Towards Retrieving and Recommending Security Annotations for Business Process Models Using an Ontology-based Data Matching Strategy

Ioana Ciuciu, Yan Tang, and Robert Meersman

Semantics Technology and Applications Research Laboratory, Vrije Universiteit Brussel, B-1050 Brussels, Belgium
{iciuciu, yan.tang, meersman}@vub.ac.be

Abstract. In the Trusted Architecture for Securely Shared Services (TAS3) EC FP7 project we have developed a method to provide semantic support to the process modeler during the design of secure business process models. Its supporting tool, called Knowledge Annotator (KA), is using ontology-based data matching algorithms and strategy in order to infer the recommendations best fitted to the user design intent, from a dedicated knowledge base. The paper illustrates how the strategy is used to perform the similarity (matching) check in order to retrieve the best design recommendation. We select the security and privacy domain for trust policy specification for the concept illustration.

Keywords: ontology-based data matching, ontology, semantic annotation, knowledge retrieval, security constraints, security policy, business process model design.

1 Introduction and Motivation

The Knowledge Annotator (KA) tool has been designed to support process modelers in designing security-annotated business process models (BPM). This work has been done in the context of the EC FP7 Trusted Architecture for Securely Shared Services (TAS3) project, whose aim is to provide a next generation trust and security architecture for the exchange and processing of sensitive personal data. The TAS3 architecture meets the requirements of complex and highly versatile business processes while enabling the dynamic, user-centric management of policies.

One of the challenges is to offer a secure business processes framework for sharing, accessing, and using personal data processing services in federated environments.

In order to make business processes secure, the business process model is annotated with security constraints which apply to authentication, authorization, audit logging, and other security issues. The business process management system (BPMS) transforms the security annotations into descriptive security policies or triggers.

1 http://www.tas3.eu/
process model extensions. It finally executes secure business processes by dedicated
system components (e.g. these components allocate actors to activities, enforce data-
specific authorizations, or trigger security-specific user involvements). This
infrastructure guarantees that business processes is performed according to the
annotated security constraints. In order to ensure semantic interoperability between
the different components of the system, we provide a security ontology which
explicitly documents the relationship between core security concepts. One goal is to
annotate all security-relevant business process specifications with a common, agreed
upon conceptualization (ontology) in order to ensure alignment and interoperability
between different actors with respect to security concepts.

The KA tool ensures the correct specification of the security annotations, by
supporting the process modeler with syntactically correct security concepts and with
annotation recommendations. The recommendations are obtained by matching the
knowledge stored in a knowledge base and an ontology of security constraints against
the process modeler request, specifying his/her design intent.

The paper focuses on the matching process for the specification of security
annotations for security (trust) policies, applied to business process models.

The rest of the paper is organized as follows: Section 2 provides background
information on the ontology-based data matching strategy. Section 3 illustrates an
approach to applying ontology-based data matching for knowledge retrieval for
annotating secure BPM. Section 4 presents the results. The related work of the paper
is discussed in Section 5. We conclude the presented work and propose our future
work in Section 6.

2 Background

In this study we applied ontology-based data matching algorithms and a strategy in
order to compute the similarity between the user request and the security-related
knowledge represented by security constraints. The Ontology-based Data Matching
Framework (ODMF [1]) has been introduced in the EC FP6 Prolix2 project for the
purpose of competency matching. Ontology-based data matching (ODM) is a new
discipline of data matching and is ontology-based. The goal of ODM is to find the
similarities between two data sets, each of which corresponds to (or can be annotated
with) one part of the ontology. There is one ontology in the particular problem,
represented as a graph. This approach brings a new precision level thanks to the
community (indirect) support.

The selected strategy for this research is the Controlled Fully Automated Ontology
Based Data Matching Strategy (C-FOAM). C-FOAM is a hybrid strategy of ODMF,
combining (1) string matching algorithms; (2) lexical matching algorithms and (3) at
least one graph-based matching algorithm.

2 http://www.prolixproject.org/
C-FOAM

The C-FOAM strategy starts with combining two character strings representing context-object pairs. If the two character strings of the contexts are the same, data objects that belong to the same object type will be compared. To resolve the actual data objects stored in the ontology, the combination of both the context term and the object term is used.

