July 2009

MANAGEMENT OF IT-ENABLED BUSINESS PROCESSES – USING FRAMEWORKS FOR CONSTRAINED MODELING

Martin Juhrisch
Technische Universität Dresden, juhrisch@uni-muenster.de

Gunnar Dietz
Multimedia Kontor Hamburg gGmbH, g.dietz@mmkh.de

Werner Esswein
Technische Universität Dresden, werner.esswein@tu-dresden.de

MANAGEMENT OF IT-ENABLED BUSINESS PROCESSES – USING FRAMEWORKS FOR CONSTRAINED MODELING

Martin Juhrisch
University of Technology Dresden
Chair of Business Informatics, esp. Systems Engineering
Dresden 01062, Germany
martin.juhrisch@tu-dresden.de

Gunnar Dietz
Multimedia Kontor Hamburg GmbH
Project eCampus
Hamburg 22081, Germany
g.dietz@mmkh.de

Werner Esswein
Professor of Business Informatics
University of Technology Dresden
Chair of Business Informatics, esp. Systems Engineering
Dresden 01062, Germany
werner.esswein@tu-dresden.de

ABSTRACT

A model-driven approach to semi-automatic manage IT based business processes implies to interlink language concepts of the business and IT domain. The paradigm of service-oriented architecture provides a technological basis for reacting to business requirements in a distributed application environment. Services encapsulate functionality to be reused in different processes and can be easily described in design models that should be the result of analyzing the business requirements described in conceptual models. However, this process is nontrivial, iterative, and cannot be fully automated. This paper presents an approach to introduce artifacts 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.

Keywords: Enterprise models, model transformation, Web services, SOA.

INTRODUCTION

The intention of the Service-oriented architecture (SOA) paradigm is a reorientation of the organization away from a primarily technology-focused IT use towards one that is geared to business processes and the associated value creation goals. In an organization there exists a variety of interrelated services and their underlying processes. On the other hand, there are a number of Web services offering the potential for reorganization of this process landscape. These include Web services from document management, workflow management or archiving systems, etc. The complexity of the process landscape in conjunction with the multitude of available services confronts decision makers with the problem of finding adequate areas of application for services in their process landscape. To benefit from existing enterprise models and thus to increase the use of models generally more efficient, they should not only be used for the identification of reorganziation potential and process improvements, but also to structure and configurate the service-oriented architecture.

The test whether certain business processes can be implemented through service compositions is the central concern of this paper. A test is being a comparison between analysis and design models. Organizational models must be adjusted with the available services, and then be prepared for the

translation into service compositions. The prerequisite is the establishment of a connection between analysis models and distributed services.

The aim of this work is the use of business models for the configuration of information systems in general and in particular for service-oriented architectures. Unlike previous transformation approaches, especially findings related to the semantic model comparison and model migration will be integrated into the method construction. The increased efficiency is mainly achieved through automation of all automatable tasks in the transformation process. As a result, algorithms based on a set of rules perform the model operations.

The paper is structured as follows: After this introduction the research methodology is introduced that should be used in this paper. We then describe the goal of this paper and the described approach. From this we derive requirements for achieving this goal. Based on these requirements this paper then presents a solution in the form of the Description Kit Language and a mapping algorithm, which then should be used in the original motivation: The configuration of a service-oriented architecture. The paper ends with a discussion and the future research.

1 RESEARCH METHODOLOGY

To enhance the traceability of the results and to verify their validity, an own research design will be outlined (Becker et al. 2004; Schütte 1998). To systematize the own epistemological position Braun and Esswein suggested an approach that should be followed here (Braun et al. 2006). Therefore at first the epistemological point of view will be fixed first and after that the scope of gaining knowledge will be established. The own epistemological position is therefore determined by disclosing the ontological and epistemological point of view as well as by distinguishing an understanding of truth used to verify the results.

In the present paper a constructivist position will be used regarding the subject-object relation, that is based on an open ontological point of view (Becker et al. 2004). Closely associated with this is an own understanding of truth. The consensus theory as a adequation theory of truth should be followed here (Becker et al. 2004). A statement is regarded as true if and only if in optimal and ideal conditions everybody can accept it rationally.

The goal of the present paper is the creation of a method for administrate and configure service-oriented architectures. From this research goal a mission of IS research could be deduced. So the mission of the present paper is only in part a functional mission and should focus on gaining knowledge about the usability of the method in question for university business processes.

Considering the method to develop this paper primarily has a design approach. The goal for gaining knowledge should be a theoretical understanding of how the configuration of a SOA on the basis of organizational requirement models could be done.

The present paper follows the constructivist paradigm of design science. Knowledge will be gained by creating and evaluating artifacts in the form of models, methods or systems. In contrast to empirical research the goal is not necessarily go evaluate the validity of research results with respect to their truth, but to the usefulness of the built artifacts as a tool to solve certain problems (Hevner 2004).

The creation of the theory, i.e. the necessary hypothesizes for creating the method will be done deductively. Starting with existing theory fragments (general statements) own hypothesizes for the concrete method to be developed will be explored, assembled and integrated. There is no demand for the creation of an own theory and therefore we forebear from using the concept of a theory. However, in this case the decisions for the design concept should be disclosed (decision management). In contrast to an approach driven by theory the basis for the design not necessarily have to be formulated as hypothesizes.

