Modeling of Product Software Businesses: Investigation into Industry Product and Channel Typologies

Sjaak Brinkkemper, Ivo van Soest, Slinger Jansen

Institute of Information and Computer Sciences

Utrecht University, the Netherlands

{S.Brinkkemper, I.Soest, S.Jansen}@cs.uu.nl

www.cs.uu.nl/groups/OI

Abstract. The product software industry lacks a method for describing their products and business models on a high abstraction level. The lack of good methods to model a software product makes it harder to evaluate a business model especially for people with less knowledge of software architectures and more knowledge about the business side of creating software. This paper presents a model consisting of two diagrams the Product Context Diagram (PCD) which describes the context in which a software product operates and the Software Supply Network (SSN) diagram which describes the different parties involved in the delivery and deployment of a software product or service. Furthermore this paper presents a typology for both diagrams. The proposed diagrams are simple and easy to create therefore providing a quick insight to the core of a business model, creating the diagrams supports the process of evaluating the business model.

1 Introduction

The software industry is a relative young industry that is still developing at a high pace (OECD, 2000/2001). It is therefore surprising that there is little research being conducted in the area of business models for software products. The product software industry accounts for substantial economic activity all over the world (OECD 2000/2001). In 2001 the total market of the product software industry was estimated to be 196 billion USD, which is just 9% of the overall ICT spending of 2.1 trillion USD worldwide. "The product software sector is among the most rapidly growing sectors in OECD countries, with strong increases in added value, employment and R&D investments." (OECD 2001). Xu and Brinkkemper (2005) developed the following definition of product software:

"A software product 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."

Product software differs from other types of software in the following two ways:

- One copy is sold multiple times whereas tailor-made software is sold only once.
- The product sold is the software itself in contrast with embedded software where the software comes on a device that is sold as one product.

The software business is a special industry where it costs about the same to make one copy or one million copies of a software product (Cusumano 2004). Investing in a software product can result in substantial productivity gain and strategic advantage (Brynjolfsson 1993), much like in the movie, music and medicine production industries.

Business and product modeling

The starting point of this research is the fact that there is a need within the product software industry to discuss their product options and marketing on a strategic level. For instance, the choice to sell the product directly to customers or only through resellers is a main issue at board level or the decision to change from a proprietary product DBMS to a standard DBMS from another vendor. The software industry has developed several methods and models to describe their products on a very technical level. Some examples of these established methods are ARIS (Scheer 2000), and the Unified Process (Jacobson et al. 1999). These models are excellent in describing a product on detailed level but lack the ability to describe a software product to people with less knowledge of software architectures and more knowledge about the business side of creating software.

In the past software companies developed their software completely in house, where today most software companies develop their software around and on top of generic software components supplied by other software companies. This evolution has led to a change in the software industry. Software companies are transforming from mainly self-oriented to firms that are part of a network of other software companies (Farbey and Finkelstein, 2001). Being part of a network also creates dependencies between the companies in a network. These dependencies are becoming more important but the software industry lacks a method to identify and visualize the networks behind a software product.

Weill and Vitale (2001) developed an e-business model schematic that is a graphical representation highlighting important elements that comprise an e-business model. Although e-business is related to product software the schematic of Weill and Vitale lacks specific features to model the supply chain of a product software company. Nevertheless parts of their schematic are closely related to our research.

Jansen and Rijsemus (2005) showed that platform providers in a software supply network determine the speed of development, not only for their own products, but also for those of third parties. Furthermore, Jansen et al (2007) describe software supply network models for a company that provides "printing" machines for software products, enabling retail stores to always have the latest version of a product available and create product CDs, manuals, and boxes on demand. These works have a strong focus on the software and product development process, instead of business models, however.

In this paper we propose an integrated business model that defines and evaluates a software product. The proposed model must be considered as an addition to the existing modeling techniques. Were other techniques like BPMN, DFD, UML etc have technical and process approaches our proposed model focuses on the business and (inter) organizational aspects of a software product.

This model consists of two diagrams, being the Product Context Diagram (PCD) and the Software Supply Network (SSN) diagram. We show several example models and present validation through an extensive case study.

Research Method

As stated earlier there is hardly any research in the field of business modeling for product software companies. Therefore this research can be qualified as design research. Vaishnavi and Kuechler (2004/6) formulated the following description for design research related to Information Systems: "Design research involves the analysis of the use and performance of designed artifacts to understand, explain and very frequently to improve on the behavior of aspects of Information Systems."

The design cycle consists of five process steps: (a) problem awareness, (b) suggestion, (c) development, (d) evaluation, and (e) conclusion. This research project was carried out to the according design cycle. The starting point of this research was the question whether there is a need for a uniform business modeling method for a product software company. The research problem therefore is defined as: How to define and evaluate a set of models that describes the business model of a product software company?

To develop a functional model a literature study has been conducted to find related research and models. With the knowledge gathered from literature study during the suggestion step it was possible to develop the proposed model. To evaluate the model several case studies have been carried out according the case study methods of Yin (2003). These case studies resulted in models for a number of software products. The product software companies evaluated the created model by determining if the model reflects their actual situation. After the evaluation it was possible to assess

whether the proposed model was able to describe the real world situation. Furthermore it was possible to generate product and channel typologies and propose suggestions for further research.

