8-6-2011

Inherence of Ratios for Service Identification and Evaluation

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

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

Knut Großmann
Technische Universität Dresden, IWM, groszman@iwm.mw.tu-dresden.de

Inherence of Ratios for Service Identification and Evaluation

Journal:	*17th Americas Conference on Information Systems*
Manuscript ID:	AMCIS-0445-2011.R1
Submission Type:	Paper
Track:	Analysis and Design of Service Systems < Design Science, Analysis and Design for Service-Oriented Enterprises < Systems Analysis and Design (SAND)

Inherence of Ratios for Service Identification and Evaluation

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

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

Knut Großmann
Dresden University of Technology, Dresden
groszman@iwm.mw.tu-dresden.de

ABSTRACT

Service-oriented architectures (SOA) offer a conceptual and technical solution for many problems. However, the realization and even more the management of a SOA often remains a very hard task. A continuous evaluation of existing services and service candidates is necessary to implement, maintain and enhance a SOA. This is far from being trivial because of the semantic gap between requirements and technical implementations. Model-based solutions may offer a possibility to cope with the complexity of the necessary information, but a series of conflicts needs to be resolved especially for distributed modeling projects. These conflicts create several difficulties when trying to generate ratios that can measure the quality of a service or service candidate. This article presents an approach to generate ratios that are inherent to a service (candidate). Inherence means that the ratios are only dependent on the service and not on how the necessary information was modeled.

Keywords

Conceptual Modeling, Service-Oriented Architectures, Service Identification, Convolution, Description-Kit Approach

INTRODUCTION

In the future the trend to automize the development and configuration of application systems will increase more and more. Experts increasingly get aware of the usefulness of models for successful Business Process Reengineering (BPR) projects both in general and in the context of the development of information systems. However, until now only a rough understanding of the specific requirements for a model-based procedure exists. The knowledge about how to handle problems where models have to be compared or integrated is still fuzzy and vague, and there is little agreement regarding compositional facets.

Highly interesting is the interaction between models in shared modeling projects — for example between requirement models and service implementations in a service-oriented architecture (SOA). The increasing use of conceptual models beyond the technical aspects of software and the rising number of model creators as well as the comparability requirement for models or model evaluation, respectively, necessitate a restrictive use of semi-formal conceptual modeling languages. Both the high degree of freedom in modeling and the missing standardization of model elements in semi-formal modeling languages induce a lot of conflicts in model integration. Several attempts are made to solve this problem. However, each of these attempts has also some backsides, so that this problems remains an active research field.

Methodologies that help to solve business problems by using models must describe and integrate three processes: the model creation, the model transformation, and the model usage. Such methods are called model-based methodologies. Model-based methodologies influence both conceptual modeling and how models are used in intra-organizational collaborations.

In this paper a model-based methodology is presented that bases on the so-called Description Kit Approach, which is used to generate ratios that are inherent to services. This means they are mainly independent from the way how these services are modelled, but only depend on the (real) service itself. These ratios are used in service identification and evaluation.

Our research is a matter of design science (Hevner et al., 2004). Following the research method presented in Verschuren and Hartog (2005), the paper is structured as follows. In the next section the theoretical background for the paper is prepared. This includes a discussion of services in general as well as a short presentation of several use cases for the approach. It ends with a

discussion of integration conflicts. In the next chapter the Description Kit Approach, which is the foundation for this article, is presented. This includes the definition of several ratios, that are used in the next section for service identification and evaluation. The paper finishes with a conclusion.

THEORETICAL BACKGROUND

Services

In a functional viewpoint a *service* represents a self-contained piece of functional solution with an implementation in human, mechanical or software resources, with an interface in a standardized form, such that it is reusable in different situations. The functional viewpoint emphasizes the state transitions of business objects (e.g. materials in production or documents in administration), which are caused by a service. The service is characterized by generation certain state transitions with help of "external processing factors" for external needs. External processing factor here means that the invoking the service implies a participation of the recipient of the service.

One or several tasks of business processes are interrelated by a contract about the properties about functionality and implementation of a service (Dietzsch and Goetz, 2005; p. 1522). A service combines a group of *service functions*. A service function is a function of an application system that can perform a certain set of organizational tasks (Turowski, 2003; p. 19). If a service contains only one service function, both terms can be used synonymically. The service is offered via a network (e.g. the internet). The design principles of a service ensure that it is combinable with other services (Szyperski et al., 2002; pp. 192).

