Optimizing the Supplier Selection and Service Portfolio of a SOA Service Integrator

Jan Christian Lang*, Thomas Widjaja‡‡, Peter Buxmann‡‡, Wolfgang Domschke* and Thomas Hess§

*Chair of
Operations Research
Institute of Business Administration
Technische Universität Darmstadt
Hochschulstr. 1
64289 Darmstadt, Germany

‡‡Chair of
Information Systems
Institute of Business Administration
Technische Universität Darmstadt
Hochschulstr. 1
64289 Darmstadt, Germany

§Institute for
Information Systems and New Media
Munich School of Management
Ludwig-Maximilians-Universität München
Ludwigstr. 28 VG
80539 Munich, Germany

Abstract

The Service-Oriented Architecture (SOA) paradigm promises to enable software vendors to compose software systems using services purchased from various suppliers. In this paper, we analyze the impact of the SOA paradigm on the structure of the value chain in the software industry. We consider a new actor, the SOA Service Integrator (SeI), who integrates services from various suppliers in his portfolio in addition to own services in order to be able to fulfill heterogeneous functionality requirements of customers. His decision problem of selecting suppliers and deciding which services to integrate is modeled as a linear 0–1 programming model with a profit maximization objective. The model incorporates service make-or-buy decisions, and considers various cost types, namely the costs of establishing supplier relationships, integrating and using services. Based on this model and generated test problems, we analyze the optimal supplier selection and service portfolio under various cost scenarios.

Introduction

Software based on the Service-Oriented Architecture (SOA) paradigm promises a couple of advantages compared to traditional monolithic software systems. One of those advantages is that a SOA system based on open standards can be enriched by software services of specialized suppliers. The main technical basis of this development is the widespread use of an open communication standard (namely the web services technology), which significantly decreases the costs of integrating software services implemented by niche suppliers on the syntactical layer. Using the SOA approach, a company could integrate services directly purchased from various suppliers in its software landscape, or alternatively, an intermediary could provide the company with these services.

Our hypothesis is that the shift from mainly monolithic software architectures to the SOA paradigm will change the structure of the value chain in the software industry and in particular the degree of vertical integration. This implicates the emergence of new business models, possibly with new roles, in the software industry.

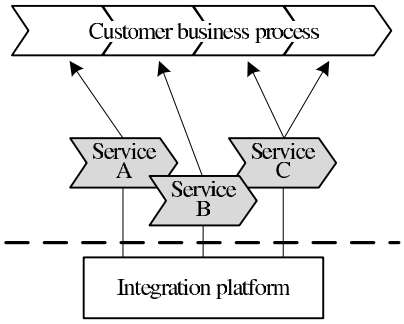

Figure 1: Steps of a customer's business process supported by various services interacting over a SOA integration platform

A promising new role could be a *SOA Service Integrator (SeI)*, i.e. a company that manages a portfolio of various specialized software services based on a certain platform and is thus able to offer each customer a tailored software solution.[1] A SeI could manage his service portfolio using a repository that contains metadata describing the services. The mentioned platform can be seen as the hub in the communication infrastructure of the services. To our understanding, the platform of an SeI can be characterized by the set of syntactical and semantical standards that it supports. It is composed of technical entities, such as the enterprise service bus (ESB), as well as more abstract entities, amongst others the used (implicit) definitions of business terms. Up to now, it is not clear whether established software vendors or entirely new players will take the SeI role.

In this article, we consider the SeI's perspective and analyze his specific decision problem. We focus our research on the business-to-business market for Enterprise Resource Planning

[1] We use SeI as an abbreviation for the term "SOA service integrator" to avoid confusion with the abbreviation SI which is commonly used for the term "system integrator".

1530-1605/08 $25.00 © 2008 IEEE

(ERP) software. An ERP software based on the SOA paradigm is typically composed of different services that communicate via a service bus (see figure 1). Throughout this paper, the *arrow* symbol denotes a process step whereas a *grayed arrow* stands for a software service. Services can either be developed by the ERP vendor himself or by specialized suppliers. A niche supplier could e.g. contribute a specialized demand forecasting service in the context of inventory management to an existing ERP system. Another example of a service could be a Manufacturing Execution System (MES) provided by a specialized supplier.

We develop an optimization model to provide a tool for determining the optimal number and composition of suppliers and services for a certain SeI in the ERP domain. This model is used to analyze the influence of various parameters on the characteristics of the optimal SeI strategy. We incorporate the reduction of relationship and integration costs caused by the SOA paradigm in our analysis. We also intend to show that the emergence of the SeI role is beneficial for the customers.

LITERATURE REVIEW

Our model is related to the research areas of information systems (especially SOA), supply chain management and operations research.

SOA services and software in general are digital goods. Previous research has identified special characteristics for this type of good (see e.g. [1]–[3]), of which several are relevant for our research. An important property is that software does not wear out by usage like typical physical goods, e.g. machines. Aside from this, a very specific cost structure is typical for the production process of software: The production (i.e. design, implementation and testing) of the first copy frequently incorporates high sunk costs and in contrast to that the creation of any further unit is nearly cost free. This results in strong economies of scale. Furthermore, software shows the characteristics of an experience good [4], i.e. a potential buyer cannot evaluate the quality of software before actual use.

There is a broad literature stream regarding the technical realization of the SOA paradigm. Although the concept of SOA is not bounded to a certain technology, the discussion is mainly focused on the web service technology (see e.g. [5]) and the related open standards (see e.g. [6], [7]).

The examination of the economic aspects of the SOA paradigm can either be conducted from the vendors' or the users' perspective. So far, a vast amount of research has been performed considering the users' perspective. Particularly the SOA advantages of the companies enhanced agility, reduced IT costs and the straightforward integration of heterogeneous IT environments are topics currently discussed (see e.g. [7]). Another focus of this field of research is the relationship between the SOA paradigm and Business Process Management (BPM) (see e.g. [7], [8]).

Until now, the perspective of the companies that produce SOA based software solutions has received little attention. A promising new role is the aforementioned SeI. In this context, three research areas in economics are relevant for our research: Intermediaries, network effect theory, and multi-sided markets. The emergence and desirability of the role of an intermediary, i.e. an actor that helps buyers and sellers to interact, has been discussed in various articles (see e.g. [9] and especially [10]). Network effects occur if the utility that a certain user derives from a good depends on the number of other users of the same good (see [11]–[14]). The literature on multi-sided markets is related to the network effect theory. Following [15], we use the term "multi-sided market" to refer to a market "in which the volume of transactions between end-users [i.e. user groups] depends on the structure and not only on the overall level of the fees charged by the platform [for enabling interactions between the user groups]" (see [16], [17] for alternative definitions). The research conducted concerning "platform leadership" (see [18], [19]) is closely related to this literature stream and also covers important aspects of the considered SeI strategy. Note that in this context, the term "platform" refers to a component of a technological system that is strongly functionally interdependent with the other parts of the overall system. Furthermore, the users demand only the overall system and have no or a very low willingness to pay for the separate components. The research concerning the optimal depth of the vertical integration of a platform owner has high relevance to our research question.

