2011

A Cost-Benefit-Based Analytical Model for Finding the Optimal Offering of Software Services

Khin Swe Latt
Seoul National University, khinswelatt@gmail.com

Jörn Altmann
Seoul National University, jorn.altmann@acm.org

A Cost-Benefit-Based Analytical Model for Finding the Optimal Offering of Software Services

Khin Swe Latt
Technology Management, Economics, and Policy Program (TEMEP)
College of Engineering, Seoul National University
599 Gwanak-Ro, Gwanak-Gu
Seoul 151-744, South-Korea
khinswelatt@gmail.com

Jörn Altmann
Technology Management, Economics, and Policy Program (TEMEP) & Department of Industrial Engineering
College of Engineering, Seoul National University
599 Gwanak-Ro, Gwanak-Gu
Seoul 151-744, South-Korea
jorn.altmann@acm.org

ABSTRACT

In this paper, we introduce an analytical model for maximizing social welfare, which can be used for finding the optimal offering of a set of software services. The analytical model also explains the impact of service flexibility on customer's selection of business services and on the revenue of service providers. The analytical model is based on a utility model and a cost model. The cost model uses the number of lines of code as the basic measure for cost and applies linear and polynomial cost functions. The utility model is derived from a customer-provider relationship model, which relates the user's utility to the functionality of business services. The result of the analytical model shows that the distribution of functions of an existing business service to a large number of new business services does not generate any additional revenues for the service provider from existing customers. Instead, additional revenue is generated through the offering of business services with fewer functions at lower price. This business services attract customers, which could not afford the original software service of the provider. The result of the analytical model also shows that there is an optimal number of business services that maximizes the net utility of customers.

Keywords

Service-oriented architectures, economics of digital products, business process analysis, business service, customer satisfaction model, software services, service science, customer preferences, social welfare maximization and profit maximization of software service development, cost modeling, economics of service decomposition and service composition.

1. INTRODUCTION

The development of new software services is challenging in three ways: First, software service development is usually costly to the software vendor, who also may face the risk of increasing the complexity for the user and the risk of not achieving service flexibility at all [8]. Service flexibility is defined here as the possibility of users to adapt their business process according to their needs. Second, the existing service systems in enterprises are composed of complicated processes, which are interdependent to each other, making it difficult to separate these service systems into software services [11]. Third, the individually provided services are strongly tied to a specific service provider, which limit service composition (i.e., reduce service flexibility) for customers. This situation is known as the business process silo problem [6].

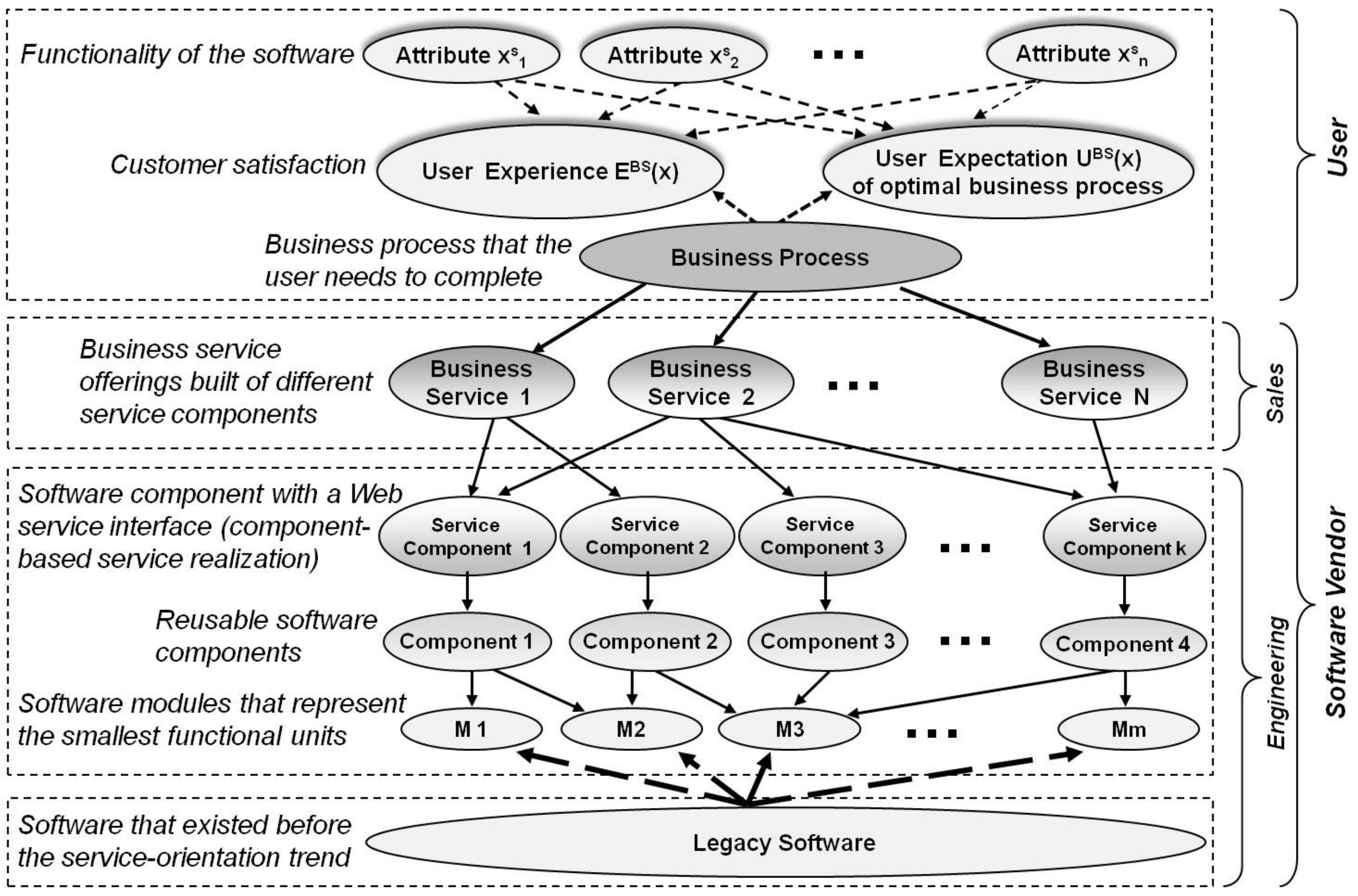
To address these challenges as much as possible, enterprises seek to apply the concept of software reuse. This concept reduces cost, the time to market, and the response time to changes in customer demand. However, it requires the decomposition of existing software into modular software components. To support this, Bennett et al. propose a dynamic service composition architecture [4]. It supports the development of software that is capable to meet changing business needs. An analysis of further software component concepts has been conducted by Kraemer [14].