A service aggregates the functionality of different components and/or classes. The resulting functionality is considered as a *black-box* (Parnas, 1972; p. 1056). To use the service, no knowledge about implementation and internal data structures is necessary. With the help of metadata for the interface of a service the functionality can be determined, but no details about the implementation (Newcomer, 2004; p. 76). A service consumer passes only data to a service, but no control information (Newcomer, 2004; p. 75).

In contrast to the above, the early literature about SOA had a high focus on Web services and therefore use a technical definition of a service. The main issue here was to overcome the technical heterogeneity of distributed application systems (Erl, 2005; p. 282); (Overhage and Turowski, 2007; pp. 3). These considerations resulted in the definition of the Web service standards due to the recommandations of the *W3C Consortium* (Erl, 2005; pp. 47); (Vossen and Westerkamp, 2007; pp. 290); (Westerkamp, 2006; pp. 77). Often the terms service and Web service are even used synomymically (Szyperski et al., 2002; p. 224); (McGovern et al., 2006; p. 17).

Service Design Criteria

There are two general ways for designing services: top-down — deriving service designs from descriptions of business processes — and bottom-up — deriving service implementations from descriptions of existing application systems. The bottom-up approach is used frequently in practice, since a SOA is often introduced on the basis of existing application systems (Zimmermann et al., 2003; pp. 567); (Arsanjani, 2004b); (Zimmermann et al., 2004); (Erl, 2005; p. 36 and p. 366).

The top-down approach is based on the identification of those processes that can be implemented appropriately as a service within a SOA (Erl, 2005; p. 363); (Winkler, 2007; pp. 258). However, the top-down approach relies on considering design criteria. The main problem when designing services is a deficit of clear guidelines for the service functionality. A decision for a design has to be done considering the business processes that have to be supported by that service, the business objects involved in these process and the importance of business goal to be achieved. Hard design criteria for a systematic identification of service candidates in business process models are indispensable (Arsanjani, 2004a); (Erradi et al., 2006; p. 152). There are of course methods for service identification in business process models (Ivanov and Stähler, 2005; pp. 2); (Aier and Schönherr, 2007; pp. 203), but they have limited applicability when faced with a huge amount of (distributedly created) process models and a high interrelation between them. Services have to designed in a way that they can be used as parts of complex sequence of processing (Richter et al., 2005; p. 414). Services should be reusable in scenarios and domains whose characteristics have not been considered when designing that service. In contrast to design artifacts, as e.g. objects, services should inherit an autonomy with respect to their concrete application context.

SCHWEMM et al. deduce five design principles from literature: *business orientation*, *self-containedness*, *modularity*, *interface orientation* and *interoperability* (Schwemm et al., 2006; pp. 31). Services are business oriented if their functional scope is geared to the required objects. Services are modular and self-contained if resources with high dependency to each other are combined in one service. The design principles interface orientation and interoperability base upon the assumption that services represent stable interfaces that are entirely specified using technical and business metadata (Schwemm et al., 2006; pp. 31).

The design principle of *business orientation* refers to the granularity of a service function. The granularity equates to the scope of functionality that is provided with the service function (Griffel, 1998; p. 43 and p. 48). A service is business oriented if it contains exactly those business objects that are essential to perform a certain business activity (Schwemm et al., 2006; pp. 32). These objects can be modeled and interrelated as information objects using a conceptual data modeling language and then assigned to processes in business process models. *Service candidates* in this case are processes or a sets of processes that perform a common business task and access similar information objects. Hence, the information objects of the processes that constitute a service must show a high *coherence*. A measure of the coherence of a system is the *cohesion* (McCabe, 1976; p. 315). A high degree of cohesion describes a high coherence of the elements of a service, while the contrary implies a low coherence.

To which extend a service is *self-contained* determines its maintainability. Following SIMON, self-contained systems are better to maintain compared to dependent systems since modifications just imply marginal modifications at neighboring systems (Simon, 1962; p. 467); (Wand and Weber, 1990; pp. 1282). SIMON operationalizes self-containedness with help of *coupling* (Simon, 1962; p. 467). Coupling is a measure for the pairwise coherence between several subsystems (Wand and Weber, 1990; pp. 1282). A single process or a set of processes could be identified as a service candidate if this process or this set is independent of other processes. A process is independent of other processes if other processes do first not use its business objects and second the objects transferred to other processes are of little *complexity* (Yourdon, 1979; pp. 108). Accordingly this process could be automated as service without interfering with other processes.

