Noname manuscript No.
(will be inserted by the editor)

Collaborative Feature Location in Models through Automatic Query Expansion

Francisca Pérez · Jaime Font · Lorena Arcega · Carlos Cetina

Received: date / Accepted: date

Abstract Collaboration with other people is a major theme in the information-seeking process. However, most existing works that address the location of features during the maintenance or evolution of software do not support collaboration, or they are focused on code as the main software artifact. Hence, collaborative feature location in models has not enjoyed much attention to date. In this work, we address this concern by proposing an approach, CoFLiM, that enables the collaboration of several domain experts in order to locate the model fragment of a target feature. CoFLiM uses the feature descriptions of the domain experts and their self-rated confidence level to automatically reformulate the relevant feature descriptions in a single query. This query guides the evolutionary algorithm of our approach that finds the model fragment of the feature being located. We evaluate CoFLiM in a real-world case study from our industrial partner. We analyze the impact of CoFLiM in terms of recall, precision, and the F-Measure. Moreover, we compare the reformulation of CoFLiM with four baselines. We also perform a statistical analysis to show that the impact of the results is significant. Our results show that collaboration pays off in the location of features in models. The results also show that the self-rated confidence level can be used to locate features in models. Finally, the results show that there are no significant improvements when more than three domain experts are involved, which is relevant in those industrial contexts where the availability of domain experts is scarce.

Keywords Query expansion · Collaborative Information Retrieval · Feature Location · Search-Based Software Engineering · Model Driven Engineering

F. Pérez, J. Font, L. Arcega and C. Cetina
SVIT Research Group, Universidad San Jorge.
Autovía A-23 Zaragoza-Huesca Km.299, 50830, Zaragoza, Spain. Tel.: +34 976060100
E-mail: {mfperez, jfont, larcega, ccetina}@usj.es

J. Font and L. Arcega
Department of Informatics, University of Oslo.
Postboks 1080 Blindern, 0316 Oslo, Norway

1 Introduction

Feature Location (FL) is one of the most important tasks during the maintenance and evolution of software [Dit et al., 2013]. It is known as the process of finding the set of software artifacts that realize a specific functionality. However, most of the existing works in the literature have been focused on code as the software artifact that materializes the feature to locate [Rubin and Chechik, 2013; Dit et al., 2013], whereas model artifacts have not received yet enough attention.

Moreover, collaboration ideas have never been evaluated in the location of features in models even though collaboration is a useful and often necessary component of complex projects when the task at hand is difficult or cannot be carried out by one individual [Shah, 2010; Morris, 2007]. Collaboration needs to be considered in industrial contexts since a vast amount of software is accumulated over the years and this software has been developed and maintained by different individuals.

In this work, we propose an approach to achieve collaborative Feature Location in Models (CoFLiM). In CoFLiM, several domain experts collaborate to locate the model fragment that materializes a target feature. First, each domain expert provides both a feature description and an estimation of confidence level. CoFLiM uses the confidence level to identify relevant feature descriptions. Then, CoFLiM automatically reformulates the relevant feature descriptions in a single query using a technique that is based on Rocchio's method [Salton, 1971]. The resulting query is used to guide the evolutionary algorithm of our CoFLiM approach in order to find the model fragment that realizes the feature being located.

We analyze the impact of CoFLiM in a real-world industrial case study from the railway domain. Our industrial partner, Construcciones y Auxiliar de Ferrocarriles (CAF)[1], is a worldwide leader in train manufacturing. We also compare the impact of the automatic query reformulation that CoFLiM performs with four alternatives as baselines to study the impact on the results. We also analyze how the number of domain experts collaborating influences the quality of the solution. Moreover, we analyze whether the inclusion of the domain experts' confidence produces an improvement in the solution.

To perform these analyses, we extract a case study from our industrial partner that includes both the models of software that control and manage the trains and the oracle (the realization of features validated by our industrial partner). Then, we involve 19 domain experts from our industrial partner to obtain feature descriptions and confidence levels as the input of our CoFLiM approach. We compare the results of CoFLiM with the oracle (which is considered to be the ground truth) in terms of recall, precision, and the F-measure. Finally, we perform a statistical analysis (following the guidelines by [Arcuri and Briand, 2014]) in order to provide quantitative evidence of the impact of the results and to show that this impact is significant.

[1] `www.caf.net/en`

The results show that collaboration pays off in the location of features in models. Our CoFLiM approach improves the results of locating features without collaboration by 27.42% in recall and 25.78% in precision. The results also show that there are no significant improvements when more than three domain experts collaborate. This challenges the general recommendation [Carpineto and Romano, 2012], which requires the collaboration of five domain experts. This is relevant in those industrial contexts where the availability of domain experts is scarce. Finally, our results suggest that leveraging the self-rated confidence in CoFLiM leads to improvement in the results obtained without using the self-rated confidence (an average improvement of 24.11% in recall and 21.97% in precision).

The rest of the paper is structured as follows: Section 2 provides the required background. Section 3 presents an overview of our approach. Section 4 presents the automatic query expansion technique. Section 5 describes the location of relevant model fragments for a given query. Section 6 describes the evaluation, and Section 7 shows the results. Section 8 describes the threats to validity. Section 9 reviews the related work. Finally, Section 10 concludes the paper.

2 Background and motivation

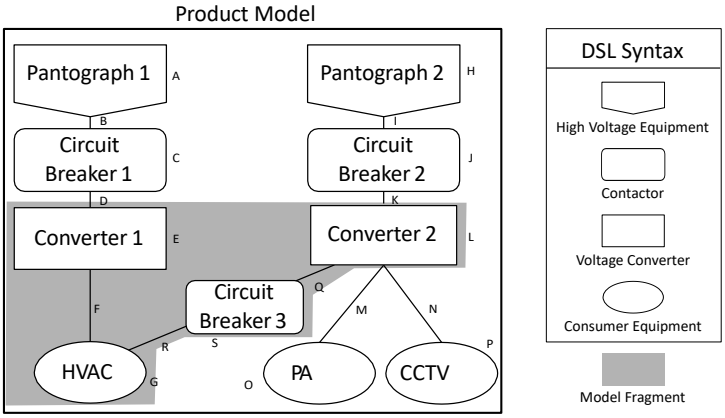
This section introduces the railway domain of our industrial partner and the Domain Specific Language (DSL) that was used to formalize the products manufactured as well as an example of model fragment that realizes a feature. We also motivate the need of an approach to achieve collaborative feature location in models.

Our industrial partner furnishes train units with multiple pieces of equipment through its vehicles and cabins. These pieces of equipment are often designed and manufactured by different providers, and their aim is to carry out specific tasks for the train. Some examples of these devices are: the traction equipment, the compressors that feed the brakes, the pantograph that harvests power from the overhead wires, or the circuit breaker that isolates or connects the electrical circuits of the train. The control software of the train unit is in charge of making all of the equipment cooperate to achieve the train functionality while guaranteeing compliance with the specific regulations of each country.

For example, one of the scenarios from the high voltage connection sequence is initiated under demand by the train driver through interface devices fitted inside the cabin. The control software is in charge of raising the pantograph to harvest power from the overhead wire and of closing the circuit breaker, so the energy can get to converters that adapt the voltage to charge batteries, which, in turn, power the traction equipment. Another example of the functionality is the coupling between train units. The DSL of our industrial partner has the expressiveness required to describe both the interaction between the main pieces of equipment installed in a train unit and the non-functional aspects

that are related to regulation. For the sake of understandability and legibility and due to intellectual property rights concerns, we present an equipment-focused simplified subset of the DSL. This subset of the DSL will be used throughout the rest of the paper to present a running example.

Fig. 1 depicts an example of a product model from a real-world train. It shows two separate pantographs (High Voltage Equipment) that collect energy from the overhead wires and send it to their respective circuit breakers (Contactors), which, in turn, send it to their independent Voltage Converters. The converters then power their assigned Consumer Equipment: the HVAC on the left (the train's air conditioning system), and the PA (public address system) and CCTV (television system) on the right.



Model Fragment Encoding:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	1	1	1

Fig. 1 Example of a model, a model fragment and its encoding

The elements of Fig. 1 that are highlighted in gray show an example of a model fragment, which is the realization of the feature: HVAC Assistance. This model fragment allows the passing of current from one converter to the HVAC that is assigned to its peer for coverage in case of overload or failure of the first converter.

The model fragment (which always belongs to a product model) is encoded using a string of binary values that contains as many positions as elements in the product model. Each position in the string has two possible values: 0 in case the element does not appear in the fragment, or 1 in case the element does appear in the fragment. The lower part of Fig. 1 shows the encoding of the model fragment that is highlighted in gray. Since elements E, F, G, L, Q, R and S conform the model fragment, the corresponding values are set to 1 in its binary string representation.

Although the product model and the model fragment that realizes the feature of the example of Fig. 1 makes that manual feature location in models may appear easy, it becomes very complex in the models of our industrial partner. Suppose we ask the domain experts to manually locate the model elements that correspond to the 121 features of the data set provided by CAF. Taking into account that the data set comprises 23 trains and the model of each train has more than 1200 model elements, at least 27600 model elements should be evaluated. To assess a model element, it is reasonable to consider its properties. In the data set, each element has about 15 properties. Therefore, about 414000 properties of model elements should be considered. Assuming that a domain expert only needs 1 second to consider a property of a model element, the domain expert needs 4.79 days to manually locate each feature. Considering the 121 features and the 19 domain experts, the result is 30.17 years.

Domain experts could make use of simple text search tools in the models, but these tools would not prevent domain experts from first knowing the models of the trains. There is no domain expert who knows all models completely in CAF. Although we ignore that domain experts can forget models that belong to trains manufactured over two decades as is the case in CAF, the models have always been created by several different domain experts. Moreover, the models may have been maintained by other domain experts who have not participated in the creation. Time improvements because of the learning effect, or locating several features simultaneously are not accounted here, but these improvements could also be source of errors which take time to fix.

In addition, the 30.17 years do not include the time that is necessary to reach a consensus on 19 solutions for each of the 121 features. In an industrial environment like in CAF, the domain experts are distributed in three different cities of Spain (Zaragoza, Beasain and Bizkaia). This geographical distribution implies that the domain experts are not used to carrying out consensus tasks, which can negatively influence the time they need to agree on the solutions.

Therefore, feature location in real-world models is not a trivial task. From the 121 features of the data set, only the model elements of 43 features are documented in CAF. The documentation of these 43 features is the result of months of manual work with external consultants to address certification needs or bugs. Moreover, this data set is made up of tramway models, but the need to locate features is also present in more CAF models of similar complexity as subway models, or in more complex models such as suburban and high speed.

3 Overview of the CoFLiM approach

Fig. 2 shows an overview of our proposed CoFLiM approach. First, each domain expert involved provides both a feature description and a self-rated confidence level for the feature description as input. Second, feature descriptions are ordered from the highest to the lowest confidence level. The first feature description is set as base query, and the k subsequent feature descriptions are

set as relevant documents. Third, the base query is automatically reformulated to expand it with the most representative terms found in the relevant documents. Finally, both the models of a product family and the reformulated query are taken as input to locate the relevant model fragments. The evolutionary algorithm of CoFLiM explores the model fragments guided by the similitude of each model fragment with the reformulated query. The result is a ranking of relevant model fragments that is ordered by the similitude to the input reformulated query.

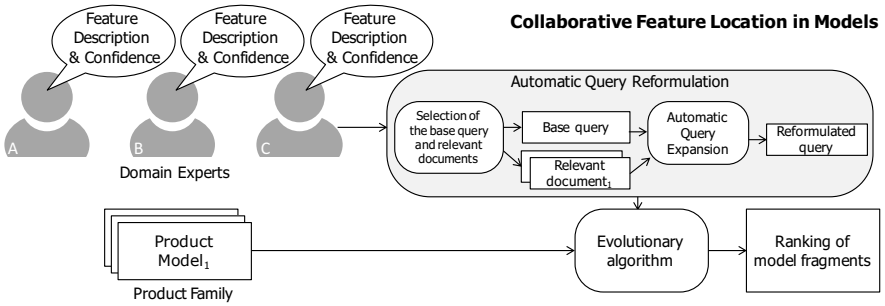


Fig. 2 Overview of our approach for collaborative feature location in models

In the next two sections, we describe the phase that selects the base query and the relevant documents to automatically obtain the reformulated query, and the evolutionary algorithm that retrieves the ranking of relevant model fragments for the reformulated query.

4 Automatic Query Reformulation

Domain experts can just discuss together and collaboratively draft a better description of the feature but this manual reformulation is time-consuming in complex projects that accumulates a vast amount of software. This is because the combination of a set of feature descriptions in a single query requires much time since domain experts have to both understand all feature descriptions and discuss about the most adequate terms to locate the target feature. In order to help domain experts, the combination of a set of feature descriptions in a single query can be automatically performed.

To enable the collaboration from different domain experts' feature descriptions, our approach sets one of the feature descriptions as the base query. Afterwards, the base query is automatically reformulated to expand it with the most representative terms found in the domain experts' feature descriptions set as relevant documents.

4.1 Selection of the base query and relevant documents

To determine which feature description is selected as the base query, CoFLiM sorts the feature descriptions from the highest to the lowest self-rated confidence level in order to select the feature description in the first position (i.e., the highest self-rated confidence) as the base query. The self-rated confidence level is supplied for each feature description using a Likert scale ranging from 7 (the highest self-rated confidence) to 1 (the lowest self-rated confidence). In case that more than one feature description ties the maximum self-rated confidence level, CoFLiM selects the longest feature description as the base query since it can include more terms that are relevant to locate the target feature. CoFLiM then selects k feature descriptions sorted by confidence level, where k is the number of domain experts who collaborate to reformulate the base query. Each of the selected feature descriptions is set as a relevant document.