In case the data object is denoted by several terms a lexical lookup is done taking synonyms into account. If a given object term could not be found in the ontology and lexicon, the best fitting data object is returned from the ontology using fuzzy matching based on string similarity (e.g. using JaroWinklerTFIDF algorithm [2,3]). The similarities between the given data object and the most similar data object in the ontology should be above a certain threshold, which is set in the application configuration of our tool. If the data object is found based on fuzzy matching then a penalty percentage will be used on the confidence value for the matching score.

C-FOAM is based on two main modules: (1) the Interpreter module and (2) the Comparator module, as shown in Fig. 1.

![Fig. 1. C-FOAM model for ontology-based data matching.](image)

The Interpreter module makes use of the lexical dictionary, WordNet\(^3\), the domain ontology and string matching algorithms to interpret end users’ input. Given a term that denotes either (a) a concept in the domain ontology, or (b) an instance in the ontology, the interpreter will return the correct concept(s) defined in the ontology or lexical dictionary, and an annotation set of the concept.

The Comparator computes the similarity between two found data objects annotated with binary facts from the ontology base. A graph based algorithm or a combination of different graph-based algorithms (e.g. OntoGram, LeMaSt developed within ODMF) is used by the comparator to find the similarities between the two annotation sets. In case more than one graph algorithm is used, a positive percentage must be specified for each of them in the configuration file.

The ontology is modeled following the Developing Ontology Grounded Methodology and Applications (DOGMA [4]) framework. In DOGMA, the ontology is two-layered in order to make the reuse of facts easier. It is separated into 1) a lexon base layer (binary facts represented in semi-natural language) and 2) a commitment layer (defining constraints on the committing lexons). An example of lexon is 
\[ <\gamma, Security\ Annotation, has, is\ of, Parameter >\] which represents a fact that “within the context identified by \(\gamma\), a Security Annotation has a Parameter and a

\(^3\) [http://wordnet.princeton.edu/](http://wordnet.princeton.edu/)
Parameter is of a Security Annotation”. A commitment on this lexon can be, i.e. each Security Annotation has at least one Parameter, which is a mandatory constraint.

The security annotations built on top of the BPM are semantically annotated with lexons from a dedicated security constraints ontology. The binary facts are disambiguated via a context handler.

3 Approach

The knowledge annotator is designed as a user-friendly system, intended to assist the process modeler during the specification of the security-specific constraints and to learn from the process modeler by using a dedicated knowledge base. This is realized by capturing the process modelers’ modeling intentions via a user-friendly interface (UI) and by presenting him/her with recommendations (as shown in Fig. 2). The recommendations are determined by an ontology-based data matching operation between the user input, the security constraints ontology, user-defined dictionaries, Synsets from lexical databases (e.g. WordNet) and the collected security annotations retrieved from the knowledge base. This operation represents the focus of this paper and will be detailed in the following sections.

![Fig. 2. User-system interactions for the annotation of security constraints.](image)

3.1 The Knowledge Annotator, Recommender

The KA encapsulates several functions in a web service, which are supported by six architectural components: (1) the capturer – for capturing the user design intent; (2) the annotator – for annotating objects with concepts from the security ontology; (3) the indexer – for indexing elements for efficient retrieval; (4) the retriever – for retrieving information; (5) the comparator – for comparing user input with the knowledge base; and (6) the presenter – for presenting the user with recommendations and for user query specification. We refer to [5] for details regarding the overall architecture and functionality of the KA.
In order to understand the basic data element used by the KA to retrieve recommendations, we first give the definition of a security annotation.

**Security Annotation.** A security annotation is a text annotation attached to a BPMN element. The syntax of a security annotation is specified by an annotation term, followed by a list of parameters (mandatory or optional) with their corresponding values:

```
<<AnnotationTerm: list(parameter="value")>>.
```

Currently, our security language supports auditing, authentication, authorization, data and message flow security, delegation, and user interactions [6].

The concept of security annotation, represented in DOGMA and modeled with Collibra Business Semantics Glossary⁴, is illustrated in Fig. 3. A security annotation, specified as above, is applied to one or more BPMN elements.