Modularity refers to the question if a service can be used as a part of a more complex task (Richter et al., 2005; pp. 414). By complying with the modularity principle when designing a service, the complexity of the service can be reduced and parallel execution of services can be realized (Baldwin, 2000; pp. 63). Fundamental work on modularization has been done by PARNAS (Parnas, 1971; pp. 339). Closed functional collections are combined and equipped with well-defined interfaces (Balzert, 1998; pp. 571).

USE CASES FOR THE APPROACH

Service-oriented Architectures

Application systems (or software — both terms are often used synonymously) have a certain functionality. The task that an application system can perform have a dependency to the organizational domain, since they support users in business, economy, and administrations in performing their respective tasks (Disterer et al., 2003; p. 21). Therefore, when developing an administration system, organizational requirements have to be respected and create references between functionality and the organizational domain.

The term *system architecture* may refer in a wider sense to application software and hardware together as parts of an application system (Esswein and Weller, 2008; p. 8). A system architecture defines the organization of components of an application systems, their relationships with each other and with their environment, and the principles that determine the design and development of the system (Hasselbring, 2006; p. 48); (Dern, 2006; p. 21).

Service-oriented architectures (SOA) have been established as the main paradigm for a development of an information system architecture that is geared towards business processes and the associated value creation goals (Juhrisch, 2010; p. 30). In contrast to component orientation, service orientation has a focus on a high autonomy of a service and the extend of the functionality that a service offers (Baker and Dobson, 2005; pp. 634); (Matthes, 2005; p. 19).

Identity Management

Identity Management Systems (IdMS) are introduced to bridge the gap between administrative processes, especially human resource management, and processes that are concerned with IT security. Access to resources within a company needs to be controlled. Resources can be Web sites, Web services, network access (like WLAN), file servers, but also locking systems or libraries (where rights to borrow books are nowadays also controlled by IT systems). In the past, each of such systems had it's own user database. IdMS therefore synchronize the user databases on these systems and automatically grant access rights based on the function of a person within the organization (Buecker et al., 2008). The latter is often done by assigning roles to persons to determine their function, together with a set of rules that determine which access rights should be granted to which roles. This is called Role Based Access Control (RBAC) (Ferraiolo and Kuhn, 1992). The use of RBAC raises the following question: How to determine a useful set of roles for this and how to implement the policies and how to assign them correctly to the users?

The answer to that question heavily depends on the organizational structure, the business processes and their implementation

as a SOA. Role management means to answer the question: Who is doing what within the organization and why? The three question words here represent the following: "Who" here means user management, "what" means to have an overview of necessary applications, services and access rights, while "why" reflects the role management. (An accountant has access to accounting information because his role as an accountant reflects the need for corresponding access rights.) After a useful set of roles has been identified, the ordinary process of access management becomes relatively easy with the help of identity management systems.

This leads to two main applications:

- Model-Based Access Control: Use role information in models to (semi-)automatically configure an IdMS. The IdMS will then provision application systems or the rights management of a SOA and automatically grant (and reject) those access rights that are necessary to fulfil certain tasks.
- Model-Based Role Identification: Use the information in models to identify a good set of roles.

A role needs to bundle certain groups of persons and certain access rights. This is similar to the service identification scenario previously described, where also a service may bundle certain tasks.

Hospital Management

Another use case for the presented approach is hospital management. A Hospital Information System (HIS) is a holistic information processing and storing subsystem of a hospital, where this term not only refers to the computer-based parts (Winter et al., 1996). Amongst other things, it includes the Workflow Management System (WfMS) that allows to support the business processes using information technology (Sedlmayr and Rose, 2009). The management of an HIS is a major challenge, especially in hospitals where cost pressure and technological progress demand that HIS have to be implemented in in less time by the available staff.

Conceptual models, in particular process models, are increasingly used in health care for the documentation of medical recommendations and requirements to the process-oriented organization. *Medical guidelines* are created to aggregate medical knowledge, and *clinical pathways* are created to optimize the clinical processes; last but not least their integration into the HIS — particularly the automation — still raises open questions. The models facilitate the communication between the participants in a project, but they are typically not used for further development. To reduce administrative costs, a direct link between the medical guidelines, the clinical pathways and the HIS, which runs the automated part of the clinical pathway, has been implemented with help of the DKA. We refer to Juhrisch et al. (2011) for a detailed description of this use case.

Production Process Management

The demand for short development and market launch times in the producing industry leads to the need for innovative methods for information and knowledge management, since long-term empirical research became unfeasible. With the aim to generate knowledge about new materials and their methods of processing the research project ECEMP uses a model-based approach to describe and analyze technical process chains (Wiemer et al., 2010).

