Internet social networking - Distinguishing the phenomenon from its manifestations in web sites

Thomas Widjaja
*Technische Universität Darmstadt*, widjaja@is.tu-darmstadt.de

Peter Buxmann
*Technische Universität Darmstadt*, buxmann@is.tu-darmstadt.de

Follow this and additional works at: [http://aisel.aisnet.org/ecis2009](http://aisel.aisnet.org/ecis2009)

**Recommended Citation**

Widjaja, Thomas and Buxmann, Peter, "Internet social networking - Distinguishing the phenomenon from its manifestations in web sites" (2009). ECIS 2009 Proceedings. 153.

[http://aisel.aisnet.org/ecis2009/153](http://aisel.aisnet.org/ecis2009/153)
SERVICE-ORIENTED ARCHITECTURES: MODELING THE SELECTION OF SERVICES AND PLATFORMS

Widjaja, Thomas, Technische Universität Darmstadt, Hochschulstraße 1, 64289 Darmstadt, Germany, widjaja@is.tu-darmstadt.de
Buxmann, Peter, Technische Universität Darmstadt, Hochschulstraße 1, 64289 Darmstadt, Germany, buxmann@is.tu-darmstadt.de

Abstract

The Service-Oriented Architecture (SOA) paradigm promises to facilitate the integration of software services provided by different vendors and thus enables users to benefit from Best-of-Breed solutions. In order to support software architects we present the Multilayer Standardization Problem (MSP) to analyze the trade-off between possibly enhanced utility versus higher assembling costs of Best-of-Breed SOA solutions. We implemented a software prototype to support decision makers during the data input and the subsequent analysis of the solution’s robustness. The MSP for the SOA-case is formulated as a linear 0–1 optimization model and extends the established Standardization Problem (SP) by modeling the user preferences and considering varying granularity as well as integration relationships in addition to communication relationships. These characteristics are common to numerous systems – thus the general MSP can serve as a basis for further research in this field.

Keywords: Compatibility, Decision Support Systems, Service-Oriented Architecture (SOA), Standards
1 INTRODUCTION

The basic idea behind the SOA paradigm is the support of business processes by IT systems consisting of services. Those services are clearly encapsulated, and loosely coupled entities, which deliver a defined business functionality. (Erl 2006, p. 290ff; Papazoglou & van den Heuvel 2007, p. 389). Current literature accredits SOA-based software systems various benefits compared to traditional monolithic systems such as enhanced agility, straightforward integration of heterogeneous IT environments, etc. (Krafzig et al. 2006, p. 251 ff). However, in order to leverage those benefits, new or adapted methods and tools are needed to support decision makers.

The technical realization of the SOA paradigm is the focus of various research efforts, see e.g. (Erl 2006; Krafzig et al. 2006). The SoA concept itself is technology-independent; however SOA is mostly seen in direct correlation with the Web-Service technology and associated standards. This technical standardization significantly decreases the costs of integrating software services implemented by different vendors, thus making the realization of Best-of-Breed software systems more feasible. (Erl 2006, p. 63) This paper focuses on the trade-off between possibly enhanced utility versus higher assembling costs of Best-of-Breed SOA solutions, i.e. solutions in which the SOA system is composed of components that are provided by different (and often specialized) vendors.

Software systems based on the SOA paradigm can be structured in two layers: Based on an integration platform (layer 0), loosely coupled services (layer 1) are combined to support business processes (see Figure 1). In this paper the term integration platform is used in a broad sense to refer to all infrastructure components of an SOA-based software system, e.g. enterprise service bus (ESB), service repository, application server, and process server. See (Krafzig et al. 2006) for a more detailed description of the integration platform. This abstract and two layered perspective is a simplification but it is possible to extend the proposed model in order to consider the SOA platform in greater detail (see chapter 5 for further research).

