A Metaheuristic Approach for the Configuration of Business Process Families

Ivana Ognjanović*, Bardia Mohabbati†, Dragan Gašević†‡, Ebrahim Bagheri‡ and Marko Bošković§
*University Mediterranean, Montenegro Email: ivana.ognjanovic@unimediteran.net
†Simon Fraser University, Canada Email: mohabbati@sfu.ca
‡Athabasca University, Canada Email: {dragang, ebagheri}@athabascau.ca
§Research Studios Austria, Austria Email: boskovic@researchstudios.at

***Abstract*—Business process families provide an over-arching representation of the possible business processes of a target domain. They are defined by capturing the similarities and differences among the possible business processes of the target domain. To realize a business process family into a concrete business process model, the variability points of the business process family need to be bounded. The decision on how to bind these variation points boils down to the stakeholders' requirements and needs. Given specific requirements from the stakeholders, the business process family can be configured. This paper formally introduces and empirically evaluates a framework called *ConfBPFM* that utilizes standard techniques for identifying stakeholders' quality requirements and employs a metaheuristic search algorithm (i.e., Genetic Algorithms) to optimally configure a business process family.**

I. Introduction

Reusability is a desirable attribute in business process modeling. Recently, business process modeling research has identified a high demand for a new class of business process models that facilitate reusability through the so-called *business process model families (BPMFs)* [1]. BPMFs (also known as configurable business processes [2]) capture processes that are recurrent to many organizations/situations through a systematic modeling of variability and commonality. While reference business processes have a similar objective (e.g. [3]), BPMFs can be supplemented with computational support for resolving variability in a way that best satisfies stakeholders' requirements. Aiming to address the needs for such a process, the present research has built on the following two disciplines: i) business processes modeling – through the extensions of existing process modeling languages, which can guide the configuration process (e.g., questionnaire-based configuration) [4]; ii) Software Product Line Engineering (SPLE) – through the use of well-known variability modeling techniques (e.g., feature models) that can also guide the configuration process of other software artifacts (i.e., business process models) [5]. Both research directions produce automated configuration approaches built on sound formal methods [6], [1], which can assure the behavioral correctness of each business process configuration derived from a BPMF.

The research on BPMFs has so far only focused on the configuration process based on functional requirements of (multiple) stakeholders. To our knowledge, no attention has been paid to non-functional requirements. Given that the implementation of modern business processes is often based on service-oriented architectures, the consideration of non-functional requirements is essential. This is needed in order to assure that services selected for implementation of the activities of a business process optimally satisfies quality expectations of the stakeholders (e.g., minimal delivery cost). Traditionally, approaches for the selection of services based on non-functional requirements leverage Integer Linear Programming (ILP) [7]. As discussed in Sect. VI, recent research on service selection has identified scalability as the one of the limitations of ILP approaches for this purpose. Similarly, the interaction between complex non-functional requirements used in practice are often not of linear nature, and thus, the use of ILP is not possible for an arbitrary quality function.

To this end, this paper proposes the ConfBPMF framework for configuration of BPMFs based on both functional and non-functional requirements. In ConfBPMF, services selection is a constitutive part of the configuration process. The first contribution of ConfBPMF is a framework for capturing non-functional requirements through: *preferences* expressing relative importance of quality properties (e.g., high cost is acceptable if response time is low); and *constraints* over the values of quality properties (e.g., implementation cost must be less than $1000). For such non-functional requirements, the second contribution of ConfBPMF facilitates business process quality measurement based on the well-known Analytic Hierarchy Process [8]. This measurement is a foundation for the third contribution of this paper – a genetic algorithm for the configuration of BPMFs; the algorithm optimizes the degree of satisfaction of non-functional requirements and preserves the behavioral correctness of each business process which can be derived from a BPMF.

The paper is organized as follows: an overview of the approach with an illustrative example is given in Sect. II. Section III introduces an approach to measuring the quality of BPMF configurations. The GA for BPMF configuration is presented in Sect. IV. Section V reports on the evaluation results, while related work is discussed in Sect. VI.

II. Overview of the Approach

In this section, we give an overview of the proposed work – *ConfBPMF*. First, we present how BPMFs are modeled. Then, we describe how requirements are modeled and ranked. Later on we describe the types of the requirements captured for BPMF configuration by *ConfBPMF*.

1) Representation of BPMFs: ConfBPMF uses well-known SPLE principles for representing BPMFs. In particular, we use model templates, whereby a family is represented by a variability model which defines the variability points in model templates (i.e., business process models of the entire family). For BPMFs, ConfBPMF uses: 1) a Feature model (FM) to describe the variability of the target domain of a BPMF; 2) Business Process Model Template (BPMT), which describes a

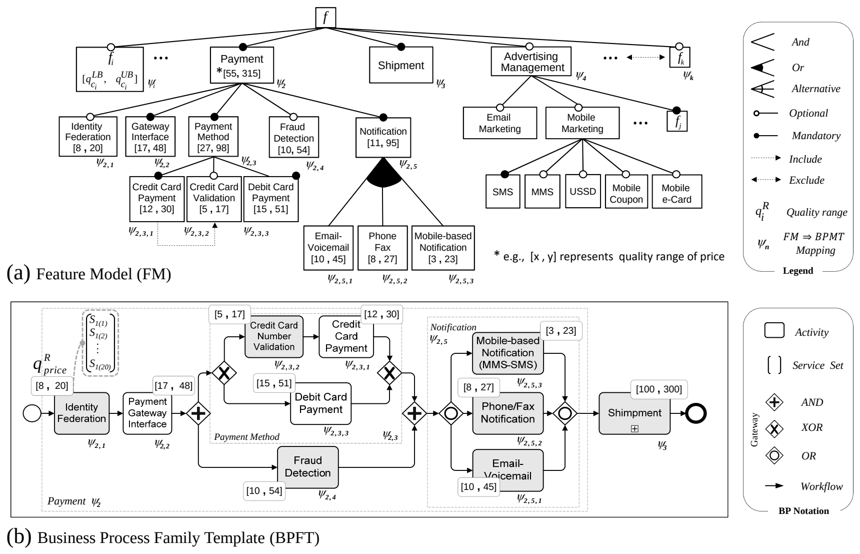


Fig. 1. a) A Feature model. b) A part of BPMF representing the realization and composition of payment and shipment features.

set of business processes (i.e., reference process) of the target domain of the BPMF. ConfBPMF also uses 3) a set of available services per activity in the BPMF, including their QoS characteristics according to the standards such as ISO 8402 and ITU. These three elements of the BPMF representation are illustrated through the running example of the e-shop BPMF given in Fig. 1.

