P2S: A Methodology to Enable Inter-organizational Process Design through Web Services

Devis Bianchini[1], Cinzia Cappiello[2], Valeria De Antonellis[1], and Barbara Pernici[2]

[1] University of Brescia
{bianchin,deantone}@ing.unibs.it
[2] Politecnico of Milan, Milan (Italy)
{cappiell,pernici}@elet.polimi.it

Abstract. With the advent of Service Oriented Architecture organizations have experienced services as a platform-independent technology to develop and use simple internal applications or outsource activities by searching for external services, thus enabling inter-organizational interactions. In this scenario, services are units of work provided by service providers and offered to the other organizations involved in a collaborative business process. Collaboration should be facilitated by guaranteeing a homogeneous description of services at the right level of granularity. We propose a methodology to support the designer of a business process in the identification of services that compose the process itself. The methodology should allow collaborative partners to standardize process modelling through component services, enabling effective inter-organizational service discovery. The methodology is presented by means of a running example in a real case scenario.

Keywords: service-based process decomposition, service-based collaborative processes.

1 Introduction

Internet and related technologies demonstrate that information systems and IT can provide a strategic platform to support the collaboration among Internetworked Enterprises (IE) [1], that is, borderless organizations that share applications, services and knowledge and whose processes are transformed and integrated with the ones of their partners. Heterogeneity of such collaborative environments implies the adoption of standards and infrastructures to communicate. With the advent of Service Oriented Architecture (SOA), organizations have experienced services as a platform-independent technology to develop and use simple internal applications or outsource activities by searching for external services, thus enabling inter-organizational interactions. In particular, IE can share their own applications by using the Software as a Service paradigm

P. van Eck, J. Gordijn, and R. Wieringa (Eds.): CAiSE 2009, LNCS 5565, pp. 334–348, 2009.

(SaaS). They can place their own implemented functionalities at the other organizations' disposal by creating services and providing them across the Internet, thus designing operating information systems able to connect IEs each other. In this scenario, services are units of work provided by service providers and offered to the other organizations involved in a collaborative business process. By eliminating the need to install and run the application on the customer's own computer, SaaS alleviates the customer's burden of software maintenance, ongoing operation and support. In such a way, the adoption of SOA technology also enables small and medium enterprises (SMEs) to join IE networks, since it promises to reduce costs and complexity for connecting systems and business processes. At the heart of the SOA paradigm there is a catalog of available services that can be shared across IE and reused to build up collaborative processes. Collaboration should be facilitated by guaranteeing a homogeneous description of services at the right level of granularity. A uniform level of granularity enables the comparability among different services. In fact, a coarse-grained service (e.g., a service associated with many tasks of a process) could not be compared with an elementary service (e.g., a service that corresponds to a single task).

We propose the P2S (from process to services) methodology to support the designer of a business process in the identification of the component services. The methodology starts from a process represented by means of a workflow-based language (e.g., BPMN) and supports the designer through the semantic annotation of the business process elements and the identification of component services on the basis of the notion of values produced for the actors that collaborate in the process. The methodology also provides metrics to evaluate the quality of the performed decomposition and to guide the designer in improving the solution found, given the widely accepted definition of services as platform-independent, self-contained, loosely coupled units of work. The P2S methodology should allow collaborative partners to standardize process modelling through component services, enabling effective inter-organizational service discovery.

The paper is structured as follows: in Section 2 basic definitions are provided; in Section 3 the proposed methodology is presented and discussed by means of a running example in a real case scenario; in Section 4, the described approach is compared with related solutions proposed in the literature; finally, Section 5 gives some hints about future work and concludes the paper.

2 Basic Definitions

Roughly speaking, a business process $\mathcal{BP}$ can be defined as a combination of a set $\mathcal{T}$ of simple tasks through control structures (e.g., sequence, choice, cycle or parallel) to form composite tasks, also denoted as sub-processes. Each simple task $t_i \in \mathcal{T}$ of the business process $\mathcal{BP}$ is described through the operation that it performs and by its I/O data. Data exchanged between tasks and control flows connecting them are modelled as data dependencies and control flow dependencies, respectively. Processes are usually defined at a business level using a workflow-based notation (e.g., BPMN[1]), independently from implementation

[1] http://www.bpmn.org/

technology and platforms [2]. As usual, a process has an entry point (start event) and one or more stop events. Furthermore, actors participating in the process must also be considered. Actors are represented as abstract entities that interact with the business process as responsible of one or more simple tasks. Actors can be grouped into organizations. In workflow-based notation, task responsibilities are represented through *swimlanes*. For example, BPMN supports swimlanes with two main constructs: *pools*, to represent organizations participating in the process, and *lanes*, that constitute sub-partitions within pools and are used to organize activities which are logically related to each other (e.g., when they are performed by the same department). We can provide the formal definition of a simple task $t_i \in \mathcal{T}$ as follows:

$$t_i = \langle n_{t_i}, IN(t_i), OUT(t_i), f_{t_i}, A_{t_i}, O_{t_i}, \{t_{i-1}\} \rangle \tag{1}$$

where: n_{t_i} is the task name: $IN(t_i)$ and $OUT(t_i)$ are the sets of task inputs and outputs, respectively; $f_{t_i} : IN(t_i) \rightarrow OUT(t_i)$ is the transformation associated with the task; A_{t_i} and O_{t_i} are the actor and the organization responsible for the task, respectively; $\{t_{i-1}\}$ is the set of tasks that precede the task t_i according to control flow dependencies. According to widely accepted data structure diagrams (e.g., UML, XML complex data types) each input $in \in IN(t_i)$ can be described as $\langle n, \mathcal{P} \rangle$, where n is the input name and $\mathcal{P} = \{p_i\}$ a set of properties or attributes that further detail the input data. Outputs can be described in the same manner.

