEPR Discussion Paper 417, London School of Economics* March.March (1999) (cit. on p. 25).

[63] Ralf Reichwald, Frank T Piller, and Kathrin Möslein. "Information As a Critical Success Factor for Mass Customization or: Why Even a Customized Shoe Not Always Fits". In: *ASAC-IFSAM 2000 Conference* (2000), p. 10 (cit. on pp. 24, 26).

[64] Jean-charles Rochet and Jean Tirole. "Platform Competition in Two-Sided Markets". In: (2002), pp. 1–47 (cit. on pp. 25, 27, 33, 38).

[65] Alvin E. Roth. "The Economist as Engineer: Game Theory, Experimentation, and Computation as Tools for Design Economics". In: *Econometrica* 70.4 (2002), pp. 1341–1378. DOI: 10.1111/1468-0262.00335 (cit. on pp. 25, 26).

[66] Toni Ruokolainen, Sini Ruohomaa, and Lea Kutvonen. "Solving service ecosystem governance". In: *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC* (2011), pp. 18–25. DOI: 10.1109/EDOCW.2011.43 (cit. on pp. 24, 25).

[67] Markku Sääksjärvi, Aki Lassila, and Henry Nordström. "Evaluating the software as a service business model: From CPU time-sharing to online innovation sharing". In: *IADIS international conference e-society*. Citeseer. 2005, pp. 177–186 (cit. on pp. 24–26).

[68] B. F. Schmid and M.a. Lindemann. "Elements of a reference model for electronic markets". In: *Proceedings of the Thirty-First Hawaii International Conference on System Sciences* 4 (1998). DOI: 10.1109/HICSS.1998.655275 (cit. on pp. 24, 25).

[69] Shashi Shekhar, Michael Dietz, and Dan S. Wallach. "AdSplit: Separating smartphone advertising from applications". In: *Proceedings of the 21st USENIX conference on Security symposium* (2012), p. 28. arXiv: 1202.4030. URL: https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final101.pdf$%5Cbackslash$nhttp://arxiv.org/abs/1202.4030 (cit. on p. 25).

[70] M D Smith, J P Bailey, and E Brynjolfsson. "Understanding digital markets : review and assessment". In: *MIT Press* (1999), p. 42. DOI: 10.2139/ssrn.290326 (cit. on pp. 25, 26, 39).

[71] Jayashankar M. Swaminathan and Sridhar R. Tayur. "Models for Supply Chains in E-Business". In: *Management Science* 49.10 (2003), pp. 1387–1406. DOI: 10.1287/mnsc.49.10.1387.17309 (cit. on p. 24).

[72] Mikko Välimäki and Ville Oksanen. "The impact of free and open source licensing on operating system software markets". In: *Telematics and Informatics* 22.1 (2005), pp. 97–110 (cit. on pp. 24, 26, 33, 39).

[73] Mikko Välimäki and Ville Oksanen. "The impact of free and open source licensing on operating system software markets". In: *Telematics and Informatics* 22.1-2 (2005), pp. 97–110. DOI: 10.1016/j.tele.2004.06.008 (cit. on pp. 24, 33).

[74] M.S. Van Houweling. "Cultivating Open Information Platforms: A Land Trust Model". In: *Journal on Telecommunications and High Technology Law* (2002) (cit. on p. 39).

[75] I. van de Weerd and S. Brinkkemper. "Meta-modeling for situational analysis and design methods". In: *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*. Ed. by M. R. S. Syed and S. N. Syed. IGI Global, 2009 (cit. on p. 14).

[76] C. Wohlin et al. *Experimentation in Software Engineering*. Springer, 2012 (cit. on p. 18).

[77] Eric Yu and Stephanie Deng. "Understanding software ecosystems: A strategic modeling approach". In: *CEUR Workshop Proceedings* 746 (2011), pp. 65–76 (cit. on p. 25).

[78] Vladimir Zwass. "Electronic Commerce: Structures and Issues". In: *International journal of electronic commerce* 1.1 (1996), pp. 3–23 (cit. on p. 25).

List of Figures

List of Tables