![Fig. 3. Representation of the security annotation concept (Collibra Business Semantics Glossary screenshot).](image)

**Basic data object of the KA.** According to the above definition, the basic data object used by the knowledge annotator is represented by the Security Annotation Term – Element – Parameter – Value (STEPV) object. The STEPV object encapsulates the five entities needed to completely define a security annotation: the security constraint, the BPMN element being annotated, the parameters and their corresponding values. In case the value of the parameters is ignored, the object will be referred to as STEP object.

When the process modeler wants to define a security annotation, he/she fills in the fields corresponding to his/her design intent and knowledge (STEPV elements). This represents the input to the web service call. The system captures the input and analyzes it against the ontology base and the knowledge base in order to make the best fitted recommendations to the process modeler. The STEPV elements returned are considered valid annotations according to the constraints defined in the commitment layer of the ontology.

This study focuses on components (4) and (5) of the KA:

(4) The retriever component retrieves similar fragments from the knowledge base (e.g., all existing security annotations which share at least one common element with the input object). The similarity measure can be defined according to the user needs. For example, the user could only be interested in STEPV objects with a particular value for the “name” parameter. The knowledge base contains semantic annotation instances (STEPVA objects) of the security constraints.

(5) The comparator component performs a matching operation in order to compare the process modeler’s demand (input object) with the resulted elements retrieved from the knowledge base in the previous step.

3.2 C-FOAM for Recommendations

The advanced C-FOAM matching strategy was applied in order to match the user defined search patterns with the knowledge existing in the data repositories of the system (the security constraints ontology, the security upper common ontology, the knowledge base, WordNet Synsets, user-defined dictionaries, etc.). As previously explained, C-FOAM makes use of string matching, lexical matching and graph-based matching to calculate the similarity score. The following paragraphs illustrate the three matching techniques performed by C-FOAM when retrieving the recommended security annotations.

String Matching. Various string matching algorithms (SecondString and ODMF-specific), are used to calculate the similarity of two objects belonging to the same super-ordinate concept based on the string similarity of their description. For example, two security annotations descriptions ‘Set Trust Policy name’ and ‘Set Trust Policy type’ can be compared using different string matching algorithms. An ODMF-specific string matching algorithm based on fuzzy matching, ODMF.JaroWinklerTFIDF, can be used to compensate for the user typing errors (e.g. when the user types in ‘Trsut’ the algorithm can be used to find ‘Trust’).

Matching at string level is easy to implement and does not require a complex knowledge resource. A natural language description of the security annotation and/or its composing elements is sufficient.

The Interpreter component in C-FOAM uses string matching, in particular the JaroWinklerTFIDF algorithm.

Lexical Matching. The lexical matching algorithms calculate the similarity of two objects that belong to the same super-ordinate concept based on the semantic similarity of their descriptions. The object descriptions should be terminologically annotated. For example, the security annotation description ‘Set Trust Policy name’ could be annotated with the terms ‘set’, ‘trust’, ‘policy’, and ‘name’. This set of terms may be then compared to a second set of terms to calculate the similarity between the two semantic descriptions, using the Jaccard similarity coefficient:

\[
\text{Jaccard}(\text{Set1}, \text{Set2}) = \frac{|\text{Set1} \cap \text{Set2}|}{|\text{Set1} \cup \text{Set2}|}.
\]
In addition to plain string matching techniques, linguistic information is used to improve the matching. Two techniques are used for improving the matching:

1. **Tokenization and lemmatization.** Tokenization is the process of identifying the tokens, i.e., words and punctuation symbols, in a character string. Lemmatization is the process of determining the lemma of a given word. A lemma is the base form of a word as it appears in the index of a dictionary or a terminological database.

2. **An ontologically structured terminological database.** In the form of a categorization framework, such a database is used in order to take into account synonyms and/or translation equivalents of a given term. Hyponyms or hypernyms may be used to take into account more generic or more specific terms of a given term. C-FOAM makes use of the concepts, concept relations, and terms based on WordNet and automatically tokenizes and lemmatizes the description of the security annotations.

