Using Goals and Customizable Services to Improve Adaptability of Process-based Service Compositions

Joseph Gillain, Stéphane Faulkner, Ivan J. Jureta
PReCISE Research Center
University of Namur
{jgillain,sfaulkner,ijureta}@fundp.ac.be

Monique Snoeck
Department of Decision Sciences Information Management
University of Leuven
monique.snoeck@econ.kuleuven.be

***Abstract*—When implementing (semi-)automatic business processes with services, engineers are facing two sources of variability. One source of variability are alternative refinements and decompositions of requirements. The other source of variability is that various (combinations of) services can be used to satisfy the same requirements. We suggest a method based on the use of a goal model and customizable services able to exploit these variabilities to design executable business process. This method improves the adaptability of the business process at runtime. We illustrate the contribution of our method with an example.**

I. Introduction

Although Service-Oriented Computing (SOC) promises to leverage numerous advantages when implementing business processes, the paradigm is still facing some challenges. Service providers have to meet many varieties of customers with sometimes slight differences in their business process requirements. Moreover those requirements are changing depending on some contextual properties such as customer preferences. While providing distinct and independent services to each particular situation would reduce promised advantages such as reuse, supplying a single service encompassing functionalities of each situation would result in a large and complex service description most of which would be useless for a given customer to meet both reusability and context specificity. Service providers should supply customizable services adaptable to different situations.

Recently, more and more attention has been paid to service variability, that is the "*ability of the service to be efficiently extended, changed, customized, or configured for use in a particular context.*" [1]. Variable services can be used to improve service adaptability, i.e. the ability of a service to respond to changing circumstances. In this context, techniques such as commonality and variability analysis or feature modeling have been applied to services. They result in the definition of customizable services which are services *"whose runtime customization by a customer will result in a particular service variant matching the customer's requirement"* [2].

However, this approach implies a configuration process (i.e. to resolve variation points) which raises different difficulties. Firstly, although the use of features can reduce the number of services in repositories, customizable services can be composed of numerous features eventually resulting only in a shift of complexity to another level. Secondly, current methods for customizing services (e.g. [3], [4]) do not usually consider that business goals can be satisfied by alternative business processes resulting in the consideration of a subset of all the possibilities. Finally, deciding about which features will be part of the selected service variants taking part in the executable business process requires the definition of decision criteria for comparing services from a functionality perspective. It is necessary to measure their contributions to the satisfaction of the business goals.

This paper provides a method to semi-automatically generate a service composition and select its composing services at runtime. The composing services are derived from customizable services. The framework of customizable services we use was suggested by Nguyen et al. [2]. The method consists of 5 steps: description of a goal model, selection of customizable services to be part of the service composition, mapping between business goals and those service feature descriptions, resolution of a feature-extended goal model and finally, selection of the service variants. The innovation of this approach is twofold. Firstly, it provides a way to handle simultaneously the complexity of two different sources of variability: there exist different business process alternatives to satisfy the business goals and there exist different services that can implement a particular business process. Secondly, it improves the adaptability of the business process execution by selecting at runtime which are the service variants to be selected and how they will be composed given the particular execution context.

The remainder of this paper is structured as follows. After providing an overview of our method in Sect.II,we discuss how to describe business process goals in Sect.III. The next section presents the concept of customizable service and describes how to map service features and business process goals. Service variant identification and selection are discussed in Sect.V. We then present related work in Sect.VI and finally, we conclude in Sect.VII.

II. Overview of the method

In this section we provide an overview of our method. It consists of five steps, each of them resulting in artifacts to use in the following steps. Final artifacts are the BPEL specification of the service composition and the WSDL of the selected service variants from the partner services. The three first steps are design-time steps while the last two steps are executed at run-time for each execution of the business process. Fig.1 depicts all steps and each artifact.

The first step of the method is the drawing up of business

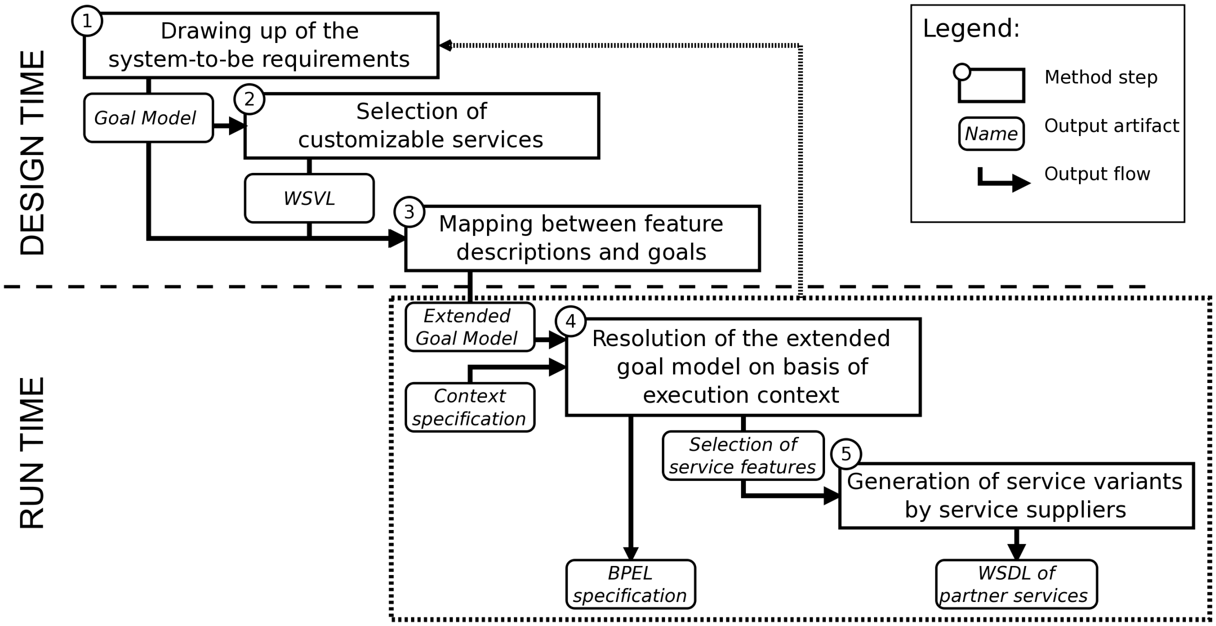

DESIGN TIME
RUN TIME
1
Drawing up of the system-to-be requirements
Goal Model
2
Selection of customizable services
WSVL
3
Mapping between feature descriptions and goals
Extended Goal Model
4
Resolution of the extended goal model on basis of execution context
Context specification
Selection of service features
5
Generation of service variants by service suppliers
BPEL specification
WSDL of partner services
Legend:
Method step
Name
Output artifact
Output flow

