9 July 2011

Model Based Identification And Measurement Of Reorganization Potential Through Service-Oriented Architectures

Martin Juhrisch
Dresden University of Technology, martin.juhrisch@tu-dresden.de

Gunnar Dietz
Dongbei University of Finance and Economics, mail@gdietz.de

ISBN: [978-1-86435-644-1]; Full paper

Recommended Citation
http://aisel.aisnet.org/pacis2011/87

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2011 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
MODEL BASED IDENTIFICATION AND MEASUREMENT OF REORGANIZATION POTENTIAL THROUGH SERVICE-ORIENTED ARCHITECTURES

Martin Juhrisch, Department of Mechanical Engineering, Dresden University of Technology, Dresden, Germany, martin.juhrisch@tu-dresden.de
Gunnar Dietz, Global Institute of Management and Economics, Dongbei University of Finance and Economics, Dalian, China, mail@gdietz.de

Abstract

The adoption of the service-oriented architecture (SOA) paradigm represents one major reconfiguration challenge of enterprises. It requires to break down business processes and resources, and to consequently disintegrate entire Information Technology (IT) environments. Thus, the implementation of a SOA can present serious challenges to professionals working in an organization. Not much is known so far about the process by which individuals seek to evaluate the potential impact of SOA paradigm in their business. Most decisions nowadays are done without a systematic analysis of the current situation. Those decisions often come with too high expectations about a success of a service-oriented application environment. Furthermore the necessary efforts to realize the SOA paradigm for the own business processes is often not systematically estimated and therefore not included into a comprehensive cost-benefit analysis. We present a methodology to develop a conceptual model of the enterprise by which individuals might assess the impact of a SOA before an IT implementation. The paper provides insights into how and why managers should react to the given challenges. The model recognizes that the dynamic readjustments not only impact on the business task level, but also on the organizational level of work tasks and role relations. The model provides a starting point for a further study of the evaluation criteria for professionals using a SOA.

Keywords: Conceptual Modeling, Service-Oriented Architectures, Service Identification, Convolution, Description-Kit Approach
1 INTRODUCTION

Management decisions concerning the service-oriented architecture (SOA) paradigm often either attempt to reconfigure entire enterprises or only consider thoughts on a technical integration via web services. However, both positions are only half of the story. To economically evaluate and realize the use of a SOA involves a deep understanding of organizational responsibility, workflows and processes.

Decision makers might expect an economic benefit from a close relationship between organization and system design (Hagel and Brown, 2003). In a SOA a process-oriented quality management controls if the actual process operation complies to the specifications and strategy of the organization. If process adjustments are necessary, the service that implements the functionality of a business process step can be identified directly. The ability to exchange information system functionality facilitates the implementation of an adaptation of the corresponding business processes. However, this benefit is not easily transferable because of a multitude of influencing factors that vary in different enterprises (Thomas and vom Brocke, 2009). Thus, the goals of a better support and customization of business processes are not reached today as the necessary transformation steps during the implementation of a business process into an executable workflow are difficult to understand.

Companies (or organizations like universities) often are not able to deal with this complexity without the help of consulting companies, so that such projects are often reduced to technical projects with low impact. This is often even the case when a lot of information about organizational structure and processes exists, for example from other process optimization projects. In many cases this information exists in the form of conceptual models. A natural question therefore is: Is it possible, at least semi-automatically, to use this information to put decision makers in a position to evaluate the impact of the adoption of a SOA and to generate a useful setting for configuring service compositions from their individual situation? The aim of this paper is to give a first positive answer to this question.

The remainder of the paper is organized by a brief background section introducing fundamental concepts before stating the research problem. The following method outlines a certain modeling approach — the so-called Description Kit Approach (DKA) — especially used to solve integration conflicts in modeling. The paper provides insights into the necessary foundation for delving into the problem of model-based evaluation. The evaluation section allows a reflection of findings in the broader context of service catalogs and the paper will complete with summarizing conclusion.