Feature models are widely used to describe a family of applications for a given domain in terms of their common and unique features, and feature interdependencies. *Feature models* (e.g., Fig. 1(a)) support the representation of *mandatory* features – child features that must always appear with their parents; *optional features* – child features that optionally appear with their parents; *alternative* feature groups – a group of features from which only one can be included in any application; and *or* feature groups – a group of features that are optional to be included but at least one needs to be selected. Additionally, interdependencies between features are specified using *integrity constraints*. Two of the most widely used constraints are: *includes* – when one feature implies the inclusion of the other; and *excludes* – when the appearance of one feature excludes some other features. Fig. 1(a) shows a part of the simplified feature model for an e-shop family.

A Business Process Model Template (BPMT) (Fig. 1(b)) consists of a collection of the business processes for the entire family in a superimposed way [9]. In this paper, we assume the following well-formedness conditions on a business process structure: i) a business process model has one start and one end node; ii) every activity node has a single incoming and a single outgoing edge; iii) for every node with multiple outgoing arcs (i.e., a split), there is a corresponding node with multiple incoming arcs (a join), such that the set of nodes between the split and the join form a single-entry-single-exit region [10]. These well-formedness rules are well-accepted for business process modeling and appealing correctness requirements.

To connect BPMTs and feature models, there is an injective (i.e., one-to-one) mapping model ($\psi_i : FM \Rightarrow BPMT$) available which interconnects each feature in the feature model to the corresponding activity in the reference business process model. This mapping and the above BPMT well-formedness constraints, according to the formal verification approach [6], guarantee that every configuration of the business process model is valid with respect to both behavioral characteristics of the business processes (expressed through control flow patterns) and configuration characteristics through the feature models. Our approach uses presence conditions (PC) [9] as annotation properties for each activity within the business process model. The PC of an activity is formulated as a boolean expression of ψ_i variables corresponding to the features mapped to the activity. Both the feature and activity constructs refer to the elements of feature models and BPMTs. Thereby, when domain engineers map features to activities, the activities' PCs are defined. During the configuration process, when a feature f_i is removed from the configuration, its corresponding ψ variables are set to *false*.

Each activity a_j, $1 \leq i \leq |A|$ in the BPMT can be implemented using multiple services $S_{a_i} = \{s_1, \ldots, s_{|S_{a_i}|}\}$ (i.e., third input type), of which each service may have equivalent functionality but different QoS characteristics. Table I describes some examples of services with different quality properties; these services offer possible implementations for the *Identify Federation* and *Payment Gateway Interface* activities.

2) Representation and ranking of requirements: ConfBPMF can handle both functional and non-functional requirements. The handling of *functional requirements* expressed as features of feature models is a well-researched topic in SPLE with a number of different algorithms. ConfBPMF builds on the work from [11], which guarantees that the final configuration has the maximal coverage of functional requirements (completeness), which can be a valid (i.e., consistent) configuration (soundness) w.r.t. the feature model. The configuration produced based on functional requirements is a subset of the original BPMF (i.e., feature model and BPMT), in which there are still unresolved variation points (e.g., not all optional features are pruned out or all alternative feature groups are resolved). The rest of the configuration is handled based on non-functional requirements, which is exactly the focus of this paper.

Non-functional requirements in ConfBPMF are represented as: *preferences* represented in terms of expected quality properties and their mutual relations; and *(hard) constraints* over the values of quality properties. The representation builds on the Preview framework well-known in requirements engineering [12]. Concerns and qualifier tags are the two key elements of Preview. Concerns are sets of quality properties that represent the important matters of interest to the stakeholders such as cost, response time, availability, or security. Qualifier tags represent the possible enumerations for each concern (e.g., the qualifier tags for cost could be cheap, expensive, and reasonable). Following the well-known framework for expressing and ranking user requirements – Analytical Hierarchy Process (AHP) [8] – ConfBPMF allows the stakeholders to define their requirements as relative importance between concerns, and between qualifier tags of a concern. Relative importance is typically defined with odd numbers ranging from 1 (equal importance) to 9 (extreme importance of one over the other). The options (e.g., features or services) available to the stakeholders are also associated with qualifier tags. Furthermore, ConfBPMF allows for conditional preferences. For example, stakeholders are often aware that requirement of low cost is hard to meet, so they may define a compromise: they are only prepared to pay a high cost if the RespTime is kept at a very minimum; otherwise, they are only willing to pay a low price.

Once the preferences are defined, ConfBPMF can leverage an extended AHP framework (CS-AHP), which is adopted for use in SPLE [13]. In particular, CS-AHP performs a tuned pair-wise comparison over the stakeholders' (conditional) requirements. The outcomes of the procedure are: i) local ranks of tags, i.e., ranking of qualifier tags within one concern; and ii) global ranks of concerns. Global ranks have values in the [0,1] interval. Additionally, the global ranking of qualifier tags can be computed by multiplying global ranks of concerns with local ranks of qualifier tags. These rankings are used as the main instrument for measuring the level of satisfaction of user requirements with a particular configuration(Section III) i) the set of concerns and their qualifier tags are locally ranked; ii) rank of each available feature (combination of one tag per concern) is calculated based on the ranks of the qualifier tags that are associated with that feature.

However, to select features and services for a particular BPMF configuration automatically, ConfBPMF needs to have values of quality properties for features in the feature model. The quality properties of features are obtained according to [14] by aggregating the QoS of services associated to the activities of the BPMT and by propagating the aggregated values to the features through the mapping relations between the BPMT and the feature model. The aggregation first is based on the control flow patterns of the BPMT, and then, on the variability encoded in the feature model. The outcome is intervals of numeric values (quality ranges) from which each concern can take values for each feature. The quality ranges of concerns are further divided into the quality ranges of each individual concern.

