Towards Evaluating an Ontology-based Data Matching Strategy for Retrieval and Recommendation of Security Annotations for Business Process Models

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 (TAS^3) 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 the 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. Finally, the paper discusses the evaluation of the results using the Ontology-based Data Matching Framework evaluation benchmark.

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

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[1] (TAS^3) project, whose aim is to provide a next generation trust and security architecture for the exchange and processing of sensitive personal data. The TAS^3 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)

[1] http://www.tas3.eu/

transforms the security annotations into descriptive security policies or triggers 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 are performed according to the annotated security constraints. In order to ensure interoperability between the different actors and components of the system, we provide a security ontology which explicitly documents the relationship between the core security concepts. One goal is to annotate all security-relevant business process specifications with a common, agreed upon conceptualization (ontology).

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's request, specifying his design intent.

The paper focuses on the evaluation of the matching process for the specification of security annotations for security (trust) policies, applied to business process models. The evaluation is done with the Ontology-based Data Matching Framework (ODMF [1]) evaluation benchmark.

The rest of the paper is organized as follows: Section 2 provides background information on the ontology-based data matching strategy. Section 3 proposes an approach to applying ontology-based data matching for knowledge retrieval for annotating secure BPM. Section 4 presents the matching results and their interpretation. An evaluation methodology and the evaluation results are shown in Section 5. The related work of the paper is discussed in Section 6. We conclude the presented work and propose our future work in Section 7.

The main contribution of the paper therefore consists of (1) mining the user knowledge using natural language and an ontology-based data matching strategy; (2) retrieving similar patterns from the Knowledge and Ontology Base and proposing them to the user; and (3) evaluating the results using a generic evaluation benchmark.

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) has been introduced in the EC FP6 Prolix[2] project for the purpose of competency matching. Ontology-based data matching (ODM [1,2]) 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 are annotated with one ontology and corresponds to 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.

[2] http://www.prolixproject.org/

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. C-FOAM is described in the following paragraphs.

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 [3,4]). 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.

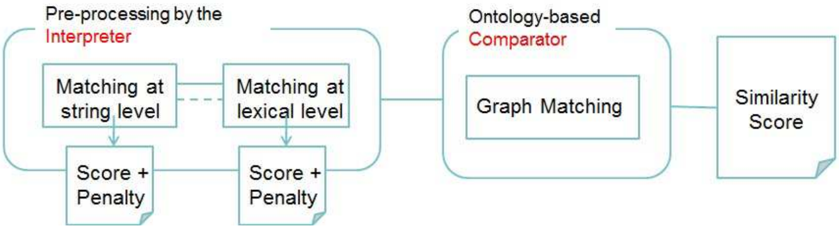


Fig. 1. C-FOAM model for ontology-based data matching.

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.

[3] http://wordnet.princeton.edu/

The *ontology* is modeled following the Developing Ontology Grounded Methodology and Applications (DOGMA [5]) 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 γ, a Security Annotation has a Parameter and a 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 dictionary, Synsets from lexical databases (e.g. WordNet) and the collected security annotations retrieved from the knowledge base. This operation and the evaluation of the outcome represent the focus of this paper and will be detailed in the following sections.

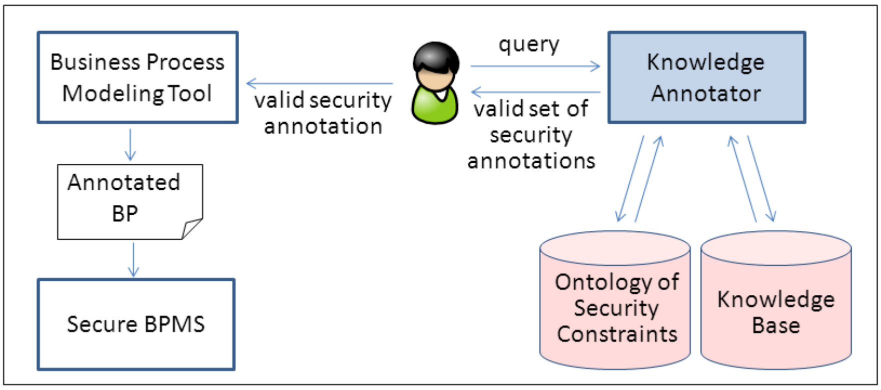

Business Process Modeling Tool
valid security annotation
query
Knowledge Annotator
valid set of security annotations
Annotated BP
Secure BPMS
Ontology of Security Constraints
Knowledge Base

Fig. 2. User-system interactions for the annotation of security constraints.

