IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2020, accepted May 7, 2020, date of publication May 11, 2020, date of current version June 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993883

Business Process Execution From the Alignment Between Business Processes and Web Services: A Semantic and Model-Driven Modernization Process

ENCARNA SOSA SÁNCHEZ, PEDRO J. CLEMENTE, JOSÉ M. CONEJERO, AND ÁLVARO E. PRIETO

Department of Computer Science, Quercus Software Engineering Group, University of Extremadura, 10003 Cáceres, Spain

Corresponding author: Encarna Sosa Sánchez (esosa@unex.es)

This work was funded by the Ministry of Science and Innovation (MCI), for the State Research Agency (AEI)-Project RTI2018-098652-B-I00; the Government of Extremadura, Council for Economy, Science and Digital Agenda under the grants GR18112 and IB18034, by the European Regional Development Fund (ERDF) and by the 4IE+ project (0499-4IE-PLUS-4-E) funded by the Interreg V-A Spain-Portugal (POCTEP) 2014-2020 program.

ABSTRACT Many companies have implemented their business processes in Web applications which must be frequently adapted so as to stay aligned with new business process requirements. Service-oriented architectures (SOAs) constitute an appropriate option to manage the continuous changes in those processes by facilitating their alignment with the changing underlying system services. In this context, firms are trying to migrate their Web applications to new software architectures such as SOAs. However, this migration is usually carried out ad-hoc by means of non-reusable and error-prone manual processes. Similarly, the alignment between the business processes and the underlying services identified is usually done by hand. This work presents a model-driven semiautomatic approach to modernize legacy Web applications to SOAs. The approach is focused on an automatic semantic process aimed at discovering the services that can be used to implement the business processes (defined by the companies), then aligning these processes with the underlying services. A semantic algorithm is provided to aid the migration architect during the alignment process. The case study carried out shows that the alignment process results obtained by the semantic algorithm presented in this paper are similar to those obtained by the experts manually. Finally, SOA orchestration artifacts are generated from the semantic algorithm results.

INDEX TERMS Legacy Web applications, service-oriented architecture, semantic algorithms, model-driven techniques.

I. INTRODUCTION

Enterprise Web applications must be continuously adapted to deal with the frequent modifications in business processes due, among others, to changes related to company policies, client requirements, relations with stakeholders, etc. [1]. However, most of these applications lack a flexible architecture that allows their quick adaptation to these changes mainly because they are not usually aligned with the company business processes [2], [3]. In this context, Service Oriented Architectures (SOAs) [4] came to the scene as an important paradigm to solve this problem by providing mechanisms not only to publish and orchestrate services offered by organizations but also to properly align them with the company business processes [5].

Thus, in the context of Service Oriented Architectures (SOA), Business Process Model and Notation (BPMN) is used to describe and execute the companies business processes. BPMN notation therefore creates a standardized bridge for the gap between the business process design and process implementation. In this sense, several tools allow executing BPMN such as Bonita Soft,[1] Apache Activiti,[2]

The associate editor coordinating the review of this manuscript and approving it for publication was Zhangbing Zhou.

[1] https://www.bonitasoft.com/
[2] https://www.activiti.org/

Camunda,[3] etc. In order to do this, BPMN models should be defined at a high operational level [6], that is, defined with a high level of detail.

In that sense, there is a current trend in companies towards the modernization of their Legacy Web Applications (LWAs) to SOAs in order to obtain the benefits claimed by this paradigm, namely interoperability, extensibility, modifiability, reliability, availability, usability, scalability, adaptability, and deployability [7] and minimize the costs of adapting their systems to continual changes in business rules. However, these modernizations are often defined as ad-hoc processes which lead to expensive and error-prone projects [8], [9]. In other words, the industry requires formal and standardized modernization or migration processes in order to reuse redundant tasks appearing recurrently in these processes and that must be defined once and again for each new project. These migration processes must include an analysis of the services offered, the implementation of the wrappers for each service, the analysis of the business processes, and the alignment with any previous service identified.

The alignment between business processes and the underlying services of the LWA is one of the most complex task [5] of these processes. This task requires an exhaustive search on service repositories to identify what services are suitable to execute a specific business process task. So, this alignment process requires checking each business process task with each service signature in order to guarantee that all the business process tasks can be suitability executed by the underlying services. The more business process tasks and services there are, the more effort is required.

Indeed, it is usually carried out by experts who must search through large service repositories and services offered by their legacy applications with the goal of identifying the most suitable one for a particular business process task [8], [10]–[12]. Note that SOAs may even integrate services that are available on a global scale (external services such as social networks services or services deployed in the cloud by third-party companies) with services identified in the legacy application. Therefore, the available information is wide and difficult to compare, and hence the matching process is error prone, which makes more obvious the need of an automatic alignment processes to ease the management of both internal service repositories and external services. In this context, the MigraSOA project, based on Model-Driven techniques, was proposed to modernize and adapt the software architectures of LWAs to SOAs [13]–[15]. Specifically, an automatic semantic process was used to automate the migration of software. The final target was therefore the business process execution which was finally composed of the business process definition (BPMN models) and the underlying web services (SOAP web services) obtained from the legacy web application. The semantic algorithm proposed in the project identified the best web services candidates to implement each specific Service Task defined at the BPMN model. Thus, it allowed defining the link between the business processes and web services. However, two main limitations were identified in this previous work and have been improved in this work. These improvements are mainly related to the semantic algorithm used during the alignment process and are explained in more detail later on.

The BPMN specification [1], [16] includes several kinds of tasks such as Manual, User, Script, or Service Task.

The BPMN models built by companies could be defined in terms of several abstraction levels. Although over time these levels have been named differently as descriptive or final, analytic or operational, executable, etc. one of the initial classifications of these levels was given in [17]. This work defined three levels of abstraction: (i) strategic (covers the activities related to the goals, objectives and policies of the organization), (ii) tactical (deals with the attainment and efficient use of the resources of the organization) and (iii) operational (procures the efficient and effective execution of the specific tasks). Specifically, our approach focuses on BPMN models defined at the operational level [6]. This level is characterized by defining tasks, their relationships and the data involved. Thus, our approach uses Service Tasks defined in the BPMN models in order to invoke the web services defined at the underlying service layer.

To do this, our approach performs the alignment between the Service Tasks (extracted from the BPMN models that are provided by the company) and the available web services (extracted from the legacy web applications). From our point of view, this alignment helps users to perform this task by focusing on the semantic relationships between the defined Tasks and the Services.

The main contributions of this paper, also with respect to previous work, are the following:

- A new semantic algorithm has been proposed to align business processes and the services layer offered by the legacy web application.
- A metamodel called BP-WS has been introduced to manage the alignment between business processes and services.
- A model weaving and a model to text transformation process have been defined in order to generate the SOA orchestration code from the alignment process.
- A new validation of the process has been performed based on a case study.

The rest of paper is structured as follows. Section II analyses similar approaches and related work. Section III briefly introduces the MigraSOA approach in order to provide the background and make the paper self-contained. Section IV deeply describes both the new algorithm for the alignment between business processes and the underlying services and the BP-WS metamodel to support the process. Section V presents the results of applying the algorithm to a case study. Section VI describes a model to text transformation in order to generate the SOA orchestration code from the alignment process. Section VII discusses

[3]https://camunda.com/

implications whilst section VIII presents the main limitations of our proposal. Finally, Section IX concludes the paper and presents further work.

II. RELATED WORK

The concept of *service discovery* is related to processes looking for a specific service in large service repositories using concrete queries such as services name, parameters name, etc. The alignment process is an extension of service discovery because the information used by the service discovery is given by the business processes. So both concepts should be properly managed with the aim to develop an alignment process from the business processes to the underlying services in a SOA. In this context it is interesting to identify in literature the service discovery approaches that could be used as a basis to provide an alignment process. Furthermore, services are usually defined using code or service descriptions like WSDL [18] while business processes are defined using well-known models like BPMN [16]. Thus, a mechanism to manage both kinds of artifacts is needed. In this sense, both services descriptions and business processes descriptions could be managed as models [16], [19]. In this way, works related with model matching may be suitable to implement the alignment process. They facilitate the identification of model elements and relationships in each defined model. Thus, model matching and model-driven technologies help SOA analysts to tackle the complex relationships between both models, focusing on essential information and leaving the specific technology issues to be managed by model-transformations.

Both, service discovery and model matching usually require a semantic analysis to improve the match performance.

Thus, the first part of this section involves a general review of semantic algorithms. Then, the second part of this section looks at different approaches to develop a service discovery process whose main goal is to find a suitable service in a service repository. Finally, works on matching business process models are considered. Both strategies must be taken into account in order to obtain executable business processes.

A. SEMANTIC ALGORITHMS AND ONTOLOGY MAPPINGS

Ontology matching aims to solve semantic heterogeneity problems such as ambiguous entity names, different entity granularity, incomparable categorization, and several instances of different ontologies [20]. Thus, semantic algorithms have been widely used to tackle several domain areas as ontology evolution [21], ontology integration [22], data integration [23] and data flow specifications (ETL jobs) [24]. Others have been applied on peer-to-peer information sharing [25], composition and coordination of web services [26] or ontologies comparison [27].

In order to achieve these goals several tools, techniques and algorithms have been discussed in [20], [28]. These reports [20], [28] sum up the ontology matching state of the art and include the following information: edit distance, I-SUB, V-Doc (Virtual Documents), Monger and Elkan similarity measure, Jaccard index, Vector distance, Object similarity, String equality, Winkler-based string metrics or substrings, Wang and Ali algorithm, etc. Additionally, the Ontology Alignment Evaluation Initiative (OAEI) [29] organizes the evaluation of several ontology matching systems. Specifically, different ontology matching systems applied in defined test cases are compared. These test cases can be based on ontologies of different levels of complexity and use different evaluation modalities.

In [30] the authors analyse specifically the performance of the string similarity metrics in the name-matching tasks. They compare the performance of: i) string metrics according to edit distance metrics such as Levenshtein, Jaro-Winkler, Needleman-Wunsch, Smith-Waterman and N-gram, ii) token-based distance metrics such as TFIDF, Cosine, Jaccard index, and iii) hybrid metrics such as Monge-Elkan measure. However, these algorithms do not take into account semantic issues such as synonyms or hyponyms which are common problems that should be analysed and resolved. In this sense, in [20] the authors said that only string similarity metrics do not provide good performance for matching. So, to increase the performance, string text processing techniques such as token-based techniques, synonymy, hyponymy, abbreviation, stop word removal, stemming and translation must be taken into account to align the business processes tasks with the web services.

For instance, token-based distance algorithms such as TFIDF (Term Frequency-Inverse Document Frequency) have been widely used in long texts to data retrieval and generally not as a measure of similarity, but to assess the relevance of a term in a document. Nevertheless, although TFIDF vector-space method is based on token-string algorithms, it can also be enhanced by looking for synonyms, hyponyms and hypernyms in WordNet. For instance, in [31] the TFIDF has been extended to support the use of synonyms. However, this algorithm is used to measure the relevance of a term in a document.

On the other hand, algorithms such as [27], [32]–[35] tackle the semantic similarity of terms based on WordNet or specific dictionaries. They are based on ontologies and hierarchical semantic knowledge, defining direct relations among elements and taken into account the path to achieve a specific concept. However, business process information and web services information usually do not have information related to hierarchical semantic knowledge. So, these algorithms could be adapted to be applied to align business processes tasks and the information technology offered by SOAP web services.

B. SERVICE DISCOVERY APPROACHES

In the work of Perin-Souza and Rabelo [36], a service discovery process is defined, taking into account the context of the application of the business processes, and including several non-functional requirements such as quality of service. The proposed service discovery process involves a

semantic algorithm whose inputs are: (i) services defined in WSDL/UDDI (Universal Description, Discovery, and Integration) or federated services, (ii) business processes defined by the UBL (Universal Business Language) ontology, and (iii) a search string related to the service being searched for. The process returns a list of candidate services that perform semantic matching with the UBL and QoS (Quality of Service) match functions. However, on the one hand, the authors do not provide information about how the semantic matching is carried out from business processes to services defined in the services repository. On the other hand, it should be noted that non-functional information is not always available for each service.

Others approaches to implement service discovery can include: i) a simple keyword-based and category-based search on UDDI; ii) identifying with Woogle [37] in WSDL (Web Services Description Language) artifacts that recommend similar services; iii) or the specialized framework proposed by Hatzi *et al.* [38] to retrieve services in both WSDL and OWL-S (Web Ontology Language for Services) standards by extending and adapting the TF-IDF (Term Frequency–Inverse Document Frequency) model. Other approaches base themselves on semantic languages: OWLS-MX (hybrid matchmaker for OWL-S services) [39]. In the work of Hobold and Siqueira [40], SAWSDL (Semantic Annotations for WSDL) semantic annotations are used to automatically find service compositions. However, these approaches have the drawback that, since semantic Web services are not widely extended, many Web service repositories do not consider semantic Web services.

The authors Li *et al.* [41] address the automation of the process of analysing and ranking services to retrieve those most closely related to the query. They analyse the features describing the functional attributes of Web services, and propose a probabilistic framework and Web service retrieval model. Since the work does not take into account specific parameters, data types, or order, it is unsuitable for the alignment of business process tasks and the Web services, although it could be used to recommend Web services on the basis of a user query. Wang *et al.* [42] apply some algorithms to extract common topic groups from Web service description documents. These common topic groups are used in order to minimize the number of candidate Web services during the process of Web service discovery. The algorithms used in this work are: (i) Latent Dirichlet Allocation (LDA) [43], used to extract unobserved groups that explain why some parts of the documents are similar, (ii) Correlated Topic Model (CTM) [44], for modeling relations between topics by replacing the Dirichlet distribution with the logistic normal distribution and (iii) the Biterm Topic Model (BTM) [45], for dealing with the short message problem.

The SWSD framework [46] proposes a keyword-based discovery process for Web service searching. The services are described using semantically enriched annotations. It makes intensive use of natural language processing techniques and a WordNet-based [47] similarity measure to match keywords (using ontology-based semantic matching algorithms). However, ontologies are not widely used, and the services defined do not normally include semantic annotations. Chen *et al.* [48] present a novel semantic similarity measure, SIMCR, for semantic Web service discovery. This similarity measure takes into account different conceptual relationships in ontologies such as *is-a*, *has-a* and *antonomy*. This semantic similarity of terms is measured (as in the previous work) on the basis of generic WordNet ontology.

Finally, Leopold *et al.* [49] focus on service identification from business process model repositories. They do not align business processes and services, but they define an automatic service derivation process to identify candidate services. To this purpose, they define a semantic algorithm using BabelNet [50] to manage the word senses defined at business processes labels. Related with the business processes, they should be defined at an operational level [6] which includes whole information of each task. Our objective is not to identify potential services from the business processes, but to align the business processes and the underlying services layer already implemented.

