Service Identification in Interorganizational Process Design

Devis Bianchini, Cinzia Cappiello, Valeria De Antonellis, and Barbara Pernici

Abstract—Service identification is one of the main phases in the design of a service-oriented application. The way in which services are identified may influence the effectiveness of the SOA architecture. More specifically, the granularity of the services is very important in reaching flexibility and reusing them. Such properties are crucial in interorganizational interactions based on collaborative business processes. In fact, collaboration is facilitated by ensuring a homogeneous description of services at the right level of granularity. In this paper, we provide a detailed description of P2S (Process-to-Services), a computer-aided methodology to enable the identification of services that compose a collaborative business process. The methodology is based on metrics defined to setup service granularity, cohesion, coupling, and reuse. A prototype tool based on the methodology is also described with reference to a real case scenario.

Index Terms—Service-based process design, service identification

1 Introduction

Internet and service-oriented technologies provide a strategic platform to support the collaboration among enterprises. Organizations are exploiting the network for sharing applications and integrating processes, services and knowledge. In particular, Service Oriented Architecture (SOA) enables such interorganizational interactions by facilitating and managing service integration [25]. In fact, service technologies should be the basis of the creation of a world where application components are easily assembled to create dynamic business processes [22]. In this scenario, services can encapsulate old or new components deriving from external and internal applications.

For the design of service-based applications, several lifecycles have been proposed. We refer to the one described in [23], that is composed of the following activities:

1. business process analysis (further composed of goal analysis, SOA project planning, service identification),
2. service analysis and specification,
3. service provisioning,
4. deployment,
5. execution & monitoring.

Service identification is defined as ''*the process of identifying candidate services and creating a service portfolio of business-aligned IT services that collectively support the business processes and goals of the organization*'' [7], [11], [14]. Such activity can be performed by using three different strategies, i.e., top-down, bottom-up, meet-in-the-middle. In the top-down strategy, the SOA lifecycle starts from a workflow-based representation of a business process and decomposes it into component services that can be used to implement one or more process tasks [16], [23]. In this approach, a repository of ready-to-use services is not available and the service identification works within the business process analysis activity only. Approaches that deal with bottom-up or meet-in-the-middle strategies mostly focus on the alignment between the ideal set of services identified in the business processes and the services available at the IT level [8]. In all the strategies, the *service identification* phase has been recognized as a fundamental step of the SOA lifecycle [28]. Service identification must guarantee a homogeneous description of candidate services at the same level of granularity. The definition of the most suitable level of granularity is not a trivial task. The higher the granularity, the higher the resulting flexibility and reuse of component services. Nevertheless, high granularity implies more data exchanges and calls between services. High granularity also means many services involved in the process execution, that is, higher complexity in their governance.

Service identification is a debated topic in the literature. Some approaches focus on a methodological perspective by providing guidelines to support the designer in the identification of functionalities as in candidate services [16], [23]. Other approaches focus on metrics to evaluate the quality of service identification [20], [31]. Such metrics enable a quantitative comparison between (given) different sets of identified services, allowing the designer to select the best one, but providig him/her a scarce feedback on the rationale behind their construction.

In this paper, we illustrate the P2S (Process-to-Services) methodology for service identification, to be applied in a top-down context or in any case in which a portfolio of

- *D. Bianchini and V. De Antonellis are with the University of Brescia, Department of Information Engineering, via Branze 38, 25123 Brescia, Italy. E-mail: {devis.bianchini, valeria.deantonellis}@unibs.it.*
- *C. Cappiello and B. Pernici are with Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, via Ponzio 34/5, 20133 Milan, Italy. E-mail: {cinzia.cappiello, barbara.pernici}@polimi.it.*

Manuscript received 30 Jan. 2012; revised 10 Feb. 2013; accepted 10 Apr. 2013. Date of publication 16 May 2013; date of current version 13 June 2014.

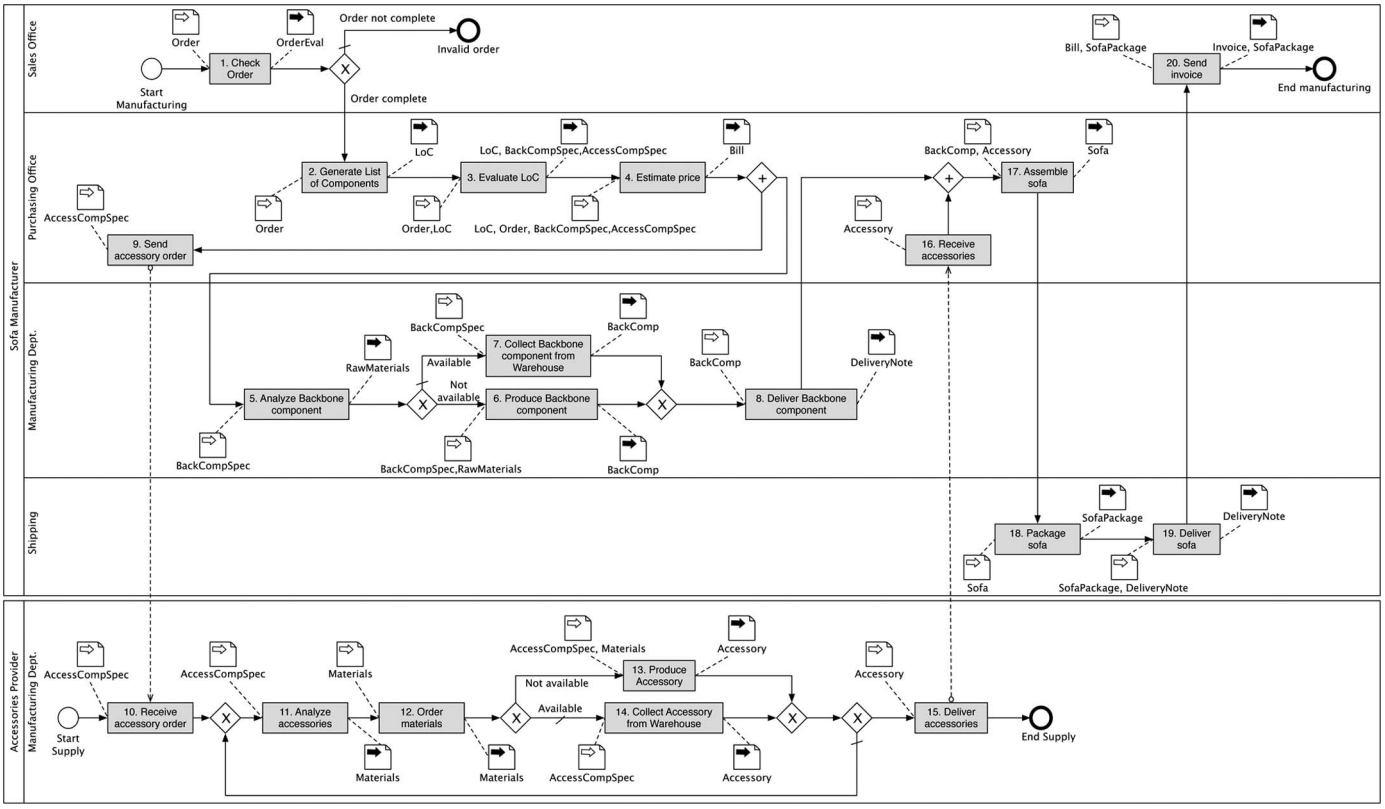

Fig. 1. BPMN 2.0 representation of the *Sofa Production* process considered as case study.

available services is not present. The methodology is designed to implement the guidelines for service design, ensuring at the same time proper metrics to automate service identification. The initial principles of the P2S methodology have been introduced in [4]. In this paper, we perform a step forward by providing: 1) the formulation of service identification metrics at a conceptual level; 2) a further automation of some aspects of the methodology based on such metrics, in particular the aggregation of candidate services and the service reconciliation algorithms; 3) a detailed evaluation of the prototype tool (P2Stool) that implements the methodology.

The paper is structured as follows. In Section 2 we provide preliminary definitions and we describe a real case scenario where the P2S methodology has been applied. Section 3 introduces the phases of the methodology that are detailed in Sections 4, 5, and 6 with reference to the real case scenario. In Section 7 the evaluation of the prototype tool that implements the methodology is discussed. A comparison with existing solutions is presented in Section 8. Finally, Section 9 gives some hints about future work.

2 Preliminary Definitions

2.1 Running Example

We consider a case study in the manufacturing domain that has been developed within the TEKNE research project.[1] A sofa manufacturer, who produces the backbone components and purchases all the accessories from trusted suppliers, wants to apply for collaborative processes following the P2S methodology. The *Sofa Production* process is shown in Fig. 1. We adopt BPMN 2.0 as workflow-based notation, independent from implementation technology and platforms.

Once the client's order is received by the sales office, it is checked and rejected if it is incomplete. Otherwise, the sales office forwards the order to the purchasing office, that is responsible for the relationships with providers of raw materials. The purchasing office generates the List of Components (LoC) and evaluates it in order to identify the required components and the providers to contact. Thus, the price is estimated. Sub-orders are created and sent to the internal manufacturing department for the backbone component production and to external providers for the production of the accessories. Each unit involved in the production step checks the received document and starts the production if the required component is already available. Finally, at the end of the production step, they commit the delivery of the realized components. The assemblage of the components is in charge of the purchasing office. The shipping department receives the final product and delivers it to the sales office, that is in charge of generating the invoice and delivering the sofa to the client.