In the following section the two diagrams that were proposed earlier are presented. In section 3 we present one case study for which the diagrams are applied. In section 4 we present a product and channel typology based on the generic patterns found in the case studies. Finally, in section 5 we discuss our conclusions and future work.

2 Business modeling approach

The term business model is a relatively vague term in scientific literature. Osterwalder et al. (2005) reviewed the literature of this subject and also discussed the origin of the term business model. The first occurrence of the term was in a scientific article from 1957. The term became mainstream at the end of the 1990s. Based on a lengthy literature study, Osterwalder et al. (2005) defined the term business model as:

"A business model is a conceptual tool that contains a set of elements and their relationships and allows expressing the business logic of a specific firm. It is a description of the value a company offers to one or several segments of customers and of the architecture of the firm and its network of partners for creating, marketing, and delivering this value and relationship capital, to generate profitable and sustainable revenue streams."

This definition is closely related to our perception of a business model for a software product, except for the fact that our model only looks at the product, the network of direct suppliers and the financial consequences. It does not address issues related to marketing and staffing as they can be treated as derivatives of the aforementioned.

As described earlier our proposed business model for a software product covers two aspects, being technical and organizational. Based on the literature found about business models and software products the decision was made to cover these aspects by making two diagrams namely the Product Context Diagram and the Software Supply Network diagram. In the next two paragraphs we describe both diagram types.

2.1 Product Context Diagram

The Product Context Diagram (PCD) describes the context in which a software product operates. The PCD consists of other software products, which are required for the main software product. Besides software products the product context diagram also shows how the different software products communicate with each other.

The diagram also indicates whether the Product of Interest (PoI) is the product for which the business model is to be made and the PoI indicates whether it is a stand-alone application or one that communicates with other software products via a medium, such as for client-server and Internet applications. If the software product is not a stand-alone application the diagram is extended with at least two extra columns. One column indicates the medium through which the two software products communicate, the second column indicates the software products needed on the other side of the medium. Figure 1 shows an example of a PCD.

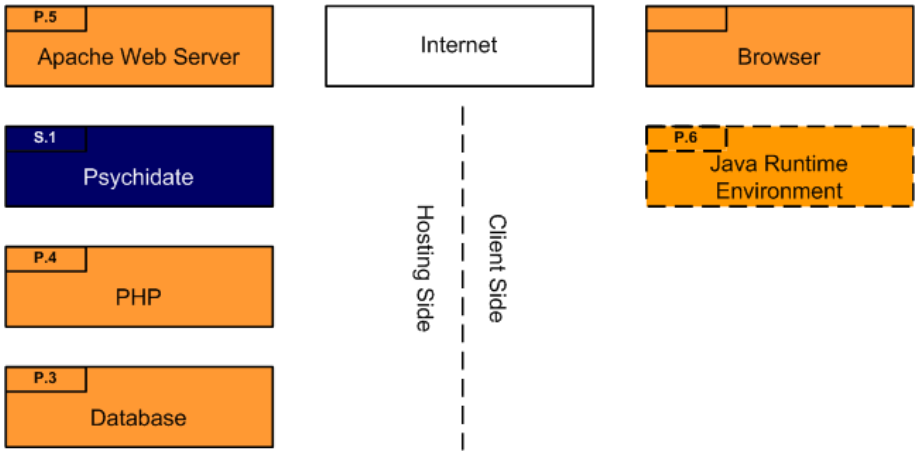
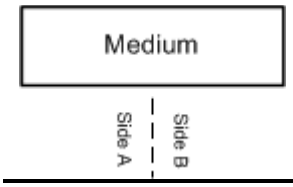


Figure 1: Example Product Context Diagram (PCD)

The PCD is a layered diagram in the sense that it stacks different software products on each other. The way of stacking is determined by the flow of information through the different software products. The PCD differentiates four components:

Table 1: PCD Components

Component	*Description*
P.1 Product of Interest	**Product of Interest**: The main software PoI in the business model. The color of the PoI is blue.
P.X Required Product	**Required Product**: Product required by the PoI to function properly. The color of a required product is orange. Examples: Database and Operating System
P.X Optional Product	**Optional Product**: Product that optionally can be used with the PoI. The color of an optional is orange and the outline is dashed. Examples: Plugins and Products that

	can interface with the PoI **Medium**: Medium which is used by two or more products to communicate. The color of a medium is white and the outlines are solid. Below the medium there is a divider line with a type description on both sides. Examples of media are LAN, WAN and the Internet.

PCD Conventions

For clarity and comparability there is a set of rules for the PCD. These rules are listed below:

Positioning

- The products in the PCD are stacked on each other according to the flow of data or flow of control through the different products.
- A medium connects two product stacks, so a medium is always between two product stacks
- It is allowed to put multiple products on the same level as some products require multiple products to function. The width of the product symbol is adapted accordingly.