If it is possible to reproduce a high-technological product depends especially on the availability to combine all single subprocesses to a complete technological process chain. Special model-based methods for describing and comprehensively analyzing the technological process chain should support this activity. The modelling of the process chain is object-oriented and based on a structural description of all sections of the process chain, and can be done with help of a graphical modelling tool (Großmann and Wiemer, 2010).

In the model all relevant participating objects are described. This includes ingoing and outgoing materials, manufacturing equipment and all influencing external factors. Process relevant properties and effects are mapped to these objects. In this way the process modelling also creates a database that is used later on for the data collection and the analysis.

For the data collection several tools are used that access the information in the models and write corresponding measurements into the database. Graphical user interfaces to manually entering data as well as a semi-automated collection of data by filters for protocol data as well as sensors are integrated in the solution. The data is then forwarded to analysis tools (e.g. MATLAB/Simulink).

Energy Management

Until now, enterprises need to implement energy management system with nearly no guidance. The DKA is the foundation of a reference model-based method that serves for setting up *Energy Management Systems* (EMS) in various industrial sectors. In

contrast to previous approaches, this approach enables a model-based certification and allows an automation of activities of the process of supervision of the consumption of energy and the appropriate reengineering of business processes.

Model-based methods provide the possibility of a guided implementation, the inclusion of knowledge management and consistency checks. Hence, transaction costs for adaption and improvement can be reduced. Of particular interest is the integration of energy consumption sensors with the management system, i.e. to identify energetically relevant products, processes and to develop adequate energy management objectives.

The DKA can be applied for the following reasons:

- To support the reuse of model content adequate adaptation rules need to be integrated into the reference model.
- The examination of the validity of reference models represents a challenging research task. The compliance of a specific model with the reference model is evaluated individually and subjectively by the model creator. However, the acceptance and thus the validity of the solution is strongly depending whether the user of the reference model could understand the context of the original model and the design decision.

We refer to Schlieter et al. (2010) for details.

INTEGRATION CONFLICTS

When comparing conceptual models, several integration conflicts emerge. They should be shortly discussed here.

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.

Name conflicts occur when different terminologies are used and cause a non-ability of resolving model-crossing references (Pfeiffer and Gehlert, 2005; pp. 112). This language conflict therefore may occur when the creators of the to compared models are not participants of the same linguistic community, or when terms for the names of model elements use technical terms for which the linguistic community has no consensus. Special cases for language conflicts are *homonym conflicts* and *synonym conflicts* (Kamlah and Lorenzen, 1990; pp. 65).

Even when these language conflicts are avoided, a semantic difference can still happen due to different explications of the mental representation of reality. This leads to:

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.

There are several types of structure conflicts: *Abstraction conflicts*, *separation conflicts*, *detail conflicts*, and *dependency conflicts* (Priemer, 1995; p. 172).

Even after all semantic heterogeneity has been avoided (and therefore no language and structure conflicts occur), further conflicts are possible when comparing models. A semantic model comparison also includes the comparison of the syntax (Priemer, 1995; pp. 127). Model elements, types, identifiers of and relationships between model elements have to be compared. This leads to:

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

The choice of a construct of the modeling grammar is dependent on the degree of freedom of the modeling. A solution to this conflict can only be found by restricting the degree of freedom by analyzing the language-based meta-model.

The Domain Conflict

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. One example from software development is the comparison between requirements models (which are often discusses on a managerial level of a company) and implementation models (which are often highly technical). This example also shows that one often has to compare models using semi-formal languages with models using formal languages.

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.

THE DESCRIPTION KIT APPROACH

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 approach in this paper. Guidelines (that result in a constrained form of modeling) include the intentional aspect of the modeling process and can't 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. As the different use cases discussed above show, a generic approach for introducing guidelines is desirable.

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.

Descriptions, Description Kits, and Description Kit Types

As a possible solution for introducing guidelines is the so-called Description Kit Approach (DKA) of the authors (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*}, 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.

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).

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 $\cup$ and $-$, which represent two different meanings of "combination". Both rely on the 1:1-mapping algorithm to detect equality (or similarity) of objects: $\cup$ 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.

See below for examples of the convolution operation for the two use cases service-oriented architectures and identity management.

Generation of Ratios and Inherence

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 (or subprocess).

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. Similarly a measurement for the coupling of processes could be established, which will be done below.

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. An easy approach to define the complexity of a (sub)process would be to add the complexity of all descriptions of a process and to add some value for different types of relations. A better measurement would include an analysis of the underlying graph of the relations: Are there loops, iterations, branches?