2.2 Business Processes, Tasks, and Services

2.2.1 Simple and Composite Tasks

We model a business process $\mathcal{BP}$ as a workflow composed of an entry point (or start event), a set $\mathcal{T}$ of simple tasks, which are the elementary work units that collectively

1. http://www.tekne-project.it/

TABLE 1
Task Descriptors and Data Dependencies in the Running Example

t_i	d_{t_i}	IN_{t_i}	OUT_{t_i}	r_{t_i}	A_{t_i}
t_1	{check, order}	{Order}	{OrderEval.}	SalesOff.	{R(Order), C(OrderEval.)}
t_2	{generate, list, component}	{Order}	{LoC}	Purch.Off.	{R(Order), C(LoC)}
t_3	{evaluate, list, component}	{LoC, Order}	{LoC, BackboneComp.Spec., AccessoryComp.Spec.}	Purch.Off.	{R(Order), U(LoC), C(BackboneComp.Spec.), C(AccessoryComp.Spec.)}
t_4	{estimate, price}	{LoC, Order, BackboneComp.Spec., AccessoryComp.Spec.}	{Bill}	Purch.Off.	{R(Order),R(LoC),C(Bill), R(BackboneComp.Spec.), R(AccessoryComp.Spec.)}
t_5	{analyze, backbone, component}	{BackboneComp.Spec.}	{RawMaterials}	Manuf.Dept.	{R(BackboneComp.Spec.), C(RawMaterials)}
t_6	{produce, backbone, component}	{BackboneComp.Spec., RawMaterials}	{Backb.Comp.}	Manuf.Dept.	{R(BackboneComp.Spec.), R(RawMaterials), C(Backb.Comp.)}
t_7	{collect, backbone, component, warehouse}	{BackboneComp.Spec.}	{Backb.Comp.}	Manuf.Dept.	{R(BackboneComp.Spec.), C(Backb.Comp.)}
t_8	{deliver, backbone, component}	{Backb.Comp.}	{DeliveryNote}	Manuf.Dept.	{R(Backb.Comp.), C(DeliveryNote)}
t_9	{send, accessory, order}	{AccessoryComp.Spec.}		Purch.Off.	{R(AccessoryComp.Spec.)}
t_{10}	{receive, accessory, order}	{AccessoryComp.Spec.}		Manuf.Dept.	{R(AccessoryComp.Spec.)}
t_{11}	{analyze, accessory}	{AccessoryComp.Spec.}	{Materials}	Manuf.Dept.	{R(AccessoryComp.Spec.), C(Materials)}
t_{12}	{order, material}	{Materials}	{Materials}	Manuf.Dept.	{U(Materials)}
t_{13}	{produce, accessory}	{AccessoryComp.Spec., Materials}	{Accessory}	Manuf.Dept.	{R(AccessoryComp.Spec.), C(Accessory), R(Materials)}
t_{14}	{collect, accessory, warehouse}	{AccessoryComp.Spec.}	{Accessory}	Manuf.Dept.	{R(AccessoryComp.Spec.), C(Accessory)}
t_{15}	{deliver, accessory}	{Accessory}		Manuf.Dept.	{R(Accessory)}
t_{16}	{receive, accessory}	{Accessory}		Purch.Off.	{R(Accessory)}
t_{17}	{assemble, sofa}	{Backb.Comp., Accessory}	{Sofa}	Purch.Off.	{R(Backb.Comp.), R(Accessory), C(Sofa)}
t_{18}	{package, sofa}	{Sofa}	{SofaPackage}	Ship.Dept.	{R(Sofa), C(SofaPackage)}
t_{19}	{deliver, sofa}	{SofaPackage, DeliveryNote}	{DeliveryNote}	Ship.Dept.	{R(SofaPackage), U(DeliveryNote)}
t_{20}	{send, invoice}	{Bill, SofaPackage}	{Invoice, SofaPackage}	SalesOff.	{R(Bill), U(SofaPackage), C(Invoice)}

$t_2 \xrightarrow{d} t_3$ $t_2 \xrightarrow{d} t_4$ $t_3 \xrightarrow{d} t_4$ $t_3 \xrightarrow{d} t_5$ $t_3 \xrightarrow{d} t_6$ $t_3 \xrightarrow{d} t_7$ $t_3 \xrightarrow{d} t_9$ $t_3 \xrightarrow{d} t_{10}$ $t_3 \xrightarrow{d} t_{11}$ $t_3 \xrightarrow{d} t_{13}$ $t_3 \xrightarrow{d} t_{14}$ $t_4 \xrightarrow{d} t_{20}$ $t_5 \xrightarrow{d} t_6$ $t_6 \xrightarrow{d} t_8$ $t_6 \xrightarrow{d} t_{17}$ $t_7 \xrightarrow{d} t_8$
$t_7 \xrightarrow{d} t_{17}$ $t_8 \xrightarrow{d} t_{19}$ $t_{11} \xrightarrow{d} t_{12}$ $t_{11} \xrightarrow{d} t_{13}$ $t_{12} \xrightarrow{d} t_{13}$ $t_{13} \xrightarrow{d} t_{15}$ $t_{13} \xrightarrow{d} t_{16}$ $t_{13} \xrightarrow{d} t_{17}$ $t_{14} \xrightarrow{d} t_{15}$ $t_{14} \xrightarrow{d} t_{16}$ $t_{14} \xrightarrow{d} t_{17}$ $t_{17} \xrightarrow{d} t_{18}$ $t_{18} \xrightarrow{d} t_{19}$ $t_{18} \xrightarrow{d} t_{20}$

achieve the workflow goal, one or more exit points (or stop events). We define a *task descriptor* $t_i \in \mathcal{T}$ as

$$t_i = \langle d_{t_i}, IN_{t_i}, OUT_{t_i}, r_{t_i}, A_{t_i} \rangle \quad (1)$$

where: d_{t_i} is the vector of terms which describe the task; they are extracted from the task name and description by applying text mining techniques, such as stop word management, camel case processing and stemming; IN_{t_i} (resp., OUT_{t_i}) is the set of task inputs (resp., task outputs); r_{t_i} is the *role* (or *actor*) which is responsible of the task execution according to BPMN 2.0 notation; A_{t_i} is the set of CRUD actions (Create-Read-Update-Delete) performed by the task on its inputs/outputs [26]. Role involvement in task execution is represented through *swimlanes*. Task inputs/outputs are expressed as business objects. A business object with name n can be either a *simple object* $\langle n, t \rangle$, described by a built-in primitive type t (e.g., boolean, string, byte), or a *structured object* $\langle n, \mathcal{P} \rangle$, described by a collection of attributes $\mathcal{P}$. Each attribute can be in turn a simple object or the reference to a structured object. The list of task descriptors for the running example is shown in Table 1.

The business process workflow also defines the order and the conditions for executing tasks, their synchronization and the flow of business objects among them. The flow structure is specified by means of a set of control constructs including sequences (`seq`), alternative choices (`alt`), parallel executions (`par`) and loops (`loop`). Control constructs can be nested to model complex structures as sub-processes. A sub-process `subp` is valid if can be expressed as a nested application of the four control constructs `seq`, `par`, `alt` and `loop`, that is

$$\begin{aligned} &\texttt{subp} = t_i \in \mathcal{T} \mid \texttt{construct}(\texttt{subp}\{\texttt{subp}\}) \mid [\texttt{cond}]\texttt{subp} \\ &\texttt{construct} = \texttt{seq} \mid \texttt{par} \mid \texttt{alt} \mid \texttt{loop} \end{aligned} \quad (2)$$

where {`subp`} means a list of zero or more `subp`. Notation [`cond`] `subp` represents the execution of sub-process `subp` if the condition `cond` is true, i.e., after the `alt` construct. For instance, the sequence of tasks t_2, t_3 and t_4 in Fig. 1 is modeled as $\texttt{seq}(t_2, t_3, t_4)$, the sub-process composed of tasks t_5, t_6, t_7 and t_8 is modeled as $\texttt{seq}(t_5, \texttt{alt}([\text{'}\texttt{Not available}\text{'}]t_6, [\text{'}\texttt{Available}\text{'}]t_7), t_8)$.

2.2.2 Flow and Data Dependencies

Given two simple tasks t_i and t_j, a *flow dependency* holds between them, denoted with $t_i \xrightarrow{f} t_j$, if one of the following conditions holds: 1) t_i and t_j are directly connected by an edge (direct flow dependency, we denote it with $t_i \overset{f}{\Rightarrow} t_j$); 2) t_i and t_j are respectively the predecessor and the successor of a split or a join connector (we still denote it with $t_i \overset{f}{\Rightarrow} t_j$); 3) there is another task t_k such that $t_i \xrightarrow{f} t_k$ and $t_k \xrightarrow{f} t_j$ (indirect connection).

Within the business process, each business object gets through the information life-cycle: it is created, can be updated and read one or more times, is finally deleted. A business object can be created or deleted just once within the scope of the business process. Given a task t_i, for each business object $bo \in (IN_{t_i} \cup OUT_{t_i})$, the CRUD actions in A_{t_i} can be recognized as follows:

- a *create* action, $C(bo)$, is recognized if bo belongs only to the output set of t_i and there is no t_j such that $t_j \xrightarrow{f} t_i$ and bo belongs to the inputs/outputs sets of t_j (bo is not used before t_i);
- a *read* action, $R(bo)$, is recognized if bo belongs only to the input set of t_i;
- an *update* action, $U(bo)$, is recognized if bo belongs both to the input and to the output set of t_i;
- a *delete* action, $D(bo)$, is recognized if bo belongs only to the input set of t_i and there is no t_j such that $t_i \xrightarrow{f} t_j$ and bo belongs to the input/output sets of t_j (that is, bo is no more used after t_i).

We remark that the *delete* action is considered, within the scope of the business process $\mathcal{BP}$, as the last action

performed on a business object bo. This not necessarily means that bo is physically deleted.

Given two tasks t_i and t_j, a *data dependency* holds between them, denoted with $t_i \xrightarrow{d} t_j$, if all the following conditions hold: (a) $t_i \xrightarrow{f} t_j$; (b) t_j uses (updates, reads or deletes) at least a business object that is created or updated by t_i. Consider for example in Table 1 the I/Os of tasks t_2, t_3 and t_4. The data dependencies $t_2 \xrightarrow{d} t_3$, $t_2 \xrightarrow{d} t_4$ and $t_3 \xrightarrow{d} t_4$ follow. Data dependencies for the running example are shown in Table 1.

2.2.3 Services

In SOA, a business process $\mathcal{BP}$ can be implemented as a set of services Σ, where each $\mathcal{S}_j \in \Sigma$ is a valid sub-process of $\mathcal{BP}$ with at most an incoming link which represents the service request and at most an outgoing link which represents the service response. We define a *service descriptor* as follows:

$$\mathcal{S}_j = \langle d_{\mathcal{S}_j}, IN_{\mathcal{S}_j}, OUT_{\mathcal{S}_j}, R_{\mathcal{S}_j}, A_{\mathcal{S}_j} \rangle \quad (3)$$

where: $d_{\mathcal{S}_j}$ is the vector of terms which describe the service, obtained as the union set of d_{t_i} for each task t_i in $\mathcal{S}_j$; $IN_{\mathcal{S}_j}$ (resp., $OUT_{\mathcal{S}_j}$) is the $\mathcal{S}_j$ input set (resp. the output set); among the business objects within $IN_{\mathcal{S}_j}$ and $OUT_{\mathcal{S}_j}$, we do not consider those that are used only inside $\mathcal{S}_j$, that is, are only associated to a control flow from two tasks inside the service; $R_{\mathcal{S}_j}$ is the set of all roles that are responsible of tasks t_i in $\mathcal{S}_j$; $A_{\mathcal{S}_j}$ is the set of CRUD actions performed on business objects in $IN_{\mathcal{S}_j}$ and $OUT_{\mathcal{S}_j}$. For example, the descriptor of a candidate service $\mathcal{S}_1$ which groups tasks t_2, t_3 and t_4 (that is, $\mathcal{S}_1 = \mathtt{seq}(t_2, t_3, t_4)$) is the following:

$$\begin{aligned} d_{\mathcal{S}_1} &= \{\mathtt{generate}, \mathtt{evaluate}, \mathtt{list}, \mathtt{component}, \\ &\quad \mathtt{estimate}, \mathtt{price}\} \\ IN_{\mathcal{S}_1} &= \{\mathtt{Order}\} \\ OUT_{\mathcal{S}_1} &= \{\mathtt{BackboneComp.Spec.}, \mathtt{AccessoryComp.Spec.}, \\ &\quad \mathtt{Bill}\} \\ R_{\mathcal{S}_1} &= \{\mathtt{Purch.Off.}\} \\ A_{\mathcal{S}_1} &= \{R(\mathtt{Order}), R(\mathtt{BackboneComp.Spec.}), \\ &\quad C(\mathtt{AccessoryComp.Spec.}), C(\mathtt{Bill})\}. \end{aligned}$$

The business object LoC is created, updated and read only inside $\mathcal{S}_1$ and is not considered among service inputs/outputs.