Tibermacine *et al.* [51] propose an identification process of web service substitutes for healing failed web service orchestrations based on the measurement of similarity between service interfaces. Specifically, they consider the substitution of one service by another or other several services. They use WSDL interface description to obtain the information of a failed web service then search in a service pool (web services candidates) that offer the same or related functionalities. They use a filtering algorithm that exploits the similarity assessment technique proposed in [52] where the authors use different structure and semantic similarity metrics to conduct a similarity assessment between different WSDL parts (operation, messages, parameters, type, etc.).

Table 1 summarizes the works studied that are related to service discovery. Each work has been studied from several points of view: i) The service discovery is based on services defined using *WSDL*; ii) The service discovery manages *Semantic Web Services*; iii) The service discovery uses a *Semantic Algorithm*; iv) The service discovery facilitates the *Business Process Alignment*; v) The service discovery takes into account *Non-functional Properties*.

MigraSOA [13] allows to describe the services using well-known service description languages (WSDL) while the business processes are defined using BPMN. The alignment process includes a semantic algorithm which will be presented in next section.

C. USING MODEL MATCHING TO IMPLEMENT SERVICE DISCOVERY

On the one hand, the business processes are defined using Business Process Model Notation (BPMN) models and, on the other hand, a conceptual representation of the underlying services defined using WSDL could be obtained [13]. Thus, model-driven techniques could be applied with the aim

TABLE 1. Comparison of works related to service discovery (X means "fully covered" and – means "partially covered").

	WSDL	Semantic WS	Semantic Algorithm	Business Process Alignment	Non-functional Properties
Souza et al. [36]	X	-	X	X (business process ontology)	X
Woogle [37]	X	-	-	-	-
Ourania Hatzi et al. [38]	X	X (OWL-S)	-	-	-
OWLS-MX [39]	-	X (OWL-S)	-	-	-
SAWSDL [40]	-	X (SAWSDL)	-	-	-
Li et al. [41]	X	-	X (probabilistic)	-	-
Wang et al. [42]	X	X (WSDL SAWSDL)	X	-	-
SWSD [46]	-	X (OWL-S)	X	-	X
Chen et al. [48]	X (OWLS-TC repository)	X	X	-	-
Leopold et al. [49]			X		
Tibermacine et al. [51]	X		X		
MigraSOA	X	-	X	X (BPMN)	-

to manage both kinds of models. Each model is based on a specific metamodel, however, model transformations and model matching could be defined with the aim to tackle the technology complexity, focusing on the relationships between model elements. As a consequence, works related with model matching can help us to develop the alignment process using model-driven techniques. It must be taken into account that both models (BP and services models) are conformed to different metamodels and defined at different abstraction levels.

Pietsch *et al.* [53] present a work where they emphasize the potential importance of semantic algorithms to support BPMN model comparison. They also identify a group of general matching problems (move, rename, etc. applied to elements) in model comparison algorithms that deliver low quality results. They describe applications in two research projects, and present examples working in BPMN. The operators (move, rename, move renamed, exchange location, update target reference) used in this approach do not allow any service discovery to be defined due to they are very basic operators to manage models. Alanen and Porres [54] calculate the difference between models conforming to the same metamodel, and describe a merging algorithm based on a set of operations (add, delete, modify, etc. applied to elements in the model) performed on the original model and producing a target model. Primarily, the algorithm lacks any semantic basis, and the operators used are applied to models based on the same metamodel. Kolovos *et al.* [55] provide an overview of the existing approaches to identifying matching model elements. They emphasize the importance of considering the semantics of the modeling language to improve the accuracy of the results.

Shahzad *et al.* [56] use five word-level WordNet semantic similarity measures (Resnik [33], Jiang and Conrath [57], Leacock and Chodorow [58], Lin *et al.* [35] and Wu and Palmer [59] similarity) and three sentence-level aggregation techniques (Greedy Pairing [60], Optimal Matching [61] and Quadratic Assignment Problem [62]) to evaluate the effectiveness of different combinations in the context of Process Model Matching (PMM). They also emphasize the importance of considering the semantics on PMM. Meilicke *et al.* [63] address the problem of the varying performance of individual process model matching techniques. To this end, they introduce a matching approach that uses the correspondences generated by a set of matchers as input. The authors highlight that more complex semantic relations are required to express interesting links between activities. To do that, it would be necessary to support different semantic relations (for example, relations with relation-specific cardinality constraints).

We agree with the use of semantic operators to carry out the model matching because business-process and service models may be defined on the basis of different metamodels (and probably with different terminology) even when they are both defined on the same domain.

Maoz *et al.* [64] develop semantic diff operators for comparison of models conforming to the same metamodel. They work with BPMN and workflow examples, and use their own modeling language and semantic domain. This is similar to the present work except that we use models conforming to different metamodels. Xing and Stroulia [65] proposes UMLDiff, a general framework for model comparison. Like our work, Xing uses an index (the Jaccard coefficient) to measure the similarity between sets of words. However, we use a semantic basis to improve the similarity measure in our algorithm.

As a summary, although our approach also performs matching between models (BPMN models and a representation of the underlying services denominated the Simple-SoaML model), it does so without imposing the requirement that the models are defined at the same level of abstraction. This is the reason why a semantic approach is needed to align individual elements between models, and to develop a service discovery system.

TABLE 2. Comparison of works related to model matching.

	Comparison/Alignment	Matching Process Analysing Entities
Pietsch et al. [53]	Ecore/BPMN	Name similarity
Alanen et al. [54]	Models (XMI)	Name similarity
Shahzad et al. [56]	BPMN	Semantic similarity (word and sentence level)
Meilicke et al. [63]	BPMN	Sequential consistency constraints
Maoz et al. [64]	BPMN/Workflows	Semantic
Xing et al. [65]	Software O.O.	Similarity (name and structure)
MigraSOA	BPMN + WS	Semantic

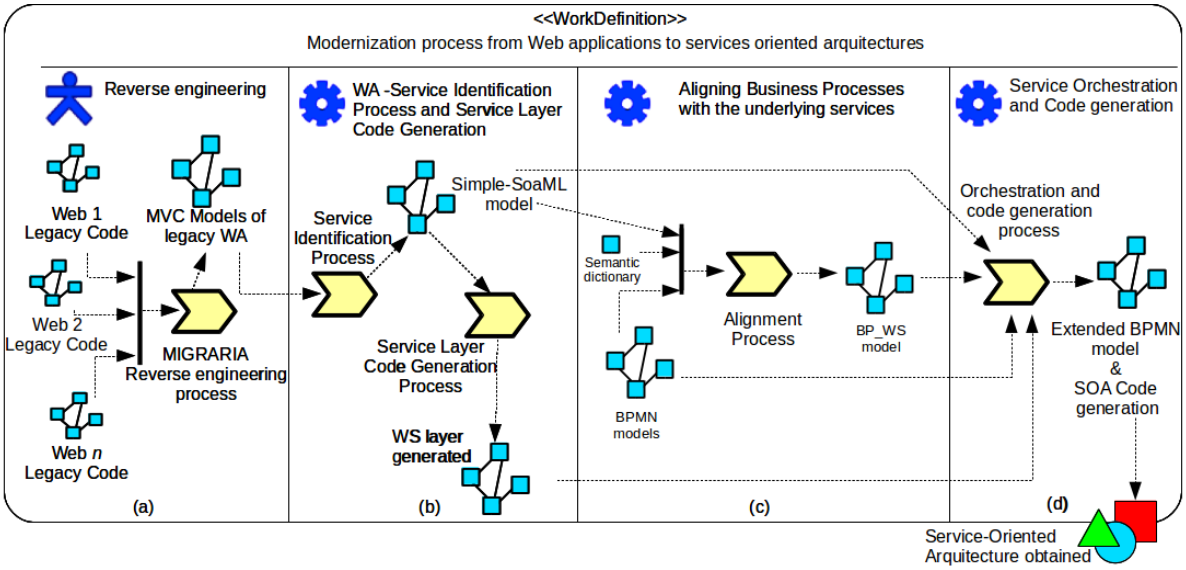

FIGURE 1. Overview of the migration of Web applications to SOA processes.

Table 2 summarizes the works studied that are related to model matching, and how those approaches allow similar elements among models to be identified. Each work has been studied taking into account the following features: i) The artifacts used to *CompareAlignment*; ii) The mechanism used to *Matching Process or Analysing Entities*.

The above review of related work has shown that the main open research issues in this context concern matching of models that are based on different metamodels. The use of semantic algorithms in implementing model matching makes it possible to align associated concepts with each other. Moreover, for service models in particular, neither non-functional properties nor semantic Web service descriptions (OWL-S or similar) have any place in the definition of the service layer. Thus, the service models used in the service discovery process should have common, WSDL-based, service descriptions which could be found in any Web service repository.

III. AN OVERVIEW OF THE MigraSOA APPROACH

To make the paper self-contained, this section provides a brief overview of the main steps defined in the MigraSOA approach [13] in order to carry out the proposed modernization process from legacy web applications to Service Oriented Architectures (SOA). These steps are depicted in Figure 1 and they are the following:

a. Reverse engineering. This step represents the starting point of the approach (identified as *(a)* in Figure 1) and is based on the application of reverse engineering techniques to the company's LWAs $(1, \ldots, n)$ to extract a model-based specification from them. This step reuses the re-engineering methods and tools previously developed in the MIGRARIA project [2], [3] to obtain this conceptual representation of the services offered by the legacy Web sites. The result is a set of technology-specific models that are then processed to obtain an integrated technology-independent model that conforms to a metamodel denominated MIGRARIA_MVC [2].

b. Service identification and service layer code generation. This step (identified as *(b)* in Figure 1) also uses reengineering techniques to obtain new models from the WAs where the different services identified are labelled. This step is of utmost importance for the approach since the appropriate service identification allows to create the service layer. The model obtained in this step (named Simple-SoaML) conforms to the Simple-SoaML metamodel that represents the underlying services

identified. The Simple-SoaML model is used to define a model-to-text transformation to automatically generate the code for the underlying service layer. In other words, this generation provides an interoperable layer to interact with the core legacy website functionality [13], [66]. The service layer generated is based on SOAP web services [18], [67], [68] where each web service includes its own WSDL (Web Service Description Language). Note that ''a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards'' [69].

c. Aligning business processes with the underlying services. The aim to this step (identified as *(c)* in Figure 1) is to align the new business processes that the company has identified and the service layer generated in the previous step. This step is represented in Figure 1 by the Service discovery algorithm [4] that takes as inputs: i) the company's business process models, defined by the company in BPMN notation, ii) the Simple-SoaML model obtained in step *(b)* representing the underlying services and iii) a semantic dictionary used in the aligning process (it will be described in section IV-C). As previously mentioned, this process is based on a semantic algorithm and it provides two types of possible outcomes: i) business process tasks already aligned with discovered services, and ii) business process tasks that have not been aligned with any service yet. Both types of tasks are included in a model called BP-WS (described in section IV-E) in which the relations between business process tasks and these services are represented.

d. Service orchestration code generation. The service orchestration layer is obtained by a model weaving process which weaves the BPMN and the BP-WS model obtained in step *(c)* [15]. The result of this weaving is a new model denominated Extended BPMN Model. This model integrates the original BPMN models with the aligned services, concretely, the BPMN models are updated by introducing into them the suitable XML code to invoke the target services. In addition, a model to text transformation from this BP-WS model generates the Java classes that wrap each service invocation from the specific BPMS. As a consequence, these new BPMN models could be executed by a BPMS (BPM Suite). In our case, these models are executed by Apache Activiti [70].

As previously mentioned, this paper focuses on step *(c)* in Figure 1 where an alignment process is designed using a semantic service discovery algorithm. In this step (c), two main limitations identified in previous work [13]–[15] and introduced in section I have been overcome: i) the alignment process core, the semantic algorithm, proposed in [14], [66] compared directly the labels of the business process tasks with the web service method signature, without taking into account additional relationships like number of common synonyms or main name in synonyms which could be measured; ii) there were not specific tools with an enough high abstraction degree to support the alignment information obtained. The former has been solved by extending the semantic algorithm presented in [32] so that a new algorithm has been introduced in the approach. To deal with the latter, a metamodel called BP-WS to relate business process models and web service models is defined. It allows managing the alignment information at a high abstraction level, facilitating the use of model driven technologies (model to text transformations) in MigraSOA step *(d)*.

Thus, this work takes as starting point the results obtained by the two first steps and it assumes that the process of identifying the services from the LWA has been properly carried out [13]. In addition, step *(d)* in Figure 1 is described in Section VI.

Note that SOA systems are usually defined based on four different layers (business processes, composite services, low level services (or atomic services) and business systems (legacy systems) [71], [72]. Low-level services typically define the underlying service layer that interacts directly with the legacy system while composite services are services based on low-level services. In MigraSOA [13] the underlying services layer is automatically obtained from the legacy system, so in this approach the four traditional layers are reduced to three: business processes, services layer and business systems (legacy systems). As a consequence, the business processes are aligned with these low-level services. Next, we describe how to identify the underlying services layer and to generate its implementation.

A. CASE STUDY TO ILLUSTRATE THE MIGRATION PROCESS

In order to illustrate the process, a case study has been used that will drive the explanation throughout the paper. This case study is based on a Web application that implements a Conference Review System (CRS) as set out in the First International Workshop on Web-Oriented Software Technology[5] where it was used as a common system for illustrating all the approaches presented.

Thus, the main functionality of the application is the creation and maintenance of the different entities needed to manage a conference reviewing processes, such as *conferences*, *subjects*, and *tracks* which are related to *papers*, *reviewers*, etc. Obviously the management of these entities include all the typical operations related to them, for instance creating, updating, reading, or deleting elements as well as accepting or rejecting papers.

[4]Note that while service identification refers to the identification of the services offered by the underlying Web application, service discovery algorithm refers to the automatic matching and alignment between these services and the business processes.

[5]http://users.dsic.upv.es/ west/iwwost01/

TABLE 3. Basic writing-task rules used to model business processes.

BP element	Notation
Task	[Infinitive verb†]+ [-all] + [object] e.g. Create Conference
Notification tasks	[Notification] + [object] + [OK/Error] e.g. Notification Conference OK
Check tasks	[Check] + [object] + [infinitive verb†] + [-all] e.g. Check Paper Copy
Task parameters	[Parameter1,] [Parameter2,][...] e.g. ID, location, name
Gateway labels†	[Infinitive verb†] + [-all] + [object] +[Ok] +['?'] e.g Create Conference Ok?

† phrasal verbs are written with a dash.