Numbering

- The labels and numbering in the left corner of a product are the same as in the software supply network. More information on the software supply network is given in paragraph 2.3.
- In cases where the product is obviously available, such as an Internet browser, the product is modeled without a label in the left corner. These products are necessary for the main product to function but the relation between the supplier of these products and the end-user is not relevant for the Software Supply Network and they do not have a P.* indicator in the left corner. (Please see the browser product in Figure 1: Example Product Context Diagram for an example).

A medium is used to connect two stacks of products. Besides the label indicating the name/type of a medium there are also two labels which describe both sides of product stacks. Common examples of side combinations are client-server, client-hosting and application-database.

2.2 Software Supply Network Diagram

The Software Supply Network (SSN) diagram displays the different parties involved in the delivery and deployment of a software product or service. The different parties in a SSN are connected to each other by trade relationships. To give a better insight in a relationship there are flows attached to a relationship. Figure 2 shows an example of a SSN diagram.

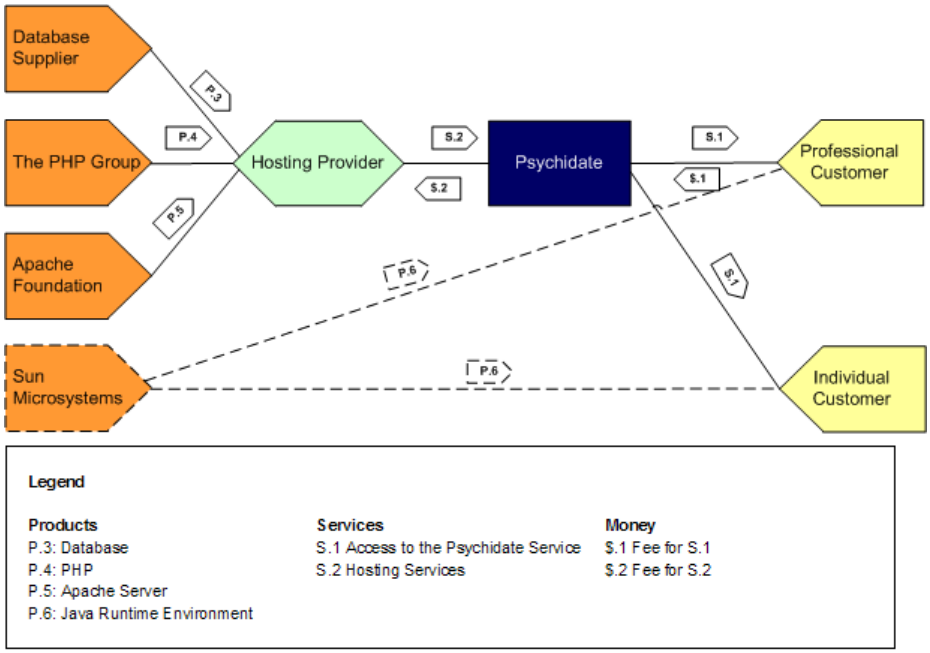


Figure 2: Example Software Supply Network (SSN) Diagram

The SSN diagram differentiates parties, relations, flows, and a legend at the bottom. The different components are listed in the table below.

Table 2: SSN Components

Component	*Description*
Company of Interest	**Company of Interest (CoI):** Company delivering the main PoI in the business model. The color of the CoI is blue.

	**Customer:** Individual or organization that has acquired the PoI. The color of the customer is yellow.
	**Supplier:** Company that direct or indirect supplies a crucial or required part or service. The color of a supplier is orange. Examples: Suppliers of Operating Systems, Database Management Systems, Plugins etc.
	**Intermediary:** Company acting as an intermediary between two or more parties in the SSN. The color of an intermediary is green. Examples are Distributors, Resellers, and Hosting providers.
	Trade relationship: A relation that connects two parties in the SSN containing one more flows.
	**Product Flow:** Software or hardware product which is obtained from a supplier. Examples: DBMS, OS, libraries, servers, etc.
	**Content Flow:** Content which is transferred between two or more parties within the SSN. Examples: XML files, symbol data file,s, game level files, etc.
	**Service Flow:** Service related to the PoI which can be obtained from a supplier. Examples: Support, Training, Consultancy, Access to a system etc.
	**Financial Flow:** Financial transaction between two parties within the SSN. Examples: License Fees, Service Fees, Payments, Subscription Fees.
	**OR operator:** Operator to enable a choice between two or more trade relationships and the related flows. The color of an OR operator is black; the color of the text label is white.

In some situations one or more parties beside the CoI are not necessarily required to deliver the final product to the customers, for which there is an option to make a party optional. Optional parties are represented with the same symbols but they have dashed outlines.

A connector line displays the trade relationship between two parties. Two parties can only have one direct connector line. Connector lines do not have arrow ends to indicate the direction of a relationship. To provide insight and direction of a relationship there are flows which can be connected to a connector line, the direction of the arrow

indicates the direction of al flow. Just like the different parties a relationship and the related flows can be optional. These relationships are also displayed by dashed lines.