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 the user input with the knowledge base; and (6) the presenter – for presenting the user with recommendations and for user query specification. We refer to [6] 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' [7].

The concept of security annotation, represented in DOGMA and modeled with Collibra Business Semantics Glossary[4], is illustrated in Fig. 3. A security annotation, specified as above, is applied to one or more BPMN elements.

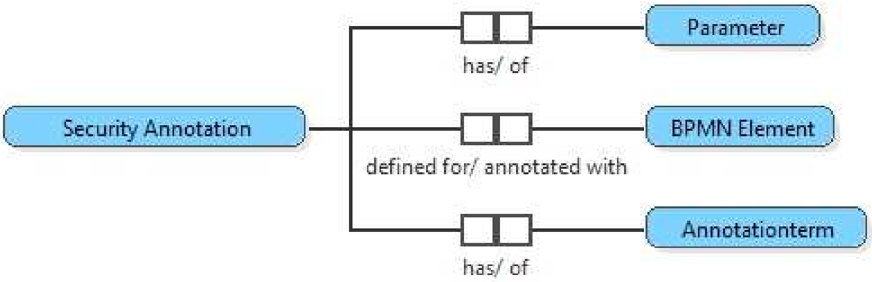


Fig. 3. Representation of the security annotation concept (Collibra Business Semantics Glossary screenshot).

Basic Data Object of the KA. According to the above definition, the basic data object used by the knowledge annotator is represented as a 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.

[4] http://www.collibra.com/products-and-solutions/products/business-semantics-glossary

When the process modeler wants to define a security annotation, he fills in the fields corresponding to his 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 (STEPV-Annotated 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:

$$\mathrm{Jaccard}(\mathrm{Set1}, \mathrm{Set2}) = |\mathrm{Set1} \cap \mathrm{Set2}| / |\mathrm{Set1} \cup \mathrm{Set2}| \,. \quad (1)$$

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. Hypernyms or hyponyms 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 1. Lexical synonyms defined for concepts in the ontology.

Synonyms	Concept in the ontology
‘user’ ‘user interaction’ ‘interaction’	‘requestToUser’
‘select service’ ‘choice of service’ ‘choice’	‘service selection’
‘task’ ‘subprocess’ ‘flow element’	‘activity’
‘trigger’ ‘message’	‘event’

Graph Matching. The graph matching algorithms are used to calculate the similarity between two objects that represent two sub-graphs of the same (ontology) graph. The similarity score can be used also to find related objects for a given object, as it is the case in this paper. 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 by 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 technique 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 << Set Trust Policy name="$name" type="$type" insertplace(*) ="$insertplace" role(*)="$role" >> can be annotated with the lexons illustrated in Table 2:

Table 2. Security annotation 'SetTrustPolicy' modeled with DOGMA.

Context	Head	Role	Co-role	Tail
SetTrustPolicy	SetTrustPolicy	has_parameter	parameter_of	name
SetTrustPolicy	SetTrustPolicy	has_parameter	parameter_of	type
SetTrustPolicy	SetTrustPolicy	has_optional_parameter	optional_parameter_of	insertplace
SetTrustPolicy	SetTrustPolicy	has_optional_parameter	optional_parameter_of	role

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:

$$\mathrm{C}=\sum_{i=1}^{n} x_i/n\,, \quad 0 \le x_i \le 1$$

$$S=\sum_{i=1}^{n} C_i \times S_i\,, \quad 0 \le S_i \le 1 \qquad (2)$$

$$\sum_{i=1}^{n} C_i = 1\,.$$

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_i=0.5). If the lexons only have the same head or tail term they contribute for 25% (x_i=0.25).

4 Results and Analysis

An annotation scenario was created in order to analyze the behavior of the KA tool. The scenario consists of three user requests (queries) composed of different elements of security annotations: security annotation name, annotation term, parameter name and BPMN element. Depending on the accuracy of the user input, the matching is performed at string, lexical or graph level. The test data (user input, KA recommendations, matching scores and justification) is given in Table 3.

Table 3. The results of the annotation scenario.