2 THEORETICAL BACKGROUND

2.1 Process Orientation

The increasing dynamic of the economy has caused a paradigm shift in the field of business organization from a functional thinking to process-orientation. This process-orientation is derived from the organizational doctrine of subordinating organizational structure to organizational workflow management (Hammer, 1990); (Hammer and Champy, 1993).

Organizational structure is concerned with responsibilities within the organization, i.e. with the planning, structuring, and distribution of the organization into smaller units (like departments), as well as the coordination of these (Grochla, 1982; p. 24). In contrast, workflow management focuses on how the work is done, i.e. workflows and procedures. Several steps of workflows are arranged and coordinated

1Sections 2.1 and 2.2 follow quite closely Weller (2010)
Questions about the content of the work, work time, and work place have to be answered (Wöhe, 1978; p. 160), as well as who is doing which part of the workflow.

Functional thinking therefore first focuses in structuring an organization into organizational units. The definition of necessary workflows is just secondary (Gaitanides and Ackermann, 2004; p. 5). For an appropriate structuring of processes, however, it is necessary to respect workflows within the organization when structuring the organization, which means to prioritize the workflow management in favor of organizational structuring (Gaitanides, 1983; p. 62).

Evidence showed that a classical functional organization prevents a functional spanning optimization of processes. Process orientation therefore focuses on a horizontal viewpoint on the organization that is independent for this functional partition (i.e. vertical structure) (Gadatsch, 2005; p. 7). This transition from functional thinking to process-orientation therefore is often called “90-degree-shift” (Osterloh and Frost, 1996; p. 30).

A process normally is understood as actions or procedures that have a common chronological dynamic (Bogaschewsky and Rollberg, 1998; p. 185). Organization theory defines a process as a set of tasks, which have a certain chronological order and serve one or several organizational goals (Nordsieck, 1972; pp. 9). These task can be ordered sequentially or parallelly (Gaitanides, 1983; p. 83), while the term task is defined by Kosiol as “…objectives for human actions with a certain aim…” (Kosiol, 1962; p. 43).

Furthermore, the term process refers to a set of similar tasks, that are often considered simultaneously and therefore need a common description. These processes are often done several times during daily routine (like a production process), nevertheless the term process is used in the singular. A process therefore refers more to a type of a task instead of a certain task. For a certain task we will use the term process instance instead, which means that a process instance refers to a concrete execution of a task. Each process may have several process instances that may be executed a different times or parallelly (Stormer and Knorr, 2001; p. 149); (Schmidt, 2002; p. 2).

All processes are operating on one or several (business) objects (Ferstl and Sinz, 2001; p. 186). If we define a process as a transformation (Osterloh and Frost, 1996; p. 31); (Schmidt, 2002; p. 1), the start and end of a process follow from the status of this object (these objects) before and after the process execution. This transformation can be described in detail by changes of attributes, or more abstract, by events (Scheer, 1998b; p. 18). However, a process may not only operate on one (or several objects), but also create or consume objects. This means that processes also can be described with help of input and output. Input may be not only (material) objects, but also operators (persons or machines) that are necessary to perform the task (Kosiol, 1962; p. 43). In producing environments the output is often the product that should be produced.

One process may produce an output that another process needs as input. This creates a relationship like supplier and customer. A supplier is creating the output for one or several certain customers (Davenport, 1993; p. 5); (Ferstl and Sinz, 2006; p. 190). Supplier and customer in this case may be internal or external (Scheer, 1998b; p. 13). An organizational unit that represents an internal customer that is responsible for an efficient and effective execution is called the process owner.

Processes can be examined with different levels of details (Scheer, 1998a; p. 23). Like several task can be grouped together to a process, several processes can be grouped into a new, higher level process again. On the other side, tasks can be seen as processes and decomposed into several more simple tasks.

We refer to parts of a process as sub-processes (that can be seen as whole processes from another viewpoint), no matter of their complexity. Often a process with highest granularity may be called a business process, while its sub-processes would not be called like that. We will use this term synonymously to the term process.
2.2 Process Optimization and Process Management

Process optimization is the planned redesign or creation of processes with the goal to maintain or increase the ability to compete (Rupper, 1994; p. 9); (Stoddard and Jarvenpaa, 1995; p. 82). Parallel to a reorganization of processes in many cases also a reorganization of organizational structures, information objects, as well as information systems is done (Österle, 1995; p. 13); (Gadatsch, 2005; p. 23). However, the process is still the central viewpoint and the reorganization of other elements of an organization is aligned to the reorganization of processes.