In some cases there is a choice between two or more products. For example in the case a customer purchases a product, which needs a DataBase Management System (DBMS). Suppose the supplier of the PoI supports two or more different DBMSs but the product only requires one. For these cases there is the OR operator. When the customer has a relation with the OR operator, the OR operator has two optional relations with the suppliers of the DBMSs. Figure 11 shows an example of the use of an OR operator for the direct and indirect relationship between the vendor and customer.

An SSN diagram is always accompanied by a legend consisting of all the flows in the model and a short description of each flow. The flows should be listed per flow type. Figure 2 shows an example of a SSN legend.

SSN Diagram Conventions

For clarity and comparability there is a set of conventions and rules for SSN diagrams. These rules are listed below:

Positioning of parties and flows

- The CoI is placed in the middle of the diagram.
- Suppliers are placed at the left side of the diagram.
- The customer is placed at the right side of the diagram.
- Flow's going to the right (towards the customer) are placed above of a relation.
- Flow's going to the left (from the customer) are placed below a relation

Scoping

The SSN diagram only shows parties and flows have a direct relation with the PoI. The following types of parties are not in the scope of the SSN diagram:

- Suppliers of basic office related materials required for running a company (office rental, furniture, computers, etc).
- Suppliers of development environments (Visual Studio, Borland Delphi, etc). An exception is when the development environment also delivers a platform/engine which is required for the PoI to function. In this case the development environment can be part of the SSN, but preferably the engine name is indicated instead of the development environment.
- Suppliers of packaging materials for the PoI.

Numbering

Using a uniform numbering for the flows makes it easier to interpret a SSN diagram. Therefore there is a set of rules for the numbering

- The PoI uses the number 1.
- Different kinds of flows that are connected to the same relation and are related to each other use the same flow number. So if for example P.1 is the main product that is sold by the CoI then there should also be a financial flow for the sold products, which is therefore labeled €.1.
- Multiple financial flows should stay separate as far as possible and may not be combined to one flow. In some cases the PoI requires an installation service (S.2) to function properly.
- If a flow is past on by a party without making major changes, the flow keeps its flow number. For example P.2 of supplier X is bought by the customer via a intermediary Y, P.2 will first be connected to the relation between supplier X and intermediary Y, But P.2 will also be connected to the relation between intermediary Y and the customer.
- Flow numbers do not need be numbered consecutively. As described earlier clarity is improved by giving related flows the same number. This can result in the fact that the sequence of numbers for a flow type is disturbed. For example a diagram can have three products flows P.1, P.2, and P.3, where P.1 and P.3 need to be bought; P.2 is distributed at no cost. Because of the fact that P.1 and P.3 need to be bought there will also be an €.1 and €.3 but there will be no €.2.

2.3 Consistency between the PCD and the SSN diagram

PCD and SSN diagrams are closely related due to the fact that all product flows mentioned in the SSN diagram must also be in the PCD. The purpose of PCD is to give a better technical insight in the way different software products cooperate with each other. This relation is formalized by using the same numbering in the PCD and SSN diagram, a product flow mentioned in the SSN for example P.1 should also be mentioned in one of the product stacks in the PCD.

There can be products in the PCD that are not in the SSN. These products are necessary for the main product to function but the relation between the supplier of these products and the end-user is not relevant for the Software Supply Network. Because of the fact that these products are not in the Software Supply Network they do not have a P.* indicator in the left corner. An example of this is a web browser, a product considered to be a standard available for most operating systems. The customer does not have to make an arrangement with a supplier, which is irrelevant for the supplier in the Software Supply Network.

3. Case studies

In total eight case studies at product software companies in the Netherlands were conducted. In this paragraph we will show the results of one case study.

To ensure validity and reliability a case study approach was developed. The case study method is based on the steps suggested by Soy (1997), which are based on the literature of well-known case study researchers Simons, Stakes and Yin. Yin (2003) formulated the following four kinds of validity threats that could influence the research quality when using case studies:

- *Construct validity.* Each case study was conducted in the same way following the case study protocol.
- *Internal validity.* The results of the first interview were checked with the interviewee and a second employee in two separate sessions.
- *External validity.* The cases studies are representative for the product software market in the Netherlands because each company is active in a different domain and each company has a different number of customers.
- *Reliability.* Repeating the case studies will generate the same results except for the situation in which the company has decided to change their business model.

3.1 PhraseWorld

PhraseWorld is a web-application which main functionality is to help the visitor translate phrases from one language into another, by showing pre-translated phrases that are quite similar to or an exact match of the user's input. The content (phrases) of PhraseWorld is generated by the visitors itself. By involving the users in the process of updating the content, the website is kept dynamic and up to date. Furthermore it enables the visitors to contribute to the quality of the service. A community feel makes the visitors feel involved with the website and induces them to keep visiting.

PhraseWorld earns money in several ways the most important are advertisements on the website, lead generation for related products such as language courses and dictionaries and the sale of digital phrasebooks (PDF) based on the content of PhraseWorld. The PhraseWorld is offered as a web application and must be accessed via a webbrowser. PhraseWorld is built on the so-called LAMP (Linux, Apache, Mysql, PHP) platform.

Product Context Diagram
Figure 3 presents the Product Context Diagram of PhraseWorld.

