Journal of Software: Evolution and Process

JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS
*J. Softw. Evol. and Proc.* 2014; **26**:1128–1149
Published online 20 May 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.1652

# Ontology-based similarity applied to business process clustering

Ricardo Pérez-Castillo[1,*,†], Danilo Caivano[2] and Mario Piattini[1]

[1]*University of Castilla-La Mancha Paseo de la Universidad 4, 13071, Ciudad Real, Spain*
[2]*University of Bari, Department of Informatics Via E. Orabona, 4, 70126, Bari, Italy*

## ABSTRACT

Reverse engineering of business process enables business process to be discovered and retrieved from existing information systems, which embed many business rules that are not available anywhere else. These techniques are especially useful when business process models are unavailable, outdated, or misaligned because of uncontrolled maintenance. Reverse engineering techniques obtain well-designed business processes, but these are often retrieved with harmful quality faults as a consequence of the abstraction. Clustering techniques are then applied to reduce these quality faults and improve the understandability and modifiability of business process models. Regrettably, the most challenging concern is how to determine the similarity between two business activities to be clustered. Formal ontologies help to represent the essential concepts and constraints of a universe of discourse and determine the similarity in accordance with the given ontology. This paper shows how to compute and use the ontology-based similarity within a clustering algorithm whose aim is to improve the quality of business process models previously obtained from legacy information systems by reverse engineering. The principal contribution of this paper is the usage of an ontology-based similarity function and its application to 43 business process models retrieved from four real-life information systems. Copyright © 2014 John Wiley & Sons, Ltd.

Received 31 October 2013; Revised 7 January 2014; Accepted 10 March 2014

KEY WORDS: conceptual modeling; ontologies; clustering, classification, and association rules; business process; empirical studies

## 1. INTRODUCTION

Business processes are a valuable intangible asset for companies, allowing companies to manage their daily operations and carry out changes in business goals to maintain their competitiveness [1]. From a software engineering viewpoint, business process is also the starting point for obtaining the requirements of new development projects or modernization projects [2]. Unfortunately, business processes are sometimes unavailable or outdated, because of uncontrolled maintenance [3]. Reverse engineering of business process enables business process to be discovered and retrieved from existing information systems, which embed many business rules that are not available anywhere else [4]. Those techniques can obtain well-designed business processes by inspecting source code. Indeed, a common reverse engineering pattern that is often applied is 'a callable unit – a candidate business task'. Regrettably, these reverse engineering techniques lead to business process models with some quality faults or bad smells related to the technical nature of the input information systems (e.g., many fine-grained business tasks almost directly related to auxiliary callable units, retrieval of redundant and non-relevant business tasks, and so forth). Clustering of business activities of such business process models are recurrently applied to reduce these quality faults and

*Correspondence to: Ricardo Pérez-Castillo, University of Castilla-La Mancha, Paseo de la Universidad 4, 13071, Ciudad Real, Spain.
†E-mail: ricardo.pdelcastillo@uclm.es

improve the understandability and modifiability (see Figure 1). This paper proposes an agglomerative, hierarchical clustering algorithm that trusts in ontologies to group business activities according to their semantic similarity.

Over the last decade, there has been an increasing amount of interest in formal ontologies in the conceptual modeling field. Conceptual modeling is a key discipline in computer science whose principal objective is to identify, analyze, and describe the essential concepts and constraints of a universe of discourse supported through a modeling language that is based on a set of basic modeling concepts [5]. An effort that is targeted towards the usage of formal ontologies with the purpose of concept modeling is termed as a domain ontology. The storage and management of our knowledge of the world or of a specific domain is a very old endeavor [6]. The definition of semantics and the distinction of entities in a certain domain is a common challenge in which ontologies have normally been used to collect and manage these semantics in a formal way. The term *formal ontology* is used to refer to an ontology defined by axioms in a formal language with the goal to provide an unbiased view on reality.

Although formal ontologies help to depict semantics in specific domains, the practical usage and computation of ontology concepts in some techniques and methods is challenging and is not well-defined. In this sense, one of the most important challenges is the matching of an actual element from the universe of discourse as regards elements previously depicted in a given formal ontology. What is more, when the element is not found in the formal ontology, the way in which the most similar element is figured out is a recurrent problem. Semantic similarity states how taxonomically close two terms are, because they share some aspects of their meanings. Particularly, the assessment of ontology-based similarity is a key task within clustering algorithms [7]. Clustering algorithms, as applied within data and process mining methodologies, are devoted to building up a classification or partition into coherent clusters from unstructured data sets [8].

This paper shows how to compute and use the ontology-based similarity within a clustering algorithm in order to improve the quality of business process models, which depict an organization's operative workflow. The objective of the proposed technique is to restructure and reduce fine-grained elements of business process models that have been obtained beforehand via the reverse engineering of information systems (Figure 1). The main contribution of this paper is twofold: (i) it first provide a proposal to improve business process models recovered from legacy information systems using an ontology-based clustering technique to support the noise reduction and quality improvement of the recovered models; and (ii) the validation of such proposal by applying it to a set of real case studies with four ontologies defined by experts.

The remainder of this paper is organized as follows. Section 2 presents the background of the proposal presented in this paper by showing the relevance of a well-founded theoretical perspective. Section 3 presents the clustering technique applied to business process models, which rely on ontology-based similarity. Section 4 provides the experimental results obtained after conducting a

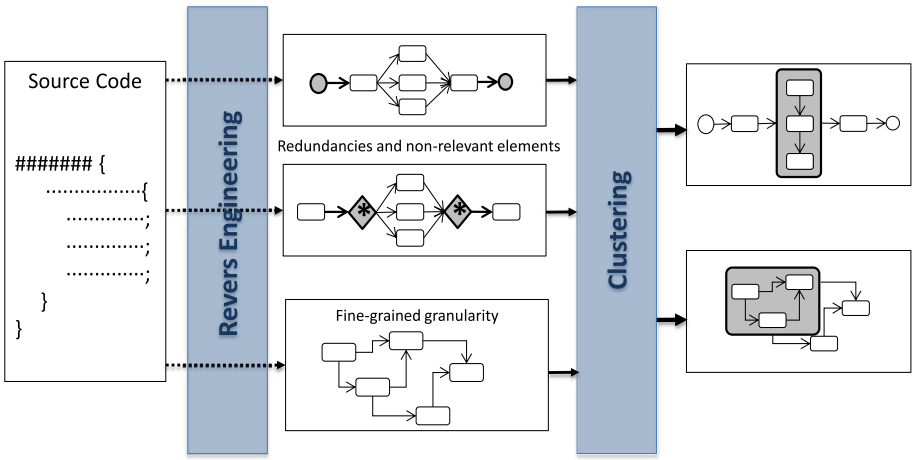


Figure 1. Business process clustering to fix quality faults derived from reverse engineering.



DOI: 10.1002/smr

case study with four real-life information systems. Finally, Section 5 discusses our conclusions and future work.

## 2. STATE-OF-THE-ART

This section shows the state-of-the-art related to the proposal in order to demonstrate the relevance of the underlying theory. Firstly, Section 2.1 presents a similarity assessment based on formal ontologies, in addition to some related works. Section 2.2 then introduces clustering techniques and their principal algorithms, some of which employ ontology-based similarity functions, whereas Section 2.3 motivates the problem of outdated and missing business process models, which are retrieved using reverse engineering and are then enhanced with clustering.

### *2.1. Ontology-based similarity*

According to *Smith et al.* [9], the term 'ontology' first appeared in the computer science field in 1967 in a data modeling work by S. H. Mealy, in which the author differentiated three realms in the field of data processing: (i) the real world itself; (ii) ideas about it that exist in the minds of men; and (iii) symbols on paper or some other storage medium. Studer *et al.* [10] define an ontology as a formal, explicit specification of a shared conceptualization. An ontology can therefore be considered as both a formal explicit description of concepts in a domain of discourse (called classes or concepts) and the relationships between these concepts [11], that is, it is a mechanism with which to represent the semantics of a specific domain. On the one hand, the term *formal ontology* is used to refer to an ontology defined by axioms in a formal language with the goal to provide an unbiased (domain- and application-independent) view on reality, which can help the modeler of domain-specific ontologies to avoid possibly erroneous ontological assumptions. Usually, 'formal' also refers to the fact that the ontology is machine-readable. On the other hand, the 'shared' term reflects the notion that an ontology captures consensual knowledge, that is, not a personal view of the target phenomenon of some particular individual, but one accepted by a group.

The similarity function uses ontologies provided by domain experts to determine the semantic similarity between two terms in relation to a common semantic reference. Ontologies are therefore designed to be used in applications that need to both process the content of information and reason about it, rather than simply presenting information to humans [7]. As a result, formal ontologies appeared in a way that was similar to that in which formal logic began. Although formal logic deals with formal logical structures (e.g., truth, validity, and consistency) independently of their veracity, formal ontologies deal with formal ontological structures (e.g., theory of parthood, types and instantiation, identity, dependence, and unity), that is, with formal aspects of entities, independent of their particular nature [5].

Several languages with which to represent and manage ontologies have been proposed with the objective of achieving machine-readable ontologies. Many authors consider the Web Ontology Language (OWL) [12], as proposed by the World Wide Web Consortium, to be the *de facto* standard. This proposal uses OWL to define and handle formal ontologies.