Fig. 3 shows an example of selection of the base query and relevant documents from different domain experts' feature descriptions and their self-rated confidence levels which are provided as input. Specifically, the figure shows three feature descriptions (identified from A-C) and their self-rated confidence levels. In this example, a total of three domain experts are involved in locating the target feature: one domain expert provides the base query, and two domain experts collaborate (k=2).

Domain Expert	Feature description	Self-rated confidence	
A	Passing of current from one converter to the HVAC assigned to its peer for coverage in case of overload or failure of the first converter	6	→ Base query
B	The circuit breaker changes to another converter in case of failure in the HVAC converter	4	→ Relevant document$_1$
C	In case of failure or overload in the converter that provides energy to the air conditioning unit, the circuit breaker provides energy from its converter	3	→ Relevant document$_2$

Three domain experts are involved in locating the target feature:
The base query from Domain expert A, and two relevant documents from Domain expert B and Domain Expert C (k=2)

Fig. 3 Example of selection of the base query and two relevant documents

In the example of Fig. 3, the feature description provided by Domain expert A is selected as the base query since it has the highest self-rated confidence level (6), and each of two subsequent feature descriptions with the highest self-rated confidence level are set as relevant document (the feature description of Domain expert B is set as relevant document 1 since its self-rated confidence level is 4, and the feature description of Domain expert C is set as relevant document 2 since its self-rated confidence level is 3). Note that, in our approach, it is important to discriminate between the base query and the relevant documents since the terms that are included in the relevant documents can be used to expand the base query. In contrast, it is not used the number assigned to each relevant document (1 and 2 in the example of the figure) and we assign this number for understandability and legibility in the figures.

It is important to highlight that when a feature needs to be located at a point in time, each domain expert provides a feature description and assigns it a confidence level. In case that only one domain expert provides a feature description, our approach also works in a solo scenario but the automatic query expansion is not performed. Hence, the feature description that was only provided as input is set as query to locate the target feature.

4.2 Automatic Query Expansion

Before the base query is expanded, our approach processes the Natural Language (NL) text of the base query and the relevant documents in order to homogenize the text. The combination of Natural Language Processing (NLP) techniques, such as the analysis of POS tags, removal of stopwords, and stemming, is a frequent practice [Hulth, 2003] that our approach adopts as follows:

1. The text is tokenized (divided into words). A white space tokenizer can usually be applied (which splits the strings whenever it finds a white space); however, for some sources of description, more complex tokenizers need to be applied. For instance, when the description comes from documents that are close to the implementation of the product, some words could be using CamelCase naming.
2. The Parts-of-Speech (POS) tagging technique is applied to analyze the words grammatically and to infer the role of each word in the text provided. As a result, each word is tagged, which allows the removal of some categories that do not provide relevant information. For instance, conjunctions (e.g., *or*), articles (e.g., *a*), or prepositions (e.g., *at*) are words that are commonly used and do not contribute relevant information to describe the feature, so they are removed.
3. Stemming techniques are applied to unify the language that is used in the text. This technique consists of reducing each word to its root, which allows different words that refer to similar concepts to be grouped together. For instance, plurals are turned into singulars (*doors* to *door*) or verb tenses are unified (*using* and *used* are turned into *us*).
4. The Domain Term Extraction and Stopword Removal techniques are applied. In order to carry out these techniques, domain experts provide two separate lists of terms: one list of both single-word and multiple-word terms that belong to the domain and must be kept for analysis, and a list of irrelevant words that have no analysis value. Both kinds of terms are also homogenized and they can be automatically filtered in or out of the final query.

Fig. 4 depicts the result of homogenizing the NL text of the base query and the NL text of the two relevant documents when each of the NLP techniques previously described above is applied (division into words, analysis of the words grammatically, root reduction, and domain term extraction and stopword removal). For example the feature description that is set as the base

query is processed from the NL text: *Passing of current from one converter to the HVAC assigned to its peer for coverage in case of overload or failure of the first converter* to the homogenized terms: *current, convert, hvac, coverag, overload, failur, convert,* and *assign.*

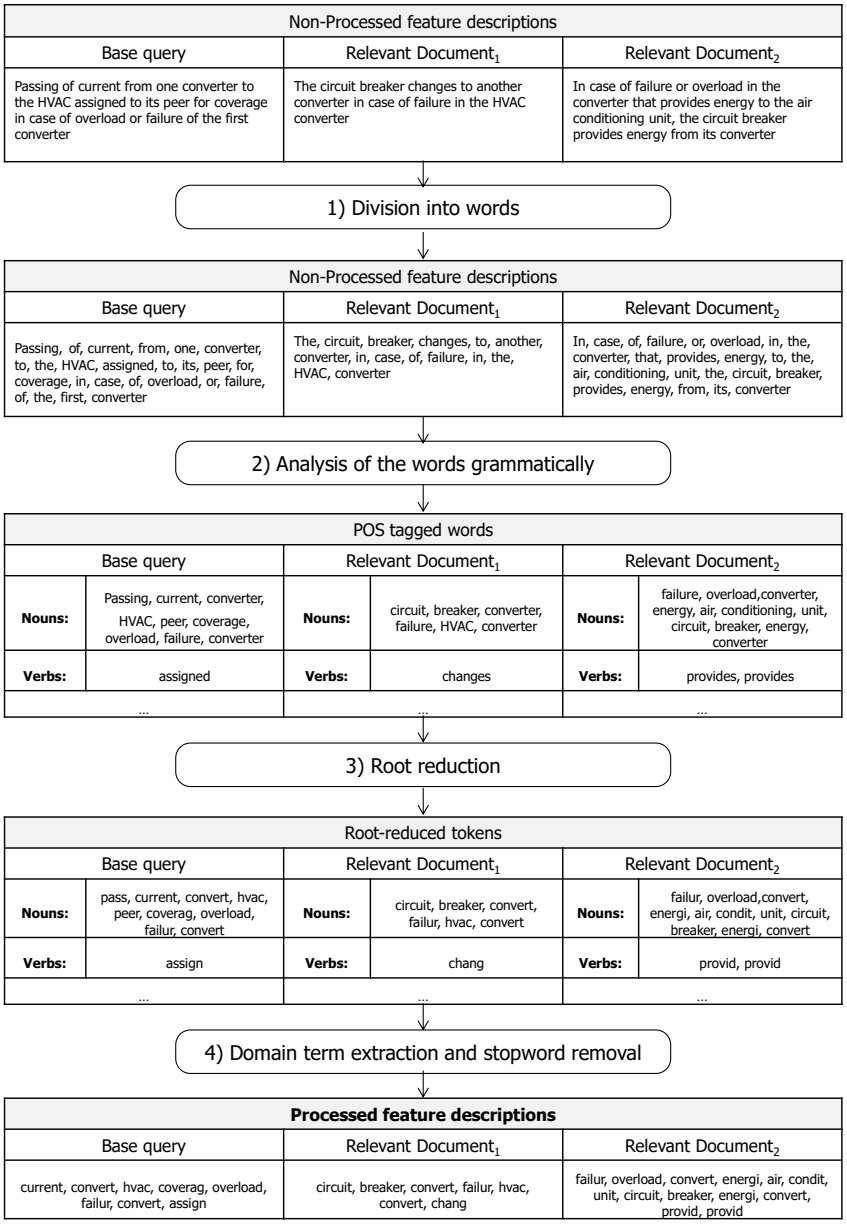


Fig. 4 Example of text homogenization using NLP techniques

Once the NL text is homogenized, our approach automatically reformulates the base query to expand it with terms of the relevant documents using a technique that is based on Rocchio's method [Salton, 1971], which is perhaps the most commonly used method for query reformulation [Sisman and Kak, 2013]. Rocchio's method orders the terms in the top K relevant documents based on the sum of the importance of each term of the K documents using the following equation:

$$Rocchio = \sum_{d \in R} TfIdf(t, d) \tag{1}$$

where R is the set of top K relevant documents in the list of retrieved results, d is a document in R, and t is a term in d. The first component of the measure is the Term Frequency (Tf), which is the number of times the term appears in a document; it is an indicator of the importance of the term in the document compared to the rest of the terms in that document. The second component is the Inverse Document Frequency (Idf), which is the inverse of the number of documents that contain that term; it indicates the specificity of that term for a document that contains it. Once the terms of the relevant documents are ordered, we consider the first 10 term suggestions to expand the base query, as is recommended in the domain literature [Carpineto and Romano, 2012].

Fig. 5 shows an example of terms and the frequency of the base query and the two relevant documents. Using Rocchio's method, the terms of the relevant documents are ordered from highest to lowest sum of importance. Then, the first 10 terms from the ordered list (*convert, energi, provid, overload, circuit, breaker, failur, hvac, air, condit*) are used to reformulate the base query by adding these terms. Hence, the reformulated query that is output has the following terms: *current, convert, hvac, converag, overload, failur, assign, energi, provid, circuit, breaker, air,* and *condit*.

5 Evolutionary algorithm guided by the reformulated query

CoFLiM relies on our Evolutionary Algorithm, which is guided by the reformulated query, to retrieve a ranking of model fragments. We use an evolutionary algorithm to locate features because the search space is too large, thus, it is impossible to explore the space of possibilities exhaustively. In a previous work [Font et al., 2015a], we limited the search space by choosing a subset of the models, or by providing restrictions of elements that do not have to appear in the solutions. However, the search space was still very large (a model of 500 elements can yield around 10^{29} potential fragments). In addition, evolutionary algorithms have obtained good results by addressing similar problems with large search spaces [Font et al., 2016b], hence we have chosen to use an evolutionary algorithm.

Fig. 6 shows an overview of the evolutionary algorithm. The algorithm is made up of three steps, which are indicated as a number in Fig. 6: Step 1,

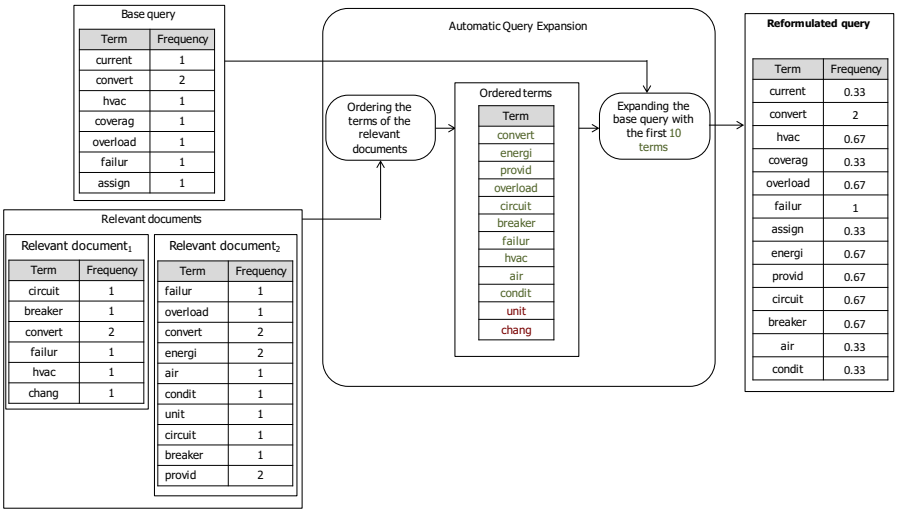


Fig. 5 Automatic Query Expansion example

Initialization of model fragments; Step 2, Genetic operations to generate model fragments that could realize the provided reformulated query; and Step 3, the Fitness function to assess the relevance of each generated model fragment according to the similitude with the reformulated query.

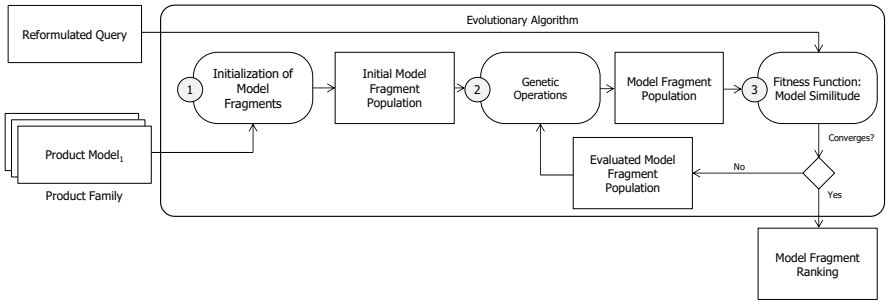


Fig. 6 Evolutionary algorithm overview

Step 1) Initialization of model fragments.

This step generates an initial model fragment population from the product models that serves as input for the evolutionary algorithm. To do this, parts of the models are extracted randomly and added to a collection of model fragments.

In order to generate random model fragments that are well-formed and correct, the algorithm starts with the selection of a random initial model element A to create a new model fragment. Then, another model element B, which is directly connected to the model element A, is taken. In case there is more than

one element directly connected to A, one of the elements is randomly chosen. Next, a random number of iterations are performed.

Note that this selection process based on direct model element connections makes that the algorithm returns a model fragment built with a subset of elements from the parent model which are contiguously connected. Since this algorithm only produces fragments that are part of the original model, the resulting model fragments are well-formed and valid (i.e., the model fragments keep the conformance to the metamodel).

Step 2) Genetic Operations.

This step generates a set of model fragments that could realize the reformulated query provided. The generation of new model fragments is based on existing model fragments and is done by applying two genetic operators that are adapted to work on model fragments: crossover and mutation.

