Relationship Intimacy in Software Ecosystems: A Survey of the Dutch Software Industry

Joey van Angeren
Department of Information and Computing Sciences, Utrecht University
j.vanangeren@cs.uu.nl

Vincent Blijleven
Department of Information and Computing Sciences, Utrecht University
vbblijle@cs.uu.nl

Slinger Jansen
Department of Information and Computing Sciences, Utrecht University
s.jansen@cs.uu.nl

ABSTRACT

Software vendors depend on suppliers to provide the underlying technology for domain specific solutions. As a consequence, software vendors cooperate with suppliers to deliver a product. This cooperation results in supplier dependence, but also leads to opportunities. We present the results of an exploratory research based on twenty-seven case studies, identifying supplier strategies and resulting trade-offs. Strategies range from fully depending on large software ecosystem orchestrators to a minimal dependency strategy. Furthermore, we identify factors at play when selecting suppliers for different components. These factors include; ecosystem health indicators, product and license type and intensive support and maintenance flows. The results presented in this paper can be used by software vendors to assess their software supply network to review supplier relationships, but also for future research.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development; K.1 [**The Computer Industry**]

General Terms

Theory, Design, Management

Keywords

software ecosystem, supplier relationship, supplier selection, software supply network, product deployment context

1. INTRODUCTION

The Dutch product software industry is flourishing and is playing an important role in the Dutch economy. Various examples of successful products are computer games, administrative software and enterprise resource planning products. Product software is defined as; "a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market" [13]. Often, a software vendor will not develop all of these components in-house, rather there are other organizations supplying them with these components, services and intellectual property. Software vendors thus become dependent on other organizations in order to leverage their products to the customer. We refer to the network of actors that is the result of this as a software ecosystem. A software ecosystem is defined as; "a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them" [9].

Software ecosystems are studied from different scope levels. These different scopes involve different abstraction levels and perspectives [8]. In the most detailed scope, a specific software ecosystem is addressed while on the total opposite relationships between different ecosystems are studied. In this paper, we use the middle scope, Software Supply Networks, or in short SSNs. This scope level comprises the network of actors of hardware, software and service organizations that cooperate to satisfy market demands [6]. The SSN provides insight into first-tier buyer-supplier relationships for one specific software vendor, including the resulting dependencies. A Product Deployment Context (PDC) for a product this software vendor, that describes the structure of software products in its running environment in a stack view, showing how the product is composed out of different components [1] can provide further insights. Combining these insights into how important a certain component is perceived to be for the portfolio of a product, including the grade of dependency on suppliers can be used to identify weaknesses. This is useful to gain insight into the factors at play when selecting a supplier. Furthermore, this is valuable information to be aware of for a software vendor, to make the right decisions on both strategic business as well as software architectural level.

In this paper we address supplier relationships from a portfolio perspective. We will propose a matrix to classify components and their perceived grade of importance for the final delivered product. This matrix will be utilized to perform a pattern analysis on the SSNs, PDCs and tables that describe the grade of intimacy of relationships with suppliers of Dutch product software companies. These data have been gathered through twenty-seven case studies. We will examine different supplier relationships, strategies and dependencies. Furthermore, we will address the choices for supplier strategies from a software ecosystems perspective by addressing reasons to join an ecosystem as a customer to obtain software components or services, or even as a partner.

This paper continues with a description of the research

MEDES'11 November 21-24, 2011, San Francisco, USA

approach and data selection process in section two. In section three, we will present a matrix to classify components of a software product and its direct running environment. In the fourth section, we present the results of the case studies concerning organizational size, delivery models and software ecosystems. An analysis and interpretation of findings from these results, will be presented in section five, including several benchmarks with similar studies. In the last section we draw the most important conclusions, discuss generalizability and provide suggestions for future research.

2. RESEARCH METHOD

We use empirical data gathered from the Dutch product software industry. We chose for a multiple case design [14]. The data were gathered between September and November 2010. The dataset consists of twenty-seven case studies.