Fig. 1. Method to select a set of service configurations

process goals and softgoals. It consists in gathering and refining the goals of the business process into a set of business tasks and domain assumptions. This goal model will be used later as basis to generate the executable business process specification. The use of a goal model instead of a common business process model language such as BPMN is motivated by the fact that a goal model allows to consider alternative generation of BPEL specifications for a specific context [5], [6].

Once the goal model is defined, we select a set of customizable services able to potentially meet goal satisfactions. Those customizable services are described with the language WSVL (Web Service Variability description Language) suggested by Nguyen et al. [2]. In this language, customizable services consists of a *Service Description* (i.e. the complete capability of the customizable service), the *Feature Description* (i.e. the description of the variability of the customizable service in terms of features) and the *Mapping Description* (i.e. the mapping between the service features and the service capabilities). Our framework allows to compare several customizable services which are candidate for the service composition to-be. E.g., if we need some payment services in order to satisfy business goals, different customizable payment services can be taken into account. The selection will be performed at run-time depending on execution context as well as some decision criteria defined in the following steps.

The third step of our method consists in mapping the service features to the tasks of the goal model. It requires as inputs both the goal model and the *Feature Description* of the selected customizable services. During this mapping, engineers have to carefully identify the set of assumptions establishing conditions of business tasks realizations. They can be classified into four types of assumptions: *task realizations assumptions*, *contextual properties*, *goal-feature interactions* and *cross-services interactions*. This step closes the design time and results in an feature-extended goal model.

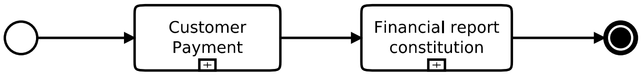


Fig. 2. The coarse-grained business process of the financial example

At run-time, when the execution of the business process is triggered, the BPEL code is generated on the basis of the execution context (i.e. which contextual properties and domains assumptions are verified). Then automatic reasoning methods are applied on the feature-extended goal model in order to identify sets of service features which are able to realize the tasks of the generated service composition (i.e. step 4). The selected features are send back to service providers which can dynamically generate the service variants that the service composition have to invoke during its execution (i.e. step 5). Eventually, the change of requirements due to new experience with selected service variants can result in the adaptation of the goal model. This is modeled with the feedback loop dotted arrow in Fig 1.

The remainder of this paper outlines the proposed method in more detail and uses an example about developing a financial service-oriented application.

III. Goal Model

In the first step of the method, we focus on the business process where customers should be provided with an analysis on financial assets after reception of a payment. The coarse-grained business process is depicted in Fig.2.

A business process (BP) is a sequence of activities an organization has to perform to achieve a particular business goal [7]. Depending on contextual properties, these business goals can change as well as the conditions to achieve them. Moreover, the particular execution of the business process will depend on runtime properties such as customer preferences (e.g. reliability, performance...), service availability and so on. Although goals are central in the definition of a business process, relatively few approaches explicitly deal directly with business goals and their contextual satisfaction. Current BP modeling techniques, such as BPMN, describe the sequences of activities and synchronization in processes, but do not model the business goals that the processes achieve. On the contrary, the use of goal models allows to specify and select between alternative activities and alternative sequencing. Lapouchnian et al. described an approach to capture alternative business process specifications satisfying a common set of goals [5]. Their approach can be used to (semi-)automatically generate executable business process (e.g. in BPEL). Moreover, Ali et al. developed a goal-oriented framework improving the self-contextualization of systems [6]. In this section, we introduce goal models and the enrichments provided by both previously cited work. These concepts are illustrated by our financial example.

Our method uses the requirement modeling language Techne [8]. It is a formal propositional language without a graphical representation (although for convenience, we provided a graphical representation of the goal model in Fig. 3). We choose this formal requirement modeling language for three main reasons. First, it is based on a non-monotonic propositional logic which allows automatic reasoning. Secondly, since it is based on the CORE ontology for requirements [9], it is compatible with other goal-oriented requirement modeling languages such as i* or KAOS. Finally, the capability to model domain assumptions allows the self-contextualization of the system.

In Techne, we assume that requirements are either atomic proposition (p, q, $r\ldots$) or well-formed formulas (wffs) (φ, ψ, $\gamma\ldots$) which are categorized according to the core ontology for requirements engineering [9]. A requirement is

g: a goal if the proposition expresses a desire, i.e., it states a property that we want to see holding; e.g., $\mathbf{g}(p_1)$ “Financial reports are provided to customers” at the bottom right in Fig. 3;

t: a task if the proposition says what to do; e.g., at the bottom left in Fig. 3, $\mathbf{t}(p_{16})$ “Execute an analysis” is a task;

q: a quality constraint if the proposition places a constraint on desirable values of a quantifiable property; e.g. $\mathbf{q}(p_8)$ states that“more than 1 asset type is provided” (Fig. 3, top left);

s: a softgoal if the proposition refers to a desirable value of a property in such a way that it is not possible to identify exactly which value, or range of values of that property it refers to; e.g., in Fig. 3, $\mathbf{s}(p_{29})$ “Interface is userfriendly” is a softgoal;

k: a domain assumption if the proposition states a belief of stakeholders; e.g., in Fig. 3 (bottom left), $\mathbf{k}(p_{17})$ is a domain assumption over the proposition “employees are allocated to the execution of analysis”. $\mathbf{k}(\varphi) \equiv \mathbf{k}((\mathbf{t}(p_{16}) \wedge \mathbf{k}(p_{17})) \rightarrow \mathbf{g}(p_{14}))$ is an example of a domain assumption that is a wff rather than an atomic proposition;