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 define the cohesion by

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

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.

The coupling on the other hand 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

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

(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.

With the help of this value one can now define the coupling of a subprocess P compared to the complete process G as

$$\kappa(P) = \left(\sum_{P_j \in G \backslash P} \frac{K\big(F(P)\big) + K\big(F(P_i)\big)}{K\big(F(P \cup P_i)\big)} \right) - |G \backslash P|$$

This is the sum of the previous terms for all process steps P_i in G that are not part of P. However, this analysis can be time-consuming, because the number of submodels of a given model is exponential in the size of the model itself. Therefore it makes

sense to use the described approach in a cascaded way: First fold small parts of the model, then fold some of the previous convolution products and so on. The modeler can support this procedure by pre-selecting some parts of the model.

SERVICE IDENTIFICATION AND EVALUATION

The Description Kit Approach (DKA) described above provides a methodology to establish a consensus on language level which enhances the comparability of models and allows to semi-automate the transformation process by weakening the strict separation of language creation and language usage. In contrast to previous approaches, this approach enables a model based configuration of service-oriented architectures that allows an automation of activities of the SOA development process that are currently carried out manually. As a result, the technical knowledge relevant to cope with the task is reduced, which at the same time shortens the time and effort for solving the overall task.

There are two main scenarios for service identification:

- Services have yet to be defined as bundles of certain process activities. For the different use cases that may mean the definition of a SOA service (Web-service), the definition of access rights, or the identification of comprehensive process chains within a set of process activities.
- A certain service landscape already exists. That may mean that a SOA or an identity management is already implemented. As a part of change management new business requirements may be met by restructuring existing tasks to new services. Requirements that cannot be met have to be identified to and may require new implementations.

In both scenarios this an examination often corresponds to a comparison (matching process) between analysis and design models (Priemer, 1995). Enterprise models must be adjusted to the available services, and later on prepared for a migration into service composition models (e.g. BPEL). A prerequisite is the establishment of a connection between enterprise models and service models.

These two scenarios can be addressed with help of the DKA and its algorithms. The second scenario is addressed by reformulating the problem into a mapping scenario that compares process models (seen as requirement models for the SOA) with service models. Are the services compliant to the business requirements? Which requirements are not met? In the latter case this may mean that an implementation has to be changed or a new implementation is necessary. If a new implementation is necessary then we are in the first scenario. If the implementation has to be changed, then — as described above — it would be desirable to find a new implementation just by composing existing services. To find these services is again a mapping problem that can be solved with the DKA.

For the first scenario we can use the algorithmic part of the DKA that calculates the ratios. As described above, a set of processes within a process model is an ideal service candidate, if the services within the set have high cohesion, high coupling with each other and low coupling with processes not belonging to the set. These ratios were defined above. What remains is to define a convolution operation which behaves well with respect to the introduced ratios.

The same is true in the context of role identification: A good role should have a low complexity with regard to the information that is used to form this role. The information here is threefold. It includes the "two sides" of a role, namely organizational structure and responsibilities on the one side and necessary access rights on the other side, but also includes the "glue" between these two sides, namely the processes that create a link between these two sides. The role information that forms a good role should furthermore have a high cohesion, which means that a lot of similar information is used to define the role. Cohesion is therefore a measurement for how large the business area is, where the role could be used.

To do so, we need to introduce certain DKLs with a certain embedding structure. For the SOA use case for example we define a DKL that reflects the fact that two services are expected to do (nearly) the same when they operate in the same way on input and output. The DKA offers a good foundation to describe the state of input and output objects. For the IdM use case we similarly define a DKT for access descriptions. See Figure 1.

In this scenario the convolution operation can be described as follows:

$$\left(\left\{\text{Interface}\left\{\text{Input}\left\{\text{Object } I_1\right\}\right\}\left\{\text{Output}\left\{\text{Object } O_1\right\}\right\}\right\} \xrightarrow{\text{Process Flow}} \left\{\text{Interface}\left\{\text{Input}\left\{\text{Object } I_2\right\}\right\}\left\{\text{Output}\left\{\text{Object } O_2\right\}\right\}\right\}\right)$$
$$\xmapsto{\text{Conv. Result}} \left\{\text{Interface}\left\{\text{Input}\left\{I_1 \cup (I_2 - O_1)\right\}\right\}\left\{\text{Output}\left\{(O_1 - I_2) \cup O_2\right\}\right\}\right\}$$