To conclude, the outline for gaining knowledge in this present paper should be the goal of developing a method for administrate and configure service-oriented architectures in a model-driven way. For this requirements will be imposed using the literature or hypothesizes will be created. The method will be constructed, implemented and tested in a real environment.

2 DESIGN RATIONALE

In view of the compositional system engineering with Web services, a strictly sequential run through analysis in the design phase is to be questioned. Instead, to design a system as a whole, several services are assembled process-oriented during design phase (Turowski 2001). In contrast to traditional object-orientation, services have a certain autonomy compared to the given application context (Szyperski 1998). Hence, they are independent and can be interchangeably reused within different service compositions. This has consequences on the phases of the system engineering process, as during the analysis phase requirements models can be influenced by existing services.

Thus, main focus of the requirement analysis modeling lies on the question of coupling of services in business terms (Turowski 2002). Generally speaking, a transformation is therefore understood as the transfer of requirement models in service compositions. If the requirement model can be covered by the available service domain, then the model is migrated into a BPEL diagram. At last, the abstract BPEL diagram must be transformed into a concrete executable service composition and thus correspond to a separate service. The modeling problem is precisely defined in line with an investigation of the internal view of the transformation. Vital clues offer the source and target states of the artifacts in the transformation process.

2.1 Source Model

As already mentioned, in the case of an underlying service-oriented system engineering, we may assume that business process modeling languages act as natural description techniques for functional requirement models. Business models are in their role as a description of the problem, input in the following transformation process. If this process ought to be automated, this makes high demands with regard to content and linguistic quality on the form of business process models. Formalization is the basic prerequisite for automating the transformation process (Krämer 1988). Content quality requirements can be drawn from Krämer's demand for freedom of interpretation in question.

If the truth or falsity of a formal expression ought to be determined regardless of the reference to their interpretation, so this implies the completeness and correctness of conceptual models. Both claims are nonsatisfiable to meet by analysis models (Juhrisch et al. 2008).

An automated analysis of conceptual models fails in its lack of suitability for a clear interpretation by an automaton. As requirements are always extracted from either organization models or the problem domain itself, they are always collected from a material domain and thus are always considered as artifacts that are needed to be interpreted.

Furthermore, it can be inferred from the requirements for quality of conceptual models that the consensus finding holds a high priority in the modeling process (Schütte et al. 1998). Techniques propagandizing the use of formal languages for modeling business requirement elide the principle of language adequacy of the Guidelines of Modeling (GoM), which are guiding principles to evaluate a modeling language in terms of its suitability for a modeling purpose (Schütte et al. 1998). If languages that have been created exclusively for the design of software solutions are used to model organizational requirements, it can lead to semantic misrepresentation of the analysis models.

Beyond, non-functional requirements have to be considered due to their high influence on the software design. Non-functional requirements in contrast to functional requirements cannot be assigned to a specific form. Their description is mostly informal, based on text, lists, or graphics (Partsch 1998). Moreover the range from individual functions of the application system to the complete system in all aspects of the application system and the development may be denoted in addition. Thus, their characteristics runs contrary to a formal description, which also can be seen that neither object-oriented nor structure-oriented modeling approaches allow an explicit modeling of non-functional requirements (Partsch 1998). In order that there is no form that grants neither a meaningful quantification nor operationalization.

2.2 Goal Model

Services play the role of an object to be analyzed or of a system (part) to be integrated. Abstracting from the specific viewpoint of conceptional modeling and from the specific type of the application systems, the goal should not be to transfer analysis models to design models meaning the creation of design models for the implementation of new services. Such a transfer would automatically establish a connection since the design models are the result of the analysis models. A mapping on the other hand relies on the existence of design models or a service catalog with service descriptions. For a mapping from analysis models one should use service models to find service candidates for implementation on the on hand, or to find (existing) services which satisfy the needs for (sub-)processes of the analysis models.

A model mapping then can be understood as a mapping of solutions in the form of existing design artifacts to problems in question. The target of the transformation represents a set of alternative solutions for the problem in question. Analysis model's duty will be to structure the problem description for human decision makers in software design. For a mapping the developer will have to be able to interpret the requirement models, so he has to be part of the speech community that has to be created during the analysis phase. A normalization of the language used between software developers and experts could nearly eradicate language conflicts (see also below) and therefore the target system be narrowed.

The developer checks the capability of a specific service to be used for an existing problem and uses it. For that he has to use an internal representation of the problem description for which the service in question is a solution. The comparison between the actual problem and the specific problem statement which could be answered by the service finally results in a decision for or against that artifact. If the model of the service doesn't use a material semantic – which normally is the case for design models – this comparison only can be done by a human authority.

An automated selection of services for a specific problem and the configuration of these services to build service compositions needs expert knowledge for the material semantic of a service. Only the description using a technical language could be used for determining the capability of a service for some functional problem. A mapping from the requirements model however is based on understanding the material semantic. The developer of course can externalize has internal representation of the problem description, which the service solves, as a model.

