Michael Becker and Stephan Klingner

Linking Process Models and Service Configuration

This paper presents a holistic approach for modelling and configuring services by integrating the process model view. First, existing process models can be reused by transformation into service models. Second, customer-individual configurations of service models can be exported into according processes. This is beneficial because it allows for validation and performance evaluation of customer-individual configurations.

1 Introduction

Due to the growing economic relevance of services, several challenges regarding service development and management emerge. One of them is the increased market pressure that needs to be addressed in different ways (Carlborg et al. 2013). An often applied approach is to use *standardisation* to increase service efficiency and provide services at lower prices than competitors. Another suitable approach is *individualisation* where services are customised according to customer requirements (Sundbo 2002).

The two goals standardisation and individualisation form a dichotomy that can be dissolved by applying the principles of mass customisation (Hart 1995; Pine 1999). According to the mass customisation paradigm, services are first divided into standardised, logical and functional independent components. During configuration, the components are combined according to customer requirements. In doing so, it is possible to create customer-individual service configurations. At the same time, scalability and manageability are preserved (Heiskala et al. 2005). The emerging challenge is the evaluation of configurations regarding validity and productivity. Complexity becomes manageable due to the fact that individual service components are independent of each other (Böhmann et al. 2003). Thus, factors influencing validity and productivity can be discussed on the level of components.

To enable efficient mass customisation of services, a modelling approach specialised on the describing and configuring complex services is necessary. In contrast, due to historic reasons, companies today possess a large number of business process models to describe and communicate their service processes. Using a process representation for complex services has various severe limitations regarding representation of a service portfolio and also regarding configuration abilities.

The work presented in the paper at hand tries to overcome the conflict between the demand for specific approaches tailored to service description and the usage of existing process models. Therefore, a three-step approach for enabling service description and configuration based on business process models is presented. The first step is to use existing process models as basis for a service description by transforming them into service models (*import*). This is followed by the customer-individual configuration of the service model (*configuration*). As a last step, the configured service is converted to a customised process model so it

can be used in existing process model engines (*export*).

The remainder of this paper is structured as follows. Sect. 2 presents the research design and the theoretical background of this work in terms of service modularisation and the respective service metamodel for describing services. It further motivates the work by presenting additional shortcomings concerning process model based service configuration. The following Sect. 3 describes the configuration approach and outlines the three steps import, configuration, and export. After presenting the implementation of corresponding software tools and evaluating the approach in Sect. 4, the paper is concluded in Sect. 5 with a discussion of limitations and future research directions.

2 Theoretical underpinnings

The work presented in this paper is based on existing preparatory work about modelling and configuring services. In this context a metamodel was developed and implemented in different software tools (Böttcher and Klingner 2011; Klingner and Becker 2012). The metamodel was established as part of the research project *KoProServ*[1] and evaluated by different partners from industry (Klingner et al. 2011). The software tools can be used by service providers offering the possibility to individualise services according to customer requirements.

In this section we give a brief overview about the fundamental research design for developing the service metamodel and the process model extension. This is followed by a discussion of necessary concepts for service configuration. To close the presentation of the theoretical underpinnings, we show how the service metamodel that lays the foundation for this work implements these configuration concepts.

[1] http://koproserv.uni-leipzig.de

2.1 Research design

The work presented here is the result of a design science research according to (Hevner et al. 2004). Thus, the following phases were passed:

Problem identification
The used service metamodel is based on concrete demand from practice for approaches concerning service modelling and customer-individual configuration of services. The metamodel was refined by experts from practice in several workshops as part of the *KoProServ*-project. In another stream of research, we were dealing with large amounts of processes in companies (Becker and Laue 2012). The idea to use existing process models for service configuration is a combination of both research streams and can be justified by shortcomings concerning configuration of process models (Rosa 2009). Having in mind that acceptance of new technology largely depends on perceived ease of use (Davis 1989), it is necessary to provide approaches for seamless integration of new technology in terms of specified service description methods.

The identified challenges led to selecting a design science approach; on the one hand, the proposed solution addresses unsolved problems, on the other hand, it tackles solved problems more effectively. First, we present a more effective way for service configuration based on existing process models. Second, we provide a solution for the rather unsolved problem on how to (automatically) transform these process models into a service representation so they can be enriched with service-specific attributes.

Requirements specification
As elucidated in the problem identification, one major requirement was to establish both efficient and effective methods for applying the service metamodel and allowing companies to establish customer-individual service configurations. In addition, it is necessary to

integrate the service metamodel approach into existing infrastructures of companies, i.e., reuse existing process models and enable using existing process execution environments.

As mentioned above, requirements were gathered in several workshops concerning service configuration with experts from practice. While the import mainly focuses on supporting the acceptance of a new technology, the export functionality is mainly driven by the necessity to increase efficiency of the service configuration and to reduce time-to-market of configured services. Therefore, it allows for the transformation of service models into process models that can be executed with the help of process engines.

Design
The design of the solution was driven by two concerns. First, to foster companies in service configuration based on existing process models, we established a catalogue of process patterns (cf. Sect. 3) that are used for transformation into the service metamodel representation. These patterns are intended to cover a broad range of models from practice. Second, the approach was divided into three rather independent steps. Thus, the configuration of services is independent of the existing infrastructure of a company.

Evaluation
Practical evaluation of the research is still an ongoing process. However, we can present first results of a preliminary evaluation in this work. So far, different software tools were developed to show the practical applicability of the approach. In addition, an argumentative evaluation referring to (Frank 2006) was conducted (cf. Sect. 4).

Regarding the results of our research, we have established the following artefacts:

Constructs
A major result of the overall research is the service metamodel that is used for configuring services according to customer requirements. The model is defined using formal logics and also contains a graphical representation.

Methods
The focus of the paper at hand is on presenting a method for implementing the service metamodel in companies by reusing existing process models. Therefore, the three-step approach as presented in Sect. 3 was developed. The method contains several transformation rules to allow for maximum coverage of possible process situations.

Instantiations
To foster using the constructs and methods, we have developed a tool chain supporting every step in the service configuration life cycle. The majority of these software tools were developed prototypically and tested on real-world examples as presented in Sect. 4.

2.2 Service configuration

The approach for configuring complex services used as a foundation in this work is one example of the research about service configuration. In recent years, a variety of different approaches based on different fundamental ideas were presented. A more detailed overview about service configuration and a classification of these approaches can be found in Becker et al. (2013a).

A major requirement for enabling customer-individual service configuration is modular service design (Harmsel 2012; Liu and Xu 2009). In compliance with Bask et al. (2010), we define a modular service as “a system built of components”. Modular services allow for replacing components and increase management possibilities. Therefore, it is necessary to describe the overall structure, the functions of components, and the relations between these components (Bask et al. 2010). Cao et al. (2006) state that flexible business processes, flexible organisational structures, and flexible

enterprise resources are prerequisites for service configuration. A modular service architecture consisting of small, functionally independent components supports dissolving the dichotomy between standardised services (to increase efficiency) and customer-individual configurations (to increase customer satisfaction).

Three aspects are fundamental for modular service design: service modules, service architecture, and service experience (Tuunanen et al. 2012). While the service module aspect focuses on decomposing a service offer into independent service components, the service architecture aspect emphasises the relations between components of a complex service with each other and with elements of the service environment. Finally, service experience deals with customer requirements that can be met based on customised service offers.