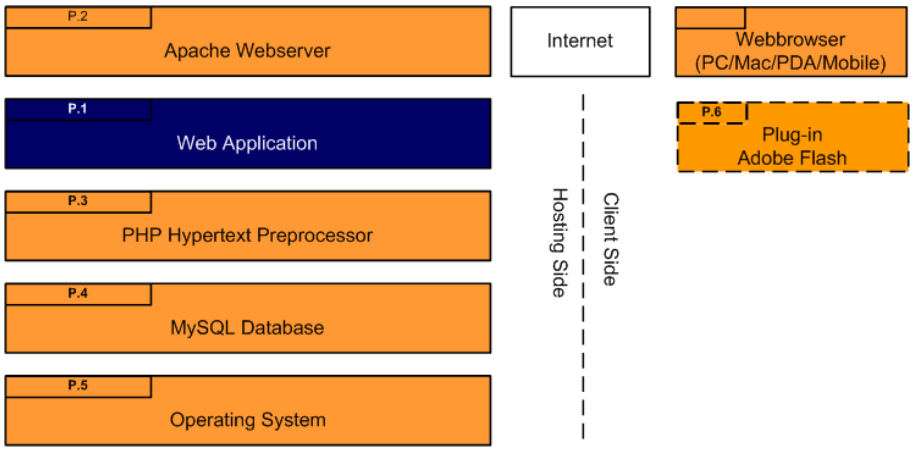


Figure 3: PCD PhraseWorld

PhraseWorld must be accessed with a webbrowser via the internet. Request from the browser are answered by the Apache Webserver (P.2), which sends the request to the PhraseWorld Web Application (P.1). The PhraseWorld Web Application (P.1) uses the PHP scripting Language (P.3) to answer the requests. Furthermore it uses the MySQL Database (P.4) to store the data related to PhraseWorld. The components on the server side run on a Linux Operating System (P.5).

Software Supply Network
Figure 4 shows the Software Supply Network of PhraseWorld.

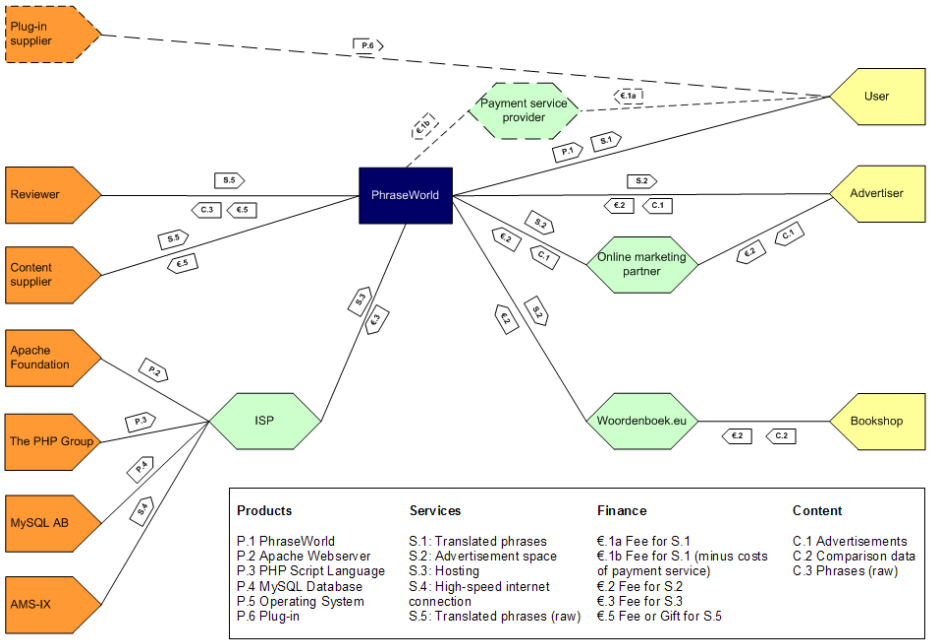


Figure 4: SSN PhraseWorld

PhraseWorld has three types of customers: users, advertisers and bookshops. The user obtains the phrases directly from PhraseWorld. In the case that the user purchases a digital phrasebook from PhraseWorld the intermediary Payment Service Provider is also involved in the transaction because they facilitate the payment.

The Advertiser has two options two advertise on PhraseWorld direct or via an intermediary Online marketing partner.

PhraseWorld also integrates the comparison-function of the Woordenboek.EU into the PhraseWorld application. Via this integration PhraseWorld generates leads for bookshops and language-courses

PhraseWorld has three suppliers which all supply services. The Reviewer and Content supplier help creating the phrase data. The ISP supplies the hosting platform consisting of three software products, PHP scripting language (P.4) from the PHP Group, MySQL DBMS (P.7) from MySQL AB and the Apache Webserver (P.3) from the Apache Foundation.

4. Typology of Generic Patterns

After conducting the case studies we studied similarities between the different cases for each of the two diagrams. The similarities led to a typology of generic patterns for both the PCD and SSN diagrams.