This model then could be used as a specification of the service since it describes the functional context of the service with the help of a conceptional modeling language. To deduce the specific functional problem directly from the service, it must be enriched be material semantic. An arbitrary description of the design models however will be lost when trying to formalize them.

To reduce the arbitrariness of the material semantic of the language, language conflicts have to been resolved. We suppose that the comparison problem could be (at least) partly automated when commonly using certain language concepts during the analysis phase and the design phase. This only can be done when getting a consensus for certain concepts within the language community for domain experts and software developers. Since the material semantic can't be completely formalized one can only normalize the language for semantic descriptions for achieving the goal of a semiautomated comparison.

A mapping of requirement models and service models requires to compare the material semantic of both conceptional models. For this one needs to establish a semantic comparability, which means to resolve the semantic heterogeneity of technical language constructs. The complete transformation process ends with a migration of the handled model into BPEL.

3 REQUIREMENTS ANALYSIS

On the one hand requirements to a method with the goal to compare models as described will be determined. They are determined by the certain goal of the model comparison. Furthermore there are

requirements as a result of formal objectives to the model transformation, especially regarding quality criteria for analysis models and cost efficiency of a method.

The method should be able to semantically compare conceptual models within different domains. This means to resolve language conflicts and structure conflicts as well as to resolve the domain conflict. Quality requirements for models of the analysis phase have to be kept.

R1: It will be evaluated how the method in question is able to distinguish between model concepts for describing real world phenomena and model concepts that have a relevancy for application development within the source model. That means restrictions to the normal modeling process for domain experts should be avoided in any case. The reason for R1 is that the descriptional instruments really should contribute to an understanding of the models and that the gap between generally described usage requirements and precise technical specifications can be closed.

R2: Resolving the domain conflict by the possibility of using artifacts from different domains commonly resp. the possibility of comparing these artifacts at least semiautomated. Therefore a procedure model or an algorithm should be able to compare certain object descriptions within the requirement model and a given object description within the service model.

The result should be a decision of the quality of the compatibility of a given service function to a certain process function, a sub-process or a whole process. As described, the best way to resolve the domain conflict is to focus on the handled objects and their states.

R3: Resolving language conflicts: The method should resolve the semantic heterogeneity of specialized language constructs. Semantically disjoint and domain specific language constructs should be placed at the modeler's disposal.

To do this one needs to create a language community which persons that want to model functional and persons that want to model technical take an active part in. To resolve language defects one needs to establish a consensus for a terminology during the process of language determination. This terminology has to be included into the language based metamodel for the modeling language. Therefore the method has to include the process of establishing this consensus.

R4: User control for the model transformation. The software developer and the domain expert should have the possibility to have a widespread control of the model transformation to BPEL. Even if the method can resolve language conflicts and structure conflicts, the domain conflict can't be avoided completely. The reason for this is that the material semantic could not completely deduced from its syntactic form. There are no conditions that allow to map service functions to analysis models in a unique way. Due to this fuzzyness the domain expert as well as the software developer should be able to control the mapping to BPEL. This controlling includes tasks that could not be fully automated by the method, e.g. some parameterizations in the beginning or some decisions between transformation alternatives during the process of the transformation.

R5: A congruent explication of real world phenomena by a model creation process that uses conventions in connection to semantically disjoint language constructs. These conventions should enforce a consistent and standardized usage of the remaining freedom of the modeling language to improve the degree of automation (or to make it possible at all).

R6: Taking non-functional requirements into account. The model comparison method should consider non-functional requirements to services when mapping these to processes resp. process functions.

R7: Definition of a model migration. The method should allow a mapping between exactly two modeling grammars. In this scenario the source model should have some arbitrary (but determined) grammar and the destination model should be BPEL. To do this one needs to define a procedure model and an algorithm, how to create a destination model from the source model and mapping policies.

All these requirements can be deduced from the formal goals of the method. Requirements that could be deduced from formal goals of some method specify the design of some new method or the evaluation of an existing one (Gehlert 2007). So these requirements address the whole process. A

method is demanded, which should be carried out by using minimal resources. This implies an automation of all tasks that could be automated. Furthermore – adopting the concept of truth used in this article – the model transformation is done correctly exactly when there is a consensus of the product. So it is required that the transformation process is comprehensible. This transparency should be accomplished by a detailed documentation of the method.

4 MODELING APPROACH

In the present paper 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 Description Kit approach centers the phase of the language generation (see Fig. 1).