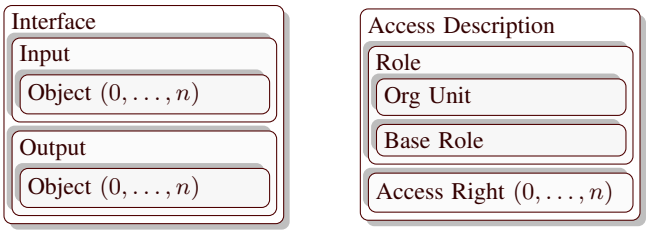

Figure 1. The Description Kit Types for process descriptions for the SOA and the IdM use case

and

$$\left(\left\{\text{Process}\left\{\text{Role}\left\{\text{OrgUnit } O_1, \text{Function } F_1\right\}\right\}\left\{\text{Resource } R_1\right\}\right\} \xrightarrow{\text{Process Flow}} \left\{\text{Process}\left\{\text{Role}\left\{\text{OrgUnit } O_2, \text{Function } F_2\right\}\right\}\left\{\text{Resource } R_2\right\}\right\}\right)$$
$$\xmapsto{\text{Conv. Result}} \left\{\text{Process}\left\{\text{Role}\left\{O_1, F_1\right\} \cup \text{Role}\left\{O_2, F_2\right\}\right\}\left\{\text{Resource } R_1 \cup R_2\right\}\right\}$$

respectively. Here $\cup$ and $-$ are the two predefined operations as defined above.

CONCLUSION

While there is a growing recognition that the adoption of models can enhance the success of Business Process Reengineering (BPR) projects in general, and in the Information Systems development context in particular, there has been limited understanding of the specific needs of model-based engineering. The authors strive for an effective treatment of the conflicts coming from mapping requirements in natural languages to implementations of application systems. This chapter proposes a theoretical model — the Description Kit Approach—not only based on new concepts, but also on a new idea of how the concepts work together toward a guideline-oriented modeling approach.

This article continues some ideas that were created when developing the Description Kit Approach (DKA). Several use cases are presented where the DKA bears fruit. The approach relies on the introduction of guidelines for modeling, which are brought to life with help of Description Kits, and some algorithms that focus on model comparison. Due to limitations in space we refer to broader discussions about the application of the DKA in some of the use cases to Dietz et al. (2009) and Juhrisch and Dietz (2010). Starting with service identification, the idea of the generation of ratios came first when the authors realized that the generic approach of the DKA was bearing fruit in several use cases, which were showing some similarities in evaluating service or process quality.

One main advantage of the DKA is that it overcomes several integration conflicts, especially the domain conflict. The algorithms are designed in a way that the generated ratios are inherent, i.e. do not depend on the way how the information was modelled. The usefulness of the DKA has been discussed in earlier publications, however, the ongoing application of this approach was bringing forward the approach. It demonstrated the usefulness of the approach not only in singular use cases, but as a general approach to model-based methodologies. The research therefore initiates a first step towards the eventual development of "model-based methods theories" that contribute to (IT) business alignment as mediators within and between business and engineering.

REFERENCES

Aier, S. and Schönherr, M. (2007). Modellbasierter entwurf strukturanaloger architekturen auf basis der partitionierung von graphen. In Oberweis, A., Weinhardt, C., Gimpel, H., Koschmider, A., Pankratius, C., and Schnizler, B., editors, *Proceedings of the Wirtschaftsinformatik Tagung 2008: eOrganisation: Service-, Prozess-, Market-Engineering*, pages 199–216, Karlsruhe.

Arsanjani, A. (2004a). Service-oriented modeling and architecture: how to identify, specify, and realize services for your soa. IBM developerWorks Web service zone.

Arsanjani, A. (2004b). Service-oriented modelling and architecture (soma). IBM developerWorks.

Baker, S. and Dobson, S. (2005). Comparing service-oriented and distributed object architectures. In *Proceedings of the International Symposium on Distributed Objects and Applications*, volume 3760 of *LNCS*, pages 631–645. Springer.

Baldwin, C. Y. (2000). *Design Rules, The power of modularity*. The MIT Press, Cambridge.

Balzert, H. (1998). *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag, Heidelberg, Berlin.

Buecker, A., Karl, W., and Perttila, J. (2008). Deployment guide series: IBM Tivoli Identity Manager 5.0. Technical Report December 2008, IBM redbooks.

Davies, I., Green, P., and Rosemann, M. (2002). Facilitating on ontological foundation of information systems with meta models. In *Proceedings of the 13th Australasian Conference on Information Systems (ACIS 2002)*, pages 937–948, Melbourne.