- The **crossover operation** enables the creation of a new individual by combining the genetic material from both the model fragment and the referenced product model. Our crossover operation does not use single point crossover (where a point on both parents is picked randomly and the bits to the right of that point are swapped between the two parents), two point (where two points are picked randomly from the parents and the bits in between the two points are swapped between the parents) or uniform crossover (where individual bits of the parents are compared and exchanged with a fixed probability). Instead, our crossover operation is based on model comparisons to ensure that the new individuals are well-formed and valid. The crossover operation looks in the second parent for the model fragment of the first parent.
  The upper part of Fig. 7 shows an example of the application of the crossover operation. The model fragment (from Parent 1) is compared with the model of Parent 2. Since the model fragment is found in the model of Parent 2, the process creates a new individual with the model fragment taken from Parent 1, but referencing the model from Parent 2. Thus, this operation enables the search space to be expanded to a different product model, i.e., both model fragments (the one from Parent 1 and the one from the new individual) will be the same. However, since each of them is referencing a different product model, they will mutate differently and provide different individuals in further generations. In the case that the comparison does not find the model fragment in the second parent, the crossover returns the first parent (the model fragment) unchanged.
- The **mutation operator** is used to imitate the mutations that randomly occur in nature when new individuals are born. Specifically, the mutation operator is applied to add or remove elements of the model fragment. In case the operation is applied to add, it is chosen one element of the model fragment that is directly related to elements that are not included in the fragment. Next, one of the related elements that are not included in the fragment is added to the fragment. In case the operation is applied to remove, an element of the fragment that is connected with only one other

element of the fragment is removed from the fragment. In the example in the bottom part of Fig.7, the mutation operation is applied to remove elements. Hence, the mutation operation takes the offspring that is produced through the crossover operation and removes one element (Pantograph 1) and its relationship since it is connected with only one other element of the fragment (Circuit Breaker 1). The resulting model fragment is a new candidate in the population for the realization of the input reformulated query.

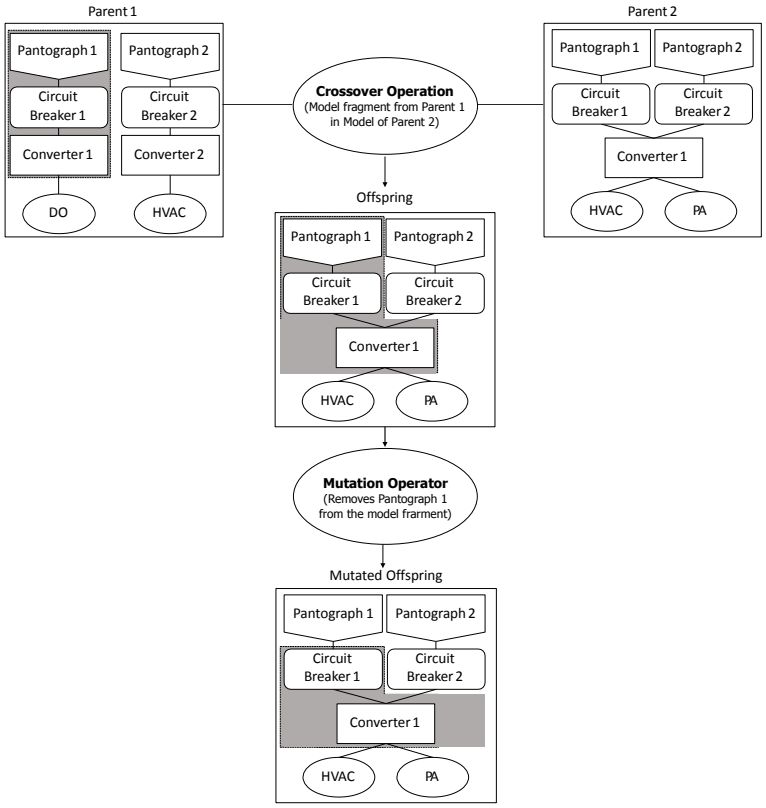


Fig. 7 Genetic operations example

Both, crossover and mutation operations are designed to produce valid individuals, by selecting subset of existing models to conform the new model fragment. For further details of the crossover operation and the mutation operator, we refer the reader to [Font et al., 2016a] and [Font et al., 2016b], respectively.

Step 3) The Fitness Function.

This step assesses the relevance of each of the produced candidate model fragments by ranking them according to a fitness function. The objective of

the fitness function is the similitude between the model fragment and the reformulated query. To do this, we apply methods that are based on Information Retrieval (IR) techniques. Specifically, we apply Latent Semantic Indexing (LSI) [Landauer et al., 1998; Hofmann, 1999] to analyze the relationships between the model fragments in the population and the reformulated query. Since the results retrieved by LSI depend greatly on the style in which the NL is written, it is beneficial to process the NL of the model fragments [Lapeña et al., 2017] the way that the NL of the base query and the relevant documents query are processed in Subsection 4.2 (i.e., by combining the analysis of POS tags, removal of stopwords, and stemming).

Then, the LSI technique can be applied taking the terms from the reformulated query and the terms from the model fragments as input. LSI constructs vector representations of a query and a corpus of text documents by encoding them as a term-by-document co-occurrence matrix. In other words, this is a matrix where each row corresponds to *terms* and each column corresponds to *documents* followed by the *reformulated query* in the last column. Each cell of the matrix contains the number of occurrences of a *term* inside a *document* or inside the *reformulated query*. In our approach, the *terms* are all of the individual terms that are extracted from the processed NL of model fragments and the reformulated query, the *documents* are the NL representations of model fragments, and the *query* is the reformulated query.

Once the matrix is built, it is normalized and decomposed into a set of vectors using a matrix factorization technique called Singular Value Decomposition (SVD) [Landauer et al., 1998]. One vector that represents the latent semantics of the NL texts is obtained for each *document* and for the *reformulated query*. Finally, the similarities between each *document* and the *reformulated query* are calculated as the cosine between both of their vectors, obtaining values between -1 and 1.

The upper part of Fig. 8 shows an example of a co-occurrence matrix. The columns are the NL representation of each model fragment in the population and the reformulated query. Each *term* row is one of the terms that is extracted from the NL texts of model fragments and the reformulated query. Each cell shows the number of occurrences of each of the *terms*.

The bottom left part of Fig. 8 shows the result of applying the SVD technique to the matrix. The vector labeled with 'Q' represents the *query*, while the ones labeled as 'MF' represent the model fragments. The bottom right part of Fig. 8 shows the scores for each model fragment, which are calculated by computing the cosine between their associated vector and the *query* vector.

Once the similitude scores are obtained, if the stop condition is met, the process will stop returning the model fragment ranking. If the stop condition is not yet met, the evolutionary algorithm will keep its execution one generation more.

The next time that the genetic operators are applied, it will be necessary to select the best candidates as parents for the new generation. This will be done based on the score obtained by each model fragment. As a result, model fragments with higher similarities will have more chances to be selected as

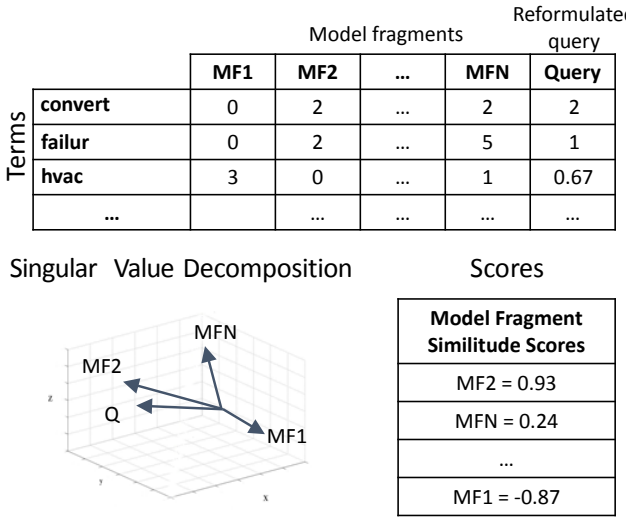


Fig. 8 Fitness by means of Model Fragment Similitude

parents of the new generation. Note that being part of more feature candidates does not guarantee a higher score for the model fragment since the similarity between a feature candidate and the reformulated query can be negative.

The process of generation of fragments, extraction of feature candidates, and assessment of those candidates is repeated until the stop condition is met. Usually, the stop condition can be a time slot, a fixed number of generations, or a trigger value of the fitness that makes the process terminate when reached. In addition, it is also possible to monitor the fitness values and determine when they are converging and no further improvements are being made by new generations. The stop condition greatly depends on the domain and the problem being solved; therefore, it is adjusted based on the results being output by the process.

Once the model fragment ranking is obtained, the inputs of the approach (i.e., the feature descriptions, their confidence, and the reformulated query) are discarded and new inputs will be necessary for executing the approach again. As occurs in other works that retrieve text from an initial query, results depend on the quality of the queries [Sisman and Kak, 2013; Hill et al., 2009]. As much as properties and values of the elements are explicitly mentioned in the query, closer will be the result to the search objective. Therefore, if irrelevant model fragments are obtained in the ranking, the domain experts can consider these solutions as a starting point from where solutions can be manually refined, or they may refine the query to automatically obtain different solutions.

6 Evaluation

This section presents the research questions that our work tackles, the baselines that we use to put the performance of our work in perspective, the data set of our real-world case study, the planning, and the execution to answer each research question.

6.1 Definition

There are several aspects that we wanted to evaluate with regard to the inclusion of more than one domain expert's feature description for locating features in models. In order to address the evaluation of these aspects, we formulated the following four research questions:

RQ_1: *Does the query formulated through automatic query expansion produce an improvement in terms of solution quality compared to other alternatives?*

RQ_2: *Does the inclusion of more than one domain expert's feature description when locating features in models produce an improvement in terms of solution quality?*

RQ_3: *If so, what is the influence of the number of domain experts input to the feature location in the quality of the solution?*

RQ_4: *Does the inclusion of the domain experts' confidence produce an improvement in terms of solution quality?*

Answering RQ_1 allows us to compare the performance results (in terms of recall, precision and the F-measure) of the query formulated through automatic query expansion with the query formulated through different alternatives. A positive answer for RQ_2 implies that more than one domain expert's feature description improves the quality of the located model fragment. In this case, answering RQ_3 allows us to determine the number of domain experts that should input our CoFLiM approach in order to achieve the best result. Answering RQ_4 allows us to determine whether or not taking into account the domain experts' confidence level provided for each feature description improves the results.

6.2 Baselines

In order to put the performance of our work in perspective, we compare it with four baselines to study the impact on the results. Each baseline explores a different alternative to obtain the query that will be used for locating a target feature. The baselines are the following:

Baseline 1: Topic modeling. Topic modeling is a widely-used unsupervised machine learning technique [Asuncion et al., 2010], based on Latent Dirichlet Allocation (LDA) [Blei et al., 2003] for automatically extracting semantic or thematic topics from a collection of text documents, without

requiring previous training data with training labels. The topics provide a high-level abstract representation of documents in a corpus, and can be used for searching, categorizing, and navigating through collections of documents. Each of the identified topics comprehends a subset of the terms that appear in the documents. The terms may belong to more than one topic at the same time. Within each topic, a term has a weight assigned by LDA, representing its overall contribution of to the topic.

Among the uses and possibilities of topic modeling, the technique has been leveraged as a means of query expansion for IR purposes [Boyd-Graber et al., 2017]. The work presented in [Zeng et al., 2012] shows an application of topic models for an straightforward query expansion process in the context of IR in medicine documents. Topic models was applied for query expansion as follows: (1) extracting topics from the documents through LDA (along with the terms comprehended by each topic), (2) matching queries with topics through the terms of the query, (3) ranking the terms from the matching topics by summing their weights in all matching topics, and (4) expanding the query with the top 10 ranked terms. This application of topic models for query reformulation is a baseline that we explore in this work.

Baseline 2: Union (K=all). Union is a simple technique for merging the terms for all information resources in order to insure that there be no loss of information. In previous works such as [Arens et al., 1996], the union of information resources was used for query reformulation. The union of the terms that are included in all feature descriptions provided as input is a baseline that we explore in this work.

Baseline 3: Union (K=5, c=self-rated). This is a variant of the union technique that limits the number of feature descriptions that are used for merging terms. In order to select the feature descriptions, it is used the self-rated confidence level provided for each feature by the domain experts. We explore this variant of the union technique as a baseline by limiting the number of feature descriptions to 5. This decision was made based on recommendations found in the literature [Carpineto and Romano, 2012]. The results serve to study the impact on the results that has both the limitation in the number of feature descriptions and the domain experts' confidence level.

Baseline 4: Domain expert query. This is a manual technique for query reformulation in which a domain expert is asked to combine the feature descriptions manually. To do this, the domain expert who has the highest sum of self-rated confidence of all feature descriptions is selected. Next, this domain expert takes as input all descriptions of a target feature to create a feature description that includes the necessary terms to locate the feature. In this work, we explore this manual technique for query reformulation as a baseline in order to study its impact in terms of solution quality.

6.3 Data set

The data set is provided by our industrial partner, CAF, which is an international provider of railway solutions all over the world. Their railway solutions can be seen in different types of trains (regular trains, subway, light rail, monorail, etc.). CAF provided us with 23 trains where each product model on average is composed of more than 1200 elements. They are built from 121 different features that can be part of a specific product model.

In addition, CAF provided us with the model fragments of 43 features from different trains. The model fragments have between 5 and 20 model elements, with an average of 13.55 model elements and a median of 14 model elements. It is important to highlight that each model element has properties that include terms, which are used to differentiate among model elements. Nineteen domain experts from our industrial partner were involved in providing a description for each feature. Moreover, CAF provided us with lists of domain terms and stopwords to process the NL. The domain terms list has around 300 domain terms, and the stopwords list has around 60 words. The list of domain terms and stopwords was obtained from the existing documentation in CAF. Specifically, it was obtained from the existing documentation that is used for training new employees. Our approach needs these two lists, but we believe that in industries similar to CAF will also have documentation for training new employees in their domain, and this documentation can be used to generate the two lists. In any case, the lists are not created collaboratively in our work.