Besides the literature mentioned above, also the fields of Supply Chain Management (SCM) and Operations Research (OR) contain several research streams that are related to our work: Supplier selection and multiple sourcing, product substitution and flexible bills-of-material, mass customization, build-/assemble-to-order and product platform/family design.

A vast amount of research has been performed to develop qualitative and quantitative criteria and optimization models for selecting the supplier(s) from which to procure a product or service (see [20], [21]). Solution approaches contained in this literature could be modified for the selection of SOA service suppliers. Supplier selection models can be classified into single and multiple sourcing models (see [22]). Single sourcing means that we decide to purchase each product from exactly one supplier, whereas multiple sourcing means that we may decide to purchase equivalent or substitutable products from two or more suppliers and split the demand between these.

For a comprehensive recent review and classification of supplier selection models, see [23]. Most of the supplier selection models to be found in the literature are geared to procurement decisions for physical goods and thus incorporate inventory control and lot-sizing decisions, but there is also a (smaller) research stream on purchasing for services. Note that in this context, "service" refers to services in general, not specifically to software services. In the literature on service supplier selection, as well auction-based procurement strategies as mixed-integer-programming (MIP) models have been used to determine the optimal service supplier mix (see [24], [25]).

Product substitution and flexible Bills-Of-Material (BOMs) is another stream in production/logistics research, which is

relevant for our work for the following reason: Each software service may be interpreted as a product, and the set of all possible SOA configurations fulfilling the requirements of a certain customer could be specified using a flexible BOM. Substituting products is possible if the customers' demands are somewhat vague, so that certain substitutes may also fulfill their demands instead of the originally requested products. Flexible bills-of-material can capture such product substitution options in multi-level product structures, where e.g. a component of an assembly may be substituted by another component using an additional adapter. Material Requirements Planning (MRP) models with flexible BOMs are considered in [26], [27]. [28] belongs to the same research stream, but additionally incorporates material compatibility aspects.

Mass customization and build-/assemble-to-order approaches are becoming increasingly popular in manufacturing (consider e.g. Dell or BMW). [29] defines mass customization as "the ability to provide customized products or services through flexible processes in high volumes and at reasonably low costs". For reviews of the literature in this field, see [29]–[31]. In some sense, the tasks that SeIs perform can be interpreted as mass customization, as they use services from a portfolio to customize a SOA architecture for each customer. Hence, the available experience with mass customization in manufacturing could provide insights and ideas to the evolving SOA industry.

Product platform design (see [32], [33]) is a research avenue in engineering management/design that is connected with our work. In this research stream, the term "product platform" refers to "a set of common components, modules, or parts from which a stream of derivative products can be efficiently developed and launched" [34]. There is a link between product platform design and the role of a SeI: We can interpret the SOA platform of a SeI including the services of suppliers integrated into it as a product platform. There are some studies that employ optimization models for product platform design (see [35], [36]).

There are a few papers related to SOA that develop optimization models for the operational dynamic composition of web services and solve them using metaheuristics (see [37]–[40]). They model the user's perspective, i.e. the decision maker in the optimization models is an entity actually using the services, not a service vendor. A key assumption underlying these papers is that the SOA is composed of a set of abstract services, each of which can be implemented by various concrete services with differing Quality-of-Service (QoS) attributes.

Optimization model

In this section, we first describe the assumptions of our model. Based on these, we introduce a mathematical notation and model the problem as a linear 0–1 programming model.

Assumptions

Our model focuses on the decisions of a certain SeI. Hence, the decisions of customers and service suppliers are modeled as exogenous. This implicates that aspects concerning the theories of network effects and multi-sided markets are not covered by the optimization model. After outlining some general aspects of the model, we describe the assumptions pertaining to the different roles, i.e. to the customers, the service suppliers and the SeI. We assume the following (business-to-business) market structure (see figure 2): Several potential customers (companies) demand ERP systems that map to their business processes. The SeI assembles tailored customer solutions, each composed of various services contained in his service portfolio. The service portfolio consists of services which are either purchased from (niche) suppliers or implemented by the SeI himself (the latter are symbolized by dark gray arrows). We assume that the SeI's portfolio already contains several existing services in the initial situation.

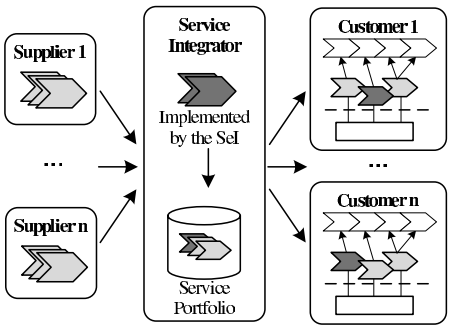

Figure 2: Supply network from the SeI's perspective

In our model, a SOA-based ERP system is composed of a set of services and a platform that ensures the communication between the services. We assume that there is a finite set of functionalities in the considered domain of ERP systems. An ERP software solution provides a subset of this overall set of functionalities. Every service is produced by a distinct service supplier and implements a subset of the overall set of functionalities. Thus, each service can be seen as a set of functionalities. A service could e.g. be a demand forecasting service in the context of inventory management that offers functionality such as forecasting using statistical methods, e.g. univariate regression, exponential smoothing, and multivariate regression.

Different services do not need to have disjunct functionality sets, and a specific functionality could be supported by various services. A solution is able to provide a certain functionality if one or more services contained in it implement the functionality. Furthermore, it is possible that the granularity of the services varies, i.e. some services may cover only small sets of functionalities, whereas others implement large sets of functionalities: With regard to the example, one demand forecasting service could offer only linear regression, another one linear regression, exponential smoothing and multivariate regression.

Every *customer* demands an individual set of functionalities that are necessarily required, which we refer to as "must functionalities". If the SeI wants to sell a solution (i.e. platform and services) to a customer, the customer's must functionalities

have to be covered. In our model, we assume that the price for a solution implementing all demanded must functionalities and the recurring payments for maintenance have already been negotiated in advance between SeI and customer. In addition to that, the customer and the SeI have negotiated prices for certain "nice-to-have functionalities" in the same manner. We assume that the negotiated prices are below the willingness-to-pay of the respective customer, thus the customer will buy the solution if it is offered by the SeI at the agreed price. An implicit assumption of our model is that there are no direct payment flows from customers to service suppliers, i.e. the customers buy solutions from the SeI, who in turn pays license fees to service suppliers for the use of services in the customer solutions. Our model neither considers technical dependencies nor bundle price aspects regarding the nice-to-have functionalities. In addition, each customer has a certain "size" that influences the license fees to be paid to the suppliers.