![Figure 1. Two layers of a SOA-based software system.](image)

In the following, we will focus our analysis on the market for Enterprise Resource Planning (ERP) software. In this market, some software vendors offer services and platforms, e.g. IBM (WebSphere), Oracle (Fusion) and SAP (NetWeaver), while others provide only a platform, e.g. Red Hat (JBoss Enterprise SOA Platform), or only services (e.g. a specialized demand forecasting service in the context of inventory management). The discussed decision problem is related to a broad literature stream analyzing the process of selecting (web) services with differing Quality of Service attributes in order to maximize user satisfaction – see e.g. (Zeng et al. 2004) for a middleware platform that addresses related issues; (Canfora et al. 2005) for a composition approach based on genetic algorithms, (Yu et al. 2007) for a related knapsack and optimal path problem; and (Blau et al. 2009) for a mechanism design approach. However, to our knowledge no other optimization model yet considers the selection of integration platforms and services and explicitly analyzes the trade-off between the possibly enhanced utility versus the higher assembling costs of Best-of-Breed SOA solutions. Such a model that investigates the decision problem from the standardization perspective could be useful for several reasons. First, it supports software architects during the selection of the services and platforms. “Intuitive” approaches can result in suboptimal solutions: An example is the selection of the set of services and the platform which provides the highest “net utility”, i.e. the highest difference between utility and necessary implementation costs (i.e. neglecting information and integration cost). Second,
the presented model can support the software architect in communicating decisions (especially in cases in which it is optimal to choose not the service preferred by the users). Third, such a model provides a first step towards a theoretical basis for the analysis of Best-of-Breed solutions in SOA-based software systems. Furthermore, the presented model can be applied to various domains beyond the SOA-case. Thus, general insights regarding the described decision problem may be even more valuable than the actual application of the model in a concrete situation. In this paper we present the Multilayer Standardization Problem (MSP) in order to support software architects determining the optimal set of services (layer 1) and platforms (layer 0). The MSP considers the following three cost categories: costs for services and platforms, costs of integrating services mutually, and costs of integrating services and platforms. We will keep the model as simple as possible in order to focus on the trade-off between possibly enhanced utility of a Best-of-Breed solution versus the higher assembly cost. Note that, the proposed model examines the described decision problem from a standardization perspective and therefore does not consider all factors that can influence the selection of services and platforms – e.g. the vendor choice is partially driven by strategic considerations and various difficult to quantify “soft factors”.

This paper is divided into five sections. Following this introduction, we will describe the MSP based on the Standardization Problem (SP). In the third section, we will present a general schema to apply the MSP to the SOA-case and a software prototype to support decision makers. Section four discusses the limitations of the presented model, and the paper closes with conclusions and possible directions for further research.

2 A MODEL TO SUPPORT THE SELECTION OF SERVICES AND PLATFORMS

In two following subsections the modeled aspects which cause the assembly cost advantage of “single-vendor” solutions are discussed. In the third subsection it is discussed how varying user preferences can result in a trade-off between the possibly enhanced utility of a Best-of-Breed solution versus the higher assembly cost. In the last subsection a possible formulation of the MSP as linear 0–1 optimization problem is presented.

2.1 Common standards facilitate the communication between services

In this section, we examine the service layer of a SOA-based software system in more detail. We focus our analysis in particular on aspects that promote a homogenous service landscape. The basic underlying assumption is that common communication standards implemented by the communication partners facilitate information exchange. There are various examples of situations in which communication standards simplify the exchange of information: a common spoken language during a discussion between two individuals, a common file format for two different word processors which exchange data, a semantic definition of business terms during the exchange of XML files between web services. We assume that all services provided by a certain vendor implement the same communication standard. Thus, the exchange of information between software services provided by the same vendor is facilitated, whereas it is not in the case of cross-vendor communication. Note that throughout this article, the term “communication standard” is used to refer to technical (e.g. data formats) as well as semantic aspects (e.g. meaning of business terms).