Using a modular service architecture, it is possible to establish customer-individual configurations of complex services. (Lampel and Mintzberg 1996) identified various configuration strategies with respect to the degree of customer involvement. The first approach that does not allow for any customer influence on design, production, and distribution of a service is called *pure standardisation*. A company supporting *segmented standardisation* uses a basic design to cover different service variants. Customers can select the variant that best matches their requirements. With *customised standardisation*, customer-individual requirements are met by a service configuration based on standardised components. *Tailored customisation* allows for even greater customer influence on service design by providing a prototype that is fit to their requirements. Using no predefined service components at all, *pure customisation* is characterised by the greatest degree of customer involvement by designing a completely individualised service offer.

2.3 Process models and service configuration

Today, companies use a large number of business process models to describe and communicate their service processes. Large scale enterprises often own process repositories consisting of hundreds or even thousands of models (Dumas et al. 2009). Usually, these repositories are managed using different techniques like intelligent process repositories (Yan et al. 2009).

However, several disadvantages exist when process models are used to offer customisable services. First and foremost, using processes there is no central representation of a company's service portfolio. This might lead to the situation that several variants of the same business process exist in a company (Rosa et al. 2013). Though different approaches for finding and eliminating duplicate or similar services exist, redundancy often occurs (Weber et al. 2011).

In particular when it comes to the configuration of customer individual services, representing services solely as process models has tremendous drawbacks, since existing process modelling notations like Event-driven Process Chains (EPC) or Business Process Model and Notation (BPMN) often lack configuration abilities (Rosemann and van der Aalst 2007). This is caused by the fact that most of the existing process modelling languages do not allow for a precise and clear description of process variability. According to Milani et al. (2012), it is possible to distinguish between *decision points* and *variations points* of process models. While the first ones are decisions that are made during process execution, the latter ones can already be made during configuration, i.e., before a process is actually executed.

Refer to Fig. 1 for an example. The presented BPMN model consists of four activities that are nested in a particular exclusive choice

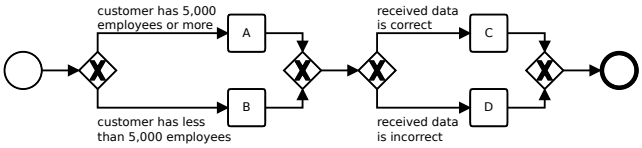


Figure 1: Process model with decision and variation points

block. During process execution, only one of the activities A and B and only one of the activities C and D can be executed depending on the respective condition. As can be seen from the description of the conditions, the first decision can clearly made before the process is even executed, since it is known beforehand whether the customer has less than 5,000 employees or not. Therefore, the first decision represents a variation point. On the other hand, it can only be decided during process runtime whether received data is valid or not. Therefore, the second decision represents a decision point.

To overcome the limitations of process models regarding customer-individual configuration of services, it seems reasonable to use modelling approaches specialised on the description and configuration of complex services. In particular, a clear distinction between decision and variation points is necessary for meaningful and effective configurations.

2.4 Modelling using the Service Metamodel

The service metamodel supports both *segmented standardisation* and *customised standardisation* according to the criteria established by Lampel and Mintzberg (1996). For segmented standardisation, service providers can define prebundled variants using service components. Though customers do not have direct influence on the variants, they can choose the variant best matching their specific requirements. The potential of the configuration approach is fully exploited when used for customised standardisation. In doing so, customers select their required components and, thus, can directly influence the characteristics of the service during configuration.

The configuration can be performed either *top-down* or *bottom-up*. During top-down configuration, customers begin with selecting high-level components and refine them accordingly. Contrary, applying bottom-up configuration, customers select very fine-grained components matching their specific requirements. Using formal dependencies between components, the configuration can be automatically completed for ensuring validity. Both, top-down and bottom-up configuration, result in valid services allowing for comparing performance impacts of different configuration outcomes.

Using the service metamodel it is possible to formally specify the structure of complex services. The aim of the metamodel is to provide software tools for supporting service modelling and configuration. In the following, core concepts of the metamodel necessary for understanding the transformation steps are presented. The interested reader is referred to the literature for a detailed description of the metamodel (Becker and Klingner 2012, 2013; Böttcher and Klingner 2011).

To establish smaller subsystems, the service metamodel allows for hierarchically structuring service components into *super components* and *subcomponents* using so-called *connectors*. Each connector is characterised by a set of *cardinalities* to quantitatively specify valid compositions during service configuration, i.e., a connector restricts the valid number of subcomponents.

Since interrelations in complex service portfolios are usually not limited to hierarchic dependencies, it is necessary to define additional configuration constraints using *logical dependencies*. Using these constraints, it is possible to define cross-hierarchical dependencies between service components. With

hierarchical and logical dependencies, it is possible to define validity constraints of configurations.

For service provision, temporal aspects need to be taken into account, too. The metamodel allows for defining these aspects using *temporal dependencies*. Without temporal dependencies, it is assumed that every service component can be provided in parallel. Using temporal dependencies, it is possible to define sequential constraints. A choice of different connector types and dependency rules is presented in Tab. 1.

The usage of the metamodel elements is depicted in the example portfolio presented in Fig. 2. The portfolio consists of a root component called *Order Processing*. Though in this example, this component is the topmost component, it can be used as a subcomponent in the overall portfolio of a company. The root component is divided into two subcomponents *Laboratory results* and *Postprocessing* connected by a K_{ALL}-connector, i.e., both components are mandatory. The semantics of the other connectors is shown in more detail in Fig. 2. In addition to the hierarchical dependencies, the example portfolio also contains one temporal dependency specifying that components *Laboratory result* needs to be executed before component *Postprocessing* can be executed.

3 Approach

In this section, the three steps of the holistic approach are presented individually. The whole lifecycle from importing existing business processes, modelling and configuring services to exporting customer-individual process models is shown. Using this approach, existing as well as generated process models are only loosely coupled, i.e., the connection between these models is established by means of import and export. This is justified by the aim to integrate the metamodel approach into existing IT infrastructures with lowest possible expenditure. The three steps are depicted in Fig. 3: *import* existing process models, *adapt and configure* service models, and *export* process models.

There are different ways to integrate the presented approach for service modelling and configuration into an existing corporate environment. First of all, it is possible to create service models according to the metamodel completely from scratch. However, this could result in unnecessary overhead, since often services are already described by existing process models, either in terms of reference models or by using company specific models (Dijkman et al. 2011). Thus, it seems natural to reuse these models via import.

The next step is to adapt and configure service models. Since process models usually do not hold all information required for service configuration, it is necessary to enrich generated models. Besides productivity-related factors this might also include logical dependencies between service components. In doing so, structural and financial effects of customer-individual configurations are made transparent. Based on an exhaustive service description, it is possible to configure a service according to customer requirements. The formal definition of the metamodel allows for validating the configuration.

The configuration according to customer requirements is followed by the provision of the service. The customer-individual service model can be used as a formal basis for service provision. Additionally, it might be feasible to generate IT services or process models that support (semi-)automated service provision. By using generated models, heterogeneities and variations in quality can be reduced - this is fostered by exporting customer-individual service configurations into executable process models.

Table 1: Choice of connector types and logical and temporal dependencies

Connectors	
K_{ALL}	All of the succeeding nodes must be selected during configuration.
K_{ONE}	Exactly one of the succeeding nodes must be selected during configuration.
K_{ANY}	An arbitrary amount of succeeding nodes can be selected during configuration. However, at least one nodes must be selected.
$K_{(n,m)}$	At least n and at most m succeeding nodes are allowed to be selected during configuration.
Logical Dependencies	
requires(A)=B	*Requirement Rule*: If component A is selected during a configuration, component B needs to be selected, too.
prohibits(A)=B	*Prohibition Rule*: If component A is selected during a configuration, component B must not be selected.
Temporal Dependencies	
before(A)=B	*Precedence Rule*: If components A and B are selected during a configuration, component B must be executed before component A.
iBefore(A)=B	*Direct Precedence Rule*: If components A and B are selected during a configuration, component B must be executed directly before component A, i.e., no other component must be executed between B and A.
repeatable(A)	*Repeatable Rule*: Component A can be executed multiple times.