Most data mining processes, and clustering in particular, require the evaluation of data attributes to detect the degree of alikeness between records or individuals. Unlike numerical data, which can be directly and easily manipulated and compared by means of classical mathematical operators, the processing of qualitative data is a challenging task [7]. Words are labels that refer to concepts, which define their semantics. The objective of semantic similarity science [13] is to estimate the alikeness between words or concepts by discovering, evaluating, and exploiting their semantics. Because semantics is an inherently human feature, methods with which to automatically calculate semantic similarity rely on evidence retrieved from one or several manually constructed knowledge sources. The goal is to mimic human judgments of similarity by exploiting implicit or explicit semantic evidence [7].

Semantic similarity states how taxonomically close two terms are, because they share some aspects of their meanings. For example, human and dolphin are similar because they are mammals. The



DOI: 10.1002/smr

computation of semantic similarity has many direct and relevant applications. For instance, estimation of the similarity between texts has been employed for classification and structuring purposes [14]. Applied domains such as biomedicine compute semantic similarity in several scenarios, such as computer vision approaches [15] or the discovery of similar protein sequences [16]. Semantic Search in the context of the Semantic Web has of course been one of the common applications [17, 18]. In particular, semantic similarity computation is often employed together with a vast amount of clustering or classification techniques, which are introduced in the following section.

In comparison with semantic similarity based on ontologies, previous similarity functions relayed on structural or syntactic concerns. For example, the syntactic distance between two terms proposed by *Levenshtein* [19], also known as the string-edit distance, computes the number of atomic string operations needed to get from one string to another. The *Levenshtein* distance considers three atomic string operations: removing a character, inserting a character or substituting one character for another. This technique proved to be the most efficient in approximating string matching [20]. However, the result this kind of similarity function can offer against semantic ones is very limited. Other structural similarity functions such as *similarity flooding* [21] checks pairwise similarity among elements of two graph-based structures, which is then propagated to other elements. Similarly to the *Levenshtein* distance, similarity flooding is not able to achieve high precision without manual intervention by experts. It is due to these kind of similarity functions that ignores domain-specific semantics.

### 2.2. *Clustering algorithms*

We currently live in a world full of data. Every day, people encounter a large amount of information and store or represent it as data for further analysis and management. One of the vital means of dealing with these data is to classify or group them into a set of categories or clusters. In order to learn a new object or understand a new phenomenon, people always attempt to seek the features that can describe it, and further compare it with other known objects or phenomena, based on their similarity or dissimilarity, which is generalized as proximity according to certain standards or rules [22]. This is the key motivation for clustering techniques.

Clustering is the cornerstone of many knowledge discovery and management methodologies in which the classification or partition into coherent clusters from unstructured data sets is necessary [23].

Traditional clustering was originally developed in a statistical context, and was applied to numerical data. However, textual data was also introduced, in which values are expressed through linguistic labels or terms and are exploited in the form of categorical features or concepts [7]. Nevertheless, in this context, categorical data is usually treated at a syntactic level, and comparisons between their values have been limited to checking their equality or inequality. Unfortunately, a semantic interpretation of terms is hardly ever carried out. Ontology-based similarity is therefore used in combination with clustering algorithms to allow semantic equality/inequality to be evaluated between different concepts and clusters.

A wide variety of clustering approaches and algorithms exist in literature [22]. This paper focuses on agglomerative hierarchical clustering algorithms which recursively build a hierarchy of clusters by following a bottom-up approach. One of these algorithms starts in its own cluster; the pair of clusters with the highest similarity value is merged as one and is moved up the hierarchy. Hierarchical clustering algorithms can use different kinds of functions to compute the semantic similarity between clusters. Hierarchical clustering constructs a hierarchy of clusters that can be organized in a tree structure. Each node of the tree, including the root, represents a cluster, and the parent–child relationship among them enables different levels of clustering granularity to be explored.

Clustering methods have been practically applied in a wide variety of fields, ranging from engineering (e.g., pattern recognition, mechanical engineering, and electrical engineering), computer sciences (e.g., web mining, spatial database analysis, image segmentation, and privacy), medical sciences (e.g., genetics, biology, microbiology, and pathology), to social sciences and economics, among others. The following section introduces business process archeology as the scenario in which a hierarchical clustering based on semantic similarity is carried out for enhancing business process models that were retrieved by using reverse engineering.



DOI: 10.1002/smr

### 2.3. Business process archeology

Business processes are the sets of structured activities that are necessary to achieve given business goals. Business process management is recognized as one of the most significant intangible business assets involved in achieving competitive advantages and improving the quality of processes and operations carried out by enterprises and organizations [1]. Most business processes in organizations are supported by their enterprise information systems. Business process management is therefore optimized when organizations also combine it with the management of their Legacy (existing) Information Systems (LIS) [24].

Business process management helps organizations to address technological and organizational changes which will consequently improve, or maintain, their competitiveness [25]. Unfortunately, business processes are sometimes unavailable or outdated. Business processes may, on the one hand, be unavailable, owing to the simple fact that the organization has never managed its business processes. Business process representations may, on the other hand, be outdated and misaligned with actual, daily operation [2]. This misalignment principally occurs because information systems (which automate most business activities) evolve and get out of control as a consequence of iterative maintenance overtime. In both cases, that is, unavailability and misalignment, it is necessary to obtain actual business process descriptions.

From a software engineering viewpoint, business processes retrieved from LIS have several advantages. LIS may continue to be modernized on more occasions thanks to mining techniques. A recent study by the Software Engineering Institute states that it is first necessary to retrieve embedded business knowledge in order to modernize systems in line with the organization's business processes [26]. Organizations can thus modernize their legacy information systems whilst they align the new systems with their actual business processes. LIS are therefore evolved rather than being immediately retired and the return of investment on such systems is improved. This is because the lifecycles of these systems are extended, which saves costs as regards new developments from scratch [27].

Business process mining [28] is a set of techniques whose objective is to discover business processes from event logs recorded during system execution. However, in addition to process mining techniques, there are also other types of reverse engineering techniques that focus on software artifacts other than event logs. These reverse engineering techniques provide an alternative mechanism, and retrieve business processes from traditional information systems rather than from event logs. According to [4], these proposals are known as Business Process Archeology. Literature contains several works dealing with business process archeology. For example, Di Francesomariono *et al.* [29] consider GUIs of web applications for the discovery of business processes, whereas Cai *et al.* [30] propose requirement reacquisition through the recovery of use cases by means of interviewing the system's users. The system is dynamically traced in accordance with these use cases, and these traces are statically analyzed to recover business processes.

The nature of non-process-aware information systems entails five particular problems that introduce noise into retrieved business processes [31], that is, deviations from the actual business processes: (i) business process definitions are described implicitly in legacy code, so it is not obvious which events should be recorded in the event log; (ii) there is often different granularity between the callable units (invokable pieces of source code such as methods, function, and procedures) of an information system and the activities of a business process; (iii) legacy code contains not only business activities, but also technical aspects that have to be eliminated when a business process is retrieved using reverse engineering; (iv) traditional systems do not explicitly define business processes, signifying that their starts and ends have to be established; and finally, (v) the absence of process awareness signifies that it is not obvious how business activities should be correlated in each execution instance.

These problems often prevent techniques from obtaining complete and accurate business process from traditional systems. These issues have been addressed in literature in various ways. For example, some solutions combine the business expert's information with heuristics in order to address problems, (i) to (iv) shown earlier, which are related to the absence of certain information [32]. Other solutions deal with the fifth problem and define algorithms, which rely on statistical



DOI: 10.1002/smr

indicators [33]. The common aspect of all of these solutions is that they are applied as a pre-processing task or as a part of the reverse engineering techniques themselves.

Solutions that provide expert information before discovering business process models could facilitate the reduction of noise. Nevertheless, noise remains critical in the business processes discovered. Other noise reduction techniques therefore are applied after the business processes have been obtained. These techniques have an advantage over those applied at the beginning, because they can inspect processes directly and detect non-relevant parts related to noise problems. In this respect, the objective of clustering algorithms is to produce enhanced business process models (Figure 1), that is, with better levels of accuracy and completeness, which lead to a better understandability and modifiability.

## 3. BUSINESS PROCESS CLUSTERING BASED ON ONTOLOGIES

The proposed clustering technique places business tasks in groups by following an agglomerative hierarchical clustering algorithm. This algorithm recursively builds a hierarchy of clusters by following a bottom-up approach. The clustering algorithm initially considers every business task as an atomic cluster; the pair of clusters with the highest similarity value is merged as one and is moved up the hierarchy. Hierarchical clustering constructs a hierarchy of clusters that can be organized in a tree structure. Each node of the tree, including the root, represents a cluster, and the parent–child relationship among them enables the different levels of clustering granularity to be explored (Figure 2). Hierarchical clustering algorithms can use different functions to determine the similarity between clusters. The proposed algorithm employs an ontology-based similarity function focused on a semantic assessment of the business task labels, based on the idea of synonymous terms.