Examples of lexical matching based on synonyms and user dictionaries are illustrated below in Table 1:

<table>
<thead>
<tr>
<th>Synonyms</th>
<th>Concept in the ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>'user'</td>
<td>'requestToUser'</td>
</tr>
<tr>
<td>'user interaction'</td>
<td></td>
</tr>
<tr>
<td>'interaction'</td>
<td></td>
</tr>
<tr>
<td>'select service'</td>
<td>'service selection'</td>
</tr>
<tr>
<td>'choice of service'</td>
<td></td>
</tr>
<tr>
<td>'choice'</td>
<td></td>
</tr>
<tr>
<td>'task'</td>
<td>'activity'</td>
</tr>
<tr>
<td>'subprocess'</td>
<td></td>
</tr>
<tr>
<td>'flow element'</td>
<td></td>
</tr>
<tr>
<td>'trigger'</td>
<td>'event'</td>
</tr>
<tr>
<td>'message'</td>
<td></td>
</tr>
</tbody>
</table>

**Graph Matching.** The graph matching algorithms are used to calculate the similarity between two objects that represents two sub-graphs. We can as well use this similarity score to find related objects for a given object. The similarity of two objects is calculated based on their classification, properties and semantic relations. For example, the comparator module of the KA computes the similarity between two security annotations. In the same way, using classification information of objects, the relations between objects and the properties of objects, it is possible to find related security annotations for a given security annotation. For example, if the user wants to find relevant BPMN elements applying to the security annotation type ‘Delegation’.

The graph is a semantic graph (ontology), in which concepts in the ontology are represented as vertices and semantic relations between concepts are represented as arcs. The arcs are bi-directed and correspond to the role and co-role in a binary fact type (lexon).

For this technique, a domain ontology and an application ontology must be available. In our case, the domain ontology is the security concepts ontology and the application ontology is the ontology of security constraints. Both ontologies act as the
model for the rule-based reasoning which applies forward chaining to infer new knowledge, based on the existing knowledge expressed in the knowledge base.

For the purpose of secure business process models design, we take as reference of comparison the security annotation (the STEPV object), with its corresponding components, as described above. In order to find a correct security annotation, the system compares all the elements specified by the user (completely or partially) for the desired security annotation with the existing elements in the ontology and the knowledge base and retrieves a set of related security annotations, ranked according to the calculated similarity score. An example will be given in the next section.

Lexon Matching Strategy (LeMaSt) is applied in the Comparator module of C-FOAM in order to compute the similarity between the user input and the knowledge in the knowledge base and in the ontology. LeMaSt calculates the similarity of two security annotations based on the semantic similarity of their descriptions represented as a set of lexons. Therefore, this techniques demands that the two object descriptions are annotated with lexons (elementary facts that are accepted to be true within the context of that object). For example, the security annotation description 

\[
\text{<< Set Trust Policy name="$name" type="$type" insertplace(*)="$insertplace" role(*)="$role" >>}
\]

can be annotated with the lexons illustrated in Table 2:

<table>
<thead>
<tr>
<th>Context</th>
<th>Head</th>
<th>Role</th>
<th>Co-role</th>
<th>Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetTrustPolicy</td>
<td>SetTrustPolicy</td>
<td>has parameter</td>
<td>parameter_of</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>SetTrustPolicy</td>
<td>has parameter</td>
<td>parameter_of</td>
<td>type</td>
</tr>
<tr>
<td></td>
<td>SetTrustPolicy</td>
<td>has optional parameter</td>
<td>Optional parameter of</td>
<td>insertplace</td>
</tr>
<tr>
<td></td>
<td>SetTrustPolicy</td>
<td>has optional parameter</td>
<td>Optional parameter of</td>
<td>role</td>
</tr>
</tbody>
</table>

To calculate the similarity of the two security annotations (STEPV objects) that are annotated with lexons, we calculate the similarity of their corresponding lexon sets. For this purpose we extended the Jaccard similarity coefficient (see Equation 1) to account for partial overlap of two lexons:

\[
C = \sum_{i=1}^{n} x_i / n, \quad 0 \leq x_i \leq 1
\]

\[
S = \frac{1}{\sum_{i=1}^{n} C_i \times S_i}, \quad 0 \leq S_i \leq 1
\]

(2)

The Jaccard similarity scores are used to calculate the contribution score C using Equation 2. The final score S is the average scores of the lexons. For each lexon, \( x_i \) is a contribution score depending on the matching items. For example, two lexons that have the same head, role, co-role and tail term contribute 100% to the result (\( x_i = 1 \)). If the lexons have the same head and tail term but different role and co-role they contribute for 50% (\( x_i = 0.5 \)). If the lexons have the same head or tail term and the
same role and co-role they contribute for 50% ($x_{\text{ratio}}=0.5$). If the lexons only have the same head or tail term they contribute for 25% ($x_{\text{ratio}}=0.25$).