To perform the process optimization in most cases single activities within a process are changed, removed, or added (Gadatsch, 2005; p. 15); (Mansar and Reijers, 2007; p. 199). Furthermore attempts are made to enhance efficiency by putting activities in parallel, by avoiding media discontinuities, or by automizing one or several activities (Gierhake, 1998; p. 9); (Davenport and Short, 1990; p. 17).

To be able to evaluate and to control processes it is necessary to define performance parameters and to measure the processes with regards to these parameters. For this one often uses the instrument of ratios (or more general metrics) (Dietzsch, 2003; p. 295); (Supply Chain Council, 2006; p. 6). Ratios can be defined for example for production times, costs for executing (sub-)processes, or the number of process instances for parallel execution. Here one has to distinguish actual and target values of these parameters. The norm DIN EN ISO 9000 for example attempts to measure quality by the ratio between the actual and the target value (DIN Deutsches Institut für Normung e.V., 2005; p. 18); (Garvin, 1984; pp. 27).

Process optimization can be seen as part of the process management, which focuses the design of processes, i. e. the planning, regulation, and controlling of processes (Gierhake, 1998; p. 16); (Schmidt, 2002; pp. 3); (Gadatsch, 2005; p. 1). Modern process management not only means sporadic changes of processes, but regular changes as a continuous optimization process (Zollondz, 2006; p. 238). This is not only done to solve actual problems and therefore to identify inefficient processes, but especially as a continuous check of existing processes that identifies unused potential and therefore increases the efficiency of the existing processes (Brutschy, 1993; p. 190). A continuous control of the process and its parameters is done (Neumann et al., 2005; p. 310). Differences between actual and target values are discussed, and ways to explain and repair these differences are evaluated. This may result in initiating a corresponding process change.

2.3 Service-oriented Architectures

The literature knows a lot of varying definitions of the term SOA: “SOA promotes well defined, published, and discoverable interfaces that deliver reusable application functionality for distributed systems.” (Baskerville et al., 2005; p. 5); or “Service based architecture (SOA) is a layered architecture that separates the usage and definition of software components, from the implementation software architecture (…)” (van Zyl, 2002; p. 249); or “A SOA is a software architecture of services, policies, practices and frameworks in which components can be reused and repurposed rapidly in order to achieve shared and new functionality.” (OASIS, 2005; pp. 30). For a good overview see Juhrisch (2010; p. 30). The conceptual development from object orientation to service orientation shows the importance of considering the business requirements for a service, its technical implementation, and its economic relevance.

To highlight the difference of service orientation and object orientation or component orientation, the following list shows some important properties of SOAs.

**Level of abstraction** A service encapsulates functions, whose extent complies with a complete process (step) in a business process (Erl, 2005; pp. 280). A service processes data that corresponds to complete business objects (McGovern et al., 2003; p. 50).
Service development  Service and service consumer are developed independently at different times (Sessions, 2004; p. 45); (Heutschi, 2007; p. 50); Veryard (2003).

Reusability by composition  The focus is on a process oriented composition of service functionality (Krafzig et al., 2004; p. 327 and p. 74); (Zimmermann et al., 2004).

Service design  The service-oriented application development respects the whole organizational information system architecture (Baskerville et al., 2005; pp. 4).

Communication  Services are deployed (in contrast to objects): The are installed and published externally. A service consumer binds a service at runtime (Cervantes and Haller, 2005; p. 10).

Commercialization  Services are IT-solutions that can be sold individually (Bullinger and Schreiner, 2006; pp. 53).

The SOA paradigm can free up a lot of potential for increasing efficiency on many levels of a firm. On the highest level — the business view — the utilization of services and resources and the transfer between different actors can be monitored and controlled internally and externally. This eases that shared use of resources along a disaggregated value chain and allows the reuse of certain resources and services in different scenarios. The ability to share resources whenever it brings benefit is highly influenced by the reusability. This fact promotes the specialization and creation of expert areas in business networks and results in an increasing level of outsourcing (Österle et al., 2001).

