Chapter 2
A Combination Framework for Exploiting the Symbiotic Aspects of Process and Operational Data in Business Process Optimization

Sylvia Radeschütz, Holger Schwarz, Marko Vrhovnik, and Bernhard Mitschang

Abstract A profound analysis of all relevant business data in a company is necessary for optimizing business processes effectively. Current analyses typically run either on business process execution data or on operational business data. Correlations among the separate data sets have to be found manually under big effort. However, to achieve a more informative analysis and to fully optimize a company's business, an efficient consolidation of all major data sources is indispensable. Recent matching algorithms are insufficient for this task since they are restricted either to schema or to process matching. We present a new matching framework to (semi-)automatically combine process data models and operational data models for performing such a profound business analysis. We describe the algorithms and basic matching rules underlying this approach as well as an experimental study that shows the achieved high recall and precision.

Introduction

"Information integration is a vibrant field powered not only by engineering innovation but also by evolution of the problem itself" [2]. The increasing number of web services available within an organization raises a new integration task: The warehousing and analysis of processes for fast adaption and optimization of processes [25]. However, these methods usually fall short or require significant manual labor [5] when it comes to integrating process data with related operational data from other business applications such as ERP systems. For example, when trying to optimize the process of a car rental company illustrated in Fig. 2.1, a highly relevant question to a business analyst is: How do trainings and work experience

S. Radeschütz (✉) · H. Schwarz · M. Vrhovnik · B. Mitschang
Universität Stuttgart (IPVS), Stuttgart, Germany
e-mail: sylvia.radeschutz@ipvs.uni-stuttgart.de; holger.schwarz@ipvs.uni-stuttgart.de; marko.vrhovnik@ipvs.uni-stuttgart.de; bernhard.mitschang@ipvs.uni-stuttgart.de

T. Özyer et al. (eds.), *Information Reuse and Integration in Academia and Industry*,
DOI 10.1007/978-3-7091-1538-1_2,

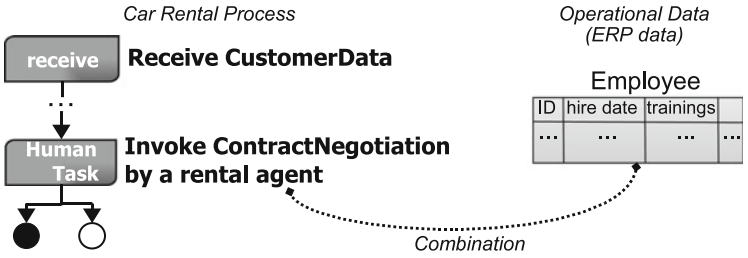

Fig. 2.1 Combination of process and operational data

affect the execution time as well as the outcome of the rental process? Answering this question requires both process data (execution data, paths taken) as well as operational data related to the employee executing the *ContractNegotiation* activity in the process (hire date, trainings). In such a situation, an effective integration of ERP data with process data makes a valuable contribution by ensuring that all relevant data is taken into account.

Our approach provides a solution to this matching problem combining process data models with operational data models of a company. It is novel, because it goes beyond mere schema or process matching. Instead, operational data models are matched with process data models. Content and structure of process models and audit trails that record process data significantly vary from operational data models, which adds complexity to the matching task. Thus, typical schema matching rules will fail. In order to distinguish our approach from these classical matching rules, we call our approach "combination". In particular, the main contributions of this paper are:

- We introduce a combination framework for processes and operational data and explain its processing pipeline covering pairing and filtering steps.
- We elaborate on the main steps of the processing pipeline and the used set of combination rules. This includes rules that are specific to the combination steps as they consider process structures (e.g. process variables as well as control and data flow structures underlying a process) and schema structures.
- Our rules benefit from the technologies developed for semantic web services and reuse their semantic annotations for variables. The fact that a standard for annotating messages and services is available [24] emphasizes the pragmatics of our approach.
- We discuss the benefits of the combination framework and describe a detailed experimental evaluation. The evaluation shows that our technology provides both high recall and high precision.

This paper extends the work presented in [16] regarding the following aspects: More details related to the combination framework and the main steps of the processing pipeline are presented. Concerning the pairing and the filtering phase, the focus of this paper is on the algorithms that allow to efficiently apply the

combination rules. Furthermore, additional experimental results as well as a more extensive discussion of related work are covered here.

The paper is organized as follows: In section "Related Work", we discuss related work. We introduce our combination framework in section "Combination Framework" and illustrate main aspects by means of an extensive sample scenario. Sections "Pairing" and "Filtering" introduce the rules and algorithms for the pairing and filter phase. In section "Evaluation", we discuss the benefits of the combination framework and present experimental results, before section "Conclusion" concludes.

Related Work

The optimization of business processes plays a major role in many companies. Business performance management [3,6,19,20] or process mining [1,18,21,22,27] allow to analyze process data and to discover new workflow models out of the audit logs. Related operational data sources are typically neglected. One of our previous research work was concerned with the optimization of processes including SQL statements in BPEL/SQL activities that were executed on operational data [23]. We developed techniques to analyze and thus to understand control and data flow. This provides one basis for reusing it here for combining process variables to related operational data.

Finding combinations between process variables and operational data models is closely related to many other matching problems. The highest similarity can be found compared to schema matching or web service matching. The database community considers the problem of automatically matching schemas [8–10, 17]. The work in this area has developed several methods that try to capture clues about the semantics of the schemas and suggest matches based on them. Such methods include linguistic analysis, structural analysis, the use of domain knowledge and reuse techniques. However, the search for matching operational data with process variables differs from schema matching in three significant ways. First, content and structure of the input sources are different: our combination copes with variables that are nested within other non-matchable process elements while in pure schema matching all elements might be combinable. Furthermore, the audit trails vary significantly from operational data storage. Secondly, not every variable is combinable. It makes no sense to match technical parts of a variable with operational data. Pure name matching produces misleading results. Thirdly, the process variables are typically much more loosely related to each other than tables in a schema, and each web service in isolation has less information than a schema. Hence, we lose tremendous semantic preciseness if we only rely on techniques for schema matching in this context.

Recent work for matching web services [4,12] proposes annotating web services manually with additional semantic information, and then using these annotations to compose new services automatically. But there, the goal is to ease the modeling

and monitoring of business processes by support of domain ontologies. In [7, 11], non-annotated web services are combined. However, all approaches refer to web services only. A combination of business process artifacts with operational data, as we describe it here, has not been suggested so far. Furthermore, they only focus on web service interfaces and not on the process itself, as we do. We consider process semantics, since we go deeper into the process structure by looking into control and data flow issues to find relationships.