We selected the CRS Web application as our case study because it has the following characteristics:

- It offers multiple services (conference management, paper management, author management, etc.).
- It is a data-driven application.
- It was developed using Struts 1.x, a well known and widely used Model-View-Controller (MVC) framework.

After applying the first two steps (*a* (of the MIGRARIA approach) and *b*) to the case study, the obtained results are: on the one hand, an underlying web services layer is generated offering services to manage *conferences*, *subjects*, *tracks*, *papers* and *reviewers*; on the other hand, a Simple-SoaML model conforms to a Simple-SoaML metamodel [13] is obtained which represents the web services defined from a model-based perspective. The obtained web services should be used for the implementation of the company business processes. However, the implementation of these services requires the alignment process that will be described in the next sections.

IV. ALIGNMENT OF BUSINESS PROCESSES WITH THE SERVICE LAYER USING A SEMANTIC SERVICE DISCOVERY ALGORITHM

This section shows the alignment process defined in the MigraSOA approach. By this process, the Simple-SoaML model and the business processes obtained by the previous steps in the approach are matched. So, the alignment process uses the business processes provided by the company. However, the alignment process could be also considered as an interesting opportunity to improve the business processes of the company by adapting them to new requirements or just improving them by incorporating new functionality. For example, the company could make adaptations to its original business processes adding activities related to social networks, such as Twitter or Facebook.

Before the alignment process is explained, the next subsection describes some rules that should be considered in order to obtain better results.

A. BUSINESS PROCESSES BASIC WRITING RULES

Since business processes can be modeled by different people in a company, our approach proposes some basic writing-task rules for unifying the style in modeling business processes, as it is also suggested in [73]. These writing-task rules mainly define how to write the expressions describing tasks, parameters, and gateways. They are summarized in Table 3. These kinds of style rules are usually established by software factories where many people collaborate in the development of business processes. For example, in business processes the labels are usually built by two words: an action followed by an object [73]. Although business processes developers can define any element proposed in BPMN specification such as Tasks, Gateways, Objects, Messages, etc. [16], we should also mention that the alignment process only takes into account the tasks of *Service* type, excluding those that are *Manual* or *User* type.

B. CASE STUDY REVISITED

Following the rules defined in the previous subsection, Figures 2 and 3 show the BPMN models defined for the *Create conference* and *Get conference* processes of our case study respectively. Note that these business processes have been modeled at an operational level, that is with a high degree of detail [6].

To illustrate the different activities involved in the alignment process, some specific parts of the BPMN models shown in Figures 2 and 3 are used. In particular, we focus on how to align the *Create Conference* and *Obtain Conference* tasks (marked by a rectangle in the diagrams) with specific services. This activity is performed by discovering services in the layer which could implement the functionality defined in those tasks. The Create Conference business process model is defined to create a new Conference checking if the conference exists before create it.

C. A SEMANTIC DICTIONARY FOR THE DOMAIN

To carry out the alignment process, the semantic algorithm, that will be later described, is based on the use of a semantic dictionary. This dictionary complements the inputs of the algorithm, which consist of: (i) the semantic dictionary, (ii) the Simple-SoaML model and (iii) the business process model defined.

The semantic dictionary[6] defines the binding among a term and its related terms within a domain. The main advantage of using a semantic dictionary is that it helps developers and analysts to define BPMN models using a

[6]https://goo.gl/K8gHdf

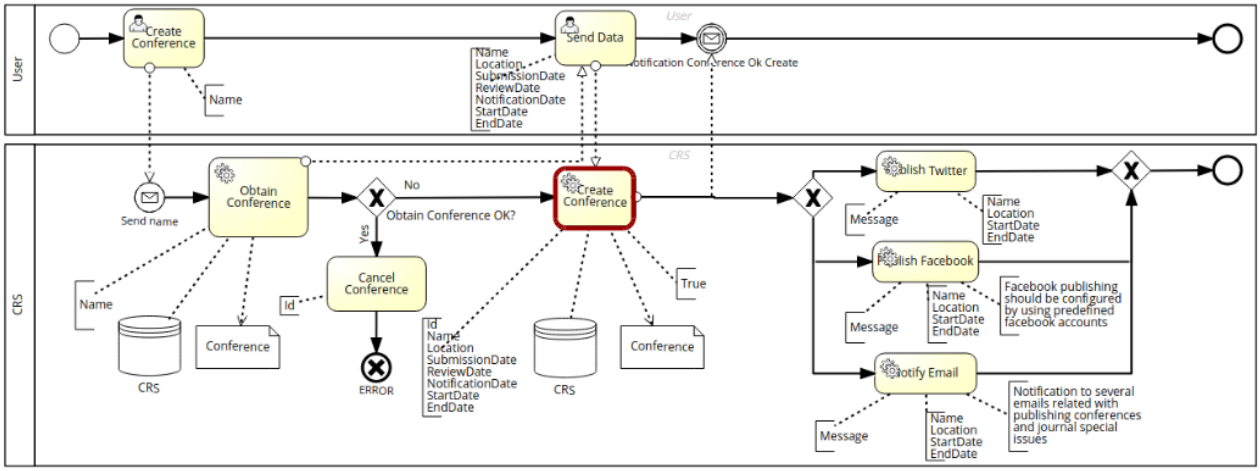

FIGURE 2. BPMN diagram representing the Create Conference process (including the *Create Conference* task).

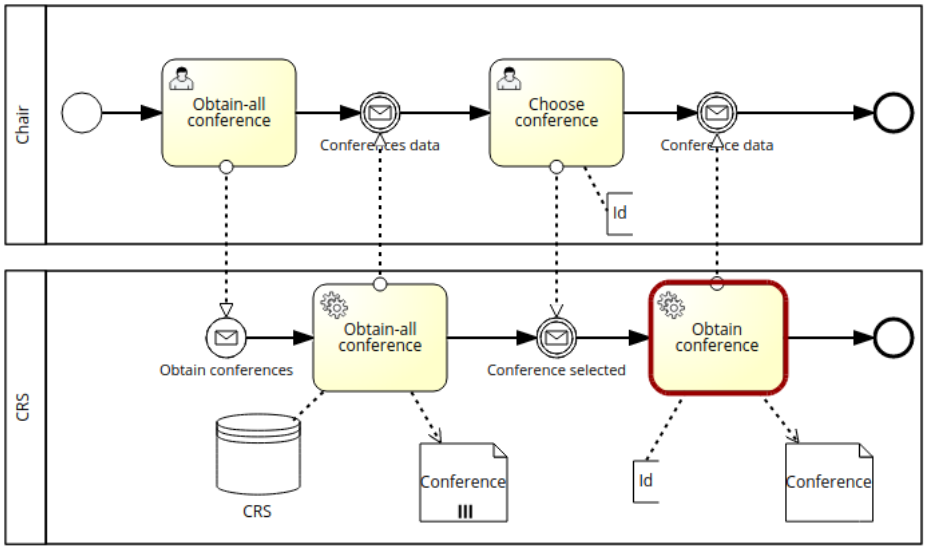

FIGURE 3. BPMN diagram representing the Get Conference process (including the *Obtain Conference* task).

common language. It is worth to mention that a general purpose dictionary would not provide enough expressiveness to describe the relationship among terms for a concrete domain (e.g. synonyms or meronyms are not usually defined on general dictionaries) and this is why a domain-based one is used. Concretely, the semantic dictionary for the particular domain is defined by business process analysts and domain experts so that the relations taken into account in the design of the semantic dictionary are those due to not only terms' closeness in accordance with the domain but also synonymy, hyponymy, meronymy or abbreviations. As an example, related to the term *conference*, the semantic dictionary stores synonymous terms (such as *seminar* or *symposium*) but also concepts with other semantic relations (meronyms such as *workshop*, hyponyms such as *meeting*, and abbreviations such as *conf*).

Additionally, in the case of our particular domain, the content of the semantic dictionary has been subjected to a validation process by an expert judgement.[7] In this validation process, two different aspects were evaluated: (i) the suitability of the inclusion of each term in the dictionary, and (ii) the proper inclusion of the different synonyms for each term.

D. A SEMANTIC SERVICE DISCOVERY ALGORITHM

One of the Service Discovery goals is to improve the search and discovery of services deployed in a system, which are registered on a repository, for instance in a service registry or as in our case, the underlying services deployed in the system and registered in a model. The way to improve searching

[7]https://sites.google.com/site/migrasoa/crs-case-study/crs-validation

services involves the semantic issues that can be found on the services interfaces syntax. Thus, using semantic algorithms to measure the similarity of concepts is the key element in a service discovery approach because the same concepts could be defined using other words. As a consequence, we need to implement algorithms that tackle the semantic similarity of terms. For instance, [27], [32]–[35] are based on ontologies and hierarchical semantic knowledge, defining direct relations between elements and taking into account the path to achieve a specific concept. However, these algorithms should be adapted to be applied to align business processes tasks with the information technology offered by SOAP web services because of the business process information and the web services information do not usually offer information related to hierarchical semantic knowledge.

In [74] a semantic comparison between ontologies based on WordNet was carried out. The authors identified the *Dieng and Hug* algorithm [32] and the *Wang et al.* algorithm [27] as suitable semantic algorithms candidates to comparing ontologies. In [14] we proposed a service discovery using a semantic algorithm based on [27]. In the work [66], we extended the semantic algorithm proposed in [14] to take into account the possible semantic incompatibilities that could arise between business process tasks and web services (incompatibilities between their names, parameters and entire signatures have been taken into account). The algorithm was also validated using a case study related to the University context. In this paper, as a paper contribution we have made an optimization of our semantic algorithm (the original and extended algorithm) [14], [66] using an adaptation of [32] to take into account hierarchical relationships between business process tasks and web services. Later on we sum up the results obtained with both algorithms.

Thus, in this work the semantic service discovery algorithm is optimized using an extension of a previously presented semantic algorithm [32] (henceforth, the *Dieng-Hug* algorithm). This algorithm was originally designed for helping a knowledge engineer to build a common ontology based on several experts' personal ontologies that are represented using the Sowa's conceptual graph formalism [75]. In order to determine the similarity between the concepts of each ontology based on their synonyms and hierarchical relationships, the algorithm uses two concept hierarchies: $Hier_1$ and $Hier_2$. Based on these hierarchies, the identification process defined by the algorithm has two phases:

Phase 1 Terminology-based identification. This phase uses the names to compare any type of $Hier_1$ with any type of $Hier_2$.

Phase 2 Context-based identification. This phase uses the context in the hierarchy of two concepts, i.e., their "relatives" such as direct supertypes and subtypes, in order to determine whether they are representing the same concept or not.

The original algorithm has been modified to be adapted to a modernization process and its specific features. We denote by $\mathcal{T}$ the set of N business process tasks: $\mathcal{T} = \{T_i\}_{i=1}^{N} = \{T_1, \ldots, T_N\}$, in which T_i denotes the i-th task of the set $\mathcal{T}$, and we denote as $\mathcal{S}$ the set of M services: $\mathcal{S} = \{S_j\}_{j=1}^{M} = \{S_1, \ldots, S_M\}$, in which S_j denotes the j-th service of the set $\mathcal{S}$. In that sense, the algorithm is adapted to compare each task and its set of synonyms identified in the business processes (as the first set of terms) with each identified service and its set of synonyms (as the second set of terms). The set of synonyms of a term t is composed by those terms that are related to t according to the different relationships considered in the dictionary.

Moreover, since business process tasks (represented in accordance with the writing-task rules given in Table 3) and services (modelled in Simple-SoaML) are frequently identified by composite words, the process needs to identify each word and assign it a specific weight in order to properly apply the algorithm. As an example, the *createSymposium* service could be divided into two words: create (action) and Symposium (data-object). This is one of the main characteristics that should be adapted in the *Dieng-Hug* algorithm [32] in order to apply it to business processes. So, in our version of the *Dieng-Hug* algorithm the concepts should be tokenized to manage individually the semantic comparisons. Then a deep description about the *Dieng-Hug* algorithm adaptation is carried out.

As aforementioned, we apply the Phase 1 of the algorithm [32] using as hierarchies the terms of our two data sets (business process tasks and identified services). Note that the application of Phase 2 of the algorithm (context-based identification) is limited in our domain since the concepts defined in a BPMN model are at the same semantic level. In Phase 1, the algorithm identifies which terms of the two data sets are "similar" according to their main names and their sets of synonyms. Given the business process task set, $\mathcal{T} = \{T_i\}_{i=1}^{N} = \{T_1, \ldots, T_N\}$, and the service set $\mathcal{S} = \{S_j\}_{j=1}^{M} = \{S_1, \ldots, S_M\}$, we divide the names of tasks and services into actions and objects. Thus, let $A_i^{\mathcal{T}}$ be the action of a task $T_i \in \mathcal{T}$ and $O_i^{\mathcal{T}}$ the object of a task $T_i \in \mathcal{T}$. Let also $A_j^{\mathcal{S}}$ be the action of a service $S_j \in \mathcal{S}$ and $O_j^{\mathcal{S}}$ the object of a service $S_j \in \mathcal{S}$.

Then, for every task T_i where $i = 1, \ldots, N$, and for every service S_j where $j = 1, \ldots, M$ we provide the following definitions:

1) The name of a task $T_i \in \mathcal{T}$ is: $name(T_i)$.
2) The number of business process tasks is given by the cardinality of the set $\mathcal{T}$: $Card(\mathcal{T}) = \mid \mathcal{T} \mid = N$.
3) Let $A_i^{\mathcal{T}}$ be an action for a task $T_i \in \mathcal{T}$. The set of the synonyms of $A_i^{\mathcal{T}}$ is the following: $Syn(A_i^{\mathcal{T}}) = \{Sy_1(A_i^{\mathcal{T}}), \ldots, Sy_{L_i}(A_i^{\mathcal{T}})\}$.
   The number of synonyms of $A_i^{\mathcal{T}}$ for a task T_i is given by the cardinality of the set $Syn(A_i^{\mathcal{T}})$: $Card(Syn(A_i^{\mathcal{T}})) = L_i$.
4) Let $O_i^{\mathcal{T}}$ be an object for a task $T_i \in \mathcal{T}$. The set of the synonyms of $O_i^{\mathcal{T}}$ is the following: $Syn(O_i^{\mathcal{T}}) = \{Sy_1(O_i^{\mathcal{T}}), \ldots, Sy_{L^i}(O_i^{\mathcal{T}})\}$.

The number of synonyms of $O_i^{\mathcal{T}}$ for a task T_i is given by the cardinality of the set $Syn(O_i^{\mathcal{T}})$: $Card(Syn(O_i^{\mathcal{T}})) = L^i$.