A requirement can be mandatory, optional, or neither. $\mathbf{g}(p_1)^{\mathrm{M}}$ “financial reports are provided to customers” is a mandatory goal, meaning that it must be satisfied. Optional goals are goals for which it is desirable that they are satisfied, but we will still accept a business process which fails to satisfy them. Triangles I in Fig. 3 refers to the inference relation where incoming arrows are linked to the premises and the outcoming arrows to the conclusion: for example, an inference node shows that we can deduce $\mathbf{g}(p_{14})$ from $\mathbf{t}(p_{16})$, $\mathbf{k}(p_{17})$, and the wff $\mathbf{k}(\psi) \equiv \mathbf{k}((\mathbf{t}(p_{16}) \wedge \mathbf{k}(p_{17})) \rightarrow \mathbf{g}(p_{14}))$, since *modus ponens* is allowed in Techne. Domain assumptions consisting of wffs such as $\mathbf{k}(\varphi)$ can then be considered as representing operationalization of goals by tasks. In other cases, such as $\mathbf{k}(\psi) \equiv \mathbf{k}(\mathbf{g}(p_2) \wedge \mathbf{g}(p_3) \rightarrow \mathbf{g}(p_1))$, a wff represents a goal refinement. In this last example, p_1 is refined by an AND-link in two sub-goals: p_2 “financial reports are constituted” and p_3 “analysis is paid.” All goals are eventually refined in tasks and domain assumptions.

Inconsistent requirements are in a conflict relation C, e.g., we assume in Fig. 3 (top, middle) that $\mathbf{g}(p_{18})$ cannot be satisfied together with $\mathbf{t}(p_{26})$, and this is represented as the assumption $\mathbf{k}(\rho) \equiv \mathbf{k}(\mathbf{g}(p_{18}) \wedge \mathbf{t}(p_{26}) \rightarrow \perp)$. If we can deduce both, then we will also conclude logical inconsistency. Notification of market events requires that customers subscribe to the service. Then the single payment method is in conflict with this goal.

A. Self-contextualization of business process

The whole set of domain assumptions K, goals G, quality constraints Q, softgoals S and tasks T forms an r-net (inferences and conflicts being included as wffs in K).

Definition 1 (R-net $\mathcal{R}$): An r-net is the tuple $\mathcal{R} = (K, G, Q, S, T)$, where K, G, Q, S, T are respectively the sets of domain assumptions, goals, quality constraints, softgoals and tasks that were elicited from the stakeholders of the system-to-be or otherwise obtained during requirements engineering.

By applying goal models to business process specification, we can reformulate the business process requirement problem as consisting in finding a set of tasks T^* satisfying business goals G, softgoals S and quality constraints Q given a set of satisfied domain assumptions K^* , formally $K^*, T^*, \mid\sim G, Q, S$ [9]. Domain assumptions set the stage for the execution of tasks, as they state conditions in which the tasks need to be executed. This reformulation highlights the crucial role played by the set K for the self-contextualization of the business process executions. Depending on which domain assumptions are confirmed at run-time, a particular business process specification should be generated in order to satisfy G,Q and S.

B. Generation of executable business process from goal model

In order to model the control flow of the business process, Techne needs to be enriched with the annotations suggested by Lapouchnian et al. Some example of the annotations are given below. For more information (e.g. for conditions, loops, event handlers..) we refer to [5].

- Parallel (“$\|$”) and sequence (“;”) annotations can be used with AND-decomposed goals to specify whether