Given two simple tasks t_i and $t_j \in \mathcal{T}$, we say that there is a data dependency from t_j to t_i if $f_{t_i}(OUT(t_j)) \neq f_{t_i}(IN(t_j))$, that is, execution of t_i depends on the execution of t_j. A task $t_i \in \mathcal{T}$ is defined as *dependent* on another task $t_j \in \mathcal{T}$ ($t_j \mapsto t_i$) if all the following conditions holds:

1. there is a path of control flow dependencies from t_j to t_i, that is, there exists a set $\{t_k\} \subseteq \mathcal{T}$ of simple tasks, for $k = 1, \ldots n$, such that $t_j \equiv t_1$, $t_i \equiv t_n$ and, for each t_k, $t_{k-1} \mapsto t_k$;
2. there is a data dependency directly from t_j to t_i.

We introduce the *task dependency graph* $\mathcal{TDG}$ as $\langle \mathcal{T}, \mathcal{TD} \rangle$, where $\mathcal{T}$ is the set of simple tasks and $\mathcal{TD} : \mathcal{T} \times \mathcal{T}$ is the set $\{t_j, t_i\}$ of task dependencies such that $t_j \mapsto t_i$.

Considering cooperation among actors that participate in the process, we define the *data exchange graph* $\mathcal{DEG} = \langle \mathcal{A}, \mathcal{E} \rangle$, where $\mathcal{A}$ is the set of actors, $\mathcal{E} : \mathcal{A} \times \mathcal{A} \times Data$ is the set $\{\langle A_i, A_j, d \rangle\}$ of data exchanges between actors, $Data$ is the union set of inputs and outputs of the simple tasks in the overall process and $d \in Data$ represents a data value that is transferred from actor A_i to actor $A_j \neq A_i$. Exchanged data are those associated with control flows crossing swimlanes boundaries. Among actors we also include the final user, that is the beneficiary of the overall process execution.

Also a service $\mathcal{S}$ can be defined as a collection of tasks. However, the definition of a service imposes a series of additional constraints with respect to that of a business process: (i) services are self-contained units of work (that is, they do not require context or state information of other services) and are connected each other using standard, dependency reducing, decoupled message-based elements

such as XML document exchanges; (ii) each service takes one or more inputs and creates an output perceived as a tangible value for the service requester. More specifically, in [3] a service is recognized as a recurrent communication pattern, where the service requester sends a request and receives a response that is of value for the requester himself/herself. On the other hand, the provider can invoke other services to execute his/her tasks, becoming requester for those services. Therefore, the overall business process can be viewed as a complex structure of nested and chained service invocations connected by control structures. The aim of P2S methodology is to support the process designer in a semi-automatic identification of component services, as well as to give metrics for evaluating the decomposition that has been identified.

2.1 Case Study

As a motivating example, we present a case study based on the cooperative scenario in which a sofa manufacturer produces all the textiles sofa components and purchases backbones from trusted suppliers (Figure 1). Different actors belonging to the sofa manufacturer enterprise have an active role in the analyzed process. In details, administrative activities are performed by the sales and purchasing offices, while production and delivery activities are performed by the manufacturing and shipping departments. Among the participating actors, the final user that sends the order and receives the sofa is implicitly considered.

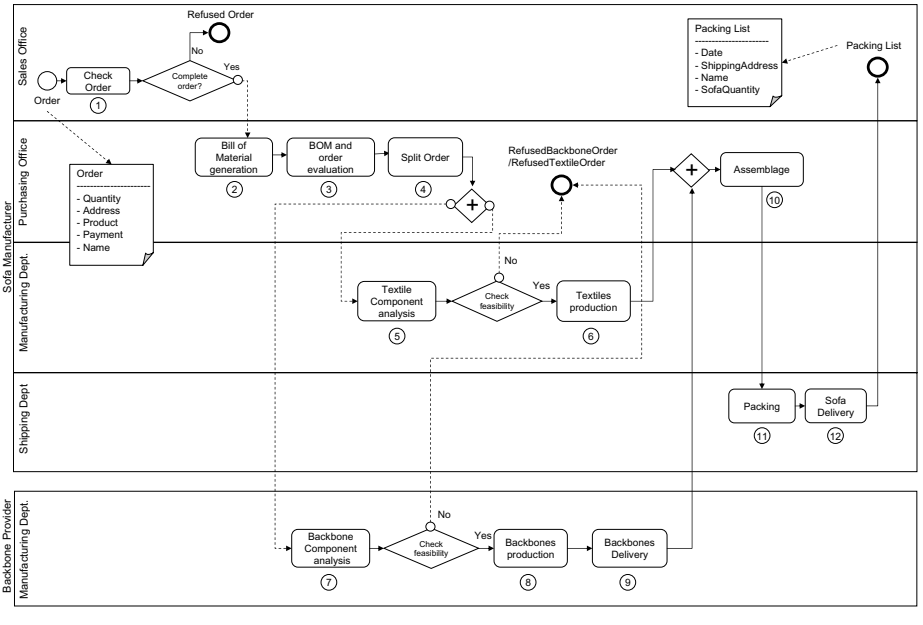

Fig. 1. Business process used as running example

Table 1. Complete task representation (details about I/O have been omitted)