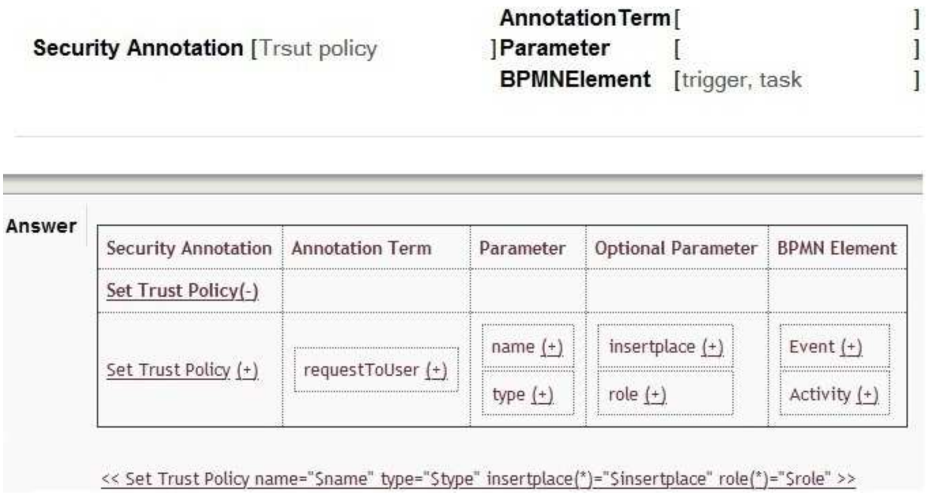
User Input				KA Recommendation SA<<AT: list (param = value)>>	Score	Remarks
SA	AT	P	BPMN E			
‘Trsut policy’ (user input text)	-	-	Trigger , task	Set Trust Policy <<request ToUser name="$name" type="$type" insertplace(*)="$insertplace" role(*)="$role">>	0.88	[TYPO+ SYNONYM+ ONTOLOGY] ‘Set Trust Policy’ ‘Activity’ Lexon term String matching Lexical matching Graph matching
-	Interaction	-	Task	User Consent <<requestToUser type="Consent" insertplace(*)="$insertplace" target="$target" display="$display" role(*)="$role">> User Assignment <<Assignment type="user" user="$user">>	0.52 0.13	[SYNONYM+ ONTOLOGY] ‘Request to User’ ‘Activity’ Lexon term Lexical matching Graph matching
‘Selecr service’	-	-	Trigger	Service Selection <<requestToUser	0.8	[TYPO+ SYNONYM+ ONTOLOGY]

(user input text)				type="SelectService" insertplace(*)="$insertplace" display(*)="$display" role(*)="$role">>		‘Service selection’ ‘Activity’ Lexon term String matching Lexical matching Graph matching
				Select Data Policy <<requestToUser type="SelectDataPolicy" role(*)="$role" target="$target" insertplace(*)="$insertplace" display="$display">>	0.5	
				Select Trust Policy <<requestToUser type="SetTrustPolicy" policy="$policy" role(*)="$role" insertplace(*)="$insertplace" >>	0.45	

Let us 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 the parts that are known to them. For this specific example, the user input concerns 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 interprets the user input by performing matching operations at different levels and infers the correct most similar annotation. The user input and the system result are illustrated in Fig. 4.

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 indicates ‘Trsut policy’ and ‘trigger, task’. These values are resolved by performing matching at different levels. In case of ‘Trsut policy’, C-FOAM performs a matching operation at string level by applying JaroWinkler 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 (see Table 1) to resolve the synonymy.

Once the correct concepts in the ontology are found together 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, which in this case is the following:



Security Annotation [Trsut policy]

AnnotationTerm [ ]

Parameter [ ]

BPMNElement [trigger, task]

Answer

Security Annotation	Annotation Term	Parameter	Optional Parameter	BPMN Element
Set Trust Policy(-)				
Set Trust Policy (+)	requestToUser (+)	name (+) type (+)	insertplace (+) role (+)	Event (+) Activity (+)

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

Fig. 4. User query for retrieving the 'Select Trust Policy' security annotation and the result.

<< 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 search by modifying fields of the retrieved STEP object.

5 Evaluation Results

The Knowledge Annotator tool is evaluated based on a generic evaluation method [1] which adapts the principles in the methodologies for program evaluation [8] and purpose-oriented evaluation [9].