Approaches in [5, 26] combine process and operational data models by hand. However, this is very cumbersome and error-prone for such huge data amounts. We think it is worthwhile to face the challenge of automatically combining this data and developed a technology for exactly this.

Combination Framework

In this section, we briefly describe the pipeline approach of our combination framework and present its input models, i.e., the process model, the operational data model as well as the annotation model. We also define the combination results of each step and the final combination result.

Pipeline and Processing Overview

The processing pipeline shown in Fig. 2.2 creates a set of combinations (*COM*) between process variables (*PV*) and operational schema elements (*S*). Each step requires a different fraction of input sets: partially-annotated process variables (*PV*) and schema elements (*S*) or their annotated components (PV_{Ont}, S_{Ont}), the annotation ontology (*Ont*), and the annotated (P_{Ont}) or partially-annotated process context (*P*) of the variables.

In order to reflect the different input data, we distinguish two basic processing pipelines: (i) partially-annotated Pairing → structure-based Filtering → manual Filtering and (ii) annotated Pairing → structure-based Filtering → manual Filtering. *Partially-annotated pairing* covers the pairing of partially annotated process context, process variables and schema elements. If all elements involved in a pairing step are annotated, we consider it as *annotated pairing*.

After being converted to the internal representation format of the framework, the process models and schema models are traversed in the pairing phase to determine at least one matching schema element for each process variable with a similarity value between 0 and 1. If different rules estimate a combination, the highest value is taken. One pairing step performs reasoning over *Ont* and annotated elements. For this *annotated pairing* a reasoner determines an intermediate combination result consisting of semantically annotated variables and corresponding semantically annotated schema elements with a similarity value. It applies combination

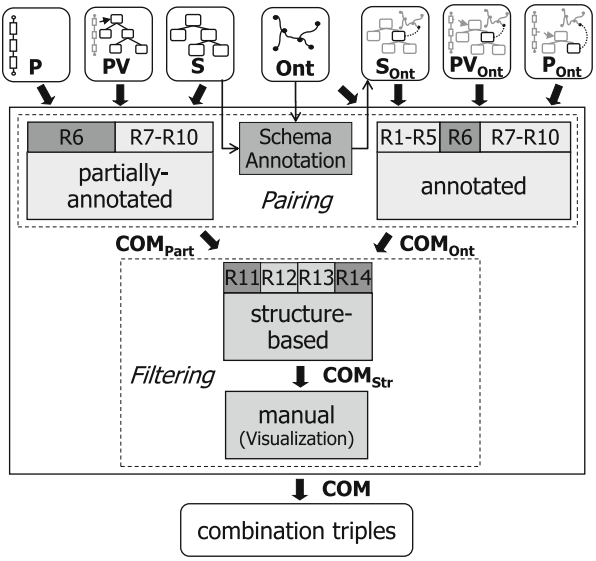

Fig. 2.2 The combination framework covers rules for annotated and partially-annotated pairing and structure-based filtering. It provides combination triples that consist of combinations of process variables and schema elements as well as a similarity value between 0 and 1

rules R1–R5. The *partially-annotated pairing* step finds combination results for combinations of annotated and non-annotated elements. In R6, we exploit well-known matching techniques using names of schema elements or process variables. R7–R10 are essential to consider the ontology, the data flow or other process features for combination. They are applied in both pairing steps. The framework also supports an easy adaption of exiting rules and the extension with new rules.

In the *structure-based filtering* step the results from the pairing steps are refined by considering element hierarchies in PV and S or process features contained in P. It uses well-known structural matching techniques (shaded in dark-gray) with features like the elements' path or data types (R11, R14), but also needs novel rules R12 and R13. The *manual filtering* step derives the final result by user interaction. Process variables are visualized together with their proposed combinations for changing them manually. For a pure automatic combination this step is skipped or just serves as a visualization output.

Schema Annotation is executed when the process side is semantically annotated but the operational data elements are not. This is quite often the case as the trend goes towards semantic web services but database annotations are still not widely-used. As our framework applies common schema matching algorithms to find matches between ontology *Ont* and partially-annotated schema elements S, we do not go into detail here. After matching, the results are stored as semantic annotations into the operational metadata S_{Ont}. Then *annotated pairing* is executed using these annotations together with the given variable annotations PV_{Ont}.

Input and Output Models

BPEL process models consist of various components, e.g. *process variables PV* that are in the focus of our combination framework. They consist of XML schema types, or their types are defined in a WSDL (Web Service Description Language) file. All process variables and also their operations and interfaces in the WSDL description of the process may be annotated by a given ontology *Ont*.

Other parts in the process also give valuable information for the combination. This includes names of activities and both data flow and control flow aspects modeled by these activities. The correlation set also supplies context information with its name and structure. It is used for routing subsequent messages between two web services to the correct process instance. How this process knowledge is exploited in the framework is shown in the next sections.

We consider an *operational data model S* as a collection of relational tables and views or a collection of XML elements and their attribute elements. Operational data models can be available with their semantic annotations as described in [14]. The combination result *COM* is a set of combination triples of a process variable element, an operational data element and a similarity value between 0 (dissimilar) and 1 (very similar) indicating the plausibility of their correspondence.

Definition 1. Let $COM = PV \times S \times sim$ be a set of *combination triples*. One triple indicates that the process variable element $v_r \in PV$ corresponds to the operational schema element $s_j \in S$ with a similarity value *sim* from an interval [0,1] of rational numbers.

Combination triples cover directed combinations because the goal is to find all match candidates for the variables of a process while accepting that schema elements may remain unmatched. This goal is different to schema matching where the elements of both sources aim to find a match partner. Without having to combine all the schema elements, the combination problem is simplified, as the amount of process variables is usually smaller than the set of schema elements. A combination is an n:m-relation, because a variable may map to many schema elements and a schema element may map to many variables. According to the combination pipeline from Fig. 2.2, our framework applies various combination steps to determine the following subsets of *COM*:

Definition 2. Let $COM_{Ont} = PV_{Ont} \times S_{Ont} \times sim_{Ont}$ be a relation whose domain is the Cartesian product of PV_{Ont}, S_{Ont} and an interval that includes rational numbers between 0 and 1. Then, the *combination set* $COM_{Ont} \subseteq COM$ contains combination triples of annotated process variable nodes in $PV_{Ont} \subseteq PV$ and annotated schema elements in $S_{Ont} \subseteq S$ and their similarity value sim_{Ont}. COM_{Ont} is created by making use of an ontology *Ont*.