Name	**Input**	**Output**
1. Check order	Order	CompletenessEvaluation
2. BoM Generation	Order	BoM
3. BoM and order Evaluation	Order, BoM	SuitableProviders, Schedule
4. SplitOrder	Order, BoM, SuitableProviders, Schedule	BackboneComponentOrder, TextileComponentOrder
5. Textile Component Analysis	TextileComponentOrder	OrderFeasibility
6. Textile Production	TextileComponentOrder	TextileComponent
7. Backbone Component Analysis	BackboneComponentOrder	OrderFeasibility
8. Backbones Production	BackboneComponentOrder	BackboneComponent
9. Backbones Delivery	BackboneComponent	DeliveredBackboneComponent
10. Assemblage	TextileComponent, DeliveredBackboneComponent	Sofa
11. Packing	Sofa	Pack
12. Delivery	Pack	PackingList
Name	**Operation**	**Organization.Actor**
1. Check order	CheckCompleteness (Order, Constraints)	SofaManufacturer.SalesOffice
2. BoM Generation	BoMDefinition (Order)	SofaManufacturer.PurchasingOffice
3. BoM and order Evaluation	Planning (Order, BoM, ProvidersList, ProductionPlan)	SofaManufacturer.PurchasingOffice
4. SplitOrder	SplitOrder (Order, BoM, SuitableProviders, Schedule)	SofaManufacturer.PurchasingOffice
5. Textile Component Analysis	Evaluate (TextileComponentOrder, ProductionPlan)	SofaManufacturer.ManifacturingDept
6. Textile Production	TextileProduction (RawMaterials)	SofaManufacturer.ManifacturingDept
7. Backbone Component Analysis	Evaluate (BackboneComponentOrder, ProductionPlan)	BackboneProvider.ManifacturingDept
8. Backbones Production	BackboneProduction (RawMaterials)	BackboneProvider.ManifacturingDept
9. Backbones Delivery	Delivery (BackboneComponent)	BackboneProvider.ManifacturingDept
10. Assemblage	Assemblage (TextileComponent, BackboneComponent)	BackboneProvider.ManifacturingDept
11. Packing	Packing (Sofa)	SofaManufacturer.ShippingDept
12. Delivery	Delivery (Pack)	SofaManufacturer.ShippingDept

The event that activates the process is the reception of an order by the sales office. The office checks the order and refuses it if it is affected by incompleteness or inaccuracy problems. If the order is well defined, then the sales office forwards it to the purchasing office, that is responsible for the relationships with raw materials and components providers. The purchasing office generates the Bill of Material (BoM) and evaluates it together with the received order to identify the required components and the providers to contact. Thus, the order is split in sub-orders for each component required. In the example, two sub-orders are created and sent to the internal manufacturing department for the textile production and to an external provider for the backbone production. In both cases, the production units check the received document: if the order is considered as feasible, they start with the production phase and, at the end of the production step, they commit the delivery of the realized components. The assemblage of the components and thus the realization of the final product is in charge of the purchasing office. The process finishes when the shipping department receives the product and delivers it to the final user.

The most of the workflow-based notation tools allow the designer to represent the process in terms of tasks and control structures. Information about the data flow is not provided. In order to identify services, a preliminary step before applying the P2S methodology is the completion of the task representation with the definition of inputs and outputs, the associated operation and responsible actors and organizations (Equation 1). The completed representation of tasks in the running example is summarized in Table 1. Each input/output can be associated to a complex type definition, that includes the I/O name and the list of properties/attributes. For example, Figure 1 also shows the definitions of Order and PackingList information at the beginning and at the end of the process.

According to basic definitions given in Section 2 and task descriptions in Table 1, the task dependency graph and the data exchange graph for the running example are the following:

$\mathcal{TDG} = \langle \mathcal{T}, \mathcal{TD} \rangle$
$\mathcal{T} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$
$\mathcal{TD} = \{\langle 2, 3\rangle, \langle 2, 4\rangle, \langle 3, 4\rangle, \langle 4, 5\rangle, \langle 4, 6\rangle, \langle 4, 7\rangle, \langle 4, 8\rangle, \langle 6, 10\rangle, \langle 8, 10\rangle, \langle 10, 11\rangle, \langle 11, 12\rangle\}$

$\mathcal{DEG} = \langle \mathcal{A}, \mathcal{E} \rangle$
$\mathcal{A}$ = {$\mathcal{A}_0$ = FinalUser, $\mathcal{A}_1$ = SofaManufacturer.SalesOffice, $\mathcal{A}_2$ = SofaManufacturer. PurchasingOffice, $\mathcal{A}_3$ = SofaManufacturer.ManufacturingDept, $\mathcal{A}_4$ = SofaManufacturer.ShippingDept, $\mathcal{A}_5$ = BackboneProvider.ManufacturingDept}
$\mathcal{E}$ = {$\langle \mathcal{A}_0, \mathcal{A}_1$,Order$\rangle$,$\langle \mathcal{A}_1, \mathcal{A}_0$,RefusedOrder$\rangle$, $\langle \mathcal{A}_1, \mathcal{A}_2$,Order$\rangle$, $\langle \mathcal{A}_2, \mathcal{A}_5$,BackboneComponentOrder$\rangle$, $\langle \mathcal{A}_2, \mathcal{A}_3$,TextileComponentOrder$\rangle$, $\langle \mathcal{A}_5, \mathcal{A}_2$,RefusedBackboneOrder$\rangle$, $\langle \mathcal{A}_3, \mathcal{A}_2$, RefusedTextileOrder$\rangle$, $\langle \mathcal{A}_5, \mathcal{A}_2$,DeliveredBackboneComponent$\rangle$, $\langle \mathcal{A}_3, \mathcal{A}_2$,DeliveredTextileComponent$\rangle$, $\langle \mathcal{A}_2, \mathcal{A}_4$,Sofa$\rangle$, $\langle \mathcal{A}_4, \mathcal{A}_0$,PackingList$\rangle$}

3 Methodology

According to the definitions given in Section 2, we consider the business process as the combination of *component services* that execute the set of simple tasks. The proposed methodology (see Figure 2), that guides the business process designer in the identification of component services, is organized into four main phases:

1. *semantic process annotation* - in a distributed heterogeneous environment, where different SMEs provide independently developed process representations, business process elements (inputs and outputs, task names) must be semantically annotated with concepts extracted from shared ontologies; in fact, process designer could use synonymies or terms that are semantically related to ontological concepts and tools and methods to figure out semantic similarities between terms should also be proposed to uniform adopted terminology;
2. *identification of candidate services* - according to an external perspective, process actors involved in the task execution and the values they expect from the business process are identified to determine a first set of candidate

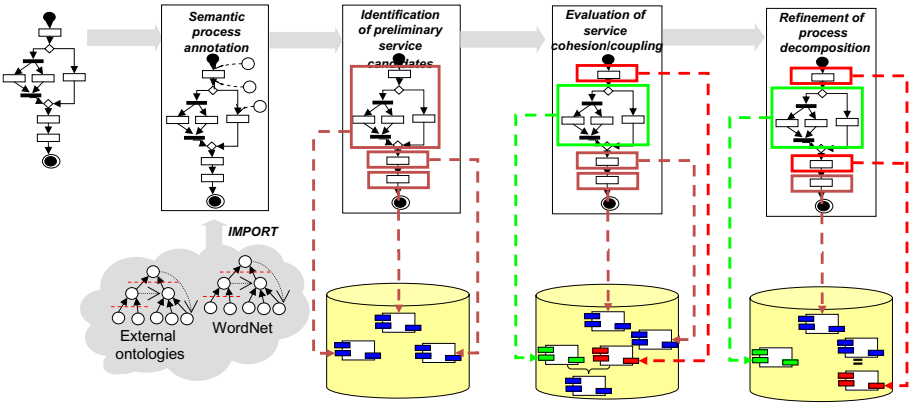

Fig. 2. Methodology phases

component services; in particular, the data exchange graph will be exploited to figure out service invocations and value exchanges between requesters and providers;

3. *evaluation of service cohesion/coupling* - according to an internal perspective, evaluation of service dependencies in terms of cohesion/coupling criteria is useful to better define service structure and granularity; to this aim, the task dependency graph will be exploited;
4. *refinement of process decomposition* - an additional phase is performed to detect multiple invocations of the same service throughout the process workflow by means of proper coefficients; such coefficients will be applied to each pair of identified services to check if they perform the same operations or operate on the same data.

Methodological phases will be detailed in the following sections, with reference to the motivating example.

3.1 Phase 1: Semantic Process Annotation

We suppose that SMEs aiming at collaborating through their business activities agree on a shared reference ontology $\mathcal{O} = \langle \mathcal{C}, \mathcal{R} \rangle$, where $\mathcal{C}$ is the set of ontological concepts (atomic concepts) and $\mathcal{R}$ is the set of semantic relationships (e.g., equivalence, subsumption) between concepts. In the semantic process annotation phase, after the completion of the task representation as shown in Section 2.1, concepts names are associated with business process elements (i.e., task names, input and output names and properties). Since terms used for business process elements do not necessarily coincide with atomic concepts, the reference ontology is extended with pre-existing, domain independent terminological knowledge extracted from an underlying lexical system (e.g., WordNet [4]), that relates each term that is not already included in the reference ontology to atomic concepts by means of synonymy (SYN), broader/narrower term (BT/NT) and related

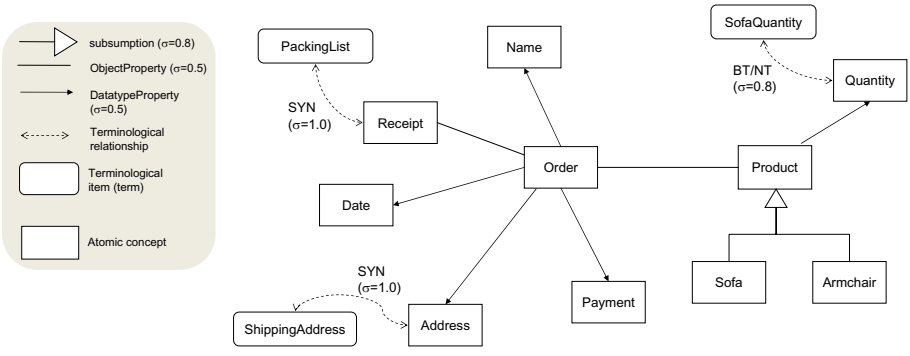

Fig. 3. A portion of shared ontology for the running example

term (RT) relationships. This approach is common and it has been successfully adopted in other contributions [5,6]. A weight $\sigma \in (0, 1]$ is associated with each kind of relationship. For example, $\sigma = 1.0$ for equivalence or synonymy relationships, $\sigma = 0.8$ for subsumption or BT/NT relationships, $\sigma = 0.5$ for the other kinds of relationships. Two generic terms n_1 and n_2 (either atomic concepts or not) can be related by a chain of weighted relationships. We define the *name affinity* value between n_1 and n_2 (denoted with $NAff(n_1, n_2)$) as the product of weights associated to the chain of relationships that relates n_1 to n_2. During the semantic annotation phase, the name affinity evaluation between a term $n_1 \notin \mathcal{C}$ adopted as business process element name and the atomic concepts in $\mathcal{C}$ is used to suggest to the designer the ontological concepts that better match with n_1. Moreover, given two data items $d_1 = \langle n_1, \mathcal{P}_1 \rangle$ and $d_2 = \langle n_2, \mathcal{P}_2 \rangle$ (either input or output of simple tasks), we define the *structural affinity* $SAff(d_1, d_2)$ as the evaluation of their semantic similarity, that is:

$$SAff(d_1, d_2) = \frac{1}{2} \cdot \left[NAff(n_1, n_2) + \frac{2 \cdot \sum_{p_1, p_2} NAff(p_1, p_2)}{|\mathcal{P}_1| + |\mathcal{P}_2|} \right] \tag{2}$$

where $p_1 \in \mathcal{P}_1$, $p_2 \in \mathcal{P}_2$, $|\mathcal{P}_i|$ is the cardinality (that is, the number of elements) of the set $\mathcal{P}_i$ of properties. For the running example, let consider the descriptions of `Order` and `PackingList` data items represented in Figure 1 and the portion of reference ontology and domain-independent terminological knowledge shown in Figure 3. The following affinity values can be evaluated:

$$NAff(\texttt{Order},\texttt{PackingList}) = 1.0 * 0.5 = 0.5$$
$$NAff(\texttt{Quantity},\texttt{SofaQuantity}) = 0.8$$
$$NAff(\texttt{Address},\texttt{ShippingAddress}) = 1.0$$
$$SAff(\texttt{Order},\texttt{PackingList}) = \tfrac{1}{2}\left[0.5 + \tfrac{2*(0.8+1.0+1.0)}{9}\right] = 0.56$$

$Aff_{TOT}(D_1, D_2)$ is the total structural affinity between two sets of data items and is defined as the sum of structural affinity for each pair of items $d_1 \in D_1$ and

$d_2 \in D_2$, normalized with respect to the cardinality of D_1 and D_2. The Aff_{TOT} coefficient will be exploited in the next phases of the methodology. Note that the proposed use of domain independent terminological knowledge can be exploited to bridge the gap between different reference ontologies (as shown in [7]), thus avoiding to constrain collaborating partners to adhere to the same reference ontology.

3.2 Phase 2: Identification of Candidate Services

For the identification of candidate services that are combined in a business process, we propose some heuristics that are derived from empirical observations about the features and the definition of a service found in literature. Services constitute units of work that are logically decoupled. Workflow-based tools and languages usually collect in the same swimlane (either pool or lane in BPMN process representation) logically coupled sets of tasks. Moreover, services are invoked by actors participating to the process to obtain tangible values from other actors. We exploit the data exchange graph defined in Section 2 to identify value exchanges. According to definitions given in [3], a value for one of the process actors can be associated to a service invocation request. For each node A of the data exchange graph (that is, an actor), the outgoing and incoming edges (that is, data transfers towards and from other actors, respectively) are considered as service requests/invocations and responses/values, respectively. The total structural affinity evaluation is performed for each pair $\langle \mathcal{D}req, \mathcal{D}res \rangle$, where $\mathcal{D}req$ is a data item associated with one of the outgoing edges and $\mathcal{D}res$ is a data item associated with one of the incoming edges. Only pairs such that $Aff_{TOT}(\mathcal{D}res, \mathcal{D}req) > 0$ are maintained, that correspond to the service invocation and the corresponding value produced through the service execution. Three cases should be distinguished: (i) an outgoing data value $\mathcal{D}req$ is not associated with any incoming data, that is, $\langle \mathcal{D}req, \emptyset \rangle$; (ii) a match is found between outgoing data $\mathcal{D}req$ and incoming data $\mathcal{D}res$, that is, $\langle \mathcal{D}req, \mathcal{D}res \rangle$; (iii) an incoming data value $\mathcal{D}res$ is not associated with any outgoing data, that is, $\langle \emptyset, \mathcal{D}res \rangle$. The first and the third cases seem correspond to borderline situations, in which an actor sends a request without receiving any value from service invocation or an actor receives a service response/value without sending any request, respectively. This could be due to the impossibility for the supporting tool to find out semantic correspondences between outgoing and incoming data items; in these cases, the process designer must intervene to explicitly set, modify or exclude this kind of matches, according to his/her own domain or process knowledge. The second case corresponds to the invocation of a service, that produces the corresponding value. This leads to the following heuristics. For each process actor that is associated with a pair $\langle \mathcal{D}req, \mathcal{D}res \rangle$, a candidate service $\mathcal{S}$ is recognized whose first task t_i is such that $IN(t_i) \equiv \mathcal{D}req$ and whose last task t_j is such that $OUT(t_j) \equiv \mathcal{D}res$. In particular, the service $\mathcal{S}$ is invoked from the service (if exists) containing the task t_k such that $OUT(t_k) \equiv \mathcal{D}req$.

The list of candidate services is proposed to the process designer, that can validate or refuse them. Let consider the sofa manufacturer example. Applying

the heuristics described above, the following pairs $\langle \mathcal{D}req, \mathcal{D}res \rangle$ are identified, with corresponding candidate services:

$\langle$Order,RefusedOrder$\rangle \Rightarrow \mathcal{S}_1 = \{t_1\}$ (requester: final user)

$\langle$Order,PackingList$\rangle \Rightarrow \mathcal{S}_2 = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}\}$ (requester: final user)

$\langle$BackboneComponentOrder,DeliveredBackboneComponent$\rangle \Rightarrow \mathcal{S}_3 = \{t_7, t_8, t_9\}$
(requester: purchasing office)

$\langle$BackboneComponentOrder,RefusedBackboneOrder$\rangle \Rightarrow \mathcal{S}_4 = \{t_7\}$ (requester: purchasing office)

$\langle$TextileComponentOrder,DeliveredTextileComponent$\rangle \Rightarrow \mathcal{S}_5 = \{t_5, t_6\}$
(requester: purchasing office)

$\langle$TextileComponentOrder,RefusedTextileOrder$\rangle \Rightarrow \mathcal{S}_6 = \{t_5\}$ (requester: purchasing office)

$\langle$Sofa,PackingList$\rangle \Rightarrow \mathcal{S}_7 = \{t_{11}, t_{12}\}$ (requester: purchasing office)

3.3 Phase 3: Evaluation of Service Cohesion/Coupling

Once the candidate services have been identified as set of tasks, services are further analyzed in terms of cohesion/coupling criteria in order to better define service structure and granularity. The adopted cohesion/coupling metrics have been inspired by their well-known application in software engineering field [8]. In our scenario, task dependencies inside and across services are identified in terms of common used data and are exploited to better aggregate tasks or to figure out parallelism among them. In this phase, the task dependency graph is exploited.