Dern, G. (2006). *Management von IT-Architekturen*. Vieweg, Wiesbaden.

Dietz, G., Juhrisch, M., and Esswein, W. (2009). On the restriction of conceptual modeling - outlining an approach to enable business driven soa. In *Proceedings of the 15th Americas Conference on Information Systems (AMCIS 2009)*, pages 1–14.

Dietz, G., Juhrisch, M., and Leyking, K. (2010). Convolution as the key for service-identification in complex process models. In *Proceedings of the ISD 2010 (to appear)*, Prague, Czech Republic.

Dietzsch, A. and Goetz, T. (2005). Nutzen-orientiertes management einer service-orientierten unternehmensarchitektur. In Ferstl, O. K., Sinz, E. J., Eckert, S., and Isselhorst, T., editors, *Tagungsband der 7. Internationalen Tagung Wirtschaftsinformatik*, pages 1519–1538, Heidelberg. Physika-Verlag.

Disterer, G., Fels, F., and Hausotter, A. (2003). *Taschenbuch der Wirtschaftsinformatik*. Carls Hanser.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall International.

Erradi, A., Anand, S., and Kulkarni, N. (2006). Soaf: An architectural framework for service definition and realization. In *Proceedings of the IEEE International Conference on Services Computing, SCC2006*, pages 151–158, Chicago.

Esswein, W. and Weller, J. (2008). Unternehmensarchitekturen - grundlagen, verwendung und frameworks. *HMD - Praxis der Wirtschaftsinformatik*.

Ferraiolo, D. and Kuhn, R. (1992). Role-based access controls. In *15th National Computer Security Conference*, pages 554–563.

Griffel, F. (1998). *Componentware. Konzepte und Techniken eines Softwareparadigmas*. dPunkt, Heidelberg.

Großmann, K. and Wiemer, H. (2010). Reproduzierbare Fertigung in innovativen Prozessketten. Besonderheiten innovativer Prozessketten und methodische Ansätze für ihre Beschreibung, Analyse und Führung (Teil 1). *ZWF*, 105(10):855–85.

Hasselbring, W. (2006). Software-architektur. *Informatik Spektrum*, 29(1):48–52.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1):75–105.

Ivanov, K. and Stähler, D. (2005). Prozessmanagement als voraussetzung für soa. *Object Spektrum*, page 60.

Juhrisch, M. (2010). *Richtlinien für die modellgetriebene Integration serviceorientierte Architekturen in Analysemodellen*. Phd thesis, Technische Universität Dresden.

Juhrisch, M. and Dietz, G. (2010). Constraints in conceptual modelling — outlining an approach to business driven web service composition. *Int. J. Internet and Enterprise Management*, 6(3):248–265.

Juhrisch, M., Schlieter, H., and Dietz, G. (2011). Supporting planning, assessment and control of application systems — a health care case study. In *Proceedings of the Seventeenth Americas Conference on Information Systems*, Detroit, Michigan.

Juhrisch, M. and Weller, J. (2008). Connecting business and it: A model-driven webservice based approach. In *Proceedings of the 12th Pacific Asia Conference on Information Systems (PACIS)*, pages 1469–1479.

Kamlah, W. and Lorenzen, P. (1990). *Logische Propädeutik: Vorschule des vernünftigen Redens*, volume 2., verb. und erw. Aufl. Bibliographisches Institut, Mannheim et al.

Matthes, F. (2005). The impact of soa on the enterprise application landscape. In *Proceedings of the SOA Days 2005 Business Conference*, Bonn. Deutsche Post World Net.

McCabe, T. J. (1976). A complexity measure. *IEEE Trans. Software Eng.*, 2(4):308–320.

McGovern, J., Sims, O., Jain, A., and Little, M. (2006). *Enterprise service oriented architectures: concepts, challenges, recommendations*. Springer Verlag, Dordrecht, Niederlande.

Newcomer, E. (2004). *Understanding Web Services: XML WSDL, SOAP, and UDDI*. Springer.

Overhage, S. and Turowski, K. (2007). Serviceorientierte architekturen - konzept und methodische herausforderung. In Nissen, V., Petsch, M., and Schorcht, H., editors, *Service-orientierte Architekturen. Chancen und Herausforderungen bei der Flexibilisierung und Integration von Unternehmensprozessen*, pages 3–17. Deutscher Universitätsverlag, Wiesbaden.

Parnas, D. L. (1971). Information distribution aspects of design methodology. *Information Processing*, 71(1):339–344.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communcations of the ACM*, 15(12):1053–1058.