The SeI has access to a finite set of potential *service suppliers*. Each of these service suppliers offers a set of various services which implement certain functionalities. Referring to the demand forecasting service example, there could be a few niche suppliers providing demand forecasting services using other, more sophisticated approaches such as neural networks, in addition to other suppliers who offer demand forecasting services that use standard approaches.

We model the decisions of the *service integrator* regarding the following question: What is the optimal number and composition of suppliers and services to maximize the profit? The underlying problem for the SeI is to compose a service portfolio in a way that solves the "functionality set covering problem" for each customer in order to maximize overall profit. This term refers to the task of selecting a number of services that cover functionalities demanded by customers. The SeI can increase his revenue by selling solutions to different customers. But to accomplish this, the SeI has to incorporate services into his portfolio (either by buying or implementing them) which incurs costs.

We distinguish five types of SeI costs for incorporating and using services of suppliers: 1) Costs of establishing a relationship to the supplier, 2) costs of integrating a certain service in the portfolio, 3) license fees that the integrator has to pay to the supplier if the service is actually used in a customer solution, 4) costs of integrating the platform and must functionalities in the existing (legacy) software landscape of the customers, and 5) costs of integrating nice-to-have functionalities in the customers' software landscapes. We assume that all five types of costs cannot be influenced by the SeI during the considered decision horizon. The costs types and the corresponding economic assumptions are described in more detail below.

Due to the experience good characteristics of software – the quality of a software service can only be examined by testing/using it – we assume that there is a significant need for a long-lasting relationship between the SeI and the service supplier. Establishing such relationships incurs costs. Those relationship costs (cost type 1) include transaction costs, e.g. search costs to find an adequate supplier and overhead cost for contract negotiation, the costs of enabling the supplier to integrate in the portfolio of the SeI (training, conferences, etc.), and the costs of providing development tools to the supplier. Note that these costs could be negative if the SeI charges the suppliers for trainings, partnership certifications etc. The costs are assumed to be fixed, regardless of the number of services that the SeI selects from this supplier. In practice, these costs are influenceable by the SeI, but to limit the complexity of our model, we assume that the SeI has already decided on the the level of these costs.

Our model also deals with the costs of including a certain service in the portfolio of the SeI (cost type 2). Those are the costs for semantical and syntactical integration of the selected service with other services in the service portfolio. The semantical compatibility is often more challenging for the SeI than the compatibility on the syntactical level, as it is widely agreed to use the web service standard on the technical layer. Particularly in the domain of ERP systems, vendors use differing definitions and taxonomies for the same business terms. Thus, it is possible that two services are syntactically compatible but incompatible on a semantical level. We assume that the integration costs of all services produced by a supplier are low if a supplier is already closely aligned to the semantical and syntactical specifications of the SeI, and correspondingly high if a supplier uses a semantic and syntax differing strongly. Note that the semantical and syntactical "alignment" of a customer partly depends on the SeI's investments (e.g. individual training) in the relationship. As we assumed earlier that the SeI has already decided on the level of these investments, this dependency is not considered in our model. In addition, we assume that due to a hub-and-spoke communication architecture of the SeI's platform, the costs to integrate a certain service into the service portfolio of the SeI are independent of the number of services that are already in that service portfolio. Almost all platforms of the vendors for current ERP systems (e.g. SAP Netweaver, Oracle Fusion and IBM Websphere) use this type of communication architecture.

The costs of using a service in a customer solution (cost type 3) are mainly the license costs that the SeI pays for the service. We assume that those costs vary with the size of the customer. Because of the hub-and-spoke communication architecture, those costs do not depend on other services that are part of the solution for this customer. Another cost type that we consider are the costs of integrating solutions into the existing IT environments of customers (cost type 4), e.g. the costs of integrating the SeI's SOA platform into a customer's legacy software landscape. Integrating nice-to-have functionalities similarly incurs costs (cost type 5).

Instead of buying the service from a service supplier, there is a second possibility for the SeI to incorporate a certain service into his portfolio: The SeI could implement the service himself. In this case, the cost structure changes dramatically: There are no fixed costs to establish a connection to a supplier

and no costs for semantical and syntactical integration, as we assume that the compatibility is already "built-in" in this case. We also assume that the costs of using a self-implemented service in a customer solution are very low because of reproduction costs of digital goods close to zero. Yet, the SeI has to bear high fixed implementation costs when implementing a service himself. Note that services obtained by acquisitions of supplier companies can be interpreted as self-implementations of services. Thus, acquisition options could be mapped in the model.

Model

We formulate the *SOA Service Integrator Portfolio Problem* (SOA-SIPP) as a linear 0–1 programming model that can be solved to optimality using a standard mixed-integer programming (MIP) solver, e.g. CPLEX or XPRESS [41]. The SOA-SIPP is formulated as a static, deterministic model using the notation given in table 1. All cost parameters and decision variables only refer to the SeI's perspective. If a binary decision variable takes the value 1, this means "yes" regarding the respective decision. We let index i denote a service, j a functionality, v a vendor and k a customer. To incorporate make-or-buy decisions of the SeI in the model, we add a virtual vendor named SeI that stands for "self-implementation" of a service. Each customer requires a set of must functionalities F_k^m and would also buy several nice-to-have functionalities that form the set F_k^n. Note that these sets are disjunct for each customer k, but a must functionality of customer k_1 could be a nice-to-have functionality of another customer k_2.

The profit maximization objective (1) is composed of the following components: (a) The revenue generated by selling solutions to customers fulfilling their must functionalities, (b) the additional revenue generated by fulfilling nice-to-have functionalities, minus the (c) the costs of integrating the must functionalities and (d) the cost of integrating the nice-to-have functionalities in the customers' IT environments, (e) the fixed vendor relationship costs, (f) the costs incurred by integrating services in the portfolio, and (g) the (license) costs of services actually required for customer solutions. One may interpret the revenue and cost parameters of the mathematical model in such a way that they represent discounted cash flows, i.e. include both initial and recurring payments.