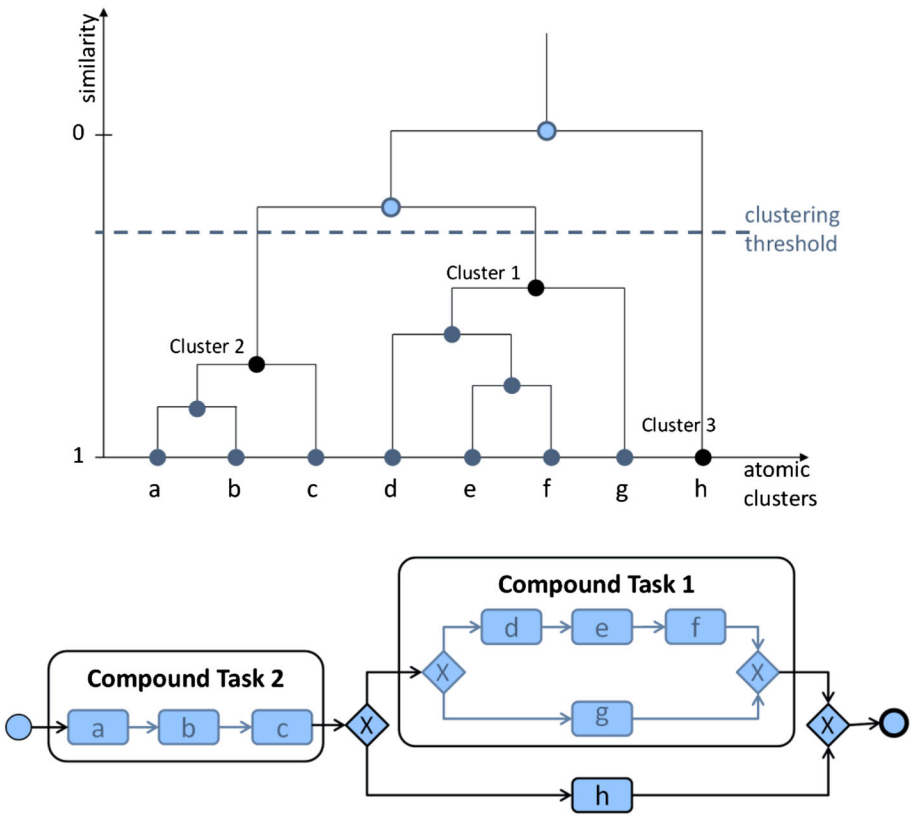


Figure 2. Agglomerative hierarchical clustering for business processes.



DOI: 10.1002/smr

**Algorithm 1.** Generic Hierarchical Clustering

**Input**: *BP (T, D, G, E, F, A); threshold*
**Output**: $BP_C$ ($T_C$, D, G, E, $F_C$, $A_C$)

```
1  T_C ← T
2  F_C ← ϕ
3  A_C ← ϕ
4  declare Clusters ← T
5  highSimilarity ← 1
6  while highSimilarity > threshold do
7      highSimilarity ← 0
8      for x, y ∈ Clusters do
9          s ← similarity(x, y)
10         if s > highSimilarity then
11             highSimilarity ← s
12             x_C ← x
13             y_C ← y
14         end
15     end
16     compoundTask ← {x_C, y_C}
17     Clusters ← Clusters ∪ {compoundTask}
18     Clusters ← Clusters - {x_C, y_C}
19     T_C ← T_C ∪ {compoundTask}
20     for each t ∈ T do
21         if t ∈ compoundTask then
22             t.subprocess ← compoundTask
23         end
24     end
25     for each f ∈F do
26         if f.source ∈ compoundTask then
27             f.source ← compoundTask
28             F_C ← F_C ∪ {f}
29         else if f.target ∈ compoundTask then
30             f.target ← compoundTask
31             F_C ← F_C ∪ {f}
32         end
33     end
34     for each a ∈ A do
35         if a.source ∈ compoundTask then
36             a.source ← compoundTask
37             A_C ← A_C ∪ {a}
38         else if a.target ∈ compoundTask then
39             a.target ← compoundTask
40             A_C ← A_C ∪ {a}
41         end
42     end
43 end
```



DOI: 10.1002/smr

Algorithm 1 formally presents how the agglomerative hierarchical clustering works. This algorithm follows a bottom-up clustering strategy (Figure 2). The algorithm takes input as a business process model, which is a tuple (T, D, G, E, F, A) in which T is a finite set of business tasks or activities; D is a finite set of data objects; G is a finite set of gateways (i.e., choice elements to open or close alternative branches in exclusive or inclusive way among other); E is a finite set of events (start, end or intermediate events). $F \subseteq (T \cup G \cup E) \times (T \cup G \cup E)$, is the set of sequence flows between tasks, gateways or events; $A \subseteq (T \times D) \cup (D \times T)$, is the set of associations between tasks and data objects. Algorithm 1 generates a clustered model $BP_C$, which is a tuple ($T_C$, D, G, E, $F_C$, $A_C$) with sets $T_C$, $F_C$, and $A_C$ modified regarding the original model.

All the tasks of a business process are initially atomic clusters (line 4), and the algorithm successively combines them into bigger clusters. It then checks the similarity between all the pairs of clusters for each of the iterations (lines 8 and 9). The pair of clusters that are most similar (lines 10–14) are grouped into a new cluster (lines 16 and 17), and the fine-grained clusters are removed from the cluster list (line 18). The tasks involved in the new cluster are merged into a compound task in the business process (lines 20–42).

Both of the clusters that represent two subprocesses are encapsulated in a compound business task. The new cluster is then connected to other tasks in the business process by means of sequence flows and associations. All the sequence flows that are adjacent to or an incident of any task of the previous clusters become adjacent and incident sequence flows of the compound task (the new cluster) (lines 25–33). Associations with object data are similarly modified in order to link the new compound tasks with all the data objects related to the subgraph (lines 34–42).

The proposed algorithm has to be set up using experts' information, which defines the thresholds for clustering. The threshold of a clustering algorithm defines the stop flag of that algorithm, that is, when the algorithm stops grouping similar clusters into a new one. Thresholds are often defined by using a similarity value of between 0 and 1. For example, a threshold of 0.4 signifies that the algorithms will stop when there is no pair of clusters with a similarity value above 0.4. The experts in charge of defining these thresholds are business experts (together with systems analysts who know the nature of the input source code). On the one hand, the business experts have to take into account the occurrence of patterns such as the number of strongly connected components or the granularity of data objects. On the other hand, the systems analysts can support the threshold definition by providing information about the nature, domain, and platform of the input information system. The definition of these thresholds is obtained from an iterative process in which experts continuously monitor results after the usage of a certain thresholds. These thresholds are, however, often selected in a heuristic manner at the outset, by considering benchmark values from similar domains. But it should be noted that the selection of clustering thresholds is a research area itself in which much empirical validation is necessary [34].

The proposed algorithm uses an ontology-based similarity function. The function searches for two terms in a given ontology and calculates the semantic distance between them. Formal ontologies are used by this algorithm as a mechanism with which to represent the semantics of a specific domain, that is, the domain related to certain business process models. The similarity function uses formal ontologies provided by domain experts to determine the semantic similarity between two terms in relation to a common semantic reference. This approach was specially designed for formal ontologies defined using OWL [12] because this specification follows an XML format, and OWL ontologies can therefore be seen and treated as trees of concepts. However, in spite the fact that ontologies and thesauri are not exactly the same [6], in this approach, OWL ontologies are considered as a thesaurus tree, in which the root is the most generic concept defining a knowledge domain. The root concept is specified into a set of child nodes representing more-specific synonyms. The evaluation of the semantic similarity between two terms is consequently equivalent to calculating the distance between two nodes in a tree.

The similarity function used by the clustering algorithm computes the similarity between two business tasks by navigating the ontology tree until both task labels match with a class or concept in the ontology tree. The similarity between both tasks is then equivalent to the distance between the three nodes.

The distance between two nodes $N_1$ and $N_2$ in a tree is affected by both the distance from $N_1$ to the *lowest common ancestor* (LCA) node, and the distance between LCA and $N_2$. LCA is the common root



DOI: 10.1002/smr

of the sub-tree that contains both nodes. This path distance is not so useful for hierarchical structures because it does not differentiate between the similarities of node pairs located at different depths of the ontology. The distance between tree nodes is thus also affected by the depth of the LCA. The similarity between two nodes is therefore the inverse of the distance (Eq. (1)), and is formally defined as the *tree node similarity* [35]. In formula (1), '*dis*' is a function that returns the number of arcs between a node and one of its ancestor nodes; $LCA_{ij}$ is the LCA of the nodes $N_i$ and $N_j$; and '*root*' represents the root node of the whole ontology tree.

$$\mathrm{sim}(Ni, Nj) = \frac{1 + dis(LCAij, root)}{1 + dis(LCAij, root) + dis(Ni, LCAij) + dis(Nj, LCAij)} \tag{1}$$

Figure 3 shows an example in an ontology defined for biologic organisms, which is used to compute the similarity between a bird and a human. The LCA node for both bird and human is vertebrate in accordance with the given ontology. The distance between LCA and the root is therefore 2, whereas the distance between LCA and both nodes is 1 for bird and 2 for human. Having applied Eq. (1), the similarity between both nodes (bird and human) is 0.5.

Algorithm 2 calculates the final sematic similarity value, which can be used in the agglomerative hierarchical clustering defined in Algorithm 2 (line 8). This function estimates the similarity between two clusters, which could be two atomic business tasks (at the beginning of the algorithm) or combinations of atomic and compound business tasks (once two or more clusters have been previously combined). Algorithm 2 calculates the similarity by comparing all the words involved in both task labels (lines 2 and 3). The algorithm first calculates the sum of the similarities between all the term pairs by means of the tree node similarity function (1) in accordance with the given ontology (lines 4 and 5). Finally, the total similarity is normalized with regard to the maximum number of words in one of the business tasks (line 8).