According to the basic principle of the program evaluation methodologies, the evaluation methodology needs to help the KA tool to enhance its functions and/or services and also help to ensure that it is delivering the correct list of recommended annotations.

According to the principles in the purpose-oriented evaluation methodologies:

- The evaluation process must be able to determine what information exists in the process of the KA tool, which is important so that engineers can analyze the processing information;
- The evaluation process must test and collect continuous feedback in order to revise the process;
- The evaluation must have a precondition analysis and a post-condition analysis of the evaluated system;
- End users must be able to judge the outcome of a system based on the evaluation methodology.

Accordingly, we have developed an evaluation method that can be used to evaluate any of the matching strategies in ODMF. Its process is described in Fig. 5.

Design a generic use case step. The step of designing a generic use case is the preparation step. In this step the problem is scoped and clear requirements are initialized for a viable use case. Terms and tests are gathered and analyzed from the test beds' materials. The output of this step is a report describing a generic use case.

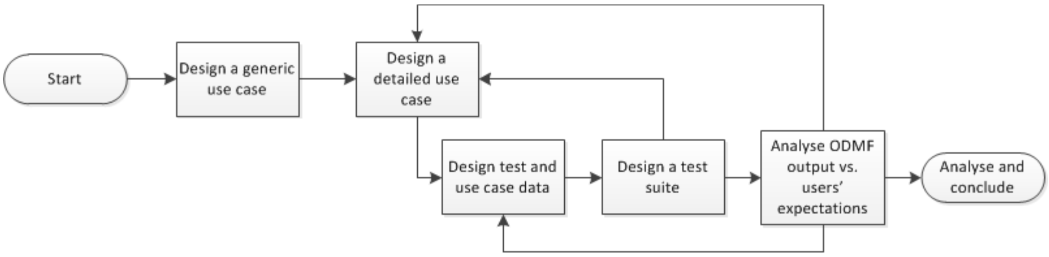

Start
Design a generic use case
Design a detailed use case
Design test and use case data
Design a test suite
Analyse ODMF output vs. users' expectations
Analyse and conclude

Fig. 5. A generic ODMF matching strategy evaluation method.

Design a detailed use case step. Here the problem is specified. The design terms from the previous step are designed by specifying types of information used by the KA tool (e.g. process information). We also analyze preconditions and post-conditions of the use case. The output of this step is a report containing a detailed use case.

Design test and test and evaluation data step. In this step we design the test data that are used by the KA tool (not by end users). The output of this step is a report containing a list of test and evaluation data.

Design test suite. In this step a knowledge engineer designs a user test suite, the data of which need to be provided by an expert. The test suite is designed based on the results from the first two steps.

Analyze ODMF output vs. users' (experts') expectations step. The output of this step is a report of comparison (KA tool's similarity scores vs. expert expected similarity scores).

Analyze and conclude step. This step is to analyze the comparison report that is produced in the previous step and draw valuable conclusions. The output of this step is a report of comparison analysis and conclusion.

We focus on the last two steps in order to evaluate the results of the KA against experts' expectations. For this, we use the satisfactory rate. The satisfactory rate is calculated based on the similarity scores generated by the KA and the experts' expected relevance levels in the test suite. The relevance levels provided by the experts need to be correctly interpreted in order to calculate the satisfactory rate. The average score provided by the KA is 0.5, the maximum score is 1 and the minimum score is 0. Therefore, the scale of the similarity scores is [0, 0.2]. We equally split is as shown below:

- Relevance level 5: similarity score > 0.8;
- Relevance level 4: similarity score > 0.6 and <= 0.8;
- Relevance level 3: similarity score > 0.4 and <= 0.6;
- Relevance level 2: similarity score > 0.2 and <= 0.4;
- Relevance level 1: similarity score <= 0.2.

If a similarity score falls in the range, then we say that the similarity score is "completely satisfied". If it does not fall in the range, then we need to calculate the bias. The bias of the evaluation set is calculated as the minimum value of low boundary bias and high boundary bias, which are calculated as shown below:

Pseudo-code for Computing the Bias of the Evaluation Set

```
IF (Similarity Score < Low Boundary)
    THEN Low Boundary Bias = Low Boundary - Similarity Score
    ELSE Low Boundary Bias = Similarity Score - Low Boundary
IF (Similarity Score < High Boundary)
    THEN High Boundary Bias = High Boundary - Similarity Score
    ELSE High Boundary Bias = Similarity Score - High Boundary.
```

For instance, if the similarity score for relevance level 4 is 0.61, then the low boundary bias is 0.61 – 0.6 = 0.01and the high boundary bias is 0.8 – 0.61 = 0.19. The bias is 0.01 (the smallest value in {0.01, 0.19}).