Given two tasks t_i and t_j, we define the task coupling coefficient as follows:

$$\tau(t_i, t_j) = \begin{cases} Aff_{TOT}(OUT(t_j), IN(t_i))) & if\ t_j \mapsto t_i \\ Aff_{TOT}(IN(t_j), OUT(t_i))) & if\ t_i \mapsto t_j \\ \frac{Aff_{TOT}(IN(t_i), IN(t_j))) + Aff_{TOT}(OUT(t_i), OUT(t_j)))}{2} & otherwise \end{cases} \quad (3)$$

The third member in the equation 3 is used for tasks t_i and t_j that are executed on parallel branches, but work on semantically related data. In details, two tasks t_i and t_j are considered: (i) *loosely coupled*, if $0 < \tau(t_i, t_j) < \delta$, where δ is a threshold set by the designer ($\delta \in [0, 1]$); (ii) *strongly coupled*, if $\tau(t_i, t_j) \geq \delta$; (iii) *decoupled*, if $\tau(t_i, t_j) = 0$. Given a service $\mathcal{S}$ identified in phase 2, the service cohesion coefficient is defined as:

$$coh(\mathcal{S}) = \begin{cases} 2 \cdot \frac{\sum_{i,j} \tau(t_i, t_j)}{|\mathcal{S}| \cdot (|\mathcal{S}| - 1)} \ \forall\ t_i, t_j \in \mathcal{S} & |\mathcal{S}| < 1 \\ 1 & |\mathcal{S}| = 1 \end{cases} \quad (4)$$

where $|\mathcal{S}|$ is the number of tasks in $\mathcal{S}$. Given two services $\mathcal{S}_1$ and $\mathcal{S}_2$, the coupling coefficient between them is defined as:

$$coup(\mathcal{S}_1, \mathcal{S}_2) = \frac{\sum_{i,j} \tau(t_i, t_j)}{|\mathcal{S}_1| \cdot |\mathcal{S}_2|} \ \forall\ t_i \in \mathcal{S}_1\ \wedge\ t_j \in \mathcal{S}_2 \quad (5)$$

Service cohesion and coupling coefficients are used to evaluate the average cohesion and coupling of the overall process $\mathcal{B}P$, respectively:

$$pcoh(\mathcal{B}P) = \frac{\sum coh(\mathcal{S}_i)}{|\mathcal{B}P|} \quad (6)$$

$$pcoup(\mathcal{BP}) = \frac{\sum_{i,j} coup(\mathcal{S}_1, \mathcal{S}_2)}{|\mathcal{BP}| \cdot (|\mathcal{BP}| - 1)} \tag{7}$$

combined in the coupling/cohesion ratio Γ:

$$\Gamma = \frac{pcoup(\mathcal{BP})}{pcoh(\mathcal{BP})} \tag{8}$$

where $|\mathcal{BP}|$ is the number of services identified in second phase. Task coupling coefficient is used to establish the goodness of the service cohesion and coupling of the set of candidate services identified in the second phase of the methodology. Tasks are aggregated according to their level of coupling by applying a hierarchical clustering algorithm. Firstly, each task constitutes a cluster with only one element (singleton). Two clusters C_1 and C_2 are merged together if there exist two tasks $t_1 \in C_1$ and $t_2 \in C_2$ that are coupled (loosely or strongly) or viceversa. Clusters are iteratively merged until a unique cluster is obtained for the overall process or there is no coupling between the tasks of clusters that have been identified. Clusters that present tasks with highest value for τ are merged first. At each merging step the coupling/cohesion ratio Γ is evaluated. The best clustering level is the one that minimized the Γ value. After clustering has been applied and optimized, the system proposes to the designer to further split services identified in the previous phase if they contain tasks belonging to different clusters to allow their parallel execution. On the other hand, the process designer may be suggested to merge services if they contain coupled tasks. The designer can evaluate the results and then decide to adopt different business process re-engineering actions.

If we apply the clustering algorithm to the running example, we obtain the tree shown in Figure 4, where, for example, the task t_{10} is decoupled from the other ones, while from phase 2 it is included with the other simple tasks as part of the $\mathcal{S}_2$ candidate service. The process designer may decide to split this candidate

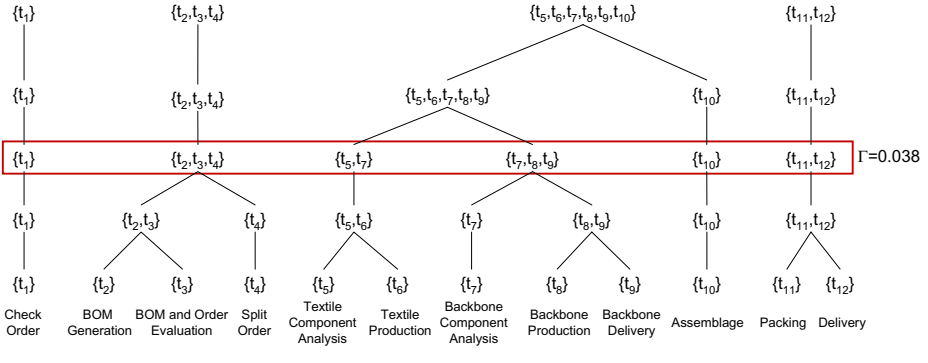

Fig. 4. Clustering of coupled simple tasks

service, thus obtaining three more services $\mathcal{S}_{21} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$, $\mathcal{S}_{22} = \{t_{10}\}$ and $\mathcal{S}_{23} = \{t_{11}, t_{12}\}$.