However, services are not generic valid sub-processes of $\mathcal{BP}$, since additional properties must be exploited to guide their identification:

- a service is a minimal set of tasks that performed together create an output that is a tangible value for a process actor (property #1);
- services are self-contained and interact among each other using decoupled message exchanges, that is, present high cohesion and low coupling (property #2);
- service design has to lead to high interoperability through high functionality reuse (property #3).

In the next sections, we will define the notion of *value* and we will show how to ensure the above properties.

3 Overview of the P2S Methodology

The P2S methodology guides the designer to identify candidate services in the business process ensuring properties #1-#3 through the execution of three main phases: *business process analysis*, *candidate service identification* and *candidate service reconciliation*.

3.1 Business process analysis

In this phase, task descriptors, flow dependencies and data dependencies are exploited to analyse the business process structure, according to two perspectives: *value analysis* and *task dependency analysis*.

During the *value analysis*, exchanges of business objects between actors and CRUD actions performed by tasks on business objects are exploited to identify values, that is, business objects that are created within the business process and are provided by one of the process actors (*service provider*) to a different actor (*service requester*) [15]. Values will be used to identify a preliminary set of candidate services (property #1).

Data dependencies and flow dependencies between business process tasks are analyzed to identify the dependency of a task from the execution of other tasks (*task dependency analysis*). The result of task dependency analysis is a matrix of dependencies that will be used in the next methodological phase to evaluate the process cohesion and coupling (property #2).

3.2 Candidate Service Identification

In this phase, results from the previous analysis are used to identify the services that compose the business process. This phase is composed of the following steps:

a) *Value-based service identification*—A preliminary identification of candidate services is performed on the basis of the value analysis.
b) *Candidate service refinement*—An iterative algorithm is applied to refine the service identification according to cohesion and coupling metrics, based on task dependency analysis. The coupling/cohesion ratio is used as a measure of the quality of the overall decomposition: the smaller the value of such ratio the higher the quality of the service identification procedure.

The service identification performed in this phase aims at defining, at design time, a set of services that properly combined are able to execute the process. The use of metrics such as the coupling/cohesion ratio allows us to define the suitable granularity that should have a positive impact on some software engineering measures, such as reusability and composability (see [29]). Note that the quality of a portfolio may also depend on the needs served by the process. For instance, a manufacturing process like the one we used in the running example typically takes several days to be completed and a good decomposition would be the one that minimizes the data (and thus communication) exchanges between component services (and related actors). On the other hand, we have also to consider that fully automated processes should be

TABLE 2
Values Identified in the Running Example

$v \in \mathcal{V}$	t_j	r_{t_j}	$\{r'\}$
OrderEval.	t_1	SalesOff.	eu
BackboneComp.Spec.	t_3	Purch.Off.	Manuf.Dept.
AccessoryComp.Spec.	t_3	Purch.Off.	Manuf.Dept.
Bill	t_4	Purch.Off.	SalesOff.
Backb.Comp.	$t_6 \mid t_7$	Manuf.Dept.	Purch.Off.
DeliveryNote	t_8	Manuf.Dept.	Ship.Dept.
Accessory	$t_{13} \mid t_{14}$	Manuf.Dept.	Purch.Off.
Sofa	t_{17}	Purch.Off.	Ship.Dept.
SofaPackage	t_{18}	Ship.Dept.	SalesOff.
Invoice	t_{20}	SalesOff.	eu

implemented through highly performant services which reduce the execution time and cost. Highly cohesive and loosely coupled services ensure limited data exchanges by definition and, at the same time, reduce the risk of communication overloading which could decrease the performance of the overall process. The execution time and cost related to the implementation of each single component service will be considered at deployment and execution time, that are steps which follow service identification in the service lifecycle we considered in this paper.

3.3 Candidate Service Reconciliation

Within the whole collaborative business process there can be tasks or groups of tasks that perform the same or similar functionalities. Similarity between tasks or groups of tasks is estimated taking into account both the business objects on which the tasks work and the actions performed to increase reuse of the components in the service portfolio (property #3). The candidate service reconciliation aims at identifying redundant candidate services viewed as groups of tasks and reconciling them for enhancing the service portfolio reuse. The P2S methodology also provides support for the interactive construction of a semantic dictionary to enable semantic agreement among all parties engaged in the process analysis and service identification. The semantic dictionary is semi-automatically built before starting the methodological phases. In the dictionary, business object names and attributes and task names, after stop word management, camel case processing and stemming, are stored as terms and organized by means of synonymy, generalization and aggregation relationships. The dictionary is built as described in [9] by relying on a *domain-specific ontology* and a *general purpose ontology*. The domain-specific ontology contains terms related to a given application domain and is built by a domain expert analyzing the terms used in the business process specification. It offers more accuracy in the relationships between terms. We use the WordNet lexical system as general purpose ontology to offer wider coverage.

The methodology is semi-automatic: candidate services are presented to the designer, properly motivated through metrics computation. The designer may confirm or reject the recommendations.

4 Business Process Analysis

4.1 Value Analysis

According to the approaches on value network modeling [15], services are units of work that are invoked by one of the actors engaged in the business process to obtain tangible values from another actor. Among the actors, we always include the external user eu, who interacts with the whole process (in the running example, the client who submits the order and receives the sofa). Identification of value exchanges between actors is based on the analysis of CRUD actions performed by tasks and on associations between actors and tasks (represented using swimlanes in the BPMN). A *valuev* for an actor r' produced by another actor $r \neq r'$ is a business object that is created (for the first time) by one of the tasks t_j such that $r_{t_j} = r$ and is used (read, deleted or updated) by one of the tasks t_i such that $r_{t_i} = r'$, that is, there is a data dependency $t_j \xrightarrow{d} t_i$ and $C(v) \in A_{t_j}$. The result of the value analysis is a set $\mathcal{V}$ of value exchanges. Specifically, value identification is performed as follows: 1) tasks t_j such that $C(bo) \in A_{t_j}$ are identified; 2) if there exists a task t_i such that $t_j \xrightarrow{d} t_i$ and $r' = r_{t_i} \neq r_{t_j}$, then bo is recognized as a value for r'; 3) a new value exchange record is created. Each value exchange record is described by the business object bo, the task t_j that creates bo, the responsible role r_{t_j} which produces the value and the set of roles r' which receive the value.

For example, let us consider the business objects Bill and BackboneComp.Spec in Table 1. The former is created by task t_4 (managed by the purchasing office) and is used by the Send Invoice task t_{20} (managed by the sales office). The latter is created by task t_3 (managed by the purchasing office), updated by task t_4 and used by task t_5 (managed by the manufacturing department). Therefore, BackboneComp.Spec. and Bill are two values produced by the purchasing office for the manufacturing department and the sales office, respectively. On the other hand, Materials is created and used only by the manufacturing department and is not recognized as a value. Table 2 lists the value exchanges identified for the running example.

4.2 Task Dependency Analysis

Let us consider a pair of tasks t_i and t_j such that $t_i \xrightarrow{d} t_j$ (that is, there is a data dependency from t_i and t_j). This means that there are business objects $\{bo\}$ created by t_i and

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18	t19	t20
t1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
t2	-	-	0,67	0,40	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
t3	-	-	-	0,86	0,50	0,40	0,50	-	0,50	0,50	0,50	-	0,40	0,50	-	-	-	-	-	-
t4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0,67
t5	-	-	-	-	-	0.67	-	-	-	-	-	-	-	-	-	-	-	-	-	-
t6	-	-	-	-	-	-	-	1,00	-	-	-	-	-	-	-	-	0,67	-	-	-
t7	-	-	-	-	-	-	-	1,00	-	-	-	-	-	-	-	-	0,67	-	-	-
t8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0,67	-
t9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
t10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
t11	-	-	-	-	-	-	-	-	-	-	-	1,00	0,67	-	-	-	-	-	-	-
t12	-	-	-	-	-	-	-	-	-	-	-	-	0,67	-	-	-	-	-	-	-
t13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1,00	1,00	0,67	-	-	-
t14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1,00	1,00	0,67	-	-	-
t15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
t16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
t17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1,00	-	-
t18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0,67	0,67
t19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
t20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Fig. 2. $\mathcal{TD}$ matrix of task dependencies $\tau(t_i, t_j)$.

updated, read or deleted by t_j, according to the definition of data dependency in Section 2.2, $\{bo\}$ both belong to OUT_{t_i} and to IN_{t_j}. We define the degree of task dependency from t_i to t_j as

$$\tau(t_i, t_j) = 2 \cdot \frac{|bo|_{t_i \xrightarrow{d} t_j}}{|OUT_{t_i}| + |IN_{t_j}|} \in [0, 1], t_i \neq t_j \qquad (4)$$

where $|bo|_{t_i \xrightarrow{d} t_j}$ is the number of objects in $\{bo\}$; $\tau(t_i, t_j) = 1$ if $OUT_{t_i} = IN_{t_j}$.

For example, let us consider the tasks t_2, t_3 and t_4. Task dependencies $t_2 \xrightarrow{d} t_3$ and $t_2 \xrightarrow{d} t_4$ hold, since t_2 creates `LoC` that is used by both t_3 and t_4. In particular, $|bo|_{t_2 \xrightarrow{d} t_3} = |bo|_{t_2 \xrightarrow{d} t_4} = 1$. Also $t_3 \xrightarrow{d} t_4$ holds, since t_3 creates `BackboneComp.Spec.` and `AccessoriesSpec.` and updates `LoC`, which are used by t_4. In particular, $|bo|_{t_3 \xrightarrow{d} t_4} = 3$. The task dependency values are the following:

$$\tau(t_2, t_3) = 2 \cdot \frac{1}{1+2} = 0.67$$
$$\tau(t_2, t_4) = 2 \cdot \frac{1}{1+4} = 0.40$$
$$\tau(t_3, t_4) = 2 \cdot \frac{3}{3+4} = 0.86.$$

Task dependency between t_3 and t_4 is the highest one, since all the outputs of t_3 are used by t_4 and three of the four inputs of t_4 come from t_3.