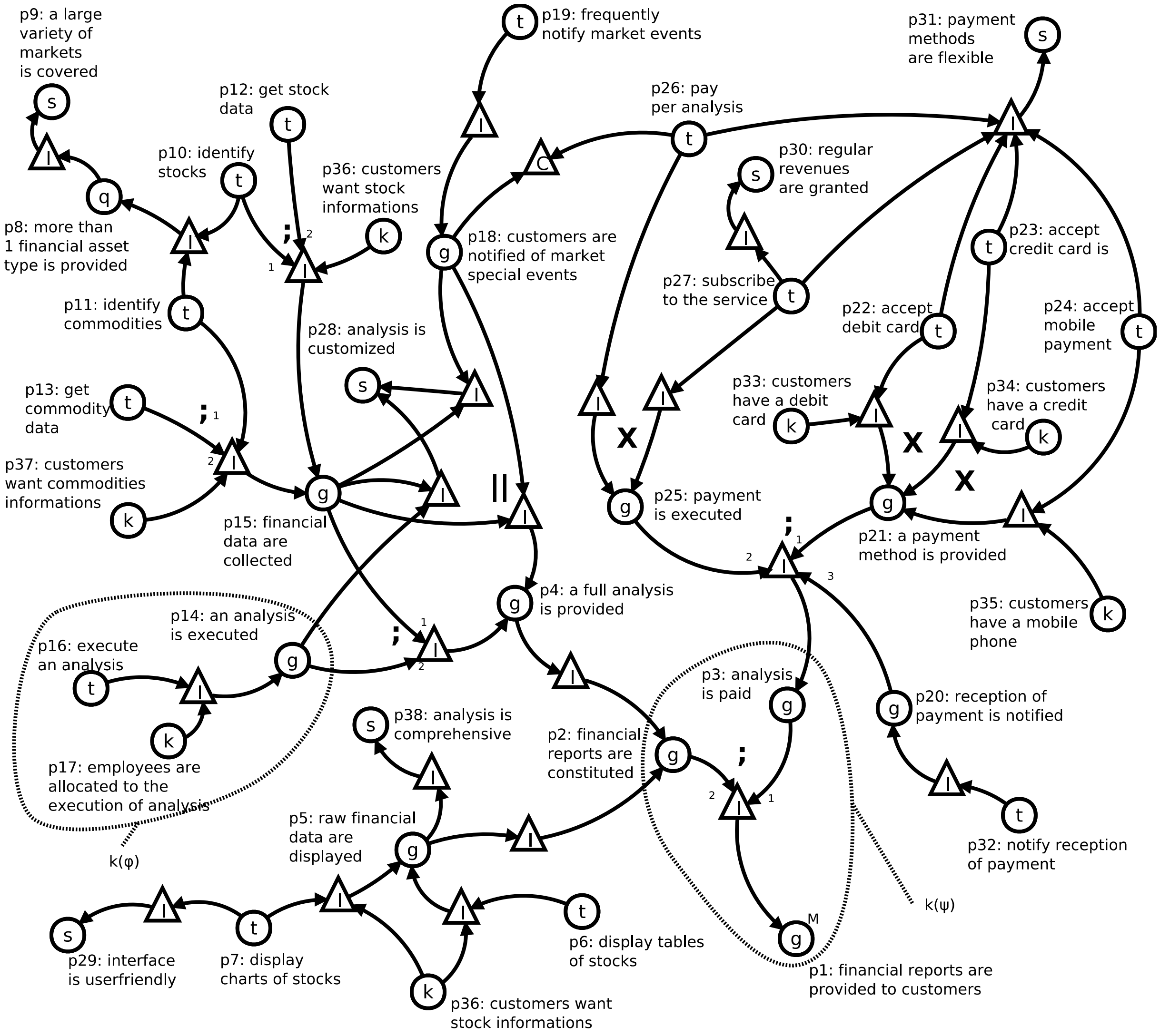

p9: a large variety of markets is covered
s
p12: get stock data
t
p19: frequently notify market events
t
p31: payment methods are flexible
s
p26: pay per analysis
t
p10: identify stocks
t
p36: customers want stock informations
p30: regular revenues are granted
s
C
q
p8: more than 1 financial asset type is provided
;
k
g
p18: customers are notified of market special events
p23: accept credit card is
t
p11: identify commodities
t
p27: subscribe to the service
t
p22: accept debit card
t
p24: accept mobile payment
t
p28: analysis is customized
s
p13: get commodity data
t
;
p33: customers have a debit card
k
p34: customers have a credit card
k
X
X
X
p37: customers want commodities informations
k
g
||
g
p25: payment is executed
g
p21: a payment method is provided
p15: financial data are collected
;
p4: a full analysis is provided
g
p35: customers have a mobile phone
k
p14: an analysis is executed
p16: execute an analysis
t
g
;
p3: analysis is paid
g
p20: reception of payment is notified
g
k
p17: employees are allocated to the execution of analysis
p38: analysis is comprehensive
s
p2: financial reports are constituted
g
;
k(φ)
p5: raw financial data are displayed
g
p32: notify reception of payment
t
k(ψ)
t
g
M
s
t
p29: interface is userfriendly
p7: display charts of stocks
p6: display tables of stocks
k
p36: customers want stock informations
p1: financial reports are provided to customers
1
2
3

Fig. 3. Goal model of our business process example - Dotted ellipses are examples of graphical representation of domain assumption over a wff

or not the subgoals are to be achieved in a temporal order. For example, customer payment $\mathbf{g}(p_3)$ and financial data providing $\mathbf{g}(p_2)$ have to be done sequentially in the process.

- By default, in goal models, OR decompositions are inclusive. Exclusive OR decompositions are marked with the "X" annotation.

These enrichment are used to automatically generate an executable business process specification in BPEL but it does not affect other reasoning methods of Techne.

IV. Customizable Services and Features Mapping

The second and third steps of our method consist in selecting several customizable services and then map their features to tasks of the business process. In this section we focus on the mapping step. For the identification process of customizable services, we refer to existing procedures such as the one proposed in [10] (see also related work section).

To illustrate our approach, we base our example on services provided by xignite.com and Amazon and we model them as two customizable services. xignite.com is a provider of on-demand financial market data. Its services cover several markets (stocks, precious metals, energy...) and provide different operations such as real-time quotation, delayed prices, historical data and so on. For payment services we focus on the Amazon Flexible Payment Service and consider it as a 3-variant customizable service (`FPS`, `MPS` and `Aggregated Payments`)[1].

Since feature modeling has proved to be a common and convenient way to document variability in software product lines, it is not surprising that recent research on service variability has focused on the adaption of these techniques. A service feature f is an increment in service functionality [11]. In this paper, we focus on the logical expression of customizable services as defined in Nguyen et al.'s framework. For

[1] http://aws.amazon.com/fr/fps/

more technical information on how to express a customizable service with WSVL see [2].

Definition 2 (Customizable Service): A customizable service $\mathcal{S}$ is a triple (SC, FD, FM) where:

- SC is the comprehensive set of Service Capabilities,
- FD is the Features Description,
- FM is the Feature Mapping

Feature descriptions (FDs) model relationships and constraints between all the features of a customizable service specifying valid feature compositions (i.e. service variants). There are several types of features: Mandatory, Optional, Alternative and Or [12]. Graphical representations (i.e. feature diagrams) are shown in the legend of Fig. 4. These feature relationships can be model as propositional formula [13]. For example, given a parent feature f_1 and a child feature f_2, Mandatory and Optional relationships are $f_1 \rightarrow f_2$ and $f_2 \rightarrow f_1$. In addition to feature types, feature diagrams can include graphical or textual constraints. They represent relationships among features that cannot be captured by feature decompositions. As both feature types and feature constraints can be represented using propositional formulas, we consider that they constitute a whole set of constraints. Fig.4 depicts the comprehensive Feature Description as a feature diagram for the xignite and Amazon.com customizable services. Formally, we define the feature description and the service variant as:

Definition 3 (Feature Description): A feature description is a tuple $FD = (F, \Phi)$ where:

- F is a set of features f_i,
- Φ is a conjunction of propositional formulas ϕ describing constraints on those features, i.e. Mandatory, Optional, Alternative, Or, Mutual exclusive and Require relations.

Definition 4 (Service Variant): A service variant sv derived from a customizable service $\mathcal{S} = (SC, FD, FM)$ is the subset $sc \subseteq SC$ such that $\mathsf{feat}(sc, FM) \models \Phi$.

where $\mathsf{feat}(sc, FM)$ is a function returning the corresponding features of a set of service capabilities. For example, a service variant for the xignite service could be the set of features $s_a = \{f_1, f_2, f_3, f_5, f_7, f_9, f_{13}, f_{15}, f_{17}, f_{18}, f_{19}, f_{20}\}$. It corresponds to the service `XigniteMetals`[2].