3.1 Use Case

A use case from practice shall illustrate the transformation steps (import, configuration, export). On the one hand, this allows for a first exemplary evaluation of the approach. On the other hand, the transformation based on the use case can be used to illustrate open research questions and challenges arising in practice. The use case is deduced from a research project in cooperation with a precision farming company. Aim of this project was to develop a system enabling a more efficient processing of orders. Process models constitute the foundation of this system and allow for a structured description of tasks. Furthermore, the processes of the system can – to some extend – be used for automating various tasks. The use case consists of two process models regarding the processing of an order and is depicted in Fig. 4 and Fig. 5.

The first process model (cf. 4) of the use case represents the overall order processing. It is possible to distinguish between two fundamental order types: soil sampling and land surveying. Based on the specific order type, different activities need to be executed. This variance is represented in the highlighted areas of the process model in Fig. 4. Both exclusive gateways in the highlighted areas are variation points, i.e., a decision is possible before process execution. Thus, only a specific part of the process model is relevant regarding the different order types. The third exclusive gateway is a decision point, i.e., a distinction between correct and incorrect received data can only be made during process execution.

The second process model of the use case (cf. Fig. 5) considers the postprocessing part. Again, the highlighted area represents a variation point, since it is known beforehand whether the customer requested a printout, an

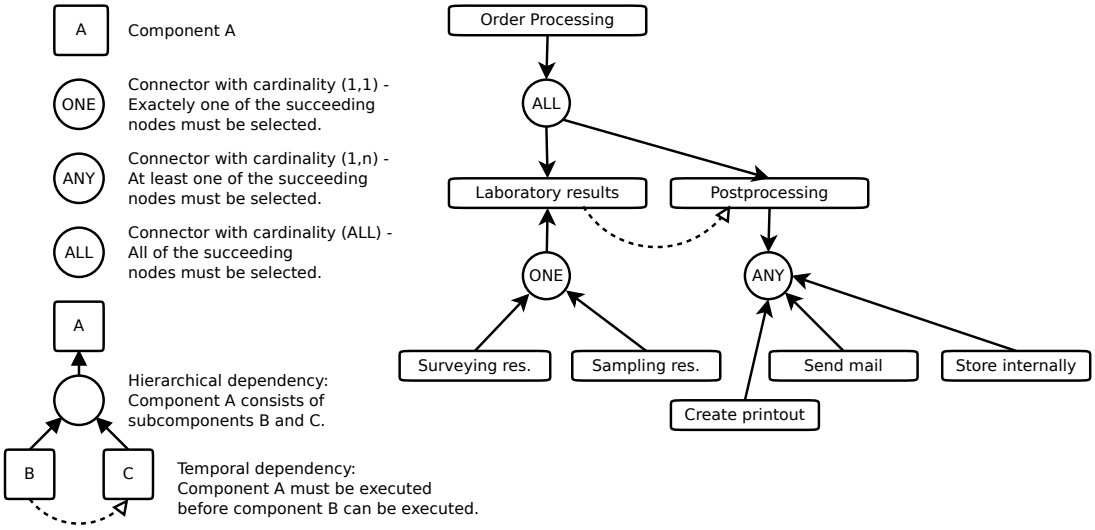


Figure 2: Example portfolio with different modelling elements

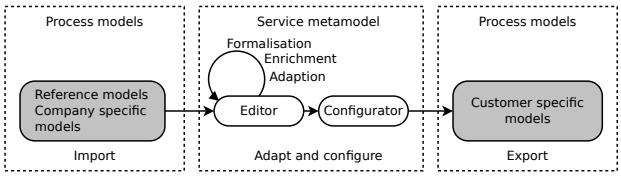


Figure 3: Three-step approach for service modelling

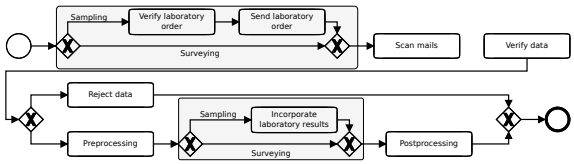


Figure 4: Use case: order processing

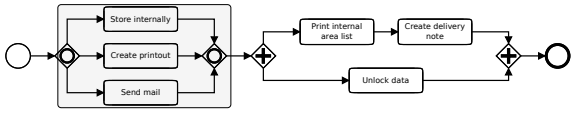


Figure 5: Use case: order postprocessing

email, both, or nothing at all. Even this small example shows the challenges concerning configuration using BPMN, since the notation does not provide direct support for a distinction between decisions that can be made during configuration (variation points) and decisions that can only be made during runtime (decision points). Due to this fact, it is often challenging to adapt process models for a specific situation. In addition, the process model representation solely does not provide any information about dependencies between possible variation points.

For example in the use case presented in Fig. 4, the variations soil sampling and land surveying occur on two occasions. However, if sampling was selected at the first variation point it also needs to be selected at the second variation point. This challenge is further increased by the existence of scattered process models in companies.

3.2 Import of Process Models

In this section we first give an overview about transforming various workflow patterns into the service metamodel representation. Afterwards, the transformation of the use case is presented.

3.2.1 Transformation rules

The general transformation is presented in the following figures. Fig. 6 shows a snippet of a

BPMN model. It consists of the five activities *A*, *B*, *C*, *D*, and *E*. While activities *B*, *C*, and *D* are encapsulated in a building block using a generic gateway, the other activities are sequentially linked. The semantics of the gateway is not defined, since it is not relevant for the general transformation process.

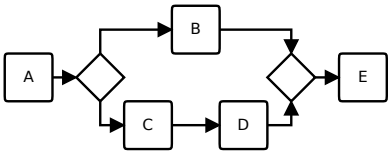


Figure 6: Transformation step 1

The transformation is performed block-wise, i.e., the single blocks of the BPMN model are independently transformed into service component representations. Afterwards these service components are connected with each other. In general, a service component *X** is generated for every activity *X* from the process model.

In the example, the first block that is transformed is the sequence of activities *C* and *D*. A super component *[C*,D*]* is created to encapsulate the service components *C** and *D**. The semantics of the connector linking super and subcomponents with each other depends on the connection type. For workflow-specific connections, the respective connector semantics is presented below (cf. Tab. 2). Since activities *C* and *D* are in sequential order, the respective components are annotated with a temporal dependency, i.e., the service model contains a dependency *before(D*)=C**. The resulting service model is presented in Fig. 7.

As a second step, the service component *B** that represents activity *B* is linked with the existing service model. Therefore, a new connector and a super component to encapsulate components *B* and *[C*,D*]* are created. Since there is no sequential relation between *B* and the other two activities in the encapsulated

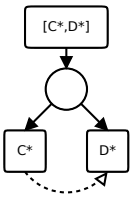


Figure 7: Transformation step 2

block of the process model, no temporal dependency is created as can be seen in Fig. 8.