3.4 Phase 4: Refinement of Process Decomposition

Services identified in the previous phases are sub-processes constituted by one or more tasks that identify service operations. However, there could be similar services that are invoked multiple times throughout the process. For example, services that check the textile component and backbone component feasibility could be recognized as similar services and, in particular, they could be viewed as different invocations of the same service. To detect possible overlapping services in the business process, coefficients already introduced in [9] are applied. In particular, the *Entity-based similarity coefficient* between two services $\mathcal{S}_1$ and $\mathcal{S}_2$, denoted with $ESim(\mathcal{S}_1, \mathcal{S}_2)$, is used to state if $\mathcal{S}_1$ and $\mathcal{S}_2$ work on the same data. Denoting with $\mathcal{S}^{IN}$ and $\mathcal{S}^{OUT}$ the union sets of inputs and outputs of the tasks in a service $\mathcal{S}$, the $ESim$ coefficient is computed as:

$$ESim(\mathcal{S}_1, \mathcal{S}_2) = Aff_{TOT}(\mathcal{S}_1{}^{IN}, \mathcal{S}_2{}^{IN}) + Aff_{TOT}(\mathcal{S}_1{}^{OUT}, \mathcal{S}_2{}^{OUT}) \in [0, 2] \quad (9)$$

The *Functionality-based similarity coefficient* between two services $\mathcal{S}_1$ and $\mathcal{S}_2$, denoted with $FSim(\mathcal{S}_1, \mathcal{S}_2)$, is used to state if $\mathcal{S}_1$ and $\mathcal{S}_2$ perform the same operations. $FSim$ is based on the *Operation Similarity coefficient* between two tasks t_1 of $\mathcal{S}_1$ and t_2 of $\mathcal{S}_2$, denoted with $OpSim(t_1, t_2)$ and computed as:

$$\begin{aligned} OpSim(t_1, t_2) = NAff(n_{t_1}, n_{t_2}) + Aff_{TOT}(IN(t_1), IN(t_2)) + \\ Aff_{TOT}(OUT(t_1), OUT(t_2)) \end{aligned} \quad (10)$$

where $OpSim \in [0, 3]$, since it is the sum of three elements in the range [0,1]. The $FSim$ coefficient is evaluated as:

$$FSim(\mathcal{S}_1, \mathcal{S}_2) = \frac{2 * \sum_{h,k} OpSim(t_h, t_k)}{|\mathcal{S}_1| + |\mathcal{S}_2|} \in [0, 3] \quad (11)$$

Entity-based and functionality-based service similarity coefficients support the designer to recognize multiple invocations of the same service throughout the process workflow.

Considering the running example, it is possible to find some similarities between the services identified in the previous phase. In details, two pairs of services are selected: $\langle \mathcal{S}_3 = \{t_7, t_8, t_9\}, \mathcal{S}_5 = \{t_5, t_6\} \rangle$, $\langle \mathcal{S}_4 = \{t_7\}, \mathcal{S}_6 = \{t_5\} \rangle$. The designer will discard the pair $\langle \mathcal{S}_3, \mathcal{S}_5 \rangle$ since they are characterized by input, output and terms similarities but the task operation has a different implementation in the two services and it is not possible to merge them in a unique component. The pair $\langle \mathcal{S}_4, \mathcal{S}_6 \rangle$ can be instead considered for a service reconciliation, since both services have as input the same document and on the basis of the production plan use the same algorithm in order to evaluate the order feasibility.

4 Related Work

Links between the world of business processes and Web services have been thoroughly analyzed in different contexts and from different perspectives in the literature. Many contributions focus on the analysis of existing Web services and

address the issue of their composition to obtain the desired process. Nowadays, organizations more and more implement collaborative businesses through component services over the Internet.

In the literature, in order to bridge the link between services and business processes some contributions consider existing Web services and define enriched process representation. In particular, they start from a business process that is manually represented as a composition of services semantically enriched with ontologies. In [10] an abstract process represents a Web process whose control and data flow are defined at design time, but the actual services are not chosen till at run-time. Run-time service selection can be automated with the semantic representation of the knowledge of the domain experts in ontologies and rules (semantic Web processes). In [11] 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 (called component 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 such as different QoS parameters, that are exploited to select the right service at run-time. These approaches are especially useful in environments where component services are relatively fixed, while the methodology presented in this paper does not require any knowledge about existing services.

Moreover, there are approaches that attempt to assist service providers and service aggregators in multi-party business processes [12,13,14]. In particular, [13] discusses how business process should be described so that services can be properly identified and provides strategies and principles regarding functional and non-functional aspects of Web service design. Furthermore, in [14] authors define a methodology that aims at defining a foundation of development principles for Web services based on which business processes can be assembled into business scenarios. In [12] a goal-based approach for the identification of service composition is proposed. All these approaches provide guidelines for the service identification without giving an operational support. The use of coupling and cohesion metrics to evaluate process decomposition into sub-processes or activities has been suggested in [8,15,16], but these approaches mainly propose techniques to compare different decompositions to choose the best one. This issue also relates to existing contributions in literature to support the transformation of legacy applications into component services [17,18,19]. For example, [17] describes the Service-Oriented Migration and Reuse Technique (SMART) as a technique that helps organizations to analyze legacy systems to determine whether their functionalities, or a subsets of them, can be reasonably exposed as services in a Service-Oriented Architecture. Other valuable guidelines in this context are provided in [18,19]. Moreover, a considerable number of research efforts have been devoted to the composition of services both in academia and industry [20,21,22]. However, all these valuable contributions do not offer operational support for the identification of process tasks that can be aggregated and exposed as services. For this purpose, the methodology described in this paper

tries to uniform component services identification at the same granularity, in order to improve and speed up the following discovery phase. To the best of our knowledge, there are no approaches in literature that propose a semi-automatic methodology that supports the designer in identifying the component services in a business process.