Once customizable services have been identified, business analysts and engineers have to map features to goal model tasks. This mapping, noted K', will be used to identify service variants S able to implement the business process tasks. Formally, we can describe this mapping problem as finding K' and S such that $K', \mathsf{S} \mid\!\sim T$, where S is a set of service variants and K' is a new set of assumptions establishing, among others, conditions of business tasks realization, i.e. the mapping between the service features and the business tasks. Resolving this problem raises two questions: (1) Can we identify some particular categories in K' setting conditions for business tasks realization? (2) What is the relation between the business process (i.e. a particular sequence of tasks satisfying the business goals) and a concrete service composition (i.e. the executable business process and its partner service variants)?

A. Assumptions on Business Tasks Realization

K' captures assumptions about how service features can realize business process tasks present in the requirements problem of the business process. The comprehensive specification of K' for our example is given Tab.I. Since those assumptions will be used to extend the r-net, we take care to define new domain assumptions either as atomic propositions or as implications with a conjunction as the antecedent.

For example, $\mathbf{k}(\gamma_6) \equiv \mathbf{k}(f'_2 \wedge f'_7 \rightarrow \mathbf{t}(p_{26}))$ states that if features f'_2 "Pay" and f'_7 "Single Transaction" are belonging to the payment service variant, then this service can realize the task $\mathbf{t}(p_{26})$ "Pay per analysis". This kind of domain assumptions which describes conditions for the realization of tasks will be referred to as *task realization assumptions*, noted K_{T}. It is interesting to note that service features are generic functionalities and can participate in the realization of different tasks (for instance f_{17}). Obviously, the generic level of a feature is dependent of the feature granularity defined in the WSVL. On the other hand, a business task can be realized by different sets of features.

However, some business tasks do not always require specific service features to be executed. For example, consider the proposition p_{16} "execute an analysis". Although this proposition is labeled as a task, we can consider that it refers to the expectation that the stakeholders will have the intention to participate in some goal satisfaction without any involvement of the feature-composed system-to-be. Using the BPMN terminology, it consists of a human activity. Consequently the single domain assumption $\mathbf{k}(\mathbf{k}(p_{39}) \rightarrow \mathbf{t}(p_{16}))$ where p_{39} states "the analysis is performed by an employee" seems enough. Those domain assumptions form the set of *contextual assumptions*, noted, K_{C}.

Due to implementation constraints, some service features can have side effects on the softgoals satisfaction of a particular business process while such relation was not considered during the analysis of the goal model of the business process. An example is a goal model where the security is a softgoal s_j of the business process. At the goal modeling level, nothing suggests that the task t_i can threaten s_j. Consider now, that feature f_a can realize t_i. However, due to some implementation issues, f_a will be in conflict with s_j. Engineers will have to take this new information into account. We see that this relation is really dependent on the interaction of features and business process models. It is possible that another customizable service provides a feature f_b which is also able to realize t_i without conflicting with s_j. On the other side, f_a can realize a task t_k in another business process without being in conflict with the security requirement of this business process. This interaction can also appear with particular combinations of features. We model those sets of interaction as a conflict between a set of features, softgoals and quality constraints and we called it the set of *goal-feature interaction assumptions* K_{I}.

Finally, we can identify some feature interactions between features of different customizable services which will be part of the same service composition, e.g. incompatible communication protocols. We denote those *cross-service interactions* as K_{X}.

We can then formally define $K' = K_{\mathrm{T}} \cup K_{\mathrm{I}} \cup K_{\mathrm{C}} \cup K_{\mathrm{X}}$. This set of new assumptions is used to extend an r-net with