The economic analysis of standardization decisions is mainly shaped by the network effect theory. Network effects occur if the utility that a user derives from a good depends on the number of other users of the same kind of good. (Farrell & Saloner 1985; Katz & Shapiro 1985) Communication standards show positive network effects: e.g. the more people who speak a certain language (thus the more possible communication partners exist), the more utility this language provides to the individual “adopters”. In the SOA-case the following example is illustrative: The more information a certain functionality exchanges with (already implemented) services that are provided by a specific vendor, the more beneficial the use of a service provided by this specific vendor for the analyzed functionality is. A broad stream of literature analyzes the implications for users and vendors of network effect goods. Topics on which these works focus include start-up-phenomena (see e.g. (Oren & Smith 1981;
path dependencies (see e.g. (Arthur 1989; David 1985; Liebowitz & Margolis 1995)), and "tippy networks" (see e.g. (Arthur 1989; Besen & Farrell 1994; Katz & Shapiro 1994)). Based on these insights, various strategies for users and vendors of network effect goods have been derived, see e.g. (Buxmann 2001) for selected results on the software industry. Most of the models regarding network effect theory are based on an aggregated perspective and thus abstracting from individual relations between the different actors. In contrast the Standardization Problem (SP) (Buxmann et al. 1999) explicitly considers these relationships and thus seems to be a sound foundation for the examination of the service layer. The SP considers the following trade-off: On the one hand, the adoption of communication standards (languages, file formats, or XML formats) facilitates information exchange. On the other hand, the implementation of communication standards often involves costs. The SP answers the resulting question: Which actor should implement which standard to maximize the aggregated savings through standardization with respect to the costs? Various research has been performed related to the SP; see e.g. (Buxmann et al. 1999; Miklitz & Buxmann 2007; Schade & Buxmann 2005; Weitzel et al. 2006; Westarp et al. 2000). One major focus of the research related to the SP is how the degree of autonomy of the different actors effects the structure of the optimal solutions; see e.g. (Buxmann et al. 1999; Weitzel et al. 2006). In the case of the centralized SP a superior instance determines – often based on perfect information – the set of standards each actor should implement. In contrast to that, in the case of the decentralized SP, every actor (or a coalition of actors), decides autonomously whether to implement a certain standard or not. Due to the analysis of the software architects perspective, i.e. a “central” decision maker, we use the centralized SP as a basis for our research. Furthermore several studies regarding efficient solution techniques of the SP have been performed. (Domschke & Wagner 2005; Kimms 2003)

Various aspects of the SP can be directly applied to the problem of selecting services and platforms for a SOA. Parallel to the SP the foundation of the MSP is a graph: Based on the business processes a function graph \( G_{\text{Com}} := (N_{\text{Service}}, E_{\text{com}}) \) with \( E_{\text{com}} \subseteq \{(i,j) | i,j \in N_{\text{Service}} \text{ and } i < j \} \) is defined. The vertices of this graph \( i \in N_{\text{Service}} \) represent (IT-) functions that support the business processes. A software architect will typically use the proposed model and the respective Decision Support System (DSS) to analyze coarse-grained services and high-level business processes. Nevertheless, the level of abstraction can vary: A vertex can represent a complex function, e.g. a set of CRM functions or also a elementary function, e.g. a simple credit assessment. Based on the analyzed business process, information exchange between functions \( i \) and \( j \) may be necessary. This is modeled by the edge \((i,j)\) in the function graph. Note that not every function necessarily exchanges information with all other functions, i.e. this graph does not need to be complete. We assume that each function \( i \in N_{\text{Service}} \) can be fulfilled by services \( r \in R_{\text{Service}} \) provided by various vendors. Some vendors provide a wide variety of services (e.g. SAP and Oracle) but other (more specialized) vendors offer only services for selected functionalities. Some services fulfill exactly one function (e.g. a service for the credit assessment) while other, more coarse granular services can fulfill several of the demanded functions (e.g. the core banking module of a large ERP system vendor). The parameter \( \alpha'_r \) takes the value 1, if and only if service \( r \) can fulfill function \( i \), otherwise the value is 0. The set \( R'_{\text{Service}} := \{r \in R_{\text{Service}} | \alpha'_r = 1 \} \) contains all services that can fulfill the function \( i \). The acquisition or implementation of a service \( r \) incurs costs, e.g. additional hardware, software licenses, and training. We model these costs in the parameter \( \alpha' \) and refer to these costs as implementation costs. Already existing services, i.e. an installed base, can be modeled by \( \alpha' = 0 \) and services that are realized as facades for legacy systems typically incur lower implementation costs than services provided by external vendors. We assume that every function has to be implemented by exactly one service to derive a valid SOA system (see Miklitz & Buxmann 2007) for a similar assumption – they propose an extension of the SP to support decision makers during the selection of IT systems in the context of Mergers & Acquisitions).