The result of task dependency analysis is a matrix $\mathcal{TD}[n][n]$, where $\mathcal{TD}[i][j] = \tau(t_i, t_j)$ and n is the number of simple tasks. In Fig. 2 the $\mathcal{TD}$ matrix of the running example is shown. □

5 Candidate Service Identification

5.1 Value-Based Service Identification

In this step, we consider the values identified in the previous analysis in order to identify a preliminary set Σ of services that will be further refined. The pseuso-code of the algorithm which implements this step is shown in Fig. 3. We will explain it with the help of the running example. First, the set χ of tasks that produce a value from the set $\mathcal{V}$ is considered as candidate services $\overline{\mathcal{S}_i}$ (rows 3-10). For example, tasks t_3 (which produces the `BackboneComp.Spec.` value) and tasks t_{13}, t_{14} (which produce the `Accessory` value) are selected. Therefore, the CandidateServicePopulation routine is executed (rows 11-13). This routine recursively adds to the candidate service $\overline{\mathcal{S}_i}$ those tasks $t_i \in \mathcal{T}$ that: 1)

ValueBasedIdentification algorithm
Inputs: the business process $\mathcal{BP}$ and the set $\mathcal{T}$ of simple tasks;
the matrix $\mathcal{TD}$ of data dependencies between tasks;
the set $\mathcal{V}$ of value exchange;
Outputs: the set Σ of candidate services in the portfolio;

(1) **begin**
(2) $\Sigma = \emptyset$; $\chi = \emptyset$; // χ will contain the tasks that produce values
(3) **foreach** ($v \in \mathcal{V}$)
(4) create a new service $\overline{\mathcal{S}_i}$ and add $\overline{\mathcal{S}_i}$ to Σ;
(5) **foreach** task t_j that produces the value v
(6) add t_j to $\overline{\mathcal{S}_i}$;
(7) add t_j to χ;
(8) remove t_j from $\mathcal{T}$;
(9) **end foreach**
(10) **end foreach**
(11) **foreach** ($\overline{\mathcal{S}_i} \in \Sigma$)
(12) $\overline{\mathcal{S}_i}$ = CandidateServicePopulation($\overline{\mathcal{S}_i}$,$\mathcal{BP}$,χ);
(13) **end foreach**
(14) **foreach** simple tasks not yet assigned to any service
(15) create a new service $\overline{\mathcal{S}_k}$ and add $\overline{\mathcal{S}_k}$ to Σ;
(16) **end foreach**
(17) **return** Σ;
(18) **end**

Fig. 3. Algorithm for value-based service identification.

present a direct flow dependency towards one of the tasks t_j already included in $\overline{\mathcal{S}_i}$ (i.e., $t_i \overset{f}{\Rightarrow} t_j$) and 2) present a data dependency towards one of the tasks t_j already included in $\overline{\mathcal{S}_i}$ (i.e., $t_i \xrightarrow{d} t_j$) and 3) do not produce a value from the set $\mathcal{V}$ by themselves (i.e., $t_i \notin \chi$). For example, consider tasks t_{13} and t_{14}. They are collected in the same service (that is, $\overline{\mathcal{S}_F}$ in Table 3); moreover, the task t_{12} is added to $\overline{\mathcal{S}_F}$ since $t_{12} \overset{f}{\Rightarrow} t_{13}$ (or, equivalently, $t_{12} \overset{f}{\Rightarrow} t_{14}$), $t_{12} \xrightarrow{d} t_{13}$ (see Table 1 on page 4) and $t_{12} \notin \chi$ (it does not produce values). Furthermore, t_{11} is added to $\overline{\mathcal{S}_F}$ since $t_{11} \overset{f}{\Rightarrow} t_{12}$, $t_{11} \xrightarrow{d} t_{12}$, $t_{11} \notin \chi$. These three conditions do not hold for other simple tasks and the CandidateServicePopulation routine stops for service $\overline{\mathcal{S}_F}$. Similarly, if we consider the candidate service $\overline{\mathcal{S}_C}$ in Table 3, the task t_3 is not added to $\overline{\mathcal{S}_C}$ since, although $t_3 \overset{f}{\Rightarrow} t_4$ and $t_3 \xrightarrow{d} t_4$, task t_3 produces values by its own (see Table 2). In this way, $\overline{\mathcal{S}_A}$, $\overline{\mathcal{S}_B}$, $\overline{\mathcal{S}_C}$, $\overline{\mathcal{S}_D}$, $\overline{\mathcal{S}_E}$, $\overline{\mathcal{S}_F}$, $\overline{\mathcal{S}_G}$, $\overline{\mathcal{S}_H}$ and $\overline{\mathcal{S}_I}$ candidate services are identified (see Table 3). Finally, those tasks that are not assigned to any $\overline{\mathcal{S}_i}$ yet are considered as additional candidate services (rows 14-16). Therefore, the other services in Table 3 are identified.

5.2 Candidate Service Refinement

The algorithm in Fig. 3 has been designed to group, in the same service, tasks that are linked through a data dependency, thus enabling a high internal cohesion. However, the value-based service identification step does not ensure that identified services are also loosely coupled. For example, let us consider services $\mathcal{S}_B$ and $\mathcal{S}_C$ identified in the previous section, the data dependency that exists between tasks t_3 and t_4 suggests that it would be better to aggregate $\mathcal{S}_B$ and $\mathcal{S}_C$ into the same service. The resulting service $\mathcal{S}_{BC}$ still produces values for the process actors (`BackboneComp.Spec.` and `Bill`), that is, $\mathcal{S}_{BC}$ is still in accordance with property #1. In the P2S methodology, the choice of the ''best solution'' (i.e., a high quality service identification) is driven by a quantitative evaluation of the overall cohesion and coupling of the identified services. Cohesion and coupling computation is based on task

TABLE 3
Preliminary Set of Services Identified for the Running Example in the Value-Based Service Identification Step

$\overline{\mathcal{S}_A}$	$\{t_1\}$	$\overline{\mathcal{S}_B}$	$\{t_2, t_3\}$	$\overline{\mathcal{S}_C}$	$\{t_4\}$	$\overline{\mathcal{S}_D}$	$\{t_5, t_6, t_7\}$
$\overline{\mathcal{S}_E}$	$\{t_8\}$	$\overline{\mathcal{S}_F}$	$\{t_{11}, t_{12}, t_{13}, t_{14}\}$	$\overline{\mathcal{S}_G}$	$\{t_{17}\}$	$\overline{\mathcal{S}_H}$	$\{t_{18}\}$
$\overline{\mathcal{S}_I}$	$\{t_{20}\}$	$\overline{\mathcal{S}_J}$	$\{t_9\}$	$\overline{\mathcal{S}_K}$	$\{t_{10}\}$	$\overline{\mathcal{S}_L}$	$\{t_{15}\}$
$\overline{\mathcal{S}_M}$	$\{t_{16}\}$	$\overline{\mathcal{S}_N}$	$\{t_{19}\}$				

dependency defined in Equation (4). The adopted cohesion/coupling metrics have been inspired by their well-known application in software engineering [31] and have been adapted to the problem of service identification and modified to consider the task dependency coefficient.

The service cohesion quantifies how much the tasks within the service are tight to provide the value associated to the service. The higher the dependency between tasks in the same service $\mathcal{S}$, the higher the service cohesion. Service cohesion must be maximized. We define the *internal cohesion* of a candidate service $\mathcal{S}$ as

$$coh(\mathcal{S}) = \begin{cases} \frac{\sum_{i,j} \tau(t_i,t_j)}{\frac{|\mathcal{S}|\cdot(|\mathcal{S}|-1)}{2}} \ \forall\, t_i, t_j \in \mathcal{S} & |\mathcal{S}| > 1 \\ 1 & |\mathcal{S}| = 1 \end{cases} \quad (5)$$

where $|\mathcal{S}|$ is the number of tasks in $\mathcal{S}$. The denominator corresponds to the number of evaluations of the task dependency coefficient, where for each pair t_i and t_j (with $t_i \neq t_j$), both the task dependency $\tau(t_i, t_j)$ and $\tau(t_j, t_i)$ are evaluated. The service coupling quantifies how much distinct services need to interact for providing their respective values. The higher the dependency between tasks belonging to two distinct services $\mathcal{S}_1$ and $\mathcal{S}_2$, the higher the coupling between $\mathcal{S}_1$ and $\mathcal{S}_2$. Service coupling must be minimized. Given two candidate services $\mathcal{S}_1$ and $\mathcal{S}_2$ (with $\mathcal{S}_1 \neq \mathcal{S}_2$), the coupling between them is computed as

$$coup(\mathcal{S}_1, \mathcal{S}_2) = \frac{\sum_{i,j} \tau(t_i, t_j)}{|\mathcal{S}_1|\cdot|\mathcal{S}_2|} \ \forall\, t_i \in \mathcal{S}_1 \ \wedge \ \forall\, t_j \in \mathcal{S}_2. \quad (6)$$

Also in this case, the denominator corresponds to the number of evaluations, where dependency between each task $t_i \in \mathcal{S}_1$ and each task $t_j \in \mathcal{S}_2$ is evaluated. Service cohesion and coupling coefficients are used to evaluate the average cohesion and coupling of the set of identified services for the process $\mathcal{BP}$, respectively

$$pcoh(\mathcal{BP}) = \frac{\sum coh(\mathcal{S}_i)}{|\Sigma|} \quad (7)$$

$$pcoup(\mathcal{BP}) = \begin{cases} \frac{\sum_{i,j} coup(\mathcal{S}_i, \mathcal{S}_j)}{\frac{|\Sigma|\cdot(|\Sigma|-1)}{2}} & |\Sigma| > 1 \\ 1 & |\Sigma| = 1 \end{cases} \quad (8)$$

where $|\Sigma|$ is the number of identified candidate services and, in the evaluation of $pcoup(\mathcal{BP})$, for each pair $\mathcal{S}_i$ and $\mathcal{S}_j$ (with $\mathcal{S}_i \neq \mathcal{S}_j$) both $coup(\mathcal{S}_i, \mathcal{S}_j)$ and $coup(\mathcal{S}_j, \mathcal{S}_i)$ are evaluated. Process cohesion and coupling coefficients are combined in the coupling/cohesion ratio Γ, that must be minimized

$$\Gamma = \frac{pcoup(\mathcal{BP})}{pcoh(\mathcal{BP})}. \quad (9)$$

Given the task dependency values shown in Fig. 2 and the candidate services identified in the previous step, the service cohesion and coupling values are shown in Fig. 4, where service cohesion values are put on the diagonal. For example, if we consider $\mathcal{S}_B = \texttt{seq}(t_2, t_3)$ and $\mathcal{S}_C = \texttt{seq}(t_4) = t_4$, since $\tau(t_2, t_4) = 0.4$ and $\tau(t_3, t_4) = 0.86$, then $coup(\mathcal{S}_B, \mathcal{S}_C) = \frac{0.4+0.86}{2\cdot 1} = 0.63$. In the running example, $pcoh(\mathcal{BP}) = 0.877$, $pcoup(\mathcal{BP}) = 0.084$ and $\Gamma = \frac{0.084}{0.877} = 0.09578$.