4.1 Typology of product contexts

The PCD evolved during the case studies. Before the case studies there where two types of PCD diagrams namely a one-column diagram for a stand-alone software product and a three-column diagram for a client server software product. Some of the products in the case studies indicated the need for a diagram with five columns. Sorting the cases by the number of columns created a categorization: stand-alone (one-column), client-server / webservice (three columns) and enterprise software (five or more columns).

After categorizing the product by their number of columns in the PCD diagram it was interesting to see what kind of products were in each category and if there are similarities between the products in a category.

Stand-alone Product

A stand-alone software product has quite a small PCD diagram which consists of one column with 1-4 stacked products. The relative small size of the PCD diagram of a stand-alone product does not indicated that product itself is technical less complex compared two other product types. The PoI can contain very advanced algorithms and logic. Figure 5 shows an example of the PCD diagram of a stand-alone software product.

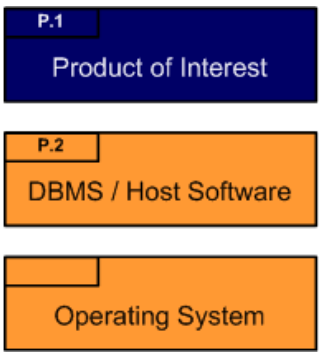


Figure 5: PCD Example of Stand Alone Software Product

The PoI only needs limited number other software products to function. Besides an Operating System (OS) a stand-alone software product some times needs one or two other software products such as for example a DBMS or a Host Software Package. In some cases there are optional products in the PCD for example software products, which can interface with the PoI or that use the PoI as a host such as plugins.

Client-server
A client-server software product has a PCD diagram with three columns. The right column consists of the PoI client, the middle column only consists of a medium (Internet, LAN, WAN, etc) the left column consist of the PoI and the required components. Figure 6 presents an example of the PCD diagram of a Client-server.

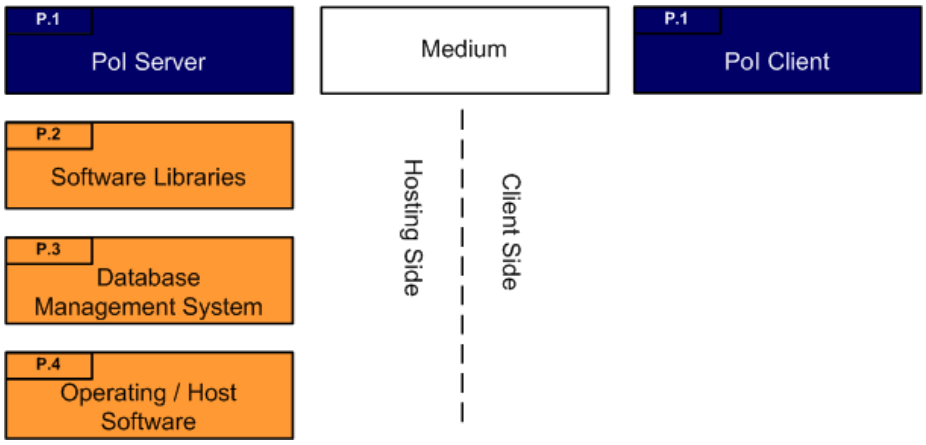


Figure 6: PCD example of a client-server software product

Besides the three columns the PCD for a client-server software product often has more software components in the PoI column. This can be explained by the fact these types of software products are used to centralize data and logic, which often require software such as application, database and file servers. The client side mostly consists of one component, which is some times extended by for example a plug-in or host component.

Web service
The web service software product is related to the client-server software product. The main difference is the fact that a web service does not have a PoI Client on the right side of the diagram. In the web service product the PoI client is replaced by a standard web browser. In some cases the right side of the diagram is extended with a plug-in or optional software product. Figure 3: PCD PhraseWorld shows an example of such an extension.

Enterprise software
An enterprise software product consists of five or more columns. An enterprise software product can be considered as an extension of client-server / web service software. An enterprise software product can be accessed through a medium. The component that accesses the enterprise software is not always directly controlled by a human such as the client / web browser. An enterprise software product is often part of a larger system such as an ERP system.