Definition 3. Let $COM_{Part} = PV \times S \times sim_{Part}$ be a combination set $Com_{Part} \subseteq COM$. Then COM_{Part} contains combination triples between process variable nodes in *PV* and schema elements in *S* and their similarity value sim_{Part}.

Definition 4. Let $COM_{Str} = PV_{Str} \times S_{Str} \times sim_{Str}$ be a relation that refines the combination sets COM_{Ont} and COM_{Part} that contain $sim_{Ont} > 0$ or $sim_{Part} > 0$ by estimating the structural similarity sim_{Str} of the nodes in Com_{Ont} and Com_{Part} respectively. Then, the *combination set* $COM_{Str} = COM_{Ont} \cup COM_{Part} \subseteq COM$ contains combination triples of process variable nodes in PV_{Str} and schema elements in S_{Str} and their similarity value sim_{Str}.

COM_{Ont} is the combination result of the annotated pairing step, whereas partially-annotated pairing leads to the result COM_{Part}. In the partially-annotated pairing step the pairing rules do not require annotations of the variables and schema elements, so it can be applied for S and PV. Structure-based filtering can be applied to both result sets COM_{Ont} and COM_{Part} and leads to COM_{Str}.

Definition 5. Let the value sim_i be a weighted mean of the similarity of sim_{i_P} received from the pairing step and $sim_{i_{Str}}$ received from the filtering step: $sim_i = w_{weight} * sim_{i_P} + (1 - w_{weight}) * sim_{i_{Str}}$, where the constant w_{weight} is in the range of 0 to 1 and sim_{i_P} is the maximum value of sim_{Ont} or sim_{part}.

A final combination triple com_i is created by calculating a weighted similarity sim_i based on the similarity values from the pairing step and the filtering step. Definition 5 defines this in analogy to [13]. We accept triples for a combination of v_r and s_j where the calculated similarity value exceeds a certain threshold ($sim_i >$ threshold).

Sample Scenario

This section describes the input and output sets of the combination framework for the sample scenario shown in Fig. 2.3. This fragment of a car rental process describes the selection of a rental car. It is supposed to be optimized in a sense that expensive long running process parts must be analyzed and revised. All process variables are marked by # in Fig. 2.3. The process receives its input data by activity *CustomerData* with information about a customer and his preferred car model and checks in activity *RentalService* if it is available. If no car is available during the desired rental period, an employee executes the human task *ContractNegotiation* to prove if the customer would also accept another car class. The task is assigned to one of the available roles. Thus, *ContractNegotiation* can be claimed and executed by all agents from departments A, B or C. If the customer does not accept an alternative car the process is canceled. Otherwise, the car is handed over to the customer by an employee of department D in human task *CarHandOver*. The operational data in our scenario (shown in tables *Customer*, *Automobile* and *Employee*) includes useful data for optimization as well. Thus, these data models are combined with the elements of the process data models: Element *custID* of variable *inputData* with *CID* of *Customer* table (1), table *Employee* with the executing roles of *ContractNegotiation* in *TaskVar* (2), and variable *ServiceInfo* with *Model* of table *Automobile* (3).

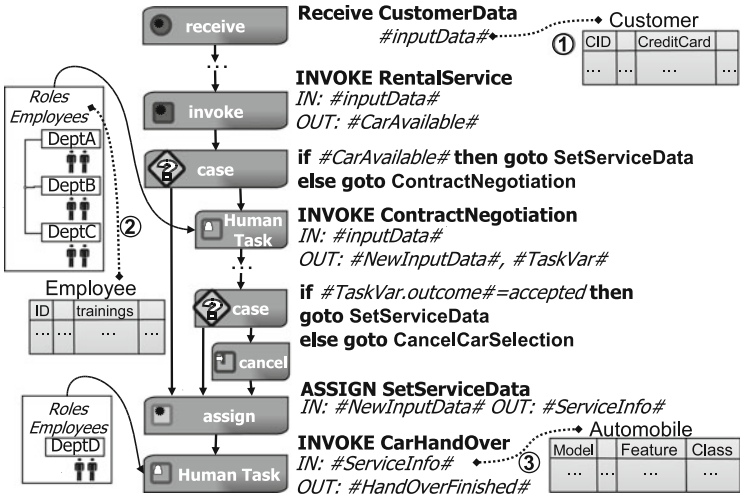

Fig. 2.3 RentalCarSelection scenario

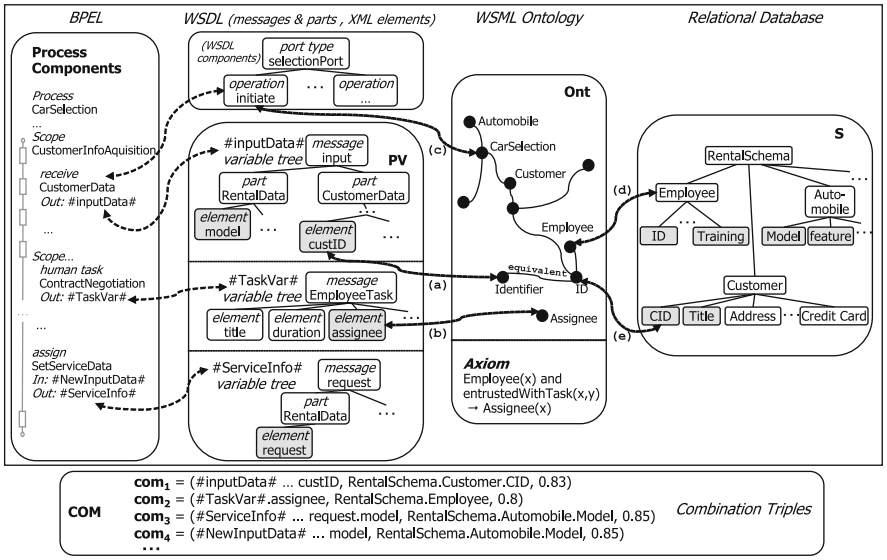

Fig. 2.4 Input and output data models of sample scenario

All input and output sets of our combination pipeline are illustrated in Fig. 2.4. In the left part of this figure, the process and its variables *PV* are shown. The variables *inputData*, *TaskVar* and *ServiceInfo* are illustrated referring to activities in the *CarSelection* process where they are used. The relevant components of a variable for the combination procedure are marked in gray. Tree components are defined in different sources: BPEL, WSDL and variable definitions in XML. On the right side,

a relational operational schema S is illustrated via a tree-structure with the columns as leaves. Some components on both sides are annotated by *Ont*. The concepts in *Ont* are represented as a graph and are connected via equivalent, subconcept or object property relations. *Ont* further contains axioms like the definition of an assignee as an employee that is entrusted with a task.