5) The set of synonyms of a task T_i is the following union: $Syn(T_i) = Syn(A_i^{\mathcal{T}}) \bigcup Syn(O_i^{\mathcal{T}})$.
6) The name of a service $S_j \in \mathcal{S}$ is: $name(S_j)$.
7) The number of services is given by the cardinality of the set $\mathcal{S}$: $Card(\mathcal{S}) =| \mathcal{S} |= M$.
8) Let $A_j^{\mathcal{S}}$ be an action for a service $S_j \in \mathcal{S}$. The set of the synonyms of $A_j^{\mathcal{S}}$ is the following: $Syn(A_j^{\mathcal{S}}) = \left\{Sy_1(A_j^{\mathcal{S}}), \ldots, Sy_{R_j}(A_j^{\mathcal{S}})\right\}$.
The number of synonyms of $A_j^{\mathcal{S}}$ for a service S_j is given by the cardinality of the set $Syn(A_j^{\mathcal{S}})$: $Card(Syn(A_j^{\mathcal{S}})) = R_j$.
9) Let $O_j^{\mathcal{S}}$ be an object for a service $S_j \in \mathcal{S}$. The set of the synonymous of $O_j^{\mathcal{S}}$ is the following: $Syn(O_j^{\mathcal{S}}) = \left\{Sy_1(O_j^{\mathcal{S}}), \ldots, Sy_{R^j}(O_j^{\mathcal{S}})\right\}$.
The number of synonyms of $O_j^{\mathcal{S}}$ for a service S_j is given by the cardinality of the set $Syn(O_j^{\mathcal{S}})$: $Card(Syn(O_j^{\mathcal{S}})) = R^j$.
10) The set of the synonymous of a service S_j is the following union: $Syn(S_j) = Syn(A_j^{\mathcal{S}}) \bigcup Syn(O_j^{\mathcal{S}})$.
11) The number of synonyms common to a task T_i and a service S_j is given by the intersection: $Card(Syn(T_i) \bigcap Syn(S_j))$.

To determine the similarity between a task $T_i \in \mathcal{T}$ and a service $S_j \in S$, three criteria are taken into account. A function $\mathcal{T} * S \rightarrow N$ is defined and an associated weight (a real number) is assigned for each criterion, as will be explained below. Since each T_i and S_j contains an action and an object, we added the assignment of a configurable weight to the identified action and another to the data-object on which the action is performed. These weights are denominated *aw* and *ow*, respectively. Initially, we assign a higher weight to the object because synonyms of *actions* are only considered when they are followed by objects that belong to the same set of synonyms. For instance, in the comparison between the *Create Conference* and *createPaper* terms, the create actions are not considered to be synonyms (since they are applied to objects that are not synonyms).

- **Criterion 1.** Same main name (SMN). T_i and S_j have the same main name: $name(T_i) = name(S_j)$. Given that we divide the names of tasks and services into two parts (actions and objects), we added a condition that also checks and takes into account whether they are exactly the same action or the same object.

$$SMN(T_i, S_j) = \begin{cases} 1, & \text{if } name(T_i) = name(S_j) \\ aw, & \text{if } A_i^{\mathcal{T}} = A_j^{\mathcal{S}} \\ ow, & \text{if } O_i^{\mathcal{T}} = O_j^{\mathcal{S}} \\ 0, & \text{otherwise,} \end{cases}$$

with $aw \in [0, 1]$ and $ow \in [0, 1]$.

In this criterion, if task's name is identical to service's name, the value obtained is the maximum, that is, 1. Otherwise, if only the actions or objects are identical, the value obtained by the algorithm is the maximum value between *ow* or *aw*. As we said before, we usually assign a higher weight to the object, therefore, the value obtained will normally be *ow*. However, note that it could be possible to obtain the value of *aw* when *aw* is higher than *ow*.

- **Criterion 2** Number of common synonyms (NCS). In the original *Dieng-Hug* algorithm the number of common synonyms of T_i and S_j (i.e., the intersection of their synonyms) must be greater than a given threshold *th* (defined taking into account the mean value of synonyms for the terms of the dictionary). Given that we divide the names of tasks and services into two parts (actions and objects), we added a condition that also checks and takes into account the number of common synonyms of the action and the object in each T_i and S_j (i.e., in $A_i^{\mathcal{T}}$, $O_i^{\mathcal{T}}$, $A_j^{\mathcal{S}}$ and $O_j^{\mathcal{S}}$). The threshold we used is the arithmetic mean because this measure represents the mean value of the synonyms of all terms since the dictionary has approximately the same number of synonyms for each term. The domain expert could also manually introduce this threshold.
To define the next function, $NCS(T_i, S_j)$, and to calculate the number of common synonyms between T_i and S_j, we have adapted the original algorithm and used the separated sets of synonyms of T_i and S_j respectively.
Therefore, we have defined this function, $NCS(T_i, S_j)$ as follows:
If $|Syn(A_i^{\mathcal{T}}) \cap Syn(A_j^{\mathcal{S}})|+ |Syn(O_i^{\mathcal{T}}) \cap Syn(O_j^{\mathcal{S}})| \leq th$, we will say that $NCS(T_i, S_j) = 0$. Otherwise:

$$NCS(T_i, S_j) = \begin{cases} 1, & \text{if } (A_i^{\mathcal{T}} = A_j^{\mathcal{S}} \text{ or } A_i^{\mathcal{T}} \in Syn(A_j^{\mathcal{S}})) \\ & \text{and} \\ & (O_i^{\mathcal{T}} = O_j^{\mathcal{S}} \text{ or } O_i^{\mathcal{T}} \in Syn(O_j^{\mathcal{S}})) \\ aw, & \text{if } A_i^{\mathcal{T}} = A_j^{\mathcal{S}} \text{ or } A_i^{\mathcal{T}} \in Syn(A_j^{\mathcal{S}}) \\ ow, & \text{if } O_i^{\mathcal{T}} = O_j^{\mathcal{S}} \text{ or } O_i^{\mathcal{T}} \in Syn(O_j^{\mathcal{S}}). \end{cases}$$

- **Criterion 3,** Main name in synonyms (MNIS). In the original *Dieng-Hug* algorithm this function is defined as: 1 if the main name of one term belongs to the set of synonyms of the other term: $name(T_i) \in Syn(S_j) \vee name(S_j) \in Syn(T_i)$, or 0 otherwise. Given again that tasks name and services name are divided into actions and objects, we have made two adaptations to the original algorithm. Firstly, we have used again the separated sets of synonyms of T_i and S_j. Secondly, we have added a condition that also checks whether they

are exactly the same action or the same object in T_i and S_j.

$$MNIS(T_i, S_j) = \begin{cases} 1, & \text{if } ((A_i^{\mathcal{T}} \in Syn(A_j^{\mathcal{S}}) \text{ and } A_j^{\mathcal{S}} \in Syn(A_i^{\mathcal{T}})) \\ & \text{and} \\ & (O_i^{\mathcal{T}} \in Syn(O_j^{\mathcal{S}}) \text{ and } O_j^{\mathcal{S}} \in Syn(O_i^{\mathcal{T}}))) \\ & \text{or} \\ & (A_i^{\mathcal{T}} = A_j^{\mathcal{S}} \text{ and } O_i^{\mathcal{T}} = O_j^{\mathcal{S}}) \\ aw, & \text{if } A_i^{\mathcal{T}} = A_j^{\mathcal{S}} \\ & \text{or} \\ & (A_i^{\mathcal{T}} \in Syn(A_j^{\mathcal{S}}) \text{ and } A_j^{\mathcal{S}} \in Syn(A_i^{\mathcal{T}})) \\ ow, & \text{if } O_i^{\mathcal{T}} = O_j^{\mathcal{S}} \\ & \text{or} \\ & (O_i^{\mathcal{T}} \in Syn(O_j^{\mathcal{S}}) \text{ and } O_j^{\mathcal{S}} \in Syn(O_i^{\mathcal{T}})). \end{cases}$$

The semantic similarity function $SS : \mathcal{T} * \mathcal{S} \rightarrow R$ calculates the similarity factor between a term of $\mathcal{T}$ and a term of $\mathcal{S}$, in accordance with the following formula:

$$\begin{aligned} SS(T_i, S_j) = & RMN * SMN(T_i, S_j) \\ & + RCS * NCS(T_i, S_j) \\ & + RMNIS * MNIS(T_i, S_j), \end{aligned}$$

RMN, *RCS and RMNIS* $\in$ [0 1] are the weight associated with the functions $SMN(T_i, S_j)$, $NCS(T_i, S_j)$ and $MNIS(T_i, S_j)$, respectively. *RMN, RCS and RMNIS* are values manually introduced by the domain expert. Therefore, the maximum value of similarity between a term of $\mathcal{T}$ and a term of $\mathcal{S}$ is the sum of the weights associated with the three functions (i.e., *RMN*, *RCS*, and *RMNIS*). We considered as optimal the values that are higher than the 80% [76] of the maximum value. We have normalized the maximum value of the sum of *RMN*, *RCS*, and *RMNIS* so that *RMN* + *RCS* + *RMNIS* is 1. For instance, if the result of the previous addition were 0.9, then the optimal values would be those between 0.72 (80% of 0.9) and 0.9. In other words, optimal values are those which allow results similar to those obtained by a manual alignment. However, in order to select the most suitable weights for *RMN, RCS* and *RMNIS*, a proof analysis was performed considering several case studies and varying these values. As an example, values lower than 80% of the maximum value imply an increment of false positives in our case study. This 80% has been successfully used in [76].

The results of the similarity function are stored by the *Dieng-Hug* algorithm in a similarity matrix denominated DH-SM, with $Card(\mathcal{T})$ rows (number of terms in the business process tasks set) and $Card(\mathcal{S})$ columns (number of terms in the services set), so that the i-th row and j-th column of this matrix contains the value $DH - SM_{i,j} = SS(T_i, S_j)$, i.e., the index of similarity given by the *Dieng-Hug* algorithm between each business process task and each service of the service layer.

TABLE 4. Example result of alignment between the *Create Conference* task and the *createSymposium* service (weights: *RMN* = 0.8, *RCS* = 1, *RMNIS* = 0.8).

Task	Service	SMN	NCS	MNIS
Create Conference	createSymposium	0.704	1	0.8

TABLE 5. Example result of alignment between the *Obtain Conference* task and the *getSymposium* service (weights: *RMN* = 0.8, *RCS* = 1, *RMNIS* = 0.8).

Task	Service	SMN	NCS	MNIS
Obtain Conference	getSymposium	0.640	1	0.8

As an example, the values of the $SMN(T_i, S_j)$, $NCS(T_i, S_j)$, and $MNIS(T_i, S_j)$ functions for the *Create Conference* task and *createSymposium* service are listed in Table 4 (with the weights: $RMN = 0.8$, $RCS = 1$, and $RMNIS = 0.8$), and the corresponding index of similarity

(i.e., $SS(Create\ Conference, createSymposium)$ or $DH - SM_{Create\ Conference,\ createSymposium}$ is 0.83.

Table 5 lists the values of the $SMN(T_i, S_j)$, $NCS(T_i, S_j)$, and $MNIS(T_i, S_j)$ functions for the *Obtain Conference* task and *getSymposium* service (with the weights: $RMN = 0.8$, $RCS = 1$, and $RMNIS = 0.8$), and the corresponding index of similarity

(i.e., $SS(Obtain\ Conference, getSymposium)$ or $DH - SM_{Obtain\ Conference,getSymposium}$ is 0.81.

All these results will be deeply discussed in Section 5 where we present an analysis and validation of the results obtained by the algorithm.

E. THE OUTPUT OF THIS PHASE: BP-WS MODEL

The data obtained from the *DH-SM* matrix contain the basis information for creating a model that relates other two ones: Simple-SoaML and BPMN. Therefore, the output of the phase *c* in Figure 1 is a model conform with the BP-WS metamodel (see Figure 4) which stores the relationships between business processes and the underlying service layer.

The main reasons for obtaining the BP-WS model are: (i) it represents the alignment process result stored as ECore model, allowing model-driven techniques and tools to be used; (ii) the model obtained can be queried using OCL (Object Constraints Language) [77], an OMG (Object Management Group) specification widely used to manage, look up, or define and validate constraints on models; and (iii) other tools based on model-to-text transformations (e.g. based on Acceleo [78]) could be used to generate code for a specific technology, in our case, the code is generated for a specific Business Process Execution engine as Apache Activiti.

Figure 4 shows an excerpt of the BP-WS metamodel where some concepts related to the Simple-SoaML or BPMN metamodels may be also identified. In particular:

- *Task* refers to the element with the same name in the BPMN metamodel.

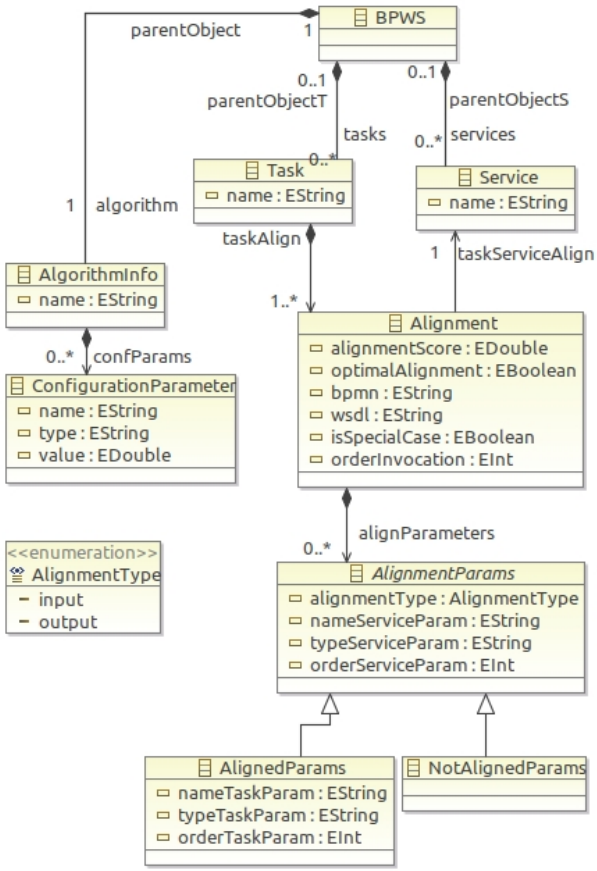

FIGURE 4. BP-WS metamodel.