A basic assumption of the MSP is that the information exchange between services is realized via communication standards. Each service \( s \in R_{\text{Service}} \) is modeled as a bundle of a communication standard (i.e. the associated semantic and syntactic standards) \( k' \in K \) and a software product that provides certain (business) functionality. Although the SOA paradigm can be realized by several technologies, current discussion focuses on the technology of web services, see e.g. (Alonso et al. 2004; Newcomer & Lomow 2005). Thus, it is often assumed that services based on this technology can be combined
easily. We argue that this perspective is overly simplified, since the standardization to the technology of web services in particular affects the syntactical compatibility of services. However, the composition of SOA-based systems by simply plugging together existing building blocks is often inhibited by semantic differences between the services. Furthermore, various vendors extend the existing web service standards by proprietary increments. In our opinion, it is necessary to model the communication of services as point-to-point communication until a technical and semantic standard (in the respective company) exists. Therefore we assume that each vendor equips all offered services \( r \in R_{\text{service}} \) with a certain communication standard \( k' \in K \). Especially “big” software vendors attempt to establish de-facto standards, and niche developers are often aligned to such a standard. Therefore, “standard-ecosystems” evolve around the standards of big players. (see for related aspects (Petrie & Bussler 2008)) Figure 2 shows a simple function graph: services \( s_1 \) and \( s_2 \) are offered by the same vendor \( B \), i.e. \( k^B = k^B = B \), which is indicated by the same shading. Furthermore, \( \alpha^v_1 = \alpha^v_2 = 1 \), since \( s_2 \) supports function \( 1 \) as well as function \( 2 \).

Figure 2 The business process as basis for the function graph.

We pointed out that communication between services provided by the same vendor is “facilitated”. In order to consider this phenomenon we model information cost for each pair \((s,t)\) of service candidates \( s \in R_{\text{service}} \) and \( t \in R_{\text{service}} \) that can fulfill functions \( i \) and \( j \) with \((i,j) \in E^{\text{com}}\). Information costs model the cost for the information exchange, e.g. costs that are associated with pre- or post-processing of data, which is partially caused by incompatibilities between the communication standards. For a possible way to evaluate information cost and a more detailed description of the concept, i.e. the incorporation of possibly higher value of the exchanged information, see e.g. (Buxmann et al. 1999). The information costs are modeled by the parameter \( c_{i,j}^{s,t}(i,j) \in E^{\text{com}} \) and \( s \in R_{\text{service}} \) and \( t \in R_{\text{service}} \). (see the right side of figure 2). We assume that, \( c_{i,j}^{s,t} \leq c_{i,j}^{s',t} \) if \( k^s = k^{s'} \) and \( k^t = k^{t'} \), i.e. information cost for the communication between services provided by the same vendor are at most equal to information cost for the information exchange between services offered by different vendors. The justification for this assumption is analogous to the argumentation of (Buxmann et al. 1999) for the SP: Technical and semantic incompatibilities between the communication standards cause additional information cost. This causes a network effect for communication standards of the services: As more functions are implemented by services provided by a distinct vendor (i.e. fulfill a distinct communication standard), the more beneficial (from the information cost perspective) the usage of services from this specific vendor (i.e. this communication standard) for the remaining functions with which information is exchanged becomes. Note that, the strength of the network effects depends on the availability, comprehensiveness, and diffusion of domain specific (open) communication standards. If a (more coarse grained) service \( r \in R_{\text{service}} \) is able to implement two communicating functions, i.e. \((i,j) \in E^{\text{com}} \) and \( \alpha_i' = \alpha_j' = 1 \), cost for information exchange can be modeled but typically those costs will be close to zero.

The function graph of the MSP is not directed, though the chronological and logical structure of business process implies a directed graph. By aggregating the mutual communication cost between the service candidates of two functions \( i \) and \( j \) into one parameter \( c_{i,j}^{s,t} \), the directed graph can be simplified; this is analogous to the corresponding modification in the classical SP. (Domschke & Wagner 2005)
2.2 Interplay of services and standards: The layer structure