We formulate the SOA-SIPP model as follows:

$$\text{Max } P(x,y,u,w,z) = \sum_{k\in C}\left(\left(p_k^m - c_k^m\right)u_k + \sum_{j\in F_k^n}\left(p_{jk}^n - c_{jk}^n\right)z_{jk}\right) - \sum_{v\in V} f_v^r x_v - \sum_{i\in S}\left(c_i^s y_i + \sum_{k\in C} c_{ik}^c w_{ik}\right) \quad (1)$$

subject to

$$y_i \le x_{v(i)} \qquad \forall i \in S,\ v(i) \ne SeI \quad (2)$$

$$w_{ik} \le y_i \qquad \forall i \in S,\ k \in C \quad (3)$$

$$u_k \le \sum_{i\in S_j} w_{ik} \qquad \forall k \in C,\ j \in F_k^m \quad (4)$$

$$z_{jk} \le \sum_{i\in S_j} w_{ik} \qquad \forall k \in C,\ j \in F_k^n \quad (5)$$

$$z_{jk} \le u_k \qquad \forall k \in C,\ j \in F_k^n \quad (6)$$

$$x_v \in \{0,1\} \qquad \forall v \in V \quad (7)$$

$$y_i \in \{0,1\} \qquad \forall i \in S \quad (8)$$

$$u_k \in \{0,1\} \qquad \forall k \in C \quad (9)$$

$$w_{ik} \in \{0,1\} \qquad \forall k \in C,\ i \in S \quad (10)$$

$$z_{jk} \in \{0,1\} \qquad \forall k \in C,\ j \in F_k^n \quad (11)$$

Table 1: Notation for SOA-SIPP optimization model

Symbol	Definition
Indices, sets and auxiliary functions	
$i \in S$	Services
$j \in F$	Functionalities
SeI	Service integrator (virtual SeI supplier)
$v \in V$	Suppliers, not including virtual SeI supplier
$k \in C$	Customers
$F_k^m \subseteq F$	"Must" functionalities required by customer k
$F_k^n \subseteq F \backslash F_k^m$	"Nice-to-have" functionalities desired by customer k
$F_i \subseteq F$	Functionalities supported by service i
$S_j \subseteq S$	Services supporting functionality j
$v(i) \in V \cup \{SeI\}$	Supplier of service i
Cost/revenue coefficients	
p_k^m	Selling price for the demanded must functionalities of customer k
c_k^m	Cost of integrating a solution (including the platform) implementing the must functionalities in the software landscape of a customer k (cost type 4)
p_{jk}^n	Selling price for nice-to-have functionality j desired by of customer k
c_{jk}^n	Cost of integrating nice-to-have functionality j in the software landscape of a customer k (cost type 5)
f_v^r	Fixed cost for establishing a relationship with supplier v (cost type 1)
c_i^s	Cost of integrating service i in the portfolio (cost type 2)
c_{ik}^c	Cost of using service i in the solution for customer k (cost type 3)
(Binary) decision variables	
x_v	Indicates whether a relationship with supplier v is established
y_i	Indicates whether the SeI decides to integrate service i into his portfolio
u_k	Indicates whether the SeI decides to sell a solution to customer k
w_{ik}	Indicates whether service i is used in the solution for customer k
z_{jk}	Indicates whether nice-to-have functionality j is provided in the solution for customer k

Constraint set (2) ensures that fixed relationship costs are incurred for a specific vendor v if the SeI integrates a service

i provided by vendor v into his portfolio. Note that there is no binary variable x_{SeI} for incorporating fixed relationship costs of the "virtual" vendor SeI because no fixed relationship costs are incurred if the SeI decides to implement services himself, i.e. $f^r_{SeI} = 0$. Thus, services supplied by the virtual vendor SeI are excluded from (2). Constraint set (3) means that if a specific service i is used in one or more customer solutions, the costs of integrating it into the SeI's portfolio are taken into account. Constraint set (4) enforces that the SeI can only sell a solution to a customer k if he implements at least the must functionalities of customer k with the services contained in the assembled solution: It makes sure that at least one service i supporting must functionality j (i.e. a service $i \in S_j$) is contained in the solution for customer k if the SeI sells it. Finally, constraint set (5) denotes that the SeI only obtains additional revenue by selling a nice-to-have functionality j to a customer k if he actually implements it with a service i in the customer solution: Similarly to (4), it enforces that at least one service i supporting nice-to-have functionality j is contained in the solution for customer k if the SeI charges the customer for the functionality. The binary domains of the variables are specified in (7)–(11). Note that our model formulation (1)–(11) bears a resemblance to set covering problems [42] and fixed-charge network flow problems [43].

EXPERIMENTS

We used a generic, self-developed Java software framework for implementing, conducting and analyzing our experiments.

Setup – Generation of test instances

In our experiments, we generated, solved and analyzed various SOA-SIPP instances. Note that, as we could not obtain realistic data on the cost/revenue parameters and the structure of suppliers and services, we had to create problem instances artificially. Based on the analysis of a certain base case, we studied the effects of varying certain parameters (ceteris paribus). The characteristics of this base case are given in table 2. Here, $\hat{N}(\mu, \sigma)$ refers to a modified normal distribution with mean μ and standard deviation σ for which all values < 0 are truncated, i.e. the probability density function is 0 for all values < 0. $U(a, b)$ refers to a uniform distribution with minimum value a and maximum value b. For distributions specified with a normalized standard deviation (σ_n), the standard deviation σ is calculated as follows: $\sigma = \sigma_n \cdot \mu$. For supplier services, the distribution of the service integration costs c^s_i is different for each vendor v, with normal distribution parameters μ and σ drawn from specified probability distributions. Similarly, the variable service usage costs c^c_{ik} have a different distribution for each customer k. Additional symbols α_k, β_j, γ_i, δ, ϵ_k, ζ, η, and κ are used to specify the instance generation process (see table 2).

We distinguish three types of services: (a) Services that are already in the portfolio of the SeI in the initial situation ("existing SeI services"), (b) services of suppliers that the SeI could purchase ("buy services"), and (c) "copies" of the suppliers' services with identical functionality that the SeI could implement himself, but which do not exist yet ("make services"). The total number of services $|S|$ includes all three types of services. Services have a certain granularity, i.e. number of supported functionalities, which is denoted by γ_i. These granularities are drawn from a uniform distribution. For each of the services of type (a) and (b), exactly γ_i supported functionalities are sampled from the overall set of functionalities. The number of services of type (a) (existing SeI services) is determined by the probability δ that a service is already in the portfolio of the SeI in the initial situation. If a service i is already in the portfolio of the SeI, the cost parameters c^c_{ik} are comparatively low. The probability $1 - \delta$ determines the number of services of type (b) ("buy services"), i.e. services offered by suppliers. We assume that each supplier has roughly the same number of services. Hence, we randomly assign "buy services" to suppliers. In addition, we add a service i_c of type (c) (make service) to the instance