Example 1: (Sample quality ranges) Range values for Cost, Response Time, Availability, and Reliability concerns and their qualifier tags: Cost - range [100, 350] where Low $<$ 200, Medium [200, 300], and High $>$ 300; RespTime - range [15,55] where Low $<$ 20, Medium [20, 40], and High $>$ 40; Availability - range [5, 20] where Low 12 and High $>$ 12; and Reliability - range [7,15] where Low 10 and High $>$ 10.

TABLE I
SERVICE SETS DELEGATED TO EACH ACTIVITY.

$IdentifyFederation(S_1)$				$PaymentGatewayInterface(S_2)$			
	q_{pr}	q_i	q_{tp}		q_{pr}	q_i	q_{tp}
$S_{1(1)}$	8	$\dots$	43	$S_{2(1)}$	27	$\dots$	32
$S_{1(2)}$	14	$\dots$	31	$S_{2(2)}$	48	$\dots$	67
$S_{1(3)}$	20	$\dots$	20	$S_{2(3)}$	17	$\dots$	15
$S_{1(4)}$	11	$\dots$	5				

In addition to preferences, (*hard*) constraints over the values of non-functional properties can be defined in ConfBPMF. For example, the stakeholders might define that the cost value for the Shipment activity should not be over 100. It means that any combination of services which has the best characteristics w.r.t. the stakeholders' other requirements and which violates this specified value of cost should be eliminated and should not be evaluated any further.

3) BPMF Configuration: Once the requirements are obtained, ranks of concerns and qualifier tags are computed and quality ranges are evaluated, the configuration of the BPMF is carried out by selecting the most desirable features and the relevant services that would collectively maximize the stakeholders satisfaction. The configuration is performed using a Genetic Algorithm introduced in Sect. IV.

III. PROCESS CONFIGURATION

A. Quality Measurement

ConfBPMF makes use of CS-AHP for measuring the quality of a given configuration of a BPMF, i.e., the combination of services which includes exactly one service per activity in the BPMF. Since CS-AHP computes ranks over the set of qualifier tags of each concern, the ranks of specific values from those intervals should be obtained as a measure of the quality of the aggregated values of an appropriate combination of services.

First, let us introduce the following notations: v_i – the appropriate aggregated value of the i^{th} concern; $qt(v_i)$ – the qualifier tag of an appropriate concern which v_i corresponds to; $r_{qt(v_i)}$ – local rank for the qualifier tag $qt(v_i)$; $L_{qt(v_i)}$, $U_{qt(v_i)}$, $m_{qt(v_i)}$ – lower bound, upper bound and middle value of the interval of values which corresponds to $qt(v_i)$; $next\_qt(v_i)$ and $prev\_qt(v_i)$- the first neighbor qualifier tags on the right and left side of $qt(v_i)$, respectively. If $qt(v_i)$ is a final left (or right) qualifier tag the value of $prev\_qt(v_i)$ is considered to be 0 ($next\_qt(v_i)$ to be 1) in case of an increasing characteristic, and vise versa in case of a decreasing characteristic. In order to deliver a rank for $v_i \in [L_{qt(v_i)}, U_{qt(v_i)}]$, we make a uniform distribution of the rank of the observed qualifier tag to ranks of the first neighbor qualifier tags ($next\_qt(v_i)$ and $prev\_qt(v_i)$).

The global rank of the aggregated value v_i of concern c_i is calculated according to the basic characteristic of CS-AHP: as multiplication $r(v_i) \cdot r_{c_i}$, where r_{c_i} is the rank of concern c_i. Finally, the quality $R(v)$ of a combination of services is defined with function f (e.g. mean, min, sum) of CS-AHP and the global ranks $v = (v_1, \dots, v_n)$ of the aggregated values of each concern $R(v) = f(r(v_i)r_{c_i}), i \in \{1, \dots, n\}$.

$$r(v_i) = \begin{cases} r_{qt(v_i)} + \frac{v_i - m_{qt(v_i)}}{U_{qt(v_i)} - m_{qt(v_i)}} \left(r_{next\_qt(v_i)} - r_{qt(v_i)}\right), \\ v_i \in [m_{qt(v_i)}, U_{qt(v_i)}] \\ \\ r_{qt(v_i)} - \frac{m_{qt(v_i)} - v_i}{m_{qt(v_i)} - L_{qt(v_i)}} \left(r_{qt(v_i)} - r_{prev\_qt(v_i)}\right), \\ v_i \in [L_{qt(v_i)}, m_{qt(v_i)}] \end{cases} \tag{1}$$

To exemplify, let us consider the previously introduced example of the e-shop and the combination of services which gives the summarized values for QoS properties *Cost*, *ResponseTime*, and *Availability* equal to 275, 19 and 15. As shown in Example 1, those values belong to ranges specified as medium *Cost*, low *ResponseTime* and high *Availability*. Let us also suppose that CS-AHP gives the rank for concern *ResponseTime*: 0.54 and its qualifier tags: low *ResponseTime*, medium *ResponseTime* and high *ResponseTime* as 0.16,0.65, 0.19, respectively.

Given that the rank of 0.65 is defined for the whole interval [15, 20], we assign it to the middle value of the specified interval (i.e. to 17.5). The uniform distribution of ranks from 0.65 to 0.19 (the rank of medium *ResponseTime*) over values of *ResponseTime* 17.5 to 20 gives rank to the value of 19 as follows – $0.65 - \frac{19-17.5}{20-17.5} \cdot (0.65 - 0.19) = 0.37$. Finally, the global rank for *ResponseTime* of 19 is calculated as follows – $0.54 \cdot 0.37 = 0.20$.

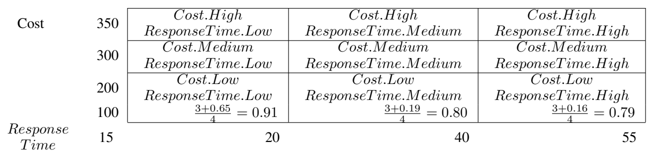
So, in our approach higher values of $R(v)$ correspond to the combination of services that are better suited for the stakeholders' requirements. Furthermore, the distance between two different configurations of services might be obtained as the difference between the measures of their qualities since both configurations essentially provide similar functionalities but have different levels of QoS.