The ability to use services as a part of the value chain is made possible on the one hand by a standardization of service descriptions, which allows to determine if a service meets certain business requirements and to discover services. On the other hand, the lose coupling of services allows the reuse of services in different value chains in an interoperable way.

On the process level the interoperability, agility, and reusability of services is accomplished by a seamless process integration (Legner and Vogel, 2008). This level should create a direct link between business processes and the application systems that support these processes. In that case semantic requirements have to be specified, which allows to assess the meaning of service calls. This level holds elementary services — in the sense of business functions — that are executed on demand. In the case of an implementation in software this normally allows to react fast and independently on changes of business requirements. Big and complex application systems are often broken down into a set of services that can easily be reused and reconfigured (Arsanjani, 2005).

On the application layer open internet standards like XML, SOAP, and WSDL support the interoperability of service compositions across different platforms. A SOA therefore leads to a homogenization of distributed information and standard based communication. This results in a highly cost-effective and scalable form of integration of external software and hardware systems, which is known referred to as “software-as-a-service” (Legner and Vogel, 2008). Only by the ability to communicate with open internet standards and therefore the ability to integrate business processes of business partners or to collaborate with different actors along the value chain makes it possible to exploit the full potential on the level of business processes and the value chain. At the same time the efforts for maintenance are reduced by the existence of sub-systems (i.e. services) with high cohesion.

3  THE DESCRIPTION KIT APPROACH

3.1 Integration Conflicts and the Domain Conflict

When comparing conceptual models, several integration conflicts emerge, see Dietz et al. (2011) and Dietz et al. (2010) for a more detailed discussion. The classically discussed integration conflicts are:
**Language conflicts**: Language conflicts affect the labels of model elements to compare. The avoidance or solution of homonyms and synonyms is the first prerequisite for model comparison (Pfeiffer and Gehlert, 2005; pp. 112), (Kamlah and Lorenzen, 1990; pp. 65).

**Structure conflicts**: Each modeler has the freedom to describe his domain at a specific level of abstraction and may choose a certain degree of detail. This leads to so-called structural conflicts (Priemer, 1995; p. 172).

**Type conflicts**: Type conflicts arise from varying choices of an appropriate grammatical concept for modeling (Davies et al., 2002; p. 5).

However, the current literature always assumes that model comparisons are always done without any transition of the described domain. This means that both models are embedded within the same domain and describe essentially (parts of) the same. However, in a real business context this is rarely the case. Domains can be quite different and reflect for example different business areas, different organizational units, different language, different levels of knowledge, or different culture. Domains can also span different organizations or different languages. The viewpoint on a certain problem depends highly on the domain it is embedded in.

The dilemma that arises from the desire to describe different things, but nevertheless with still the need to describe a mapping (or weaker said some kind of relationship) between these models, is subsumed in the domain conflict; it represents the obstacle that prevents an easy model comparison (Juhrisch, 2010); (Dietz et al., 2010).

To resolve the domain conflict the first step is to establish a linguistic community with a consensus about the terminology that should be used in the models. Models in different domains therefore should use a common terminology nevertheless, and the semantic of the terminology must be clear to all sides (all roles that are participating in the modeling process). This involves of course a practicability of the commonly used language for each individual domain.

### 3.2 Solving the Domain Conflict

Since modeling methodologies are normally just concerned with the creation of models and don’t generally focus on the ability to solve functional problems they may have no reference to the problem domain. A model based methodology therefore not only consists of integrated language and procedure descriptions of the model language(s) to use, but also of artefacts of the problem solving techniques and a reference to the problem domain. All this should be subsumed as the intentional aspect of modeling.

To solve the domain conflict, linguistic communities that agree on a common language within their domain are necessary. Since — as previously mentioned as a problem for DSMLs — language develops, an approach is necessary that not only fixes a language before the modelling but remains adaptable during the modelling. A modelling approach that underlies guidelines is the central idea of the Description Kit Approach (DKA).