Service decomposition is useful if a customer requests a new service, which requires a fraction of what the original software service can deliver. It offers an option for substituting a complex service with a set of simple services, potentially reducing costs and improving service flexibility. Decomposition of services into basic services can also benefit service providers. Any change in customer demand does not increase the business risk for the service provider and, therefore, has a low impact on the service provider's business. At the same time, by using these modular software components as basic building blocks, several new and innovative services can be composed [8].

As a preferred technology that supports these concepts, Web services have been chosen. Web services allow dynamic service compositions [15][18]. In a wider context, Web services belong to an emerging technology concept, which is called service-oriented computing (SOC) [14][20].

In order to enable customers of software services to align their business processes with these set of IT services, a new discipline called service science emerged. Besides the technology suites (e.g., Web services), it comprises business process management and performance assessment. Service science also deals with the formalization of interactions between services, allowing enterprises to evaluate the impact of business services on their business processes [23].

10th International Conference on Wirtschaftsinformatik,
16th - 18th February 2011, Zurich, Switzerland



Functionality of the software
Attribute x^s_1
Attribute x^s_2
Attribute x^s_n
Customer satisfaction
User Experience $E^{BS}(x)$
User Expectation $U^{BS}(x)$ of optimal business process
Business process that the user needs to complete
Business Process
User
Business service offerings built of different service components
Business Service 1
Business Service 2
Business Service N
Sales
Software component with a Web service interface (component-based service realization)
Service Component 1
Service Component 2
Service Component 3
Service Component k
Reusable software components
Component 1
Component 2
Component 3
Component 4
Software modules that represent the smallest functional units
M 1
M2
M3
Mm
Engineering
Software Vendor
Software that existed before the service-orientation trend
Legacy Software

Figure 1. Service composition model.

Within this service science framework, we analyze the costs and the value creation of software services. In particular, we propose a conceptual model for software composition and software decomposition (called service composition model), which allows analyzing the value creation (i.e., user utility) in terms of flexibility and cost (Figure 1). In particular, the model conceptualizes the value creation of business processes. The model also exhibits the dependencies of business services on the composition of service components. In particular, the business service functionality depends on the service components composed and their modularized software components. The software modules are the result of the decomposition of legacy software.

As Figure 1 shows, this conceptual model can be divided into a user side and a software vendor side. From the user point of view, the user selects the business services, which generate the highest utility, and integrates (composes) these business services into his desired business process. The selection of business services is based on the user's requirements, which could be a set of functions (attributes). The user's satisfaction is determined by subtracting the user's experience of the delivered business services from his expectation of the business services offered.

From the software vendor (provider) perspective, the provider tries to fulfill the user requirements by designing business services accordingly. This task is part of the sales/marketing department of a software company. These business services are created from service components (Figure 1). A business service is a workflow of service components. The pricing of the business services is up to the sales department [21]. In order to maximize profit, the provider has to set these prices carefully and has to lower the cost of the service components, which are created by the engineering department.

This separation between business service creation and service component creation provides two advantages. First, it enables the provider to respond quickly to changes in user requirements. Second, the decomposition of existing software into services allows the provider to lower the cost of service creation. By decomposing software functionality of existing software into small, atomic service components, these service components can work as basic building blocks for new business services and, therefore, reduce development costs.

In this paper, the service composition model is used to describe the cost and benefits of service composition. We assume that the software vendor decomposes existing complex software into small, independent units (modules), representing a unique and single function. Then, the provider combines these small units into more complex, composite software components in accordance with software development requirements (e.g., software reusability, cost). Consequently, this process incurs costs for service decomposition, costs for combining modules into software components, and cost for wrapping these software components into service components (e.g., Web services), and the costs of building service component workflows. This paper explains these costs in detail and describes how the cost for services, which are composed of small and independent modules of legacy software, can be calculated. Furthermore, this paper explains the costs for composing business services (i.e., the cost for integrating business services into business processes).

The objective is to find a balance between the costs of offering a large number of business services and meeting the requirements of customers, which is a high flexibility in adapting their business processes (i.e., in adapting the workflow of their business services). Within this paper, we provide a solution to this problem by introducing an analytical model for optimizing service offerings.

The remainder of the paper is organized as follows: Section 2 describes the relationship between the customer and software vendor (provider) and explains how customer satisfaction influences the revenues of providers within the software services area. Based on this, utility functions for business services and business processes are introduced. In section 3, we introduce the cost estimation model for software services. It includes the definition of cost functions for business service components, business services, and business processes. The model can deal with workflows. Using this model, section 4 describes the net utility maximization problem, the provider profit maximization problem as well as the social welfare maximization problem for software services. The final section briefly discusses the results and concludes this paper.

2. UTILITY MODEL

2.1 Customer-Provider Relationship

In order to understand the value chain of the service composition model, we develop and analyze a customer-provider relationship model that is based on works of [2][10][13][25]. In particular, it helps defining the framework for the utility model and the cost model.