B. Optimization Goal

By introducing the CS-AHP-based quality measurement, we define the final configuration goal as finding the best configuration of services with the most preferable characteristics according to the stakeholders' requirements and goals. Thus, the BPMF configuration goal is defined as follows: *Given a BPMF with a set of stakeholders' preferences, a set of hard constraints, and a set of available services per activity of the BPMF, the goal of the configuration of the BPMF is to find a combination of features and services which maximizes the aggregated quality* $R(s_1, \ldots, s_{|A|})$*, subject to its affiliation to the most preferable rank* $r^{[]}_{c_1,\ldots,c_n}$ *of concerns and qualifier tag and hard constraints satisfaction.*

For the sake of simplicity, let us consider the case of only two concerns: *Cost* and *ResponseTime*. Lets suppose that the intervals for both of concerns obtained based on the set of available services are presented in Example 1: the value of *Cost* is in the interval $[100, 350]$ and the value of *ResponseTime* in $[15, 55]$. Also, the stakeholders have defined that high *Cost* is any value above 300 and medium *Cost* is any value less than 300 and higher then 200. Thus, in general, we cannot say whether the whole BPMF has low, medium or high *Cost* but rather it depends on the specific configuration of the BPMF. The same situation is true for *ResponseTime*, which is high in case of values greater than 40 and medium with values between 20 and 40. Furthermore, in addition to the previously defined ranks over qualifier tags of ResponseTime, the stakeholders can define that low Cost has dominant relative importance; therefore, they are only interested in combinations of QoS properties that include low values for Cost.

So, we have two different types of information to work with: 1) range values for each concern and their corresponding qualifier tags, and 2) the ranks over them. They should be combined in order to define the most preferable ranges of values for each concern. The following steps are performed:



Cost
350
300
200
100
Cost.High ResponseTime.Low
Cost.High ResponseTime.Medium
Cost.High ResponseTime.High
Cost.Medium ResponseTime.Low
Cost.Medium ResponseTime.Medium
Cost.Medium ResponseTime.High
Cost.Low ResponseTime.Low
Cost.Low ResponseTime.Medium
Cost.Low ResponseTime.High
$\frac{3+0.65}{4} = 0.91$
$\frac{3+0.19}{4} = 0.80$
$\frac{3+0.16}{4} = 0.79$
Response Time
15
20
40
55

Fig. 2. Concerns, their values and their rank

(i) Cost interval $[100, 350]$ is divided into three sub-intervals $[100, 200)$, $[200, 300]$ and $(300, 350]$, which correspond to low, medium and high possible cost, respectively. *(ii) ResponseTime* interval $[15, 55]$ is divided into three sub-intervals $[5, 20)$, $[20, 40]$ and $(40, 55)$, which correspond to low, medium and high possible response time, respectively. *(iii)* There are nine possible combinations of QoS characteristics for the BPMF; these combinations are obtained as a Cartesian product of sets of sub-intervals (Fig. 2). *(iv)* All possible combinations of QoS characteristics might be ranked according to ranks obtained by the previously specified stakeholders' requirements and the CS-AHP algorithm. As *Cost* has the dominant relative importance (i.e., rank of low Cost equals 3), it is obvious that the highest ranked combination of QoS properties would include low *Cost* and the highest ranked low qualifier tag of concern *ResponseTime*. The values obtained by CS-AHP are presented in appropriate cells of Fig. 2. *(v)* Based on the calculated values, we conclude that *Cost* being in the range $[100, 200)$ and *ResponseTime* in $[5, 20)$ are the most preferable characteristics for both the observed BPMF and the available set of services. Once we have an estimate for the desirable levels of quality, we can formulate the goal to find the best combination of services that gives the most preferable aggregated QoS characteristics.

IV. Adoption of GA for BPMF configuration

In this section, we explain how Genetic Algorithms (GAs) are adopted for BPMF configuration. GAs are a form of metaheuristic search that solve problems using algorithms inspired by the processes of the neo-Darwinian evolutionary theory [15]. They are an iterative procedure based on a constant-size population (called 'chromosomes'). Each individual in the population describes a solution and new ones are produced by applying mutation or crossover operators. The crossover operator takes two individuals (called 'parents') of the old generation and combines their bits (called 'genomes') to produce one or more individuals (called 'offsprings'). The mutation operator has been introduced to prevent convergence to local optima, in that it randomly makes modifications on individual genomes. New sets of chromosomes representing the possible solution to the optimization problem are constantly generated through different generations and are evaluated for their suitability using a fitness function. This process often continues until an adequate solution is found [15].

A. Service Selection with Genetic Algorithms

In our approach, we use GAs in order to evaluate different combinations of services that satisfy the constraints defined in the BPMF and its corresponding feature model in order to optimize stakeholders' preferences. This section describes the adoption of each element of GA as follows:

1) Service Chromosome Encoding: An array encoding $(e_1, e_2, \ldots, e_{|A|})$ is used to represent a potential solution (i.e. one combination of services) as a chromosome. If the i^{th}

activity or any of its ancestors is optional, the set of possible values for element e_i is equal to $S_{a_i} \cup \{0\}$. If the value of e_i is 0, it means that the i^{th} activity is not included in the BPMF; otherwise, e_i represents an index in the array of the set of available services S_{a_i}. In case that the i^{th} activity is mandatory, the set of possible values for e_i is S_{a_i}.

2) Initial Population and the serviceTransform algorithm: Initial population is generated randomly, and hence, initial service sets may not be valid service combinations that conform to the optionality and integrity constraints defined by the feature model. For example, a chromosome $(s_{1(1)}, s_{2(1)}, s_{3(1)}, 0, \ldots, s_{|A|(1)})$ for the e-shop feature model shown in Fig. 1, which represents a set consisting of the first services available for each activity except for the *CreditCard-Validation* activity that is unselected, does not represent a valid set of services because the constraint *CreditCardPayment includes CreditCardValidation* is violated.

To solve the problem of generating invalid elements in the population (regarding variability and integrity constraints), we use the approach similar to [16] and define a *servicesTransform* algorithm for transforming randomly generated sets into valid service selections. The input data of the algorithm is an array $(e_1, e_2, \ldots, e_{|A|})$ (a chromosome from the population) and it consists of three major steps as follows:

Step 1. Based on the set of selected services, we identify the corresponding set of leaf-features F_L for each selected service. Starting from set F_L, the set of features is reached in the bottom-up manner from the leaves to the root feature.