2.1 Data Collection

The data collection took place during the Product Software course at Utrecht University, which is part of the bachelor in Information Science curriculum. Twenty-seven couples of bachelor students conducted a case study at a Dutch product software company. A prerequisite for a software vendor to qualify for participation was that their number of employees had to be at least ten. Furthermore, companies need to be registered at the Dutch Chamber of Commerce. During two or three meetings, information was gathered about three key themes; organizational structure, business models and software ecosystems. Each assignment addressed one key theme and was graded separately. The case studies were carried out using open questions to provide a broader insight into motivations software vendors have for strategic decisions. All couples used a similar approach.

The first part of the data collection was edged on gathering information about the participating software vendors, such as the products they offer and the way in which the company is organized. The middle part of this case study was edged on business and revenue models. Data were captured by filling the business model canvas defined by Osterwalder [10]. Furthermore, accompanied by a representative of the company, they filled in a SWOT Matrix [12].

Software ecosystems were addressed in the last part of the case study. For each of the companies, one product of interest was selected. For this product, the Software Supply Network and Product Deployment Context was described, using the modeling techniques described by Boucharas, Jansen & Brinkkemper Boucharas2009 and Brinkkemper, van Soest & Jansen [2]. The labels in the PDC match with those in the SSN, this way it is visible what component is supplied by a certain supplier. The SSN also includes other suppliers of components, services and content. The SSN gives an overview of all exchange of products, services, data and money amongst actors. Furthermore, the perceived level of intimacy for each of the actors has been collected in a table.

2.2 Selection Criteria

To enhance the quality and integrity of the dataset, we formulated a number of inclusion criteria for the contributions. The contributions to be included into the dataset we use had to apply to the following criteria: (1) All three assignments have to be handed in and need to be accessible; (2) The average grade for the entire contribution has to be at least 7,5 on a scale of 1 to 10 where 1 is the lowest possible grade and 10 is the highest possible grade; (3) The grade for each assignment has to be at least 7 or higher; (4) Each assignment has to be entirely executed and complete; (5) Each company can be included only once into the dataset. In total seventeen out of twenty-seven contributions were included into the final dataset.

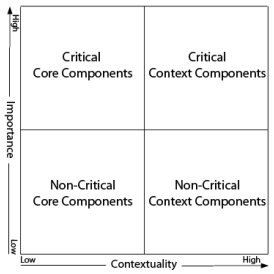

Figure 1: A matrix for the classification of components within a product deployment context

2.3 Data Analysis

The main research question of this paper is as follows; *"How does the perceived level of importance of a component, that is part of a software product, influence supplier selection"?* To be able to answer this research question and to benchmark with previous findings, we perform both qualitative and quantitative analyses on the empirical data gathered. Furthermore, notions from existing literature and about the Product Deployment Context lead to the creation of a matrix that classifies product components.

A brief quantitative analysis is performed to contextualize the dataset. We elaborate on various organizational characteristics, such as organizational size and delivery models. Using these findings as a prerequisite, a qualitative analysis is performed to answer the research question. The SSN and PDC, together with the table that describes the perceived level of intimacy with the actors within these networks, are subject to a pattern analysis. By this analysis, we identify multiple supplier strategies and trade-offs. The matrix presented in section 3 is utilized for further analysis. The analysis is employed to identify factors influencing supplier selection and to determine to what extent these factors are perceived as important when selecting types of components.

3. PRODUCT COMPONENT CLASSIFICATION MATRIX

Product software is composed out of multiple components. Hardware and software products, services and the inclusion of added value by the software vendor will result in a final software product. As already elaborated on by Jansen, Brinkkemper & Finkelstein [7], some components are more easily replaced than others. A plug-in, for example, can be replaced easily within most software products for a product with similar functionality from another supplier without affecting the end product. On the contrary, migrating a software product to be compatible with another operating system is a challenging and time consuming process. Based on this, we developed a matrix that classifies components that are part of the running environment of a product.

The matrix in figure 1 distinguishes four types of components from an architectural perspective. First, we distin-

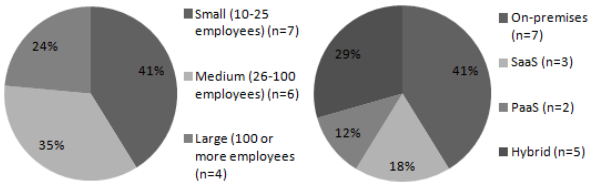

Figure 2: Contextualization of the dataset in terms of company size and delivery models

guish between core and context components. When speaking of core components, we refer to fundamental building blocks of a software product vital to allow the software product to be run and provide no value-added functionalities. On the contrary, context software components are software components that add value. For example, certain functionalities that make the product unique to a customer or that add additional functionality to the product. When examining core and context components, we make a distinction between critical and non-critical core and context components. Components are critical if they are not easily interchanged with another component and contribute to the added value of the product and its overall functionality.