Vargo & Lusch (2004) proposed the service-dominant logic (S-D Logic) [25], which defines services as the application of specialized competences (i.e., knowledge and skills) through deeds, processes, and performances for the benefit of another entity or the entity itself. S-D logic further assumes that all economies are service economies, all businesses are service businesses, and customer and provider always co-create value (i.e., customers participate in the service creation process). Since the customer-provider relationship can be described as a long-term and dynamic process, the interactions (e.g., pre-sale and post-sale) between customer and provider is very important [13]. Alter (2008) stated that customer satisfaction is affected by the complete set of activities, responsibilities, and experiences that typical customers associate with acquiring, receiving, and benefiting from a particular service [2]. Heskett et al. (1994) discussed about the service-profit chain model, which exhibits the relationships between profitability, customer loyalty, employee satisfaction, loyalty, and productivity [10]. By using these concepts, we develop the customer-provider relationship model for the software services area as shown in Figure 2. It is a more detailed view of the relationship between the user and the software vendor (i.e., sales department), which is shown in Figure 1.

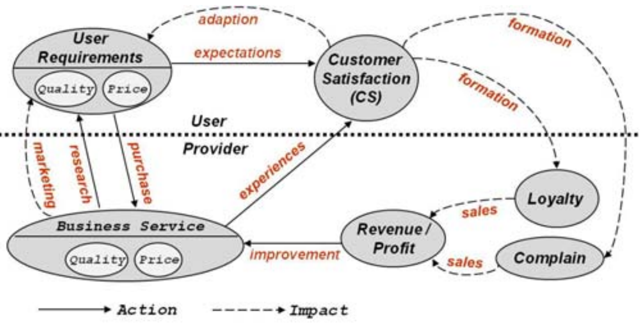


Figure 2. Customer-provider relationship model.

Within this model, customer satisfaction is defined as the satisfaction towards the business services the customer has consumed. The satisfaction is impacted by the user's experience of the consumed business service and the user's requirements for the service. The different components of the general customer-provider relationship model, as shown in Figure 2, are defined as follows:

User requirements are the needs of the user expected to be fulfilled by the provider (El-Kiki & Lawrence [9], and Lee & Ahn [16]). In our model, the user requirement is the degree of flexibility at a certain price. Price is the price of the business service, which is the cost of the service to the customer. Flexibility is defined as the ability to adapt the business process to changes in the business.

Customer satisfaction describes how the customer is satisfied with a business service provided. Customer satisfaction is defined as the difference between the utility from consuming a business service with a certain set of attributes and the expected utility from this business service.

Customer loyalty refers to a consumer's commitment to repurchase a preferred service of the same provider in the future again.

Customer complaint refers to the user's dissatisfaction with a service provided (i.e., the service did not meet the user's requirements).

Business service is the service (product) created by the provider to fulfill user requirements. The user experiences the quality of the service.

Revenue is the income from the business service sold. **Profit** is the gain from selling the services after deducting all expenses incurred through the creation of the service.

Figure 2 shows that providers identify user requirements for business services through market research. Based on this information, providers try to address users' requirements for business services by producing and marketing services in accordance with the user requirements. Created services satisfy the user, if services meet the customer's expectations about quality of service and price.

Customer satisfaction is largely influenced by the net value provided to the consumer. A high customer satisfaction leads to user loyalty to services and providers. Customer loyalty makes the customer decide to repurchase services of the same provider in the future. Consequently, it increases the revenues of providers and retains customers. Customer satisfaction also impacts the user requirements, since satisfaction of a customer determines its expectations of the quality of future services. Customer expectation increases, if user requirements have been met in terms of price and flexibility. Customer dissatisfaction, however, can also lead to complaints of customers. This will happen, if services do not fulfill customer expectations. As a consequence, customers might not continue to purchase the services, reducing the revenue of the provider. However, if customers explicitly express their dissatisfaction, it is an opportunity for providers to improve their service offerings. Concluding, customer satisfaction concerns the improvements of business services and maintenance of customer loyalty in competitive markets.

Based on this model, our customer satisfaction model for software services is defined as the difference between the utility E^{BS}, which the customer gains from a service, and the utility U^{BS}, which the customer expected to get from that service:

$$CS = E^{BS}(X) - U^{BS}(X). \qquad (1)$$

The utility U^{BS} is assumed to be 1. The variable X refers to the vector of all possible attributes x^s_j, where j presents a specific attribute of the business service s. These attributes are assumed to be functional attributes (e.g., business functions) or non-functional attributes (e.g., security, quality of service) and independent of each other.

If the customer satisfaction CS = 0, then the customer is fully satisfied. The provider has exceeded customer expectation, if CS is greater than 0. This is possible, if a provider delivers better quality on the non-functional attributes. For functional attributes, the maximum value is CS = 0, showing that the functionality has been delivered. If CS lies between -1 and 0, the customer expectation has not been fulfilled at all. That means, certain functional attributes have not been delivered or some non-functional attributes have a lower quality than expected by the customer.

2.2 Customer Utility Obtained from Business Services

Using utility functions is most appropriate in this context, since it helps identifying the value proposition of software services. In the past, researchers used utility functions for evaluating resource management approaches [22]. In particular, they used utility functions to measure the performance of management systems. Utility functions also have been applied for achieving QoS-aware service composition. The utility functions were used to select the most appropriate services (Alrifai & Risse [1]). Besides, the utility concept has also been used in decision support systems for scheduling tasks (Yang [26], Jimenez A. et al. [12]). In this paper, we use utility functions to describe the functionality of business processes and business services. The business service attributes experienced by the user determine the overall utility obtained from the business service.

To estimate the overall utility obtained from a business service, we need to define the shape of the utility functions, the range of possible service outcomes, and the weighting of attributes, which expresses the relative importance of an attribute of the business service to a customer. The weights for expressing the relative importance are normalized and the sum of all weights is equal to 1. The relative weights could be determined by the user, using, for instance, AHP or SAW [1][27]. The input parameters (i.e., attributes) are assumed to be independent to each other. Then, after having obtained the weights, the overall value of the utility function for a business service can be estimated, using an additive function.