The bottom of Fig. 2.4 illustrates four output triples received by our combination rules. Each triple com_i shows a process variable element and an operational schema element and their similarity value calculated by our rules. To calculate sim_i for all com_i, we choose a weight = 0.6 (Definition 5) and a threshold = 0.7.

Pairing

This section describes annotated and partially-annotated pairing. Depending on whether there is an annotation on none, one or both process and schema input models, different rules are applicable. Some of these rules are taken from other matching approaches, some of them are applied to our case, and some are new. Table 2.1 shows all pairing rules and the similarity value they calculate. Rules and algorithms for annotated pairing (R1–R5) are further explained in the following two sections, whereas non-annotated and partially-annotated pairing (R6–R10) is the subject of sections "Partially-Annotated Pairing Rules" and "Partially-Annotated Pairing Algorithm".

Annotated Pairing Rules

All combination rules for annotated pairing are based on reasoning over a given annotation ontology *Ont* and calculate a similarity value between annotated elements. They adapt existing approaches [2, 13] that find matches between names based on semantic relations defined in a lexicon. Our rules use an ontology instead and the respective elements are explicitly annotated. Thus, these relations carry even more weight. Rule R1 (*sameConcept*), rule R2 (*equivalentConcept*), rule R3 (*subConcept*) or rule R5 (*union*) combine those elements in PV_{Ont} and S_{Ont} that are annotated by the same ontology concept or that have a synonym, sub Conept or union relation modeled between their ontological concepts. In the sample scenario, rule R2 combines *custID* with column *CID*, as they are annotated by equivalent concepts in *Ont* (see com_1 and arrows (a) + (e) in Fig. 2.4). Rule R4 (*axiom-based*) makes use of axioms that are defined in the ontology using logical expressions. At least one antecedent of the axiom may consist of a domain concepts c_r while a consequent consists of c_j. If c_r and c_j are used for annotation like c_r for v_r and c_j for s_j and if both atoms refer to the same parameter x, the elements are combined. The similarity calculation uses the amount (#) of occurrences of concept c_j and c_r in the domains of the axiom divided by the total amount of atoms. For com_2, rule

Table 2.1 Rules for annotated pairing (R1–R5) and partially-annotated pairing (R6–R10)

Rule	Rule name	Novelty	Similarity value
R1	sameConcept	Adapted	1 (match found) or 0 (no match)
R2	equivalentConcept	Adapted	1 (match found) or 0 (no match)
R3	subConcept	Adapted	$\frac{\#c_i \text{ in subconcept definitions of concept } c_k}{\#\text{subconcepts of } c_k}$
R4	axiom-based	New	$\frac{\#c_r \text{ and } \#c_j \text{ in domain of atoms in axiom } a_t}{\#\text{atoms in axiom } a_t}$
R5	union	Adapted	1 (match found) or 0 (no match)
R6	linguistic	Old	$\frac{2*\sum_{t1\in v}[\max_{t2\in s} similarity(t1,t2)]}{\#token_v+\#token_s}$
R7	tokenOntology	Adapted	$\frac{2*\sum_{t1\in c}[\max_{t2\in s\|v} similarity(t1,t2)]}{\#token_c+\#token_{s/v}}$
R8	elimination	Adapted	0
R9	dataFlow	New	sim_i
R10	correlation	New	sim_i or cf. R6 or R7

R4 combines *TaskVar.assignee* annotated by (b) with *Employee* annotated by (d) exploiting the given axiom in *Ont*. A third atom in the axiom models the relation *entrustedWithTask*. The relation is based on the domain concept *Employee*. Thus, rule R4 calculates $sim = \frac{3}{3}$ as all atoms are based on one of the concepts used for annotation.

Annotated Pairing Algorithm

Algorithm 1 illustrates the annotated pairing procedure. It gets annotated process variables PV_{Ont}, annotated operational schema models S_{Ont} and the referred ontology *Ont* as input. It finds all meaningful combinations and stores them together with a similarity value in a triple set COM_{Ont} as output. The concepts used for annotation in PV_{Ont} and S_{Ont} must come from the same ontology to be able to apply the following rules.

In a first step, the algorithm identifies all used concept groups (*Groups*) within ontology *Ont*. Concept groups are sets of ontology concepts which are used for annotation of one or more variables or models containing equivalent concepts in one group. The algorithm calls function *getEquivalentAndUnionConcepts* in order to find all concept groups for the given annotation sets with elements referring to the *equivalent* or *union* concepts.

Algorithm 1: Annotated pairing

Input: AnnotationSet PV_{Ont}, AnnotationSet S_{Ont}, Ontology *Ont*
Output: CombinationTripleSet COM_{Ont}
  $Groups \leftarrow getEquivalentAndUnionConcepts(PV_{Ont}, S_{Ont}, Ont)$
  for all concept in Ont **do**
    $cG \leftarrow findGroup(concept, Groups)$
    if $|cG|$ is not empty **then**
      $COM_{Ont} \leftarrow addSameUnionEquivalentTriples(cG)$
      $COM_{Ont} \leftarrow addSubConceptTriples(cG, Ont)$
      $COM_{Ont} \leftarrow addAxiomTriples(cG, Groups, Ont, COM_{Ont})$
    end if
  end for

Then *Ont* is traversed. For this traversal, we build a spanning tree of the ontology graph with view to its subconcept structure. The name of the ontology defines the root. To obtain the tree we only use simple subconcept relationships, i.e., concepts that are deduced only from one concept branch. Complex subconcepts (concepts with more than one parent from different ontology branches) are allocated to the first referred sub concept in the description.