4 Results and Analysis

We take the ‘Set trust policy’ annotation to illustrate the matching process. Via the UI, the process modeler can specify a desired STEPV object by indicating parts of it he/she knows. For this specific example, the user inputs his/her choices for the Security Annotation and the BPMNElement fields. The STEPV element is therefore specified only by two fields out of five. The value parameter is excluded from this example, for simplification. In this case, we are dealing with STEP objects. The system will interpret the user input by performing matching at different levels and will infer the correct most similar annotation. The user input and the system result are illustrated in Fig. 4.

![Fig. 4. User query for retrieving the Select Trust Policy security annotation and result.](image)

Let us now analyze how these results were inferred by the system. The process starts with the user specifying fields of the STEP object. He/she indicates ‘Trust policy’ and ‘trigger, task’. These values are resolved by performing matching at different levels. In case of ‘Trust policy’, C-FOAM performs a matching operation at string level by applying Jaro-Winkler and finds the correct string ‘Set Trust Policy’ in the ontology. In case of ‘trigger’ and ‘task’, they are matched to the correct concepts ‘event’ and ‘activity’ respectively in the ontology, by performing matching at lexical level, using WordNet and a user-defined dictionary to resolve the synonymy.

Once the correct concepts in the ontology are found with their corresponding annotation sets (lexons), C-FOAM starts performing a matching step at graph (ontology) level, by applying LeMaSt. LeMaSt compares the annotation set
corresponding to the user input with the ontology of security constraints and infers the most similar annotation:

```
<< Set Trust Policy name="$name" type="$type" insertplace(*)="$insertplace" role(*)="$role" >>
```

This annotation is presented to the user as a recommendation. The user can further refine his/her search by modifying fields of the retrieved STEP object.

## 5 Related Work

Many ontology matching approaches exist in ontology engineering (OE) [7]. Several EU projects, such as Knowledge Web [8] or OpenKnowledge [9] invested a lot of effort into the creation of algorithms and tools for ontology matching/integration.

Our approach is different from ontology matching or ontology integration and can be considered as a subdomain of data matching, based on ontology. In our problem setting, there exists only one ontology. ODMF finds similarities between two data sets, each of which corresponds to one part of the ontology. Classical methods, such as using linguistic methods for concept searching, using WordNet as the external dictionary and applying graph matching principles are included in our work. ODMF builds upon these techniques while designing and implementing an innovative generic matching framework.

Regarding the recommender systems used in business process modeling, our approach uses ODMF for security annotations retrieval expressed in natural language in order to support the business modeler.

Betz [10] proposes an approach for the automatic user support based on an autocompletion mechanism during the modeling process (where business processes are represented as Petri Nets). Born [11] presents a tool for the user-friendly integration of domain ontology information in the process modeling, through matchmatching and filtering techniques. A similar approach based on linguistic analysis of process element labels and of the concept names is presented in [12] in order to support process modelers with annotation suggestions.

## 6 Conclusion and Future Work

This paper is focusing on the matching strategies used by an annotation system while retrieving recommendations to assist the business process modeler into designing secure business processes. The matching strategy is ontology-based and belongs to ODMF methodology for ontology-based data matching. The applied strategy benefits from all its composing algorithms applied at string, lexical and graph level in an iterative refinement process and from the user-system interactions.

A future work is to evaluate the results with the ODMF evaluation benchmark. Since C-FOAM is a hybrid strategy combining possibly different algorithms for different applications, it is important to submit the results to human experts for analysis in view of selecting the best matching algorithms. The enrichment of the
knowledge base with value ranges for the parameters and a mechanism for checking the correctness of the specified parameter values is work in progress.

Acknowledgments. This research is supported by the EC FP7 TAS$^3$ (Trusted Architecture for Securely Shared Services) project. The authors would like to thank all TAS$^3$ project partners for their contribution to the research.

References