5 Conclusion

The methodology presented in this paper aims at constituting a semi-automatic approach for the identification of the subset of functionalities that can be exported as services to implement a collaborative business process. The methodology starts from a process represented by means of a workflow-based language (e.g., BPMN) and supports the designer for the identification of component services. Future work will focus on the design of a CASE tool able to support the process designer through the phases of the methodology. Currently, single modules that implement the methodological phases have been developed and it is necessary to integrate them and properly test the resulting application. Additional features of the developed system will be investigated to deal with a multi-knowledge environment, where the system relies on distinct reference ontologies and different process representation languages are used and must be integrated. Finally, the proposed methodology will be further studied in the field of service orchestration and composition to implement efficient solutions.

Acknowledgements

This work has been partially supported by the TEKNE (Towards Evolving Knowledge-based internetworked Enterprise) FIRB Project (http://www.tekne-project.it/), founded by the Italian Ministry of Education, University and Research.

References

1. O'Brien, J.A.: Introduction to Information Systems: Essentials for the Internetworked Enterprise. McGraw-Hill Education, New York (2000)
2. Chang, S.H., Kim, S.D.: A service-oriented analysis and design approach to developing adaptable services. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749. Springer, Heidelberg (2007)
3. Dietz, J.L.G.: The atoms, molecules and fibers of organizations. Data & Knowledge Engineering (2003)
4. Fellbaum, C.: Wordnet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
5. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet:Similarity - Measuring the Relatedness of Concepts. In: Proc. of Nineteenth Conf. of Artificial Intelligence (AAAI 2004), Intelligent Systems Demonstration, San Jose, CA, pp. 1024–1025 (2004)

6. Corley, C., Mihalcea, R.: Measuring the Semantic Similarity of Texts. In: Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor, Michigan, pp. 13–18 (2005)
7. Bianchini, D., De Antonellis, V., Melchiori, M.: Flexible Semantic-based Service Matchmaking and Discovery. World Wide Web Journal 11(2), 227–251 (2008)
8. Vanderfeesten, I., Reijers, H., van der Aalst, W.: Evaluating workflow process designs using cohesion and coupling metrics. Computer in Industry 59(5), 420–437 (2008)
9. Bianchini, D., De Antonellis, V., Pernici, B., Plebani, P.: Ontology-based methodology for e-service discovery. Journal of Information Systems, Special Issue on Semantic Web and Web Services 31(4-5), 361–380 (2006)
10. Mulye, R., Miller, J., Verma, K., Gomadam, K., Sheth, A.: A semantic template based designer for Web processes. In: Proc. of 2005 IEEE Int. Conf. on Web Services (ICWS 2005), Orlando, Florida, USA, pp. 461–469 (2005)
11. Sheng, Q., Benatallah, B., Maamar, Z., Dumas, M., Ngu, A.: Enabling Personalized Composition and Adaptive Provisioning of Web Services. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 322–337. Springer, Heidelberg (2004)
12. Kaabi, R.S., Souveyet, C., Rolland, C.: Eliciting service composition in a goal driven manner. In: Proc. of the 2nd Int. Conf. on Service Oriented Computing, New York, NY, USA, pp. 308–315 (2004)
13. Papazoglou, M.P., Yang, J.: Design Methodology for Web Services and Business Processes. In: Buchmann, A., Casati, F., Fiege, L., Hsu, M.-C., Shan, M.-C. (eds.) TES 2002. LNCS, vol. 2444, p. 54. Springer, Heidelberg (2002)
14. Papazoglou, M.P., van den Heuvel, W.J.: Business process development life cycle methodology. Communications of ACM 50(10), 79–85 (2007)
15. Castano, S., De Antonellis, V., Melchiori, M.: A Methodology and Tool Environment for Process Analysis and Reengineering. Data and Knowledge Engineering 31(3), 253–278 (1999)
16. Baresi, L., Casati, F., Castano, S., Fugini, M., Mirbel, I., Pernici, B.: WIDE Workflow Development Methodology. In: Proc. of Int. Joint Conf. on Work Activities Coordination and Collaboration, pp. 19–28 (1999)
17. Lewis, G., Morris, E., O'Brien, L., Smith, D., Wrage, L.: SMART: The Service-Oriented Migration and Reuse Technique. Techncal Note CMU/SEI-2005-TN-029, Carnegie Mellon University, Software Engineering Institute (2005)
18. Lawrence, C.: Adapting legacy systems for SOA. Technical report, IBM (2007)
19. Microsoft: The Business Value of Legacy Modernization. Technical report, Microsoft (2007)
20. Baresi, L., Bianchini, D., De Antonellis, V., Fugini, M., Pernici, B., Plebani, P.: Context-aware Composition of *e*-services. In: Proc. of Fourth VLDB Workshop on Technologies for E-Services (TES 2003), Humboldt-University zu Berlin, Germany, pp. 49–58 (2003)
21. Benatallah, B., Sheng, Q.Z., Dumas, M.: The Self-Serv environment for Web services composition. IEEE Internet Computing 7(1), 40–48 (2003)
22. Rao, J., Su, X.: A Survey of Automated Web Service Composition Methods. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 43–54. Springer, Heidelberg (2005)