- *Service* refers to this element in the Simple-SoaML metamodel.
- *Alignment* and *AlignmentParams* elements refer to the relation between these concepts. They contain information about the alignment value calculated by the semantic algorithm. In addition, there is other information that indicates whether this value is optimal or not (*optimalAlignment* attribute) or if this alignment is a "special case" (*isSpecialCase* attribute). A more detailed explanation of these attributes and the different alignment situations that arise are explained in the previous work [66]. In [66] some types of incompatibility are analyzed:
  - incompatibility of the tasks and services names,
  - incompatibility of a pair of parameters (taking into account features such as their names, data types and order in the parameter list),
  - incompatibility of the task and service whole signature. To ensure that the entire task and service signatures can be aligned, the list of parameters is reviewed. For this reason, the *AlignmentParams* element is defined in the BP-WS metamodel. A "special case" of alignment arises when a task is not completely aligned with any of the available services, but it could be implemented by means of several sequentially invoked services. In these cases, the *isSpecialCase* and *orderInvocation* attributes are used to define the services involved.
- The metamodel also contains information on the algorithm used the semantic alignment, such as name and the possible values for comparison (*AlgorithmInfo* and *ConfigurationParameter* elements).

Given this, the model based on the metamodel BP-WS stores the following information: (i) information about the alignment values obtained between business process tasks and identified services, so that one can derive (using OCL queries) both (a) information about the aligned business process tasks, and (b) information about the business process tasks that are not aligned; and (ii) information on the semantic algorithm used. As an example, the following OCL query allows obtaining the list of unaligned tasks:

```
(a) Task.allInstances()->select(t | t.
    taskAlign->forAll(optimalAlignment =
    false))
     or
     //from BPWS context
     self.tasks->select(t |
 t.taskAlign->forAll(optimalAlignment =
     false))
```

The following query allows the migration architect to identify the tasks aligned with the underlying services layer:

```
(b) Task.allInstances()->select(t | t.
    taskAlign->exists(optimalAlignment =
    true))
     or
     //from BPWS context
    self.tasks->select(t | t.taskAlign->
       exists(optimalAlignment = true))
```

To list the services name or the *wsdl* involved in all optimal alignments, the following OCL queries could be defined:

```
(c) Task.allInstances().taskAlign->
    select(optimalAlignment = true).
    taskServiceAlign.name
     Task.allInstances().taskAlign->
         select(optimalAlignment = true).
         wsdl
     or
     //from BPWS context
     self.tasks.taskAlign->select(
         optimalAlignment = true).
         taskServiceAlign.name
     self.tasks.taskAlign->select(
         optimalAlignment = true).wsdl
```

In addition, OCL *invariants* guarantee the model validation. As an example, the following OCL constraints validate the special alignment case (that is when an *Alignment* element is identified as *isSpecialCase=true*). The following two OCL

invariants (d and e) are complementary and are applied in the context of *Alignment* and *Task* respectively.

```
(d) class Alignment{
        ...
        invariant checkSpecialCaseA:
            isSpecialCase = true and
            optimalAlignment = true
            implies  orderInvocation >=
            1;
        ...
    }
(e) class Task {
        ...
        invariant checkSpecialCaseT:
            self.taskAlign->select(a | a.
            isSpecialCase = true and a.
            optimalAlignment = true)->
            size()>1 implies (taskAlign->
            select (a | a.isSpecialCase =
             true and a.optimalAlignment
            = true).orderInvocation->
            asSet()->size() = taskAlign->
            select (a | a.isSpecialCase =
             true and a.optimalAlignment
            = true ).orderInvocation->
            size());
        ...
    }
```

As instances of the alignment values between tasks and services modeled in BP-WS, Figure 5 shows the *Obtain Conference* task, in which the highest value is the alignment corresponding to the *getSymposium* service.

Note that the BP-WS model obtained allows the management of the migration process information from a high abstraction level. In other words, this model enables the utilization of model-driven techniques. For instance, model-to-text transformations may be used to obtain the orchestration code for the underlying service layer but according to the business processes and the alignment indications set out by the process presented here. Therefore, step (*d*) in Figure 1 (Service Orchestration code generation) may be performed by reusing all the knowledge acquired throughout the modernization process since it is stored in the different previous models: Simple-SoaML, BPMN and BP-WS models.

V. ANALYSIS OF THE RESULTS OBTAINED USING THE BP ALIGNMENT PROCESS APPLIED ON A CASE STUDY

This section presents a validation of the results obtained by the application of the BP Alignment process (phase (*c*) in Figure 1) to the whole CRS used as case study. Specifically, the validation is performed by following two different steps. Firstly, a data analysis is performed in which the DH-SM matrix is studied to identify the business process tasks that have been aligned with the underlying services. Secondly,

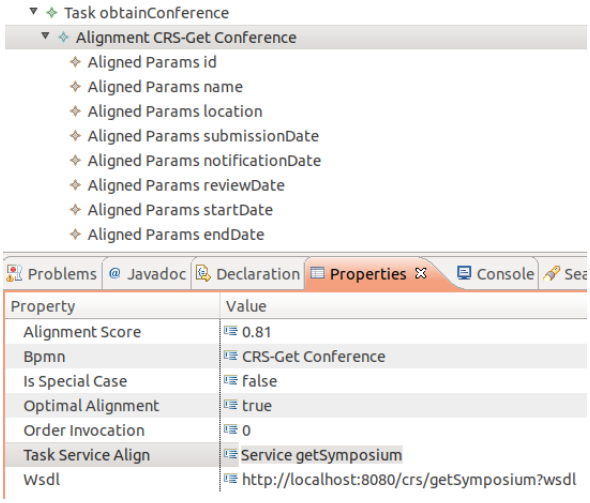

FIGURE 5. The *Obtain Conference* task modeled in the BP-WS model.

the results obtained are compared with those obtained by an alignment carried out by software engineering experts in terms of validating the results obtained by the approach. Finally, the results obtained are compared with the results obtained in [14] where other semantic algorithm was applied to the same case study.

As an example of the analysis performed, the data obtained by the semantic service discovery algorithm for a specific business process task are discussed. Note that the discussion on all the results obtained by the algorithm are not presented here for the sake of brevity. In particular, we firstly focus on the data obtained for the *Obtain Conference* task (Figure 6). In this case, different weights were applied to actions and objects (40% and 60%, respectively) whereas the weights used for the functions in the algorithm were: $RMN = 0.8$, $RCS = 1$, and $RMNIS = 0.8$ (the maximum normalized value is then 0.86, and the optimal values are those that exceed 80% of that value, i.e., 0.69). The reason for using these weights for the functions is that, although being two synonymous terms (the *RMN* function) or belonging to the same set of synonyms (the *RMNIS* function) are important for the results obtained, we consider that having a common set of synonyms (*RCS* function) is of utmost importance for the algorithm. This is why the *RCS* weight is higher than the others. In this case, the alignment between the *Obtain Conference* and *Create Conference* tasks with the identified services provides positive results. Concretely, Figure 6 shows that for the *Obtain Conference* business process task there is not any service with exactly the same name, but there are some services that could be aligned with it such as *getSymposium* (0.81), *createSymposium* and *removeSymposium* (0.29), and others such as *getReviewer* and *getSubject* (0.19). A value of 0 is obtained for the rest of the services. Therefore, the best result obtained is for the *getSymposium* service, with an alignment value above the threshold (0.69).

Similarly, Figure 7 shows that for the *Create Conference* business process task there is not any service with the same

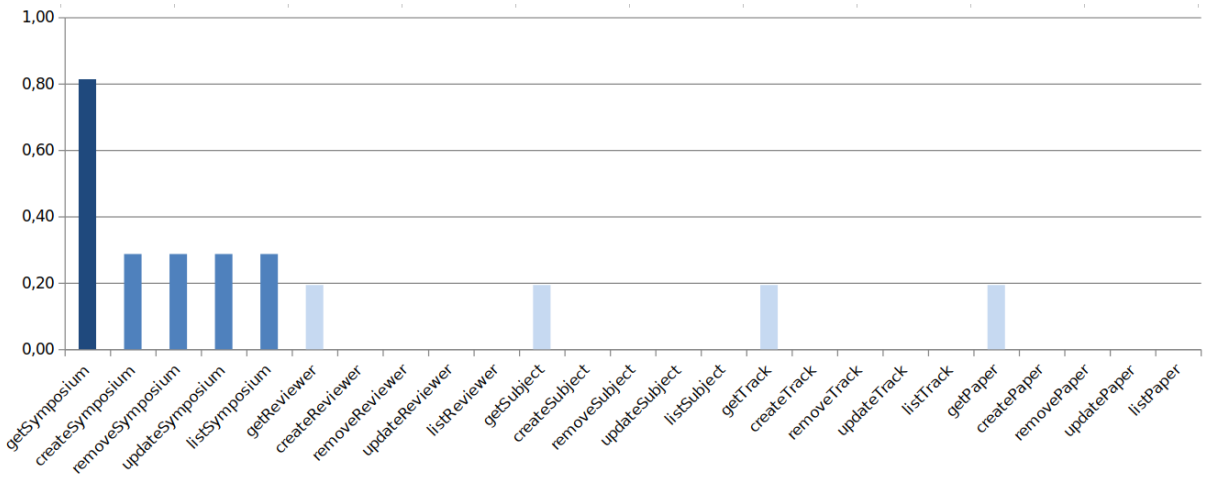

FIGURE 6. Results of executing the extended *Dieng-Hug* algorithm on the *Obtain Conference* business process task with a greater weight for objects (40%-60%) and the function weights as RMN = 0.8, RCS = 1 and RMNIS = 0.8.

name, but there is a service that could be aligned with this task, namely *createSymposium* (0.83). For other services related to the *Conference* object (such as *getSymposium* or *remove*Symposium), the values obtained are 0.29 whereas for services related to the *Create* action (such as *createReviewer* or *createSubject*) values of 0.21 were obtained.

Once the results of the algorithm had been obtained for all the tasks defined in the business process diagrams, we observed that there were ''false positives'' consisting of tasks aligned with services even when their method name does not have the same meaning. An example of this case is the alignment between task *Cancel Conference* and service *removeSymposium* (see Figure 9 column (4)). They are aligned because their actions are synonymous in the dictionary although their meaning is not the same (in this example, *cancel* action occurs when you want to register a conference that already exists, that is, it could be said it is a logical deletion of the object, when *remove* refers to a physical deletion of the object).

Likewise, ''false negatives'' were also found after applying the algorithm identified as tasks that were not aligned with any service since all the alignment values obtained were lower than the threshold, even though they apparently should have been aligned with some services. Figure 8 shows an example of a ''false negative'' with the results for *Exclude Track* task. In this case, only services related to the *Track* object (such as *getTrack* or *createTrack*) have a non-zero value (0.52). However, the *Exclude* term does not match with any synonym due to our dictionary does not include synonyms related to *exclude* term in terms like *remove* or *destroy*, although other terms like *delete* or *erase* are included. Thus, these examples: i) show us possible new terms to be added to our semantic dictionary, and ii) indicate that the dictionary is a key element in applying the semantic algorithm and in aligning the business process tasks with the underlying services, and obviously should be validated by the experts in the specific domain.

To sum up, each row in the similarity matrix (*DH-SM*) provides the results obtained for each business process (susceptible to further analysis) identifying the services that can be aligned with the corresponding business process task. Additionally, in order to facilitate the evaluation of these results, a metric denominated Business Process Alignment (BPA) has been defined that provides a general measure of the proportion of aligned tasks:

BPA = Number of tasks with similarity values above 80% of the maximum similarity value[8] / Total number of tasks[9]

For the CRS example,

$$BPA = 26/34 = 0.7647\ (i.e., 76.47\%)$$

this means that for 76.47% of the tasks defined in the CRS BPMN diagrams, the process discovered a service that implements the task's functionality and that can be found in the service layer generated in phase (*b*) of the MigraSOA project.

In addition, we have used measures such as *Precision* and *Recall* [79] that will help us to measure the alignment process. *Precision* or Confidence (as it is called in Data Mining) denotes the proportion of Predicted Positive cases that are correctly Real Positives [79], that is the number of True Positives divided by the sum of True Positives and False Positives. False Positives are highlighted in green color in Figure 9. Thus, for our CRS example will be:

$$Precision = 24/24 + 2 = 0.92\ (i.e., 92\%)$$

Recall or Sensitivity (as it is called in Psychology) is the proportion of Real Positive cases that are correctly Predicted Positive [79], that is the true positive rate (the number of

[8]It should be remembered that we considered optimal values those higher than 80% of the maximum value [76].

[9]The tasks taken into account are business processes tasks that could delegate their functionality to any service or script. Usually these business process tasks are subtypes of the BPMN task, e.g. Service Task or Script Task.

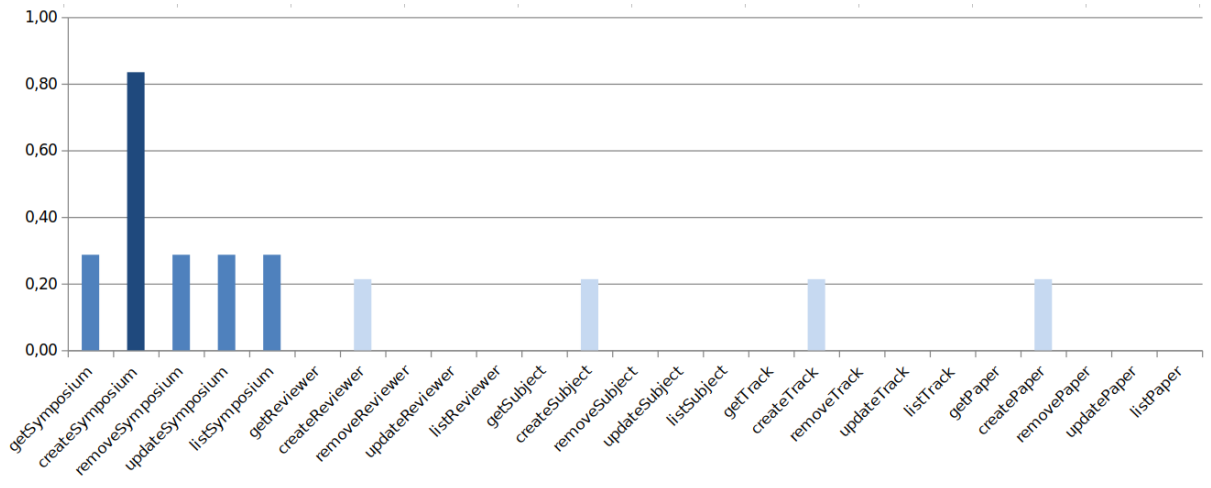

FIGURE 7. Results of executing the extended *Dieng-Hug* algorithm on the *Create Conference* business process task with a greater weight for objects (40%-60%) and the function weights as RMN = 0.8, RCS = 1 and RMNIS = 0.8.