Customer utility $E^{BS}(X)$ for a business service s, which consists of an vector X of n attributes, can be calculated by multiplying the preference weights h^s_j with the utility $v_j(x^s_j)$ of the service attribute x^s_j. A functional attribute x^s_j is 1, if the business service includes this functionality. Otherwise, it is 0. For functional attributes, the utility $v_j(x^s_j)$ is 1, if $x^s_j = 1$. Otherwise, it is 0. A non-functional attribute x^s_j, which may represent response time or throughput (Menasce & Dubey [19]), is expressed as a real number. The utility function $v_j(x^s_j)$ maps the attribute value onto a scale between 0 and 1. Based on these definitions, the customer utility E^{BS}' can be expressed as:

$$E_s^{BS}(X) = \sum_{j=1}^{n} h_j^S v_j(x_j^S), \quad \text{where} \sum_{j=1}^{n} h_j^S = 1. \qquad (2)$$

Looking at the current situation in the software industry, it becomes clear that customers demand more flexibility in the way how they can use their software purchased from a software vendor. To address this need, software vendors create services that have a reduced number of functions. The functions of those business services can easily be combined. Therefore, assuming the total number of functional attributes to be constant, it can be stated that the higher the number of business services is, the higher the flexibility for the customer is. Equivalently, it can be stated that the utility obtained from the flexibility of a set of business services with the same functions as one single business service is higher than the utility obtained from the flexibility of the single business service. The following equation gives an example, in which a single business service s is split into two business services s1 and s2:

$$E_s^{BS}(X_S) + flx(s) < E_{s1}^{BS}(X_{S1}) + E_{s2}^{BS}(X_{S2}) + flx(s1, s2),$$
$$\text{where } E_s^{BS}(X_S) = E_{s1}^{BS}(X_{S1}) + E_{s2}^{BS}(X_{S2}). \qquad (3)$$

In equation 3, the functions (i.e., functional attributes) offered by the business service s is equal to the functions of s1 and s2. The only difference is in the utility flx(s) and flx(s1,s2) that are obtained from flexibility. The utility of flexibility is larger for s1 and s2 than the utility of flexibility for s.

For calculating the customer utility $E^{BP}(X)$ that can be obtained from a set of business services s involved in the business process BP, we add the utility of all business services, which are purchased by the customer, and the value R that is obtained from executing the business process. $E^{BP}(X)$ also considers the flexibility flx(), which is expressed as the ratio of the number of functions n and the average number of functions per business process. Consequently, $E^{BP}(X)$ can be written as shown in equation 4:

$$E^{BP}(X) = \sum_{s=1}^{m} E_s^{BS}(X) + [\frac{n}{\frac{1}{m}\sum_{s=1}^{m} \sum_{j=1}^{n} x_j^S} - 1] + R, \qquad (4)$$

where m represents the number of business services that are involved in the business process BP. The total number of functions (functional attributes) offered by the service provider is denoted as n. It is assumed to be larger than 0 ($n > 0$) and larger or equal to m ($n \geq m$). The sum of x_j^s calculates the total number of functions of a business service s. If all functions are included in one business service then the service flexibility is 0. If one function is included, then the flexibility equals the maximum, which is n - 1.

3. COST ESTIMATION MODEL FOR SOFTWARE SERVICES

The costs of business services and business processes depend on software components (i.e., service components, or business services) and the composition of those components. Therefore, based on Figure 1, a more detailed illustration of the service composition model, indicating that business processes BP are workflows of business services and that business services BS are workflows of service components, has been developed (Figure 3).

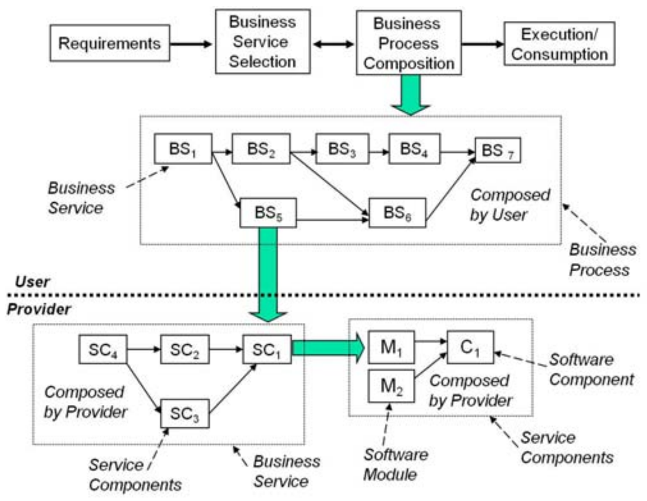


Figure 3. Cost estimation model for a business process.

A service component SC can be obtained by creating a service from a software components C. A software component C is the result of a composition of software modules, which have been obtained from a decomposition of legacy software or have been programmed from scratch. The cost calculation is explained in detail in the following sub-sections.

3.1 Background on Cost Functions

The objective of this cost model is to base the cost estimation on the programmers' effort for developing software services (i.e., for developing modules, software components, service components, business services, and business processes).

Software development cost estimation comprises the entire process of predicting the effort required to develop a software system (Leung & Fan [17]). A precise estimation of cost of a software project can help managers to manage projects adequately.

There are different metrics to size software. Each of those can be used as input to a cost model for software services. The line of code (LOC) is the most popular software sizing metric. For example, Uysal (2008) proposed a COCOMO-based equation that uses the number of lines of code as a parameter [24]. Line of code is also used in a Fuzzy logic model for measuring the software development effort (Attarzadeh & Ow [3]). In this paper, we also use the number of lines of code as the software sizing metric. By counting the number of lines of code, we estimate the provider's effort in providing software services.

A linear cost function expresses cost as a linear function of the number of lines of code. The proposed cost function C for our model is as follows:

$$C(LOC) = A + B * LOC, \qquad (5)$$

where A is the fixed cost of producing software (i.e., provisioning a software service), B is the marginal cost of an additional line of code, and LOC refers to the lines of code in programs [17].

In general, the cost function can also be a polynomial function of the number of lines of code. In this case, the cost function would be defined as:

$$C(LOC) = A + B * LOC^{K}, \qquad (6)$$

where A is the fixed cost of producing software, B is the marginal cost of an additional line of code, and LOC refers to the number of lines of code in the software. K is an empirically derived constant, which has been identified to be in the range between 1.05 to 1.2 [17]. Within this paper, we set the constant to 1.15.