Table 2: Notation and base case for the following experiments

Parameter	Value / distribution
Problem size	
$\|S\|$	200 Services
$\|F\|$	200 Functionalities
$\|V\|$	25 Suppliers
$\|C\|$	40 Customers
Generator parameters	
$\alpha_k \geq 1$	"Size" parameter of customer k; $\sim U(5.0, 10.0)$
$\beta_j > 0$	Probability that a functionality j is implemented by a certain service, obtained by normalizing $\sim \hat{N}(100.0, 30.0)$ so that $\sum_{j \in F} \beta_j = 1$
$\gamma_i \geq 1$	Granularity of service i; $\sim \|F\| \cdot U(0.03, 0.08)$
δ	Probability that a service is already in the portfolio of the SeI (make services excluded from calculation); $= 20\%$
ϵ_k	Percentage of functionalities demanded by a customer k; $\sim U(20\%, 40\%)$
ζ	Probability that a functionality j from a different functionality cluster is added to the set F_i of supported functionalities of a service i $= 20\%$
η	Percentage of must functionalities in the requirements of a customer; $= 60\%$
κ	Number of functionality clusters; $= 6$
Model parameter distributions	
f^r_v	$\sim \hat{N}(400000, 20000)$
c^s_i	0 — existing SeI services $\sim \hat{N}$ for each v with $\mu \sim \hat{N}(200000, 10000)$ normalized $\sigma_n \sim U(0.05, 0.2)$ — supplier services $\sim \hat{N}$ of supplier services $\cdot$ $U(2.0, 10.0)$ — make services
c^m_k	$\sim \alpha_k \cdot \hat{N}(3900000, 100000)$
c^n_{jk}	0
c^c_{ik}	$\sim \hat{N}$ for each k with $\mu \sim \alpha_k \cdot \hat{N}(40000, 8000)$ normalized $\sigma_n \sim U(0.05, 0.2)$ — supplier services $\sim 0.1 \cdot \hat{N}$ of supplier services — SeI services
p^m_k	$\sim c^m_k \cdot U(1.6, 2.4)$
p^n_{jk}	$\sim \alpha_k \cdot \hat{N}(10000, 3000)$

for every service i_b of type (b) that is offered by a supplier. This service i_c has identical functionalities, but the parameter $c^s_{i_c}$ is set significantly higher than $c^s_{i_b}$. The supplier of this service is the SeI himself, i.e. $v(i_c) = SeI$. In this way, we model the "make" option for the SeI of implementing a certain service himself instead of "buying" it from a supplier. The cost parameters $c^c_{i_c k}$ of each make service i_c have the same distribution as those of SeI services that were already in the SeI's portfolio in the initial situation, i.e. they are rather low. Therefore it is beneficial for the SeI to *make* services that are frequently used in customer solutions instead of buying them.

β_j denotes the probability that a certain functionality j is implemented by a supplier. We use this parameter to model that the implementation of certain functionalities requires special know-how which is only accessible to a few suppliers. Using the probabilities β_j, we draw γ_i functionalities from the set of functionalities F for each service i. We assume that β_j also describes the probability that a certain functionality j is demanded by a customer.

To ensure that the functionality set of every service is cohesive, i.e. not spread too much across various functionality areas, the functionalities are segmented into κ "functionality clusters". Each functionality belongs to exactly one functionality cluster. The assignment of functionalities to clusters is performed randomly with a uniform distribution. Each service is associated with a random functionality cluster, i.e. most of its functionalities lie in this "primary" cluster, but a few may lie in other clusters. The sampling of the services' functionality sets (F_i) is performed using random sampling without replacement with the probabilities β_j. A functionality – drawn for a service i – that does not belong to the primary cluster of i is only accepted with probability ζ. The sampling is continued until the service has γ_i functionalities, i.e. until the desired granularity/size is reached.

Every customer k demands ϵ_k percent of the overall functionalities. η of those are must functionalities, the remaining $1-\eta$ are nice-to-have functionalities. The parameter α_k is used to denote the "size" of a certain customer k. Cost and revenue parameters of a customer depend on this size parameter: The costs c^m_k of integrating a solution in the IT environment of a big customer are higher than those of integrating it in the software landscape of a small company. We furthermore assume that the integration costs c^n_{jk} of nice-to-have functionalities are zero due to the hub-and-spoke communication architecture.

Also, the costs c^c_{ik} of using a service provided by a supplier depend on α_k. We assume this because many software suppliers price licenses according to parameters related to the size of the customer, e.g. the number of workstations or users. We also assume that the selling prices p^m_k respectively p^n_{jk} of the must and nice-to-have features depend on α_k. The dependence between α_k and p^m_k is indirect over c^m_k.

Research questions and experiment design

In this section, we first briefly describe the research questions that we address with our experiments. After this, we explain the design of the experiments used to examine these questions:

1) How does the optimal configuration of the supplier selection and service/product portfolio differ between actors selling SOA-based vs. traditional software systems?

2) How does the integration of supplier services – facilitated by the SOA paradigm– influence the SeI's ability to fulfill heterogeneous customer requirements?

3) What is the impact of the facilitated integration of supplier services – due the change of the cost structure by the introduction of the SOA paradigm – on the profit of a SeI?

The experiment design was chosen as follows: Instances were generated by varying the distributions of the relationship costs f^r_v and the service integration costs c^s_i while keeping all other instance data distributed as in the base case given in table 2. We assume that relationship and service integration costs are lower for SOA-based systems than for traditional software systems due to technical standardization (e.g. web service technology), encapsulation, and loose coupling of services. We considered scenarios with high relationship and high service integration costs, which we assume to be typical for traditional software systems, as well as scenarios with low relationship and low service integration costs, as promised by the SOA paradigm. We analyzed various scenarios between these two extreme scenarios by simultaneously decreasing f^r_v and c^s_i. The mean of f^r_v was varied between 1,720,000 and 160,000 currency units in 40 steps of 40,000 each, while the mean μ of the mean distribution of c^s_i was varied between 1,190,000 and 20,000 in 40 steps of 30,000. The standard deviation of f^r_v was varied from 86,000 and 8,000 in steps of 2,000. The normalized standard deviation σ_n of c^s_i had the same distribution as in the base case. The variation of the two cost types' distributions resulted in 40 scenarios. In order to ensure representative results, we sampled 10 problem instances for each of the cost scenarios, whereby we obtained 400 problem instances in total. The metrics described in table 3 were used to characterize and compare the solutions of these instances. In the following section, we always report *mean* metric values that are averaged over the 10 instances sampled for each cost scenario. A tilde was added to all symbols for metrics.

Computational results and interpretation

The results regarding our first research question – the impact of the SOA paradigm on the optimal configuration of the supplier selection and product/service portfolio – are summarized in figures 3 and 4.[2] Figure 3 clearly indicates that the percentage of established supplier relationships $\tilde{r}_v$ and the size of the portfolio $\tilde{r}_s$ in the optimal solution increase if the parameters f^r_v and c^s_i decrease. Figure 4 shows that the ratio of purchased services in the portfolio of the SeI rises if the costs of establishing supplier relationships and integrating

[2]Note that the cost distribution parameters on the x-axes of figures 3, 4 and 5 decrease from left to right. The σ parameters can be calculated from the respective μ values of the f^r_v and c^s_i distributions (see description of experiment design).