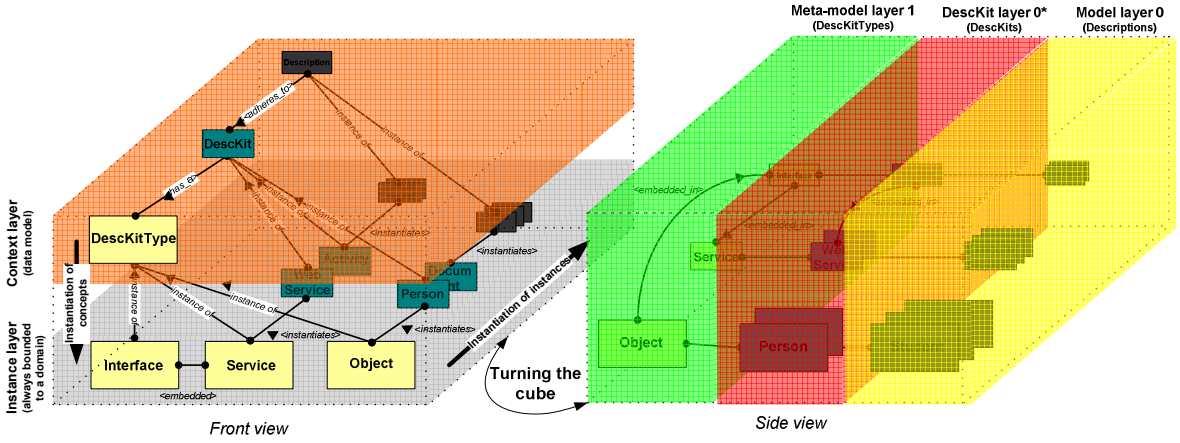

Figure 1: Layers of the Description Kit approach.

In the meta-model level at layer 1 (left illustration) the creation of the so-called Description Kit Language (DKL) follows. Here the syntax of every DescKit is determined. This contains the hierarchization of different DescKit concepts (Juhrisch et al. 2008) as well as the determination of their usage. The latter require a linkage of the meta-model of a conceptual modeling language – already existing by the time – to the model of the DKL. It is determined which DescKit concepts are possible or obligatory to which elements of the model. So the DKL is associated with the meta-model of the goal language.

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. The ideal case is that there exists a DKL that is generic in a way that each modeling information dependant on the existing modeling purpose can be modeled as a restricted domain-specific language construct.

After the disclosure of design rationales follows the definition of the concepts used in method fragments. The generic restriction of the modeling freedom on conceptual level is based on the idea of standardization of modeling (modeling of constraints) at 0 * and a constrained business modeling at level 0.

4.1 Language based meta-model

Regarding the language based meta-model we differentiate between the following concepts (see Fig. 2). The path from the upper to the lowest level in the data model is always meant in the sense of concretization / formulation in accordance with the previously formulated frameworks / conceptual constraints. All (…)-Type concepts are elements for the formulation of these constraints and the

actual modeling takes place within the DescKits (here everything runs together). Later on, a Description then uses a DescKit.

The data model consists of various squares that constitute commutative diagrams. In a commutative diagram various concatenations of concepts provide the same result and thus, both paths are compatible. Hence, the diagram commutates if the relationship between the concepts is preserved no matter what path is selected.

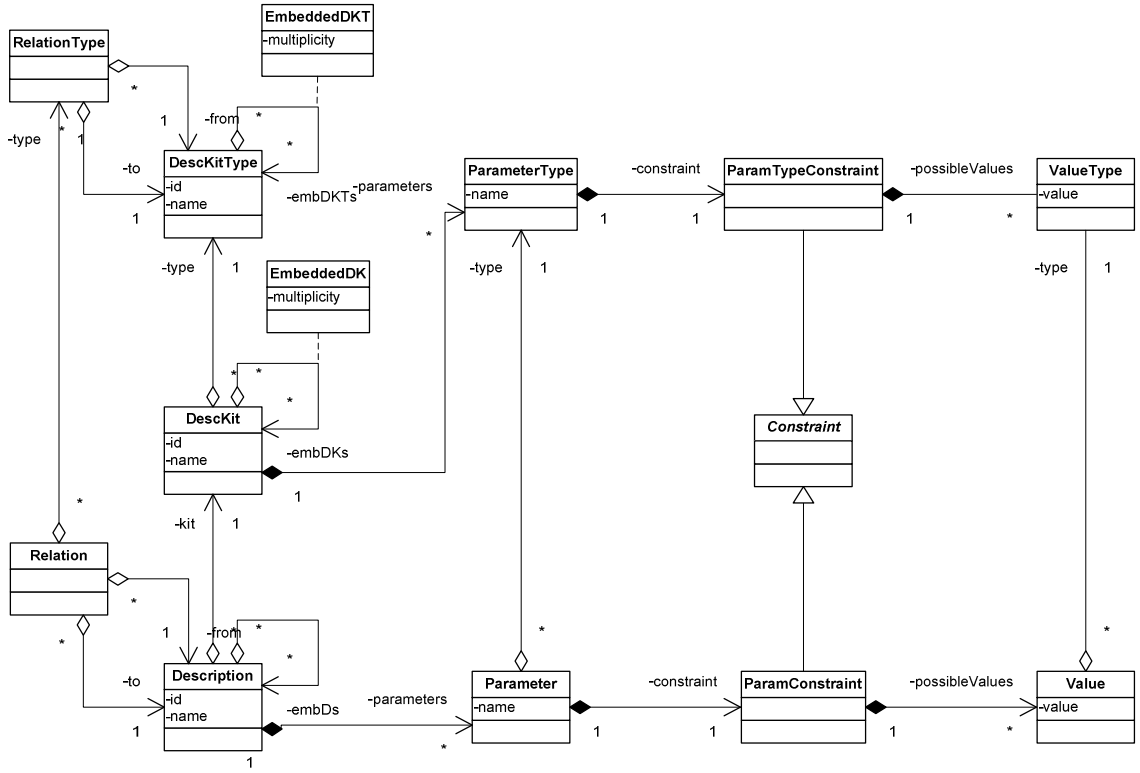

Figure 2: Data model of the Description Kit Language (context layer)

Description

Descriptions are the actual modeling constructs at level 0. A Description can contain embedded Descriptions (Relation embDs), can contain Parameters with Values or constrained Values (ParamConstraints). 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.