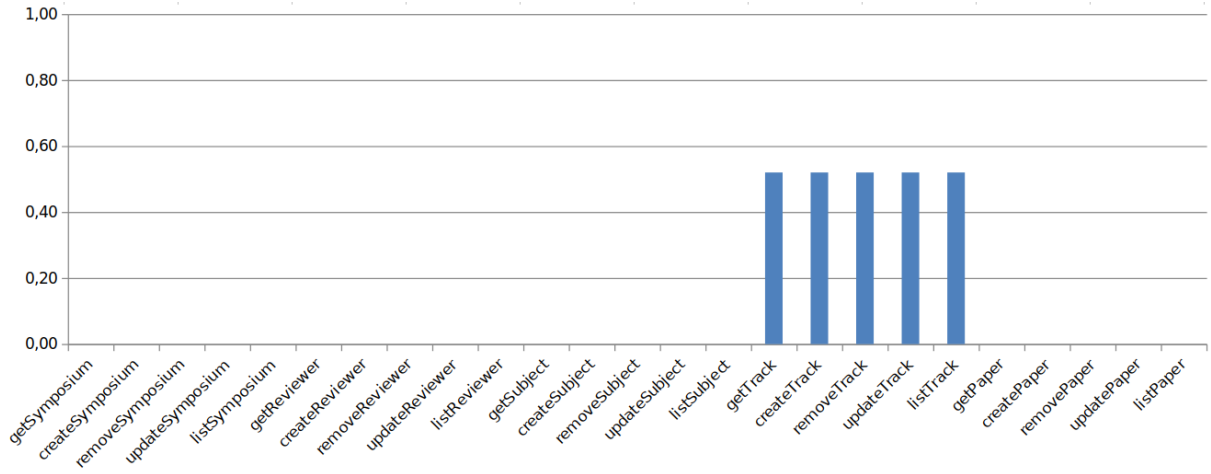

FIGURE 8. Results of executing the extended *Dieng-Hug* algorithm on the *Exclude Track* business process task with a greater weight for objects (40%-60%) and the function weights as RMN = 0.8, RCS = 1 and RMNIS = 0.8.

True Positives divided by the sum of True Positives and False Negatives). False Negatives are highlighted in yellow color in Figure 9. In case of *Recall* its value will be:

$$Recall = 24/24 + 1 = 0.96\ (i.e., 96\%)$$

In our context, a *Precision* score of 1.0 means that every case named as Real Positive does indeed is whereas a *Recall* of 1.0 means that every case of Real Positive was named as such.

Then, in order to deeply evaluate the results obtained by the algorithm, the following subsection presents a comparison between these results and those obtained by the software engineering experts.

A. EXPERTS RESULTS VS SEMANTIC SERVICE DISCOVERY ALGORITHM RESULTS

In this subsection, we shall present the comparison of the results of the semantic service discovery algorithm that we have described with those obtained by a group of experts. Concretely, this group of experts was composed of about eighteen people who belong to our research group or are analysts of a technological park in the environment of the University. Obviously, all of them (mainly researches) have knowledge about the case study domain, that is, related to a conference review process.

To carry out the validation process by the group of experts, the following procedure has been followed:

- The procedure was explained to every expert.
- The dossier with instructions and data about tasks and services was given to every expert. Specifically, the dossier comprised the BPMN models of the case study and a list of the available underlying services.
- Every expert completed the task. It should be noted that they complained about the excessive workload involved in the second execution where the methods signatures were randomized.
- After gathering the alignment report from every expert, a comparative process was applied that met the next

TASKS	(1) Wss manual alignment (Experts)	(2) WA-Wss auto alignment >80% (Sosa-Sánchez2014)	(3) WA-Wss auto alignment >50% <=80% (Sosa-Sánchez2014)	(4) DH-Wss auto alignment >80% {0.8, 1, 0.8}	(5) DH-Wss auto alignment >50% <=80% {0.8, 1, 0.8}
cancelConference			removeSymposium	removeSymposium	
cancelPaper		removePaper		removePaper	
createConference	createSymposium		createSymposium	createSymposium	
createPaper	createPaper	createPaper		createPaper	
createReviewer	createReviewer	createReviewer		createReviewer	
createSubject	createSubject	createSubject		createSubject	
createTrack	createTrack	createTrack		createTrack	
deleteConference	removeSymposium		removeSymposium	removeSymposium	
deletePaper	removePaper	removePaper		removePaper	
deleteReviewer	removeReviewer	removeReviewer		removeReviewer	
deleteSubject	removeSubject	removeSubject		removeSubject	
excludeTrack	removeTrack	removeTrack			removeTrack
modifyTrack	updateTrack	upateTrack		updateTrack	
notifyEmail					
obtain-allConference	listSymposium		listSymposium	listSymposium	
obtain-allPaper	listPaper	listPaper		listPaper	
obtain-allReviewer	listReviewer	listReviewer		listReviewer	
obtain-allSubject	listSubject	listSubject		listSubject	
obtain-allTrack	listTrack	listTrack		listTrack	
obtainConference	getSymposium		getSymposium	getSymposium	
obtainConference					
obtainPaper	getPaper	getPaper		getPaper	
obtainReviewer	getReviewer	getReviewer		getReviewer	
obtainReviewer					
obtainSubject	getSubject	getSubject		getSubject	
obtainSubject					
obtainTrack	getTrack	getTrack		getTrack	
obtainTrack					
publishFacebook					
publishTwitter					
updateConference	updateSymposium		updateSymposium	updateSymposium	
updatePaper	updatePaper	updatePaper		updatePaper	
updateReviewer	updateReviewer	updateReviewer		updateReviewer	
updateSubject	updateSubject	updateSubject		updateSubject	

FIGURE 9. Alignment between business process tasks and services: (1) software engineering experts' alignment; (2) [14] service discovery algorithm with an 80% threshold; (3) [14] service discovery algorithm with a 50% threshold (additional recommended business process tasks); (4) our approach's alignment with an 80% threshold; (5) our approach's alignment with a 50% threshold (additional recommended business process tasks).

restriction: an alignment between a BP task and a service would only be right when, at least, 80% of experts had previously identified such alignment in their reports.

The software engineering experts carried out a semantic service discovery algorithm using as running example the CRS described above. The target of the experts was to identify the business process tasks which, in their opinion and based on their knowledge, could be aligned with the available services (taking input/output parameters into account). They took as input the explained dossier with the systems' BPMN models and the Simple-SoaML model. Specifically, 25 services defined by WSDL in the service layer and 34 business processes tasks (identified from the 25 BPMNs) were used in the validation.[10]

Additionally, a comparison between: i) the alignment of the experts; ii) the semantic service discovery presented in this paper and, iii) the semantic service discovery presented in [14] is carried out.

Figure 10 shows the results of this comparison. The proportion of business process tasks aligned by the experts with the services available was 73.52% (25/34). The unaligned tasks mainly correspond to new tasks in the BPMN diagrams which could only be aligned with new service implementations or external services, for example, social network services (an example is the *Publish Facebook* task). The alignment proportion using our semantic service discovery algorithm was (76.47%) when considering only tasks aligned with services with values $>$ 80% of the maximum similarity value (which is 0.86 in this case, so we consider values $>$ 0.69) (defined in section IV-D).

The alignment proportion obtained by our algorithm is greater than the obtained by the experts. The reason can be seen in Figure 9 (columns (1) and (4)). As one observes in this figure, all the business process tasks aligned with services were also aligned by the software engineering experts with the same services. However, one finds different examples of alignment in this figure depending on the values obtained: (i) alignments with values $>$ 80% of the maximum similarity value (e.g., the *Cancel Conference* and *Cancel Paper* tasks are aligned with the services *remove Symposium* and *remove Paper* respectively. As previously explained, there are two "false positives" examples); (ii) alignments with values $>$ 50% *and* $\leq$ 80% of the maximum similarity value (e.g., the *Exclude Track* task. As previously explained, it is a "false

[10]https://sites.google.com/site/migrasoa/crs-case-study/crs-validation

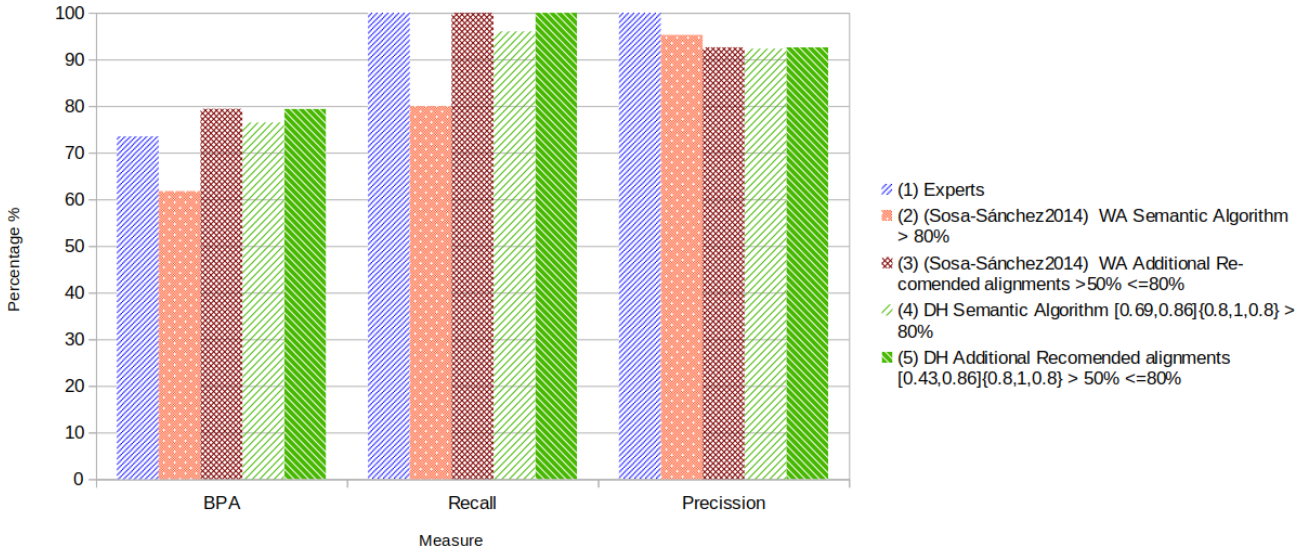

FIGURE 10. Comparison of business process tasks alignments (BPA, Recall, Precision): (1) business process tasks aligned by software engineering experts; (2) business process tasks aligned by [14] approach with an 80% threshold in the semantic algorithm; (3) business process tasks aligned by our approach with a 50% threshold in the semantic algorithm [14] (additional recommended business process tasks). (4) business process tasks aligned by our approach with an 80% threshold in the semantic algorithm; and (5) business process tasks aligned by our approach with a 50% threshold in the semantic algorithm (additional recommended business process tasks).

negative'' example for our algorithm but not for the experts); and (iii) alignments with value 0 (highlighted in red, e.g., the *Publish Facebook* task), which are tasks that were not aligned with any service because they normally discover some new functionality for the firm. There are other tasks (such as the second *Obtain Reviewer* task in Figure 9) that are not aligned with any service because their parameters do not match.

Taking into account these data, as expected, the values obtained for *Precision* and *Recall* by experts are 1. Specifically, the *Precision* value is calculated as:

$$Precision = 25/25 + 0 = 1 \ (i.e., 100\%)$$

and the value for *Recall* is calculated:

$$Recall = 25/25 + 0 = 1 \ (i.e., 100\%)$$

Finally, in Figure 10, one observes that, in adding the business process tasks aligned with services using our approach but considering similarity values between 50% and 80%, the alignment proportion is 79.41%, greater than that of the software engineering experts given that the alignment *removeTrack* is now taken into account as a true positives (as shown in column (5) of the figure).

In addition, considering similarity values between 50% and 80% the obtained value for *Recall* is 1. Again (Figure 9 column (5)), the business process tasks aligned with services are linked with the same services chosen by the experts (but including the two ''false positives'' *Cancel Conference* and *Cancel Paper*, that is why the precision obtained by the experts is 1 and by our algorithm is 92.59). One would thus present to users the business process tasks aligned with similarity values between 50% and 80% as additional recommended alignment results (obtaining a BPA of 79.41% in the present case), while business process tasks aligned with similarity values above 0.69 (threshold defined in section IV-D) would be presented to users as highly recommended alignments between tasks and services.

In order to check how suitable the semantic dictionary is, the same case study alignment has been executed using as dictionary the WordNet API [47]. The main relation between words in WordNet is synonymy. These synonyms are grouped into unordered sets (synsets). Besides this main relation, WordNet also includes relations such as hyponymy, meronymy and hyperonymy as we do. Therefore, our dictionary and WordNet are using the same relations between the terms. Table 6 summarizes the results obtained by the semantic algorithm using the ad-hoc semantic dictionary and WordNet dictionary. The results obtained using WordNet are significantly worse than using a semantic dictionary defined ad-hoc for the domain (probably because the terms included in the synsets are related to broader domains than those used in our ad-hoc dictionary). In the results obtained using WordNet there are not many cases of ''false positives'' but there are enough cases of ''false negatives''. This makes the precision have a high value (93.75%) but nevertheless, the recall has a very low value (62.50%). That means the generic semantic dictionaries are important for semantic algorithm because can be used initially, however using an ad-doc dictionary improves the alignment process results. Therefore, in spite of defining a semantic dictionary has a cost, it is important to obtain better results.

In summary, the semantic service discovery algorithm used in the BP alignment process that has been presented allows

TABLE 6. Comparative results using (1) semantic algorithms presented with ad-hoc semantic dictionary and (2) semantic algorithm with WordNet dictionary (weights: *RMN* = 0.8, *RCS* = 1, *RMNIS* = 0.8).

	(1) Semantic ad-hoc dictionary	(2) WordNet dictionary	Difference
BPA	76.47%	47.06%	29.41%
Recall	96.00%	62.50%	33.5%
Precision	92.31%	93.75%	-1.44%

suitable semantic relations between business process tasks and services to be identified. In this sense, the semantic dictionary is the key to obtaining excellent results, close or even equal to those provided by experts.

VI. CODE GENERATION AND Service ORCHESTRATION USING EXECUTABLE BPMN MODELS

Executable business process definition, where business processes are aligned with the technological infrastructure, is currently a challenge for the business processes research community [5]. To align business processes with the underlying technology, specific knowledge about the services including services path, services description and communication protocols are required. In this sense, MigraSOA defines model transformations to weave the business processes defined by BPMN models, extending them with information on how to invoke each specific web service (services aligned at MigraSOA phase *(c)*). This BPMN models extension has been previously performed using the same case study presented in the current work and can be consulted in [15]. Note that, invoking SOAP web services from the BPMN model extension is carried out using synchronous communication by means of their well-defined interfaces (WSDL descriptions).