6.4 Implementation details

We have used the Eclipse Modeling Framework to manipulate the models and CVL [Haugen et al., 2008] to manage the model fragments. The techniques used to process the NL have been implemented using OpenNLP [ope, 2016] for the POS-Tagger and the English (Porter2) stemming algorithm [sno, 2017] for the stemming algorithm (originally created using snowball and then compiled to Java). The LSI has been implemented using the Efficient Java Matrix Library (EJML [ejm, 2016]). The genetic operations are built upon the Watchmaker Framework for Evolutionary Computation [Dyer, 2016].

The crossover operation is applied with a crossover probability of 0.9. The mutation operation is applied with a probability of 0.1. The number of generations (repetitions of the genetic operations and fitness loop) that is allowed for the algorithm is 2500 since it is the value needed by our case study to converge (Note that this value is case specific). The rest of the settings are detailed in Table 1. We have principally chosen values for those settings that are commonly used in the literature [Sayyad et al., 2013].

As suggested by [Arcuri and Fraser, 2013] and confirmed in [Kotelyanskii and Kapfhammer, 2014], tuned parameters can outperform default values generally, but they are far from optimal in individual problem instances.

Table 1 Configuration parameters

Parameter description	Value
$Size$: Population Size	100
μ: Number of Parents	2
λ: Number of offspring from μ parents	2
r: Solutions replaced at population size	2
$p_{crossover}$: Crossover probability	0.9
$p_{mutation}$: Mutation probability	0.1

Therefore, the objective of this paper is not to tune the values to improve the performance of our approach but rather to compare it to a different variant of our approach using default parameter values.

To establish the stop condition, we perform a preliminary study to determine the time needed for the evolutionary algorithm to converge (point where there is no changes with further generations). For the searches performed in this domain the time needed was below 60 seconds for all the features, therefore we established the stop condition in 80 seconds (adding a margin to ensure convergence).

6.5 Planning and execution

Fig. 9 shows an overview of the process that was planned to answer the research questions. For each of the 43 features, we ask each of the 19 domain experts for both a feature description and a self-rated confidence level.

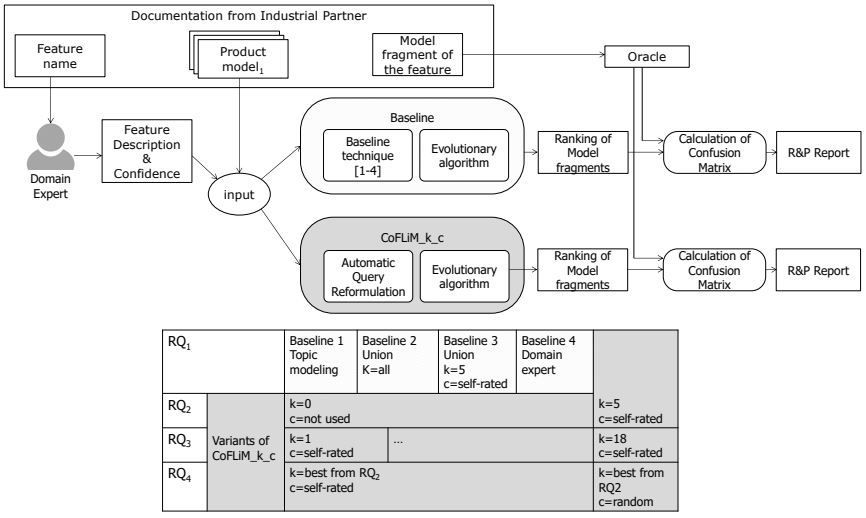


Fig. 9 Evaluation process

The process entails the execution of four baselines and different variants of our CoFLiM approach that set both k (number of domain experts who collaborate) and the type of confidence level (not used, self-rated, or random) with different values. The lower part of Fig. 9 shows a table with the baselines and the variants of CoFLiM that we use to answer each of the four research questions.

When a baseline or a variant of CoFLiM is executed, we obtain as a result a ranking of model fragments that realize a target feature. Then, we take the best solution of the ranking (i.e., the model fragment that is at position 1) to compare it with an oracle as Fig. 9 shows. The oracle is prepared using the model fragments that realize each target feature provided by our industrial partner. The oracle will be considered the ground truth and will be used to calculate a confusion matrix.

The confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data (the best solutions) for which the true values are known (from the oracle). In our case, each solution obtained is a model fragment that is composed of a subset of the model elements that are part of the product model. Since the granularity is at the level of model elements, each model element presence or absence is considered as a classification. The confusion matrix distinguishes between the predicted values and the real values classifying them into four categories:

- True Positive (TP): values that are predicted as true (in the solution) and are true in the real scenario (the oracle).
- False Positive (FP): values that are predicted as true (in the solution) but are false in the real scenario (the oracle).
- True Negative (TN): values that are predicted as false (in the solution) and are false in the real scenario (the oracle).
- False Negative (FN): values that are predicted as false (in the solution) but are true in the real scenario (the oracle).

Finally, some performance measurements are derived from the values in the confusion matrix. Specifically, we create a report for the confusion matrix including three performance measurements: recall at position 1 (R@1), precision at position 1 (P@1), and F-measure.

Recall measures the number of elements of the solution that are correctly retrieved by the proposed solution and is defined as follows:

$$Recall = \frac{TP}{TP + FN}$$

Precision measures the number of elements from the solution that are correct according to the ground truth (the oracle) and is defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

The F-measure corresponds to the harmonic mean of precision and recall and is defined as follows:

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2TP + FP + FN}$$

Recall values can range from 0% (which means that no single model element obtained from the oracle is present in the solution) to 100% (which means that all the model elements from the oracle are present in the solution). Precision values can range from 0% (which means that no single model fragment from the solution is present in the oracle) to 100% (which means that all the model fragments from the solution are present in the oracle). A value of 100% precision and 100% recall implies that both the solution and the oracle are the same.

6.5.1 Statistical analysis

To properly compare the recall and precision reports obtained from the different baselines and variants of CoFLiM, we use statistical methods following the guidelines of [Arcuri and Briand, 2014]. This statistical analysis provides formal and quantitative evidence (statistical significance) that the collaboration in feature location does in fact have an impact on the comparison metrics (i.e., that the differences in the results were not obtained by mere chance).

To enable statistical analysis, all of the approaches should be run a large enough number of times (in an independent way) to collect information on the probability distribution for each approach. A statistical test should then be run to assess whether there is enough empirical evidence to claim (with a high level of confidence) that there is a difference between the two techniques (e.g., A is better than B). In order to do this, we perform a post hoc analysis, which performs a pair-wise comparison among the results of each report to determine whether statistically significant differences exist. The post hoc analysis provides a probability value, $p - value$. It is accepted by the research community [Arcuri and Briand, 2014] that a $p - value$ under 0.05 is statistically significant.

When comparing techniques with a large enough number of runs, statistically significant differences can be obtained even if they are so small as to be of no practical value [Arcuri and Briand, 2014]. Then it is important to assess if the results of a report are statistically better than another and to assess the magnitude of the improvement. In this work, we use two non-parametric effect size measures: Vargha and Delaney's $\hat{A}_{12}$ [Vargha and Delaney, 2000; Grissom and Kim, 2005] and Cliff's delta [Cliff, 1993, 1996].

Vargha and Delaney's $\hat{A}_{12}$ measures the probability that running one approach yields higher values than running another approach. If the two techniques are equivalent, then $\hat{A}_{12}$ will be 0.5. For example, $\hat{A}_{12} = 0.7$ means that we would obtain better results in 70% of the runs with the first of the pair of approaches that have been compared, and $\hat{A}_{12} = 0.3$ means that we would obtain better results in 70% of the runs with the second of the pair of approaches that have been compared.

Cliff's delta is an ordinal statistic that describes the frequency with which an observation from one group is higher than an observation from another group compared to the reverse situation. It can be interpreted as the degree to which two distributions overlap with values ranging from -1 to 1. For instance, when comparing distribution x and distribution y: a value of 0 means no difference between two distributions; a value of -1 means that all samples in distribution x are lower than all samples in distribution b; a value of 1 means the opposite (all samples in x higher than all samples in y). In addition, threshold values were defined [Romano et al., 2006] for the interpretation of Cliff's delta effect size ($|d| < 0.147 \rightarrow$ "negligible"; $|d| < 0.33 \rightarrow$ "small"; $|d| < 0.474 \rightarrow$ "medium", $|d| \geq 0.474 \rightarrow$ "large").

6.5.2 Answering RQ_1

To answer whether the query formulated through automatic query expansion produces an improvement in the solution compared to other alternatives, we execute 30 independent runs (as suggested by [Arcuri and Fraser, 2013]) for each baseline to locate each of the 43 features, i.e., 43 (target features) x 30 (repetitions) x 4 (baselines) = 5160 independent runs.

Moreover, we execute CoFLiM_5_S (with *k=5* and *c=Self-rated*) to take as input the 19 domain experts' feature descriptions to locate each of the 43 target features. We also execute 30 repetitions of CoFLiM for each feature, i.e., 43 (target features) x 30 (repetitions) = 1290 independent runs. Specifically, we set k=5 (i.e., one domain expert's feature description is set as base query and five domain experts' feature descriptions are set as relevant documents for the location of the target features). This decision was made based on recommendations found in the literature [Carpineto and Romano, 2012].

With the obtained results, we record the mean values and standard deviations for precision, recall, and the F-measure. Also, we record a $p - value$, an $\hat{A}_{12}$ value, and a Cliff's delta value from the comparison of the results between CoFLiM_5_S and each baseline.

6.5.3 Answering RQ_2

To answer whether the inclusion of more than one domain experts' feature description produces an improvement in the solution, we execute 30 independent runs (as suggested by [Arcuri and Fraser, 2013]) of CoFLiM_0_N (with *k=0* and *c=Not used*) for each of the 43 features and the 19 domain experts to record the mean recall and precision values, i.e., 43 (features) x 19 domain experts' feature descriptions x 30 (repetitions) = 24510 independent runs.

With the obtained results, we record the mean values and standard deviations for precision, recall, and the F-measure. Also, we record a $p - value$, an $\hat{A}_{12}$ value, and a Cliff's delta value from the comparison of the results between CoFLiM_5_S and CoFLiM_0_N.

6.5.4 Answering RQ_3

To answer what the influence of the number of domain experts input to the CoFLiM approach is, we execute CoFLiM with a different value of k (from 1 to 18) to set a different number of relevant documents for the location of the 43 features. Hence, 43 (features to locate) x 18 values of k for CoFLiM x 30 (repetitions) = 23220 independent runs.

We record the mean F-measure value of locating the 43 features for each value of k. The best F-measure value allows us to determine whether k=5 obtains the best result as the literature recommends (i.e., 5 domain experts' feature descriptions are set as relevant documents to collaborate in the location of features), or whether a different value of k improves the quality of the solution.

6.5.5 Answering RQ_4.

To answer whether the inclusion of the domain experts' confidence produces an improvement, we first execute the 30 repetitions of CoFLiM for each of the 43 features by setting k with the value obtained as answer in the previous research question, and by selecting the base query and relevant documents according to their self-rated confidence level as described in Section 4.2.

We also execute the 30 repetitions of CoFLiM for each of the 43 features by also setting k with the value obtained as answer in the previous research question but selecting the base query and relevant documents randomly (i.e., without using the self-rated confidence level).

With the obtained results, we record the mean values and standard deviations for precision, recall, and the F-measure for the 1290 runs (43 features x 30 repetitions) of CoFLiM using the self-rated confidence and the 1290 runs of CoFLiM with the random confidence. The set of results were also compared though statistical analysis recording a $p - value$, an $\hat{A}_{12}$ value, and a Cliff's delta value.

7 Results and Discussion

In this section, we present the results obtained to answer each of the four research questions as well as the discussion of the results.

7.1 Research Question 1

Fig. 10 shows four charts with the recall and precision results of executing each baseline in the industrial domain. The bottom part of Fig. 11 shows the chart with the recall and precision results of executing CoFLiM_5_S in the industrial domain. Each chart shows the recall and precision results of locating each of the 43 target features. A dot in a chart represents the mean result of precision and recall of executing the 30 repetitions of a target feature.