[2] http://www.xignite.com/xMetals.asmx?WSDL

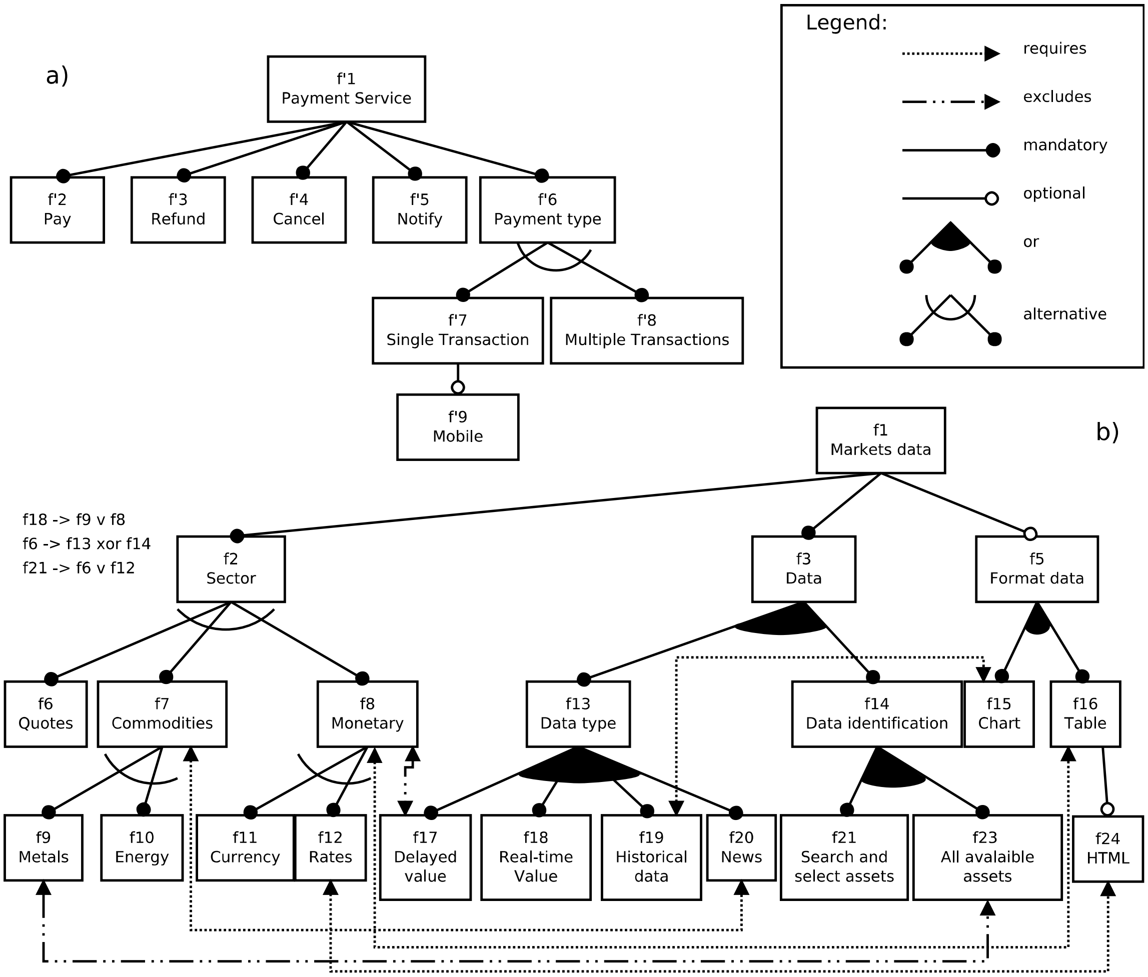


Fig. 4. Feature diagram of Amazon (a) and xignite.com (b) customizable services

information about service features and how they can realize business tasks. Those assumptions are used in order to form a feature-extended r-net.

Definition 5 (Feature-extended r-net ($\mathrm{r_f}$-net)): Given a set of feature descriptions $\mathcal{FD} = \{FD_1, ..., FD_n\}$ and a r-net $\mathcal{R}$, an r_f-net is defined as a tuple $\mathcal{R}_e = (\mathcal{R}, \mathcal{FD}, K')$ where K' is the set of feature related assumptions.

In Sect. III-A, we defined the business process specification problem as consisting in finding a set of tasks T satisfying some goals G, softgoals S and quality constraints Q given some context K, formally $K, T \mid\!\sim G, S, Q$. We then defined, in Section IV, the mapping problem as consisting in finding a set of service variants S as well as the conditions K' under which they realize the business tasks, formally $K', \mathsf{S} \mid\!\sim T$. One could then suggest that if those two relations are true, then we can conclude that $K, K', \mathsf{S} \mid\!\sim G, S, Q$, i.e. we have found the service variants able to satisfy the same goals as the tasks T did. This is however not always the case since some goal-feature interactions can block the satisfaction of some softgoals and quality constraints. Actually, services have per se some technical constraints which restrict the business goal satisfaction. They are modeled in our framework as a piece of information (K_I) which is added and restricts previous established conclusions on regarding goals, quality constraints, or softgoals satisfaction. This issue illustrates one reason for the need for non-monotonic reasoning which justified the use of Techne.

Another property of S is that it can realize different requirement solutions. Consequently, it is interesting to notice that the relationship between S and goal model solutions is a *m*n* relationship because a goal model solution can be implemented by different set of service variants and a set of service variants can implement different goal model solutions.

V. Service Variants Identification and Selection

In the last two sections, we discussed design-time related considerations. In this section, we focus on run-time issues of identifying and selection service variants.

At runtime, when the business process start is triggered, the execution context (i.e. verified K' and K) and customer preferences (e.g. which goals and softgoals are preferred) have to be specified. On the basis of this context, the BPEL specification is generated and all candidate solutions are identified thanks to Techne automatic reasoning methods [8].

Definition 6 (Candidate Solution in r_f-net): Given a $\mathrm{r_f}$-net $\mathcal{R}_f$, a tuple $s = (F^*, K'^*, K^*, \mathsf{SLA})$ is a candidate solution of a service composition where F^* is a set of service features

TABLE I. SET OF ASSUMPTIONS ON TASK REALIZATIONS

$\mathbf{k}(\gamma_1) \equiv \mathbf{k}(f_6 \wedge f_{15} \wedge f_{17} \wedge f_{23} \rightarrow \mathbf{t}(p_7))$	$\mathbf{k}(\gamma_8) \equiv \mathbf{k}(f'_5 \rightarrow \mathbf{t}(p_{32}))$
$\mathbf{k}(\gamma_2) \equiv \mathbf{k}(f_6 \wedge f_{16} \wedge f_{17} \wedge f_{23} \rightarrow \mathbf{t}(p_6))$	$\mathbf{k}(\gamma_9) \equiv \mathbf{k}(f'_6 \rightarrow \mathbf{t}(p_{22}))$
$\mathbf{k}(\gamma_3) \equiv \mathbf{k}(f_7 \wedge f_{21} \rightarrow \mathbf{t}(p_{11}))$	$\mathbf{k}(\gamma_{10}) \equiv \mathbf{k}(f'_6 \rightarrow \mathbf{t}(p_{23}))$
$\mathbf{k}(\gamma_4) \equiv \mathbf{k}(f_6 \wedge f_{21} \rightarrow \mathbf{t}(p_{10}))$	$\mathbf{k}(\gamma_{11}) \equiv \mathbf{k}(f'_9 \rightarrow \mathbf{t}(p_{24}))$
$\mathbf{k}(\gamma_5) \equiv \mathbf{k}(f_2 \wedge f_{20} \rightarrow \mathbf{t}(p_{19}))$	$\mathbf{k}(\gamma_{12}) \equiv \mathbf{k}(f_7 \wedge f_{17} \rightarrow \mathbf{t}(p_{13}))$
$\mathbf{k}(\gamma_6) \equiv \mathbf{k}(f'_2 \wedge f'_7 \rightarrow \mathbf{t}(p_{26}))$	$\mathbf{k}(\gamma_{13}) \equiv \mathbf{k}(f_6 \wedge f_{17} \rightarrow \mathbf{t}(p_{12}))$
$\mathbf{k}(\gamma_7) \equiv \mathbf{k}(f'_2 \wedge f'_8 \rightarrow \mathbf{t}(p_{27}))$	$\mathbf{k}(\gamma_{14}) \equiv \mathbf{k}(\mathbf{k}(p_{39}) \rightarrow \mathbf{t}(p_{16}))$

and K'^* and K^* are respectively the set of tasks realizations conditions and the set of domain assumptions, if:

- $F^* \in \mathcal{P}(\mathcal{F})$, where $\mathcal{F}$ is the union of all the service features of $\mathcal{FD}$,
- K'^*, K^* and F^* are not logically inconsistent,
- $K'^*, K^*, F^* \mid\!\sim G^*, Q^*, S^*$, where $G^* \subseteq G$, $Q^* \subseteq Q$ and $S^* \subseteq S$,
- G^*, Q^* and S^* include all mandatory goals, quality constraints and softgoals,
- F^* is a valid set of service variants, i.e. for each service variant sv_i described in F^*, it exists a $FD \in \mathcal{FD}$ such that $sv_i \models \Phi$ and $Phi \in FD$.
- K'^*, K^* and F^* are minimal, i.e. $\nexists \tilde{K}', \tilde{K} \in \mathcal{R}_f$ or $\nexists F' \subset F^*$ such that $\tilde{K}' \cup \tilde{K} \subset K'^* \cup K^*$ or F' is a valid set of service variants, and $\tilde{K}', \tilde{K}, F' \mid\!\sim G^*, Q^*, S^*$,
- $\mathsf{SLA} : F^* \rightarrow (Y_1 \times \ldots \times Y_m)$ associates a tuple of properties to F^*.

The service level agreement (SLA) function is used as additional decision criteria for selecting service variant satisfying the same set of goals, softgoals and quality constraints under the specified contextual conditions. The tuple dimension m of the SLA output depends on the number of factors used to evaluate the quality of a service variant. In our example, we used a simple case of SLA which returns a real number representing the cost of the service variants F, i.e. $\mathsf{SLA} : F \rightarrow \mathbb{R}$. It is illustrated in Fig. II.

In order to illustrate our approach, consider the identification step for a runtime context in which the domain assumptions $\mathbf{k}(p_{34})$ "customers have a credit card" and $\mathbf{k}(p_{36})$ "customers want stock information" would be verified. Then several service variants can be identified. For the service partner Amazon, two service variant candidates are identified: $sv_{A1} = \{f'_1, f'_2, f'_3, f'_4, f'_5, f'_6, f'_7\}$ and $sv_{A2} = \{f'_1, f'_2, f'_3, f'_4, f'_5, f'_6, f'_8\}$. Regarding the financial service variants, there are three service variant candidates $sv_{x1} = \{f_1, f_2, f_3, f_5, f_6, f_{13}, f_{14}, f_{15}, f_{17}, f_{23}\}$, $sv_{x2} = \{f_1, f_2, f_3, f_5, f_6, f_{13}, f_{14}, f_{16}, f_{17}, f_{23}\}$ and $sv_{x3} = \{f_1, f_2, f_3, f_6, f_{13}, f_{14}, f_{17}, f_{20}, f_{21}\}$. Those five service variants can be used to compose 5 different web service compositions (since sv_{A1} is not compatible with sv_{x3} because of the conflict relationship). Those webservice compositions are described in Table II. For convenience, we illustrated the service composition with BPMN and not BPEL.

Once all candidate solutions have been identified, we select the optimal solution by applying a decision method taking into account customer preferences. Determining such a decision method is out of this paper' scope but broadly speaking, it consists of selecting the service composition which maximizes the utility function.

VI. RELATED WORK

In this paper, we suggest a method to semi-automatically specifying a service composition by generating a BPEL specification and selecting service variants. We based our work on two main fields of research. Firstly, research on service variability and secondly, research focusing on the link between goal models and business process specifications.

Modeling services variability thanks to features or variation points is an emerging approach and more and more research is conducted in this direction. As already mentioned, Nguyen et al. suggested a framework for specifying customizable services [2].

Chang and Kim [3] studied the comparison between Software Product Line Engineering and SOC on the basis of different criteria. They suggest that core assets from SPLE and services from SOC have slightly different characteristics. They also make a distinction between four kinds of variability in service variability: workflow, composition, interface and logic variability.

In order to reduce the complexity of large services, Stollberg and Muth proposed a meta-model introducing variability concerns in services [14]. The approach allows to reduce complexity of large services by providing consumers with customized services in place of a large generic service. Moreover, their research focuses on large and complex services instead of service families. They mainly deal with the granularity issue.

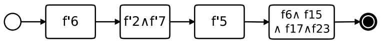
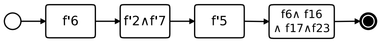
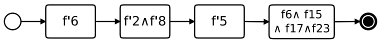
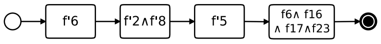
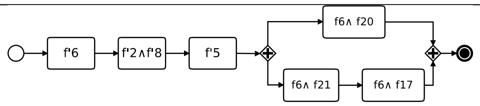
Feature modeling is not the only solution that received attention from academics to model service variability or business process variability. Other approaches have been suggested such as methods using UML or business process diagrams [7], [15].

In order to ease the identification of services, Zachos and Maiden proposed an algorithm for retrieving web services in domains that are analogical to a current requirements problem [10]. This could be used in the second step of our method.

The selection of services was also studied by Comuzzi et al. [16]. They describe a matchmaking algorithm for the ranking of functionally equivalent services. Our method differs from their proposal as it allows to compare functionally different services.

Regarding links between goal models and business processes, we can mention several works. Lapouchnian et al. [5] have used goals to configure business processes. They provide some goal model enrichments in order to annotate them with data dependencies or precedence constraints which are important in business processes.

TABLE II. IDENTIFIED SERVICE COMPOSITIONS AND SERVICE VARIANTS FOR A RUNTIME CONTEXT WHERE $\mathbf{k}(p_{34})$ AND $\mathbf{k}(p_{36})$ ARE VERIFIED - THE COST IS GIVEN FOR 600 OPERATIONS PER MONTH

Service Composition	BPMN	s_{28}	s_{29}	s_{38}	Cost
$wc_1 = sv_{A1} \cup sv_{x1}$			✓	✓	\$2687
$wc_2 = sv_{A1} \cup sv_{x2}$				✓	\$2587
$wc_3 = sv_{A2} \cup sv_{x1}$			✓	✓	\$2487
$wc_4 = sv_{A2} \cup sv_{x2}$				✓	\$2387
$wc_5 = sv_{A2} \cup sv_{x3}$		✓			\$971

Ghose et al. have studied the applicability of such methods, i.e. deriving business processes from goals, in particular, the challenge of quickly generating large numbers of effective process designs [17].