Step 2. Starting from the set F_1, the valid set of features F_2 for the whole FM is selected by traversing to a leaf feature.

Step 3. Based on set F_2, the valid set of services S_{conf} is selected and it represents a valid configuration for the BPMF.

The serviceTransform algorithm uses a time counter to limit the time spent attempting to repair leaf-feature selections. Our approach is adopted for leaf-feature selection and the basic characteristics are the same as those of [16]. Furthermore, the serviceTransform ensures that the proposed adoption of GA produces always a valid configuration w.r.t. both configuration and behavioral properties of a BPMF by using the BPMF validation algorithm from [6].

To illustrate the *serviceTransform* algorithm, let us consider the feature model from Fig. 1(b) and the part of the BPMF presented in Fig. 1(b) which corresponds to features *Payment* and *Shipment*. For this purpose, let us consider *Advertising Management* and initial encoding array of services as $(s_{1(1)}, s_{2(1)}, s_{3(1)}, 0, s_{5(1)}, s_{6(1)}, s_{7(1)}, s_{8(1)}, s_{9(1)}, 0, ..., 0)$ for activities *IdentifyFederation* (A_1), *GatewayInterface* (A_2), *CreditCardPayment* (A_3), *DebitCardPayment* (A_4), *Email-Voicemail* (A_5), *PhoneFax* (A_6), *Mobile-basedNotification* (A_7), and *Shipment* (A_8). The steps performed based on the *serviceTransform* algorithm for developing a valid configuration are presented in Table II. *ServiceTransform* produces the following configuration $S_{conf} = \{s_{1(1)}, s_{2(1)}, s_{3(1)}, s_{5(1)}, s_{7(1)}, s_{4(2)}, s_{12(2)}\}$, which in addition to the initially selected services for activities *IdentifyFederation* (A_1), *GatewayInterface* (A_2), *CreditCardPayment* (A_3), *DebitCardPayment* (A_4), and *PhoneFax* (A_6) contains randomly selected services of available activities *CreditCardValidation* (A_9) and *SMS* (A_{10}) based on variability.

3) Fitness Evaluation: In order to evaluate the solutions and select the best one, a fitness function which represents the performance of each individual is needed [16]. To generate and select the set of services with the best performance, we need to optimize the overall function. There are a variety of studies regarding the definition of appropriate fitness functions [17], [18] and the major recommendation is to use fitness functions that penalize individuals that do not meet the problem constraints [16]. The main advantage of this approach is that we can incorporate any constraint into the fitness function, along with an appropriate penalty measure for it, and we can expect the GA to take this constraint into account during optimization.

1) The configuration goal is to find the configuration of services with the most preferable combination of qualifier tags. Therefore, we should choose the combination of services with a more preferable combination of qualifier tags. To include this important fact in our fitness function, we define the penalty factor as the relative distance of the most preferable combination of qualifier tags. That is, if the combination of services has the most preferable combination of qualifier tags, there is no penalty; but, the penalty value will increase when the combination becomes further away from the most preferable combination. To make the penalty factor independent of the values, we choose the relative ratio instead of the absolute distance between the ranks of the most preferable combination of qualifier tags $r^{[]}_{c_1,\ldots,c_n}$ and the ranks of qualifier tags of the running configuration of services, annotated with $r^{[]}_{c_1,c_n}(e_1, \ldots, e_{|A|})$.

2) An additional penalty factor is defined for structural constraints, which can be applied for special sub-processes for limiting the corresponding quality properties. This penalty factor is based on a hard constraint and its violation should directly eliminate the corresponding combination of services. The distance from constraint satisfaction can be defined as: $D(e_1, \ldots, e_{|A|}) = \sum_{i=1}^{l} cl_i(e_1, \ldots, e_{|A|}) \cdot y_i$ where $y_i = \begin{cases} 0, cl_i(e_1, \ldots, e_{|A|}) \leq u_i \\ 1, cl_i(e_1, \ldots, e_{|A|}) > u_i \end{cases}$.

If the weight for the penalty factor is low, there is the risk that individuals will not be discarded although they violate the constraints [16]. Therefore, we should penalize the constraints more heavily than the requirements, as we must respect the hard constraints in order to get a feasible set of services. In the literature [19], the penalty weight in a fitness function is dynamically increased with the number of generations. The fitness function is defined as:

$$Fitness(e_1, \ldots, e_{|A|}) =$$
$$\frac{1}{R(v_1(e_1,\ldots,e_{|A|}),\ldots,v_n(e_1,\ldots,e_{|A|})} + w_1(gen)D(e_1, \ldots, e_{|A|})$$
$$+w_2(gen)\frac{r^{[]}_{c_1,\ldots,c_n} - r^{[]}_{c_1,\ldots,c_n}(e_1,\ldots,e_{|A|})}{r^{[]}_{c_1,\ldots,c_n}} \tag{2}$$

Finally, we need to define a stopping criterion for the GA. One possible way is to set a fixed maximum number of iterations. Alternatively, it is possible to:

1) Iterate until both, all constraints are met (i.e. $D(e_1, \ldots, e_{|A|}) = 0$) and the obtained combination of services $(e_1, \ldots, e_{|A|})$ has the most preferable combination of qualifier tags. If this does not happen within the fixed maximum number of generations, then one of the following will hold:

serviceTransform.STEP1		serviceTransform.STEP2		serviceTransform.STEP3	
$F_L = \{A_1, ..., A_8\}$ $F1 = F_L$		$F_1 = \{A_1, ..., A_4, A_6, A_8, A_{11}, A_9, A_{12}, A_{13}\}$ $F_2 = F_1 \cup \{root\}$		$F_1 = \{A_1, ..., A_4, A_6, A_8, A_{11}, A_9, A_{12}, A_{13}, A_{14}, A_{15}, A_{10}\}$ $S_{conf} = \emptyset$	
s1.3a	$F_1 = F_1 \cup \{A_{11}\}$	s2.1	$F_2 = F_2 \cup \{A_{14}\}$	s3.1 A_1	$S_{conf} = S_{conf} \cup \{s_{1(1)}\}$
s1.3a	$F_1 = F_1 \cup \{A_9\}$	s2.1	$F_2 = F_2 \cup \{A_{15}\}$	s3.1 A_2	$S_{conf} = S_{conf} \cup \{s_{2(1)}\}$
s1.2	$F_1 = F_1 \cup \{A_{12}\}$	s2.1	$F_2 = F_2 \cup \{A_{10}\}$	s3.1 A_3	$S_{conf} = S_{conf} \cup \{s_{3(1)}\}$
s1.2	$F_1 = F_1 \setminus \{A_5\}$			s3.1 A_4	$S_{conf} = S_{conf} \cup \{s_{5(1)}\}$
s1.2	$F_1 = F_1 \setminus \{A_7\}$			s3.1 A_6	$S_{conf} = S_{conf} \cup \{s_{7(1)}\}$
s1.3a	$F_1 = F_1 \cup \{A13\}$			s3.2 A_9	$S_{conf} = S_{conf} \cup \{s_{4(2)}\}$
				s3.2 A_{10}	$S_{conf} = S_{conf} \cup \{s_{12(2)}\}$

TABLE II
THE DEMONSTRATION OF THE **SERVICETRANSFORM** ALGORITHM.

a) If the hard constraints are not satisfied, then no solution has been found;
b) If the hard constraints are satisfied, but the combination of services with the most preferable combination of qualifier tags is not found, then the solution is the combination of services with the lowest value of fitness function;

2) Once $D(g) = 0$ and the combination of services with the most preferable combination of qualifier tags is found, then the process is continued for a fixed number of iterations in order to reach lower values of fitness function. Alternatively, we iterate until the best fitness individual remains unchanged for a given number of iterations.

We can see that stopping criterion (1) ensures satisfaction of at least, the hard constraints and if possible, the stakeholders' requirements about preferable characteristics. Criterion (2) enables finding the most preferable solution based on the hard constraints and the stakeholders' requirements.

4) Crossover and Mutation: Because of the non-binary nature of our genomes, we use the n-point crossover operator. It chooses n random crossover points, splits along these points and glues the parts, alternating between parents. The random point mutation operator randomly chooses the position in the array $(e_1, \ldots, e_{|A|})$ and changes its value with randomly selected values. The mutation operator is preformed after crossover and as a result of both operations, invalid chromosomes might be generated. To handle this, we use previously introduced *serviceTransform* algorithm, as a reparation method which restores feasibility in the chromosome.

V. ANALYSIS OF THE PROPOSED APPROACH

As our approach uses GAs, it is necessary to analyze whether it is effective and efficient for the problem of BPMF configuration. We have chosen the *closeness to an optimal solution* as the criterion which is used to estimate the efficiency of our approach.

A. *Descriptive parameters for the* ConfBPMF *approach*

As the initial inputs for our approach are: 1) FM; 2) BPMF; 3) a set of available services per each activity in the BPMF; and 4) a set of requirements and hard constraints; thus, the independent variables are the following: 1a) the number of features in the FM; 1b) the percentage ratio of commonality and variability patterns in the FM; 2a) the number of activities in the BPMF; 2b) the percentage ratio of composition patterns in the BPMF; 3a) the number of services per each activity; 3b) the set of QoS properties of each available service; 4a) the set of concerns and qualifier tags; and 4b) the stakeholders' hard constraints and requirements about the most preferable characteristics of the whole family.

As previously mentioned, we have made some assumptions considering only well-structured business process models [14] and one-to-one mappings between the FM and BPMF. This allows for mappings between the percentage ratio values of commonality and variability (in FM) and composition patterns (in BPMF) as well as equality between the variables 1a and 2a. As stated formerly, the ***serviceTransform*** algorithm is one of the main parts of the adoption of GA which resolves the problem of generating invalid combination of services with respect to the optionality and integrity constraints defined by both models. With this in mind, we decided to analyze how the descriptive parameters which correspond to those constraints influence closeness to an optimal solution.

B. *Hypotheses*

Our main goal is to estimate differences between qualities of solutions as a measure of how close our proposed algorithm is to an optimal solution. Also, we analyze if the algorithm significantly depends on different patterns in both FMs and BPMFs. This is further refined into the following more specific and concrete hypotheses that we tested in our experiments:

H1. an increase or decrease in the percentage ratio of optional and mandatory features in the feature model will have significant impact on the optimality of the obtained solution.
H2. an increase or decrease in the percentage ratio of different variability patterns in the feature model will have significant impact on the optimality of the obtained solution.
H3. an increase or decrease in the percentage ratio of variability patterns in the BPMF will have significant impact on the optimality of the obtained solution.

For testing the hypotheses and making an estimation of the average distance to an optimal solution, we performed several experiments which are explained in the next section. Given the type of the collected data, we analyzed them with standard descriptive statistics (as reported in [20] to be a common practice) including mean (M) and standard deviation (SD) values. The hypotheses are tested by ANOVA to see whether significant difference can be observed as a result of changing the percentage ratio of optional features in FM (H1), and the percentage ratio of different patterns (H2 and H3).

C. *Experimental Setup*

As previously indicated, there are several input parameters in our approach and in order to parametrically change their values, the following generators are developed: 1) generator of FMs which takes as input the number of features and percentage ratios of variability and commonality patterns; 2) generator of BPMFs, which takes as input the FM generator parameters, the distribution of corresponding patterns in the BPMF and the stakeholders' requirements regarding concerns and qualifier tags. To the best of our knowledge, the only FM generator is developed in [16] and we developed the others and implemented *ConfBPMF* as an Eclipse plug-in. Also, we implemented the brute-force algorithm in order to obtain the optimal solution and compared it with the solution of *ConfBPMF*. In our previous empirical research of feature models [21], we found that the number of features in the available feature models was in the range between 14 and 287 where $M = 76.86$ and $SD = 96.25$. Accordingly, we preformed our experiments with FMs and BPMFs of the size equal to this empirically found mean value. Our previous work also suggests that a two-layered structure should consist of at most 7 qualifier tags, as this number is manageable by human users [13]. The maximum number of available services per activity was set to 100 and their values of QoS characteristics are generated randomly.