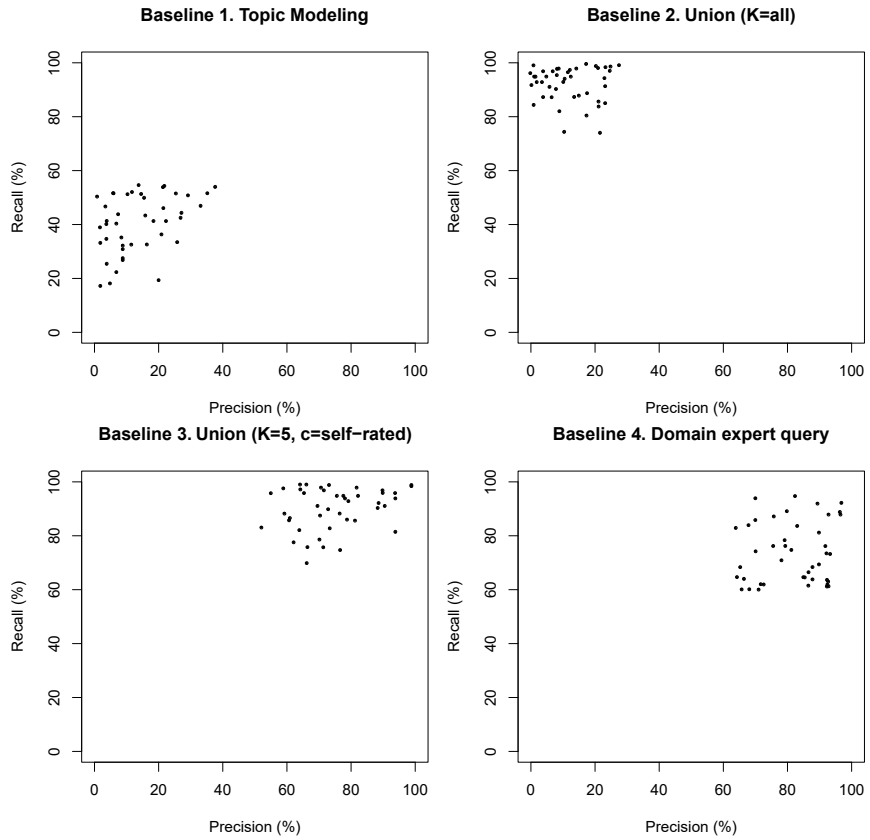


Fig. 10 Mean Recall and Precision values for each baseline

Table 2 shows the mean values of recall, precision, and the F-measure of the charts of the baselines and CoFLiM_5_S. In terms of precision and the F-measure, CoFLiM_5_S outperforms the four baselines since CoFLiM_5_S obtained the best result, providing an average value of 91.29% and 89.78%. 89.12%, respectively. In terms of recall, the Baseline 2 obtains the highest value 92.25%, which slightly outperforms the Baseline 3 and CoFLiM_5_S (90.16% and 89.12%, respectively).

Table 2 Mean values and standard deviations for Precision, Recall, and the F-measure for the baselines and CoFLiM_5_S in the industrial case study

	Recall ± (σ)	Precision ± (σ)	F-measure ± (σ)
Baseline 1: Topic modeling	40.71 ± 10.83	14.18 ± 10.05	19.45 ± 11.57
Baseline 2: Union (K=all)	92.25 ± 6.55	12.25 ± 8.26	20.61 ± 12.87
Baseline 3: Union (K=5, c=self-rated)	90.16 ± 7.88	75.10 ± 12.19	81.42 ± 8.75
Baseline 4: Domain expert query	74.01 ± 11.27	81.84 ± 10.49	77.06 ± 8.20
CoFLiM_5_S	89.12 ± 8.87	91.29 ± 7.11	89.78 ± 5.56

Table 3 shows the results of the statistical analysis between CoFLiM_5_S and each baseline, which comprise the comparison of the mean result of recall, precision and the F-measure for each of the 43 feature descriptions. Specifically, the second, third and fourth column of Table 3 shows the $p - Values$ of Holm's post hoc analysis for recall, precision and the F-measure. In recall, two of four $p - Values$ are smaller than the corresponding significance threshold value (0.05). Specifically, the $p - Values$ that comprise the comparison of CoFLiM_5_S with Baseline 1 and Baseline 4. In precision and F-measure, all $p - Values$ are smaller than the corresponding significance threshold value (0.05). Hence, there are significant differences in recall between CoFLiM_5_S and Baseline 1 and between CoFLiM_5_S and Baseline 4, and there are significant differences in precision and F-measure between CoFLiM_5_S and all baselines.

Table 3 also shows the values of the effect size statistics ($\hat{A}_{12}$ and Cliff's Delta). In recall, the $\hat{A}_{12}$ value indicates that CoFLiM_5_S outperforms with a pronounced superiority Baseline 1 and Baseline 4 (100% of the runs CoFLiM_5_S would obtain better results than the Baseline 1, and 84.69% of the runs CoFLiM_5_S would obtain better results than Baseline 4). This superiority of CoFLiM_5_S is also evidenced by the Cliff's Delta values, which can be interpreted as large according to the magnitude scales [Romano et al., 2006] in Baseline 1 (with a Cliff's Delta value of 1) and in Baseline 4 (with a Cliff's Delta value of 0.69).

Table 3 Statistical analysis for comparing CoFLiM_5_S with each baseline

	Holm's post hoc			Effect size					
	$p - Value$			$\hat{A}_{12}$			Cliff's Delta		
	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure
CoFLiM_5_S vs Baseline 1	$1.5x10^{-14}$	$1.5x10^{-14}$	$1.5x10^{-14}$	1	1	1	1	1	1
CoFLiM_5_S vs Baseline 2	0.16	$1.5x10^{-14}$	$1.5x10^{-14}$	0.4064	1	1	-0.19	1	1
CoFLiM_5_S vs Baseline 3	0.34	$5.2x10^{-10}$	$2.8x10^{-9}$	0.4749	0.8626	0.7810	-0.05	0.73	0.56
CoFLiM_5_S vs Baseline 4	$1.5x10^{-8}$	0.00012	$2.5x10^{-10}$	0.8469	0.7753	0.8891	0.69	0.55	0.78

In precision and F-measure, the values of the effect size statistics of Table 3 show that CoFLiM_5_S outperforms with a pronounced superiority all baselines. The comparisons that entail the most pronounced superiority entail Baseline 1 and Baseline 2. This superiority of CoFLiM_5_S in all baselines for precision and F-measure is also evidenced by the Cliff's Delta values, which can be interpreted as large in all baselines.

RQ_1 answer. From the results, we conclude that the query reformulated through automatic query expansion produces an improvement in terms of solution quality compared to the alternatives of the baselines. Moreover, the results reveal that manual query reformulation does not obtain the best solution.

7.2 Research Question 2

The first and the second chart of Fig. 11 shows the recall and precision results for the industrial domain of the variants of CoFLiM that do not address collaboration of domain experts (k=0). A dot in the first chart represents the mean result of precision and recall of executing the 30 repetitions of CoFLiM_0_N for each of the 817 feature descriptions (43 feature descriptions for each of the 19 domain experts). The second chart is a view of the first chart that only shows the dot that has the highest confidence level for each of the 43 features (HighestConfidence(CoFLiM_0_N)). We consider it relevant to highlight these results since, in a solo scenario, those domains experts with the highest confidence level are supposed to achieve the best results.

Table 4 shows the mean values of recall, precision, and the F-measure of CoFLiM_0_N with the goal of comparing these results with CoFLiM_5_S. In terms of recall and precision, CoFLiM_5_S outperforms CoFLiM_0_N since CoFLiM_5_S obtained the best results in recall and precision, providing an average value of 89.12% in recall and 91.29% in precision. The second best results are obtained by HighestConfidence(CoFLiM_0_N) (65.82% in recall and 68.8% in precision), which slightly outperforms CoFLiM_0_N (62.65% in recall and 66.28% in precision).

Table 4 CoFLiM mean values and standard deviations for Precision, Recall, and the F-measure in the industrial case study

	Recall ± (σ)	Precision ± (σ)	F-measure ± (σ)
CoFLiM_0_N	62.65±12.84	66.28±15.94	62.85±10.43
HighestConfidence(CoFLiM_0_N)	65.82±14.99	68.80±13.86	66.06±11.11

Table 5 shows the results of the statistical analysis between CoFLiM_5_S and HighestConfidence(CoFLiM_0_N), which comprise the comparison of the mean result of recall, precision and the F-measure for each of the 43 feature descriptions. Specifically, the second column of Table 5 shows the $p - Values$ of Holm's post hoc analysis for recall, precision and the F-measure, which are smaller than the corresponding significance threshold value (0.05). Hence, there are significant differences in recall, precision and F-measure between CoFLiM_5_S and HighestConfidence(CoFLiM_0_N).

Table 5 Statistical analysis for CoFLiM_5_S vs. HighestConfidence(CoFLiM_0_N)

	Holm's post hoc	Effect size	
	$p - Value$	$\hat{A}_{12}$	Cliff's Delta
Recall	$2x10^{-11}$	0.9005	0.80
Precision	$6.6x10^{-11}$	0.9103	0.82
F-measure	$1.6x10^{-13}$	0.9665	0.93

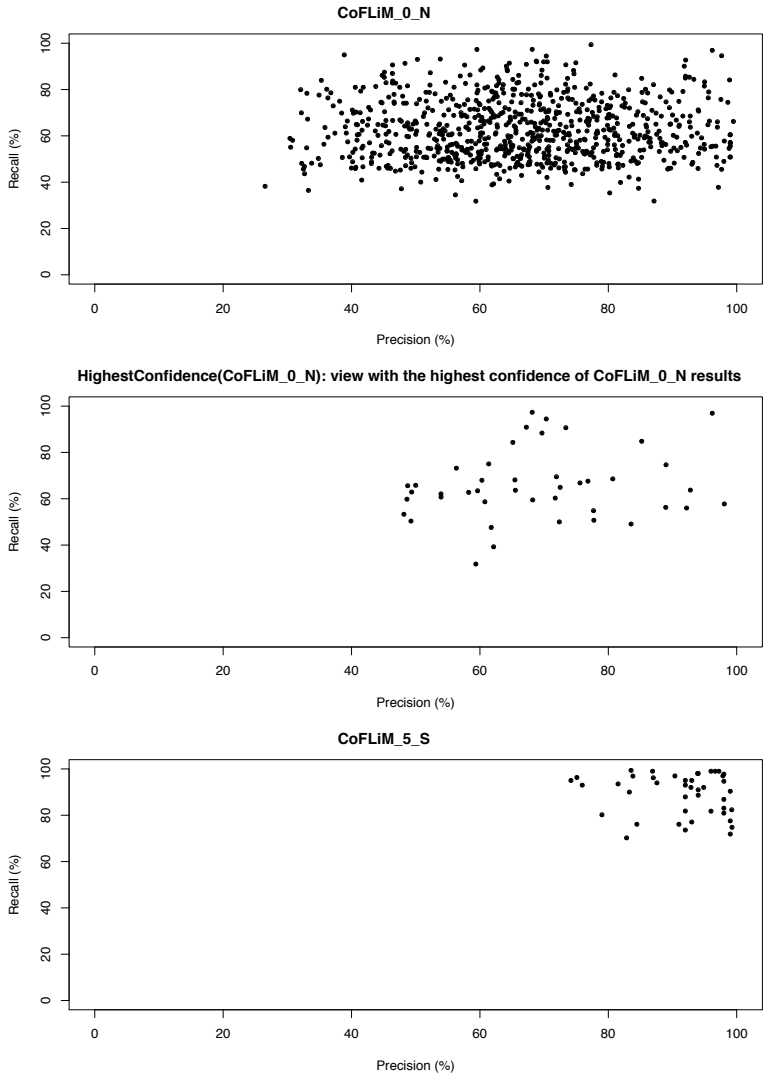


Fig. 11 Feature Location without Collaboration (top and center) vs. Feature Location with Collaboration (bottom)

The third and fourth column of Table 5 show the values of the effect size statistics. In the third column, the $\hat{A}_{12}$ value is 0.9005 for recall, 0.9103 for precision and 0.9665 for F-measure. This indicates that CoFLiM_5_S outperforms HighestConfidence(CoFLiM_0_N) with a pronounced superiority since CoFLiM_5_S would obtain better results for recall in 90.05% of the runs, for precision in 91.03% of the runs, and for F-measure in 96.65% of the runs. This superiority is also evidenced by the Cliff's Delta, with values of 0.80 for

recall, 0.82 for precision and 0.93 for F-measure. According to the magnitude scales [Romano et al., 2006], these values can be interpreted as large.

RQ_2 answer. From the results, we can conclude that the inclusion of more than one domain expert is feature descriptions when locating features in models produces an improvement in terms of solution quality. Furthermore, the results also reveal that the domain experts who have the highest self-rated confidence level do not have the required knowledge to locate all of the elements of the target feature. This lack of knowledge can be alleviated by collaborating with other domain experts who do not have the highest self-rated confidence but who can provide relevant information to locate the feature.

7.3 Research Question 3

Fig. 12 shows a chart in which each dot represents the value of k used to execute CoFLiM from 1 to 18 and the mean F-measure obtained as result of locating the 43 features. The chart also shows the standard deviation of each point using a blue shadow.

As the chart shows, the F-measure value increases until k=3, where it reaches the best value (90.84). The F-measure values are slightly lower from k=4 to k=6 (90.38, 89.78, 90.06, respectively). The values decrease down to 70.78 from k=7 to k=9, whereas the F-measure values get worse than the value obtained for k=1 from k=10 to k=18.

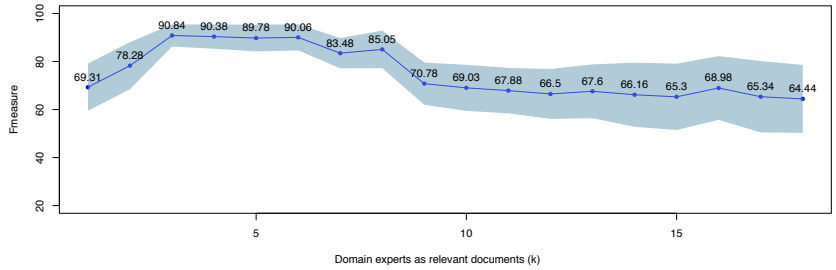


Fig. 12 Results of analyzing how the number of domain experts influences the solution

RQ_3 answer. From the results, we can conclude that the quality of the solution is positively influenced by the domain experts up to k=3, where it reaches the highest F-measure. This result suggests a reduction in the number of domain experts that are necessary for collaboration in accordance with the recommendation of the literature (k=5). This finding is useful in industrial contexts where domain experts are a scarce asset due to availability restrictions.