The proposed matrix complies with the PDC modeling approach defined by Boucharas, Jansen & Brinkkemper [1]. Within the stack view of a product and its direct running environment, a distinction can be made between optional and required components. Furthermore, the place a product has within the stack view of a PDC facilitates distinction. Typically products that are low on a stack are core components (e.g. operating systems, frameworks). The description that always accompanies a PDC provides for a last distinction within the four identified categories.

Drawing a PDC implies a degree of abstraction, hence the matrix is created from a simplified architectural perspective. Since the degree of coupling and lines of code necessary to incorporate a certain component into the final product cannot be measured by assessing the PDC, they are not considered factors influencing the classification of components.

4. RESULTS

To conceptualize the dataset, we divided the participants in three distinct company size categories. Companies employing ten to twenty-five employees (small), companies consisting of twenty-sex to one hundred employees (medium) and companies employing more than one hundred employees (large). Furthermore, software vendors were divided by their delivery models; on-premises, Software as a Service, Platform as a Service or a combination of the previous (hybrid). The result of this can be found in figure 2.

The core of the dataset is formulated around Software Supply Networks, Product Deployment Contexts and a table that describes the perceived level of intimacy with the actors within the SSN, accompanied by a textual description to provide insight into the motivation for the decisions made by the company. Out of this data, we distinguish four main supplier strategies, those are included in table 1. When choosing for one or more of these supplier strategies, software vendors are confronted with trade-offs. Choosing to be less dependent, for example, can result in having to dedicate more resources to development or having fewer benefits.

Some companies choose to integrate their products with a hardware component supplier, to streamline the implementation process to reduce complexity. Companies make a trade-off between a streamlined integration process by creating an intimate relationship with a hardware supplier and thus becoming dependent on this vendor, or a less streamlined product integration process but preventing dependency.

Other companies choose to become fully dependent on large software ecosystem orchestrators. Often this is regarded as an opportunity rather than a threat. One of the main reasons companies choose to rely on large software ecosystem orchestrators, is to benefit from niche creation within the ecosystem. In case of heavy dependence, it becomes common to join a partnership model. This model brings additional benefits for both parties, for example, education for employees and shortened maintenance and support lines. Another trade-off needs to be made here, whether to rely on a large software ecosystem orchestrator and benefit from this relationship, or to remain independent.

Software vendors indicate to be reviewing their supplier relationships because of open source software components that can be used as alternatives for proprietary components. With the benefits from open source, such as reduced vendor lock-ins, also come new supplier selection challenges, for example, examining licenses of open source software to avoid liability issues. Having to leverage continuous support and maintenance to customers is perceived as a drawback of open source since more responsibilities end up with the vendor.

The larger companies within the dataset strive for less supplier dependence. Critical components for the leveraged product are developed in-house if resources are available. A trade-off is made between the advantages of not needing additional resources because of being dependent on a supplier, or developing components in-house to decrease direct supplier dependence. This in contrary to smaller companies that rely on core components from suppliers. They create additional value on top of this to serve a niche.

5. ANALYSIS

In the results we distinguished supplier strategies and their trade-offs, but did not yet discuss factors that influence these decisions. The component classification matrix serves as a basis for analysis of the results. The goal of this analysis is to capture the influence of the type of a software component, as defined in the matrix, on supplier selection criteria and supplier relationships.

With regard to the **perceived level of intimacy**, a relationship with a supplier can be classified as intimate, familiar or unfamiliar. In this sense, the scale ranges from partnerships to ordinary buyer-supplier relationships. As noted in section 4, vendors select suppliers based on their capability to supply components without disruptions, their company size or reputation. Furthermore, they strive for participation in partnership models. They are keen on intimate relationships with large software ecosystem orchestrators, especially when it concerns a critical component supplier. An example of this is the relationship with a platform owner.