Function *findGroup* estimates for each concept in *Ont* whether it belongs to a concept group. If so, we store the concept group in *cG* and apply our rule set for *cG*. This way, all combination triples are determined in an iterative way, until we have processed all rules. First, combination triples in a concept group are searched (rules R1 (*sameConcept*), R5 (*union*) and R2 (*equivalentConcept*)) with their similarity value. If a concept group contains both variable and operational model annotation sets, each variable is combined with each schema element in this group (*addSameUnionEquivalentTriples*). Afterwards function *addSubConceptTriples* finds triples where the variable concept and the operational model concept are located in different concept groups in the path from the leaf to the concept group *cG* in *Ont*. As this algorithm works on models instead of data instances, we use simple subconcept relations in rule R3. For complex relations, we mark these combination triples and prove the combinability later in the cleansing phase on the instance data.

Function *addAxiomTriples* is based on rule R4 and parses all axioms in *Ont*. Axioms create new relationships between concepts in an ontology with the use of logical expressions. The function seeks axioms that have one of the concepts in *cG* in the antecedents and a concept of the consequences in another concept group and that are not yet in COM_{Ont}. For simple axioms that consist only of different concepts, compatible elements can be combined, if one concept is used in the annotation set PV_{Ont} and the other one in S_{Ont} and if both axiom concepts contain the same parameters. In complex axioms containing also properties, domain and range concepts of the properties are extracted before concepts of different annotation sets can be correlated if they refer to the same parameters in the axiom. In rule R4, the results are also marked and validated on instance data in the cleansing phase.

Partially-Annotated Pairing Rules

The partially-annotated pairing aims to find combinations between elements in PV and S, which are not or only partially annotated. Rule R6 (*linguistic*) applies common matching techniques to the names of the elements as discussed in other papers. See [13] for the similarity calculation. Rule R7 (*tokenOntology*) adapts R6 and compares the element names with concept names that are associated with one of these elements. This enables the usage of *Ont* for further derivations.

Rules R8–R10 do not obey Definition 5, but have their own similarity calculation. Rule R8 (*elimination*) discards defined elements like preposition and articles [13]. We extend this idea to exclude variables with pure controlling purpose from the result. All combinations containing such an element are eliminated, i.e., their similarity value is set to 0. Rule R9 (*dataFlow*) exploits data dependencies considering assign activities that copy data from one variable v_i to a variable v_j. From the found triples derived for v_i, rule R9 copies an operational match partner and similarity value sim_i to the variable v_j. For com_3 in Fig. 2.4, rule R9 uses com_4 that was found by other combination rules. Rule R9 copies *Automobile.Model* and the similarity 0.85 to *ServiceInfo... request* and receives com_3, due to the assign activity *SetServiceData*. Rule R10 (*correlation*) considers shared aliases referenced in a correlation set of the process. The alias contains property labels for different variables v_i, v_r. If v_i is already in COM its matched element s_j and sim_k are copied to v_r. Otherwise, the labels of the alias and the properties are tried to be matched to all $s \in S$. Finding a partner s_j leads to the triples (v_i, s_j, sim_k) and (v_r, s_j, sim_k).

Partially-Annotated Pairing Algorithm

Algorithm 2 illustrates the partially-annotated pairing procedure based on a *RuleSet* consisting of rules R6–R10. It gets process variables PV and operational schema models S that contain also the annotated sets PV_{Ont} and S_{Ont} as input as well as the referred ontology *Ont* and the process context in P. It finds all meaningful combinations together with their similarity value and stores them in a triple set COM_{Part} as output.

We linearize the given variable and schema trees (function *getList*). Algorithm 2 traverses the given linearizations of the input set PV. We apply our rule set for each variable and schema node. This way, all combination triples are determined in an iterative way, until we have processed all rules.

First, it checks if the current node *pv* can be skipped by rule R8 (*elimination*). Otherwise it traverses the linearizations of the input set S and the rules in the rule set until a combination is found or all rules have been traversed. Once a rule can be applied by function *applyTokenRules*, the combination triple is stored in COM_{Part} with its similarity value calculated by the rules. Finally, the rules R9 and R10 are applied for all found combination triples in COM_{Part} and—if we do not separate the

Algorithm 2: Partially-annotated pairing

Input: VariableSet PV, SchemaSet S, Ontology Ont, Process P
Output: CombinationTripleSet COM_{Part}

```
COM_Part ← ∅
for all pv in getList(PV) do
    if (R8(pv)) = false then
        for all s in getList(S) do
            repeat
                rule ← getNextRule(RuleSet)
                COM_Part ← applyTokenRules(pv, s, rule, Ont)
            until RuleSet fully traversed
        end for
        for all com_i in COM_Ont ∪ COM_Part do
            COM_Part ← applyR9(pv, P, COM_Ont, COM_Part)
            COM_Part ← applyR10(pv, P, COM_Ont, COM_Part)
        end for
    end if
end for
```

execution of both partially-annotated and annotated pairing rule sets—in COM_{Ont}. They evaluate the process context P if pv is able to be combined with a schema element s that was used in com_i.

Filtering

The filtering steps refine the received combination triples by taking the context of the elements into account and allowing users to adjust false and missing combinations. At the end, our framework calculates a weighted *sim* value (Definition 5) and returns the final triple set *COM*.

Structure-Based Rules

The structure-based rules exploit the context of the variable and the operational schema elements given in a combination triple to achieve higher accuracy for the found triples in *COM*. By using the similarity values in Table 2.2, we define a combined structure-based similarity value $sim_{str} = \alpha * sim_{R11} + \beta * sim_{R12} + \gamma * sim_{R13} + \delta * sim_{R14}$ with the weights $\alpha,\beta,\gamma,\delta \geq 0$ and $\alpha + \beta + \gamma + \delta = 1$. In the sample scenario all these weights are equally set to 0.25. The rule R11 (*path*) and the rule R14 (*dataType*) are based on [13]. R11 refines results in *COM* by considering matches between their parent elements, e.g., between their parent names. Rule R14 inspects the similarity between elements by their data type category. Rule R12 (*processStructure*) looks for hints in the control flow. It considers e.g., the names

Table 2.2 Structure-based rules

Rule	Rule name	Novelty	Similarity value
R11	path	Old	$\frac{combined\ sim\ of\ parent\ matches\ of\ v_i\ and\ s_i}{Min(\#pathToRoot(v_i),\#pathToRoot(s_i))}$
R12	processStructure	New	$\frac{combined\ sim\ of\ matches(controlflow(v_i),\ path(s_i))}{\#controlFlowParents(v_i)}$
R13	WSDLAnnotation	New	$\frac{1}{minDistance(c_p,\ c_s)}$ or 0 (if not annotated or $sim_i > 1$)
R14	dataType	Old	1 (same datatype group) or 0 (different datatype)

of partner links or activities that work on the respective variable. Furthermore, it estimates the similarity via matching the names of these process components to the operational partner element or its parents. Rule R13 (*WSDLAnnotation*) exploits the annotation c_p of a process component that employs a variable v_i of a combination $com_i = (v_i, s_i, sim_i)$. It traverses the ontology subtree below concept c_p to find the minimum distance between the annotations c_p and c_s of s_i. sim_{R13} calculates the fraction using this distance between the concepts c_p and c_s.