Guidelines (that result in a constrained form of modeling) include the intentional aspect of the modeling process and cannot be described by classical means of metamodeling (data or process modeling respectively). This intentional aspect interacts with the language and modeling process descriptions. Guidelines also offer the possibility of different ways of using language in different problem domains, without losing the comparability of such created models. The term “problem domain” includes several aspects, all based on the requirement of solving a problem within a certain situation. This is deeply influenced by the operational but also cultural context and certain use cases (actual vs. theoretical comparison, aggregation of domains).
To reach a conjoint understanding of certain data within two (or more) separated domains we influence the modeling process. The use of guidelines helps to control the process of describing certain data either in conceptual or design models. Thus, a common understanding of shared models is forced by following the guidelines. Afterwards, the models can be utilized for the model based problem solution. This approach is formulated generically and already used in different use cases.

We restrict the freedom in modeling within conceptual models in order to limit the language vocabulary to an amount of domain-specific, semantically disjoint language constructs. With this not only the designed conceptual model but also the modeling language has a semantic connection to the application domain (Pfeiffer, 2007). Therefore, from the application point of view semantically meaningful operations in conceptual models can be defined already at a language level.

3.3 Descriptions, Description Kits, and Description Kit Types

As a possible solution for introducing guidelines is the so-called Description Kit Approach (DKA) (Juhrisch and Dietz, 2010), (Dietz et al., 2009), (Juhrisch and Weller, 2008); Description Kits (DescKits) are introduced that cover restricted describable ancillary information in adequately enriched conceptual models. DescKits represent the consensus of the speech community in terms of the amount and structure of certain linguistic concepts relevant for the business analysis. The Description Kit approach is generic enough to restrict every kind of modeling information in their description relating to the present modeling purpose. Concrete descriptions of business information in analysis models concretize the imagination of the modeler at purely linguistic level within the scope of given DescKits.

The DKA has similarities to meta-modeling, but introduces an additional intermediary layer on which the guidelines are introduced. These guidelines act as some kind of “glue” between language creation on the meta-modeling layer $M^1$ and language use (the modeling itself) on modeling layer $M^0$. In the meta-model level at layer 1 the creation of the so-called Description Kit Language (DKL) is done. Here the syntax of every DescKit is determined. This contains the hierarchization of different DescKit concepts (Juhrisch and Weller, 2008) as well as the determination of their usage. The DKL can be kept generic in a way that one or more description languages of this kind are created only once in advance and these are then used in different contexts.

Descriptions (Desc or D) are the actual modeling constructs at level $M^0$. A Description can contain embedded Descriptions, can contain parameters with values and constraints. Each Description uses a certain Description Kit and fills it with life. At this point, the notion of inter-level instantiation differs from the classical understanding of meta-modeling. Instead of the linguistic or object-oriented perspective on the inter-level relationship of an individual element of the model hierarchy (Strahringer, 1998), instantiation within the Description Kit Approach means keeping the constraints for a description given in the corresponding Description Kit. This view of the inter-level relationship is appropriate for exactly three modeling levels.

Description Kits (DescKit or DK) represent a framework for constrained modeling using Descriptions. One DescKit provides the framework for its Descriptions and thus, represents a constraint regarding how real world phenomena can be described in analysis models. At Description Kit level, a new modelling layer $M^{0\ast}$, constraints are created, determining, which Parameters may be included in Descriptions using the corresponding DescKit.

A Description Kit Type (DescKitType or DKT) is a generic concept for Description Kits, which indicates of what type a Description Kit and accompanying Descriptions are. In particular, these types go into the mapping algorithms described below. The Description Kit layer is the counterpart to the meta-modelling layer in classical meta-modelling. The result is a set of relatively generic, but already domain specifically
adapted DKTs. A DKT corresponds to the actual concept behind a constrained modeled facet of the domain.

3.4 The Mapping Algorithm

Using the DK approach an algorithm is introduced that is able to compare process models using a DKL as source models and a design model describing services also using the same DKL as a target model (Dietz et al., 2009). The original idea of this algorithm is to find service candidates for process functions within a process model (see the SOA use case), but it is described completely generic to be useful also in other scenarios.