Previous indicated motives for supplier selection, show parallels with ecosystem health indicators, defined by Iansiti & Levien [4, 5] as an overall performance indicator of an ecosystem. This is further defined by three determinants; robustness, productivity and niche creation. For supplier selection, especially **robustness** is relevant, since it indicates

Table 1: Supplier selection strategies and their resulting trade-offs

Strategy	Trade-off	
Product integration with a hardware component supplier	Y	(+) Streamlined integration process by working with an "intimate hardware supplier" (-) Become dependent on a supplier
	N	(+) Independent of hardware supplier (-) Less streamlined integration process
Depending on large software ecosystem orchestrator	Y	(+) Benefit from the participation in a partnership model of a large software ecosystem orchestrator (+) Benefit from niche creation (+) Direct contact & support lines with the supplier (-) Become dependent on a large software ecosystem orchestrator
	N	(+) Remain independent (-) No benefits from niche creation (-) Less partnership model possibilities (-) Indirect contact & support lines with the supplier
Inclusion of open source components	Y	(+) Ability to steer an open source software project into a desired direction (+) Less license fees (-) Possible liability issues (-) More support and maintenance responsibilities
	N	(+) Avoid liability issues (+) Less support and maintenance responsibilities (-) Few strategic influence on the development of components
Minimal dependency on suppliers	Y	(+) Develop components in-house to decrease direct supplier dependencies (-) More resources required
	N	(+) No additional resources required (-) Remain dependent on suppliers

the capability of an ecosystem to face and survive disruptions. Software vendors indicate that for them, **continuity of a software ecosystem and its products** is vital when selecting a supplier for valuable components of a product. Related to this, are some of the indicators to measure ecosystem health, as defined in [3]. The **visibility of a software ecosystem within the market** is indicated as another reason to choose for a certain supplier, especially for large organizations supplying core components. Furthermore, **niche creation** is an important reason to join a software ecosystem, especially when selecting a platform. A medium sized software vendor, for example, chooses to fully depend on Microsoft or SAP, providing additional functionality on top of their platforms to serve niches within the ecosystem.

Choosing for different **product types**, in the form of open source components, becomes more interesting for software vendors. Hence, **open source licenses** need to be examined when selection suppliers. Inclusion of any component with a restricting open source license can have consequences for the terms of use or redistribution of the total software product. Ruffin & Ebert [11] discuss several legal aspects and three major risks related to this and how to mitigate them during product development. The first risk concerns integrating open source software into the source code, and then reproducing or selling the product without permission of the licensor. As a consequence, the licensor can claim damages or force the vendor to stop production, delivery and sale. Secondly, because open source software is often a compilation of code from many sources, it is difficult to identify which parts, if at all, relate to protected intellectual property. Infringement of patents of third parties or intellectual property rights can be the consequence. Third, when an open source tool is used to, for instance, generate tool-created comments, the resulting work might be considered a derivative work. In this case, the owner of the tool will be given certain rights to the product. Software vendors thus indicate to be cautious with the inclusion of open source.

Several large companies, mentioned that for certain components they prefer open source over closed source because it results in less supplier dependence. They did not, however, mention anything about possible copyright violations related to inclusion of open source software, making them vulnerable for the risks as described in the beginning of this section. Only a few software vendors indicate to employ strict policies about which open source licensed components can be included into a product.

Support and maintenance flows play a decisive role when selecting both critical core and context component suppliers. For critical components an organized flow is essential. Therefore, continuity of maintenance and support and direct lines with suppliers are identified as triggers for supplier selection. These maintenance and support interactions make supply management in the (product) software industry different from this practice in other industries [7]. As a result, partnering with these suppliers becomes interesting to shorten these lines, especially when these support and maintenance are additional services that a software vendor offers to create additional revenue streams.

Table 2 classifies the results presented in this section. We created this table to identify the factors that are important when selecting a supplier for a certain type of component and what the average grade of intimacy is for this supplier relationship. It is important to note, that selection criteria related to functionalities or added values of a certain component and costs are not included in this table, since they are applicable to all categories and are therefore trivial.