In com_1 of Fig. 2.4 structure-based similarities $sim_{R11}=\frac{1}{2}$, $sim_{R12}=\frac{1}{2}$, $sim_{R13}=\frac{1}{3}$, and $sim_{R14}=1$ are combined to $sim_{Str1} = \frac{7}{12}$. Rule R11 calculates the minimum path (#pathToRoot(*CID*)=2) to root *RentalSchema* and summarizes the similarity of parent matches, i.e., of *Customer* and *RentalSchema* with parent element names of *custID* in the variable tree *inputData* and receives 1 (*CustomerData* matches *Customer*). R12 counts three control flow parents that use *custID*: *CustomerData*, *CustomerInfoAquisition* and *CarSelection*. As only the first two match with the operational parent *Customer*, we get a combined $sim = 1 + \frac{1}{2} + 0$. Rule R13 results in $\frac{1}{3}$, as the distance in *Ont* from the concept *ID* of *CID* (e) and the concept *CarSelection* that annotates the WSDL operation *initiate* (c) that uses *custID*, adds up to 3. sim_{R14} is *1*, because both *custID* and *CID* have the same datatype integer. The overall similarity sim_1 results in $0.6 * 1 + 0.4 * \frac{7}{12} = 0.83$.

Structure-Based Algorithm

Algorithm 3 illustrates the structure-based filtering procedure. It gets as input the process *P* with its variables *PV* and an operational schema model *S*. Both may contain annotated sets. Further inputs are the referred ontology *Ont*, the annotation set of further process description elements P_{Ont} as well as already found combination triples in COM_{Ont} and COM_{Part}.

The goal of the algorithm is to receive a refinement of these combinations as output COM_{Str}. It traverses the given combinations in $COM_{Ont} \cup COM_{Part}$ found by Algorithms 1 and 2. First, it extracts the variable node *pv* by function *getPV* and the schema node *s* by function *getSValue* of the combination triple com_i with

Algorithm 3: Structure-based filtering

Input: Process P, VariableSet PV, SchemaSet S, COM_{Ont}, COM_{Part}, Ontology Ont, AnnotationSet P_{ont}, weights α, β, γ, δ
Output: CombinationTripleSet COM_{Str}

```
COM_Str ← ∅
for all com_i in (COM_Ont ∪ COM_Part) do
    if sim_i > 0 then
        pv ← getPV(com_i)
        s ← getSValue(com_i)
        psim ← applyPathRule(pv, s, P, S)
        csim ← applyProcessStructureRule(pv, s, P, S)
        osim ← applyWSDLAnnotationRule(pv, s, S, Ont, P_Ont)
        dsim ← getDatatypeSimilarity(pv, s, sim_Str)
        sim_Str ← getSimValue(psim, osim, dsmin, α, β, γ, δ)
        COM_Str ← mergeSimilarity(com_i, COM_Str, sim_Str)
    end if
end for
```

$sim_i > 0$. For each triple, all structure-based rules are applied and a similarity value is calculated in all rules: *psim* (R11), *csim* (R12) and *osim* (R13). *Csim* is estimated by *applyProcessStructureRule* between *pv* and *s* using their related process P and schema set S to find relationships between further process parts and schema elements. *Psim* is estimated by the function *applyPathRule*. The more levels there are in the two tree paths, the more valuable becomes this rule.

Function *applyWSDLAnnotationRule* is illustrated in Algorithm 4. It returns *osim* by extracting all annotated description elements in P_{Ont} that use *pv* via function *getDescriptionElements* and stores them in $P_{Ont|pv}$. Function *getSchemaParents* extracts all parents of the given schema element *s*. For each description element p_i annotated by c_k and all parents in *sParent searchOntologyMatches* aims to find the nearest concept c_j that annotates *sParent*. If it is successful ($c_j \neq \emptyset$), *calculateOntologySimilarity* estimates *osim* as distance between c_j and c_k. Function *getDatatypeSimilarity* applies R14 to determine the correspondence of the datatypes of the found combination elements.

At the end of Algorithm 3, function *getSimValue* calculates the similarity sim_{Str} of the current triple com_i combining all received structural similarity values using the given weights alpha, beta, gamma and delta. Then function *mergeSimilarity* estimates the combination triple set Com_{Str} by merging com_i in conjunction with sim_{Str} with all triples in Com_{Str} watching that one (variable – schema) combination exists only once, namely the combination with the highest similarity value.

Manual Filtering

Given the fact that no fully automatic solution is possible, a user-friendly interface is essential for the practicability of a match system. The graphical user interface of the editor described in [15] provides the user with many ways to influence the

Algorithm 4: applyWSDLAnnotationRule

Input: VariableNode pv, SchemaElement s, SchemaSet S_{Ont}, Ontology Ont, AnnotationSet P_{Ont}
Output: osim

```
osim = 0
P_{Ont|pv} ← getDescriptionElements(pv, P_{Ont})
for all p_i(c_k) in P_{Ont|pv}) do
    sParent ← getSchemaParents(s, S_{Ont})
    c_j ← searchOntologyMatches(sParent, c_k, Ont)
    if c_j ≠ ∅ then
        osim ← calculateOntologySimilarity(c_j, c_k, Ont)
    end if
end for
```

match process. It allows to configure the reasoners before combination, to iteratively refine the proposed correspondences during combination, as well as to manipulate the obtained match results after combination. To provide feedback, the user can remove false matches or add missing ones. The manually added combinations are automatically provided with the highest similarity 1 and are stored in *COM*.

Evaluation

In this section, we discuss the benefits of the combination framework and analyze the effectiveness of the combination rules.

Benefits of the Combination Framework

Pure schema matching would not find any matches at all if we apply it to the raw source process without extracting the XML variables from the BPEL process. In our sample scenario, standard schema matching would perhaps be able to find com_1, if it is able to work on ontology mappings. However, com_2 and com_3 would not be found, as user-defined ontology axioms and process context are usually not considered during schema matching. Instead, we would find a wrong match between *TaskVar.title* and *Customer.title*. Rule R8 excludes this combination.