The algorithm for a comparison of two models or two parts of models (sub-models) is done in several steps:

- First a 1:1-mapping algorithm is introduced that is able to compare two single Descriptions (however, including all embedded Descriptions).
- The a so-called convolution operation is defined, that is able to “fold” a complete model into a single (artificial) description. (This description should include all necessary information of the whole model, and therefore may be quite complex.) This convolution operation makes heavy use of the 1:1-mapping algorithm.
- Then the 1:1-mapping algorithm is invoked again for the convolution results.

The 1:1-mapping algorithm is able to yield already good results, since a Description may embed other Descriptions and therefore may already contain rich information. Because of the generic approach the algorithm is controlled by certain characteristic numbers for each DKT. For details we refer to Dietz et al. (2009).

3.5 Convolution of Process Chains

Also the convolution algorithm is formulated generically. The convolution operates step-by-step on a model and “combines” descriptions along all relations until the model completely falls together into a single description. This means that we need a convolution operation for each relation type. This operation takes two Descriptions (belonging to two maybe different Description Kit Types) and a relation (belonging to a certain relation type) and yields a new Description that combines the two previous Descriptions. How to combine two Descriptions depends on their Description Kit Type and the relation type.

Different scenarios in different domains may have different convolution operations. Defining such convolution operations can be done with help of two predefined operations ∪ and −, which represent two different meanings of “combination”. Both rely on the 1:1-mapping algorithm to detect equality (or similarity) of objects: ∪ represents a union in a natural way, while − is an operation that removes temporary objects (intermediary results of one process activity that are consumed later on by another process activity and not used anymore later on) from the results.

In addition to these two operations, convolution operations may also include mathematical formulas for numerical non-functional requirements. These formulas determine the value of the new “combined” description based on the values of the two descriptions that were convoluted and their mapping quality. Qualitative information may be treated in the same way if it can be treated numerically (e.g. with help of codes like 0=false, 1=true) and the formulas ensure the assignment of a meaningful value.
In the present case the Description Kit Type and Description Kits of Figure 1 are used. The convolution operation works like follows: Two Descriptions using Description Kits of Type “Service” will both contain a list of “Interface”s and “Implementation” information. The convolution combines these two Descriptions to a new Description, where the operation on the “Interface” part looks like follows:

\[
\left\{ \text{Interface} \left\{ \text{Input} \left\{ \text{Object } I_1 \right\} \right\} \left\{ \text{Output} \left\{ \text{Object } O_1 \right\} \right\} \right\}
\]

Here “Process Flow” refers to the relation between both Descriptions and \( \cup \) and \(-\) are the two predefined operations as defined above. This means especially that two redundant service task descriptions are combined into only one common implementation description. In a SOA realization that means that these two tasks can be commonly implemented by one Web service instance and become part of a composition, see Figure 2.

Of high importance for the current article is the convolution operation on the “Implementation” part. It can be described as follows:

\[
\left\{ \text{CBA} \left\{ \text{ImplementationCost } IC_1; \text{SOAIntegrationCost } SC_1; \text{BenefitPerYear } B_1 \right\} \right\}
\]

where

\[
IC = \varphi(\sigma) \cdot (IC_1 + IC_2)
\]

\[
SC = \varphi(\sigma) \cdot (SC_1 + SC_2) + \sigma \cdot SC_{\text{fix}}
\]

\[
B = B_1 + B_2
\]

where \( \sigma \in [0, 1] \) is the matching quality that the 1:1-mapping algorithm yields and \( \phi \) is a function that monotonically falls from \( \varphi(0) = 1 \) to \( \varphi(1) = \frac{1}{2} + \varepsilon \) with a small \( \varepsilon > 0 \). This means that costs just add if the two services do not match at all, but the cost is about the average of the separate costs if the services
Figure 2. Inner-organizational composition and cross-organizational choreography

(nearly) fully match. \( \varepsilon \) here takes a small overhead into account that comes from considering two sets of implementation requirements, even if both implementations are nearly the same. \( IC_{\text{fix}} \) is a constant value indicating fixed costs for multiple SOA integration of a certain (Web) service. The benefit from realizing both services is just the sum of benefits for the single services.