The ‘instantiation' of a Description Kit provides a concrete description (Description) at model level. A description does not necessarily describe an entire concrete real world phenomenon, but may also specify only a few aspects of a phenomenon. Basically, everything is describable in line with the modeled constraints.

Contrary to the understanding of the shallow instantiation the influence of a concept at level 1 is not solely restricted to its direct instances at level 0*, but extends to the level 0. The Description Kit approach also offers the possibility of constructing a business model exclusively with Descriptions. In this case of a pure DK-language, Descriptions are inter-related. The concept of Relation describes the relationship between the Descriptions at level 0. At level 0 Descriptions possess a structural form conform to the Description Kits used. They also have the ability to build relationships with one another, provided that this provision was made at level 1. Its Parameters include a specific Parameter

Type, and thus are subject to conditions, defined at level 0*. A condition explicates itself for example in the amount of possible Parameter Value assignments of a Parameter Value Type (ValueType).

Description Kit (DescKit)

Description Kits (DescKit) 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 (e. g. objects, processes, etc.) – in the sense of restricting the degree of modeling freedom. A Description Kit is modeled at level 0*, which occupies a meta-role with respect to the object level 0. As already stated about the inter-level instantiations, a DescKit is not a concept of a meta-model, e. g. in terms of an object type. In fact, a DescKit is ultimately a model for a Description, not a language. The total set of DescKits and their relations can be regarded as model of a language. This does not apply to an individual DescKit.

It is indicated how Descriptions (instances of certain DescKits) may be nested together (EmbeddedDK). The multiplicities are thereby constraints, which have to be complied with at the level of Descriptions. Likewise, at Description Kit level constraints are created, determining, which Parameters may be included in Descriptions using the corresponding DescKit. The reference to ParameterType implicates that the constraint modeling is abstracted also here. A Description Kit is a collection of all Kits and Parameters, which could be used for a specific instance of the Description Kit.

To sum up, a Description Kit is an ordered collection of conceptual properties and value assignments, arranged according to aspects, which concern the descriptive concept or a certain instance of the Description. In the present case, the DKs document the consensus of the speech community regarding the quantity of specific language concepts, which are relevant for the business analysis, and the way of their description possibility. The consensus is reached as part of a consulting service concurrently by methods developers and the clients (business and technical staff).

Description Kit Type (DescKitType)

A Description Kit Type (DescKitType) 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. At level 1, constraints are made for embeddings (EmbeddedDKT) that have to be adhered to at the level of Description Kits – only just been specified for the different DescKits of DescKitTypes. Likewise, for a pure modeling language composed of Descriptions constraints are made for their relations (RelationType). This concept describes possible relationships between corresponding Descriptions.

Beside the syntactic definition of a Description Kit and the constraints (presettings) for relationships between Descriptions of that DescKitType, a DescKitType additionally references to the concepts of the utilized conceptual modeling language. These include the assignment of DescKitType instances to elements of the conceptual modeling language and the determination whether Description Kits or Attributes are obligatorily or optionally to be integrated into the conceptual model. Thus, a forced usage of certain DescKits for certain modeling elements can additionally not only mean a restriction of the freedom in modeling in terms of the DescKits itself but also in terms of the modeling. Hence, the meta-model based method joins the procedure of language creation and language usage. Due to the necessary linkage between DescKitTypes and the concepts of a given conceptual modeling language, applying a meta-modeling approach like the E3 approach (Greiffenberg 2004) appears being useful.

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. Most likely it represents an aspect of the business domain, which is fed to a generic restriction of its descriptiveness at level 0*. The rigid frame of DKTs offers the advantage to be subject of a relatively small need for changes compared to the Description Kits.

Parameter

Parameters fill a Description with life and form (with their value constraints) the core of the descriptions of real world phenomena. Hence, filling Descriptions with Parameters means bringing them into being. Each Parameter has a certain Parameter Type, which is like a DKT the actual concept behind a concrete Parameter. Even if parameters play a role only at the level of the DKs and below, the actual ParameterType concept tends to belong to the same level as DKTs, except that the use of this concept takes place one level further to actually model the constraints for Descriptions.

Constraints

Parameters include values - but not necessarily a fixed value, but a constraint for values. The data model indicates the simplified case that the potential range of values for a parameter is specified. Hence, a ParamConstraint is a value constraint; here a collection of values (Values) (relation possibleValues). This concept can (and should) be extended to more complex constraints – something almost prepared by the concept of Constraint. Each ParamConstraint is of a certain type – here in the simplest case, this might be a value type (String, Integer, default values like from-to etc.). At the upper level, at a more abstract level, again the constraining frameworks are modeled for the Descriptions: Each Parameter is an instance (as defined above) of a ParameterType and each value instance of a ValueType. Even at this level conditions are formulated for possible values (ParamTypeConstraint), which have to be adhered to at the "concrete" level of modeling. Also on this level conditions for possible values are formulated (ParamTypeConstraint), which are to be kept on „the concrete “level.