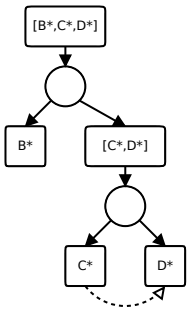


Figure 8: Transformation step 3

As a final step, the components *A** and *E** are included in the model, too. The root component *[A*,B*,C*,D*,E*]* encapsulated all generated components and the temporal dependencies *before([B*,C*,D*])=A** and *before(E*)=[B*,C*,D*]* reflect the sequential order of the activities. The resulting service model is presented in Fig. 9.

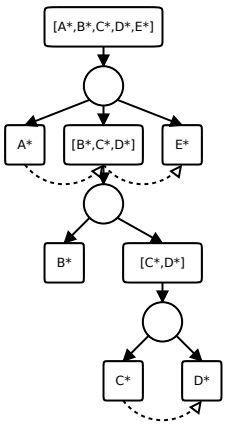


Figure 9: Transformation step 4

Based on this general procedure, it is pos-

sible to establish transformation for a variety of workflow patterns that frequently occur in process models, e.g., van der Aalst et al. 2003. Tab. 2 shows different transformations for exemplary chosen basic patterns. A more thorough discussion about the specific transformations can be found in Klingner and Becker (2014). Though empirical validation of the frequency of workflow patterns is still sparse (van der Aalst and Hofstede 2012), it is a feasible assumption that the presented patterns cover a broad range of models from practice.

3.2.2 Transformation of the use case

Using the above presented rules, it is possible to transform the postprocessing use case process of Fig. 4 and Fig. 5 into a configurable process model. There are several workflow patterns occurring in the process: sequence, parallel split and synchronisation, and exclusive choice and simple merge. The result of the transformation is shown in Fig. 10 and Fig. 11. While the left-hand side (Fig. 10) presents the result of the transformation of Fig. 4, the right-hand side (Fig. 11) presents the transformed process of Fig. 5.

As Fig. 10 shows, the exclusive gateways were transformed into K_{ONE}-connectors, i.e., it is only allowed to select one of the following components during configuration. Furthermore, sequential activities (e.g., components *Scan mails* and *Verify data*) are transformed into components and an K_{ALL}-connector, since all components are mandatory. The temporal dependencies that need to be taken into account due to sequential constraints are depicted as dashed arrows in Fig. 10.

The configuration tree shown in Fig. 10 is a graphical representation of the formally defined metamodel. In the metamodel, temporal dependencies are specified using linear temporal logics (LTL). Thus, the sequential

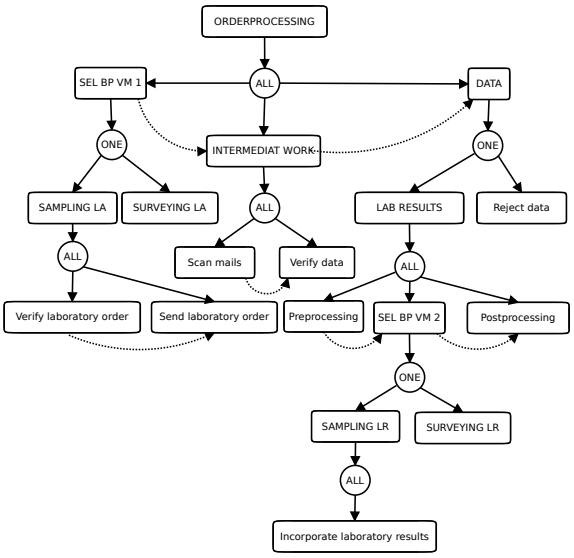


Figure 10: Automatically transformed use case: transformed order processing

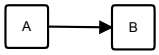
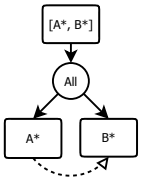
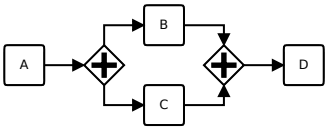
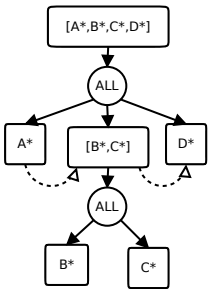
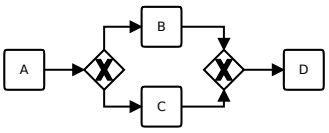
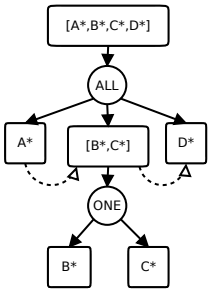
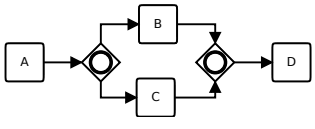
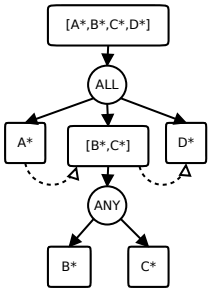
dependency between components *Scan mails* and *Verify data* is defined by the formula *iBefore(Verify data) = Scan mails*. This formula is evaluated during configuration according to predefined LTL expressions (Becker and Klingner 2012).

3.3 Service Configuration

The automatically generated service model can be adapted and extended to fit validity constraints and can serve as a basis for creating customer-individual configurations. In this section, necessary adaptations and possible extensions are presented.

During the transformation of process models into service models, components are derived from activities and connectors from gateways. Furthermore, temporal rules of service components are derived from the control flow of the process model. However, an automatically generated service model only contains elements relevant for process modelling. A number of features that are necessary for a meaningful service modelling are still missing and need to be added.

Table 2: Transformation of basic workflow patterns

Workflow pattern	Service metamodel	Description
		*Sequence*: For an ordered sequence of activities A and B, the respective service components $A*$ and $B*$ are connected with a direct precedence rule: *iBefore(B*) = A**. Both components are encapsulated in the super component *[A*,B*]*.
		*Parallel split and synchronisation*: Using the parallel split, both activities B and C must be executed. After finishing execution, the process is synchronised again. In terms of a service representation, components $B*$ and $C*$ are both mandatory and, thus, connected using an K_{ALL} connector. There is no sequential dependency between components $B*$ and $C*$ but between $A*$, *[B*,C*]*, and $D*$ to reflect the sequential order between the respective parts of the process model.
		*Exclusive choice and simple merge*: In the process model, exactly one of the activities B or C is executed. Analogously, the service components $B*$ and $C*$ are connected using a K_{ONE}-connector, i.e., exactly one of these two components needs to be selected during configuration. Furthermore, the same temporal dependencies as in the previous example apply.
		*Multi choice and synchronising merge*: An arbitrary combination of activities B and C is executed using the multi-choice pattern, i.e., either only B, only C, or both B and C are executed. This is reflected by the K_{ANY}-connector in the service model representation. During configuration, at least one of both components needs to be selected but it is possible to select both components, too. Again, the temporal dependencies are equal to the first example.