7.4 Research Question 4

Fig. 13 shows the charts with the recall and precision results of analyzing whether the inclusion of domain experts' confidence produces an improvement in terms of solution quality. A dot in the chart of the upper half of Fig. 13 represents the mean result of precision and recall of executing the 30 repetitions of CoFLiM_3_S for each of the 43 features where the base query and the relevant documents are set according to the domain experts' self-rated confidence level as described in Section 4.2. We set k=3 because this value reached the best F-measure value in the results of the previous research question. A dot in the chart of the lower half of Fig. 13 represents the mean result of precision and recall of executing the 30 repetitions of CoFLiM_3_R for each of the 43 features by randomly setting the base query and the relevant documents (i.e., without taking into account the domain experts' confidence).

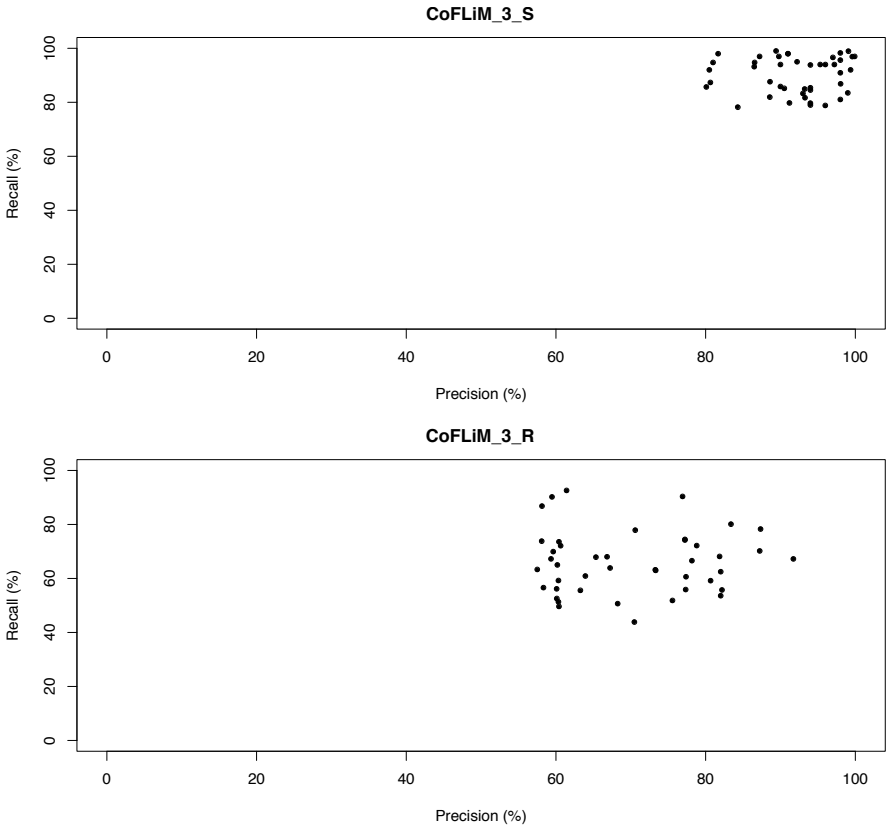


Fig. 13 Impact of domain experts' confidence level

Table 6 shows the mean values of recall, precision, and the F-measure of the graphs. In terms of recall and precision, CoFLiM_3_S yields the best results, obtaining an average value of 90.07% in recall and 92.06% in precision, whereas CoFLiM_3_R obtains an average value of 65.96% in recall and 70.09% in precision.

Table 6 Mean values and standard deviations for Precision, Recall, and the F-measure according to the domain experts' confidence level

	Recall ± (σ)	Precision ± (σ)	F-measure ± (σ)
CoFLiM k=3 self-rated confidence	90.07±6.76	92.06±5.76	90.84±4.62
CoFLiM k=3 random confidence	65.96±11.57	70.09±10.06	67.16±7.69

Table 7 shows the results of the statistical analysis between CoFLiM_3_S and CoFLiM_3_R. Specifically, the second column of Table 7 shows the $p - Values$ of Holm's post hoc analysis for recall, precision and the F-measure, which are smaller than the corresponding significance threshold value (0.05). Hence, there are significant differences in recall, precision and F-measure. The third column of Table 7 shows the values of the effect size statistics. The $\hat{A}_{12}$ value is 0.9557 for recall, 0.9648 for precision and 0.9978 for F-measure. This indicates that CoFLiM_3_S outperforms CoFLiM_3_R for recall in 95.57% of the runs, for precision in 96.48% of the runs, and for F-measure in 99.78% of the runs. This superiority is also evidenced by the Cliff's Delta, with values of 0.91 for recall, 0.93 for precision and 0.99 for F-measure, which can be interpreted as large according to the magnitude scales [Romano et al., 2006].

Table 7 Statistical analysis for CoFLiM k=3 with self-rated confidence vs. CoFLiM k=3 with random confidence

	Holm's post hoc	Effect size	
	$p - Value$	$\hat{A}_{12}$	Cliff's Delta
Recall	$6.7x10^{-13}$	0.9557	0.91
Precision	$1.5x10^{-14}$	0.9648	0.93
F-measure	$1.5x10^{-14}$	0.9978	0.99

RQ_4 answer. From the results, we conclude that the domain experts' self-rated confidence level produces an improvement in terms of solution quality to perform collaborative feature location in models. Automatic query expansion techniques can only work if the base query is reasonably strong enough to retrieve at least some of the relevant results [Sisman and Kak, 2013]. The results suggest that we can rely on the self-rated confidence level to determine the base query and the relevant documents.

7.5 Discussion

In the answers of the four research questions, it is shown that collaboration through automatic query reformulation pays off in the location of features in models. Although we explored four alternatives as baselines to obtain the query that will be used for locating a target feature, they did not outperform the automatic query reformulation of our approach. We found the following issues inherent to the baselines:

- Baseline 1 (Topic modeling) did not confirm the good results that the explored application of topic modeling [Zeng et al., 2012] obtained. In the previous work, they used long documents (clinical documents) to apply topic modeling. In contrast, our work identified the topics using the feature descriptions, which are small (in comparison to clinical documents) and do not serve to identify topics that improve the quality of the solution.
- Baseline 2 (Union, K=all) obtained the worst results in precision among the baselines. Although this baseline was a simple technique, the fact of merging the terms of all feature descriptions made that the model fragment that was retrieved included many model elements (which improved recall results) but many of these model elements were not included in the solution (which worsened precision results).
- Baseline 3 (Union, K=5, c=self-rated) obtained higher results in recall but it obtained worse results in precision compared to our approach. Similar to our approach, this baseline limits the number of feature descriptions that are used for merging terms but it does not limit the number of terms that are included in the reformulation as our approach does. Hence, more terms that were not relevant to the target feature were added and therefore, the precision was worse.
- Baseline 4 (Domain expert query) obtained worse precision and recall results than our approach. An issue that we found to explain these results is that the domain expert who manually combined feature descriptions cannot choose the full set of terms that describe the target feature. This was because the domain expert did not have the updated knowledge of the target feature since features are evolved over time by different domain experts. Hence, the domain expert query included terms that were not included in the last version of the feature, or it did not include terms from other queries since the domain expert thought that these terms did not belong to the target feature.

Our results also revealed that the domain experts who have the highest self-rated confidence level do not have the required knowledge to locate all of the elements of the target feature by themselves. An issue that we found to explain this result is similar to the issue that we found in Baseline 4. In this case, domain experts set as a high self confidence level if they have participated in the development or maintenance of the target feature. However, the target feature was evolved after by different domain experts, so the terms that were included in the description were either not complete or different to the

terms used in the last version of the feature. In addition, other domain experts that modified part of a feature after set a lower confidence level because they thought that their knowledge was not enough to locate the whole feature. Nevertheless, the knowledge of these experts, who performed the most recent modifications but do not have high confidence, was important to locate the target feature.

In order to improve the quality of the solutions, new experiments need to be performed. It is needed to further study whether domain experts should provide additional information for each feature description. The results shown that the self-rated confidence level that domain experts provided contributed to improve the quality of the solution. However, additional information can be used to select a different base query, which can influence the quality of the solutions. For example, domain experts can also provide information about the time that has elapsed since their last modification. According to the issues that we have found, it is relevant having knowledge of the most recent modifications for certain information retrieval purposes. Specifically, our results suggest that it is important for feature location having knowledge about when a domain expert has modified a feature. How this information should be collected as well as new ways of query reformulation or even using a multi-objective version of the algorithm constitute our future work. This future work comprises the application of both the lessons learned about the self-rated confidence and exploring the temporal relationship with the feature.

According to our industrial partner, maintenance contracts are extended for long periods of time. Specifically, the typical maintenance contracts of our industrial partner last 25 years and they include software modifications. In this context, it is relevant the approach for feature location that is presented in this paper since the domain experts may not be available during the 25 years of software maintenance. This is because domain experts left the company (due to retirement or due to the pursue of other job opportunities), or they have been assigned to another project. Therefore, our industrial partner faces the challenge of assembling tailored teams for maintenance tasks.

The challenge of assembling tailored teams for maintenance tasks is not exclusive to our industrial partner. Therefore, tailored teams may need to be created to perform the software maintenance tasks in other organizations. To tailor the teams, decisions need to be taken related to how many domain experts are going to collaborate and whether the self-rated confidence about the features is important. The answers of the research questions that have been presented in this paper help to take these decisions. To conclude, our results in collaborative feature location in models through automatic query expansion are encouraging, so we think we should further research this field.

8 Threats to validity

In this section, we present some of the threats to the validity of our work. We follow the guidelines suggested by De Oliveira et. al [de Oliveira Barros and Neto, 2011] to identify those threats that apply to this work.

Conclusion validity threats: The first identified threat of this type is not accounting for random variation. To address this threat, we considered 30 independent runs for each feature description and execution. The second threat is the lack of statistical tests. In this paper, we employed standard statistical analysis following accepted guidelines [Arcuri and Fraser, 2013] to avoid this threat. The third threat is the lack of a good descriptive analysis. In this work, we have used the recall, precision, and F-measure measurements to analyze the confusion matrix obtained; however, other measurements could be applied. In addition, some works argue that the use of the Vargha and Delaney $\hat{A}_{12}$ measurement can be misrepresentative [Arcuri and Fraser, 2013] and that the data should be pre-transformed before applying it. We did not find any use cases for data pre-transformation that applied to our case study.

Internal validity threats: The first identified threat of this type are the poor parameter settings. In this paper, we used standard values for the algorithms. As suggested by Arcuri and Fraser [Arcuri and Fraser, 2013], default values are good enough to measure the performance of location techniques. These values have been tested in similar algorithms for Feature Location [Lopez-Herrejon et al., 2015]. With regard to the parameter of the time required to produce a solution (stop condition), we used 80 seconds since it was the time needed by our approach in the real-world industrial context. Nevertheless, we cannot yet claim how this time scales in other real-world industrial contexts with a larger size of the search space (models) or a larger size of the solution to be found (model fragment). In the future, we would like to reduce these threats by evaluating all of the parameters of our algorithm. With regard to the number of terms used to expand the base query, we used 10 as recommended in the literature [Carpineto and Romano, 2012]. However, at this stage, we do not know how using a different value would impact the results. The second threat is the lack of real problem instances. The evaluation of this paper was applied in a real-world industrial case study from the railway domain.

Construct validity threats: The identified threat is the lack of assessing the validity of cost measures. To address this threat, we performed a fair comparison among the executions by generating the same number of model fragments. Also, our evaluation is performed using three measures: precision, recall, and the F-measure. These measures are widely accepted in the software engineering research community [Salton and McGill, 1986].

External validity threats: The identified threat of this type is the lack of a clear object selection strategy. This threat is addressed by using an industrial case study. Our instances are collected from real-world problems.

With regard to what extent it is possible to generalize the results, we selected 43 different features to locate and 19 domain experts were involved.

Even though the number of domain experts (19) might not seem large enough to generalize results, it is important to note that the involvement of domain experts in an industrial environment makes an interesting contribution in an area where most experiments are conducted with students or artificial problems.

Our approach has been designed to be applied in the domain of our industrial partner as well as in other different domains. The requisites to apply our approach are that the set of models where features have to be located conform to MOF (the OMG metalanguage for defining modeling languages) and the feature description must be provided as a textual description. Furthermore, the fitness function can also be applied to any MOF-based model. The text elements associated to the models are extracted automatically by the approach using the reflective methods provided by the Eclipse Modeling Framework.

As occurs in other works [Sisman and Kak, 2013; Hill et al., 2009], results depend on the quality of the queries. Poor queries assign a high rank to irrelevant model fragments. It is also worth noting that the language used for the textual elements of the models and the feature descriptions in the query provided must be the same. This language is specific to each domain; however, as long as both elements are built using the same terminology, the LSA will work. Eventually, some tweaks can be applied to narrow the gap between both elements (different tokenizers, stemming, or POS tagging techniques). For instance, the naming conventions used by companies for model elements, properties, and functions can follow different formats, but the approach can be tailored to handle them. In our case study, some model elements follow the CamelCase convention while others follow the Underscore convention. To address that, we applied different tokenizers in order to obtain the terms properly.

Hence, even though our approach can be applied to locate features in MOF-based models from a real-world industrial context, our approach should be applied to other domains before assuring its generalization.

9 Related work

Approaches related to the work presented in this paper can be divided into two areas: collaborative information retrieval and feature location.