4.2 The Mapping Algorithm

Using the DK approach an algorithm should be 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. The original idea of this algorithm is to find service candidates for process functions within a process model, but it is described completely generic to be useful also in other scenarios.

Because of the generic approach the algorithm is controlled by certain characteristic numbers for each DKT. This will be described in detail below.

Since Ds (and DKs, DKTs) can be embedded into each other and every D (DK) can have parameters, values and constraints, the structures to be compared are quite complex and the algorithm has to take into consideration the embedding location of each artifact to compare. To simplify the data model the algorithm operates on, we map all parameters and its values of Ds to so-called virtual Ds, which have the DKT “virt_DescKit” and are instances of DKs “virt_Param”, “virt_Value”, resp. “virt_Constr”. This leads to a model transformation which results in a model using only Ds (of certain DKs and DKTs) without parameters, values or constraints.

As a result we get for each side (source and destination of the comparison) the following data:

For the left hand side a set $\{D_\alpha\}_{\alpha \square A}$ of descriptions with the following notations:

D_α : A Description

DK_α : The Description Kit which D_α is an instance of

DKT_α : The Description Kit Type which DK_α is an instance of

$id(D_\alpha)$: ID of the Description D_α

$id(DK_\alpha)$: ID of the Description Kit which D_α is an instance of

$id(DKT_\alpha)$: ID of the Description Kit Type of DK_α

$p(D_\alpha)$: The Description that contains D_α (parent)

$\lambda_1(DKT_\alpha),\ldots,\lambda_k(DKT_\alpha)$: The characteristic numbers for the DKT of Dα

For the right hand side we get a set $\{D_\beta\}_{\beta \square B}$ with corresponding notation. Arriving there we have done the first two steps shown in Figure 3, and the main part of the algorithm can begin.

In the next step we move through the sets of description kits step by step to compare them individually. This means we loop over $2^{|A|}$ and $2^{|B|}$ (i.e. $\alpha \square A$ and $\beta \square B$) and determine how good D_α and D_β correspond:

i. Check, if $id(DKT_\alpha)=id(DKT_\beta)$; if not, break

ii. Check, if $id(DK_\alpha)=id(DK_\beta)$; if not, break

iii. Check, if $id(D_\alpha)=id(D_\beta)$; if not, break

iv. Take the embedding position into consideration: Do the same checks i.-iv. for all parents, i.e. recursively compare D_α and $p(D_\beta)$, $p(D_\alpha)$ and D_β, as well as $p(D_\alpha)$ und $p(D_\beta)$

The results of this check should be evaluated and weighted by the characteristic numbers for the DKL, to get as a result a number $\sigma(D_\alpha,D_\beta) \square [0,1]$, which reflects how good the two descriptions correspond. For this step the characteristic numbers control some scenarios: How important e.g. is the fact that some embedded description is missing on the right hand depending on where it is embedded on the left hand.