The difference between a linear cost function and a polynomial cost function is the weighting of the number of lines of code. Because of this fact, there is no difference in choosing one over the other. The only implication is its impact on the selection of the appropriate mathematical method for solving the optimization problems that will be introduced in section 4.

Therefore, these cost functions can be applied to calculate the cost for programming modules, software components, service components, business services, and business processes. These cost functions are the basis for providers to decide on how to structure their software. For example, it can be used to decide on how many service components should be created in order to be able to offer many different business services to consumers. Ultimately, these cost functions are the basis for finding a balance between the costs of offering a large number of business services and meeting the requirements of customers, which is a high flexibility in combining services (i.e., the possibility to re-arrange business processes). In the following subsections, the cost functions for service components, business services, and business processes are described in detail.

3.2 The Cost Function of Service Components

Service components are the basic units for composing a business service. They are created by wrapping software components with service interfaces (e.g., Web services). Software components are, as mentioned, a set of software modules that work together (Figure 1 and Figure 3). These different units of software code are considered for defining the cost function of service components.

The cost of a service component is the sum of the cost for wrapping a software component with a service interface (e.g., Web service), the cost for programming the component based on modules, and the cost of the modules themselves. Therefore, the cost of the service component C^{SC} can be expressed as shown in equation 7:

$$C^{SC} = x^{M} * \sum_{i=1}^{k} LOC_i^{M} + x^{C} * \sum_{j=1}^{l} LOC_j^{C} + x^{WC} * LOC^{WC}, \quad (7)$$

where LOC_i^M is defined as the total number of lines of code of module i, LOC_j^C as the total number of lines of code for component j, and LOC^{WC} as the total number of lines of code needed to wrap a software component with a service interface. x^M represents the cost per module line of code (i.e., the unit is [$ / LOC^M]), x^C represents the cost per component line of code (i.e., the unit is [$ / LOC^C]), and x^{WC} denotes the cost per wrapping line of code (i.e., the unit is [$ / LOC^{WC}]).

3.3 The Cost Function of Business Services

The cost function of business services is defined as the sum of the service components costs and the cost for programming the service components workflow. The cost of constructing a business service C^{BS} can be described with the following equation:

$$C^{BS} = C^{SCWF} = \sum_{l=1}^{d} C_l^{SC} + x^{SCWF} * LOC^{SCWF} \ , \quad (8)$$

where C^{SCWF} denotes the cost of programming a workflow of service components, representing the cost of a business service. LOC^{SCWF} represents the total number of lines of code used in programming the workflow of service components. x^{SCWF} is defined as the cost per service component workflow line of code. The unit is [$ / LOC^{SCWF}]. C^{SC} represents the total cost of a service component that is used within the business service (equation 7).

3.4 The Cost Function of Business Processes

The cost function of a business process is defined as the sum of the costs of all business services involved and the cost for programming the workflow with business services. This cost for constructing a business process C^{BP} can be expressed with the following equation:

$$C^{BP} = C^{BSWF} = \sum_{s=1}^{m} C_s^{BS} + x^{BSWF} * [LOC^{BSWF}]^{1.15} \ , \quad (9)$$

where C^{BP} equals C^{BSWF}, representing the cost of programming the workflow of business services. LOC^{BSWF} denotes the total number of lines of code used in programming a workflow of business services. x^{BSWF} represents the cost per business service workflow line of code. The unit is [$ / LOC^{BSWF}]. C^{BS} is the cost of providing a business service, as defined in equation 8. In this case, it is assumed that all business services are build from a disjoint set of service components. In case that business services are build on some identical service components, equation 9 has to be modified such that the C^{SC} of those service components is not counted twice.

Besides, the cost function of business processes is a polynomial function, whereas the two cost functions for programming service components (C^{SC}) and for programming business services (C^{BS}) used linear functions (equation 7 and equation 8). The reason is that the number of service components and software modules is assumed to be fixed. The number of business services used for programming the business process, however, is assumed to be flexible. This is necessary as this paper investigates the effect of the number of business services on the utility of customers and the revenue of providers. A higher number of business services increases the complexity of programming a business process.

4. ANALYTICAL MODEL FOR OPTIMIZING SERVICE OFFERINGS

After having introduced the different cost functions in the previous section, we introduce three optimization problems for software services. The solutions to these optimization problems help answering questions like how many business services should be offered by a software vendor in order to maximize the profit of the vendor.

Economics-based optimization has been proposed by many researchers. Derbel et al. proposed an optimization approach considering user preferences in multi-services IP networks [7]. Yang (2008) proposed a utility-based decision support system, using separate utility functions for time and cost [26]. The objective of optimization in our research is similar. The objective is to maximize the net utility of the parties involved by minimizing the cost and maximizing the flexibility in creating new business services and business processes. In particular, we formulate the following three optimization problems: customer net utility maximization, provider profit maximization, and social welfare maximization.

Within the following optimizations, we only calculate the additional costs that incur if additional business services are offered in addition to an existing software solution. The basis for calculating the additional costs is the cost for providing the software (e.g., a monolithic software solution) that solves the entire business process need of a user.

4.1 Customer Net Utility Maximization

The calculation of the customer net utility that is gained from buying software services can help customers (i.e., businesses) to understand how many business services they should buy. Since a business process generates utility to a customer, we calculate the net utility at the business process level. Therefore, the customer net utility U^{net} is calculated as the utility E^{BP} gained from the business process minus the cost of all business services being used within the business process and the cost for programming the workflow of business processes, as shown in the following equation:

$$U^{net} = E^{BP}(X) - C^{BP} \ . \quad (10)$$

Consequently, the net utility maximization problem can be expressed as:

$$\max[E^{BP}(X) - C^{BP}], \quad (11)$$

where $E^{BP}(X)$ denotes the customer utility from consuming the business process BP. C^{BP} represents the cost for provisioning the business process. X is the vector of all service attributes. Using equations 4 and equation 9, equation 11 can be expressed as:

$$\max_m [\sum_{s=1}^{m} E_s^{BS}(X) + [\frac{n}{\frac{1}{m}\sum_{s=1}^{m}\sum_{j=1}^{n} x_j^s} - 1] + R - [\sum_{s=1}^{m} C_s^{BS} + p^{BSWF} * [LOC^{BSWF}]^{1.15}]]. \quad (12)$$

In our model, the cost of programming the business process workflow is only incurred for the customer. Besides, it is assumed that the number of attributes n is fixed. For simplification, we assume that n attributes represent n functional attributes (e.g., software functions). That means that the n attributes are distributed across m business services. Therefore, the more business services exist, the less attributes (i.e., functions) the business service contains on average.