For estimating the average distance between solutions, each input parameter is randomly generated. Further, for testing our hypotheses, simulations with different percentage ratio values of different patterns in both models are performed as follows. We considered the following percentage ratio values of optional features in the FM: 25%, 50%, 75% and 100% (these are referred to as groups 1-4, respectively). Analogously, we considered the following distributions of patterns in FM: 33% (equal percentage for AND, OR and XOR feature gropups – group 5), random percentages in case of only two patterns for each of the combinations (AND-OR, AND-XOR, OR-XOR – groups 6-8), and 100% (only one pattern in the whole model – groups 9-11). Based on the one to one mapping between FM and BPMF, most of the patterns in BPMF are determined by FMs except sequential and parallel-AND patterns, which correspond to the AND group in FMs. Thus, our simulations have percentage ratio values of sequential and parallel-AND patterns in BPMF: 25%, 50%, 75% and 100% (groups 12-15). Each simulation is preformed 1000 times and the collected data are further analyzed.

D. Results

As the first part of our analysis, the mean value of the obtained relative distances in the configurations is calculated and is equal to 10.23% ($SD = 0.987\%$). Thus, the optimality of our approach is around 90%, i.e., the result of our approach has approximately 10% lower estimated quality as compared to the optimal solution.

In order to analyze the influence of distributions on optional features in the FM, we considered that our data was divided into four groups (1–4) according to the percentage ratio of optional and mandatory features in the model. The description statistics for groups 1–4 are as follows: 1 – $M = 9.87\%, SD = 0.64$; 2 – $M = 10.13\%, SD = 0.72$; 3 – $M = 9.98\%, SD = 0.91$; 4 – $M = 10.35\%, SD = 0.93$. As the collected data were not normally distributed, a one way ANOVA test was used over log-transformed data to compare the means of the dependent variable. The results show a non-significant difference between groups 1-4: $F(7, 784) = 23.33, p = 0.634$. Thus, hypothesis H1 is not accepted and we can conclude that the distribution of optional features does not have a significant impact on the optimality of our approach. This finding is important, as optional features determine how many possible configurations the BPMF can have; that is, any increase in the configuration space of the BPMF, driven by optional features, does not have a significant impact on the optimality of our algorithm.

Similarly, in order to analyze the influence of variability patterns on the optimality of our solution, we analyze if there exist significant differences between groups 5–11 (with different percentage ratio in the FM). The descriptive statistics for groups 5–11 were as follows: 5 – $M = 9.98\%, SD = 0.96$; 6 – $M = 10.25\%, SD = 0.92$; 7 – $M = 10.51\%, SD = 0.98$; 8 – $M = 10.43\%, SD = 0.95$; 9 – $M = 10.27\%, SD = 0.96$; 10 – $M = 10.37\%, SD = 0.93$; 11 – $M = 10.35\%, SD = 0.96$. As the collected data were not normally distributed, a one way ANOVA test was used over log-transformed data to compare the means of the dependent variable. The results show a significant difference in distances to the optimal solution in case of different patterns in FMs ($F(7, 2460) = 69.53, p = 0.000$). The Tukey post-hoc test revealed that there was a significant difference only between group 5 and groups 9–11. Thus, hypothesis H2 is accepted; that is, the higher percentage of variability patterns in FMs decreases the optimality of our approach.

To analyze the influence of composition patterns on the optimality of our solution, we analyze if a significant differences between groups 12–15 exists (with different distributions of patterns in the BPMF). The descriptive statistics for groups 12–15 were as follows: 12 – $M = 9.91\%, SD = 0.93$; 13 – $M = 10.56\%, SD = 0.99$; 14 – $M = 10.75\%, SD = 0.97$; and 15 – $M = 10.32\%, SD = 1.03$. The ANOVA test showed that there was no significant difference between groups 12-15 (F(3,572)=33.33, p=0.580). Hence, hypothesis H3 is not accepted. This finding is important if we have in mind that most of the patterns in BPMF are determined by FMs except sequential and parallel-AND patterns, and this hypothesis shows that optimality of our approach does not depend on their distributions.

To summarize the results of the experiments, we can conclude that variability patterns as represented in the FM are the source for significant influence on the obtained results and that affirms only hypothesis H2 but at the same time undermines the others. The main reason is that FMs are the main source of variability, i.e., they are essentially developed to capture variability; therefore, it is expected from them to exhibit such influence on the experiments. Hence, the variability in FMs should be treated as the main source of deviance from an optimal solution. Also although the effectiveness of an SPL approach directly depends on how well feature variability is implemented and managed throughout the development lifecycle, the obtained mean value of 90% as compared to the optimal solution indicates that our approach is still efficient and effective enough in all cases and can successfully overcome the dependence on modeling characteristics.

VI. Related Work

Configurable business processes and workflow variants have been extensively studied in the context of configurable workflow models and configurable event-driven process chains [22]. They only provide a partial solution. That work only looks at variation modeling, based on a specific language, and does not support mechanisms for automatic or semi-automatic configuration and service selection based on QoS characteristics.

QoS-driven service selection in service-oriented applications has been studied by many researchers in the recent years. GA-based solutions are widely adopted to find near-optimal solutions more efficiently. Canfora et al. [23] proposed a genetic approach to QoS-aware service selection and composition. Their approach utilizes a one-dimensional chromosome-encoded method to describe the combination of services. They select the genome with a fitness function containing a penalty factor. Likewise, Gao et al. [24] developed a tree-coded GA to support service selection and composition. In [25], Jaeger and Mühl compare GA and Hill-Climbing (HC) approaches. Their results indicate that the GA-based approach outperforms HC in terms of the overall QoS reached and achieving near optimal solutions. In [23], Canfora et al. present empirical data to assess the performance of GA-based methods against ILP solutions. Conducted experiments show that GA is more scalable while being slower than ILP solutions. GA can scale-up when the number of concrete services per abstract service increases. Furthermore, the authors of [26] have claimed that numerous heuristic techniques for many NP-hard problems (including ILP) cannot directly support SPL feature selection optimization because they are not designed to handle various structural and semantic constraints defined in the FM. It becomes even more complex as the BPMF configuration problem includes the additional task of configuration of features (i.e. activities) which should be previously selected to obtain optimal final configuration.