Applying the pairing and structure-based algorithms to the sample scenario reveals multiple combination triples shown in Fig. 2.4 that turn out to describe the useful relations marked in Fig. 2.3. We may discover in (1) a correlation between the assets of customers (type of credit card) and canceled processes. In order to win wealthy customers, they should be routed to special services. The performance of *ContractNegotiation* in (2) depends on the employee and his skills. To increase

the number of accepted tasks, a reorganization of these roles is needed. In (3), an adequate provisioning of resources is needed for cars with certain features.

While for non-annotated attributes our framework provides partially-annotated combination rules, for annotated process and operational data it critically depends on ontologies and annotations. At the moment we still have an overhead of annotating this data. However, more and more web services will be provided with annotations for enabling a semantic service detection at runtime. So our framework will also benefit from these ambitions.

Experimental Setup

The experimental setup consists of the data models of the business processes, the operational database schema and an ontology for semantic annotation. For evaluation, we used 20 different BPEL processes averaging 10 variables each with 4–10 matchable elements and one schema with about 400 attributes. Processes and schemas come from the car rental domain. The ontology was modeled in WSML and contained all concepts needed for annotation.

To evaluate the quality of our rules, we compared the manually determined real combinations R with the combinations T returned by our tool. We determined the correctly identified combinations C. Based on the cardinalities of these sets, two quality measures are computed (cf. [8]): $Precision = \frac{|C|}{|T|}$ estimates the reliability of the combination predictions, $recall = \frac{|C|}{|R|}$ specifies the share of real combinations that is found. Because of sparse element interrelations in the variables, the structural results are lower weighted as the rest and w_{weight} (see Definition 5) is set to 0.6. Our tool determines T by applying our rules with a threshold of 0.7 that was set based on a significant number of test runs.

Results

In a first set of experiments, we focused on the annotated pairing rules and the structure-based rules applied to annotated elements. Figure 2.5(a) shows the combination results. As expected, the recall is very high because all models are correctly annotated. A detailed analysis reveals that the missing 5 % comes from some axioms defined in the ontology that contain five predicates and some even more in their antecedents resulting in a similarity value below the threshold. The precision results are almost just as well, but some wrong combination results have been found. Due to the missing check via instance data, the rules R3, R4 and R13 found combinations that cannot be validated with concrete data values. A later cleansing phase could improve this result.

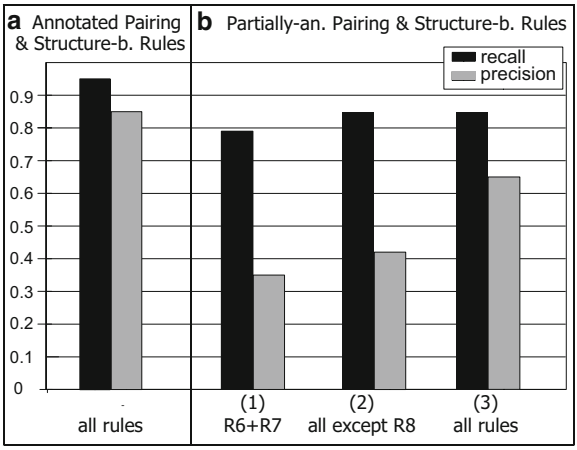

Fig. 2.5 The evaluation of combination rules compares precision and recall for using (**a**) all annotated paring rules and all structure-based rules, and (**b**) subsets of the partially-annotated pairing rules and all structure-based rules

The second set of experiments addresses the significance of rules R8–R10 that exploit process knowledge. It provides the results of partially-annotated-pairing and structure-based rules applied to non-annotated elements. Figure 2.5(b) shows combinations obtained by these rules in three different scenarios: (1) shows the results obtained with rule R6 (*linguistic*) and rule R7 (*tokenOntology*) as the only pairing rules, (2) is based on all partially-annotated-pairing rules except rule R8 (*elimination*) and in the last one (3) all partially-annotated-pairing rules are applied. Comparing recall and precision results of (1) and (2), there is only a small impact of rule R9 and rule R10. Due to our precondition of qualified element names and lexicon definitions, rules R9 and R10 promise an improvement only in few cases where the elements are ambiguous and, thus, have been incorrectly matched before.

In (3), the precision value rises due to the application of rule R8. Wrong combinations with elements in human task variables, e.g., their human task titles, are deleted from the result set. The rules R6 and R7 reveal some problems, since they only find tokens with clear separation hints in the name like underscore or capitalization leading to a still low precision rate. Some other wrong combinations originate from tokens that appear very often in the variables of the rental domain, e.g., the token *car*. This leads to wrong combinations of elements containing this token, e.g., *CarInsuranceID* and *CarSaleID*. A workaround would be to declare *car* and other frequent words as stop word. In some cases, the path rule failed to find structural matches because in flat hierarchies as in operational data models, e.g. of type relational, there are only two possible levels in the path (table and columns).

Comparing the results in Fig. 2.5(a) and (b3) with the results of well-known rules R6 and R7 in Fig. 2.5(b1) shows a big gap in precision. That verifies our claim that the new rules raise precision of the combination. They focus on semantic annotations (Fig. 2.5(a)) as well as on process-specific issues (*elimination*, *data flow*, *correlation* in Fig. 2.5(b3)) that have not been taken into account by well-known rules.

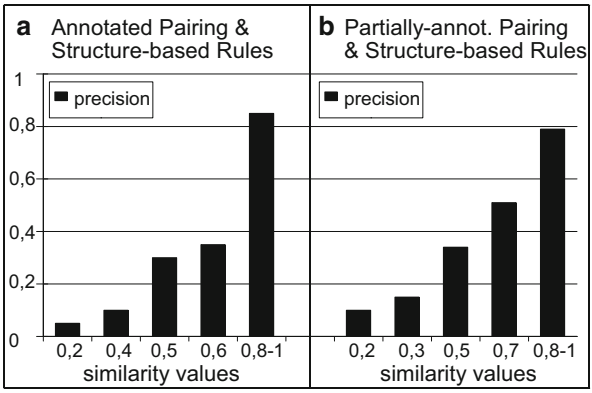

Fig. 2.6 Evaluation of the precision of calculated similarity values. (**a**) Annotated pairing & structure-based rules. (**b**) Partially-annot. pairing & structure-based rules

Faulty calculated similarity values also take effects on precision and recall. The tool finds true combinations in fact, but might exclude them in some cases again due to a low, but wrongly calculated similarity value. Thus, we evaluated the whole result set of all combinations with respect to their precision. Figure 2.6a calculates the precision for each similarity group for all combinations that were found by annotated pairing rules. Figure 2.6b illustrates the same calculations for the partially-annotated pairing rules. In both tables, the precision results for combinations the tool found with a similarity value below 0.7 are very low. In contrast most combinations found with similarity value above 0.7 are correct combinations (high precision). This is why we set the threshold to 0.7 during the experiments. Amongst others, the correct setting of the threshold depends on the complexity of the rules. Further experiments and analysis revealed that in other settings, e.g., if the combinations are basically derived by the *dataFlow* rule using complex assignment activities, the similarity threshold should be adjusted to a lower value.