Aggregation of candidate services to minimize Γ is implemented by the algorithm whose pseudo-code is shown in Fig. 5. □

The current coupling/cohesion ratio is computed (row 2). At each iteration of the procedure, two distinct candidate services $\mathcal{S}_i$ and $\mathcal{S}_j$ are selected to be aggregated (row 6). The selection is performed taking into account both the coupling between identified services and the structure of the process. To be aggregated, $\mathcal{S}_i$ and $\mathcal{S}_j$ must be:

1. contiguous subprocesses in a sequence, or
2. subprocesses in a parallel or alternative execution, or
3. subprocesses within the same loop execution, or
4. subprocesses within the same swimlane.

	S_A	S_B	S_C	S_D	S_E	S_F	S_G	S_H	S_I	S_J	S_K	S_L	S_M	S_N
S_A	1,000	-	-	-	-	-	-	-	-	-	-	-	-	-
S_B	-	0,667	0,629	0,233	-	0,175	-	-	-	0,250	0,250	-	-	-
S_C	-	-	1,000	-	-	-	-	-	0,667	-	-	-	-	-
S_D	-	-	-	0,222	0,667	-	0,444	-	-	-	-	-	-	-
S_E	-	-	-	-	1,000	-	-	-	-	-	-	-	-	0,667
S_F	-	-	-	-	-	0,389	0,333	-	-	-	-	0,500	0,500	-
S_G	-	-	-	-	-	-	1,000	1,000	-	-	-	-	-	-
S_H	-	-	-	-	-	-	-	1,000	0,667	-	-	-	-	0,667
S_I	-	-	-	-	-	-	-	-	1,000	-	-	-	-	-
S_J	-	-	-	-	-	-	-	-	-	1,000	-	-	-	-
S_K	-	-	-	-	-	-	-	-	-	-	1,000	-	-	-
S_L	-	-	-	-	-	-	-	-	-	-	-	1,000	-	-
S_M	-	-	-	-	-	-	-	-	-	-	-	-	1,000	-
S_N	-	-	-	-	-	-	-	-	-	-	-	-	-	1,000

coup(S_N,S_G) coh(S_N)

Fig. 4. Service cohesion and coupling values for the running example.

AGGREGATESERVICES algorithm
Inputs: the business process $\mathcal{BP}$ according to Eqn (2);
the matrix $\mathcal{TD}$ of data dependencies between tasks;
the matrix of service cohesion and coupling;
the set Σ of identified candidate services;
Outputs: the refined set Σ of candidate services;

(1) **begin**
(2) compute the coupling/cohesion ratio Γ;
(3) aggregate=TRUE;
(4) **while**(aggregate and $|\Sigma|$ =!1)
(5) $\Sigma' = \Sigma$; // buffer set
(6) select a pair of services $\mathcal{S}_i$ and $\mathcal{S}_j$ in Σ' to aggregate;
(7) **if** $\mathcal{S}_i$ and $\mathcal{S}_j$ not found
(8) aggregate=FALSE;
(9) **break**;
(10) **end if**
(11) aggregate $\mathcal{S}_i$ and $\mathcal{S}_j$ into a new service $\mathcal{S}_{ij}$;
(12) remove $\mathcal{S}_i$ and $\mathcal{S}_j$ from Σ';
(13) add $\mathcal{S}_{ij}$ to Σ';
(14) compute the coupling/cohesion ratio Γ';
(15) **if** ($\Gamma' < \Gamma$)
(16) $\Sigma = \Sigma'$;
(17) $\Gamma = \Gamma'$;
(18) **else**
(19) aggregate=FALSE;
(20) **end if**
(21) **end while**
(22) **end**
...designer's validation ...

Fig. 5. Algorithm that suggests service aggregation to minimize the process coupling/cohesion ratio.

Among the pairs of services that can be aggregated, the ones with the highest coupling are selected first. Conditions 1-4 ensure that the aggregated service is a valid subprocess of $\mathcal{BP}$. In particular,

condition 1) avoids that, for example, a candidate service $\{t_2, t_4\}$ is identified, upsetting the right sequence $\{t_2, t_3, t_4\}$;
condition 2) avoids, for example, that a candidate service $\{t_5, t_6\}$ is identified, splitting the condition checking into two parts, executed both inside and outside the service;
condition 3) avoids that, for example, a candidate service $\{t_{10}, t_{11}\}$ is identified, breaking the loop involving tasks t_{11}, t_{12}, t_{13} and t_{14}. Finally,
condition 4) avoids the identification of the service across different partners.

The conditions are checked by relying on the $\mathcal{BP}$ representation according to Eqn (2).

The services $\mathcal{S}_i$ and $\mathcal{S}_j$ are aggregated into a new service $\mathcal{S}_{ij}$ (rows 11-13). After aggregating the two services, the new coupling/cohesion ratio Γ' is evaluated (row 14) and, if $\Gamma' < \Gamma$, the set of identified services is updated (rows 15-17) and the aggregation procedure is repeated. Otherwise, the *aggregate* flag is put to FALSE and the algorithm stops (rows 18-20). The procedure continues until the coupling/cohesion ratio Γ does not further decrease or all the candidate services identified in the previous step are aggregated (see condition on row 4). It is worth mentioning that, after the execution of the algorithm, services to be aggregated are suggested to the designer, who may accept or reject the suggestion. The iterative execution of the aggregation algorithm produces an aggregation tree, whose instantiation for the running example is shown in Fig. 6, with corresponding variations in the coupling/cohesion ratio from $\Gamma 1$ to $\Gamma 6$. After aggregation of tasks t_{17}, t_{18}, t_{19} and t_{20} the value of Γ starts to increase and the aggregation procedure stops.

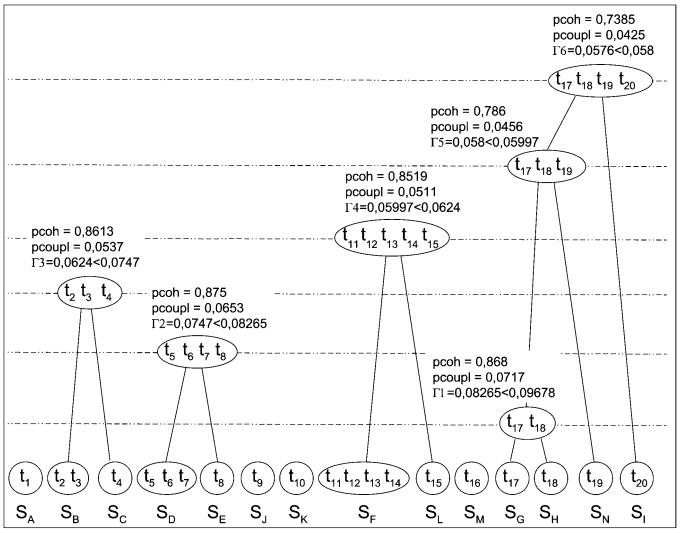

Fig. 6. Service aggregation tree for the running example.

6 CANDIDATE SERVICE RECONCILIATION

Candidate service reconciliation is applied to service descriptors defined according to Eqn (3). It is possible that some services in the set Σ overlap, thus denoting: 1) the execution of the same or similar tasks in different points of the process; 2) slightly different versions of the same service [1]. In order to support the designer in identifying overlapping services, proper coefficients have been introduced to calculate similarity between service descriptors. Such coefficients enable service similarity computation independently of service granularity (i.e., the number of tasks). Similarity between services is estimated by considering the business objects on which services operate and actions performed on such objects.

Let us consider two business objects bo_i and bo_j. Similarity between them, denoted with $BOSim(bo_i, bo_j)$, is evaluated as a combination of their name similarity ($NameSim$) and structural similarity ($StrucSim$)

$$BOsim(bo_i, bo_j) = \alpha \cdot NameSim(nbo_i, nbo_j) + \beta \cdot StructSim(bo_i, bo_j) \in [0, 1] \quad (10)$$

where nbo_i and nbo_j are the names of bo_i and bo_j, respectively, $0 \leq \alpha, \beta \leq 1$, $\alpha + \beta = 1$ are weights used to balance the relevance of each kind of similarity (in the examples we chose $\alpha = \beta = 0.5$ since we equally weight name and structural similarity). Generally speaking, the name similarity $NameSim \in [0, 1]$ function compares two names and evaluates how similar they are with respect to their closeness within the semantic dictionary. In the literature, several approaches are proposed to compute the similarity between terms, by relying on the relationships between terms in a given reference ontology. We do not commit to a specific similarity function. In our experimentation, we used the function defined in [9], given its proved efficacy in business process reengineering [10]

and service discovery [3]. Here we simply state that nbo_i and nbo_j are transformed in two sets of terms by applying stop word management, camel case processing and stemming. Similarity of each pair of terms (one from nbo_i and one from nbo_j) is evaluated by seeking a path of relationships between them in the semantic dictionary. Finally, pairs of terms to be considered for $NameSim$ computation are selected according to a maximization function that relies on the assignment in bipartite graphs and selected term similarity values are combined through the Dice formula [30] to obtain $NameSim$.

The structural similarity $StructSim(bo_i, bo_j)$ is evaluated in different ways depending on the structure of the business objects

$$\begin{cases} TypeComp(tbo_i, tbo_j) & bo_i, bo_j \ simple \\ 0 & bo_i \ simple, \ bo_j \ structured \\ & or \ viceversa \\ \frac{2 \cdot \sum_{p_i, p_j} BOSim(p_i, p_j)}{|\mathcal{P}(bo_i)| + |\mathcal{P}(bo_j)|} & bo_i, bo_j \ structured \end{cases} \quad (11)$$

where tbo_i and tbo_j are the types of bo_i and bo_j, respectively, and $TypeComp(tbo_i, tbo_j)$ is a function which measures the compatibility between types, based on the approach proposed in [9]; $p_i \in \mathcal{P}(bo_i)$, $p_j \in \mathcal{P}(bo_j)$ are the attributes of bo_i and bo_j, respectively, $|\cdot|$ denotes the set cardinality. If bo_i and bo_j are structured objects, $BOSim$ is recursively applied to their properties, one from $\mathcal{P}(bo_i)$ and one from $\mathcal{P}(bo_j)$. If $p_i \in \mathcal{P}(bo_i)$ presents a similarity with more than one $p_j \in \mathcal{P}(bo_j)$, the pairs of p_i and p_j to be considered in computation are selected by applying the same maximization function used for $NameSim$ computation.