The classical SP focuses on communication relationships among actors and examines the trade-off between standardization costs and possible savings of information cost between communication partners by standardization. The MSP extends the SP by considering a second type of relationship: the integration relationship. This extension is necessary to analyze the described decision problem in the SOA context because the costs to use a service (layer 1) depend to a significant degree on the integration platform (layer 0) the service is supposed to run on (see Figure 1). Since many platform vendors “enrich” open integration standards by various proprietary extensions, services provided by a given vendor typically operate better on that vendor’s platform than on any other platform. For example, the integration of a SAP service with the NetWeaver platform is often easier than with any other platform. These extensions allow for example better performance or ensure a higher security level. Typically, the services that are offered by the platform vendor take advantage of these extensions without any modification. Vendors that only provide services – but no own platform – often “align” (and certify) their services for use with a specific platform. Similar problems are discussed in the literature related to “Mix and Match” and the “Hardware Software Paradigm” (see e.g. (Economides 1996; Einhorn 1992; Katz & Shapiro 1994; Matutes & Regibeau 1988)). To model these aspects in the SOA-case, two layers are necessary: Layer 1 consists of functions \( i \in \mathcal{N}_{\text{Service}} \) that are fulfilled by services provided by possibly different vendors in order to support a business process. On layer 0 – the platform layer – one abstract function \( n_p \) (i.e. the integration platform) is modeled (see Figure 3). Thus, the graph of the MSP for the SOA-case consists of one service layer with \( n \) vertices (layer 1) and one platform layer (layer 0) with one vertex \( n_p \). As already described, the term “integration platform” relates to all infrastructure components that are necessary to support the services. The set \( N \) of functions is defined as \( N := \mathcal{N}_{\text{Service}} \cup \mathcal{N}_{\text{Platform}} \) with \( \mathcal{N}_{\text{Platform}} := \{ n_p \} \). In the following we subsume services and platforms as artifacts. The set of artifacts is defined as \( R := R_{\text{Service}} \cup R_{\text{Platform}} \). Analogous to \( R_{\text{Service}} := \{ r \in R_{\text{Service}} | \alpha'_i = 1 \} \) the sets \( R_{\text{Platform}} := \{ r \in R_{\text{Platform}} | \alpha'_i = 1 \} \) and \( R := \{ r \in R | \alpha'_i = 1 \} \) are defined. Analogous to services the implementation of a platform \( p \) incurs implementation cost \( d^p \). In a first step, we assume that platforms fulfill no other functions except \( n_p \), i.e. \( \alpha'_i := 0 \forall i \in \mathcal{N}_{\text{Service}} \) and \( r \in R_{\text{Platform}} \). Furthermore, we assume that a valid solution implements exactly one platform – federated system landscapes with multiple “platforms” can be modeled in the extended model (see chapter 5).

The MSP considers vertical (in)compatibilities by integration relationships and a compatibility graph weighted with integration costs \( d^p \) (see Figure 3 for an example). In the SOA context integration costs can be interpreted as costs that are necessary to make a service executable on a certain platform (e.g. cost for the deployment and fine adjustment).

Figure 3 Integration relationships weighted with integration cost.

We assume that each service \( s \) has to be integrated with a platform \( p \) to receive a functional SOA system. The integration of a service and a platform is realized via interfaces, which fulfill “integration standards”. Note that besides the communication standards on the service layer, integration standards between the two layers also influence the assembly cost of the SOA system. For the purpose of simplification, we assume that each vendor offers exactly one integration standard for all artifacts (i.e. services and platforms). This allows us to summarize integration and communication standards for the artifact \( r \in R \) in the parameter \( k' \in K \). Thus \( k' \in K \) can be interpreted as vendor of artifact \( r \). Full-
range suppliers such as SAP or Oracle offer services besides the platforms to fulfill nearly every functionality – thus in such cases a “full standardization” on both layers is possible. Pairs of services and platforms \((s,p)\) which are provided by the same vendor fulfill \(k^s = k^p\). The integration of such artifacts is often preconfigured, well documented, and supported by specialized tools. For these reasons we assume that \(d^{sp} \leq d^{pq} \) holds, if \(k^s = k^p \) and \( k^s \neq k^q \). If a certain service \(s\) is totally incompatible to a platform \(p\), \(d^{sp} := M\) holds.