**Algorithm 2.** Semantic Similarity

**Input**: *t1*; *t2*; *ontology*
**Output**: *similarity*

```
1  declare totalSimilarity←0
2  for i from 1 to |t1.terms| do
3          for j from 1 to |t2.terms| do
4          totalSimililarity←totalSimililarity +
5          treeNodeSimilarity(ontology, t1.terms[i], t2.terms[j])
6      end
7  end
8  similarity←totalSimilarity/ maximum (|t1.terms|, |t2.terms|)
```

## 4. EXPERIMENTATION

In order to facilitate the usage of business process models retrieved from LIS within the software engineering field, the proposed technique has been validated via an industrial multi-case study involving four information systems: (i) a system devoted to university student enrollments; (ii) a laboratory system for the water and waste industry; (iii) a system that supports the management and simulation of decision tables; and (iv) a mobile healthcare application for diabetes patients. The case study follows the formal protocol used to conduct case studies in software engineering proposed by [36]. This protocol defines the following sections: research goals and questions, design and variables, case selection, execution procedure, data collection, analysis and interpretation, and threats to validity.



DOI: 10.1002/smr

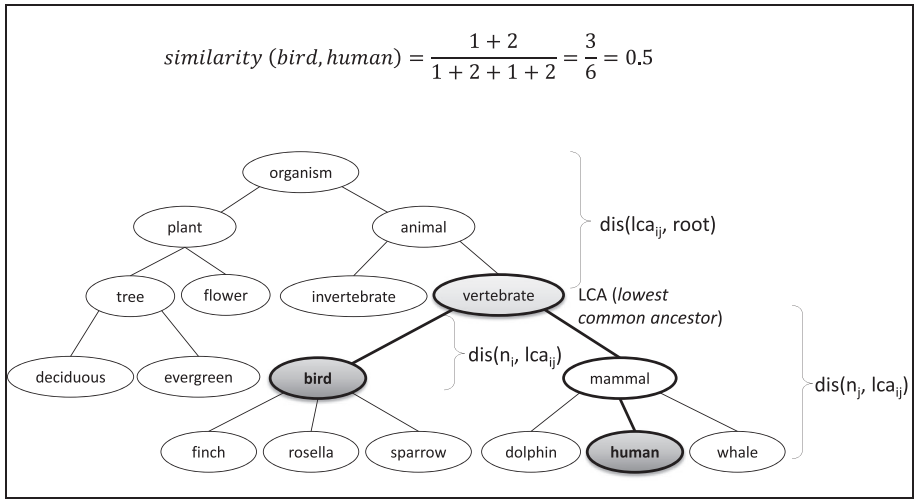


Figure 3. Tree node similarity with reference to a particular ontology.

A replication package is available online in [37], which contains an additional material in the involved ontologies, all the business process models under study and the refactored ones and the data sheets collected during the experiment.

### *4.1. Research goal and questions*

The *object* of study is the clustering technique with which to reduce noise in business process models retrieved using reverse engineering, and an improvement to their quality levels. The goal of this study is to quantitatively evaluate the specific properties of the technique, such as the best thresholds, its effectiveness, and scalability.

There are three main research questions as regards the research goal. Firstly, RQ1 attempts to figure out the best thresholds by evaluating the effect of considering different threshold values during clustering execution. Different thresholds, which work like stop flags, lead to differently clustered business process models. The goal is to determine which range will obtain the best result. Secondly, RQ2 deals with the effectiveness of the technique by testing whether the quality of business process models really is improved after applying the clustering technique. Finally, RQ3 evaluates the efficiency of the technique by testing its scalability to larger and more complex business processes.

RQ1. How does the clustering work as regards the different thresholds? (Thresholds).
RQ2. Can the technique improve the quality (in terms of understandability and modifiability) of the business process models obtained from information systems using reverse engineering? (Effectiveness).
RQ3. Is the technique scalable to large and complex business process models? (Scalability).

### *4.2. Design and variables*

In order to answer the research questions, the study evaluates the related properties (best threshold, effectiveness, and scalability) by analyzing all of the business processes modified after the ontology-based clustering algorithm has been applied. Certain measures are evaluated to provide quantitative answers to the proposed research questions.

The study is a multi-case study, because it considers various information systems. The study also follows an *embedded* design [36], because each study focuses on several analysis units within each single case. The analysis units are the various business process models retrieved from each LIS using reverse engineering. Both the MARBLE technique and its respective tool [38] have been used to retrieve business process from information systems. The sample used in the study consists of 43 business process models obtained from the four systems (with 9, 13, 13, and 8 models for each



DOI: 10.1002/smr

system), which are considered to be the independent variable, that is, the measures defined will be evaluated for each business process model. The dependent variables are a set of measures, which are evaluated in order to answer the aforementioned research questions (see Table I).

Firstly, the number of clustered business tasks is considered in order to measure the clustering ratio for each threshold used (RQ1). Because the behavior of business process models may also vary when different threshold values are used, the study uses the four measures in combination with RQ1 to evaluate RQ2. The objective of these measures is to quantify the effectiveness of the clustering algorithm, that is, effectiveness relates to getting the right things done (Table I).

- *Size*. This is the number of nodes in a business process (i.e., the number of tasks, data objects, gateways, and events). This measure is related to the clustering results, because a larger size may be related to noise problems (e.g., many tasks retrieved erroneously). It is thus expected that the size will decrease after the clustering algorithm has been applied.
- *Density*. This is the ratio between the total number of arcs (i.e., sequence flows and associations) in a business process model and the theoretical maximum number of arcs (1). The maximum number of arcs in a graph of $n$ nodes is $(n \cdot (n-1))/2$. Density is an indicator of noise problems, because a high density could indicate that a business process has been retrieved with many redundant sequence flows and associations.
- *Connectivity.* This evaluates the ratio of the total number of arcs in a business process to its total number of nodes (2). With regard to noise problems, low connectivity could indicate that a business process has been retrieved without the necessary connections between tasks. The connectivity should be higher after the clustering algorithm has been applied.
- *Separability*. This is the ratio of the number of cut vertices (articulation points), that is, nodes that solely connect two strongly-connected components, in relation to the total number of nodes in the business process (3). A higher separability indicates that the business process has many noise problems such as isolated tasks and smaller sub-graphs. This measure should therefore be lower after the semantic clustering technique has been applied.

Finally, in order to answer RQ3 (Table I), which is related to the scalability of the clustering technique, the study assesses how long it took to execute the clustering algorithm.

### 4.3. Case selection

There are four information systems under study. These cases were selected according to four criteria:

- **C1,** which guarantees that the LIS selected is an information system that supports an organization's business operation. This criterion discards, for example, embedded systems or real-time control systems with specific-purposes, which do not necessarily support the respective business organization's business activities.
- **C2,** which ensures that the system chosen really is an LIS. This criterion considers the amount of modifications (from the time at which the system was first released) that have altered the business processes supported by the system.
- **C3**, which ensures that the system is not a toy program and it is in production currently.

Table I. Relationship between research questions and measures.

| RQ | Quality feature | Measure | Formula |
|---|---|---|---|
| RQ1 | Best threshold | Clustering ratio | Number of clustered business tasks |
| RQ2 | Effectiveness | Size | Number of nodes |
| | | Density | $\text{Den} = \frac{2 \text{ number of arcs}}{\text{number of nodes(number of nodes-1)}}$ (1) |
| | | Connectivity | $\text{Conn} = \frac{\text{number of arcs}}{\text{number of nodes}}$ (2) |
| | | Separability | $\text{Sep} = \frac{\text{number of cut vertices}}{\text{number of nodes}}$ (3) |
| RQ3 | Scalability | Clustering time | Measured in milliseconds |



DOI: 10.1002/smr

- **C4**, which guarantees that the system is a traditional (non-process aware) information system based on the Java platform, because the reverse engineering tool for retrieving the sample of business processes works especially with Java-based systems.

After evaluating several systems that were made available by partner companies and organizations, according to the aforementioned criteria, four systems were selected for study. Table II shows the name of each system, along with a brief description of it and the size of its source code in thousands of lines.

The four systems under study are traditional Java-based systems, that is, these systems do not explicitly support the management of business processes. In fact, this kind of systems was particularly desired during the selection procedure because of these systems are prone to produce a high level of noise problems when reverse engineering techniques are applied to discover and retrieve business process models. A high level of noise make it possible to have a good understanding on how the clustering algorithm works.

### *4.4. Execution procedure*

The multi-case study was executed in a finite set of steps, which was partially supported by a tool specially developed to execute the proposed clustering algorithm.

After meetings between researchers and staff of both candidate, partner companies, the information systems under study were finally selected according to the case selection criteria. At this point, the business experts from partner companies were appointed. These experts provide the information needed to define the domain-specific ontologies. Steps 2 to 4 were then repeated for each information system.

1. Business processes were obtained from the legacy source code of the information system by using reverse engineering. MARBLE tool [38] was used to retrieve business processes. MARBLE follows the model-driven development principles, that is, it treats all the artifacts involved as models and provides model transformations between models at different abstraction levels. MARBLE implements a set of business patterns with which to transform pieces of source code into sketches of business processes. The whole sample of 43 business processes is available online in [37]. The values of all the measures were collected with original models before clustering. Size, density, connectivity, and separability were automatically computed by the tool. These experts measured precision and recall by (i) identifying all the relevant tasks before examining the business process model to then detect those relevant tasks that had not been retrieved; and (ii) discarding tasks that were also retrieved but which were not relevant.
2. Business experts related to each information system defined formal ontologies in OWL in accordance with the knowledge domain of their systems. They carried out this activity with the assistance of the authors of this paper and by means of the *Protege* tool. These ontologies are available online in [37]. Each system under study is clustered by using a particular domain-specific ontology.
3. The business process models retrieved in step 2 were then used to apply the clustering algorithm (cf. Section 3). The clustering algorithm was executed by using different similarity thresholds (from 0.9 to 0.1). The respective measures were collected after each clustering had taken place.