For example, let us consider the services $\mathcal{S}_i = \{t_5, t_6, t_7, t_8\}$ and $\mathcal{S}_j = \{t_{11}, t_{12}, t_{13}, t_{14}, t_{15}\}$ identified in the previous step for the running example. The descriptors of the two services are the following:

$$\begin{aligned} d_{\mathcal{S}_i} &= \{\texttt{analyze}, \texttt{backbone}, \texttt{component}, \texttt{produce}, \\ &\quad \texttt{collect}, \texttt{warehouse}, \texttt{deliver}\} \\ IN_{\mathcal{S}_i} &= \{\texttt{BackboneComp.Spec.}\} \\ OUT_{\mathcal{S}_i} &= \{\texttt{BackboneComp.}, \texttt{DeliveryNote}\} \\ R_{\mathcal{S}_i} &= \{\texttt{SofaManufacturer.ManufacturingDept}\} \\ A_{\mathcal{S}_i} &= \{R(\texttt{BackboneComp.Spec.}), \\ &\quad C(\texttt{BackboneComp.}), C(\texttt{DeliveryNote})\} \\ d_{\mathcal{S}_j} &= \{\texttt{analyze}, \texttt{accessory}, \texttt{order}, \texttt{material}, \\ &\quad \texttt{produce}, \texttt{collect}, \texttt{warehouse}, \texttt{deliver}\} \\ IN_{\mathcal{S}_j} &= \{\texttt{AccessoryComp.Spec.}\} \\ OUT_{\mathcal{S}_j} &= \{\texttt{Accessory}\} \\ R_{\mathcal{S}_j} &= \{\texttt{AccessoriesProvider.ManufacturingDept}\} \\ A_{\mathcal{S}_j} &= \{R(\texttt{AccessoryComp.Spec.}), C(\texttt{Accessory})\}. \end{aligned}$$

The $\mathcal{S}_i$ and $\mathcal{S}_j$ inputs/outputs are the following:

$\texttt{BackboneComp.Spec.}$
$= \{\texttt{quantity :: number}, \texttt{description :: text}\}$
$\texttt{BackboneComp.}$
$= \{\texttt{specification :: BackboneComp.Spec.}, \texttt{price :: float}\}$
$\texttt{DeliveryNote}$
$= \{\texttt{address :: string}, \texttt{totalPrice :: float}\}$
$\texttt{AccessoryComp.Spec.}$
$= \{\texttt{name :: string}, \texttt{model :: string}, \texttt{description :: string}, \texttt{quantity :: integer}\}$
$\texttt{Accessory}$
$= \{\texttt{specification :: AccessoryComp.Spec.}, \texttt{price :: float}\}.$

The $NameSim$ values for this example are shown in the following table (note that $NameSim(t_i, t_j) = 1.0$ if $t_i = t_j$): Since $TypeComp(\texttt{number}, \texttt{integer}) = 1.0$ and $TypeComp(\texttt{string}, \texttt{text}) = 1.0$, the following similarity values follow:

$BOSim(\texttt{quantity :: number}, \texttt{quantity :: integer})$
$= 0.5 \cdot 1.0 + 0.5 \cdot 1.0 = 1.0$
$BOSim(\texttt{description :: text}, \texttt{description :: string})$
$= 0.5 \cdot 1.0 + 0.5 \cdot 1.0 = 1.0$
$StructSim(\texttt{BackboneComp.Spec.}, \texttt{AccessoryComp.Spec.})$
$= \frac{2 \cdot [1.0 + 1.0]}{2 + 4} = 0.67$
$BOSim(\texttt{BackboneComp.Spec.}, \texttt{AccessoryComp.Spec.})$
$= 0.5 \cdot 1.0 + 0.5 \cdot 0.67 = 0.66$
$BOSim(\texttt{specification :: BackboneComp.Spec.}, \texttt{specification :: AccessoryComp.Spec.})$
$= 0.5 \cdot 1.0 + 0.5 \cdot 0.66 = 0.83$
$BOSim(\texttt{BackboneComp.}, \texttt{Accessory})$
$= \frac{2 \cdot [0.83 + 1.0]}{2 + 2} = 0.92.$

Similarity of two services $\mathcal{S}_i$ and $\mathcal{S}_j$ based on the business objects on which they operate is given by

$$OSim(\mathcal{S}_i, \mathcal{S}_j) = \frac{2 \cdot \sum_{in_i, in_j} BOsim(in_i, in_j)}{|IN_{\mathcal{S}_i}| + |IN_{\mathcal{S}_j}|} + \frac{2 \cdot \sum_{out_i, out_j} BOsim(out_i, out_j)}{|OUT_{\mathcal{S}_i}| + |OUT_{\mathcal{S}_j}|} \in [0, 2] \quad (12)$$

where $in_i \in IN_{\mathcal{S}_i}$, $in_j \in IN_{\mathcal{S}_j}$, $out_i \in OUT_{\mathcal{S}_i}$, $out_j \in OUT_{\mathcal{S}_j}$. Given two CRUD actions $a_i \in A_{\mathcal{S}_i}$ and $a_j \in A_{\mathcal{S}_j}$ performed on business objects bo_i and bo_j, respectively, the similarity between a_i and a_j, denoted with $ASim(a_i, a_j) \in [0, 1]$, is calculated as follows:

$$ASim(a_i, a_j) = \begin{cases} 0 & if \ a_i \neq a_j \\ BOSim(bo_i, bo_j) & if \ a_i = a_j. \end{cases} \quad (13)$$

Similarity between services based on actions they perform, denoted with $FSim$ (*functional similarity*), is evaluated as follows:

$$FSim(\mathcal{S}_i, \mathcal{S}_j) = Sim(d_{\mathcal{S}_i}, d_{\mathcal{S}_j}) + \frac{2 \cdot \sum_{a_i, a_j} ASim(a_i, a_j)}{|A_i| + |A_j|} \in [0, 2] \quad (14)$$

where $Sim(d_i, d_j)$ is the similarity between task descriptions, obtained by applying the Dice formula to the similarities between terms in d_i and d_j. The $OSim$ and $FSim$ coefficients

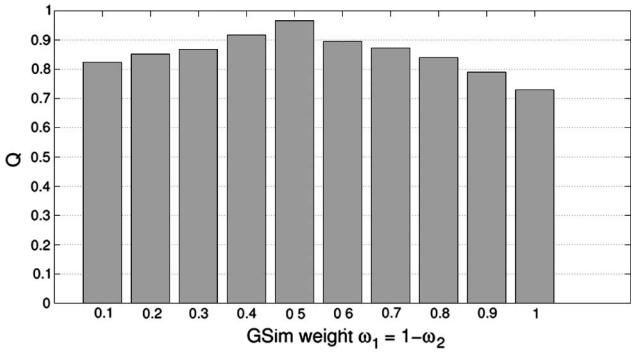

Fig. 7. Variation in the quality of service reconciliation with respect to weights ω_1 and ω_2.

are normalized in the range $[0, 1]$ and linearly combined to obtain a *Global SimilarityGSim*, defined as follows:

$$GSim(\mathcal{S}_i, \mathcal{S}_j) = \omega_1 \cdot NormOSim(\mathcal{S}_i, \mathcal{S}_j) + \omega_2 \cdot NormFSim(\mathcal{S}_i, \mathcal{S}_j) \in [0, 1] \quad (15)$$

where: $0 \leq \omega_1, \omega_2 \leq 1$ and $\omega_1 + \omega_2 = 1$ are weights used to assess relevance to each kind of similarity. If $GSim(\mathcal{S}_i, \mathcal{S}_j)$ is equal or greater than a similarity threshold $\gamma \in [0, 1]$, then the two candidate services are proposed to the designer for their reconciliation. The designer may analyze the proposed similar services and may decide to merge them or maintain them as distinct candidate services. The setup of weights ω_1 and ω_2 and of the threshold γ will be detailed in the experimental evaluation section. The $GSim$ coefficient for services $\mathcal{S}_i$ and $\mathcal{S}_j$ considered in the example is the following:

$$Sim(d_{\mathcal{S}_i}, d_{\mathcal{S}_j}) = \frac{2 \cdot [1.0+1.0+1.0+1.0+1.0+0.64]}{7+8} = 0.75$$

$$FSim(\mathcal{S}_i, \mathcal{S}_j) = 0.752 + \frac{2 \cdot [0.66 + 0.915]}{2+3} = 1.38$$

$$OSim(\mathcal{S}_i, \mathcal{S}_j) = \frac{2 \cdot 0.66}{1+1} + \frac{2 \cdot 0.915}{2+1} = 1.27$$

$$GSim(\mathcal{S}_i, \mathcal{S}_j) = 0.5 \cdot \frac{1.382}{2} + 0.5 \cdot \frac{1.27}{2} = 0.66.$$

Indeed the similarity of these two services was quite evident: they both perform analysis, production or collection from warehouse, delivering. Anyway, the manual identification of service redundancies is made difficult by different granularities ($\mathcal{S}_i$ contains four tasks, while $\mathcal{S}_j$ contains five tasks) and by the overall process complexity (see tests in the next section).

7 System Evaluation

A prototype tool supporting the P2S methodology, called P2Stool, has been developed in Java, within an Eclipse BPMN plug-in (http://eclipse.o2rg/bpmn/).[2] We ran several experiments, in order to test: 1) the quality of service identification obtained with the support of the P2Stool; 2) how the P2Stool is able to mitigate the gap between middle-level and high-level skilled users to perform service identification; 3) the performance of the P2Stool in terms of time consumed to complete each phase of the methodology. All the experiments have been performed on an Intel laptop, with a 2.53 GHz Core 2 CPU, 2GB RAM and Linux operating system.

7.1 Experimental Setup

To run experiments, we generated a dataset containing 20 processes. In particular, the dataset has been built by considering: 1) the size of the business processes, computed as the number of simple tasks (ranging from 5 to 40); 2) their structural complexity, computed as the number of parallel and alternative branches and the number of loops (ranging from 3 to 40); 3) their data flow complexity, computed as the average number of inputs/outputs of simple tasks in the processes (ranging from 1 to 7). A setup has been performed on the weights and threshold used for reconciliation of similar services. To setup the threshold γ for $GSim$ computation (see Equation (15)), we randomly selected the decompositions of ten processes, obtained through the identification phase, and we asked the domain expert to manually detect pairs of similar services. We repeated the experiment by computing $GSim$ with the P2Stool by varying the threshold $\gamma \in [0, 1]$. Let be χ ($\|\chi\| = m$) the initial number of services in the decomposition, the total number of pairwise comparisons between services to evaluate their similarity is $n = \frac{m \cdot (m-1)}{2}$; let be fp the number of false positives identified by the P2Stool and fn the number of false negatives: the value $Q = 1 - \frac{fp+fn}{n}$ has to be maximized. As expected, for too high values for γ, only very similar services, with very close descriptors, are proposed to be reconcilied, that is, more redundant services are identified as separated ones, thus increasing the complexity of the service implementation and deployment. On the other hand, too low γ values increase the complexity during service identification, since very different services are proposed to the designer for reconciliation and very complex merging procedures are required. Finally, we chose $\gamma = 0.5$. A setup for weights ω_1 and ω_2 with $\gamma = 0.5$ has been performed too, by varying $\omega_1 \in [0, 1]$. Fig. 7 displays the results. Unbalanced ω_1 and ω_2 weighting decreases the quality of $GSim$ computation. For instance, if we weight the I/O similarity $OSim$ the most, also services containing tasks that are functionally different, but operate on the same I/Os, are candidates for reconciliation (see for instance t_{15} and t_{16} in the running example). On the other hand, if we weight the functional similarity $FSim$ the most, we strongly rely on tasks who compose services, thus reducing the possibility of merging together services composed of different tasks or structures (see, for example, services $\mathcal{S}_i$ and $\mathcal{S}_j$ considered in Section 6).