A method of self-optimization of service based applications (SBA) was suggested by Gehlert et al. [18]. They show how to decide whether a SBA should use a newly available services. In order to make this decision, they focus on goals satisfied by the new service and the current application. Goals are described by means of a Tropos goal model in which plans are service descriptions. However, they make no distinction between service variability and business process variability. Moreover, they focus on service selection and do not approach the service composition generation.

In [19], the authors focus on a service tailoring process in the homecare domain. Their method allows to compose and configure existing services, based on the user's specific requirements which are expressed in terms of goals and preferences. However, they map tasks directly to services and do not use the concept of features what reduces the approach to a service selection method and not a service customization method.

VII. Conclusion and Further Work

In this paper, we suggested a method using goals and customizable services to generate service compositions and select partner services depending on execution conditions at runtime. This method consists of 5 steps: description of a goal model, identification of customizable services, mapping between goals and features, resolution of a feature-extended goal model and finally generation of the service variants. It is based on and extends previous work completed in the domains of service variability and goal-driven business process specification.

The suggested method exploits both alternative BP specifications and service variability in order to satisfy business goals. Moreover, non-functionally similar services can be compared and their contribution to goals satisfaction are assessed. By means of an example, we illustrated the contribution of our approach.

We intend to continue our research by developing a tool supporting our method. We also need to integrate cardinalities in the service feature description and relate them with Techne. Selection of multiple service variants requires more investigation, in particular in the way we handle multiple variants selection of the same customizable service for a given business process specification.

References

[1] M. Svahnberg, J. van Gurp, and J. Bosch, "A taxonomy of variability realization techniques: Research articles," *Softw. Pract. Exper.*, vol. 35, pp. 705–754, July 2005.

[2] T. Nguyen, A. Colman, and J. Han, "Modeling and managing variability in process-based service compositions," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, G. Kappel, Z. Maamar, and H. Motahari-Nezhad, Eds. Springer Berlin / Heidelberg, 2011, vol. 7084, pp. 404–420.

[3] S. H. Chang and S. D. Kim, "A variability modeling method for adaptable services in service-oriented computing," *Software Product Line Conference, International*, vol. 0, pp. 261–268, 2007.

[4] B. Mohabbati, N. Kaviani, and D. Gaševic, "Semantic variability modeling for multi-staged service composition," in *Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL 2009)*, 2009.

[5] A. Lapouchnian, Y. Yu, and J. Mylopoulos, "Requirements-driven design and configuration management of business processes," in *Business Process Management*, ser. Lecture Notes in Computer Science, G. Alonso, P. Dadam, and M. Rosemann, Eds. Springer Berlin / Heidelberg, 2007, vol. 4714, pp. 246–261.

[6] R. Ali, R. Chitchyan, and P. Giorgini, "Context for goal-level product line derivation," in *3rd International Workshop on Dynamic Software Product Lines (DSPL09) held at SPLC'09*, 2009.

[7] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing variability in business process models: the provop approach," *J. Softw. Maint. Evol.*, vol. 22, pp. 519–546, October 2010.

[8] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *2010 18th IEEE International Requirements Engineering Conference*. IEEE, Sep. 2010, pp. 115–124. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5636885

[9] I. J. Jureta, J. Mylopoulos, and S. Faulkner, "Revisiting the core ontology and problem in requirements engineering," *IEEE International Requirements Engineering Conference*, pp. 71–80, 2008.

[10] K. Zachos and N. Maiden, "Inventing requirements from software: An empirical investigation with web services," in *Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*, ser. RE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 145–154. [Online]. Available: http://dx.doi.org/10.1109/RE.2008.39

[11] D. Batory, D. Benavides, and A. Ruiz-Cortes, "Automated analysis of feature models: challenges ahead," *Commun. ACM*, vol. 49, pp. 45–47, December 2006. [Online]. Available: http://doi.acm.org/10.1145/1183236.1183264

[12] K. Czarnecki, *Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models*. Computer Science Department, Technical University of Ilmenau,, 1998. [Online]. Available: http://www.prakinf.tu-ilmenau.de/~czarn/diss/diss.pdf

[13] K. Czarnecki and A. Wasowski, “Feature diagrams and logics: There and back again,” *Software Product Line Conference, International*, vol. 0, pp. 23–34, 2007.

[14] M. Stollberg and M. Muth, “Service customization by variability modeling,” in *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, ser. Lecture Notes in Computer Science, A. Dan, F. Gittler, and F. Toumani, Eds. Springer Berlin / Heidelberg, 2010, vol. 6275, pp. 425–434.

[15] M. Razavian and R. Khosravi, “Modeling variability in business process models using uml,” in *Proceedings of the Fifth International Conference on Information Technology: New Generations*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 82–87. [Online]. Available: http://dl.acm.org/citation.cfm?id=1396808.1397503

[16] M. Comuzzi and B. Pernici, “A framework for qos-based web service contracting,” *ACM Trans. Web*, vol. 3, no. 3, pp. 10:1–10:52, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541822.1541825

[17] A. Ghose, N. Narendra, K. Ponnalagu, A. Panda, and A. Gohad, “Goal-driven business process derivation,” in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, G. Kappel, Z. Maamar, and H. Motahari-Nezhad, Eds. Springer Berlin / Heidelberg, 2011, vol. 7084, pp. 467–476.

[18] A. Gehlert and A. Heuer, “Towards goal-driven self optimisation of service based applications,” in *Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science, P. Mähönen, K. Pohl, and T. Priol, Eds. Springer Berlin / Heidelberg, 2008, vol. 5377, pp. 13–24.

[19] M. Zarifi Eslami, A. Zarghami, B. Sapkota, and M. S. van, “Service tailoring: Towards personalized homecare systems,” in *Proceedings of the 4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing, ACT4SOC 2010*. Athenes, Greece: SciTePress, July 2010, pp. 109–121. [Online]. Available: http://doc.utwente.nl/73920/