Table II. Information systems under study.

| Id | Name | Description | Size (KLOC) |
|---|---|---|---|
| S1 | University enrollments | Supports university students' self-enrollments in various Spanish universities | 26.7 |
| S2 | Villasante lab | Manages a water and waste industry laboratory | 28.8 |
| S3 | Magic table | Creates, manages, and simulates decision tables used to associate conditions with domain-specific actions | 33.3 |
| S4 | Diabetes care | An Android application for diabetes patients, which analyzes blood (using an external device) and suggests diet plans. | 9.9 |



DOI: 10.1002/smr

The semantic clustering algorithm employed four different ontologies (depending on the particular domain), which were defined by business experts. Because of the space limitation, all the source business process models as well as the clustered ones are available online [37] together with the replication package. Anyway, in order to illustrate how the approach works, Figure 4 shows a source business process model and its respective clustered model. These business process diagrams corresponds to the 30 model, which was obtained from the third system under study. This source model was clustered with a threshold of 0.4. The original size was 25, which was reduced to 6 after clustering. Density and connectivity, respectively, were 8 and 32 times more than original values. Finally, separability was reduced to 15% with respect to the original separability.

4. All of the key information related to the generation of business process models (step 2) and their clustering (steps 3) was collected according to the data collection plan (cf. Subsection 4.5).
5. The data collected were then analyzed and interpreted in order to answer the research questions (cf. Subsection 4.6).

### *4.5. Data collection*

Both the data to be collected and their sources were defined before the case study commenced in order to ensure repeatability. Table III shows the data concerning the application of the clustering algorithm and presents (i) the number of the case under study (cf. Section 4.3); (ii) the similarity threshold considered in each execution (between 0.9 and 0.1); (iii) the number of clusters obtained for a particular threshold; (iv) the size of the business process; (v) the density of the process; (vi) the

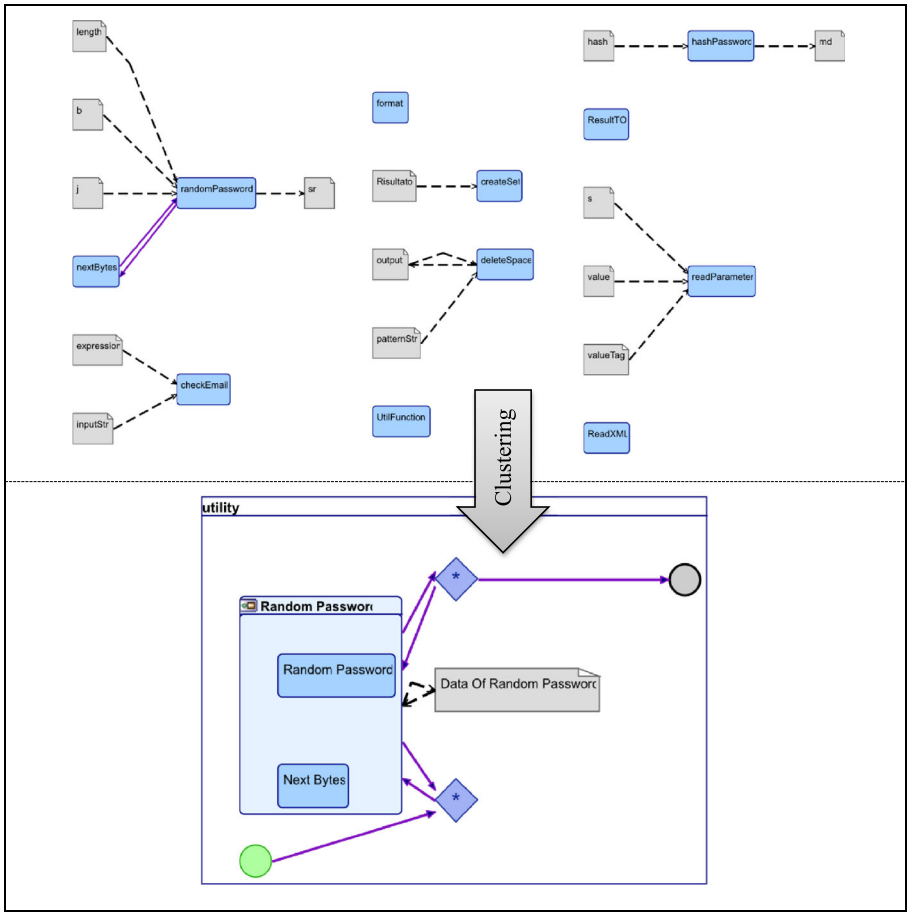


Figure 4. Example of a business process model (top) and its respective clustered model (bottom).



DOI: 10.1002/smr

Table III. Measures collected for each system under study.

| Case | Threshold | # Clusters | Size | Density | Connectivity | Separability | Time (ms) |
|---|---|---|---|---|---|---|---|
| S1 | 0.9 | 0.889 | 72.000 | 0.137 | 5.044 | 3.778 | 462 |
| S1 | 0.8 | 1.000 | 71.889 | 0.137 | 5.054 | 3.778 | 448 |
| S1 | 0.7 | 1.444 | 71.333 | 0.142 | 6.174 | 3.778 | 470 |
| S1 | 0.6 | 1.889 | 70.556 | 0.142 | 6.250 | 3.778 | 437 |
| S1 | 0.5 | 2.667 | 69.667 | 0.143 | 6.354 | 3.667 | 427 |
| S1 | 0.4 | 3.778 | 67.667 | 0.143 | 6.619 | 3.667 | 423 |
| S1 | 0.3 | 5.222 | 62.889 | 0.144 | 7.670 | 3.556 | 423 |
| S1 | 0.2 | 5.111 | 57.778 | 0.163 | 10.636 | 3.444 | 518 |
| S1 | 0.1 | 2.750 | 67.972 | 0.144 | 6.725 | 3.681 | 451 |
| S2 | 0.9 | 0.318 | 9.545 | 0.067 | 4.400 | 2.364 | 24 |
| S2 | 0.8 | 0.318 | 9.545 | 0.067 | 4.400 | 2.364 | 23 |
| S2 | 0.7 | 0.364 | 9.500 | 0.067 | 4.409 | 2.364 | 24 |
| S2 | 0.6 | 0.409 | 9.409 | 0.067 | 4.427 | 2.364 | 23 |
| S2 | 0.5 | 0.545 | 9.273 | 0.068 | 4.443 | 2.364 | 27 |
| S2 | 0.4 | 0.864 | 8.818 | 0.069 | 4.720 | 2.364 | 24 |
| S2 | 0.3 | 1.091 | 8.364 | 0.069 | 4.821 | 2.364 | 23 |
| S2 | 0.2 | 0.909 | 7.818 | 0.071 | 5.135 | 2.364 | 25 |
| S2 | 0.1 | 0.636 | 7.227 | 0.074 | 6.659 | 2.364 | 24 |
| S3 | 0.9 | 0.538 | 33.923 | 0.133 | 5.450 | 2.462 | 187 |
| S3 | 0.8 | 0.692 | 33.692 | 0.134 | 5.710 | 2.462 | 185 |
| S3 | 0.7 | 0.846 | 33.538 | 0.134 | 5.719 | 2.462 | 196 |
| S3 | 0.6 | 1.692 | 32.462 | 0.134 | 5.804 | 2.462 | 208 |
| S3 | 0.5 | 2.077 | 31.692 | 0.134 | 5.877 | 2.462 | 208 |
| S3 | 0.4 | 2.538 | 30.385 | 0.134 | 6.034 | 2.462 | 189 |
| S3 | 0.3 | 2.692 | 29.615 | 0.134 | 6.179 | 2.462 | 186 |
| S3 | 0.2 | 2.462 | 27.846 | 0.134 | 6.726 | 2.462 | 187 |
| S3 | 0.1 | 2.000 | 26.538 | 0.138 | 8.069 | 2.462 | 195 |
| S4 | 0.9 | 0.875 | 35.375 | 0.181 | 3.787 | 2.875 | 59 |
| S4 | 0.8 | 1.000 | 35.250 | 0.181 | 3.827 | 2.875 | 61 |
| S4 | 0.7 | 1.000 | 35.250 | 0.181 | 3.827 | 2.875 | 59 |
| S4 | 0.6 | 1.000 | 35.250 | 0.181 | 3.827 | 2.875 | 62 |
| S4 | 0.5 | 1.000 | 35.250 | 0.181 | 3.827 | 2.875 | 69 |
| S4 | 0.4 | 1.000 | 35.250 | 0.181 | 3.827 | 2.875 | 58 |
| S4 | 0.3 | 1.000 | 35.250 | 0.181 | 3.827 | 2.875 | 61 |
| S4 | 0.2 | 1.125 | 34.625 | 0.183 | 4.014 | 2.875 | 54 |
| S4 | 0.1 | 1.875 | 33.625 | 0.185 | 4.184 | 2.875 | 57 |

connectivity ratio; (vii) the separability; and finally, (viii) how long it took to execute the clustering algorithm, which is specified in milliseconds.

Table IV summarizes the same information in columns. However, the rows represent the normalized values for the means of the four systems. The means are calculated by means of a harmonic mean in order to allow the differences between the four systems to be addressed.

### *4.6. Analysis and Interpretation*

After data collection, the data analysis obtained the evidence needed to answer research questions RQ1 to RQ3, which are related to thresholds, effectiveness, and scalability (cf. section 4.1).