9.1 Collaborative information retrieval

A good deal of research that addresses collaborative information retrieval has focused on reformulating the query of a user [Shah, 2010] based on relevant documents such as source code and Internet sites. Several approaches have been proposed to reformulate queries in a semi-automatic or automatic way by expanding or reducing the terms of the given query.

On the one hand, query expansion is based on the idea that the more query terms there are, the more documents are retrieved and ranked according to

the similarity to the query. Several approaches propose to automatically expand the query to add words that are either similar or related in some way to the query terms. For example, Yang and Tan [Yang and Tan, 2012] reformulate the query by extracting synonyms, antonyms, abbreviations, and related words from the source code. Rivas et al. [Rivas et al., 2014] add relevant terms from a scientific documental database to a query to improve the documents initially retrieved. Hill et al. [Hill et al., 2009] also obtain possible query expansion terms from the code. Lu et al. [Lu et al., 2015] improve code search by expanding the query with synonyms. Marcus et al. [Marcus et al., 2004] expand the query using LSI in order to determine the terms from the source code that are most similar to the query. Gay et al. [Gay et al., 2009] present a semi-automated approach for concept location in code. Their approach requires the intervention of a developer, which is based on using user relevance feedback to get the meaning of the query closer to that of relevant documents. Haiduc et al. [Haiduc et al., 2013] propose an approach that is trained with a sample of queries and relevant results in order to automatically reformulate the query to improve the performance.

Other approaches expand the query by adding information from external sources of information such as public repositories [Dumitru et al., 2011] or by adding semantically similar words from websites [Tian et al., 2014]. For example, Dietrich et al. [Dietrich et al., 2013] improved the efficacy of future queries using feedback captured from a validated set of queries and traceability links. Lv et al. [Lv et al., 2015] enrich each API with its online documentation to match the query based on text similarity.

Table 8 compares the above query expansion works with our work. As the table shows, the base query that is going to be expanded is obtained from a human, who can play different roles (developer, user, analyst, and domain expert). The relevant documents used to find the terms to expand the query are usually source code, online documentation, or text. In contrast, to support collaboration in our work, we use other domain experts' feature descriptions as relevant documents in order to enrich the base query feature description with the knowledge of other domain experts. Also, the works mentioned above use a higher k value (i.e., the number of relevant documents used to obtain the terms to expand the base query), which ranges between the general recommendation of 5 [Carpineto and Romano, 2012] and the Lines of Code. In contrast, our work uses a lower k value (3) to obtain the best result, which can be beneficial in promoting collaboration in industrial contexts where the availability of domain experts is scarce.

Also, none of the above works have been validated in an industrial domain. As Ambreen et al. [Ambreen et al., 2016] state, there is a need to conduct empirical studies in industry since the context is not the same as in academia. Thus, our work aims to cover the dearth of studies of query expansion techniques in industry for supporting collaborative feature location. Finally, the query expansion works mentioned above retrieve information from code or text artifacts, whereas our work locates features by using models.

Table 8 Comparison with query expansion works

Author	Base query	Relevant documents	k Value	Industrial domain	Artifact
[Yang and Tan, 2012]	Developer	Source code	LoC	No	Code
[Rivas et al., 2014]	User	Biomedical articles	10	No	Text
[Hill et al., 2009]	Developer	Source code	LoC	No	Code
[Lu et al., 2015]	Developer	Internet site	20	No	Code
[Marcus et al., 2004]	User	Source code	5	No	Code
[Gay et al., 2009]	Developer	Source code	1, 3, and 5	No	Code
[Haiduc et al., 2013]	User	Source code	5	No	Code
[Dumitru et al., 2011]	User	Internet sites	25	No	Product specifications
[Tian et al., 2014]	User	Internet site	10,000	No	Text and code
[Dietrich et al., 2013]	Analyst	Requirement traces	-	No	Code and documents
[Lv et al., 2015]	User	Online documentation	10	No	Code
Our work	Domain expert	Domain experts	3	Yes	Models

In our previous works [Pérez et al., 2017, 2018], we also performed query reformulation to locate features in models. In Pérez et al. [Pérez et al., 2017], we involved a fixed number of domain experts and two cores (IR and Linguistic rules) to locate the relevant model fragments, instead of using an evolutionary algorithm and a variable number of domains experts as this work does. Our previous results shown how the different cores affect the quality of the solution, which are worse than the results that this work presents using the evolutionary algorithm. Furthermore, in contrast to this work, in [Pérez et al., 2017] was not explored how the quality of the results was affected by: (1) different alternatives to reformulate the feature descriptions, (2) the number of domain experts, and (3) the inclusion of the domain experts' confidence.

In [Pérez et al., 2018], we also reformulated the base query with documents that are other train models instead of using feature descriptions produced by domain experts as this work does. In Model-Driven Development, models are the main software artifact, and these models were used as documents to reformulate the base query. The results after the reformulation were worse than without the reformulation. This was because the proposed terms to expand the base query were always terms from the metamodel. These metamodel terms encoded very generic domain knowledge and either they were already present in the base query, or they were not relevant at all for the target feature. In this paper, we also reformulate the base query but the documents for the

reformulation are feature descriptions produced by domain experts, and on this occasion the results are better than without the domain experts' collaboration.

On the other hand, query reduction is based on the idea that long queries contain both important information as well as words that do not contribute to the search goal. For example, Bendersky and Croft [Bendersky and Croft, 2008] propose an approach to reduce long queries by learning and identifying the key concepts. Kumaran and Allan [Kumaran and Allan, 2008] propose an interactive approach with users to completely drop unnecessary terms from long queries. Kumaran and Carvalho [Kumaran and Carvalho, 2009] reduce queries by analyzing the most promising subsets of terms from the original query.

In contrast to the above approaches for query reduction, our work aims to expand queries for supporting collaboration among domain experts. Our results show that collaboration improves precision. However, our results do not reach 100% in precision. Hence, query reduction techniques can be introduced as future work to improve the precision by reducing the words that do not contribute to the location of the feature.

9.2 Feature location

There are many feature location approaches that have been proposed to find features in code by taking textual information as input [Dit et al., 2013]. Cavalcanti et al. [Cavalcanti et al., 2014] used IR techniques to assign change requests in software maintenance or evolution tasks based on context information. Kimmig et al. [Kimmig et al., 2011] proposed an approach for translating NL queries to concrete parameters of the Eclipse JDT code query engine. Wang et al. [Wang et al., 2014] proposed a code search approach, which incorporates user feedback to refine the query. Zou et al.[Zou et al., 2015] investigated the "answer style" of software questions with different interrogatives and proposed a re-ranking approach to refine search results.

Even though these approaches improve the effectiveness of feature location, they require an additional effort to enrich the code, to keep the documentation synchronized with the changes in the code throughout maintenance and evolution activities, and to incorporate users' feedback. In contrast, our work does not require additional efforts to enrich the models (in our work, we locate features in models instead of code). Our work leverages the collaboration among domain experts to improve the quality of the solution.

Some works [Wille et al., 2013; Holthusen et al., 2014; Zhang et al., 2011, 2012; Martinez et al., 2015a; Font et al., 2015b; Martinez et al., 2015b] focus on the location of features in models by comparing the models with each other to formalize the variability among them in the form of features of a Software Product Line (SPL) [Clements and Northrop, 2001]:

- Wille et al. [Wille et al., 2013] present an approach where the similarity between models is measured following an exchangeable metric, taking into account different attributes of the models. Then, the approach is further

refined [Holthusen et al., 2014] to reduce the number of comparisons needed to mine the family model.
- The authors in [Zhang et al., 2011] propose a generic approach to automatically compare products and locate the feature realizations in terms of a CVL model. In [Zhang et al., 2012], the approach is refined to automatically formalize the feature realizations of new product models that are added to the system. A similar approach is proposed in [Font et al., 2015b], where the feature location results are validated against an industrial environment.
- Martinez et al. [Martinez et al., 2015a] propose an extensible approach that is based on comparisons to extract the feature formalization in a family of models. In addition, they provide means to extend the approach to locate features in any kind of asset based on comparisons.
- The MoVaPL approach [Martinez et al., 2015b] considers the identification of variability and commonality in model variants as well as the extraction of a Model-based Software Product Line (MSPL) from the features identified on these variants. MoVaPL builds on a generic representation of models, making it suitable for any MOF-based models.

Nevertheless, all of these approaches are based on mechanical comparisons among the models, classifying the elements based on their similarity and identifying the dissimilar elements as the features realizations. In contrast, our work does not rely on model comparisons to locate the features. Specifically, in our work, humans are involved in the search. Domain experts become part of the process by contributing their knowledge of the domain in order to tailor the approach to the feature description. Model fragments obtained mechanically are less recognizable by software engineers than those obtained with the participation of domain experts [Font et al., 2015a].

Search-based Software Engineering (SBSE) [Harman, 2010] uses search-based optimization techniques (mainly those from the evolutionary computation literature) to automate the search for optimal or near-optimal solutions to software engineering problems. Our work also uses evolutionary computation to automate the search of solutions to the feature location problem. Harman et al. [Harman et al., 2014] performed a survey on the topic of SBSE applied to SPLs. They present an overview of recent articles that are classified according to themes such as configuration, testing, or architectural improvement. Lopez-Herrejon et al. [Lopez-Herrejon et al., 2014] performed a preliminary systematic mapping study at the connection of SBSE and SPL. These two surveys indicate that SBSE is being applied to SPLs. However, these surveys do not identify works that are focused on finding model fragments that materialize features as our work does.

Font et al. [Font et al., 2016a] use an evolutive algorithm to locate features among a family of models in the form of a variation point. Their approach is refined in [Font et al., 2016b], where the authors use the evolutive algorithm to find sets of suitable feature realizations. The authors first cluster model fragments based on their common attributes into feature realization candidates through Formal Concept Analysis. Then, Latent Semantic Indexing ranks the

candidates based on the similarity with the feature description. In contrast our approach, locates model fragments instead of variation points. Font et al. [Font et al., 2017] rely on an Evolutionary Algorithm to automate feature location in model fragments by comparing the text of each model fragment with the search query. Our approach also differs from [Font et al., 2016a], [Font et al., 2016b] and [Font et al., 2017] regarding collaboration. In our approach, automatic query reformulation enables domain experts' collaboration to locate the model fragment that materializes a target feature. Moreover, we also analyze the impact that the number of domain experts fed to the feature location and the inclusion of domain experts' confidence have on the results, which is something that [Font et al., 2016a], [Font et al., 2016b] and [Font et al., 2017] do not tackle.

10 Concluding remarks

Because people are searching together on a regular basis, there is a need to support collaborative feature location during maintenance or evolution of software. Supporting collaborative feature location is especially necessary both in industrial contexts where complex projects require the collaboration of different individuals and in software artifacts such as the models that have been neglected to date.

To address this drawback, we propose CoFLiM, which is an approach to achieve collaborative feature location in models. CoFLiM automatically reformulates the domain expert's feature description that has the highest confidence level to expand it with terms of other relevant domain experts' feature descriptions. The resulting query is used to guide the evolutionary algorithm of our CoFLiM approach to find the model fragment that realizes the feature being located.

We have evaluated our approach in a real-world case study of our industrial partner, who is a worldwide leader in train manufacturing. We also compare the impact of the automatic query reformulation that CoFLiM performs with four alternatives as baselines to study the impact on the results. Moreover, we had 19 domain experts participate in order to analyze how the number of domain experts collaborating influences the quality of the solution. We also analyzed whether or not the inclusion of the domain experts' confidence improves the solution. The results show that CoFLiM improved the results of locating features without collaboration by 27.42% in recall and 25.78% in precision, and also that taking into account the domain experts' confidence obtained an average improvement of 24.11% in recall and 21.97% in precision. Also, the statistical analysis show that the results of CoFLiM had a significant impact.

Furthermore, the results reveal four findings that are relevant for collaborative feature location approaches in models:

– Collaborating improves the results obtained by domain experts independently of each other.

- Even the domain experts that have the highest confidence can benefit from collaborating with other domain experts that have lower confidence.
- Results are not improved when more than three domain experts collaborate.
- The self-rated confidence value is reliable for collaborative feature location.

We hope that these results encourage other researchers to introduce collaboration into their feature location approaches, especially in long-living industrial domains where the location of features is an endeavor that can benefit from the collaboration of domain experts.

Acknowledgements

This work has been partially supported by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the project Model-Driven Variability Extraction for Software Product Line Adoption (TIN2015-64397-R).

References

Efficient java matrix library. `http://ejml.org/`, 2016.

Apache opennlp: Toolkit for the processing of natural language text. `https://opennlp.apache.org/`, 2016.

English (porter2) stemming algorithm. `http://snowball.tartarus.org/algorithms/english/stemmer.htm`, 2017.

Talat Ambreen, Naveed Ikram, Muhammad Usman, and Mahmood Niazi. Empirical research in requirements engineering: trends and opportunities. *Requirements Engineering*, pages 1–33, 2016. ISSN 1432-010X. doi: 10.1007/s00766-016-0258-2. URL `http://dx.doi.org/10.1007/s00766-016-0258-2`.

Andrea Arcuri and Lionel Briand. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Softw. Test. Verif. Reliab.*, 24(3):219–250, May 2014. ISSN 0960-0833. doi: 10.1002/stvr.1486. URL `http://dx.doi.org/10.1002/stvr.1486`.

Andrea Arcuri and Gordon Fraser. Parameter Tuning or Default Values? An Empirical Investigation in Search-Based Software Engineering. *Empirical Software Engineering*, 18(3), 2013.

Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2):99–130, Jun 1996. ISSN 1573-7675. doi: 10.1007/BF00122124. URL `https://doi.org/10.1007/BF00122124`.