The large majority of existing work based on GAs (e.g., [24], [25], [23]) consider simple weighting schema to reflect stakeholders' preferences towards the required quality aspects. For instance, most of them use the Simple Additive Weighting (SAW) to define fitness functions. In consequence, they are not applicable to real-world service selection problems due to the fact that objectives often have different value ranges and priorities. Whereas, our proposed approach considers a complex weighting mechanism for the ranking and prioritization of stakeholders' concerns, which are reflected in the fitness function in order to address near real-world problem instances. To the best of our knowledge, none of the existing approaches perform service selection in presence of *variability* within composition specifications, as we have discussed in the context of business process families (and SPLE).

VII. Conclusion

The paper has presented a novel approach for configuration of business process families based on the use of genetic algorithms to maximize the level of satisfaction of quality requirements. While the concept of business process families and their configuration are already known in the literature [4], to our knowledge, this is the first approach that considers *quality-driven* configuration of business process families. In our proposal, not only is the selection of process activities performed, but also the selection of services with the most suitable quality attributes for each activity is achieved. In our future research, we will empirically investigate the impact of other structural elements of business process families (e.g., constraints on feature models) on the optimality of final solution. We will also (empirically) compare the proposed genetic algorithm-based approach to alternatives based on integer linear programming [5].

References

[1] M. L. Rosa, M. Dumas, A. H. M. ter Hofstede, and J. Mendling, "Configurable multi-perspective business process models," *Inf. Syst.*, vol. 36, no. 2, pp. 313–340, 2011.

[2] M. Rosemann and W. M. P. van der Aalst, "A configurable reference modelling language," *Inf. Syst.*, vol. 32, no. 1, pp. 1–23, 2007.

[3] T. Curran, G. Keller, and A. Ladd, *SAP R/3 business blueprint: understanding the business process reference model*. Prentice-Hall, Inc., 1998.

[4] M. L. Rosa, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede, "Questionnaire-based variability modeling for system configuration," *Software & System Modeling*, vol. 8, no. 2, pp. 251–274, 2009.

[5] B. Mohabbati, M. Hatala, D. Gašević, M. Asadi, and M. Bošković, "Development and configuration of service-oriented systems families," in *SAC'11*, 2011, pp. 1606–1613.

[6] G. Gröner, C. Wende, M. Bosković, F. S. Parreiras, T. Walter, F. Heidenreich, D. Gašević, and S. Staab, "Validation of families of business processes," in *CAiSE*, 2011, pp. 551–565.

[7] L. Zeng, B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[8] T. Saaty, *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. New york: McGraw-Hill, 1980.

[9] K. Czarnecki, "Mapping features to models: A template approach based on superimposed variants," in *Proc. GPCE2005*, 2005, pp. 422–437.

[10] J. Vanhatalo, H. Völzer, and F. Leymann, "Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition," in *Service-Oriented Computing - ICSOC*, ser. LNCS, vol. 4749. Springer, 2007, pp. 43–55.

[11] E. Bagheri, T. D. Noia, D. Gasevic, and A. Ragone, "Formalizing interactive staged feature model configuration," *J. of Software Maintenance and Evolution: Research and Practice*, 2011.

[12] I. Sommerville and P. Sawyer, "Viewpoints: principles, problems and a practical approach to requirements engineering," *Annals of Software Engineering*, vol. 3, no. 1, pp. 101–130, 1997.

[13] I. Ognjanovic, D. Gasevic, E. Bagheri, and M. Asadi, "Conditional preferences in software stakeholders' judgments," in *SAC'11*, 2011, pp. 683–690.

[14] B. Mohabbati, D. Gašević, M. Hatala, M. Asadi, E. Bagheri, and M. Bošković, "A quality aggregation model for service-oriented software product lines based on variability and composition patterns," in *ICSOC*, 2011, pp. 436–451.

[15] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[16] J. Guo, J. White, G. Wang, J. Li, and Y. Wang, "A genetic algorithm for optimized feature selection with resource constraints in software product lines," *J. on Systems and Software*, vol. 84, pp. 2208–2221, 2011.

[17] U. Bhowan, M. Johnston, and M. Zhang, "Developing new fitness functions in genetic programming for classification with unbalanced data," *Trans. Syst. Man Cybern B Cybern*, in press.

[18] W. Fan, E. A. Fox, P. Pathak, and H. Wu, "The effects of fitness functions on genetic programming-based ranking discovery for web search: Research articles," *J. Am. Soc. Inf. Sci. Tech.*, vol. 55, pp. 628–636, 2004.

[19] M. H. Ghiasi, B. Dehghan-Manshadi, and A. Abedian, "Effects of a linear-exponential penalty function on the gas efficiency in optimization of a laminated composite panel," *Int. J. of Computational Intelligent*, vol. 2, pp. 5–11, 2005.

[20] N. Blaikie, *Analysis of Quantitative Data: From Description to Explanation*. SAGE Publications Ltd, 2093.

[21] E. Bagheri and D. Gasevic, "Assessing the maintainability of software product line feature models using structural metrics," *Software Quality J.*, vol. 19, no. 3, pp. 579–612, 2011.

[22] W. M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, and M. H. Jansen-Vullers, "Configurable process models as a basis for reference modeling," in *Business Process Management Workshops*, 2005, pp. 512–518.

[23] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for qos-aware service composition based on genetic algorithms," in *Proc. 2005 Conf. Genetic and Evolutionary Computation*, 2005, pp. 1069–1075.

[24] C. Gao, M. Cai, and H. Chen, "Qos-aware service composition based on tree-coded genetic algorithm," in *Proc. 31st Annual Int. Computer Software and Applications Conference*, 2007, pp. 361–367.

[25] M. Jaeger and G. Mühl, "QoS-based selection of services: The implementation of a genetic algorithm," in *KiVS 2007*, 2007, pp. 359–370.

[26] J. Guo, J. White, G. Wang, J. Li, and Y. Wang, "A genetic algorithm for optimized feature selection with resource constraints in software product lines," *Journal of Systems and Software*, vol. 84, no. 12, pp. 2208 – 2221, 2011.