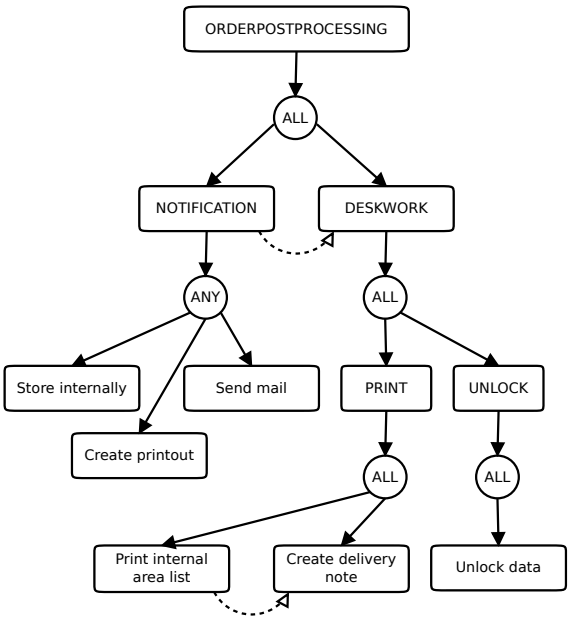


Figure 11: Automatically transformed use case: transformed order postprocessing

Establish integrated portfolio: By means of the automated transformation of process models, introducing the service metamodel in company infrastructure is simplified. However, manual reworking is necessary to establish a meaningful service portfolio of a company. First of all, using automated transformation generates a service model for every process model. To analyse company-wide dependencies, it is necessary to establish a holistic portfolio. In the use case in Fig. 11, the service model *ORDERPOSTPROCESSING* is directly connected with the component *Postprocessing* from the service model *ORDERPROCESSING*.
Thus, the *ORDERPOSTPROCESSING* model can be connected to establish an integrated portfolio.

Remove runtime decisions: Besides integrating different service models with each other, it is necessary to analyse the model regarding meaningfulness of the generated components. The decision between the two process variants soil sampling and land surveying can be made already during configuration. However, this is not the case for the decision between accepting and denying data. This is a runtime decision because it cannot be decided whether the data are correct or not before process execution. It is not possible to distinguish between these two decision types during transformation without adding additional information. Instead, the generic model needs to be adapted. A valid configuration model that represents the correct distinction between configuration and runtime decision can be established by removing the components *Reject data*. The corresponding activities need to be defined using an embedded process model in the component *DATA*.

Identify remote dependencies: Using process models, it is only possible to define local dependencies between activities using gateways. Contrary, it is rather complicated to define remote dependencies, e.g., between different decisions during process execution. Regarding the use case *Order processing* from Fig. 4, there is an implicit remote dependency: If the sampling was selected in the first gateway block, it also needs to be selected in the second gateway block. In case of service configuration, it is necessary to define a dependency between the respective components for enabling valid configuration options. Thus, two rules need to be defined in the transformed use case: *requires(SAMPLING LA) = SAMPLING LR* and *requires(SAMPLING LR) = SAMPLING LA*. Because of the K_{ONE}-connectors, the respective surveying components are automatically disabled during configuration. Due to this explicit definition of configuration dependencies, invalid configurations can be avoided.

Define KPI: Key performance indicators (KPI) are used to represent characteristics related to the productivity of components. A KPI can be defined as a fixed value for a specific component. In addition, it is possible to aggregate KPI of different components by

using mathematical operations. In doing so, it is possible to evaluate different configuration variants according to their productivity. This allows for selecting the most productive service configuration. Besides productivity, other KPI like risk of service provision can be taken into account. This might be necessary, since single components can increase the risk of the whole service (Klingner et al. 2012).

Clean portfolio: During automated transformation, several intermediate components are generated but not required for a precise service description. For increasing comprehensibility of the generated service model, these components should be removed. Furthermore, it is possible to modify a generated service model according to the underlying configuration decisions to reduce the amount of logical dependencies between components and, thus, increase comprehensibility and satisfiability of validity checking (Mendonca et al. 2009). For example, in the use case, the components belonging to soil sampling and these belonging to land surveying can be bundled within specific components. This allows for a better overview about the content of specific service components.

Define non-functional characteristics: For a precise description of a service component, it is necessary to define additional characteristics besides validation options. Using non-functional properties, it is possible to specify such characteristics of components, e.g., temporal or local availability, rights and obligations of contracting parties, and languages the service can be provided in (O'Sullivan 2008). In the metamodel, attributes that are assigned to components are used to define non-functional properties. Besides increased transparency regarding service configuration properties, this also allows for filtering complex service models according to specific non-functional properties. In doing so, it is possible to restrict the selection possibilities to components that are relevant for the specific requirements of customers.

Define service environment: Since services cannot be analysed without considering the environment in which they are provided, it is necessary to include characteristics of this environment during service configuration. The environment of a service might influence different service properties like productivity, price, and configuration options. Using the metamodel, it is possible to define the service environment by means of external impact factors (Becker et al. 2011).

The various service specific extensions for the use case are shown in Fig. 12. Each configurable component has been enriched with a specific price (depicted using the attribute p) that can be summed up regarding customer-individual configurations. The portfolio was cleaned up, i.e., unnecessary intermediate components were removed and logical dependencies between remotely coupled decisions were established. In addition, the component *Reject data* was removed, since it does not represent a variation point. For increased comprehensibility, the logical and temporal dependencies are textually specified, too.

Using the KPI and the specification of dependencies, it is possible to establish valid customer-individual configurations by selecting required components. For example, in a soil sampling offer, the component *Incorporate laboratory results* is automatically included. The price of a configuration is calculated by the sum of all selected components. For the example of a land surveying order including a printout of the results the overall price is determined by the price of the components *Surveying* (300), *Scan mails* (15), *Verify data* (75), *Preprocessing* (150), *Create printout* (80), *Print internal area list* (20), *Create delivery note* (20), and *Unlock data* (5) resulting in a total price of 665.

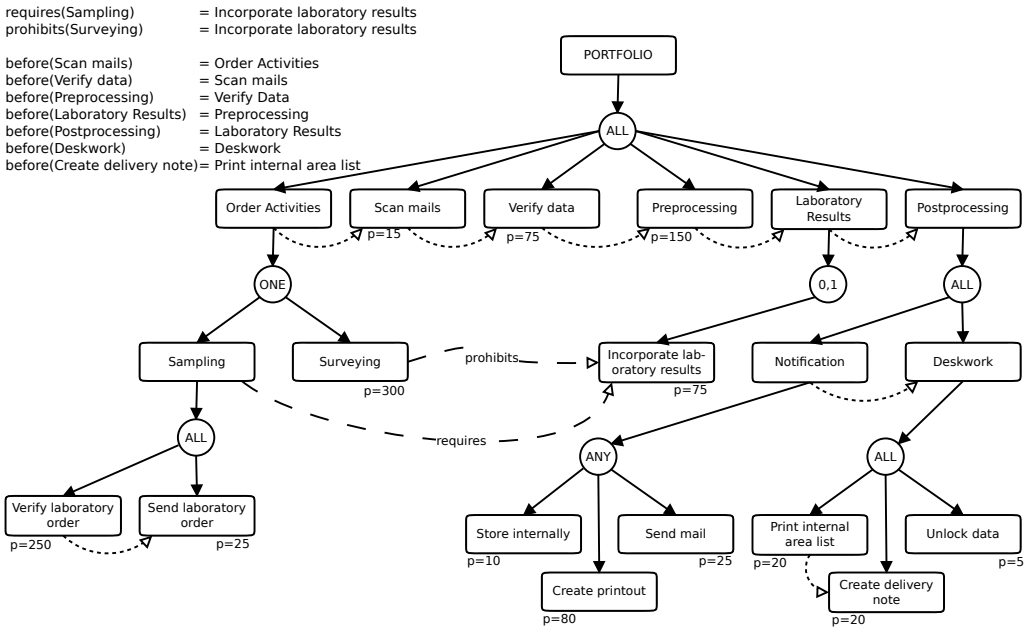


Figure 12: Manually enriched use case

3.4 Export of customer-individual service configurations

Based on the customer-individual configuration of a service, exporting the configuration model into an executable process model can form the basis for (semi-)automated service provision. The export is performed with respect to the temporal dependencies defined in the service model. Components without any temporal dependencies are assumed to be executable in parallel, since no restrictions are defined. Temporal dependencies between components are specified declaratively, for example by using the above mentioned rules *before* (precedence rule) or *iBefore* (immediate precedence rule).