2.3 Preferences for services and platforms

The aspects of the decision problem already described – i.e. the possible savings of integration and communication cost by choosing only one vendor to provide both the demanded services and the platform advocate the choice of a “single vendor” solution. However, in some situations, there is no single vendor who offers artifacts for all necessary functions. Another reason to decide on more than one vendor is, that the users have different preferences regarding the services and platforms and not all preferred artifacts will likely be provided by a single vendor. In order to consider the preferences of the users, it is necessary to model the utility of the services and platforms. The preference of the user for artifact \(r \in R\) implemented in function \(i \in N\) is expressed in monetary units in the utility parameter \(b_i^r\). The monetary estimation of the preference can be supported by a weighted decision-matrix. The user defines a set of weighted functional and non-functional criteria upon which the potential artifacts are scored. An example of possibly relevant criteria for service candidates are maximal data transfer rate, security aspects, and the design of the graphical user interface. Note that the utility \(b_i^r\) of an artifact is function specific but the implementation cost \(a_i^r\) are associated with the artifact itself. This means that the cost \(a_i^r\) are incurred if artifact \(r\) fulfills at least one function. Note that the model allows to consider “function-specific” costs by subtracting them from the utility \(b_i^r\) of the artifact - receiving a “implementation specific net-utility”. For simplification, we assume throughout this article that all costs are “artifact-specific”. This aspect is related to a major difference between communication and integration relationships. Communication relationships exist between implementations of functions by artifacts, whereas integration relationships exist directly between artifacts, i.e. the service-platform pairs \((s,p)\). Furthermore, the integration relationship is directed: a distinct service needs a platform to operate, not vice versa. If the first service is used, at least one platform has to be implemented to host this service and then for subsequent implemented services the usage of this platform is “free of charge”. This points out the infrastructure character of the platform. Since every function \(i \in N = N_{Service} \cup N_{Platform}\) has to be fulfilled by one artifact in the presented version of the model, this infrastructure character is hidden.

(Chou 2007) advertises that in hybrid systems, i.e. systems that are composed of components provided by different vendors, utility losses due to incompatibilities are possible. We observed in the SOA-case that not every service-platform combination \((s,p)\) harmonizes in the same manner. For example, the data transfer rate of a certain service \(s_i\) could be vastly higher than the maximum data transfer rate of a platform \(p_i\). This means that the whole combination of \(s_i\) and \(p_i\) can only operate at the data transfer rate of the platform. Thus the relevant characteristics of the complete system are determined to a certain extent by the composition of the system and not only by the addition of the individual utilities. We propose the following approach: During the assessment of the artifacts, an “ideal” environment should be assumed – thus no utility losses due to incompatible complementary components should be considered. Afterwards the assumed utility losses can be integrated by additional costs added to the parameter \(d^{sp}\).

2.4 Model for the MSP in the SOA-case – A mathematical formulation

We introduce four binary variables \(w', x_i'^r, y_i'^p, z_i'^p \in \{0,1\}\) for the formulation of the MSP as a linear optimization problem of the MSP. Table 1 summarizes the introduced notation.

The objective function maximizes the utility derived by the artifacts implementing the functions with respect to the implementation, information, and integration costs. Note that all terms are expressed in monetary units.
### Table 1 Notation of the model for the MSP

The model for the SOA-case MSP is formulated as follows:

**Maximize** \( F(w, x, y, z) = \sum_{i \in N} \sum_{r \in R^i} b'_i x'_r - \sum_{r \in R} \alpha'_r \cdot w'_r - \sum_{(i, j) \in E^{Comm}} \sum_{s \in R_{Service}} \sum_{t \in R_{Service}} c''_{ij} y''_{ij} - \sum_{s \in R_{Service}} \sum_{p \in R_{Platform}} d^{sp} z^{sp} \)** \[1\]

s.t.