*4.6.1. Thresholds (RQ1).* In order to answer RQ1, the study executed the ontology-based clustering algorithm by using all the different similarity thresholds between 0 and 1, which were discretized every 0.1. This signifies that nine different values were considered, from 0.9 to 0.1. Figure 5 and Figure 6 shows the average evolution of the measure values taken from each system throughout all of the different thresholds. Figure 5 first shows the evolution of the number of clustered business tasks for all the systems. This chart shows the clustering ratio for all the similarity thresholds. This ratio increases at the same time as the similarity threshold decreases. However, the number of clustered tasks surprisingly decreases at around 0.3.



DOI: 10.1002/smr

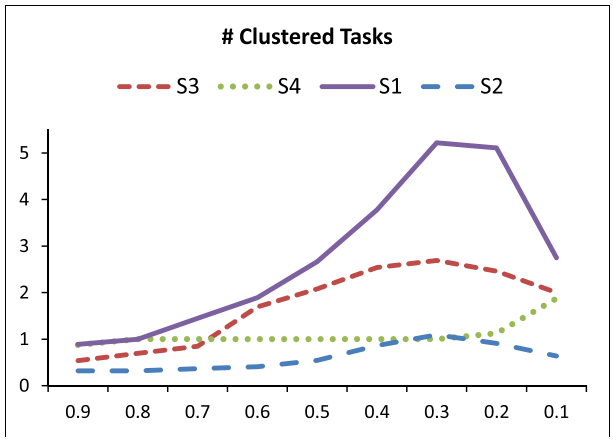


Figure 5. Number of clustered tasks achieved for each system with different similarity threshold.

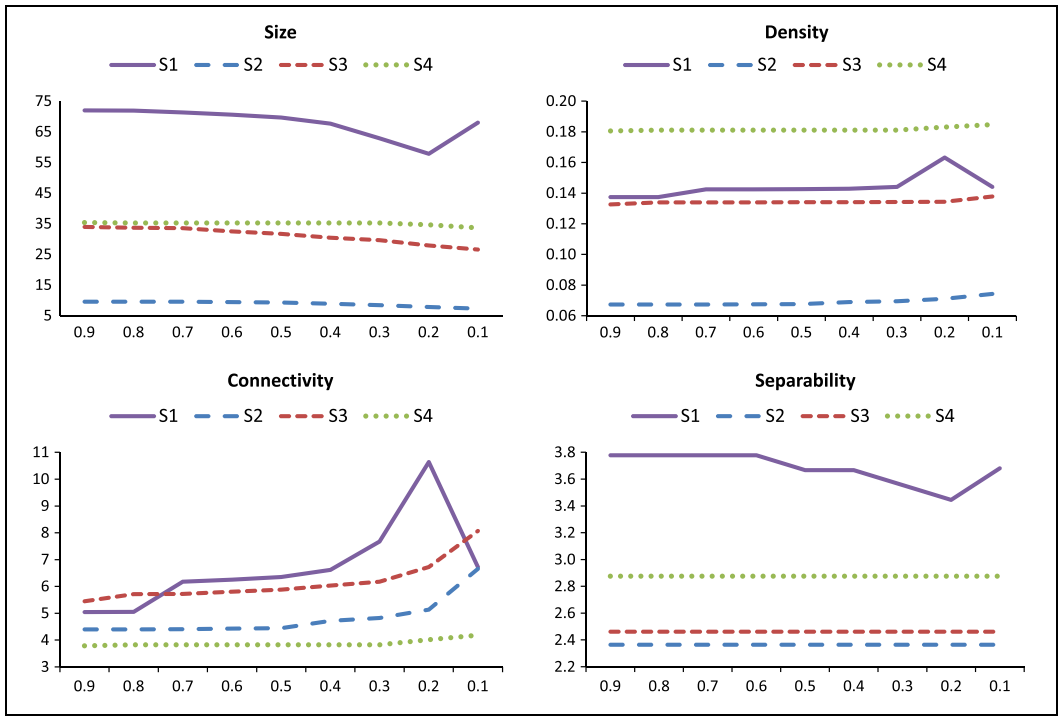


Figure 6. Size, density, connectivity and separability values for different similarity thresholds.

Figure 6 provides the trend that each measure follows. Size and separability decrease with lower thresholds, whereas density and connectivity increase. Suffice it to say that there are significant differences between the four systems, in addition to the existence of peaks in certain similarity thresholds for some systems.

Because some measures increase while others decrease, obtaining the best threshold becomes an optimization problem, which can be solved with a *minimax* approximation [39]. Table V presents the means of the measures obtained in all the systems under study. These values are also normalized between 0 and 1. The last column of Table V provides the *minimax* value obtained by combining all the values (i.e., number of clustered tasks, size, density, connectivity, and separability). This function considers the same weight (0.2) for the five measure values. Table V shows that similarity thresholds of around 0.4 and 0.2 lead to the ontology-based clustering algorithm's best performance.



DOI: 10.1002/smr

Table IV. Normalized mean values for all the systems under study.

| Threshold | # Clusters | Size | Density | Connectivity | Separability | Time (ms) |
|---|---|---|---|---|---|---|
| 0.9 | −1.290 | −0.888 | −0.962 | −1.055 | −0.816 | 0.010 |
| 0.8 | −1.152 | −0.864 | −0.858 | −0.935 | −0.816 | −0.538 |
| 0.7 | −0.901 | −0.807 | −0.545 | −0.529 | −0.816 | −0.184 |
| 0.6 | −0.502 | −0.646 | −0.509 | −0.466 | −0.816 | −0.463 |
| 0.5 | 0.001 | −0.465 | −0.466 | −0.400 | 0.093 | 2.610 |
| 0.4 | 0.813 | 0.040 | −0.110 | −0.107 | 0.093 | −0.505 |
| 0.3 | 1.280 | 0.557 | 0.122 | 0.274 | 1.049 | −0.563 |
| 0.2 | 1.113 | 1.267 | 1.636 | 1.394 | 2.053 | −0.110 |
| 0.1 | 0.638 | 1.806 | 1.692 | 1.824 | −0.023 | −0.257 |

Table V. Normalized mean values for all the systems under study.

| Threshold | # Clustered tasks | Size | Density | Connectivity | Separability | *Minimax* value |
|---|---|---|---|---|---|---|
| 0.9 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.8 | 0.05 | 0.01 | 0.04 | 0.04 | 0.00 | 0.03 |
| 0.7 | 0.15 | 0.03 | 0.16 | 0.18 | 0.00 | 0.10 |
| 0.6 | 0.31 | 0.09 | 0.17 | 0.20 | 0.00 | 0.15 |
| 0.5 | 0.50 | 0.16 | 0.19 | 0.23 | 0.32 | 0.28 |
| 0.4 | 0.82 | 0.34 | 0.32 | 0.33 | 0.32 | 0.43 |
| 0.3 | 1.00 | 0.54 | 0.41 | 0.46 | 0.65 | 0.61 |
| 0.2 | 0.93 | 0.80 | 0.98 | 0.85 | 1.00 | 0.91 |
| 0.1 | 0.75 | 1.00 | 1.00 | 1.00 | 0.28 | 0.81 |

These results can also be graphically contrasted in Figure 7. This figure provides a chart in which all the curves (according to the normalized mean values) are overlapped, showing the cutoff points at which some measures are maximum while others are minimum.

*4.6.2. Effectiveness (RQ2).* Once the range of the best similarity thresholds had been obtained, RQ2 was assessed by means of a comparison between the measure values taken before and after the application of the clustering algorithm. This comparison was made by taking into account the measure values obtained with a similarity threshold of 0.3. Table VI shows the means of size, density, connectivity, and separability obtained for each system under study. The last row additionally provides the average measure values for all the systems. Table VI also provides the gain (positive or negative) obtained for each cell. The gain obtained for each measure and system can be graphically observed in Figure 8. Black bars signify that the measure has decreased, whereas white bars signify that the measure is higher with regard to the original business process models before the clustering technique was applied.

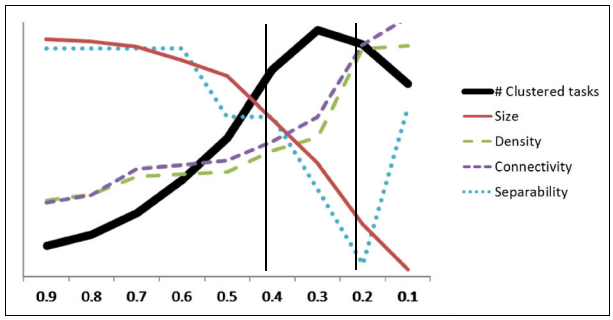


Figure 7. Normalized mean values for all the measures as regards different similarity thresholds.