In addition, the Extended BPMN model and Java artefacts are obtained from the BP-WS model (MigraSOA phase *(d)*). The Extended BPMN metamodel is based on a concrete specification of each BPMS. Consequently, MigraSOA phase (*d*) should be based on the specific platform because our final goal is to obtain executable business processes. As a consequence, the BPMN extensions implemented by the BPMS are needed in this phase. In this regard, Apache Activiti [70] is used as a target BPMS. Thus, the automatic model transformations carried out allow: i) adding into the .BPMN files the required code to invoke the services aligned in MigraSOA phase (*c*) and ii) generating the Java classes which wrap each service invocation from the specific BPMS. Therefore, each business process task should be located at the BPMN models (.BPMN file) and then the information about the specific service to be invoked should be attached. It includes the URL where the WSDL web service descriptor is located, the operation name that will be invoked and a mapping of input and output parameters.

Besides, an *org.activiti.engine.delegate. JavaDelegate* interface should be implemented that will be used by Activiti engine to delegate the execution of business processes task into a specific Java class. In this case, this interface implementation has been developed using Apache CXF [80], an JAX-WS [11] implementation, which facilitates building method calls on web services. Note that implementation of *org.activiti.engine.delegate.JavaDelegate* interface includes an *execute* method with a *DelegateExecution execution* input parameter. This parameter includes the WSDL descriptor URL, the operation name, and a list of input and output parameters. As a consequence, this implementation allows: i) to obtain an WS dynamic instance using *JaxWsDynamicClientFactory* class [80]; ii) to build an WS client from the WSDL descriptor to be invoked; iii) to invoke the WS using as arguments the operation name and the list of input parameters; and iv) finally, the results returned are stored in a result output list.

With the aim to link each BP aligned task with the generated class that implements the service invocation, the following arguments are labeled at the task: i) *TaskType* where the technology used to implement the task is defined: Java class; ii) *Class Name* defines the class generated previously that implements the *org.activiti.engine.delegate.JavaDelegate* interface; iii) configurable parameters, including the following: *wsdl, operation, alignedParamsIn, alignedParamsOut* and *notAlignedParamsOut*. The parameters *alignedParamIn, alignedParamOut* and *notAlignedParamOut* are obtained from the MigraSOA phase (*c*), where the BP Tasks have been aligned with the services which implement them. All this information allows to extends the BPMN models defined at .BPMN files which are coded in XML. In Code- listing 1 an XML excerpt with this information could be observed.

Finally, as a result of this MigraSOA phase *(d)*, the BP defined by the company have been aligned with the WS. Specifically, the BPMN models have been extended to include, for each Task previously aligned, the WS invocation. So, the BPMN models could be executable by the BPMNS, in our case Apache Activiti. Note that, in the BPMN models could be found not aligned tasks, which should be manually reviewed because they require an ad-hoc web service implementation.

VII. IMPLICATIONS

Next we are going to describe several implications related to the presented approach: semantic algorithm issues, how to manage business process collections, algorithm optimization to reduce the processing time, why the results obtained are similar to those of the experts, and how the approach applicability can be extended beyond a migration process.

Firstly, the semantic service discovery algorithm core, and its results improved significantly if the input semantic

[11] https://jcp.org/en/jsr/detail?id=224

Algorithm 1 XML Code in .Bpmn File Related to the BP Tasks Aligned With a Service

```
<serviceTask id="serviceTask1" name="
    ServiceTaskName"
   activiti:class="classes.WSDelegate">
   <extensionElements>
        <activiti:field name="wsdl">
           <activiti:expression>
             URL where the WSDL is
                 available
           </activiti:expression>
        </activiti:field>
        <activiti:field name="operation"
            >
            <activiti:expression>
               operationName
            </activiti:expression>
        </activiti:field>
        <activiti:field name="
            alignedParamsIn">
            <activiti:expression>
               ${paramIn1}
            </activiti:expression>
        </activiti:field>
        <activiti:field name="
            alignedParamsOut">
            <activiti:expression>
               ${paramOut1}
            </activiti:expression>
        </activiti:field>
        <activiti:field name="
            notAlignedParamsOut">
           <activiti:expression>
               ${notAlignedParamOut1}
           </activiti:expression>
        </activiti:field>
   </extensionElements>
</serviceTask>
```

dictionary is close to the business processes domain. In this sense, the semantic relationship between words should be described with accuracy. To this purpose, the semantic dictionary should be defined by domain experts.

Secondly, the BP alignment process allows automatically aligning business process tasks collections with large service layer. As a consequence, the manual tasks carried out by a software architect are delegated to the last step in the process, where aligning should be supervised. This means that traditional, tedious and error prone alignment tasks are avoided. Moreover, the semantic service discovery algorithm could be executed as many times as necessary, specially when the business process collection shall be updated. As a consequence, many software quality attributes, such as adaptability and extensibility would improve. Adaptability would improve because the whole system could be adapted from the business process models collection, while extensibility would improve taking into account that an extension of the business processes or an extension of the service layer only requires a new service discovery execution to update the whole system.

Thirdly, the semantic algorithm runs in seconds or a few minutes, according to the business process models collections size. The semantic algorithm has been executed in a laptop with an Intel i7 processor, 16 GB RAM and Ubuntu 18.04 as the Operating System. As a consequence, in our opinion the time required to carry out the semantic service discovery algorithm execution is not relevant when compared with the time required to carry out the same service discovery process manually.

Fourthly, the alignment between business processes and the underlying information technology is a well-known challenge [5]. In this sense, the results obtained using the automatic service discovery proposed are similar to those obtained manually by the experts. As a consequence, software engineering approaches based on semantic algorithms like this could decrease the time, effort and cost to carry out this kind of semantic service discovery algorithm.

Finally, the semantic service discovery algorithm presented could be applied beyond the software migration context. It could be broadly applied to align collections of business processes models with an available underlying service layer. This is an important point because the underlying service layer could be built from the legacy systems, developed ad-hoc or based on third-party services (e.g. Software as a Service – SaaS). In this regard, these third-party services or own web services could be incorporated ad-hoc into the MigraSOA process. Concretely, they should be added into the Simple-SoaML model, which is the MigraSOA phase (*c*) input.

VIII. LIMITATIONS

Next, we describe several limitations and challenges of this work, including requirements about the business processes notation, different web services protocols and labeling business process tasks.

Business processes should be defined at an operational level, that is, they should be defined at a very detailed level. However, this is not usual in small or medium firms, which require a specific effort to carry out this task. On the contrary, big firms have usually defined their business processes in detail, which facilitates their management. Perhaps, this is the main limitation of the presented approach because this is the initial point to carry out the alignment process.

Furthermore, the information related with web services (defined using WSDL) should be stored in Simple-SoaML model. Currently, this process is carried out during the MigraSOA phase (*b*), using as a service target those that have been generated from the LWA. However, users could include in Simple-SoaML models information from external services which could be used during the alignment process. Thus, the results obtained could be improved if current business process

tasks not aligned in a previous execution could be aligned with external services. Besides, considering that the process could be used in other contexts beyond a migration process, a text transformation from WSDL to Simple-SoaML model is needed in order to automatically populate the Simple-SoaML models. Currently, this is a limitation if we want to use this semantic service discovery algorithm in other contexts. Nevertheless, this process could be manually carried out improving the alignment results because these new services would be available to be aligned.

Semantic algorithm works successfully when the business processes labels are defined following the writing rules identified in section IV-A. Currently, we are working in an additional version of the semantic algorithm which recognizes word senses as [49] does.

Finally, with the aim to execute our proposal, the BPMN models should be defined using the Activiti BPMN editor which facilitates that BPMN models be executed on the Activiti BPM Platform. That is, we are linked to a concrete BPM Platform mainly because there is not interoperability among BPMN tools.

IX. CONCLUSIONS AND FURTHER WORK

The alignment process between business processes and the underlying services is an open research line to both the definition a new SOA and the migration of a legacy system into SOA [5]. In this sense, the MigraSOA approach has been defined to tackle the complexity of migrating projects from LWAs into SOA. It is based on an intensive use of model-driven techniques including metamodeling, and text-to-model, model-to-model, and model-to-text transformations. It allows an interoperable service layer to be obtained that is based on Web Services from the LWAs and which could be reused to implement the firm's business processes. To that end, a semantic service discovery algorithm has been defined to facilitate the alignment with the generated service layer of business processes defined in BPMN at different abstraction levels. This service discovery algorithm uses a semantic algorithm based on the *Dieng-Hug* algorithm. The aim is to identify the services that could potentially be used by the business processes defined by the firm. The results provided evidence the benefits obtained by the approach according to the high percentage of alignment obtained between tasks and services. Additionally, the tasks that were not aligned help us to identify new functionality for the firm (for example, tasks related to social networks).

Moreover, the alignment results given by the adaptation of the *Dieng-Hug* semantic algorithm were similar to those provided by domain experts. In this sense, the present automatic alignment process reduces migration costs and the propensity to error. It offers, on the one hand, an appropriate solution for developers and, on the other hand, a software engineering approach based on standards such as BPMN, Metamodeling, Web Services, and Business Process Management Systems (BPMS).

Our future work plans include the use of WordNet [47] or BabelNet [50] during the alignment process as a complement to the main semantic dictionary and carrying out a comparative with other semantic algorithms related to our approach. In this sense, these target semantic algorithms should be adapted to take into account how the business processes and the underlying services are usually defined.

Besides, in order to decrease the cost of building the semantic dictionary we will offer users an initial version of the semantic dictionary based on WordNet or BabelNet. Additional alignment rules will be defined to achieve new cases of alignment, for instance, when an alignment relies on other previous alignment. Furthermore, we are evaluating the use of BPEL [81] as an orchestration code which facilitates the adoption of this approach from BPMN models not defined using a concrete tool like Apache Activiti. In this sense, phase (*d*) should be redesigned to implement a model to text transformation, from BPMN models, BP-WS model and Simple-SoaML model to BPEL code. Then, we might use the Apache Orchestration Director Engine (Apache ODE) as a target platform to execute the code generated for these business processes. Note that, for instance, the Intalio BPMS uses BPEL as the engine for execution of business process models.

ACKNOWLEDGMENT

This work was funded by the Ministry of Science and Innovation (MCI), for the State Research Agency (AEI) - Project RTI2018- 098652-B-I00; the Government of Extremadura, Council for Economy, Science and Digital Agenda under the grants GR18112 and IB18034, by the European Regional Development Fund (ERDF) and by the 4IE+ project (0499-4IE-PLUS-4-E) funded by the Interreg V-A Spain-Portugal.

REFERENCES

[1] R. K. L. Ko, S. S. G. Lee, and E. Wah Lee, "Business process management (BPM) standards: A survey," *Bus. Process Manage. J.*, vol. 15, no. 5, pp. 744–791, Sep. 2009.

[2] R. Rodríguez-Echeverría, J. M. Conejero, M. Linaje, J. C. Preciado, and F. Sánchez-Figueroa, "Re-engineering legacy Web applications into rich Internet applications," in *Proc. 10th Int. Conf. Web Eng.* (Lecture Notes in Computer Science). Springer, 2010, pp. 189–203.

[3] J. M. Conejero, R. Rodríguez-Echeverría, F. Sánchez-Figueroa, M. Linaje, J. C. Preciado, and P. J. Clemente, "Re-engineering legacy Web applications into RIAs by aligning modernization requirements, patterns and RIA features," *J. Syst. Softw.*, vol. 86, no. 12, pp. 2981–2994, Dec. 2013, doi: 10.1016/j.jss.2013.04.053.

[4] (2011). *SOA-Manifesto*. [Online]. Available: http://www.soa-manifesto.org

[5] A. Ullah and R. Lai, "A systematic review of business and information technology alignment," *ACM Trans. Manage. Inf. Syst.*, vol. 4, no. 1, pp. 1–30, Apr. 2013.