Hazeline U Asuncion, Arthur U Asuncion, and Richard N Taylor. Software traceability with topic modeling. In *Proceedings of the 32nd ACM/IEEE international conference on Software Engineering-Volume 1*, pages 95–104. ACM, 2010.

Michael Bendersky and W. Bruce Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 491–498, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4. doi: 10.1145/1390334.1390419. URL http://doi.acm.org/10.1145/1390334.1390419.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Jordan Boyd-Graber, Yuening Hu, and David Mimno. Applications of topic models. *Foundations and Trends®in Information Retrieval*, 11(2-3):143–296, 2017. ISSN 1554-0669. doi: 10.1561/1500000030. URL http://dx.doi.org/10.1561/1500000030.

Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50, January 2012. ISSN 0360-0300. doi: 10.1145/2071389.2071390. URL http://doi.acm.org/10.1145/2071389.2071390.

Yguaratã Cerqueira Cavalcanti, Ivan do Carmo Machado, Paulo A. da Mota S. Neto, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. Combining rule-based and information retrieval techniques to assign software change requests. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, pages 325–330, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3013-8. doi: 10.1145/2642937.2642964. URL http://doi.acm.org/10.1145/2642937.2642964.

Paul C. Clements and Linda Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. Addison-Wesley, 2001.

Norman Cliff. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin*, 114(3):494, 1993.

Norman Cliff. Ordinal methods for behavioral data analysis. 1996.

Márcio de Oliveira Barros and Arilo Claudio Dias Neto. Threats to validity in search-based software engineering empirical studies. Technical Report 0006/2011, 2011.

Timothy Dietrich, Jane Cleland-Huang, and Yonghee Shin. Learning effective query transformations for enhanced requirements trace retrieval. In *2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE)*, pages 586–591, Nov 2013. doi: 10.1109/ASE.2013.6693117.

Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. Feature location in source code: a taxonomy and survey. *Journal of Software: Evolution and Process*, 25(1):53–95, 2013. URL http://dblp.uni-trier.de/db/journals/smr/smr25.html#DitRGP13.

Horatiu Dumitru, Marek Gibiec, Negar Hariri, Jane Cleland-Huang, Bamshad Mobasher, Carlos Castro-Herrera, and Mehdi Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 181–190, 2011. ISBN 978-1-4503-0445-0. doi: 10.1145/1985793.1985819. URL http://doi.acm.org/10.1145/1985793.1985819.

Daniel Dyer. The watchmaker framework for evolutionary computation (evolutionary/genetic algorithms for java). http://watchmaker.uncommons.org/, 2016.

Jaime Font, Lorena Arcega, Øystein Haugen, and Carlos Cetina. Building software product lines from conceptualized model patterns. In *Proceedings of the 19th International Conference on Software Product Line (SPLC)*, pages 46–55, 2015a. doi: 10.1145/2791060.2791085.

Jaime Font, Manuel Ballarín, Øystein Haugen, and Carlos Cetina. Automating the variability formalization of a model family by means of common variability language. In *Proceedings of the 19th International Conference on Software Product Line (SPLC)*, pages 411–418, 2015b. doi: 10.1145/2791060.2793678.

Jaime Font, Lorena Arcega, Øystein Haugen, and Carlos Cetina. Feature Location in Model-Based Software Product Lines Through a Genetic Algorithm. In *Proceedings of the 15th International Conference on Software Reuse: Bridging with Social-Awareness*, 2016a.

Jaime Font, Lorena Arcega, Øystein Haugen, and Carlos Cetina. Feature Location in Models Through a Genetic Algorithm Driven by Information Retrieval Techniques. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, 2016b.

Jaime Font, Lorena Arcega, Øystein Haugen, and Carlos Cetina. Achieving feature location in families of models through the use of search-based software engineering. *IEEE Transactions on Evolutionary Computation*, PP (99):1–1, 2017. ISSN 1089-778X. doi: 10.1109/TEVC.2017.2751100.

Gregory Gay, Sonia Haiduc, Andrian Marcus, and Tim Menzies. On the use of relevance feedback in ir-based concept location. In *ICSM*, pages 351–360. IEEE Computer Society, 2009. ISBN 978-1-4244-4897-5. URL http://dblp.uni-trier.de/db/conf/icsm/icsm2009.html#GayHMM09.

Robert J. Grissom and John J. Kim. *Effect sizes for research: A broad practical approach*. Mahwah, NJ: Earlbaum, 2005.

Sonia Haiduc, Gabriele Bavota, Andrian Marcus, Rocco Oliveto, Andrea De Lucia, and Tim Menzies. Automatic query reformulations for text retrieval in software engineering. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 842–851, Piscataway, NJ, USA, 2013. IEEE Press. ISBN 978-1-4673-3076-3.

Mark Harman. Why the virtual nature of software makes it ideal for search based optimization. In *Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering*, FASE'10, pages 1–12, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-12028-8, 978-3-642-12028-2.

Mark Harman, Yue Jia, Jens Krinke, William B. Langdon, Justyna Petke, and Yuanyuan Zhang. Search based software engineering for software product line engineering: A survey and directions for future work. In *Proceedings of the 18th International Software Product Line Conference - Volume 1*, SPLC '14, pages 5–18, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2740-4. doi: 10.1145/2648511.2648513. URL http://doi.acm.org/10.1145/

2648511.2648513.

Øystein Haugen, Birger Møller-Pedersen, Jon Oldevik, Gøran K. Olsen, and Andreas Svendsen. Adding standardized variability to domain specific languages. In *Software Product Line Conference, 2008. SPLC '08. 12th International*, pages 139–148, Sept 2008. doi: 10.1109/SPLC.2008.25.

Emily Hill, Lori Pollock, and K. Vijay-Shanker. Automatically capturing source code context of nl-queries for software maintenance and reuse. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 232–242, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-3453-4. doi: 10.1109/ICSE.2009.5070524. URL http://dx.doi.org/10.1109/ICSE.2009.5070524.

Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, 1999.

Sönke Holthusen, David Wille, Christoph Legat, Simon Beddig, Ina Schaefer, and Birgit Vogel-Heuser. Family model mining for function block diagrams in automation software. In *Proceedings of the 18th International Software Product Line Conference: Volume 2*, pages 36–43, 2014. ISBN 978-1-4503-2739-8. doi: 10.1145/2647908.2655965.

Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223, 2003.

Markus Kimmig, Martin Monperrus, and Mira Mezini. Querying source code with natural language. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, ASE '11, pages 376–379, 2011. ISBN 978-1-4577-1638-6. doi: 10.1109/ASE.2011.6100076. URL http://dx.doi.org/10.1109/ASE.2011.6100076.

Anton Kotelyanskii and Gregory M. Kapfhammer. Parameter tuning for search-based test-data generation revisited: Support for previous results. In *2014 14th International Conference on Quality Software*, pages 79–84, Oct 2014. doi: 10.1109/QSIC.2014.43.

Giridhar Kumaran and James Allan. Effective and efficient user interaction for long queries. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4. doi: http://doi.acm.org/10.1145/1390334.1390339. URL http://portal.acm.org/citation.cfm?id=1390339.

Giridhar Kumaran and Vitor R. Carvalho. Reducing long queries using query quality predictors. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 564–571, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1572038. URL http://doi.acm.org/10.1145/1571941.1572038.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. An Introduction to Latent Semantic Analysis. *Discourse processes*, 25, 1998.

Raúl Lapeña, Francisca Pérez, and Carlos Cetina. On the influence of models-to-natural-language transformation in traceability link recovery among requirements and conceptual models. In *ER FORUM 2017*, 2017.

Roberto E. Lopez-Herrejon, Javier Ferrer, Francisco Chicano, Lukas Linsbauer, Alexander Egyed, and Enrique Alba. A hitchhiker's guide to search-based software engineering for software product lines. *CoRR*, abs/1406.2823, 2014. URL `http://arxiv.org/abs/1406.2823`.

Roberto E. Lopez-Herrejon, Lukas Linsbauer, José A. Galindo, José A. Parejo, David Benavides, Sergio Segura, and Alexander Egyed. An assessment of search-based techniques for reverse engineering feature models. *J. Syst. Softw.*, 103(C):353–369, May 2015. ISSN 0164-1212.

Meili Lu, X. Sun, S. Wang, D. Lo, and Yucong Duan. Query expansion via wordnet for effective code search. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 545–549, March 2015. doi: 10.1109/SANER.2015.7081874.

Fei Lv, Hongyu Zhang, Jian-Guang Lou, Shaowei Wang, Dongmei Zhang, and Jianjun Zhao. Codehow: Effective code search based on API understanding and extended boolean model. In *Automated Software Engineering (ASE2015)*, 2015.

Andrian Marcus, Andrey Sergeyev, Vaclav Rajlich, and Jonathan I. Maletic. An information retrieval approach to concept location in source code. In *Proceedings of the 11th Working Conference on Reverse Engineering*, WCRE '04, pages 214–223, Washington, DC, USA, 2004. ISBN 0-7695-2243-2. URL `http://dl.acm.org/citation.cfm?id=1038267.1039053`.

Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Bottom-up adoption of software product lines: a generic and extensible approach. In *Proceedings of the 19th International Conference on Software Product Line (SPLC)*, pages 101–110, 2015a. doi: 10.1145/2791060.2791086.

Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Automating the extraction of model-based software product lines from model variants (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 396–406, Nov 2015b. doi: 10.1109/ASE.2015.44.

Meredith R. Morris. Interfaces for Collaborative Exploratory Web Search: Motivations and Directions for Multi-User Designs. In *CHI 2007 Workshop on Exploratory Search and HCI*, 2007.

Francisca Pérez, Ana Cristina Marcén, Raúl Lapeña, and Carlos Cetina. Introducing collaboration for locating features in models: Approach and industrial evaluation. In *Proceedings of the 25th International Conference on Cooperative Information Systems, CoopIS*, pages 114–131, 2017. doi: 10.1007/978-3-319-69462-7_9.

Francisca Pérez, Jaime Font, Lorena Arcega, and Carlos Cetina. Automatic query reformulations for feature location in a model-based family of software products. *Data & Knowledge Engineering*, 2018. ISSN 0169-023X. doi: https://doi.org/10.1016/j.datak.2018.06.001.

Andreia R. Rivas, E.L. Iglesias, and L. Borrajo. Study of query expansion techniques and their application in the biomedical information retrieval. *The Scientific World Journal*, 2014.

Jeanine Romano, Jeffrey D Kromrey, Jesse Coraggio, and Jeff Skowronek. Appropriate statistics for ordinal level data: Should we really be using t-test and cohensd for evaluating group differences on the nsse and other surveys. In *annual meeting of the Florida Association of Institutional Research*, pages 1–33, 2006.

Julia Rubin and Marsha Chechik. A survey of feature location techniques. In *Domain Engineering*, pages 29–58. Springer, 2013.

Gerard Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., 1971.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986. ISBN 0070544840.

Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. Scalable product line configuration: A straw to break the camel's back. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 465–474, Nov 2013. doi: 10.1109/ASE.2013.6693104.

Chirag Shah. Collaborative information seeking: A literature review. *Exploring The Digital Frontier Advances In Librarianship*, 32, 2010. ISSN 0065-2830.

Bunyamin Sisman and Avinash C. Kak. Assisting code search with automatic query reformulation for bug localization. In *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, San Francisco, CA, USA, May 18-19, 2013*, pages 309–318, 2013. doi: 10.1109/MSR.2013.6624044.

Yuan Tian, David Lo, and Julia Lawall. Automated construction of a software-specific word similarity database. In *IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE)*, pages 44–53, 2014. doi: 10.1109/CSMR-WCRE.2014.6747213.

András Vargha and Harold D. Delaney. A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000. doi: 10.3102/10769986025002101.

Shaowei Wang, David Lo, and Lingxiao Jiang. Active code search: Incorporating user feedback to improve code search relevance. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, pages 677–682, 2014. ISBN 978-1-4503-3013-8. doi: 10.1145/2642937.2642947. URL `http://doi.acm.org/10.1145/2642937.2642947`.

David Wille, Sönke Holthusen, Sandro Schulze, and Ina Schaefer. Interface variability in family model mining. In *Proceedings of the 17th International Software Product Line Conference: Co-located Workshops*, pages 44–51, 2013. ISBN 978-1-4503-2325-3. doi: 10.1145/2499777.2500708.

Jinqiu Yang and Lin Tan. Inferring semantically related words from software context. In *Mining Software Repositories (MSR)*, pages 161–170, 2012. doi: 10.1109/MSR.2012.6224276.

Qing T Zeng, Doug Redd, Thomas Rindflesch, and Jonathan Nebeker. Synonym, topic model and predicate-based query expansion for retrieving clinical documents. In *AMIA Annual Symposium Proceedings*, volume 2012, page 1050. American Medical Informatics Association, 2012.

Xiaorui Zhang, Øystein Haugen, and Birger Moller-Pedersen. Model comparison to synthesize a model-driven software product line. In *Proceedings of the 2011 15th International Software Product Line Conference (SPLC)*, pages 90–99, 2011. ISBN 978-0-7695-4487-8. doi: 10.1109/SPLC.2011.24.

Xiaorui Zhang, Ø. Haugen, and B. Møller-Pedersen. Augmenting product lines. In *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific*, volume 1, pages 766–771, Dec 2012. doi: 10.1109/APSEC.2012.76.

Yanzhen Zou, Ting Ye, Yangyang Lu, John Mylopoulos, and Lu Zhang. Learning to rank for question-oriented software text retrieval. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)*, pages 1–11, 2015. ISBN 978-1-5090-0025-8. URL `http://dblp.uni-trier.de/db/conf/kbse/ase2015.html#ZouYLM015`.