Pfeiffer, D. (2007). Constructing comparable conceptual models with domain specific languages. *Proceedings of the 15th European Conference on Information Systems (ECIS2007)*, pages 876–888.

Pfeiffer, D. and Gehlert, A. (2005). A framework for comparing conceptual models. In Desel, J. and Frank, U., editors,

Enterprise Modelling and Information Systems Architectures: Proceedings of the Workshop in Klagenfurt, number P-75 in Lecture Notes in Informatics, pages 108–122, Bonn. Köllen Druck + Verlag GmbH.

Priemer, J. (1995). *Entscheidungen über die Einsetzbarkeit von Software anhand formaler Modelle*, volume 1. Pro Universate Verlag, Sinzheim.

Richter, J. P., Haller, H., and Schrey, P. (2005). Serviceorientierte architektur. *Informatik-Spektrum*, 28(5):413–416. 0170-6012.

Schlieter, H., Juhrisch, M., and Niggemann, S. (2010). The challenge of energy management — status-quo and perspectives for reference models. In *Proceedings of the 14th Pacific Asia Conference on Information Systems (PACIS 2010)*.

Schwemm, J. W., Heutschi, R., Vogel, T., Wende, K., and Legner, C. (2006). Serviceorientierte architekturen: Einordnung im business engineering. Working Paper of the HSG, Universität St. Gallen.

Sedlmayr, M. and Rose, T. (2009). Unterstützung medizinischer leitlinien - von der zielorientierten modellierung zur proaktiven assistenz. In *Proceeding of 9th Internationale Tagung Wirtschaftsinformatik*, pages 739–758, Wien. vieanna:ocg.

Simon, H. A. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467. 0003-049X.

Strahringer, S. (1998). Ein sprachbasierter metamodellbegriff und seine verallgemeinerung durch das konzept des metaisierungsprinzips. In *Proceedings of the Modellierung 1998, Astronomical Society of Australia*.

Szyperski, C., Gruntz, D., and Murer, S. (2002). *Component Software - Beyond Object-Oriented Programming*, volume 2. Aufl. Addision-Wesley and ACM Press.

Turowski, K. (2003). *Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme*. Shaker Verlag, Aachen.

Verschuren, P. and Hartog, R. (2005). Evaluation in design-oriented research. *Quality and Quantity*, 39(6):733–762.

Vossen, G. and Westerkamp, P. (2007). Service-oriented provisioning of learning objects. In Koohang, A. and Harman, K., editors, *Learning Objects: Applications, Implications, & Future Research*, pages 287–324. Informing Science Press.

Wand, Y. and Weber, R. (1990). An ontological model of an information system. *IEEE Trans. Software Eng.*, 16(11):1282–1292.

Westerkamp, P. (2006). *Flexible Elearning Platforms: A Service-Oriented Approach*. Logos-Verlag.

Wiemer, H., Wagenführ, A., Pfriem, A., Siegel, C., Großmann, K., Helbig, M., Fischer, S., Bremer, M., Gohrbandt, A., Feldner, A., Hufenbach, W., Kupfer, R., Neinhuis, C., and Horbens, M. (2010). Characterising the application potential of biological compounds and using model-based methods for the tech- nological transmission into composite materials. In Hufenbach, W., editor, *Proceedings of the International Colloqium of the Cluster of Excellence ECEMP 2010*, pages 219–240, Dresden. ECEMP — European Centre for Emerging Materials and Processes, Technische Universität Dresden.

Winkler, V. (2007). Identifikation und gestaltung von services: Vorgehen und beispielhafte anwendung im finanzdienstleistungsbereich. *Wirtschaftsinformatik*, 49(4):257–266.

Winter, A., Zimerling, R., Bott, O., Hasselbring, W., Haux, R., Heinrich, A., and Jaeger, R. (1996). Das management von krankenhausinformationssystemen: Eine begriffsdefinition. In *Tagungsband zur 41. GMDS-Jahrestagung*, pages 34–38, Bonn. MMV Medizin Verlag.

Yourdon, E. (1979). *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice Hall, Upper Saddle River.

Zimmermann, O., Krogdahl, P., and Gee, C. (2004). Elements of service-oriented analysis and design. IBM developerWorks.

Zimmermann, O., Tomlinson, M., and Peuser, S. (2003). *Perspectives on web services: applying SOAP, WSDL, and UDDI to real-world projects*. Springer, Berlin.

Project Part-Financed by the European Union

European Regional Development Fund