If the bias is less than 0.2 (one interval), then we say that this similarity score is "satisfied". If it is more than 0.2 and less than 0.4 (two intervals), then we say it is "not really satisfied". All the remaining scores are considered "completely unsatisfied".

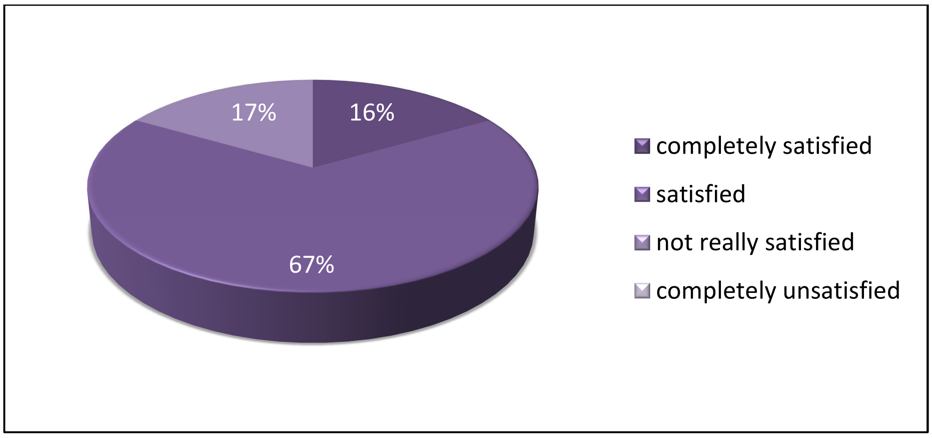


Fig. 6. C-FOAM similarity scores vs. expert expected scores.

For the particular case of this paper, the results of the comparison between the KA similarity scores and the experts' expected scores (see Fig. 6) are as follows: out of six recommended annotations, one similarity score completely satisfies the experts' expectations; four similarity scores satisfy the experts' expectations; only one score does not really satisfy the experts' expectation; there are zero completely unsatisfied similarity scores.

The satisfactory rate is 83%: 16% completely satisfied + 67% satisfied.

The results of the evaluation are recorded in Table 4.

Table 4. Conclusion of evaluating the Knowledge Annotator.

Knowledge Annotator using C-FOAM
1. Difficulty of managing the required knowledge resource.
Usage level basic to professional. Knowledge engineers need to know how to configure the penalty scores and understand the meaning of these scores.
2. Difficulty of using the matching strategy
C-FOAM is the most difficult of all the ODMF strategies because it is the composition of matching algorithms and/or other matching strategies (e.g. LeMaSt & JaroWinkler).
3. Satisfactory rate
Satisfactory rate is 83% (completely satisfied and satisfied, see Fig. 6**)**.
What affects the matching score
Any factors that affect the selected algorithms are counted as the factors that affect the similarity scores. In addition, the two penalty values are the factors that affect the final similarity scores.
Advantage
The advantages of C-FOAM contain the advantages of all the combined algorithms and strategies.

6 Related Work

Many ontology matching approaches exist in ontology engineering (OE) [10]. Several EU projects, such as Knowledge Web [11] or OpenKnowledge [12] invested 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. This is one of the contributions of this paper.

Betz [13] proposes an approach for the automatic user support based on an auto completion mechanism during the modeling process (where business processes are represented as Petri Nets). Born [14] presents a tool for the user-friendly integration of domain ontology information in the process modeling, through match matching and filtering techniques. A similar approach based on linguistic analysis of process element labels and of the concept names is presented in [15] in order to support process modelers with annotation suggestions.

In this paper, we have illustrated how to use the generic evaluation method [1], which adapts the principles in the methodologies for program evaluation [8] and purpose-oriented evaluation [9], to evaluate the matching result. We can as well use other classic evaluation methods, such as Turing test, for the evaluation. We are interested in the further investigation on how we can extend our evaluation method by involving different evaluators and how to take the evaluation result for adjusting the strategy.