Conclusion

We have shown a promising approach to combine business process variables with operational schemas. It adds another level of combination on top of well-known schema matching approaches by considering the impact of business process features. Based on a prototype and a case study, we have evaluated that our approach derives significant combination results that previous approaches have not found. In future work, we will extend our combination framework with respect to input models and well-known matching rules.

References

1. Agrawal R et al (1998) Mining process models from workflow logs. In: Schek H-J, Saltor F, Ramos I, Alonso G (eds) 6th international conference on extending database technology, advances in database technology – EDBT'98, Valencia, 23–27 Mar 1998
2. Bernstein PA, Haas LM (2008) Information integration in the enterprise. Commun ACM 51(9):72–79
3. Bruckner RM, List B, Schiefer J (2002) Striving towards near real-time data integration for data warehouses. In: 4th international conference on data warehousing and knowledge discovery, DaWaK 2002, Aix-en-Provence, 4–6 Sept 2002
4. Cardoso J, Sheth AP (2006) Semantic web services, processes and applications. Springer, New York
5. Casati F et al (2007) A generic solution for warehousing business process data. In: Koch C et al (eds) Proceedings of the 33rd international conference on very large data bases, University of Vienna, Vienna, 23–27 Sept 2007
6. Castellanos M, Casati F, Dayal U, Shan M-C (2004) A comprehensive and automated approach to intelligent business processes execution analysis. Distrib Parallel Databases 16(3):239–273
7. Corrales JC et al (2008) BeMatch: a platform for matchmaking service behavior models. In: Kemper A et al (eds) 11th international conference on extending database technology EDBT 2008, Nantes, 25–29 Mar 2008
8. Do H, Melnik S, Rahm E (2003) Comparison of schema matching evaluations. In: Web, web-services, and database systems. Springer, Berlin/Heidelberg/New York
9. Do H, Rahm E (2002) COMA – a system for flexible combination of schema matching approaches. In: Proceedings of 28th international conference on very large data bases VLDB 2002, Hong Kong, 20–23 Aug 2002
10. Doan A et al (2004) Ontology matching: a machine learning approach. In: Handbook on ontologies (International Handbook on Information Systems). Springer, Berlin/Heidelberg
11. Dong X et al (2004) Similarity search for web services. In: Nascimento MA et al (eds) (e)Proceedings of the thirtieth international conference on very large data bases, Toronto, 31 Aug–3 Sept 2004
12. Hepp M et al (2005) Semantic business process management: a vision towards using semantic web services for business process management. In: Lau FCM, Lei H, Meng X, Wang M (eds) 2005 IEEE international conference on e-business engineering, ICEBE 2005, Beijing, 18–21 Oct 2005
13. Madhavan J, Bernstein PA, Rahm E (2001) Generic schema matching with Cupid. Technical report, Microsoft Research
14. Radeschütz S, Mitschang B (2008) An annotation approach for the matching of process variables and operational business data models. In: Harris FC Jr (ed) Proceedings of the ISCA 21st international conference on computer applications in industry and engineering, CAINE 2008, Honolulu, 12–14 Nov 2008
15. Radeschütz S et al (2010) BIAEditor – matching process and operational data for a business impact analysis. In: Manolescu I et al (eds) 13th international conference on extending database technology EDBT 2010, Lausanne, 22–26 Mar 2010
16. Radeschütz S, Vrhovnik M, Schwarz H, Mitschang B (2011) Exploiting the symbiotic aspects of process and operational data for optimizing business processes. In: Proceedings of the IEEE international conference on information reuse and integration, IRI 2011, Las Vegas, 3–5 Aug 2011. IEEE Systems, Man, and Cybernetics Society
17. Rahm E, Bernstein PA (2001) A survey of approaches to automatic schema matching. VLDB J 10(4):334–350
18. Rubin V et al (2007) Process mining framework for software processes. In: Wang Q, Pfahl D, Raffo DM (eds) International conference on software process, software process dynamics and agility ICSP 2007, Minneapolis, 19–20 May 2007

19. Sayal M, Casati F, Dayal U, Shan M-C (2002) Business process cockpit. In: Proceedings of 28th international conference on very large data bases VLDB 2002, Hong Kong, 20–23 Aug 2002
20. Schiefer J, Jeng J-J, Bruckner RM (2003) Real-time workflow audit data integration into data warehouse systems. In: Ciborra CU et al (eds) Proceedings of the 11th European conference on information systems, ECIS 2003, Naples, 16–21 June 2003
21. van der Aalst WMP (2001) Re-engineering knock-out processes. Decis Support Syst 30(4):451–468
22. van der Aalst WMP (2011) Process mining: discovery, conformance and enhancement of business processes. Springer, Berlin/Heidelberg/New York
23. Vrhovnik M et al (2007) An approach to optimize data processing in business processes. In: Koch C et al (eds) Proceedings of the 33rd international conference on very large data bases, University of Vienna, Vienna, 23–27 Sept 2007
24. W3C (2007) Semantic annotations for WSDL and XML schema. Available: http://www.w3.org/TR/sawsdl/
25. Weerawarana S et al (2005) Web services platform architecture. Prentice Hall, Upper Saddle River
26. zur Muehlen M (2004) Workflow-based process controlling. Logos, Berlin
27. zur Muehlen M, Shapiro R (2009) Business process analytics. In: Handbook on business process management, vol 2. Springer, Berlin

http://www.springer.com/978-3-7091-1537-4

Information Reuse and Integration in Academia and Industry

Özyer, T.; Kianmehr, K.; Tan, M.; Zeng, J. (Eds.)

2013, XII, 306 p. 103 illus., Hardcover
ISBN: 978-3-7091-1537-4