DOI: 10.1002/smr

Table VI. Gain obtained for each effectiveness measure with a threshold of 0.3.

| System | Size | Density | Connectivity | Separability |
|---|---|---|---|---|
| Reference | 72.89 | 0.11 | 4.07 | 4.00 |
| S1 | 62.89(−14%) | 0.14 (+26%) | 7.67 (+88%) | 3.56(−11%) |
| S2 | 8.36 (−89%) | 0.07 (−39%) | 4.82 (+18%) | 2.36 (−41%) |
| S3 | 29.62(−18%) | 0.13 (+18%) | 6.18 (+52%) | 2.46 (−38%) |
| S4 | 35.25(−59%) | 0.18 (+59%) | 3.83(−6%) | 2.88 (−28%) |
| Overall | 34.03(−16%) | 0.13 (+16%) | 5.62 (+38%) | 2.81 (−30%) |

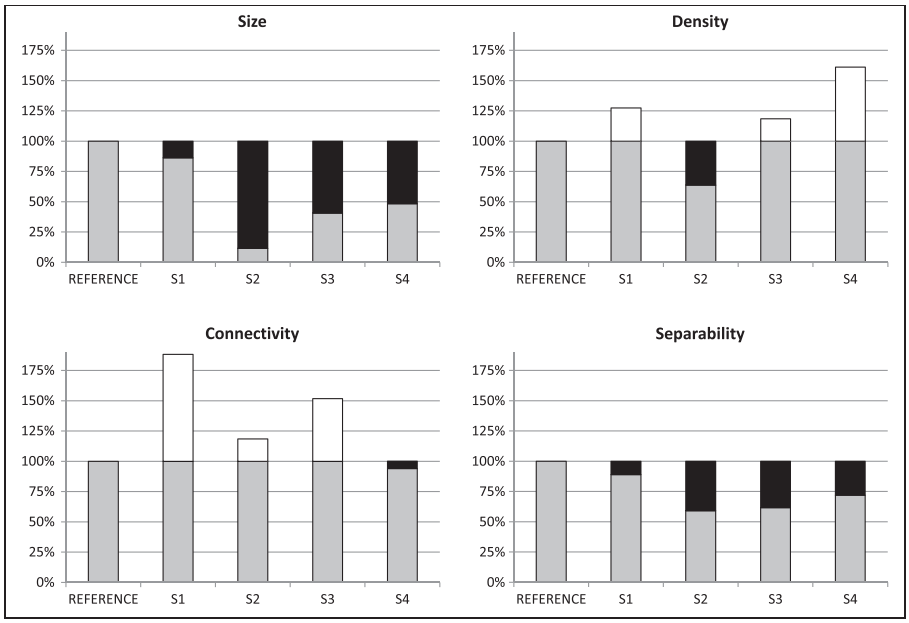


Figure 8. Size, density, connectivity and separability gain obtained with a threshold of 0.3.

The gain values show that both size and separability have been reduced in accordance with the expected behavior (cf. Section 4.2). The reduction is 16% for size and 30% for separability. The highest gain was obtained in system S2 for these two measures, whereas the lowest gain was obtained in system S1. Density is an average of 16% higher after applying the clustering algorithm. However, density, which should also have been minimized, is increased in three of the four systems under study (Figure 8). This collateral effect might have occurred as a result of the fact that the theoretical number of arcs has been reduced, owing to a lower number of business tasks after clustering, whereas the actual number of arcs has been preserved. Finally, with regard to connectivity, the results obtained are in accordance with the expected result because the gain was positive in three of the four systems, and connectivity was in fact the measure that was most improved, with an average gain of 38% (Table VI).

In spite of the fact that density was not improved, RQ2 can be positively answered, because size, connectivity, and separability were improved after applying the algorithm with a similarity threshold of 0.3. This signifies that it would be feasible to use the ontology-based clustering algorithm to obtain better business process models.

*4.6.3. Scalability (RQ3).* The scalability of the clustering algorithms was tested by using a regression model. A linear regression model considers the clustering time as a dependent variable, and the size of the business process models is seen as the independent variable. Figure 9 provides the scatter chart for the size/time of the clustering algorithm applied to the 43 business process models obtained from the four systems.



DOI: 10.1002/smr

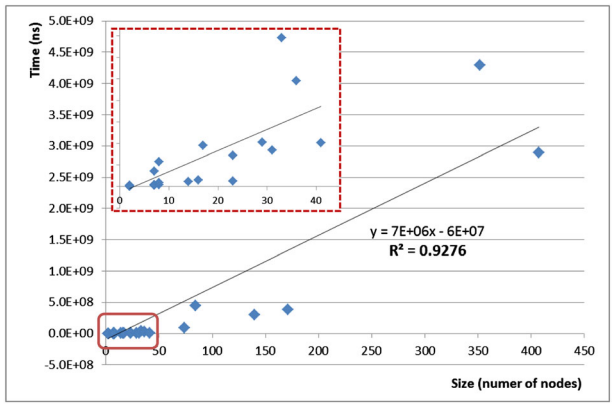


Figure 9. Linear regression model for the ontology-based clustering algorithm.

Figure 9 also shows the regression line ($y = 7 \cdot 10^6 \cdot x - 6 \cdot 10^7$), which has a positive linear relationship with $R^2 = 0.93$. The correlation coefficient $R^2$ (between $-1$ and 1) is the degree to which the real values of the dependent variable are close to the predicted values. The $R^2$ value obtained is high, and is very close to 1, and the proposed linear regression model is thus suitable for explaining the data obtained in this study, that is, there is no quadratic or exponential relationship between the clustering time and the size. The increase in time for larger business process models will consequently be linear, and this time may be assumable.

RQ3 can consequently be answered positively, thus signifying that the proposed ontology-based clustering algorithm is efficient and could be applied to larger and more complex systems with a controllable effort as regards to the size of business process models.

### 4.7. *Threats to validity*

This section shows the threats to the validity of the multi-case study, which are categorized into internal, construct and external validity. With regard to the *internal* validity, a sample of 43 business process models was retrieved from four different information systems, and it was thus possible to obtain statistically representative results. Nevertheless, the study could be replicated by using more information systems in order to attain a larger sample of business processes. There are also two decisive threats to the internal validity. The first of these is related to the way in which the business process models were obtained. MARBLE, the supporting tool used to obtain the business process models, might have affected the initial sample of the business process models. The second is that if the study were to be replicated with the same information systems but with different business experts, different formal ontologies might be defined, and the results of the clustering algorithm might be biased. This is owing to the fact that each expert would probably provide her/his subjective viewpoint during the definition of the formal ontology. The aforementioned threats could be mitigated by replicating the study using different configurations.

With regard to the *construct* validity, the measures selected were an appropriate means to answer the research questions. Firstly, size, density, connectivity, and separability are measures from the business process management field that are used to assess the understandability and modifiability of business process models. These measures are therefore suitable for quantifying whether the clustering technique is effective. Regarding the assessment of thresholds, the focus on the clustering ratio (i.e., the number of clustered business tasks) might result simplistic. For example, higher ratios are not necessarily better because the clustering might result in a wrong business process model from the semantic point of view. The possible loss of semantics as well as the understandability level should be assessed in future replications based on experts' opinions because of the subjective nature of these quality features. Regrettably, although semantic preservation in business process models could be checked based on the business experts' opinion, the implementation is difficult due to the time constraints of experts who ideally should be involved in the experimentation.



DOI: 10.1002/smr

Another important threat to the construct validity is the 0.40 clustering threshold used by the ontology-based similarity function. These thresholds were arbitrarily chosen based on the previous experience in industrial projects of reverse engineering and refactoring of business process models. However, different thresholds may lead to different results. As a consequence, the assessment of the optimal thresholds should be carried out through different empirical studies by using specific-purpose techniques like the Vender method or ROC-curves like similar approaches did [40, 41]. However, the study and determination of these optimal thresholds is a huge endeavor that in this case is outside of the scope of this paper.

On the other hand, although the clustering time computed was appropriate as regards discovering the scalability of the technique, the time taken to build the formal ontology has not been taken into account. Fortunately, clustering requires manual work before the algorithm only the first time that a formal ontology is defined for a particular domain. Ontologies for the same (or even similar) domains can be reused or at least modified with less effort than that needed to build the ontology from scratch. What is more, a vast amount of available open ontologies can be directly downloaded and easily adapted. Another important factor is the different levels of expertise of the practitioners who define the formal ontologies, which could affect the time. The time spent on this manual activity was therefore discarded from the scalability analysis. A more in depth study of this aspect may take place in future replications.

Finally, *external* validity evaluates whether the results are true and not biased for the whole population to which we wish to generalize them. This study considers the whole population to be business process models retrieved from legacy information systems using reverse engineering. The results obtained can be strictly generalized to this population with the particularity that all the information systems under study are based on the Java platform. This restriction is related to MARBLE, the tool used to retrieve business processes. The specific platform of the cases chosen is a threat, which could be mitigated by replicating the study using LIS implemented on different platforms.

## 5. CONCLUSIONS

Business processes are a valuable intangible asset for companies, allowing companies to manage their daily operations and maintain their competitiveness. From a software engineering viewpoint, business process is also the starting point for obtaining the requirements of new development projects or modernization projects. Regrettably, business processes are sometimes unavailable or outdated, because of uncontrolled maintenance. Thus, reverse engineering of business process enables business process to be discovered and retrieved from existing information systems, which embed many business rules that are not available anywhere else. Those techniques obtain well-designed business processes, but these are often retrieved with quality faults. Clustering of business activities of such business process models are recurrently applied to reduce these quality faults and improve the understandability and modifiability. This paper has proposed an agglomerative, hierarchical clustering algorithm that relays on ontologies to group business activities according to their semantic similarity.

This approach takes advantage of formal ontologies, which help to depict semantics in specific domains. Despite this, the practical usage and computation of domain ontologies in some concept modeling techniques and methods is unclear and is not well defined. For example, the matching of an actual element of the universe of discourse as regards elements previously depicted in a given formal ontology is an important challenge. When the element is not found in the formal ontology, the way in which the most similar element is chosen is a recurrent problem. Semantic similarity states how taxonomically near two terms are, because they share some aspects of their meaning. The assessment of ontology-based similarity is a key task within clustering algorithms.

This paper shows how to compute and use the ontology-based similarity within a clustering algorithm to improve the quality of business process models, which depict an organization's operative workflow. The objective of the proposed technique is to restructure and reduce fine-grained elements of business process models that had previously been obtained from information systems by using reverse engineering. The principal contribution of this paper is the usage of an



DOI: 10.1002/smr

ontology-based similarity function and the way in which it is computed through the use of the formal ontology tree.