[6] M. Weske, *Business Process Management-Concepts, Languages Architectures*, 2nd ed. Springer, 2012, doi: 10.1007/978-3-642-28616-2.
[7] L. O'Brien, L. Bass, and P. F. Merson, "Quality attributes and service-oriented architectures," Softw. Eng. Inst., Tech. Rep., 2005. [Online]. Available: http://repository.cmu.edu/sei/449/
[8] S. R. Tilley, D. B. Smith, and H. A. Müller, "Migrating to SOA: Approaches, challenges, and lessons learned," in *Proc. Conf. Center Adv. Stud. Collaborative Res.*, J. W. Ng, C. Couturier, H. A. Müller, and A. G. Ryman, Eds. Toronto, ON, Canada: ACM, 2010, pp. 371–373, doi: 10.1145/1923947.1923998.
[9] A. Umar and A. Zordan, "Reengineering for service oriented architectures: A strategic decision model for integration versus migration," *J. Syst. Softw.*, vol. 82, no. 3, pp. 448–462, Mar. 2009.
[10] P. Bhallamudi and S. Tilley, "SOA migration case studies and lessons learned," in *Proc. IEEE Int. Syst. Conf.*, Apr. 2011, pp. 123–128.
[11] M. Galinium and N. Shahbaz, "Case studies: Business and technical perspectives in migration of legacy systems to service oriented architecture," 2014, *arXiv:1412.7959*. [Online]. Available: http://arxiv.org/abs/1412.7959
[12] R. Khadka, A. Saeidi, S. Jansen, J. Hage, and G. P. Haas, "Migrating a large scale legacy application to SOA: Challenges and lessons learned," in *Proc. 20th Working Conf. Reverse Eng. (WCRE)*, R. Lämmel, R. Oliveto, and R. Robbes, Eds. Koblenz, Germany: IEEE Computer Society, Oct. 2013, pp. 425–432 [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6656028 and [Online]. Available: http://www.computer.org/csdl/proceedings/wcre/2013/9999/00/index.html
[13] E. Sosa, P. J. Clemente, J. M. Conejero, and R. Rodriguez-Echeverria, "A model-driven process to modernize legacy Web applications based on service oriented architectures," in *Proc. 15th IEEE Int. Symp. Web Syst. Evol. (WSE)*, Sep. 2013, pp. 61–70.
[14] E. Sosa-Sanchez, P. J. Clemente, M. Sanchez-Cabrera, J. M. Conejero, R. Rodriguez-Echeverria, and F. Sanchez-Figueroa, "Service discovery using a semantic algorithm in a SOA modernization process from legacy Web applications," in *Proc. IEEE World Congr. Services*, Jun. 2014, pp. 470–477, doi: 10.1109/SERVICES.2014.90.
[15] E. Sosa Sanchez, P. J. Clemente, A. E. Prieto, J. M. Conejero, and R. Rodriguez Echeverria, "MigraSOA: Migration of legacy Web applications to service oriented architectures (SOA)," *IEEE Latin Amer. Trans.*, vol. 15, no. 7, pp. 1306–1311, 2017.
[16] (Jan. 2011). *BPMN2.0*. [Online]. Available: http://www.omg.org/spec/bpmn/2.0/
[17] R. Anthony, *Planning and Control Systems: A Framework for Analysis*. Boston, MA, USA: Harvard University Graduate School of Business Administration, Jan. 1965.
[18] A. Walsh, *UDDI, SOAP, and WSDL: The Web Services Specification Reference Book*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.
[19] C. Armstrong, "Modeling Web services with UML," in *Proc. OMG Web Services Workshop*, 2002, p. 4.
[20] S. Anam, Y. S. Kim, B. H. Kang, and Q. Liu, "Review of ontology matching approaches and challenges," *Int. J. Comput. Sci. Netw. Solutions*, vol. 3, no. 3, pp. 1–27, 2015.
[21] N. F. Noy, A. Chugh, W. Liu, and M. A. Musen, "A framework for ontology evolution in collaborative environments," in *Proc. Int. Semantic Web Conf.* Springer, 2006, pp. 544–558.
[22] A. Isaac, S. Wang, C. Zinn, H. Matthezing, L. van der Meij, and S. Schlobach, "Evaluating thesaurus alignments for semantic interoperability in the library domain," *IEEE Intell. Syst.*, vol. 24, no. 2, pp. 76–86, Mar. 2009.
[23] P. P. Talukdar, Z. G. Ives, and F. Pereira, "Automatically incorporating new sources in keyword search-based data integration," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, 2010, pp. 387–398.
[24] S. Dessloch, M. A. Hernandez, R. Wisnesky, A. Radwan, and J. Zhou, "Orchid: Integrating schema mapping and ETL," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Apr. 2008, pp. 1307–1316.
[25] M. Atencia, J. Euzenat, G. Pirro, and M.-C. Rousset, "Alignment-based trust for resource finding in semantic P2P networks," in *Proc. Int. Semantic Web Conf.* Springer, 2011, pp. 51–66.
[26] L. Vaccari, P. Shvaiko, and M. Marchese, "A geo-service semantic integration in spatial data infrastructures," *Int. J. Spatial Data Infrastruct. Res.*, vol. 4, no. 4, pp. 24–51, 2009.
[27] J. Z. Wang, F. Ali, and R. Appaneravanda, "A Web service for efficient ontology comparison," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2005, pp. 843–844.
[28] P. Shvaiko and J. Euzenat, "Ontology matching: State of the art and future challenges," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 158–176, Jan. 2013.
[29] A. Algergawy, D. Faria, A. Ferrara, I. Fundulaki, I. Harrow, S. Hertling, E. Jimenez-Ruiz, N. Karam, A. Khiat, and P. Lambrix, "Results of the ontology alignment evaluation initiative 2019," in *Proc. CEUR Workshop Proc.*, vol. 2536, 2019, pp. 46–85.
[30] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string metrics for matching names and records," in *Proc. KDD Workshop Data Cleaning Object Consolidation*, vol. 3, 2003, pp. 73–78.
[31] M. Kumari, A. Jain, and A. Bhatia, "Synonyms based term weighting scheme: An extension to TF. IDF," *Procedia Comput. Sci.*, vol. 89, pp. 555–561, 2016.
[32] R. Dieng and S. Hug, "Comparison of personal ontologies represented through conceptual graphs," in *Proc. 13th ECAI*, Brighton, U.K., 1998, pp. 341–345.
[33] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," 1995, *arXiv:cmp-lg/9511007*. [Online]. Available: https://arxiv.org/abs/cmp-lg/9511007
[34] Y. Li, Z. A. Bandar, and D. McLean, "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 871–882, Jul. 2003.
[35] D. Lin, "An information-theoretic definition of similarity," in *Proc. ICML*, vol. 98, 1998, pp. 296–304.
[36] A. P. de Souza and R. J. Rabelo, "A model for dynamic services discovery over largely distributed providers based on QoS and business processes contexts," in *Proc. IEEE World Congr. Services*, Jul. 2011, pp. 347–354. [Online]. Available: http://dblp.uni-trier.de/db/conf/services/services2011.html#Perin-SouzaR11
[37] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for Web services," in *Proc. 13th Int. Conf. Very Large Data Bases*, vol. 30, 2004, pp. 372–383.
[38] O. Hatzi, G. Batistatos, M. Nikolaidou, and D. Anagnostopoulos, "A specialized search engine for Web service discovery," in *Proc. IEEE 19th Int. Conf. Web Services*, Honolulu, HI, USA, Jun. 2012, pp. 448–455.
[39] M. Klusch, B. Fries, and K. Sycara, "Automated semantic Web service discovery with OWLS-MX," in *Proc. 5th Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, Hakodate, Japan, 2006, pp. 915–922.
[40] G. C. Hobold and F. Siqueira, "Discovery of semantic Web services compositions based on SAWSDL annotations," in *Proc. IEEE 19th Int. Conf. Web Services*, C. A. Goble, P. P. Chen, and J. Zhang, Eds. Honolulu, HI, USA: IEEE, Jun. 2012, pp. 280–287, doi: 10.1109/ICWS.2012.97.
[41] C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun, "A probabilistic approach for Web service discovery," in *Proc. IEEE Int. Conf. Services Comput.*, Washington, DC, USA, Jun. 2013, pp. 49–56.
[42] J. Wang, P. Gao, Y. Ma, K. He, and P. C. K. Hung, "A Web service discovery approach based on common topic groups extraction," *IEEE Access*, vol. 5, pp. 10193–10208, 2017.
[43] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[44] D. Blei and J. Lafferty, "Correlated topic models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 18, 2006, p. 147.
[45] X. Cheng, X. Yan, Y. Lan, and J. Guo, "BTM: Topic modeling over short texts," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2928–2941, Dec. 2014.
[46] J. Sangers, F. Frasincar, F. Hogenboom, and V. Chepegin, "Semantic Web service discovery using natural language processing techniques," *Expert Syst. Appl.*, vol. 40, no. 11, pp. 4660–4671, Sep. 2013.
[47] P. U. WordNet. (2014). *WordNet Project*. [Online]. Available: http://wordnet.princeton.edu
[48] F. Chen, C. Lu, H. Wu, and M. Li, "A semantic similarity measure integrating multiple conceptual relationships for Web service discovery," *Expert Syst. Appl.*, vol. 67, pp. 19–31, Jan. 2017.
[49] H. Leopold, F. Pittke, and J. Mendling, "Automatic service derivation from business process model repositories via semantic technology," *J. Syst. Softw.*, vol. 108, pp. 134–147, Oct. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215001235
[50] R. Navigli and S. P. Ponzetto, "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network," *Artif. Intell.*, vol. 193, pp. 217–250, Dec. 2012, doi: 10.1016/j.artint.2012.07.001.
[51] O. Tibermacine, C. Tibermacine, and F. Cherif, "A process to identify relevant substitutes for healing failed WS-* orchestrations," *J. Syst. Softw.*, vol. 104, pp. 1–16, Jun. 2015, doi: 10.1016/j.jss.2015.02.028.

[52] O. C. Tibermacine and C. Foudil, "A practical approach to the measurement of similarity between wsdl-based Web services," in *Proc. 6ème Conf. Francophone Sur Les Architectures Logicielles (CAL)* (RNTI), vol. L-7, E. Exposito and M. Zouari, Eds. Montpellier, France: Hermann-Editions, 2012, pp. 3–18. [Online]. Available: http://editions-rnti.fr/?inprocid=1002037
[53] P. Pietsch, K. Müller, and B. Rumpe, "Model matching challenge: Benchmarks for ecore and BPMN diagrams," *Softwaretechnik-Trends*, vol. 33, no. 2, pp. 95–100, May 2013.
[54] M. Alanen and I. Porres, "Difference and union of models," in *The Unified Modeling Language. Modeling Languages and Applications* (Lecture Notes in Computer Science), vol. 2863, P. Stevens, J. Whittle, and G. Booch, Eds. Berlin, Germany: Springer, 2003, pp. 2–17.
[55] D. S. Kolovos, D. Di Ruscio, A. Pierantonio, and R. F. Paige, "Different models for model matching: An analysis of approaches to support model differencing," in *Proc. ICSE Workshop Comparison Versioning Softw. Models*, Washington, DC, USA, May 2009, pp. 1–6.
[56] K. Shahzad, I. Pervaz, and A. Nawab, "WordNet based semantic similarity measures for process model matching," in *Proc. BIR Workshops*, 2018, pp. 33–44.
[57] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," 1997. *arXiv:cmp-lg/9709008*. [Online]. Available: https://arxiv.org/abs/cmp-lg/9709008
[58] C. Leacock and M. Chodorow, "Combining local context and WordNet similarity for word sense identification," *WordNet, Electron. Lexical Database*, vol. 49, no. 2, pp. 265–283, 1998.
[59] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proc. 32nd Annu. meeting Assoc. Comput. Linguistics*, 1994, pp. 133–138.
[60] C. Corley and R. Mihalcea, "Measuring the semantic similarity of texts," in *Proc. ACL Workshop Empirical Modeling Semantic Equivalence Entailment (EMSEE)*, 2005, pp. 13–18.
[61] V. Rus and M. Lintean, "A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics," in *Proc. 7th Workshop Building Educ. Appl. (NLP)*, Jun. 2012, pp. 157–162.
[62] M. Lintean and V. Rus, "An optimal quadratic approach to monolingual paraphrase alignment," in *Proc. 20th Nordic Conf. Comput. Linguistics (NODALIDA)*, 2015, pp. 127–134.
[63] C. Meilicke, H. Leopold, E. Kuss, H. Stuckenschmidt, and H. A. Reijers, "Overcoming individual process model matcher weaknesses using ensemble matching," *Decis. Support Syst.*, vol. 100, pp. 15–26, Aug. 2017.
[64] S. Maoz, J. O. Ringert, and B. Rumpe, "A manifesto for semantic model differencing," in *Proc. Int. Conf. Model Driven Eng. Lang. Syst.* (MODELS). Berlin, Germany: Springer-Verlag, 2011, pp. 194–203.
[65] Z. Xing and E. Stroulia, "UMLDiff: An algorithm for object-oriented design differencing," in *Proc. 20th IEEE/ACM Int. Conf. Automated Softw. Eng.*, New York, NY, USA, 2005, pp. 54–65.
[66] E. Sosa-Sanchez, P. J. Clemente, A. E. Prieto, and C. Ortiz-Caraballo, "Aligning business processes with the services layer using a semantic approach," *IEEE Access*, vol. 7, pp. 2904–2927, 2019.
[67] J. Snell, D. Tidwell, and P. Kulchenko, *Programming Web Services With SOAP: Building Distributed Applications*. Newton, MA, USA: O'Reilly Media, 2001.
[68] E. Newcomer and G. Lomow, *Understanding SOA With Web Services*. Reading, MA, USA: Addison-Wesley, 2005.
[69] W3C. (Feb. 2004). *Web Services Glossary*. [Online]. Available: https://www.w3.org/tr/2004/note-ws-gloss-20040211/#webservice
[70] A. Activiti. (2015). *Apache Activiti, a Business Process Management (BPM) Platform*. [Online]. Available: http://www.activiti.org
[71] J. Davis, *Open Source Soa*, 1st ed. Greenwich, CT, USA: Manning Publications, 2009.
[72] B. Portier. (2006). (Soa Terminology Overview, Part 1: Service, Architecture, Governance, And Business Terms). [Online]. Available: http://www-128.ibm. com/developerworks/webservices/library/ws-soa-term1/index.html
[73] J. Mendling, H. A. Reijers, and J. Recker, "Activity labeling in process modeling: Empirical insights and recommendations," *Inf. Syst.*, vol. 35, no. 4, pp. 467–482, Jun. 2010, doi: 10.1016/j.is.2009.03.009.
[74] E. Sosa, A. Lozano-Tello, and Á. E. Prieto, "Semantic comparison of ontologies based on WordNet," in *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst.*, 2008, pp. 899–904.
[75] J. F. Sowa, "Conceptual graphs for a data base interface," *IBM J. Res. Develop.*, vol. 20, no. 4, pp. 336–357, Jul. 1976, doi: 10.1147/rd.204.0336.
[76] V. W. Chu, R. K. Wong, W. Chen, I. Paik, and C.-H. Chi, "Service discovery based on objective and subjective measures," in *Proc. IEEE Int. Conf. Services Comput.*, Santa Clara, CA, USA, Jun. 2013, pp. 360–367, doi: 10.1109/scc.2013.33.
[77] OMG. (Jan. 2012). *OMG Object Constraint Language (OCL), Version 2.3.1*. Object Management Group. [Online]. Available: http://www.omg.org/spec/OCL/2.3.1/
[78] Obeo. (2012). *Acceleo Project*. [Online]. Available: http://www.acceleo.org
[79] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," in *Informedness, Markedness and Correlation. Journal of Machine Learning*. Bioinfo Publications, 2011.
[80] Apache CXF, "CXF user's guide," Apache, Tech. Rep., 2015. [Online]. Available: http://cxf.apache.org/docs/
[81] WS-BPEL2.0. (Apr. 2007). *Web Services Business Process Execution Language Version 2.0*. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

ENCARNA SOSA SÁNCHEZ received the B.Sc. degree in computer science from the University of Granada, in 1995, and the Ph.D. degree in computer science in 2018. She is currently pursuing the Ph.D. degree with the Computer Science Department, University of Extremadura, Spain. She is currently an Assistant Professor with the Computer Science Department, University of Extremadura. She has published several peer-reviewed articles in international journals, workshops, and conferences, and is involved in several research projects. Her research interests include service-oriented architectures, business process modeling, and model-driven development.

PEDRO J. CLEMENTE received the B.Sc. degree in computer science from the University of Extremadura, Spain, in 1998, and the Ph.D. degree in computer science in 2007. He is currently an Associate Professor with the Computer Science Department, University of Extremadura. He has published numerous peer-reviewed articles in international journals, workshops, and conferences. His research interests include component-based software development, aspect orientation, service-oriented architectures, business process modeling, and model-driven development. He is involved in several research projects. He has participated in many workshops and conferences as a speaker and a member of the program committees.

JOSÉ M. CONEJERO received the Ph.D. degree in computer science from the Universidad de Extremadura, in 2010. He is an Assistant Professor with the Universidad de Extremadura. He is the author of more than 20 articles of journals and conference proceedings and has also participated in different journals and conferences as a member of the program committee. His research areas include the aspect-oriented software development, requirements engineering, model-driven development, and ambient intelligence.

ÁLVARO E. PRIETO received the B.Sc. degree in computer science from the University of Extremadura, Spain, in 2000, and the Ph.D. degree in computer science in 2013. He is an Assistant Professor of computer languages and systems with the University of Extremadura. He is a member of the Quercus Software Engineering Group. His research interests include ontologies, workflows, and linked open data. He is currently involved in various research and development and innovation projects.