The approach presented in this paper contributes to the state of the art by focusing on capturing and evaluating the user knowledge. Regarding the evaluation methodology and the delivery of security annotations suggestions, they are based on the ontology based data matching methodology, which contributes from the knowledge of the community (e.g. members of multiple organizations sharing a common goal).

7 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.

The similarity score between the user input and the knowledge stored in the ontology and knowledge bases is used by the system in order to infer the best fitted security annotations recommendations. An evaluation methodology has been developed to evaluate the (ontology-based) matching strategy used. The evaluation methodology shows a satisfactory rate of 83% when comparing the results delivered by the annotator with the experts' expectations.

The knowledge (i.e. security annotations) is modeled using the DOGMA ontology, which has the advantage of being grounded in natural language.

A future work is to consider various statistics during the recommendation which capture the user's characteristics and annotation behavior (e.g. to infer the user context and knowledge from his inputs). The enrichment of the knowledge base and a mechanism for checking the correctness of the specified parameter values is work in progress. The outcome of the annotator and the quality of the recommendations are dependent on the knowledge base updates and on the domain expert responsible for managing the knowledge base.

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

1. Tang, Y., Meersman, R., Ciuciu, I.G., Leenarts, E., Pudney, K: Towards Evaluating Ontology Based Data Matching Strategies. In: Proceedings of 4th IEEE Research Challenges in Information Science (RCIS'10), pp. 137--146, ISBN: 978-1-4244-4839-5, Nice, France (2010)
2. De Baer, P., Tang, Y., Meersman, R.: An Ontology-Based Data Matching Framework: Use Case Competency-Based HRM. In: Proceedings of the 4th Int. OntoContent'09 Workshop, OTM, LNCS, pp. 514--523, Portugal (2009)
3. Jaro, M. A.: Probabilistic linkage of large public health data files (disc: P687-689). Statistics in Medicine 14, 491--498 (1995)
4. Cohen, W.W., Ravikumar, P.: Secondstring: An open source java toolkit of approximate string-matching techniques, http://secondstring.sourceforge.net (2003)
5. Spyns, P., Tang, Y., Meersman, R.: An ontology engineering methodology for DOGMA, Journal of Applied Ontology 3 (1-2), 13--39 (2008)
6. Ciuciu, I., Zhao, G., Mülle, J., von Stackelberg, S., Vasquez, C., Haberecht, T., Meersman, R., Böhm, K.: Semantic Support for Security-Annotated Business Process Models. In: Proceedings of BPMDS/CAISE'11, Springer LNBIP (joint with EMMSAD), London (2011)
7. Mülle, J., von Stackelberg, S, Böhm, K.: A Security Language for BPMN Process Models. Technical Report, Karlsruhe Institute of Technology (KIT), no. 2011-09, Karlsruhe (2011).
8. Patton, M.Q.: Qualitative Research and Evaluation Methods, 3rd edition, Sage Publications, Inc, London, UK, ISBN: 0-7619-1971-6 (2002)
9. Bhola, H.S.: Evaluating "Literacy for development" projects, programs and campaigns: Evaluation planning, design and implementation, and utilization of evaluation results. Hamburg, Germany: UNESCO Institute for Education; German Foundation for International Development (1990)
10. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z., Ontology Alignment for LOD, Proceedings of ISWC'10, Springer-Verlag, Heidelberg, (2010)
11. Knowledge Web FP6-507482, http://knowledgeweb.semanticweb.org
12. OpenKnowledge FP6-027253, http://www.iiia.csic.es/en/project/open-knowledge
13. Betz, S.; Klink, S.; Koschmider, A.; Oberweis, A.: Automatic User Support for Business Process Modeling. In: Proceedings of the Workshop on Semantics for Business Process Management at the 3rd European Semantic Web Conference 2006, pp. 1--12, Budya, Montenegro (2006)
14. Born, M., Dorr, F., Weber, I.: User-friendly Semantic Annotation in Business Process Modeling. In: Proceedings of the Workshop on Human-friendly Service Description, Discovery and Matchmaking (2007)
15. Di Francescomarino, C., Tonella, P.: Supporting Ontology-Based Semantic Annotation of Business Processes with Automated Suggestions. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009, LNCS, vol. 29, pp. 211--223, Springer, Heidelberg, (2009)