Table 3: SOA-SIPP solution metrics

Symbol	Metric
$\tilde{P}$	Total profit of the SeI
$\tilde{r}_v$	Percentage of possible supplier relationships established: # established supplier relationships / # potential suppliers
$\tilde{r}_s$	Percentage of services contained in SeI service portfolio: # services in SeI portfolio / total # of services
$\tilde{r}_{make}$	Percentage of services implemented by SeI (*make* decisions) in the portfolio: # services in portfolio implemented by SeI / # services in SeI portfolio
$\tilde{r}_{buy}$	Percentage of purchased supplier services (*buy* decisions) in portfolio: # services in portfolio purchased from suppliers / # services in SeI portfolio
$\tilde{r}_{existing}$	Percentage of services owned by the SeI in the initial situation: # services in portfolio owned by the SeI / # services in SeI portfolio
$\tilde{r}_c$	Percentage of customers served by SeI: # customers to which a solution (with all must functionalities and possibly several nice-to-have functionalities) is sold / total # of potential customers
$\tilde{r}_{nf}$	Percentage of nice-to-have functionalities fulfilled: # of nice-to-have functionalities fulfilled / # of nice-to-have functionalities of those customers that are served (each functionality is only counted once, also if it is nice-to-have for >1 customers)

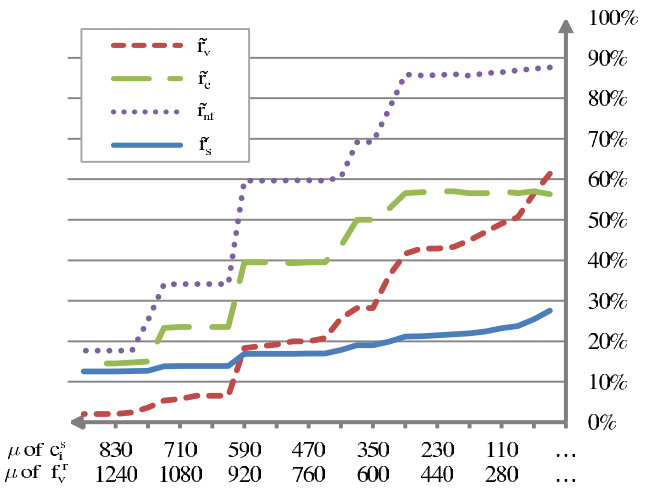

Figure 3: Supplier relationships ($\tilde{r}_v$), portion of customers served ($\tilde{r}_c$), service usage ($\tilde{r}_s$) and fulfillment of nice-to-have functionalities ($\tilde{r}_{nf}$) – Average values for various f_v^r and c_i^s cost scenarios

services decrease. If those costs are extremely high, the SeI does not integrate any services of suppliers in his portfolio. The proportion of make services ($\tilde{r}_{make}$ is low if f_v^r and c_i^s are high, as the SeI does not serve many customers and hence has no need to implement services. As the costs decrease, $\tilde{r}_{make}$ first increases because it is now economic to serve more customers and – due to the relatively high relationship and integration costs – provide them with self-implemented services. From a certain point, $\tilde{r}_{make}$ drops again because the buy option gets more attractive than the make option.

Figure 3, which also refers to our second research question, indicates a positive effect of a decrease of f_v^r and c_i^s on the percentage $\tilde{r}_c$ of customers served. The percentage of customers that could be covered by the functionalities contained

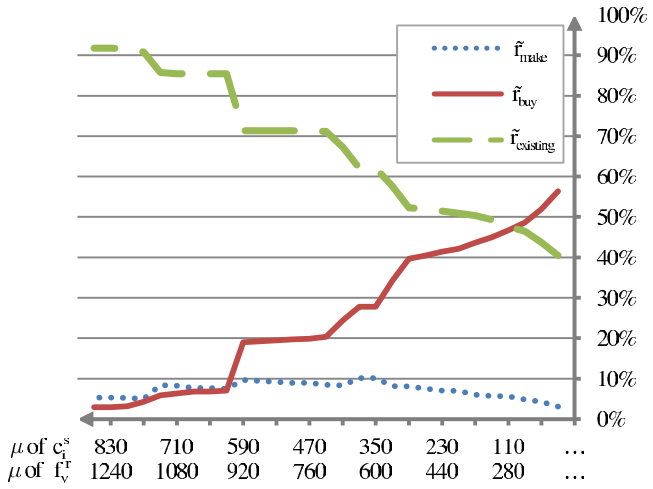

Figure 4: Sources of services in the portfolio for various f_v^r and c_i^s cost scenarios – Average values of the percentages of services in the portfolio already owned by the SeI ($\tilde{r}_{existing}$), purchased from suppliers ($\tilde{r}_{buy}$), and implemented by the SeI ($\tilde{r}_{make}$)

in the overall set of services has a mean of only 60.75%, as some customers have must functionality requirements that none of the SeI's and the suppliers' services cover. Therefore, $\tilde{r}_c$ converges to that mean for low values of f_v^r and c_i^s. In addition, we observe that the proportion of functionalities covered by the SeI increases constantly: First, the number $\tilde{r}_c$ of customers served raises, which implicates that more must functionalities are covered. Second, the percentage $\tilde{r}_{nf}$ of nice-to-have functionalities fulfilled increases steadily. This metric does not reach the 100% level because certain nice-to-have functionalities may be demanded by only a single customer and at the same time only be included in a service that is offered by a supplier that offers no other "useful" functionalities. In such situations, establishing a relationship to the supplier to be able to provide the nice-to-have functionality would not be economic due to the high associated fixed relationship cost.

Regarding our third research question, figure 5 shows that the profit $\tilde{P}$ of the SeI increases steadily if the cost parameters f_v^r and c_i^s decrease. This effect can be explained by three causes: (a) The SeI sustains more relationships to suppliers and includes more services and thus more functionalities in his portfolio. This enables the SeI to serve more customers, due his ability to fulfill demand for must functionalities (see figure 3). (b) The same argument holds true for the number of nice-to-have functionalities that the SeI can provide to existing and new customers (see figure 3). (c) Moreover, the linear decrease of the cost parameters trivially has a positive effect on the SeI's profit. We assume that causes (a) and (b) are the main reasons for the shape of the curve in figure 5. In a situation with relatively high cost parameters, the number of customers served (and also the number of nice-to-have functionalities fulfilled) is still limited because fulfilling the customers' must functionalities would be too expensive. The values of $\tilde{r}_c$ and $\tilde{r}_{nf}$ increase gradually (see figure 3). After the cost parameters get sufficiently low so that all possible