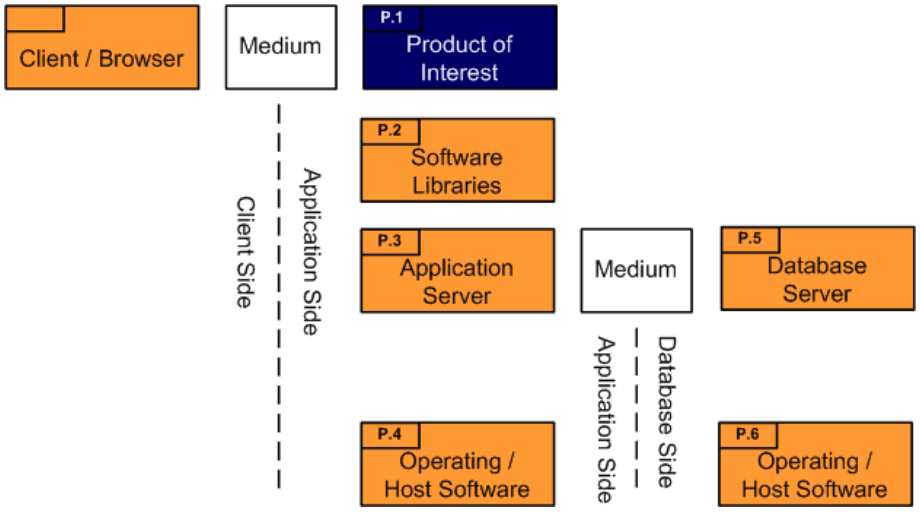


Figure 7: PCD example of an Enterprise software product

The reason behind the use of two or more extra columns is mainly because enterprise software products require more resources which are supplied by multiple physical servers systems, each performing a specific task which in most cases requires different operating systems on each physical server. A common example of this situation is an application server running on a Linux OS and a database server (MS SQL / Oracle) on Windows OS. The use of more columns often results in more software components to facilitate the communication between the different components.

4.2 Typology of Software Supply Networks

The SSN diagrams did not show a clear distinction between different types of software products when we considered the complete diagram. If we separate the SSN diagram into two parts namely the supply part (left of the CoI) and the delivery part (right of the CoI) we do find similarities and the delivery part. The delivery part describes the sale and delivery of a software product to the customer and is often

referred to as the distribution channel. In the case studies the following examples were found:

A. Direct Channel
B. Indirect Channel
    1. Reseller
    2. Agent + Reseller
    3. Value Added Partner / Reseller
C. Direct/Indirect Channel Combinations
    1. Direct + Reseller
    2. Direct + Value Added Partner / Reseller

Direct Channel
This is the most simple and common way of selling a software product. The customer purchases the product direct from the CoI.

Figure 8: Example of a SSN Direct Channel design

Indirect Channel
As the name states there is no direct relation between the CoI and the customer. The delivery of the product goes via one or more third parties / intermediaries. The indirect channel is a common used method of selling software especially shrink-wrapped software. The product stays the same through out the process, the financial flows changes because the reseller takes its margin from the retail price.

Figure 9: Example of an Indirect Channel SSN Agent + Reseller design.

Value Added Reseller/Partner
In this design a reseller creates a new product (combination) by combining the software of the CoI with a product from another supplier before selling it to the customer. The difference between a Value Added Reseller and a Value Added Partner lies in the fact that a reseller adds a product where a partner adds a service to the PoI.

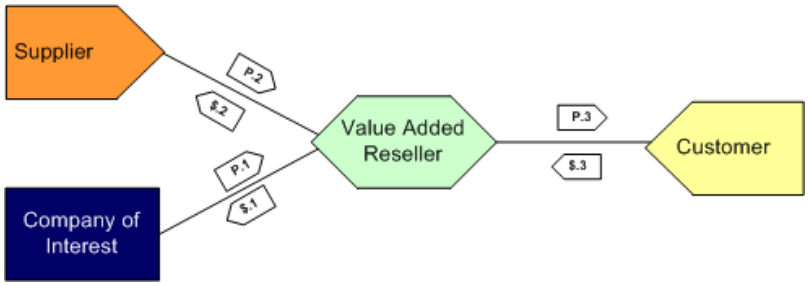


Figure 10: Example of a SSN Value Added Reseller/Partner Design.

Direct/Indirect Channel combinations
It is also possible to combine two of the above mentioned designs. The most common two combinations are:

Direct and Reseller combination, this is a combination of the Direct and Reseller design. The main reason for selecting this design is the opportunity to sell more products with the help of the reseller.

Direct and Value Added Reseller/Partner combination, Figure 11 shows an example of the Direct and Value Added Reseller/Partner Design. In this design two different products are sold namely the PoI (P.1) and the combination product (P.2) of the PoI (P.1) and a second product (P.3). This design is often used because combining and reselling products is not the field of expertise of the CoI.

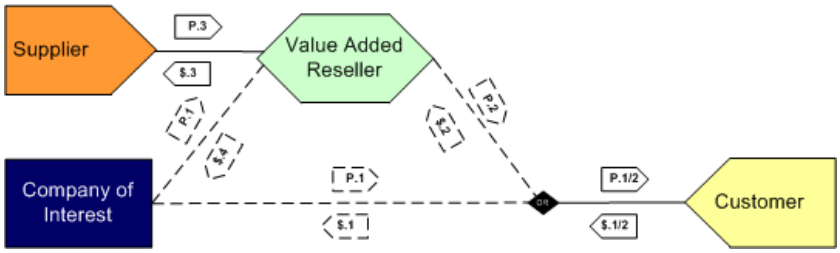


Figure 11: Example of a SSN Direct + Value Added Reseller / Partner design

5. Discussion and conclusions

The main research question in of this paper was formulated as: How to define and evaluate a set of models that describes the business model of a product software company? The research questions were answered in several steps resulting in a business model consisting of the PCD and SSN diagrams.

Since its inception the PCD and SSN diagrams have been used to model approximately 40 businesses. The 8 case studies for this research were evaluated by means of expert validation by company representatives. Furthermore a study was conducted into the gaming industry in the Netherlands which resulted in 5 models for gaming companies. The other models were created during two consecutive years of a course in ICT entrepreneurship at Utrecht University. In this course external evaluators read and assessed business models of IT startups.