This paper has presented an industrial multi-case study that involves four legacy information systems from which 43 business process models were retrieved. The study assessed how the clustering technique works with different similarity thresholds; the effectiveness obtained after clustering, and the scalability of the proposed technique. The application of the aforementioned technique in four industrial case studies has provided a better understanding of how to restructure the business process models that have been obtained from large and complex information systems using reverse engineering, by means of ontology-based clustering.

Contrary to the results expected, the data obtained in the case study showed that the range of similarity thresholds that provides a better performance is between 0.2 and 0.4. All the measures considered in this range, with the exception of density (i.e., size, connectivity, and separability), were greatly improved after the clustering algorithm had been applied. The proposed techniques have additionally proved to be linearly scalable to larger business process models.

Our future research lines will address alternative similarity functions with which to refine the computation of the similarity between business tasks using the ontology tree. The ontology-based clustering algorithm, which is a part of business process archeology, will also be applied in other scenarios.

## ACKNOWLEDGEMENTS

This work was supported by the R&D projects MOTERO (JCCM and FEDER, PEII11-0366-9449) and GEODAS-BC project (Ministerio de Economía y Competitividad and FondoEuropeo de Desarrollo Regional FEDER, TIN2012-37493-C03-01). Finally, this work was partially supported by the University of Bari.

## REFERENCES

1. Weske M. Business Process Management: Concepts, Languages, Architectures. Springer-Verlag Berlin Heidelberg: Leipzig, Germany, 2007; 368.
2. Heuvel W-Jvd. Aligning modern business processes and legacy systems: a component-based perspective (cooperative information systems). The MIT Press, 2006.
3. Paradauskas B, Laurikaitis A. Business knowledge extraction from legacy information systems. *Journal of Information Technology and Control*, 2006; **35**(3):214–221.
4. Pérez-Castillo R, de Guzmán IG-R, Piattini M. Business process archeology using MARBLE. *Information and Software Technology* 2011; **53**:1023–1044.
5. Guizzardi G, Halpin T. Ontological foundations for conceptual modelling. *Applied Ontology* 2008; **3**(1–2):1–12.
6. Grabar N, Hamon T, Bodenreider O. Ontologies and terminologies: continuum or dichotomy? *Applied Ontology* 2012; **7**(4):375–386.
7. Batet M. Ontology-based semantic clustering. *AI Communications* 2011; **24**(3):291–292.
8. Mirkin B. Clustering for data mining: a data recovery approach (Chapman & Hall/CRC computer science), Chapman \\& Hall/CRC, 2005.
9. Smith B. Ontology: Philosophical and Computational, 2004 [cited 2013 29/01/2013]; Available from: http://ontology.buffalo.edu/ontology(PIC).pdf.
10. Studer R, Benjamins VR, Fensel D. Knowledge engineering: principles and methods. *Data & Knowledge Engineering* 1998; **25**(1–2):161–197.
11. Noy NF, McGuinness DL. Ontology development 101: a guide to creating your first ontology, 2001 [cited 2012 01/13/2012]; Available from: http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html.
12. McGuinness DL, Van Harmelen F. OWL web ontology language overview. *W3C recommendation*, 2004; **10**:2004–03.
13. Tversky A. Features of similarity. *Psychological Review* 1977; **84**(2):327–352.
14. Cheng Y. Ontology-based fuzzy semantic clustering. In *Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology, IEEE Computer Society* 2008; **02**:128–133.
15. Deselaers T, Ferrari V. Visual and semantic similarity in ImageNet. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, IEEE Computer Society* 2011; 1777–1784.
16. Lord PW, Stevens RD, Brass A, Goble CA. Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. *Bioinformatics* 2003; **19**(10):1275–1283.
17. D'Amato C, Fanizzi N, Fazzinga B, Gottlob G, Lukasiewicz T. Ontology-based semantic search on the Web and its combination with the power of inductive reasoning. *Annals of Mathematics and Artificial Intelligence* 2012; **65**(2–3):83–121.
18. Mustapha NB, Zghal HB, Aufaure M-A, Ghezala HB. Enhancing semantic search using case-based modular ontology, in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM: Sierre, Switzerland, 2010; 1438–1439.



DOI: 10.1002/smr

19. Levenshtein V. Binary codes capable of correcting deletions, insertions, and reversals. *Journal of Cybernetics and Control Theory* 1966; **10**:707–710.
20. Cohen WW, Ravikumar P, Fienberg SE. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.
21. Melnik S, Garcia-Molina H, Rahm E. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002.
22. Rui X, II Wunsch D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on* 2005; **16**(3):645–678.
23. Han J, Kamber M. Data mining: concepts and techniques. Morgan Kaufmann, 2006.
24. Jeston J, Nelis J, Davenport T. Business process management: practical guidelines to successful implementations. 2nd ed. Butterworth-Heinemann (Elsevier Ltd.): NV, USA, 2008; 469.
25. Castellanos M, Medeiros KAd, Mendling J, Weber B, Weitjers AJMM. Business process intelligence. In Handbook of Research on Business Process Modeling, Cardoso JJ, van der Aalst WMP (eds.). Idea Group Inc.: E. Chocolate Ave. Hershey, PA 17033, USA, 2009; 456–480.
26. Lewis GA, Smith DB, Kontogiannis K. A research agenda for service-oriented architecture (SOA): maintenance and evolution of service-oriented systems, Software Engineering Institute, 2010; 40.
27. The Standish Group. Modernization. clearing a pathway to sucess. The Standish Group International, Inc, 2010.
28. van der Aalst W, Weijters AJMM. Process Mining, in Process-Aware Information Systems: Bridging People and Software Through Process Technology, Dumas M, van der Aalst W, Ter Hofstede A (eds.). John Wiley & Sons, Inc.: Hoboken, USA, 2005; 235–255.
29. Di Francescomarino C, Marchetto A, Tonella P. Reverse engineering of business processes exposed as Web applications, in 13th European Conference on Software Maintenance and Reengineering (CSMR'09). IEEE Computer Society: Fraunhofer IESE, Kaiserslautern, Germany, 2009; 139–148.
30. Cai Z, Yang X, Wang W. business process recovery for system maintenance—an empirical approach, in 25 th International Conference on Software Maintenance (ICSM'09). IEEE Computer Society: Edmonton, Alberta, Canada, 2009; 399–402.
31. Pérez-Castillo R, Weber B, García Rodríguez de Guzmán I, Piattini M. Toward obtaining event logs from legacy code. Business Process Management Workshops (BPI'10). Lecture Notes in Business Information Processing ((LNBIP 66 - Part 2)), 2010; 201–207.
32. Pérez-Castillo R, Weber B, García Rodríguez de Guzmán I, Piattini M. Generating event logs from non-process-aware systems enabling business process mining. *Enterprise Information System Journal* 2011; **5**(3):301–335.
33. Motahari-Nezhad HR, Saint-Paul R, Casati F, Benatallah B. Event correlation for process discovery from Web service interaction logs. *The VLDB Journal* 2011; **20**(3):417–444.
34. Dijkman R, Dumas M, van Dongen BF, Käärik R, Mendling J. Similarity of business process models: metrics and evaluation. *Information Systems* 2011; **36**(2):498–516.
35. Hliaoutakis A, Varelas G, Voutsakis E, Petrakis EGM, Milios E. Information Retrieval by Semantic Similarity. IGI Global: E. Chocolate Ave. Hershey, PA 17033, USA, 2006; 55–73.
36. Runeson P, Höst M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 2009; **14**(2):131–164.
37. Pérez-Castillo R. Experiment results about clustering techniques for reducing noise of business process mining, 2012 [cited 2012 23/01/2012]; Available from: http://alarcos.esi.uclm.es/per/rpdelcastillo/Experiments.html#clustering.
38. Pérez-Castillo R, Fernández-Ropero M, García Rodríguez de Guzmán I, Piattini M. MARBLE. A business process archeology tool, in 27th IEEE International Conference On Software Maintenance (ICSM'11). IEEE Computer Society: Williamsburg, Virginia, USA, 2011; 578–581.
39. Ekeland I, Temam R. Convex analysis and 9 variational problems, 1976.
40. Pérez-Castillo R, Sánchez-González L, García F, Piattini M, García Rodríguez de Guzmán I. Obtaining thresholds for the effectiveness of business process mining. In *5th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'11)*. IEEE Computer Society: Banff, Alberta, Canada, 2011; 453–462.
41. Sánchez González L, García F, Ruiz F, Mendling J. A study of the effectiveness of two threshold definition techniques. In *Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on*, 2012.

## AUTHORS' BIOGRAPHIES

**Ricardo Pérez-Castillo** is an IT professional at Itestra GmbH. He holds a PhD degree in Computer Science from the University of Castilla-La Mancha. His research interests include architecture-driven modernization, model-driven development, and business process archeology.



DOI: 10.1002/smr

**Danilo Caivano** received a degree with honors in Informatics at the University of Bari, Italy. After graduating, he started his PhD degree in software engineering at the Department of Informatics in the University of Bari, working within the Software Engineering Research Laboratory (SER_Lab). His research interests are mainly focused on software process improvement and software maintenance. Currently, he is collaborating on several research projects and carries out controlled and in-field experimentation within small and medium enterprises. He is a member of the IEEE Computer Society.

**Mario Piattini** is a full professor at the UCLM. His research interests include software quality, metrics, and maintenance. He holds a PhD degree in Computer Science from the Technical University of Madrid and leads the Alarcos Research Group at the Universidad de Castilla-La Mancha. He has received certifications in CISA, CISM, and CGEIT by ISACA.



DOI: 10.1002/smr