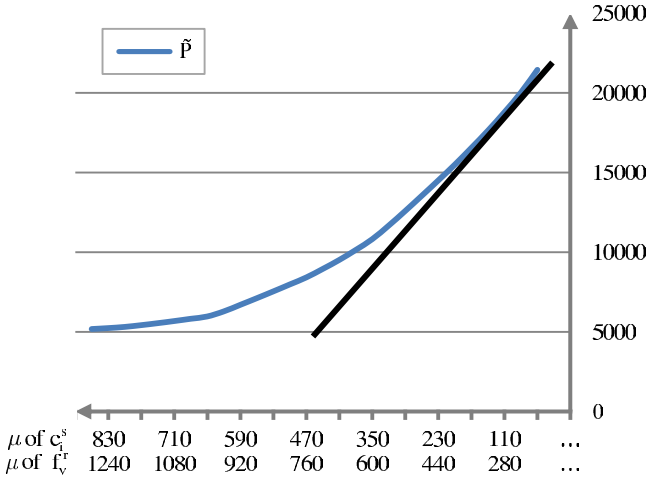

Figure 5: Average overall profit of the SeI ($\tilde{P}$) for various f_v^r and c_i^s cost scenarios. All data is in 1,000 currency units.

customers are served and their nice-to-have functionalities fulfilled, the profit increases linearly with the decrease of the cost parameters. Referring to figure 3, the slope of the profit curve converges to a finite value, which is symbolized by a straight line with the respective slope in the figure.

395 of the 400 generated test instances, i.e. 98.75%, were solved to optimality by CPLEX within a time limit of 5 minutes, using a relative MIP optimality gap tolerance of $10^{-3} = 0.1\%$. For the 5 instances that could not be solved within the time limit, the average optimality gap was 2.2%.

Beside the experiments regarding our main research questions, we also varied further parameters. Due to space restrictions, we just briefly sum up one selected additional result: In a further experiment with 1200 instances, we varied the proportion of services already in the portfolio of the SeI (δ) between 0% and 50%. One general finding that holds for various distributions of the parameters f_v^r and c_i^s is that the percentage of possible supplier relationships established ($\tilde{r}_v$) is almost zero for both very low and very high values of δ, whereas values of δ between 20% – 30% result in relatively high $\tilde{r}_v$ values. Trivially, the profit of the SeI increases parallel to the raise of δ. We presume that low values of δ result in a bad "start position" for the SeI, i.e. additional supplier relationships are not beneficial. Yet, if the value of δ exceeds a certain level, the SeI can cover almost all demanded functionalities with his existing services and needs no additional supplier relationships. Hence, it seems more advantageous to take the SeI role for actors with a certain base of functionalities, e.g. current ERP vendors, than for actors without an existing portfolio.

The conclusions drawn from our experiments should only be generalized with care because we analyzed artificially created instances of the SOA-SIPP. Realistic data on the cost and revenue parameters and the structure of suppliers and services were not available. However, we tried to choose model and parameter assumptions as realistic as possible and performed sensitivity analyses of several parameters in order to ensure plausibility and stability of our computational results and thus draw proper conclusions. In addition, the sales strategy assumed in this paper with the SeI acting as the sales channel for all suppliers' services might not be feasible in practice because the SeI's sales department would need a huge knowledge base to be able to sell all services competently.

Conclusions and Future Research

In this paper, we examined the impact of the SOA paradigm on the software industry's value chain structure. We considered the vendors' perspective and analyzed the influence of decreased relationship and service integration costs on the optimal service portfolio composition and supplier selection of a SeI. We developed a linear 0–1 programming model addressing this optimization problem and examined the impact of several parameter variations (ceteris paribus) on the structure of the optimal solutions. Our main findings are the following effects of the decreased relationship and integration costs: The SeI's profit increases, and due to the breadth of his service portfolio achieved by integrating more services from various suppliers, he is able to deliver solutions to a larger number of customers and also to provide more nice-to-have functionalities. Thus, the SeI role gets more economically attractive.

With regard to the SOA-SIPP optimization model, different solution techniques may be required for large instances sizes, as the computational efforts required by our approach might get unacceptably high. Possible approaches could (a) be reformulations that provide tighter upper bounds for pruning in the B&B algorithm, (b) metaheuristics, or (c) specialized heuristics. The self-developed generic software framework used in our experiments turned to facilitate and speed up the implementation and analysis of our experiments.

We see various opportunities to get further interesting insights by extending the SOA-SIPP optimization model – e.g. to a multi-period or stochastic model – or embedding it into a multi-agent simulation. Especially by including the decisions of the other actors, i.e. customers and suppliers, various new research questions related to network effects and multi-sided markets theory can be analyzed. A widely spread example of a two-sided market is the market for video game platforms (e.g. Nintendo Wii, Sony Playstation and Microsoft X-Box). On the one hand, the platform owners have to attract game developers, i.e. the first user group, to provide games for the specific platform to convince the gamers, i.e. the second user group, to buy the console. On the other hand, the game developers prefer to design games for consoles with a large community of gamers. A closely related phenomenon is observable regarding SeIs: The SeI's platform enables the two user groups (service suppliers and customers) to interact, and the utility structure of the two groups shows dependencies. A question that received much attention in multi-sided markets theory is which price a platform owner should charge the different customer groups in order to maximize profit.

Another interesting problem that could be examined by future research models is the way in which the profits are distributed between the different actors [44]. Here, we see

connections to the research on supply chain contracting [45], [46]. In addition, a broad empirical study could be useful to obtain realistic data on the parameters of our model and validate its assumptions.

ACKNOWLEDGMENT

We would like to thank Sonja Lehmann, Dr. Georg Rau, and the two anonymous reviewers for their many useful suggestions. Thomas Widjaja's research was supported with a research grant from the FAZIT-Stiftung.

REFERENCES

[1] S.-Y. Choi, D. O. Stahl, and A. B. Whinston, *The economics of electronic commerce*. Indianapolis, Ind.: Macmillan Technical Publ., 1997.

[2] C. Shapiro and H. R. Varian, *Information rules: A strategic guide to network economy*. Boston, Massachusetts: Harvard Business School Press, 1998.

[3] M. L. Katz and C. Shapiro, "Antitrust in software markets," in *Competition, Innovation and the Microsoft Monopoly: Antitrust in the Digital Marketplace*, J. Eisenach and T. Lenard, Eds. Boston: Kluwer Academic Publishers, 1999.

[4] P. Nelson, "Information and consumer behavior," *The Journal of Political Economy*, vol. 78, no. 2, pp. 311–329, 1970.

[5] G. Alonso, *Web Services: Concepts, architectures and applications*. Berlin: Springer, 2004.

[6] T. Erl, *Service-oriented architecture: Concepts, technology, and design*, 5th ed. Upper Saddle River, NJ: Prentice Hall, 2006.

[7] D. Krafzig, K. Banke, and D. Slama, *Enterprise SOA: Service-oriented architecture best practices*, 6th ed. Upper Saddle River, New Jersey: Prentice Hall, 2006.

[8] F. Leymann and D. Roller, *Production workflow: Concepts and techniques*. Upper Saddle River, NJ: Prentice Hall, 2000.