The temporal rules as defined in Tab. 1 are applied during transformation of configured service models into process models. It is necessary to note that a temporal rule is only applicable if all contained elements are selected during configuration. For example, a rule containing components A and B does not apply if only component A is selected.

The transformation of configured service models into process models can be seen as the reverse of the transformation of workflow patterns as described above. Contrary to existing process models, these transformed process models do only contain elements required for service provision, since they are generated from a configuration, i.e., every activity in the generated process is necessary. Therefore, the only above presented workflow patterns that can occur in these types of models are sequence, parallel split, and synchronisation. Thus, during generation of the configured process models neither inclusive nor exclusive gateways are created. A generated process model may only contain these gateway types by encapsulating process models into components for describing the workflow of atomic components (the leafs in the hierarchic tree).

In the use case from Fig. 12, it is possible to distinguish between two configuration variants. On the one hand, the service can be configured for soil sampling. On the other hand, land surveying can be configured. Depending on the selected configuration variant, different

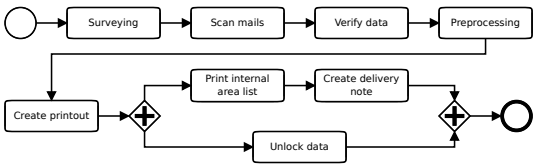


Figure 13: Export of the land surveying configuration

process models are generated. As can be seen in Fig. 13 for the land surveying configuration and selected printout, the generated process is less complex than the original process including both configuration variants. As stated above, the runtime decision about data verification was encapsulated in the component *Verify data*. Thus, it is not represented in the generated process model but rather included as a subprocess of the specific activity.

4 Evaluation

Currently, there are two evaluation approaches for the presented work. First software tools were developed to show the practical feasibility of the findings. Furthermore, an argumentative evaluation according to (Frank 2006) is presented.

4.1 Software implementation

The theoretical concepts presented above were prototypically implemented in various, loosely coupled software tools. Therefore, it is possible to reuse existing process models transformed to a configurable service model representation and configure the service model in practice, too. The tools are freely available and are subsequently presented in more detail.

The tool *PM2SM*[2] enables the transformation of process models to service models. Currently, EPC models can be imported by transforming EPC functions into components and removing events. Future extensions of this tool will allow for transforming BPMN models, too.

The web based application *Service Modeller*[3] is used to model and configure services. Besides importing models transformed using *PM2SM*, it also allows for creating new service models from scratch. Since the application is web based, no installation is necessary. For supporting productivity analysis, the *Service Modeller* provides a variety of predefined KPI that can be used for a more detailed description of service components. The KPI library is the result of an exhaustive literature study as presented in Freitag et al. (2011).

The configuration of complex services is supported by an automated validation of dependencies between different elements of the service models. The configuration view of the *Service Modeller* is shown in Fig. 14. Components that are selected during configuration are marked with a green tick. Besides verifying the configuration validity, the configuration view also allows for comparing different configuration variants regarding their impact on productivity, e.g., by depicting the total price of a service configuration.

Using the tool *SM2PM*[4], it is possible to transform a completed service configuration into a BPMN model. This process model can be used as input for a workflow management system to support process execution. The temporal dependencies between components of the service models serve as a foundation for defining the order of activities in the process model. By means of a web service, it is possible to initialise the transformation within the *Service Modeller* web application. For increasing comprehensibility of the generated process model, the *SM2PM* tool also includes a so-called *Layouter*. The task of this module

[2] http://sourceforge.net/projects/kpstools/

[3] http://europa.informatik.uni-leipzig.de/servicemodeller/

[4] Available as part of the *Service Modeller*, source code yet to release.

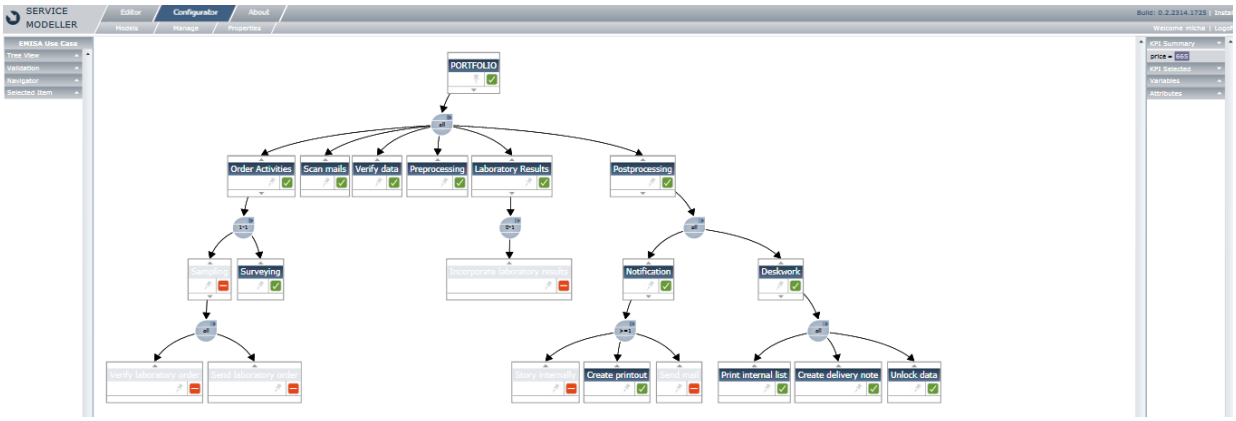


Figure 14: Screenshot of the service modeller: configuration view

is to enrich generated models with information concerning diagram interchange based on the BPMN Diagram Interchange (BPMNDI) specification. A generated process model from the configured service is shown in Fig. 15.

4.2 Argumentative evaluation

Frank (2006) has proposed a framework for evaluating reference models which can be applied for an argumentative evaluation of this work, too. The framework consists of four perspectives that comprise several criteria.

Economic perspective: The proposed service metamodel was established in several design iterations with feedback from practice in terms of workshops. It is founded on well-known service modelling principles that can be easily applied using the provided software tools. Therefore, the costs for training, adaption, and application should be minimal. To reduce introduction costs, the process model transformations as presented in this work can be applied. Using these transformation, companies can reuse their existing models and do not have to design their service portfolio from scratch. However, it should not be unmentioned that several manual adaption still is necessary which might be rather time-consuming. The transformations use existing standards like BPMN and, thus, can be easily integrated with existing IT tools in companies, e.g., process engines. Furthermore, the tools necessary for applying the presented approach are available for free and can be adapted to company-specific needs.

Deployment perspective: As stated above, the metamodel uses well-known service modelling concepts. In addition, companies are familiar with process modelling and, thus, can comprehend the used terminology. Using the *Service Modeller* is explained in different technical documentation, e.g., in Becker et al. (2013b). The presented tools are prototypically implemented and might require adaptations to existing IT infrastructure. The *not-invented-here-syndrome* might apply, since companies usually have an established tool chain concerning service provision. However, it can be faced, since the metamodel is defined using formal logics and can, thus, be implemented in company-specific software tools.

Engineering perspective: The main requirement – reusing existing process models – can be met using the proposed transformation approach and is applicable as demonstrated by the tool development. However, it is necessary to note that although using the workflow patterns a majority of processes can be covered, it might not be possible to transform every situation that can occur in process models.