Figure 4 illustrates the different cost factors, the utility, and the customer net utility U^{net}, depending on the number of business services m. It shows that the cost of programming shifts from the software vendor to the user with increasing flexibility.

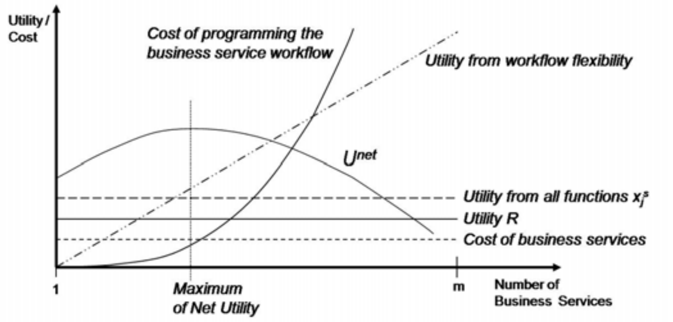


Figure 4. Customer net utility.

In particular, Figure 4 illustrates that there is a net utility maximum with respect to the number of business services. This point is indicated with a vertical line crossing the U^{net} curve. This means that the customer should try to reach this number of business services. Only in this case, the customer can obtain the maximum benefit from service flexibility. The reason is the cost of programming of the business process workflow. It is a significant cost factor that reduces the benefit obtained from service flexibility.

4.2 Provider Profit Maximization

The calculation of the provider profit helps providers to understand the impact of adapting their existing software to a service environment on their revenues. In general, the profit F' of a service provider is defined as the revenue H_S from selling one business service minus the cost C^{BS} of provisioning the business service:

$$F^{'} = H_S - C^{BS} \ . \quad (13)$$

By applying equation 13 to all business services, the profit F can be calculated by adding the revenue H_s from each business services sold minus the costs for provisioning the business services s:

$$F = \sum_{s=1}^{m}[H_s - C_s^{BS}]. \quad (14)$$

A business service comprises different functionality with different costs. In order to increase customer satisfaction (i.e., improve the flexibility for the customer in creating his business processes), the service provider should produce as many business services as possible. However, since the cost for creating business services also increases, an optimum has to be found. The question is how many business services should be produced in order to maximize the profit. Thus, the provider profit maximization problem is:

$$\max_m \sum_{s=1}^{m}[H_s - C_s^{BS}]. \quad (15)$$

Applying equation 8, equation 15 can be re-written as:

$$\max_m [\sum_{s=1}^{m} H_s - \sum_{s=1}^{m}[p^{SCWF} * LOC_s^{SCWF}] - \sum_{l=1}^{d} C_l^{SC}], \quad (16)$$

where m represents the number of business services and d the total number of service components that have been created. H_s is the revenue from business service s. p^{SCWF} denotes the cost for one line of code and LOC^{SCWF} the cost for programming a service component workflow. C_l^{SC} is the cost of service component l.

For our analysis, we assume that the cost of all service components is equal. This is justified if we assume that the entire software is split into code segments of equal size and that the existing software had been structured into components following software management principles from the very beginning. Furthermore, we assume that the total number of service components is fixed. Any business service is created based on these fixed number of service components.

The costs for programming a business service (i.e., a workflow of service components) is assumed as high as the cost for providing the monolithic solution. Because of that, the cost of business service programming increases linearly. (Note: The cost would actually be decreasing with increasing number of business services, since programming effort is shifted to the user. Therefore, this programming effort of the user is depicted as a polynomial increasing curve in Figure 4.) We consider this an upper bound.

The cost is incurred once for each business service, independent of the number of sales of the business service. Therefore, assuming that there is only one customer, who would have belonged to the customer base of the monolithic software solution, the profit from all business services is the same, Y. That means that Y is independent of the number of business services offered by the provider:

$$\sum_{s=1}^{m} H_s = Y \quad \forall m . \qquad (17)$$

These discussion results are illustrated in Figure 5.

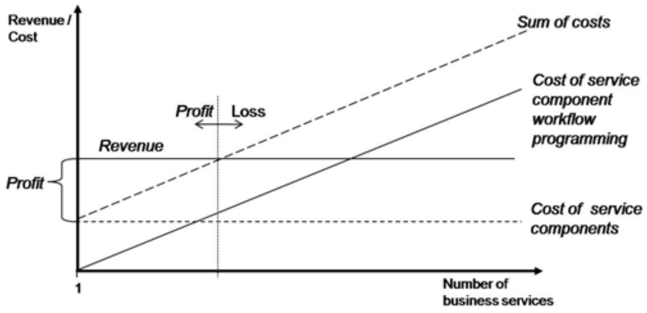


Figure 5. Provider profit in case of a single user.

In particular, Figure 5 depicts that the provider does not get any additional revenue from offering more business services (i.e., from a set of business services with the same functionality as the monolithic software) to its existing customer base. After creating a certain number of business services, the provider would even incur some losses. Even if we would assume that the provider sets the price such that it catches the net utility U^{net} that the customer gets from service flexibility (Figure 4), business service creation were finally to result in losses. Therefore, we can state that adapting existing software to a service environment does not allow extracting any additional surplus from existing customers. It simply helps to increase customer satisfaction and stay competitive within the software market.

However, the revenue of providers H_s will change, if we assume that the software vendor could attract new customers. Those new customers are businesses, who could not afford to buy business services that provide the full functionality but have sufficient funds available to buy business services with less functionality. Figure 6 illustrates the customer distribution with respect to the budget and the number of business services consumed.