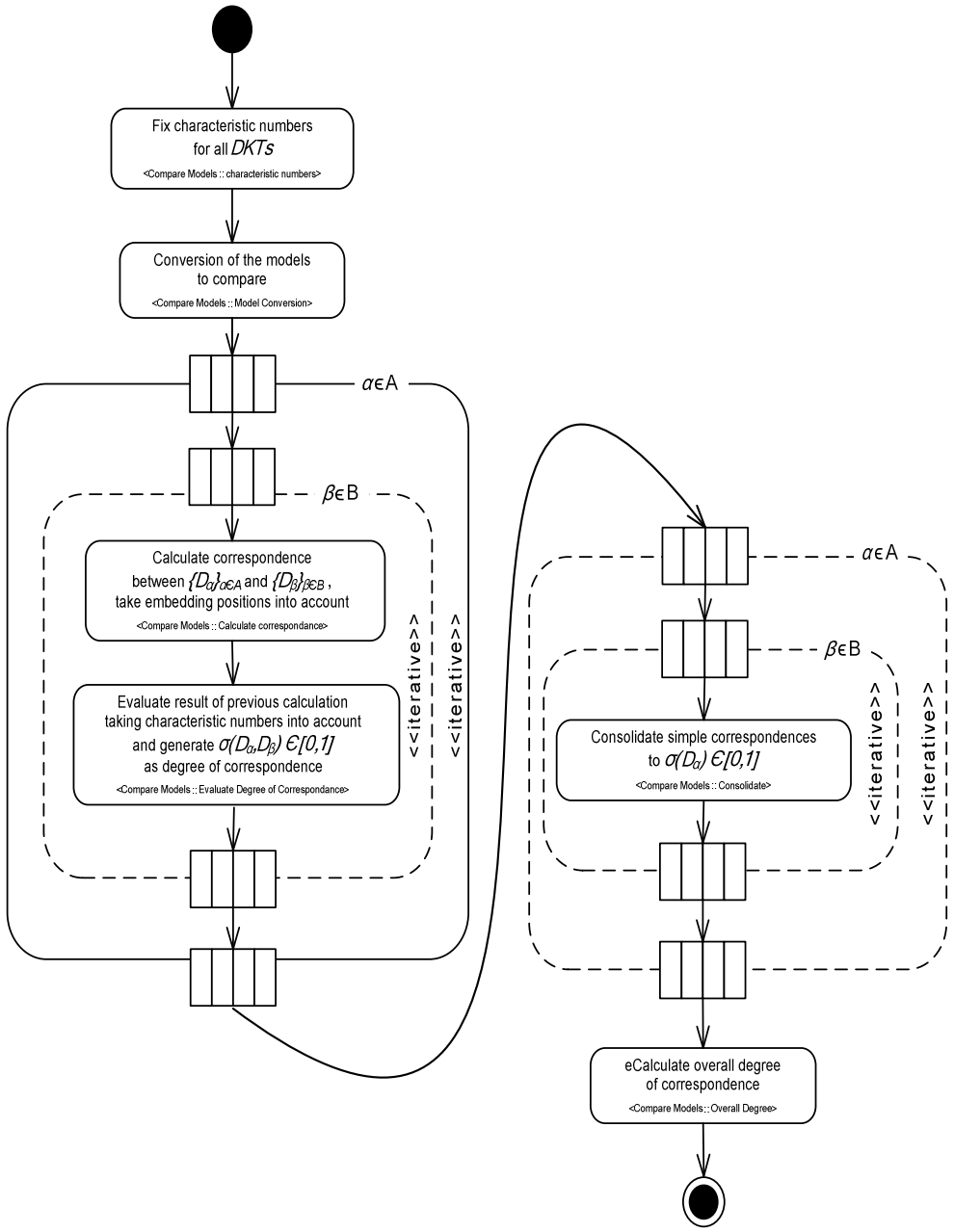

Figure 3: Procedure model for the mapping algorithm

After calculating these numbers the results are to be consolidated to numbers

$\sigma(D_\alpha) \in [0,1]$

which reflects how good a description D_α is matched on the right hand altogether. This step is also controlled by the characteristic numbers which control in which situations it is good or bad e.g. to have multiple matches (or no matches). (See next chapter for an example.) In the last step this numbers will be consolidated again to get a number $\sigma \in [0,1]$ which describes the degree how good the complete model data $\{D_\alpha\}_{\alpha \in A}$ and $\{D_\beta\}_{\beta \in B}$ correspond. Since parameters, values and constraints have been mapped to virtual descriptions (with own characteristic numbers), they are of course part of all these calculations.

5 USE CASE: SERVICE-ORIENTED ARCHITECTURES

The mapping algorithm described in the previous chapter will be used for the use case of configuring service oriented architectures. To do this a Description Kit Language has been modeled primarily focusing on interfaces. These DescKitType and their derived DescKit and Descriptions are then used to enrich EPC (event driven process chain) models as requirements models and to describe existing or planned services in a service catalog or in design models.

Central part is a DescKitType for processes and services that embeds a DescKitType for interface containing input and output as sets of a DescKitType for objects. Using these DKTs (especially the DescKitType "object") then certain DescKits can be created in an adapted way to a certain use case or domain. E.g. the use case of a SOA in a university context (as the original motivation of this approach) would use other DescKits than in other scenarios. In the models (and/or service catalogs) these DescKits are then used to add descriptions. See Figure 4 for an example of the usage of a DescKit "application" used in the enrolment process for a student.

By this, each activity in a process model and each service (function) is described by an interface of ingoing and outgoing objects and their concrete meanings (which DescKit is used?) and states (embedded Ds, Parameters, Values and Constraints). The mapping algorithm then can be used to operate on this data to evaluate service candidates for a given process function or sub-process. Changing requirements then would alter the analysis models, and the algorithm would check for changes needed on the SOA side, or vice versa. This semi-automated would result (with humans help for accepting or rejecting or adjusting the algorithm's suggestion and after migrating) would be BPEL models that really reflect the requirements.