7.2 Quality of Service Identification

We asked a domain expert to manually apply the service identification on each process in the dataset, following the guidelines described in [29]. We computed the Γ value on the obtained set of candidate services (see Equation (9)). Let us denote with $\overline{\Gamma}_i$ the value on the set of identified services obtained by the domain expert for the i-th process in the

2. A demo video of the P2Stool can be found here: www.ing.unibs.it/~bianchin/P2Stool_demo.avi.

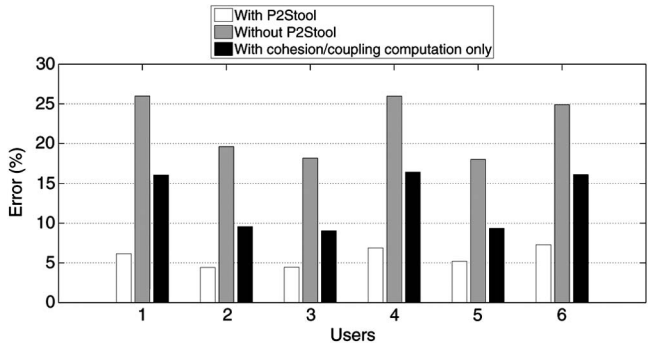

Fig. 8. Quality of service identification with and without P2Stool support and with the support of a system that only provides the computation of coupling/cohesion metrics.

dataset $(i = 1 \dots 20)$. We monitored six users, who performed the service identification on each process in the dataset with and without the support of the P2Stool and with the support of a system that only provides the computation of coupling/cohesion metrics (see for example [31]). Users have either middle-level (users 1, 4 and 6) or high-level skill in business process modeling (users 2, 3 and 5). Users with low-level skill have not been considered since they are not among the target users of our methodology. We computed the Γ_i^u value for the service identification performed on the i-th process by user u $(i = 1 \dots 20)$. Finally, for each user u and each process in the dataset, we evaluated the percentage error $(|\Gamma_i^u - \overline{\Gamma}_i|/\overline{\Gamma}_i) \cdot 100$.

In Fig. 8 we show the average results for the six users on 20 processes of the dataset. The main difficulty for the user supported by the system that only provides the computation of coupling/cohesion metrics is to perform the identification of an initial set of services, that is further evaluated in terms of cohesion and coupling. Our system, on the other hand, also suggests such an initial identification, according to the value analysis. Fig. 8 shows that the adoption of P2Stool also mitigates the skill gap between different users who perform service identification. Fig. 9 shows how process size, structural complexity and data flow complexity affect the quality of service identification with and without the support of P2Stool. To perform experiments in Fig. 9a, we selected ten processes from the dataset with comparable structural complexity (20 ± 2 gateways and loops) and data flow complexity (4 ± 1 average I/Os) and increasing process size (from 20 to 38). Each process has been decomposed by six users and figure shows the average results. Experiments whose results have been shown in Figs. 9b and 9c have been performed in similar way, limiting the process size to 20 ± 2 simple tasks and varying the structural and the data flow complexity, respectively. Figures show how the factor that influences the users the most during service identification without P2Stool is the structural complexity. This intuitively could be inferred from the fact that gateways and loops are the costructs that visually make more complex the BPMN representation of processes.

7.3 Performance Evaluation

Performance has been evaluated to check the efficiency of the P2Stool in terms of time consumed to complete each phase of the methodology. Since the methodology has been meant to be executed at design-time, the delays introduced by the designer who is in charge of validating the suggestions provided by the P2Stool have not been considered. Since the steps are mainly based on task I/Os and data dependencies, we expect that execution time is influenced by the size of the business processes and by their data flow complexity, and is less influenced by their structural complexity. In this experiment, the goal was to evaluate system performances and not its quality. Therefore, we generated a new synthetic dataset starting from the 20 processes considered in the previous experiments. In Fig. 10a each execution time measure has been taken on ten processes with the same process size and comparable structural complexity (20 ± 2) and data flow complexity (4 ± 1), then the average has been computed. Experiments in Figs. 10b and 10c have been performed in a similar way for processes with comparable size (20 ± 2).

We observed a small linear increase of the execution time as the number of simple tasks increases (Fig. 10a). Nevertheless, also for the biggest processes, the system scales well and takes no more than 4 seconds during the most time-consuming phase (that is, the identification of similar services), that is an acceptable result for a tool to be used at design-time. Results in Fig. 10c show, as expected, that the average number of I/Os affects the execution time of the P2Stool, although the scalability is still preserved, while the structural complexity slightly affects the execution time, with only very marginal differences in execution time (Fig. 10b).

8 Related Work

Service identification is considered as a precondition for a successful implementation and governance of SOA [7]. It has a direct impact on the composability of loosely-coupled services and the reusability of individual services in

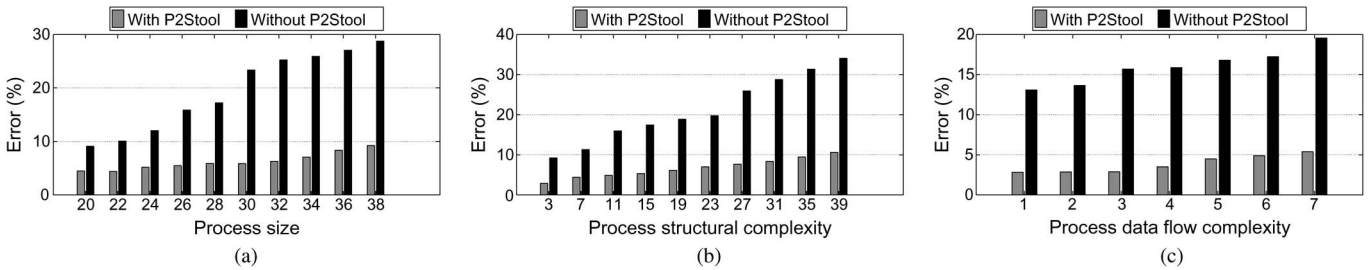

Fig. 9. Variation in the quality of service identification with respect to. (a) Process size. (b) Structural complexity. (c) Data flow complexity.

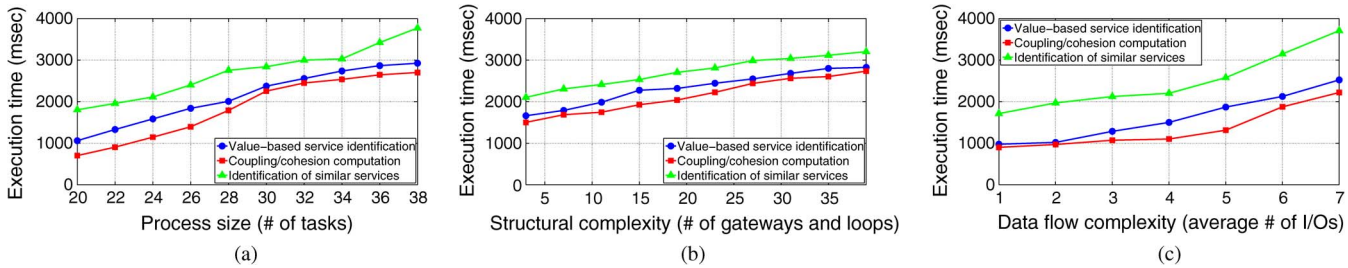

Fig. 10. Execution time of the P2Stool with respect to. (a) Process size. (b) Structural complexity. (c) Data flow complexity.

different contexts [17]. Anyway, several comparisons highlight that the issue of identifying services using an appropriate granularity has not been satisfactorily addressed yet [7], [12], [19].

In Table 4 a comparison between the P2S methodology and some of the most representative service identification methods is shown. Analysis dimensions in Table 4 are the ones suggested in [6]: the strategy (top-down, bottom-up, meet-in-the-middle), the use of quantitative metrics to evaluate the identification phase, the availability of procedural guidelines and/or tools, and the type of performed validation. Most of the consolidated approaches either suggest guidelines for service identification without providing quantitative metrics to evaluate the quality of the identified services or use a limited set of metrics, without providing a service identification procedure. The main contribution of the P2S approach is the combination of a methodological perspective with metrics which quantitatively guide the designer through the identification phase. A first attempt to provide this kind of approach has been made in [4]. This paper validates the methodological phases exposed in [4] and provides a support tool. Moreover, some aspects of the methodology have been improved: the service refinement step now takes into account also the business process structure, while service reconciliation can be performed here regardless the granularity (i.e., the number of tasks) of the compared services.

8.1 Service Identification Methodologies

For what concerns service identification strategy, top-down approaches (also known as *domain decomposition*) focus on the analysis of business domains and business process modeling to identify services, components and flows that will be used to orchestate them, while bottom-up and meet-in-the-middle strategies are especially useful in environments where component services are relatively fixed and processes are designed on the basis of the available services [11]. [23] proposes a methodology to define development principles for Web services on the basis of the business processes that can be assembled into business scenarios. [16] and [18] propose other top-down methodologies based on a goal-based approach for the identification of service composition, without quantitative models to support the analysis. The approach described in [20] provides a set of measures, combined as a multi-objective problem solving, that allow designers to validate service identification. With respect to this approach, the P2S methodology provides a step-by-step procedure to assist the designer during the identification. As shown by the experimental evaluation, the structure of the business process is taken into account and a better feedback is given to the designer. In [28], several methods are combined in order to identify services starting from an analysis of organizational domain and processes. The designer is guided by the order through which the different techniques should be used and by some tips for the evaluation of the results. Finally, in [17] service identification is performed by considering a business process and using a clustering algorithm to merge the process tasks included in a single service. The use of cohesion/coupling metrics in the service identification has been suggested in several contributions such as [2], [10], [31], but these approaches

TABLE 4
Comparison of the P2S Methodology with Other Service Identification Approaches

	Development direction	Use of quantitative metrics	Availability of procedural guidelines	Availability of tool support	Validation
P2S	Top-down	YES (value-based analysis, coupling/cohesion evaluation, similarity evaluation, structural analysis)	YES (service identification, refinement and reconciliation)	YES	Case study and Quality evaluation
[23]	Top-down	NO	YES (high-level development principles)	NO	NO
[16]	Top-down	NO	YES (goal-based procedure)	NO	Example
[20]	Top-down	YES	NO	YES	Case Study
[28]	Top-down	YES	NO	NO	Case Study
[13]	Bottom-up	YES (logic-based matchmaking)	NO	NO	NO
[21]	Bottom-up	YES (runtime service selection)	NO	NO	Project
[27]	Bottom-up	YES (process analysis based on templates, QoS-based service selection)	NO	YES	Example
[11]	Meet-in-the-middle	YES (service comparison based on logical affinity, redundancy analysis)	YES	NO	Case study
[17]	Top down	YES (clustering algorithm)	YES	NO	Case Study
[18]	Top-down	NO	YES	NO	Case study

mainly propose techniques to select the most suitable identification starting from different available solutions.