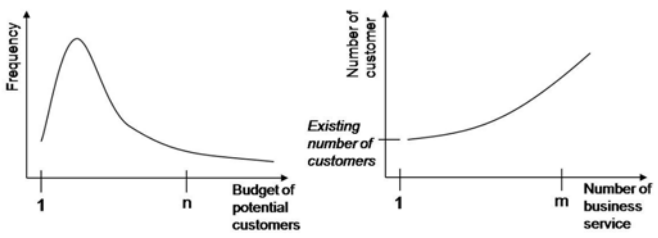


Figure 6. Budget distribution and customer distribution

Although we can only assume the amount of increase in customers through offering new business services (Figure 6), i.e., business services with less functionality than the monolithic software, any increase will have an impact on the revenue of the provider. Figure 7 illustrates this impact. It shows the revenue of the provider, the profit of the provider, and the same cost as the cost shown in Figure 5.

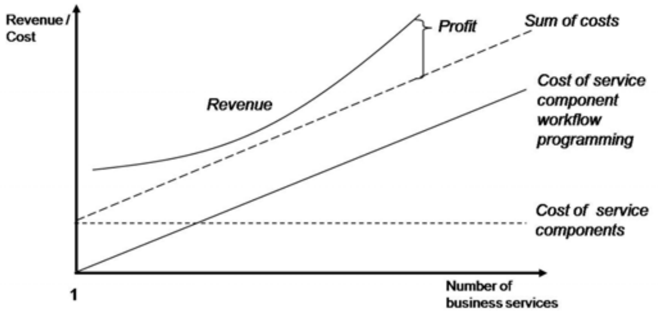


Figure 7. Provider profit in case of many customers.

Figure 7 shows that additional revenue is generated from offering different business services, which differ in their combination of functions. The revenue curve, which is based on the one in Figure 6, is justified, since a large variety of business services can address a larger variety of needs of customers.

4.3 Social Welfare Maximization

Social welfare SW is the sum of all user's benefits (Courcoubetis & Weber [5]), i.e., the sum of all customer surplus and producer surplus. It is the sum of the net utility of all customers and the provider profit PP:

$$SW = \sum_{i=1}^{u} U^{net} + PP . \qquad (18)$$

The equation 18 is equivalent to the utility of all customers minus the costs of programming a business service workflow for each customer, the cost of programming all service component workflows, and the cost of producing all service components. Since the revenue of the provider is equal to the costs of business services for all customers, they are not shown in the social welfare equation:

$$SW = \sum_{i=1}^{u}[\sum_{s=1}^{m} E_s^{BS}(X) + [\frac{n}{\frac{1}{m}\sum_{s=1}^{m}\sum_{j=1}^{n} x_j^s} - 1] + R_u] - \sum_{i=1}^{u}[p^{BSWF} * [LOC^{BSWF}]^{1.15}] - \sum_{s=1}^{m}[p^{SCWF} * LOC_s^{SCWF}] - \sum_{l=1}^{d} C_l^{SC}, \quad (19)$$

where the first term of the equation represents the utility of the customers, while the second, third, and fourth term shows the three cost factors. Note, the utility used is the sum of the utility of all u customers that are served by the service provider. R is assumed to be an average utility across all customers. Then, the social welfare maximization can easily be formulated as:

$$\max_{m} SW . \quad (20)$$

Since the utility function is concave and the cost function is convex, the social welfare maximization problem can be solved by applying the Lagrange approach.

Having the same assumptions as in the previous two sections, Figure 8 illustrates the social welfare from offering business services within this service system.

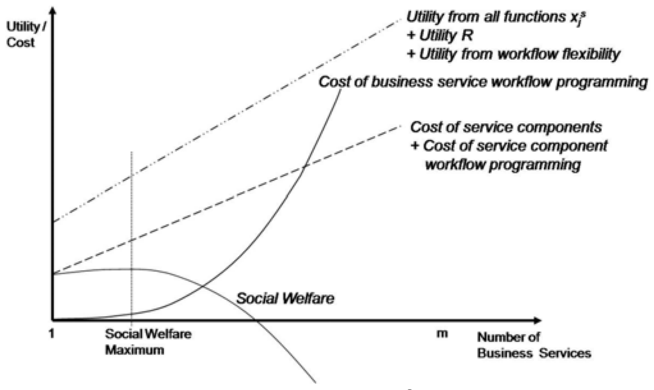


Figure 8. Social welfare.

In particular, it illustrates that in a service-based industry, the flexibility of services and costs of delivering the flexibility impact the social welfare of the system of customers and provider. In order to improve the social welfare of customers and provider, it is necessary to balance the amount of functionality offered with business services. That means, on the one hand, customer satisfaction and profit increase can be achieved by splitting functionality into many different business services. On the other hand, the cost for splitting the existing software into many business services increases the cost for consumers. Therefore, the provider has to find a set of business services that satisfies the customers' demand for flexibility at a reasonable cost. Only then, a pre-requisite for a successful software service industry has been fulfilled.

5. DISCUSSION AND CONCLUSION

In this paper, we introduced a cost-based analytical model for analyzing software service provisioning in a service system. The benefit of this study is the gain in understanding of how customers and service provider interact in an on-demand service-oriented business environment, how costs vary with respect to the customer requirement in flexibility, and how to optimize the consumption and provisioning of business services.

The analytical model is based on a customer-provider relationship model, a utility model, and a cost model for service provisioning. The customer-provider relationship model describes the interdependencies between customer satisfaction, user requirements, business services, and service quality. The utility model defines the customer utility functions for business services and business processes. The utility function allows describing service composition with respect to customer satisfaction for functional and none-functional attributes. The cost model describes in detail the cost of business processes, business services, and business service components. It explains how the costs of services are incurred, especially focusing on costs of service workflow creation. The service provisioning cost is assumed to depend on the programming effort, which is measured in units of line of code.

In particular, the analytical model describes the interdependency between revenue and service provisioning. It explains the conditions under which an increase in the number of service offerings would be profitable and under which an increase would incur losses. The model also explains how to determine the maximum of the customer net utility.