A good function for \( \varphi \) is:

\[
\varphi(\sigma) = 1 - \frac{1 - c}{e^{\alpha} - 1} \left( e^{\alpha \sigma} - 1 \right)
\]

with e.g. \( \alpha = 2, c = \frac{1}{2} + \varepsilon = 0.6 \)

3.6 Generation of Ratios

Using this convolution operation we are now able to convert model data into descriptions that serve as a good foundation to generate ratios. The following can be done in a very generic way again, but for a better understanding we restrict our attention to process models. Since the convolution process considers the model structure and can detect object flows along relations, the resulting description contains a certain object only once, but still stores the information related to what happened to this object. Some details of the object may be used in the beginning phase of a process, others in a latter part, but the convolution result contains both parts. Therefore the convolution can be used to detect or measure coherence of a given process, subprocess, or collection of processes (process group).
A highly coherent process would “collapse” during the convolution. All similar occurrences of (descriptions of) an object will be matched and collected to only one part in the resulting big convolution description. Therefore a measurement of coherence would be to compare the complexity of the original process and the complexity of the convoluted process. This could be used to analyze all business processes in search of service candidates.

First we need a measurement for complexity. The complexity of a (sub)process could best be derived from the objects, which means here in the case of the DKA their descriptions. The DKA with its embedded structures has the big advantage that complexity now can be derived not only by the number of objects, but a “look into” the objects can be done by analyzing the descriptions. Counting different description elements with different weights (depending on their type) yields a measurement for the complexity of a description.

Nevertheless, complexity defined in that way is not an inherent measurement for a process, since it depends on how this process was modeled, especially on the granularity. Here another advantage of the DKA is, that variance in granularity is limited because the inherent guidelines for (a restricted) modeling control this granularity. However, the complexity still remains a vague measurement. We denote the complexity of a (sub)process $P$ by $K(P)$.

Now we can define the cohesion by

$$\Lambda(P) = \frac{K(P)}{K(F(P))}$$

Here $F(P)$ denotes the convolution result of $P$. This formula is exactly what was mentioned above: Compare the complexity of the original process with the folded one. By this construction now a variance in granularity cancels out. A courser description of $P$ would result in higher $K(P)$, but also higher $K(F(P))$, so the coherence is independent of that. We get a measurement of the cohesion of a process that is an inherent value for the process.

The cohesion is a measurement of how well all functionalities that are part of a (sub)process fit together in the sense that they operate on information objects with high connection or context. The convolution operator given above has the property that objects with high context fall together while different objects still remain separated. Furthermore the convolution process is done along the relations, so that not only the objects (including their status) are considered, but also the structure of the process influences this measurement.

In a similar way a ratio that measures the coupling of two processes can be defined. The coupling is a measurement that is in some sense working in the other direction compared to cohesion. Two subprocesses are coupled (have high coupling) if they — at least partly — operate on the same or similar objects. A measurement for this is the value

$$\pi(P_1, P_2) = \frac{K(F(P_1)) + K(F(P_2))}{K(F(P_1 \cup P_2))} - 1$$

(This formula uses the operator $\cup$ as described above.) This value is 1, if the convolution of the union of $P_1$ and $P_2$ has the same complexity as $P_1$ or $P_2$ respectively, which means the case of highest coupling, or 0, if the complexity of the union is just the sum of the complexities of $P_1$ and $P_2$, which means that both subprocesses have nothing in common and therefore are not coupled at all.