[9] D. F. Spulber, "Market microstructure and intermediation," *The Journal of Economic Perspectives*, vol. 10, no. 3, pp. 135–152, 1996.

[10] J. P. Bailey and Y. Bakos, "An exploratory study of the emerging role of electronic intermediaries," *International Journal of Electronic Commerce*, vol. 1, no. 3, pp. 7–20, 1997.

[11] J. Farrell and G. Saloner, "Standardization, compatibility, and innovation," *The RAND Journal of Economics*, vol. 16, no. 1, pp. 70–83, 1985.

[12] M. L. Katz and C. Shapiro, "Network externalities, competition, and compatibility," *The American Economic Review*, vol. 75, no. 3, pp. 424–440, 1985.

[13] J. Farrell and G. Saloner, "Installed base and compatibility: Innovation, product preannouncements, and predation," *The American Economic Review*, vol. 76, no. 5, pp. 940–955, 1986.

[14] M. L. Katz and C. Shapiro, "Technology adoption in the presence of network externalities," *The Journal of Political Economy*, vol. 94, no. 4, pp. 822–841, 1986.

[15] J. C. Rochet and J. Tirole, "Two-sided markets: A progress report," *The RAND Journal of Economics*, vol. 37, no. 3, 2006.

[16] ——, "Platform competition in two-sided markets," *Journal of the European Economic Association*, vol. 1, pp. 990–1029, 2003.

[17] D. S. Evans, A. Hagiu, and R. Schmalensee, "A survey of the economic role of software platforms in computer-based industries," *CESifo Economic Studies*, vol. 51, no. 2-3, pp. 189–224, 2005.

[18] A. Gawer and M. A. Cusumano, *Platform leadership: How Intel, Microsoft, and Cisco drive industry innovation*. Boston: Harvard Business School Press, 2002.

[19] A. Gawer and R. Henderson, "Platform owner entry and innovation in complementary markets: evidence from intel," *Journal of Economics and Management Strategy*, vol. 16, pp. 1 – 34, 2007.

[20] M. Sonmat, "A review and critique of supplier selection process and practices," Tech. Rep., 2005. [Online]. Available: http://hdl.handle.net/2134/2160

[21] C. Hsu, V. Kannan, G. Leong, and K. Tan, "Supplier selection construct: Instrument development and validation," *The International Journal of Logistics Management*, vol. 17, no. 2, pp. 213–239, 2006.

[22] G. J. Burke, J. E. Carrillo, and A. J. Vakharia, "Single versus multiple supplier sourcing strategies," *European Journal of Operational Research*, 2006.

[23] N. Aissaoui, M. Haouari, and E. Hassini, "Supplier selection and order lot sizing modeling: A review," *Computers & Operations Research*, vol. 34, no. 12, pp. 3516–3540, 2007.

[24] W. Elmaghraby, "Supply contract competition and sourcing policies," *Manufacturing & Service Operations Management*, vol. 2, no. 4, pp. 350–371, 2000.

[25] R. Oliveira and J. Lourenco, "A multicriteria model for assigning new orders to service suppliers," *European Journal of Operational Research*, vol. 139, no. 2, pp. 390–399, 2002.

[26] A. Balakrishnan and J. Geunes, "Requirements planning with substitutions: Exploiting bill-of-materials flexibility in production planning," *Manufacturing & Service Operations Management*, vol. 2, no. 2, pp. 166–185, 2000.

[27] J. Geunes, "Solving large-scale requirements planning problems with component substitution options," *Computers and Industrial Engineering*, vol. 44, no. 3, pp. 475–491, 2003.

[28] M. O. Ball, C.-Y. Chen, and Z.-Y. Zhao, "Material compatibility constraints for make-to-order production planning." *Operations Research Letters*, vol. 31, no. 3, pp. 420–428, 2003.

[29] G. Da Silveira, D. Borenstein, and F. Fogliatto, "Mass customization: Literature review and research directions," *International Journal of Production Economics*, vol. 72, no. 1, pp. 1–13, 2001.

[30] K. Ramdas, "Managing product variety: An integrative review and research directions," *Production and Operations Management*, vol. 12, no. 1, pp. 79–101, 2003.

[31] A. Gunasekaran and E. Ngai, "Build-to-order supply chain management: A literature review and framework for development," *Journal of Operations Management*, vol. 23, no. 5, pp. 423–451, 2005.

[32] J. Jiao, T. Simpson, and Z. Siddique, "Product family design and platform-based product development: A state-of-the-art review," *Journal of Intelligent Manufacturing*, 2006.

[33] T. Simpson, "Product platform design and customization: Status and promise," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 18, pp. 3–20, 2004.

[34] M. Meyer and A. Lehnerd, "The power of product platforms: Building value and cost leadership," 1997.

[35] O. de Weck, E. Suh, and D. Chang, "Product family and platform portfolio optimization," in *Proceedings of the ASME International Design Engineering Technical Conference (DETC'03)*, 2003.

[36] J. Jiao, Y. Zhang, and Y. Wang, "A generic genetic algorithm for product family design," *Journal of Intelligent Manufacturing*, 2006.

[37] M. D. P. L. Aversano and K. Taneja, "A genetic programming approach to support the design of service compositions," in *First International Workshop on Engineering Service Compositions (WESC'05), Amsterdam, The Netherlands*, 2005.

[38] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*. ACM Press, 2005, pp. 1069–1075.

[39] W. Chang, C. Wu, and C. Chang, "Optimizing dynamic web service component composition by using evolutionary algorithms," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, 2005, pp. 708–711.

[40] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for QoS-aware web service composition," in *Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, 2006, pp. 72–82.

[41] A. Atamtürk and M. Savelsbergh, "Integer-programming software systems," *Annals of Operations Research*, vol. 140, no. 1, pp. 67–124, 2005.

[42] A. Caprara, P. Toth, and M. Fischetti, "Algorithms for the set covering problem," *Annals of Operations Research*, vol. 98, no. 1, pp. 353–371, 2000.

[43] G. Guisewite and P. Pardalos, "Minimum concave-cost network flow problems: Applications, complexity, and algorithms," *Annals of Operations Research*, vol. 25, no. 1, pp. 75–99, 1990.

[44] P. Buxmann, J. Strube, and G. Pohl, "Cooperative pricing in digital value chains – the case of online music," *Journal of Electronic Commerce Research*, vol. 8, no. 1, 2007.

[45] A. Tsay, S. Nahmias, and N. Agrawal, *Quantitative Models for Supply Chain Management*. Kluwer Academic Publishers, 1999, ch. Modeling supply chain contracts: A review, pp. 299–336.

[46] G. Cachon, *Handbooks in Operations Research and Management Science: Supply Chain Management*. North-Holland, 2002, ch. Supply chain coordination with contracts, pp. 629–646.