\[\begin{align*}
x'_r - w'_r & \leq 0 \quad \forall i \in N \text{ and } r \in R^i \quad \text{[2]} \\
x'_r & = 1 \quad \forall i \in N \quad \text{[3]} \\
y''_{ij} - x'_r & \leq 0 \quad \forall (i, j) \in E^{Comm} \text{ and } s \in R_{Service} \text{ and } t \in R_{Service} \quad \text{[4]} \\
y''_{ij} - x'_r & \leq 0 \quad \forall (i, j) \in E^{Comm} \text{ and } s \in R_{Service} \text{ and } t \in R_{Service} \quad \text{[5]} \\
\sum_{s \in R_{Service}} \sum_{t \in R_{Service}} y''_{ij} & = 1 \quad \forall (i, j) \in E^{Comm} \quad \text{[6]} \\
w'^i - \sum_{p \in R_{Platform}} z^{sp} & \leq 0 \quad \forall s \in R_{Service} \quad \text{[7]} \\
z^{sp} - x'_r & \leq 0 \quad \forall s \in R_{Service} \text{ and } p \in R_{Platform} \quad \text{[8]} \\
w'_r, x'_r, y''_{ij}, z^{sp} & \in \{0, 1\} \quad \forall (i, j) \in E^{Comm} \text{ and } i' \in N \text{ and } p \in R_{Platform} \text{ and } r \in R \\
\text{and } r' \in R' \text{ and } s \in R_{Service} \text{ and } s' \in R_{Service} \text{ and } t \in R_{Service} \quad \text{[9]} \\
\end{align*}\]
The set of constraints [2] guarantees that variable $w_r$ has to take the value 1 if at least one implementation of the artifact $r$ is part of the solution. Note that $w_r$ can take the value 1 if $a_r = 0$, even if the artifact is not implemented. Constraint [3] guarantees that each function is implemented by exactly one artifact. This constraint is necessary since all functions have to be fulfilled by a service in order to receive a valid SOA. Note that the constraints [4] and [5] are similar to the formulation of the SP by (Domschke & Wagner 2005). Constraint [6] ensures that each pair of communicating functions incurs information costs. Constraint [7] ensures that each service is hosted on one platform. Constraint [8] guarantees that, if a platform hosts any services, this platform must be implemented. The set of constraints in [9] specifies the binary domains of the variables.

3 THE APPLICATION OF THE MSP IN THE SOA-CASE

Figure 4 shows the general schema of the application of the MSP in the SOA-case. We implemented a Eclipse-based software prototype to support decision makers during data input (e.g. import of BPEL-based process descriptions), to solve the MSP and to analyze the solution’s robustness (figure 5).

![Figure 4 The application of the the MSP in the SOA-case. The whole process is supported by the DSS.](image)

The input parameters of the MSP are classified in four groups: Input 1) the communication graph $G_{\text{Com}} = (N_{\text{Service}}, E_{\text{Com}})$ on the service layer (layer 1), Input 2) information regarding the artifacts (i.e. implementation cost $a_r$, functions that the artifact can implement $a_r'$, function-specific utility $b_r'$, and vendor of the artifact $k_r'$), Input 3) information cost $c_{ij}$ for the relevant pairs of services, and Input 4) integration cost $d_{r}^w$ for the service-platform combinations. In the first step, the functions that should be covered by the SOA system must be identified, i.e. the set $N$. Note that, a finer granularity analysis of the functions induces a higher complexity during the data collection and solution process of the resulting MSP instance. Based on the relevant business processes the communication relationships $(i,j) \in E_{\text{Com}}$ between the functions have to be assessed. The software prototype allows to import existing BPEL process descriptions (e.g. created in other process modeling tools). In the next step, different service and platform candidates $R = R_{\text{Service}} \cup R_{\text{Platform}}$ must be assigned to the functions in order to determine $a_r'$. It seems to be useful to consider only those service and platform candidates that are able to fulfill mandatory requirements of the decision makers (especially with regard to security and data transmission aspects). The user has to assess for each artifact the implementation cost $a_r$ and the (expected) utility $b_r$. Based on the expected intensity of communication and the degree of the compatibility of the communication standards, the information cost $c_{ij}$ for all pairs of functions $(i,j) \in E_{\text{Com}}$ are assessed and finally, the integration costs $d_{r}^w$ for each service-platform combination are estimated. Based on these parameters the (under the given assumptions) optimal configured SOA is determined by the open source Mixed Integer Programming (MIP) solver “lp_solve” (version 5.5). (see http://lp.solve.sourceforge.net for details)
Different types of parameter validations

Graphical model of the business process

Service and platform candidates

Input of the parameters related to the currently selected artifact

Graphical editing tools

Figure 5 Screenshot of the software prototype. (View: Input of the problem description).