Table 2: Classification of factors influencing supplier relationships and selection per product component type

Category	Factor	Critical core component	Non critical core component	Critical context component	Non critical context component
Supplier related factors	Perceived level of intimacy	Intimate	Familiar	Intimate	Unfamiliar
	Continuity	Y	Y	Y	N
	Visibility within the market	Y	Y	N	N
	Niche creation	Y	N	N	N
Product related factors	Product & license type	Y	Y	Y	Y
	Support & maintenance	Y	N	Y	N

6. CONCLUSION AND DISCUSSION

In this paper, we addressed supplier selection and strategies out of a software ecosystems and portfolio perspective, using data gathered through twenty-seven case studies with Dutch product software companies. Software products consist of multiple components, only part of which developed in-house. We created a matrix that classifies components that are part of the direct running environment of a product. First of all, a distinction between core and context components was made. Core components are the essential building blocks of the product while context components provide additional functionalities resulting in added-value. In a second level of decomposition, we distinguished between critical and non-critical components. A critical component is a component that is not easily interchangeable or that adds significant added value to the product. The presented matrix can be valuable from both a Product Deployment Context perspective as well as a system architectural perspective.

Four main supplier strategies were identified. Some software vendors choose to become fully dependent on a large software ecosystem orchestrator. While increasing supplier dependence, it brings opportunities because of a perceived guarantee of continuation of support as well as offered additional benefits. On the total contrary, software vendors opt for a minimal dependence. Software vendors also indicated to be reviewing current supplier relationships because of the advent of open source. Having to leverage continuous support and maintenance, however, is experienced as a drawback of open source. In addition, open source software licenses need to be reviewed to avoid risks associated with the inclusion of open source components within a product.

Software vendors employ criteria when selecting suppliers. Critical core component supplier relationships have a high perceived level of intimacy compared to non-critical component suppliers. In addition, suppliers are selected based on their software ecosystem health, a performance indicator of an ecosystem. Also, support and maintenance plays a role when selecting both core and context component suppliers, especially when a component is classified as critical.

The generalizability of the results is limited, because of the relatively small number of case employed. More research needs to be employed to provide further evaluation and validation. Furthermore, studies need to be addressed on software ecosystem selection to gain insight into mechanisms that are at play when deciding on what ecosystem to join.

7. REFERENCES

[1] V. Boucharas, S. Jansen, and S. Brinkkemper. Formalizing software ecosystem modeling. In *Proceedings of the 1st International Workshop on Open Component Ecosystems*, pages 41–50, 2009.

[2] S. Brinkkemper, I. van Soest, and S. Jansen. Modeling of product software businesses: Investigation into industry product and channel typologies. In *Information Systems Development*, pages 307–325. Springer, 2009.

[3] E. den Hartigh, M. Tol, and W. Visscher. The health measurement of a business ecosystem. In *Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting*, 2006.

[4] M. Iansiti and R. Levien. *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability.* Harvard Business School Press, 2004.

[5] M. Iansiti and R. Levien. Strategy as ecology. *Harvard Business Review*, 82(3):68–78, 2004.

[6] S. Jansen, S. Brinkkemper, and A. Finkelstein. Providing transparency in the business of software: A modeling technique for software supply networks. *Advances in Information and Communication Technology*, 243:677–686, 2007.

[7] S. Jansen, S. Brinkkemper, and A. Finkelstein. Component assembly mechanisms and relationship intimacy in a software supply network. In *15th International Annual EurOMA Conference, Special Interest Session on Software Supply Chains*, 2008.

[8] S. Jansen, S. Brinkkemper, and A. Finkelstein. Business network management as a survival strategy: A tale of two software ecosystems. *1st International Workshop on Software Ecosystems*, 505:34–48, 2009.

[9] S. Jansen, A. Finkelstein, and S. Brinkkemper. A sense of community: A research agenda for software ecosystems. In *ICSE 09: Proceedings of the 31st ICSE Conference on Software Engineering*, pages 187–190, 2009.

[10] A. Osterwalder and Y. Pigneur. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers.* Wiley, 2010.

[11] C. Ruffin and C. Ebert. Using open source software in product development: a primer. *IEEE Software*, 21(1):82–86, 2004.

[12] J. L. Ward and J. Peppard. *Strategic Planning for Information Systems.* Wiley, 3rd edition edition, 2002.

[13] L. Xu and S. Brinkkemper. Concepts of product software. *European Journal of Information Systems*, 16(5):531–541, 2007.

[14] R. K. Yin. *Case Study Research: Design and Methods.* Sage Publications, Inc, 2008.