Bottom-up approaches focus on existing IT assets and services; quantitative metrics such as cohesion and coupling are used to evaluate the quality of existing assets and, eventually, to perform reengineering strategies [13], [21], [27]. The enrichment of the process description is addressed by [13], in which the main issue is the identification of the Web services that match the designer's specification. The authors explain how the use of an appropriate language (i.e., WSMO) gives a unified view on business processes allowing designers to easily link process activities to services. In [21] an abstract process represents a Web process whose control and data flow are defined at design time, but the actual services are not chosen until run-time. In [27] the notion of process template is introduced. Process templates are reusable business process skeletons that are devised to reach particular goals and are made up of states and transitions. A state corresponds to the execution of a service that is member of a Web service community. A community is a collection of services with a common functionality, but different non-functional properties (e.g., QoS) that drive the selection of the most suitable service at run-time. *Meet-in-the-middle* approaches, also referred to as *goal-service modeling*, decompose a generalized statement of business goals relevant to the scope of the business process into subgoals that must be met by existing services. In [11] a meet-in-the-middle strategy is applied. The last step is service refactoring and rationalization. The refactoring is performed by grouping lower-level services that have some kind of logical affinity. Subsequently, the rationalization is applied as a set of criteria to resolve whether a candidate service should be exposed, based on the evaluation of business alignment, composability, redundancy.

8.2 Validation of Service Identification

In the literature, most of the service identification approaches are evaluted putting them into practice [12]. Some approaches are validated showing their effectiveness in real life projects or experimenting them in case studies. Other approaches provide only some examples to explain the proposed service identification method. For the validation of the P2S methodology we have used a case study in order to better explain how the methodology works and we also evaluate the perfomance and quality of the P2S tool by using a user-based study.

9 Concluding Remarks

The P2S methodology aims at providing a semi-automatic approach to support designers to analyze a business process and identify subset of functionalities that can be exported as services. The methodology is designed to be applied in the first steps of a SOA lifecycle, in a top-down approach, or in any case in which a portfolio of available services is not present and the goal is to identify suitable sub-processes, that fit well defined properties, such as low coupling and high cohesion. The P2S methodology will be extended to address issues related to meet-in-the middle approaches, i.e., in situations in which the service identification should be driven also by the availability of existing services. In particular, in [5] a work-in-progress project for meet-in-the middle service identification is described, where available services are retrieved by relaxing the identification obtained through the P2S methodology by moving on the aggregation tree. This project shows how the results of the methodology can be used as a starting point to design a tool that is able to support the designer through all the activities of the SOA lifecycle. Future efforts will also be devoted to the study of the relationship between service identification and process abstraction, that is an orthogonal issues whose goal is to provide different representations of the same model according to different abstraction levels, in order to make the process visualization more readable by BPM experts [24]. Service identification requires a detailed specification of the process to deploy it in a SOA context. However, metrics exposed in this paper could be useful to define new criteria to perform business process abstraction and will be investigated. Finally, another open issue concerns the consistency maintenance between high-level business process models, as used by BPM experts, and service composition workflow, as used by IT departments. To this aim, in [8] the (manual) definition of mappings between these two levels is proposed. Exploitation of the identification performed by the P2S methodology to setup mappings similar to the ones suggested in [8] will be investigated.

Acknowledgment

This paper has been partially funded by the TEKNE FIRB project of the Italian Ministry of Education, University and Research (http://www.tekne-project.it) and the European Network of Excellence S-Cube (http://www.s-cube-network.eu).

References

[1] V. Andrikopoulos, S. Benbernou, and M. Papazoglou, ''On the Evolution of Services,'' *IEEE Trans. Softw. Eng.*, vol. 38, no. 3, pp. 609-628, May/June 2012.

[2] L. Baresi, F. Casati, S. Castano, M. Fugini, I. Mirbel, and B. Pernici, ''WIDE Workflow Development Methodology,'' in *Proc. WACC*, 1999, pp. 19-28.

[3] D. Bianchini, V. De Antonellis, and M. Melchiori, ''Flexible Semantic-Based Service Matchmaking and Discovery,'' *World Wide Web J.*, vol. 11, no. 2, pp. 227-251, June 2008.

[4] D. Bianchini, C. Cappiello, V. De Antonellis, and B. Pernici, ''P2S: A Methodology to Enable Inter-Organizational Process Design Through Web Services,'' in *Proc. CAiSE*, 2009, pp. 334-348.

[5] D. Bianchini, F. Pagliarecci, and L. Spalazzi, ''From Service Identification to Service Selection: An Interleaved Perspective,'' in *Proc. Formal Model., Actors, Open Systems, Biol. Syst.*, 2011, pp. 223-240.

[6] N. Bieberstein, R.G. Laird, K. Jones, and T. Mitra, *Executing SOA: A Practical Guide for the Service-Oriented Architecture.* Boston, MA, USA: Pearson Education, 2008.

[7] R. Boerner and M. Goeken, ''Service Identification in SOA Governance Literature Review and Implications for a New Method,'' in *Proc. IEEE DEST*, 2009, pp. 588-593.

[8] S. Buchwald, T. Bauer, and M. Reichert, ''Bridging the gap Between Business Process Models and Service Composition Specifications,'' in *Proc. Serv. Life Cycle Tools Technol., Methods, Trends Adv.*, 2011, pp. 124-153.

[9] S. Castano, V. De Antonellis, and S. De Capitani di Vimercati, ''Global Viewing of Heterogeneous Data Sources,'' *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 2, pp. 277-297, Mar./Apr. 2001.

[10] S. Castano, V. De Antonellis, and M. Melchiori, ''A Methodology and Tool Environment for Process Analysis and Reengineering,'' *Data Knowl. Eng.*, vol. 31, no. 3, pp. 253-278, Nov. 1999.
[11] S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, K. Holley, and A. Arsanjani, ''SOMA: A Method for Developing Service-Oriented Solutions,'' *IBM Syst. J.*, vol. 47, no. 3, pp. 377-396, July 2008.
[12] Q. Gu and P. Lago, ''Service Identification Methods: A Systematic Literature Review,'' in *Proc. ServiceWave*, 2010, pp. 37-50.
[13] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel, ''Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management,'' in *Proc. ICEBE*, 2005, pp. 535-540.
[14] S. Inaganti and G.K. Behara, ''Service Identification: BPM and SOA Handshake,'' *BPTrends*, vol. 3, pp. 1-12, Mar. 2007.
[15] J.L.G. Dietz, ''The Atoms, Molecules and Fibers of Organizations,'' *Data Knowl. Eng.*, vol. 47, no. 3, pp. 301-325, Dec. 2003.
[16] R.S. Kaabi, C. Souveyet, and C. Rolland, ''Eliciting Service Composition in a Goal Driven Manner,'' in *Proc. ICSOC*, 2004, pp. 308-315.
[17] Y. Kim and K. Doh, ''Formal Identification of Right-Grained Services for Service-Oriented Modeling,'' in *Proc. WISE*, 2009, pp. 261-273.
[18] T. Kohlborn, A. Korthaus, T. Chan, and M. Rosemann, ''Identification and Analysis of Business and Software Services—A Consolidated Approach,'' *IEEE Trans. Serv. Comput.*, vol. 2, no. 1, pp. 50-64, Jan. 2009.
[19] A. Krammer, B. Heinrich, M. Henneberger, and F. Lautenbacher, ''Granularity of Services—An Economic Analysis,'' *Bus. Inf. Syst. Eng.*, vol. 3, no. 6, pp. 345-358, June 2011.
[20] Q. Ma, N. Zhou, Y. Zhu, and H. Wang, ''Evaluating Service Identification with Design Metrics on Business Process Decomposition,'' in *Proc. SCC*, 2009, pp. 160-167.
[21] R. Mulye, J. Miller, K. Verma, K. Gomadam, and A. Sheth, ''A semantic Template Based Designer for Web Processes,'' in *Proc. ICWS*, 2005, pp. 461-469.
[22] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, ''Service-Oriented Computing: A Research Roadmap,'' *Int. J. Cooper. Inf. Syst.*, vol. 17, no. 2, pp. 223-255, June 2008.
[23] M.P. Papazoglou and W.J. van den Heuvel, ''Business Process Development Life Cycle Methodology,'' *Commun. ACM*, vol. 50, no. 10, pp. 79-85, Oct. 2007.
[24] A. Polyvyanyy, S. Smirnov, and M. Weske, ''Business Process Model Abstraction,'' in *Proc. Int. Handbook Business Process Managem.*, 2010, pp. 149-166.
[25] M.A. Serhani, N. Al-Qirim, and A. Benhareef, ''Enterprise Services (Business) Collaboration Using Portal and Soa-Based Semantics,'' in *Proc. DEST*, 2010, pp. 450-455.
[26] M. Sharifi, S. Mansour, and P. Jamshidi, ''To Establish Enterprise Service Model from Enterprise Business Model,'' in *Proc. SCC*, 2008, pp. 93-100.
[27] Q.Z. Sheng, B. Benatallah, Z. Maamar, M. Dumas, and A.H.H. Ngu, ''Enabling Personalized Composition and Adaptive Provisioning of Web Services,'' in *Proc. CAiSE*, 2004, pp. 322-337.
[28] H.M. Shirazi, N. Fareghzadeh, and A. Seyyedi, ''A Combinational Approach to Service Identification in SOA,'' *J. Appl. Sci.*, vol. 5, no. 10, pp. 1390-1397, Oct. 2009.
[29] R. Sindhgatta, B. Sergupta, and K. Ponnalagu, ''Measuring the Quality of Service Oriented Design,'' in *Proc. ICSOC-ServiceWave*, 2009, pp. 485-499.
[30] C.J. van Rijsbergen, *Information Retrieval*. London, U.K.: Butterworth, 1979.
[31] I. Vanderfeesten, H.A. Reijers, and W.M.P. van der Aalst, ''Evaluating Workflow Process Designs Using Cohesion and Coupling Metrics,'' *Comput. Ind.*, vol. 59, no. 5, pp. 420-437, May 2008.