The PCD and SSN Diagrams proved to be simple to apply and create, providing a quick insight into the core of a business model. Furthermore, the creation of the diagrams supports the process of evaluating business models. During this process opportunities and threats can be discovered which can result in alternative business models.

Due to the relatively simple structure of PCD and SSN diagrams they can be used to communicate business models to persons without in-depth IT knowledge. The PCD gives insight in the deployment architecture and operational environment of a software product. The SSN diagram also determines the internal organization design. The decision between a direct or indirect channel approach, for example, has a large influence on the organization design. A direct channel approach requires expertise within the CoI on areas like marketing, sales, etc. By making use of an indirect channel approach this expertise is shifted from the CoI to a reseller or another third party. The product and channel typologies help to analyze the alternative business models to make the right decisions when choosing between two business models.

As a company grows its PCD and SSN diagrams tend to change. The diagrams can be used to illustrate and evaluate a developing product and business model of a software company through time. Such diagrams can elegantly demonstrate how product software vendors develop and change their strategies throughout their lifetime. We hope to conduct more research in this area.

In this paper an integrated business model was proposed to define and evaluate a software product. This model was evaluated by doing eight extensive case studies into the product software industry in the Netherlands. The case studies showed that it was possible to create valuable diagrams that reflect the real world situation for eight companies. The case studies showed that there are generic patterns in the diagrams of the business model and were used to develop typologies of software supply networks and product contexts.

Further Research

During the research we encountered various options for further research. As described one of the results of this research is a formal method for creating the diagrams. This formalization is limited to the design conventions and rules presented in

this paper. The method can be taken a step further by formalizing the data related to the objects of a diagram. In the current diagrams the data related to an object is limited to basic information, such as name and price of a product this can be extended with all kinds of other information such as license type, requirements, programming language, etc. This information can be used for detailed analysis on subjects like license structures and dependencies within the SSN. At the moment the diagrams are modeled with Microsoft Visio Stencils. The current modeling method can further be described to create a tool that ensures consistent use of the concepts and modeling techniques.

An important part of a business model is financial analysis. In our research we also studied financial consequences by analyzing the financial flows within the SSN diagram. Results from these analyses were limited, but this subject still remains interesting for further research.

Due to several constraints the number of case studies in this research was limited. More case studies will result in a better understanding and wider classification of business models within the product software industry.

Acknowledgements
The authors wish to thank the companies that participated in this research. Without their help this research would not have been possible. We would also like to thank the students of the Informatics Business course at the Utrecht University. The students provided us with cases and valuable feedback. Furthermore we would like to thank the students of the Junior College Utrecht for their research into the gaming industry in the Netherlands.

References

Brynjolfsson, E., The productivity paradox of information technology. Communication of the ACM, 36(12):66–77, 1993.
Booch, G., Rumbaugh, J., Jacobson, I. (1999). The unified modeling language user guide. Redwood City, CA: Addison Wesley Longman Publishing Co., Inc.
Cusumano, M.A., The Business of Software. Free Press, 2004.
Farbey, B. and Finkelstein, A. “Software Acquisition: a business strategy analysis” Proc. Requirements Engineering (RE01), 2001.
Jansen, S. Brinkkemper, S., Finkelstein A. (2007). Providing transparency in the business of software: a modeling technique for software supply networks. Submitted for publication
Jansen, S. Rijsemus, W. (2005). Balancing Total Cost of Ownership and Cost of Maintenance Within a Software Supply Network, International Conference on Software Maintenance 2005, industry track
OECD (2000). The software sector: Growth, structure and policy issues. OECD Report DSTI/ICCP/IE(2000)8/REV2.
OECD (2001). Highlights of the OECD information technology outlook 2002. OECD Report.

Ostenwalder, A., Pigneur, Y. and Tucci, C.L. (2005). Clarifying Business Models: Origins, Present, and Future of the Concept. Communications of AIS, Volume 15, Article 40

Scheer, A. (2000), ARIS – Business Process Modeling. 3nd ed., Springer, Berlin.

Soy, S.K. (1997). The Case Study as a Research Method published online at http://fiat.gslis.utexas.edu/~ssoy/usesusers/l391d1b.htm, retrieved at 6th of March, 2007.

Vaishnavi, V. and Kuechler, W. (2004/6). "Design Research in Information Systems" February 20, 2004, last updated January 18, 2005. URL: http://www.isworld.org/Researchdesign/drisISworld.htm

Weill P, & Vitale M.R., (2001). Place to space: Migrating to eBusiness Models, Harvard Business School Press, Boston.

Xu, L., & Brinkkemper, S., (2005). Concepts of Product Software: Paving the Road for Urgently Needed Research, In: Proceedings of 1st International Workshop on Philosophical Foundations of Information Systems Engineering (PHISE'05). Orlando Belo, Johann Eder, Oscar Pastor and Joao Falcao eCunha (Eds.), FEUP edicoes, Porto, 2005.

Yin, R.K. Case study research - design and methods. SAGE Publications, 3rd ed., 2003.