Furthermore we simply define now implementation cost, SOA integration cost, and benefit per year for a whole subprocess as the corresponding values for the convoluted description:

$$IC(P) = IC(F(P))$$

$$SC(P) = SC(F(P))$$

$$B(P) = B(F(P))$$
Here \( P \) can be a subprocess, but \( P \) may also include “everything” (within a certain service area), i.e. every service that is considered for possible integration into a SOA. We are then looking for services or service areas with a high benefit-cost ratio

\[
BCR(P) = \frac{B(P)}{IC(P) + SC(P)}.
\]

### 3.7 Evaluation of the Reorganization Potential through Service-oriented Architectures

Important to realize the approach is the availability of cost and benefit data in the sense of non-functional requirements for all services under consideration. In an ideal scenario this data is available in a service catalog together with business process models that describe the services in the catalog. One advantage of the DKA is that models that use the DKA can be assessed electronically in a meaningful way, so that an easy integration of a service catalog with a case-tool is possible. This allows an integrated view on functional and non-functional information about the cataloged services.

The convolution algorithm is the key for generating all the ratios in question and heavily relies on the mapping algorithm. The first step is to use complexity, cohesion, and coupling are ratios that allow to assess the applicability of a family of tasks to be realized as a service. This allows already a technical optimization with regards to composition and orchestration of services to increase reusability and to avoid redundancies. This has been already described in earlier research of the authors (Dietz et al., 2010).

New here is the aspect of a cost-benefit analysis (CBA) that is done at the same time. While the previous analysis has a functional viewpoint, the CBA now adds a non-functional evaluation of the applicability of the realization of certain tasks as a service. Note that this only makes sense when the service has a well-defined functional meaning that meets business requirements. On the other hand, the avoidance of redundancies and the optimization of reusability potential also highly influences the benefit cost ratio. This mutual dependency is a driving factor for doing service identification and a cost-benefit analysis at the same time.

It is important to note that the approach makes most sense if it is not only used to analyse all services in question (in the sense of “SOA yes or no?”), but if it is used for searching for meaningful functional areas where a realization as a SOA is most beneficial. The question therefore is: Use the proposed approach to find highly coherent subgroups of services with a high benefit-cost ratio. Start with realizing these sub-groups as a SOA.

### 4 CONCLUSION

The question if it makes sense to realize a service-oriented architecture is a question that not simply has a yes/no-answer. It may make sense to realize certain functional areas as a SOA while realizing other functional areas otherwise (which may be to leave them as they are, e.g. manual services or as isolated services within an application). Two main viewpoints are essential when assessing the potential of the realization as a SOA: A functional viewpoint that includes the assessment of the applicability of services to be used in service orchestration and choreography, and a non-functional viewpoint that includes the assessment of possible costs and benefit in the case of an automation and realization as a service within a SOA.

The mutual dependency of both viewpoints asks for an assessment of both viewpoints at the same time. The convolution algorithm of the DKA is the key for the proposed approach. The intermediary results of the convolution operation can be seen as a “what-if analysis” in the sense of “what would happen and
how beneficial would it be if we would realize this group of tasks as a service?” for both viewpoints at
the same time. Ratios allow an easy assessment of the applicability.

In this article the non-functional analysis focuses on a cost-benefit analysis. Since the DKA is formulated
generically, it also allows different kinds of analysis. In other articles of the authors a functional analysis
of access and identity management with help of the DKA has been done (Dietz and Juhrisch, 2010),
(Dietz and Juhrisch, 2008). In the same way as here also a non-functional analysis of security aspects
would be highly interesting.

References

tensible architectures: The strategic value of service-oriented architecture in banking. In Proceedings
of the 13th European Conference on Information Systems.
berg.
Seitschek, V., editors, Schlüsselaktor Qualität: Total Quality Management erfolgreich einführen und
Verlag Franz Vahlen, Berlin.
Dahanayake, A., editors, Service-Oriented Software System Engineering: Challenges and Practices,
Davenport, T. H. and Short, J. E. (1990). The new industrial engineering: Information technology and
Davies, I., Green, P., and Rosemann, M. (2002). Facilitating on ontological foundation of information
123–128, Munich, Germany.
In Proceedings of the 14th Pacific Asia Conference on Information Systems (PACIS 2010), Taipeh,
Taiwan.
an approach to enable business driven soa. In Proceedings of the 15th Americas Conference on
Dietz, G., Juhrisch, M., and Leyking, K. (2010). Convolution as the key for service-identification in


Morgan Kaufmann, San Francisco.