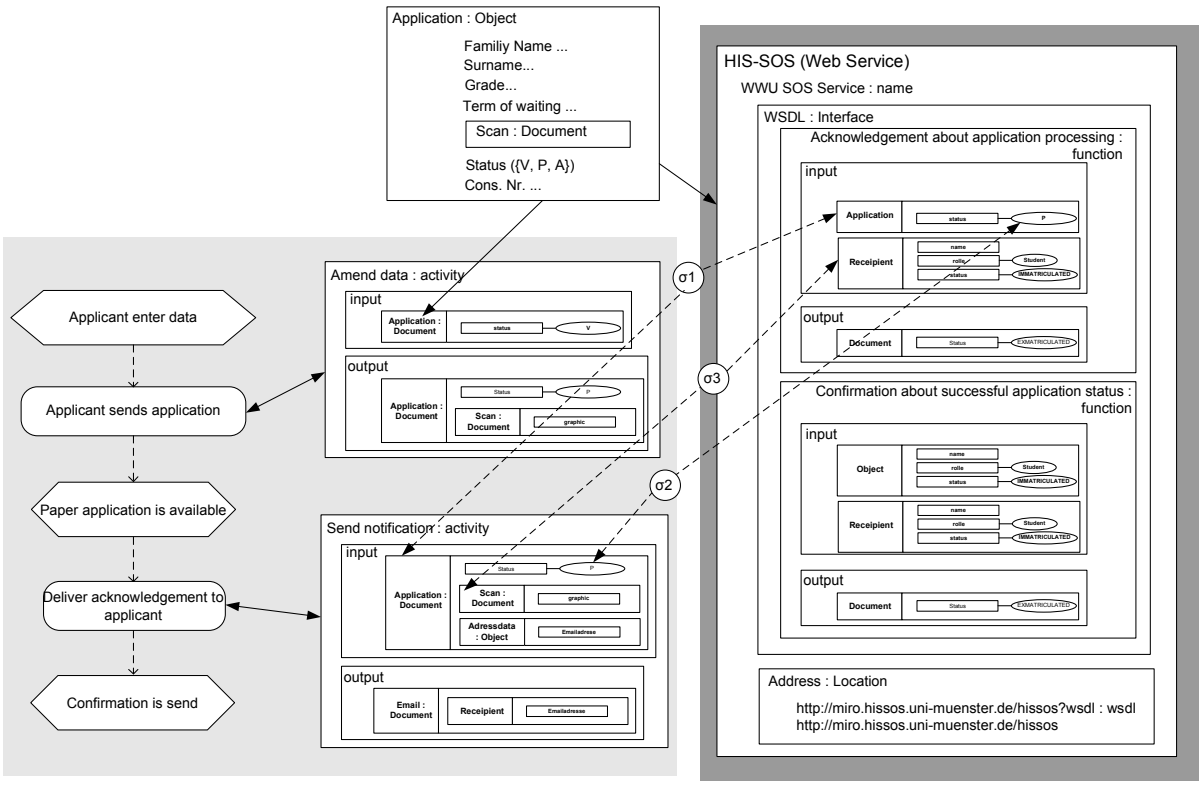

Figure 4: Using the mapping algorithm for a process model on the left side and a (catalog of) services on the right hand

6 DISCUSSION AND FUTHER RESEARCH

Future work lies on the comprehensive evaluation in a pilot study conducted at the University of Münster. Going productive with cubetto® Toolset in the near future and involving decentral IT-support organizations of the university after that, the acceptance of the method will become apparent. The consistent usage of the modeling tool at the university will help to use it as a medium of choice for documenting and managing of SOA and to consider it as a part of the integrated information management.

REFERENCES

Becker, J., Niehaves, B., and Knackstedt, R. "Bezugsrahmen zur epistemologischen Positionierung der Referenzmodellierung," in: *Referenzmodellierung: Grundlagen, Techniken und domänenbezogene Anwendung*, J. Becker and P. Delfmann (eds.), Physica-Verlag, Heidelberg, 2004, pp. 1-17.

Braun, R., and Esswein, W. "Eine Methode zur Konzeption von Forschungsdesigns in der konzeptuellen Modellierungsforschung," *Integration, Informationslogistik und Architektur: Proceedings der DW2006*, Köllen Druck+Verlag GmbH, 2006, pp. 143-171.

Gehlert, A. *Migration fachkonzeptueller Modelle.*Logos Berlin, Berlin, 2007, p. 414.

Greiffenberg, S. *Methodenentwicklung in Wirtschaft und Verwaltung.* Dr. Kovac, 2004.

Hevner, A.R. "Design Science in Information Systems Research," *MIS quarterly* (28:1) 2004, p 75.

Juhrisch, M., and Weller, J. "Connecting Business and IT – A Model-driven Web service based Approach," *Proceedings of the 12th Pacific Asia Conference on Information Systems*, Suzhou, 2008.

Krämer, S. *Symbolische Maschinen. Die Idee der Formalisierung in geschichtlichem Abriß*, Darmstadt, 1988.

MDA "Guide Version 1.0.1," in: Document Number: omg/2003-06-01, 2003.

Ouyang, C., van der Aalst, W.M.P., Dumas, M., and Ter Hofstede, A.H.M. *"Translating BPMN to BPEL."*

Partsch, H. *Requirements-Engineering systematisch: Modellbildung Für Softwaregestützte Systeme*, 1998.

Schütte, R. *Grundsätze ordnungsgemäßer Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle.* Betriebswirtschaftlicher Verlag Dr. Th. Gabler GmbH, Wiesbaden, 1998.

Schütte, R., and Rotthowe, T. *"The guidelines of modeling: An approach to enhance the quality in information models",* Lecture Notes in Computer Science (1507) 1998, pp 240-254.

Strahringer, S. *"Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips",* Proceedings of the Modellierung 1998, Astronomical Society of Australia, 1998.

Szyperski, C. *Component Software: Beyond Object-Oriented Programming*, (2 ed.) Addison-Wesley, Harlow, 1998.

Turowski, K. *"Fachkomponenten - Komponentenbasierte betriebliche Anwendungssysteme,"* University of Magdeburg, Magdeburg, 2001.

Turowski, K. *"Vereinheitlichte Spezifikation von Fachkomponenten",* Gesellschaft für Informatik (GI), Augsburg.