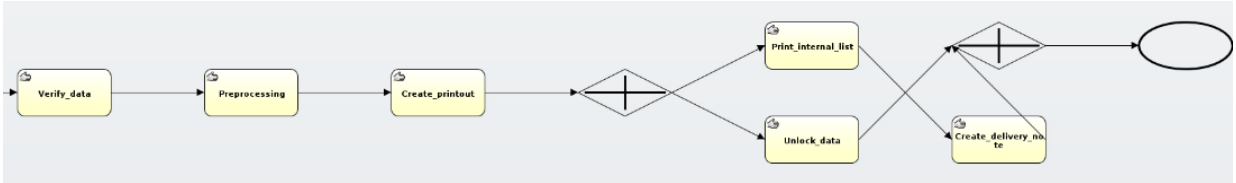


Figure 15: Generated process model

.

The applicability of the export module was analysed in the research project EUMONIS[5]. In this project, a service platform was conceptualised that enables automated provision of IT services of different providers. The export functionality can be used to automatically connect service components to the necessary IT services (Sonnenberg et al. 2014).

Epistemological perspective: The underlying core concepts of the work presented here were published in several articles and, thus, part of the scientific discussion. Though we focus on using BPMN models in this work, the approach is abstract enough to be applied to different notations like EPC. In addition to theoretical application, the concepts presented here were also applied in practice and found useful in terms of representing reality (service components and process models). Existing limitations are discussed in more detail in the next section. Critical distance is ensured by realising that the transformation cannot be applied in every situation and that manual postprocessing is necessary. The amount of necessary postprocessing should be assessable by future application of the approach in different use cases.

5 Conclusion

This work presented an approach for customer-individual configuration of services. Since process models are an elementary constituent of service description, the focus of this work was on how to reuse existing process models of companies for service modelling. Based on workflow patterns, process models can be transformed into configurable service models. For validating different configuration variants, these service models can be enriched with additional information using hierarchical and logical dependencies. Furthermore, it is possible to define company-specific KPI to allow for deriving information about possible performance impacts of different configuration variants. The configured service models can be transformed in process models to support service provision, e.g., by importing these process models into workflow engines.

Due to the different conceptual basis of service models and process models, a few challenges emerge that also reflect current limitations of the presented approach. For an effective and mostly automatic transformation it has to be assumed that the modelled activities within the processes conform to the modules of the service portfolio. Although, in real world it is not always meaningful to transform every activity into a module of the service portfolio, as the service model contains mainly the service parts required for configuration. Thus, both modelling approaches focus different levels of abstraction and modelling objects.

Therefore, it is to be expected that the import of process models requires adaptation steps and imported models can merely serve as guidance in many cases. Thus, there is manual postprocessing necessary when process models are transformed. In particular in companies with a large number of existing

[5]http://eumonis.org

processes this might be error-prone and possibly too time-consuming for being applicable. To overcome this limitation, it is planned to automatise additional cleaning steps like removing intermediate components. Besides intermediate components, decision points are another factor that adds to the amount of manual postprocessing. A first approach for identifying differences between decision and variation points can be found in Milani et al. (2012). Based on this work, several automation techniques are conceivable to minimise necessary manual work.

Regarding the export of service models to process models, reciprocal challenges emerge. Whereas service models to create customer-individual configurations contain mostly service components of a rather high abstraction level, the export of process models describing the detailed customer-individual process is desirable. To include the required information, the attachment of process models on the leafs of a service model can provide a way to augment the service portfolio with additional processual data. However, several challenges might emerge when process models are included in the leafs. For example, duplicate activities can occur when equal activities are embedded in different service components. It is necessary to analyse approaches on how to deal with this challenge.

Generally, both previously described problems can be counteracted by aligning the abstraction level of the process and service model. However, due to different modelling objects and objectives, this goal might be hard to achieve. Nevertheless, there are a couple of reasons for using the service model instead of the process model for configuring customer-individual services:

1. *Clearly defined distinction between configuration and runtime decisions*: Based on the distinction of variation and decision points, both the complexity of the configuration as well as the complexity of service provision are reduced. In both cases it is only necessary to make decisions according to the specific service lifecycle phase. This is of special importance, since in most companies different responsibilities for service sales and provision exist. Due to the fact that the generated process models do not contain any configuration decisions, the comprehensibility for the responsible employees is increased.
2. *Formally defined metamodel for service configuration*: Nowadays, the structured description of services is a great challenge for companies (Gronau et al. 2010). The metamodel that is used in this work provides a rather simple approach for hierarchic modelling of complex services. While the basic concepts are easy to learn, the metamodel also provides advanced concepts like KPI, logical dependencies, and service constraints that allow for describing complex dependencies between components of a service portfolio.
3. *Integration of information regarding productivity*: One of the challenges that today's service companies face is to distinct themselves from competitors, e.g., by providing services more efficient than these competitors. For being able to do so, it is of great importance to analyse and improve service productivity. Using KPI it is possible to assess potential productivity impacts of different configuration variants in a specific application area. On the one hand, this allows for more realistic risk assessment. On the other hand, it is possible to compare different configuration variants and select the best alternative.
4. *Integrated representation of different services*: Especially in larger companies, a great number of process models is used to describe both internal and external services (Dumas et al. 2009). This leads to the fact that necessary information are often

dispersed over different sources and need to be searched manually with great effort. Managing these processes usually requires relatively complex process repositories (Yan et al. 2009). By providing an integrated representation of the portfolio, the service metamodel allows for making and documenting changes at one central point. In addition, it is possible to validate the consistency of a portfolio already during modelling of newly created or modified service components.

5. *Holistic tool support*: The tool chain presented above enables companies to manage their complete service portfolio at one central place. The tool are publicly available which is of particular advantage for SME, since these companies usually cannot afford buying and managing complex IT infrastructures. By using public standards like BPMNDI and other XML based interchange formats, it is possible to integrate these tools into existing company infrastructures.

Though the presented approach provides a variety of different benefits, it is necessary to note that service models cannot replace process models, since it is not possible to model all process relevant functionalities. For example, events cannot be modelled and the interaction with other companies can be defined only indirectly (e.g. by annotating service components). For that reason, service models need to be transformed into process models eventually. These generated models can be enriched by further technical details that are of relevance for process execution.

6 Acknowledgement

Parts of the work were funded by grants of the German Ministry of Education and Research in the context of the joint research project “IPS” (01IS12013B) under the supervision of the PT-DLR. We also thank the anonymous reviewers for their valuable feedback that helped to improve the contributions of this paper.

References

Bask A., Lipponen M., Rajahonka M., Tinnilä M. (2010) The concept of modularity: diffusion from manufacturing to service production. In: Journal of Manufacturing Technology Management 21(3), pp. 355–375

Becker M., Klingner S. (2012) Towards Customer-Individual Configurations of Business Process Models. In: Bider I., Halpin T., Krogstie J., Nurcan S., Proper E., Schmidt R., Soffer P., Wrycza S. (eds.) Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing Vol. 113. Springer Berlin Heidelberg, pp. 121–135

Becker M., Klingner S. (2013) Formale Modellierung von Komponenten und Abhängigkeiten zur Konfiguration von Product-Service Systems. In: Thomas O., Nüttgens M. (eds.) Dienstleistungsmodellierung 2012. Springer Fachmedien Wiesbaden, pp. 114–140

Becker M., Laue R. (2012) A comparative survey of business process similarity measures. In: Computers in Industry 63(2), pp. 148–167

Becker M., Klingner S., Böttcher M. (2011) Configuring services regarding service environment and productivity indicators. In: Ganzha M., Maciaszek L., Paprzycki M. (eds.) Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on. IEEE Computer Society Press, Szczecin, Poland, pp. 505 –512