Finally, the analysis of the social welfare of the service system shows that it is necessary to balance the amount of functionality offered per business service. On the one hand, customer satisfaction and profit increase can be achieved by splitting functionality into many different business services, meeting customers' demand for flexibility. On the other hand, for consumers, the cost of integrating business services into business process has to be limited by offering consumers a minimum set of business services. Only if both is achieved, social welfare is maximized.

6. ACKNOWLEDGMENTS

This research was supported by the International Research & Development Program of the National Research Foundation of Korea of the Ministry of Education, Science and Technology of Korea (Grant number: K21001001625-10B1300-03310).

7. REFERENCES

[1] Alrifai, M. and Risse, T., 2009. Combining global optimization with local selection for efficient QoS-aware service composition. *18th International Conference on World Wide Web, WWW2009.*

[2] Alter, S., 2008. Service system fundamentals: work system, value chain, and life cycle, *IBM Systems Journal*, 47, 1.

[3] Attarzadeh, I. and Ow, S.H., 2009. Software development effort estimation based on a new fuzzy logic model. *International Journal of Computer Theory and Engineering*, 1, 4.

[4] Bennett, K., Layzell, P., Budgen, D., Brereton, P., Macaulay L., and Munro, M., 2000. Service-based software: the future for flexible software. *IEEE APSEC 2000, Asia-Pacific Software Engineering Conference*, IEEE Computer Society, 214-221, Singapore.

[5] Courcoubetis, C., Weber, R., Coe, M., 2003. Pricing communication networks: economics, technology, and modeling, Wiley-Interscience Series in Systems and Optimization, Wiley, ISBN 978-0470851302.

[6] Demirkan, H., Kauffman, R. J., Vayghan, J. A., Fill, H.-G., Karagiannis, D., and Maglio, P. P., 2008. Service-oriented technology and management: perspectives on research and practice for the coming decade. *Electronic Commerce Research and Applications*, 7.

[7] Derbel, H., Agoulmine, N., and Salaun, M., 2007. Service utility optimization model based on user preferences in multiservice IP networks. *IEEE DANMS*, Washington, D.C., USA.

[8] Dörr, E and Winiwarter, W., 2004. Decomposition and reuse of mobile services. *6th International Conference on Information Integration and Web-based Applications and Services*, OCG Press, Vienna, Austria.

[9] El-Kiki, T. and Lawrence, E., 2007. Mobile user satisfaction and usage analysis model of mGovernment services. *Second European Conference On Mobile Government*, Brighton, UK.

[10] Heskett, J. L., Jones T. O., Loveman G. W., Sasser, W. E., and Schlesinger, L. A., 1994. Putting the service-profit chain to work. *Harvard Business Review*. 164–174.

[11] IBM SSME Symposium, 2007. Succeeding through service innovation: a service perspective for education, research, business and government. *Cambridge Service Science, Management and Engineering Symposium*, Churchill College, Cambridge, Uk.

[12] Jimenez, A., Rıos-Insua, S., and Mateos A., (2002). A decision support system for multiattribute utility evaluation based on imprecise assignments. *Decision Support Systems* 36, 65–79.

[13] Kowalkowski, C. and Malmgren, M., 2008. Dynamics of value co-creation in buyer-supplier relationships. *Australian & New Zealand Marketing Academy Conference, ANZMAC*, Sydney, Australia.

[14] Krämer, B. J., 2008. Component meets service: what does the mongrel look like? *Innovations Syst Softw Eng*, 4, 4, 385-394.

[15] Krämer, B. J., Papazoglou, M. P., Schmidt, H. W., 1998. Information systems interoperability. *Advanced software development series (6)*. Research Studies Press, Taunton, Somerset, England. ISBN 0-86380-228-1.

[16] Lee, D.-J. and Ahn, J.-K. ,2007. Factors affecting companies' telecommunication service selection strategy. *Omega*, 35, 5, 486-493. doi:10.1016/j.omega.2005.09.004.

[17] Leung, H., and Fan, Z., 2002. Software cost estimation. *Handbook of Software Engineering*, Hong Kong Polytechnic University.

[18] Liu, D. and Deters, R., 2008. Management of service-oriented systems. *SOCA*, 2, 51–64. DOI 10.1007/s11761-008-0028-1.

[19] Menasce, D. A and Dubey, V., 2007. A heuristic approach to optimal service selection in service oriented architectures. *IEEE International Conference on Web Services (ICWS)*.

[20] Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F. Krämer, B.J., 2006. Service-oriented computing: a research roadmap. *Dagstuhl Seminar Proceedings*, 05462, Schloss Dagstuhl, Germany.

[21] Rohitratana, J. and Altmann, J., 2010. Agent-based simulations of the software market under different pricing schemes for software-as-a-service and perpetual software. *Workshop on the Economics of Grids, Clouds, Systems, and Services, GECON 2010*, Springer LNCS, Ischia, Italy.

[22] Rohloff, K., Ye, J., Loyall, J., and Schantz, R., 2006. A hierarchical control system for dynamic resource management. In: *2006 IEEE Real-Time and Embedded Technology and Applications Symposium*, San Jose, CA, USA.

[23] Spohrer, J., Anderson, L. C., Pass, N. J., Ager, T., and Gruhl, D., 2008. Service science. *Journal of Grid Computing*, 6, 3, 313–324 . DOI 10.1007/s10723-007-9096-2.

[24] Uysal, M., 2008. Estimation of the effort component of the software projects using simulated annealing algorithm. *World Academy of Science, Engineering and Technology*, 41.

[25] Vargo, S. L. & Lusch R. F. 2004. Evolving to a new dominant logic for marketing. *Journal of Marketing*, 68, 1–17.

[26] Yang, I.-T., (2008). Utility-based decision support system for schedule optimization. *Decision Support Systems*, 44, 595–605.

[27] Zahedi F., and Ashrafi, N., 1991. Software reliability allocation based on structure, utility, price, and cost. *IEEE Transactions on Software Engineering*, 17, 4.