The configuration is optimized with respect to the information cost \( c_i^w \), integration cost \( d_r^w \), the user preference \( b_i^r \), and the standardization costs \( a_r^i \) of each artifact and function. The implemented DSS allows to conduct a Monte Carlo analysis with various graphical outputs to visualize the robustness of selected solutions.

4 LIMITATIONS OF THE PROPOSED MODEL

The limitations of the MSP model are similar to the limitations of the classical SP models, which are well studied in the literature. According to e.g. (Schade & Buxmann 2005) three types of limitations of the SP model exist; the data problem, the complexity problem and the implementation problem. The data problem refers to the difficulty collecting accurate data for the necessary parameters. Note that the MSP requires additional parameters compared to the SP. In general, the user must consider to which extent additional costs incurred by data collection (in particular regarding the mentioned “soft factors”) are justifiable by the improvement of data-quality. Especially the determination of parameter \( b_i^r \) is related to the general problem of evaluating an IT-system’s economic value. However, IT-investment-decisions are economic decision-problems and therefore economic methods should be applied. Preliminary experiences from two case studies currently being conducted indicate that it is better to estimate the parameters in workshops with enterprise architects, rather than solely based on the (existing) business case for the investment decision. If detection and assignment of utility in a specific instance of the problem is too expensive, the cost-minimal solution for this SOA system can be achieved by defining \( b_i^r := 0 \forall i \in N \) and \( r \in R^i \). In this case, the parameter \( d_r^w \) does not incorporate costs for utility losses due to incompatibilities in hybrid systems. The complexity problem arises in situations where bigger instances of the SP are solved, i.e. complexity increases when the number of vertices in the graph or the number of available artifacts increases. (Schade & Buxmann, 2005) By transformation into a warehouse location problem (WLP) it can be shown that the SP with more than two standards is NP-hard. (Domschke & Wagner, 2005) The implementation problem refers to the difficulties that arise when the implementation of a certain configuration is to be enforced.

The aim of the MSP is to analyze the described decision problem from a standardization perspective, knowing that various other relevant perspectives exist which are also of possible interest; e.g. it is observable that some software architects prefer to ensure a certain heterogeneity of the IT architecture to reduce dependency on single software vendors. Although the suggested model is a static one, some parameters can exhibit dynamic aspects, e.g. operating costs. These dynamic rates of the parameters can be derived by estimation and discounting. Nevertheless the presented model doesn’t allow an iterative transition between different IT-architecture landscapes. A further general limitation of the proposed model is the simplified (i.e. deterministic and linear) objective function.
5 CONCLUSIONS AND AVENUES FOR FURTHER RESEARCH

The aim of this article was to analyze the optimization problem regarding the choice of services and platforms in a software system based on the SOA paradigm in order to support decisions of software architects. We address two interdependent questions of the decision makers, which arise during the composition of SOA-based software systems: Which set of services should be chosen, and which platform should host those services? We presented an application of the Multilayer Standardization Problem (MSP) to the two layered SOA-case, its formulation as a linear 0–1 optimization problem and a according Decision Support System (DSS). The MSP extends the Standardization Problem (SP) by incorporating differing user preferences (utility of the artifacts), varying granularity of services and considering integration relationships in addition to communication relationships.

The general MSP is able to analyze more complex systems (n layers), which is useful to model the integration platform in a more detailed way, i.e. to model the integration platform as a system which itself contains communication and integration relationships. Another application of the general MSP is the examination of other domains e.g. a PC system with three layers: layer 0 (hardware), layer 1 (operating system), layer 2 (application software). A further possible direction for future research is the formulation of a dynamic decision model and the investigation of decentralized multilayer standardization problems. It would also be interesting to develop enhanced solution techniques for large problem instances – possible approaches include reformulations to provide tighter upper bounds for pruning in a Branch and Bound algorithm, or the development and use of (meta)heuristics. This article focused on the software architect’s perspective during the composition of SOA-based ERP systems. Future research regarding the analysis of strategies for software vendors’ can use the MSP to anticipate the reaction of users. In this context, strategies regarding compatibility and bundling (service/service as well as service/platform) are interesting topics.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. We also thank Stephan Faßbender, Joachim Kanbach and Florian Weichand for their valuable contributions to the implementation of the presented DSS. Thomas Widjaja’s research is supported by a research grant from the FAZIT-Stiftung.

References