Becker M., Klingner S., Sonnenberg M., Ritter J. (2013a) A comparative survey of service configuration approaches. In: RESER 2013: Finding growth through service activities in barren times. RESER. Aix en Provence, France

Becker M., Klingner S., Schumacher F. (2013b) Werkzeug zur komponentenbasierten Modellierung und Konfiguration von Dienstleistungen. In: Klingner S., Meiren T., Becker M. (eds.) Produktivitätsorientiertes Service Engineering. Leipziger Beiträge zur Informatik Vol. 38. Leipziger Informatik Verbund LIV, pp. 113–146

Böhmann T., Junginger M., Krcmar H. (2003) Modular Service Architectures: A Concept and Method for Engineering IT Services. In: 2013 46th Hawaii International Conference on System Sciences 3(1), 74b

Böttcher M., Klingner S. (2011) The Basics and Applications of Service Modeling. In: SRII Global Conference 2011

Cao J., Wang J., Law K., Zhang S., Li M. (Apr. 2006) An interactive service customization model. In: Inf. Softw. Technol. 48(4), pp. 280–296

Carlborg P., Kindström D., Kowalkowski C. (2013) A lean approach for service productivity improvements: synergy or oxymoron? In: Managing Service Quality 23, pp. 291–304

Davis F. D. (Sept. 1989) Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. In: MIS Q. 13(3), pp. 319–340

Dijkman R., Dumas M., van Dongen B., Käärik R., Mendling J. (2011) Similarity of business process models: Metrics and evaluation. In: Information Systems 36(2), pp. 498 –516

Dumas M., Garcia-Banuelos L., Dijkman R. (2009) Similarity Search of Business Process Models. In: IEEE Data Engineering Bulletin 32(3), pp. 23–28

Frank U. (2006) Evaluation of Reference Models. In: Reference Modeling for Business Systems Analysis Fettke P. (ed.)

Freitag M., Lamberth S., Klingner S., Böttcher M. (2011) Method of collecting and categorising performance indicators to measure the productivity of modular services using an IT tool. In: Ganz W., Kicherer F., Schletz A. (eds.) RESER 2011 Productivity of Services NextGen - Beyond Output / Input. Conference Proceedings. Fraunhofer Verlag, Hamburg, Germany

Gronau N., Bahrs J., Hake M., Heinze P., Lembcke R., Scharff C., Vladova G. (2010) Wissensorientierte Modellierung im Lebenszyklus von Dienstleistungen. In: Thomas O., Nüttgens M. (eds.) Dienstleistungsmodellierung 2010. Physica-Verlag HD, pp. 3–23

ter Harmsel M. (2012) Mass customization as a solution for the service industry. MA thesis, University of Twente, Twente, Netherlands

Hart C. W. (1995) Mass customization: conceptual underpinnings, opportunities and limits. In: International Journal of Service Industry Management 6(2), pp. 36–45

Heiskala M., Paloheimo K.-S., Tiihonen J. (2005) Mass Customization of Services: Benefits and Challenges of Configurable Services. In: Frontiers of e-Business Research (FeBR 2005). Tampere University of Technology, Tampere, Finland, pp. 206–221

Hevner A. R., March S. T., Park J., Ram S. (Mar. 2004) Design Science in Information Systems Research. In: MIS Q. 28(1), pp. 75–105

Klingner S., Becker M. (2012) Formal Modelling of Components and Dependencies for Configuring Product-Service-Systems.. In: Enterprise Modelling and Information Systems Architectures 7(1), pp. 38–56

Klingner S., Becker M. (2014) Konfiguration von Dienstleistungen – Ein Ansatz zur Verbindung von Geschäftsprozessen mit Dienstleistungsmodellen, German. In: Thomas O., Nüttgens M. (eds.) Dienstleistungsmodellierung 2014. Springer Fachmedien Wiesbaden, pp. 76–96

Klingner S., Böttcher M., Becker M., Döhler A. (2011) Managing complex service portfolios. In: Ganz W., Kicherer F., Schletz A.

(eds.) RESER 2011 Productivity of Services NextGen - Beyond Output/Input. Conference Proceedings

Klingner S., Becker M., Döhler A., Swialkowski R. (2012) Modellierung von Dienstleistungsportfolios - Ist-Stand und zukünftige Herausforderungen der Praxis. In: Meyer K., Abdelkafi N. (eds.) Smart Services and Service Science Vol. 36. LIV, Leipzig, Germany, pp. 119–127

Lampel J., Mintzberg H. (1996) Customizing Customization. In: MIT Sloan Management Review Fall, pp. 21–30

Liu L., Xu W. (2009) Research on Mass Customization Strategies in Non-physical Products Service Industries. In: Information Science and Engineering, International Conference on 0, pp. 4441–4444

Mendonca M., Wasowski A., Czarnecki K. (2009) SAT-based analysis of feature models is easy. In: Proceedings of the 13th International Software Product Line Conference. SPLC '09. Carnegie Mellon University, San Francisco, California, pp. 231–240

Milani F., Dumas M., Matulevicius R. (2012) Identifying and Classifying Variations in Business Processes. In: Bider I., Halpin T., Krogstie J., Nurcan S., Proper E., Schmidt R., Soffer P., Wrycza S. (eds.) Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing Vol. 113. Springer Berlin Heidelberg, pp. 136–150

O'Sullivan J. J. (2008) Towards a precise understanding of service properties. PhD thesis, Queensland University of Technology, Faculty of Information Technology

Pine B. J. (1999) Mass Customization: The Frontier in Business Competition, 2nd ed. Harvard Business School Press, Cambridge, MA, USA

Rosa M. L. (2009) Managing Variability in Process-Aware Information Systems. PhD Thesis, Queensland University of Technology, Brisbane, Australia

Rosa M. L., van der Aalst W. M., Dumas M., Milani F. P. (2013) Business process variability modeling : A survey. Queensland University of Technology. Last Access: ACM Computing Surveys

Rosemann M., van der Aalst W. (2007) A configurable reference modelling language. In: Information Systems 32(1), pp. 1 –23

Sonnenberg M., Schmidt J., Nitzschke M. (2014) IIntegrierte Service- und Softwareengineering. In: EUMONIS Abschlussband. in press. LIV

Sundbo J. (2002) The Service Economy: Standardisation or Customisation? In: The Service Industries Journal 22(4), pp. 93–116

Tuunanen T., Bask A., Merisalo-Rantanen H. (2012) Typology for Modular Service Design: Review of Literature. In: International Journal of Service Science, Management, Engineering, and Technology (IJSSMET) 3(3), pp. 99–112

van der Aalst W., ter Hofstede A. (2012) Workflow patterns put into context. In: Software and Systems Modeling 11 (3), pp. 319–323

van der Aalst W., ter Hofstede A., Kiepuszewski B., Barros A. (2003) Workflow Patterns. In: Distributed and Parallel Databases 14 (1), pp. 5–51

Weber B., Reichert M., Mendling J., Reijers H. A. (2011) Refactoring large process model repositories. In: Computers in Industry 62(5), pp. 467 –486

Yan Z., Dijkman R., Grefen P. (2009) Business Process Model Repositories - Framework and Survey. Working Papers 292. Technische Universiteit Eindhoven. Eindhoven

Michael Becker
University of Leipzig, Department of Business Information Systems, Hainstr. 11, 04109/Leipzig
mbecker@informatik.uni-leipzig.de

Stephan Klingner
University of Leipzig, Department of Business Information Systems, Hainstr. 11, 04109/Leipzig
klingner@informatik.uni-leipzig.de